A Bayesian Tensor Factorization Algorithm to Predict Drug Response in Cancer Cell Lines

Nathan H Lazar
Bachelor of Science in Mathematics, Ohio University, 2002
Masters of Science in Mathematics, Portland State University, 2007

Presented to the
Bioinformatics & Computational Biology
within the Oregon Health & Science University
School of Medicine
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Bioinformatics & Computational Biology

October 2017

Bioinformatics & Computational Biology
School of Medicine
Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the Ph.D. dissertation of
Nathan H Lazar
has been approved.

Dr. Kemal Sönmez, Thesis Advisor
Associate Professor

Dr. Mehmet Gönen
Assistant Professor, Koç University, Turkey

Dr. Shannon McWeeney
Professor

Dr. Lucia Carbone
Assistant Professor

Dr, Laura Heiser
Assistant Professor

Dedication and acknowledgements

Dedicated to my patient and loving wife Angela and
my supportive family and friends.

I would like to thank my committee including my advisor Dr. Kemal Sönmez for encouraging me to embark on a new career path, my co-advisor Dr. Lucia Carbone for being a constant source of advice and support, Dr. Mehmet Gönen for inspiring the approaches used here, Dr. Shannon McWeeney for pushing me to do my best work and Dr. Laura Heiser for help in obtaining data and insight into the problem space. Also, many thanks to my fellow students in the Department of Medical Informatics and Clinical Epidemiology and the very capable staff (Diane, Andrea, Lynne, Virginia and others). Finally, a huge debt of gratitude to the National Library of Medicine for the training grant that funded this research (T15LM007088).

Contents

Introduction

Cancer is now the biggest single cause of mortality worldwide, incidence has increased by 11% in four years and cases are forecast to rise by 75% over the next 20 years [1]. Despite the huge amount of resources devoted to combating this terrible disease, progress has been slow. Most patients are still treated with painful blunt instruments like radiation and chemotherapy while new targeted treatments addressing specific genetic causes often only help a small portion of the population. Why haven't scientists and doctors been able to do more? Because each patient is different, each cancer is unique and we lack the framework for collecting, processing and understanding the mountain of information that we can generate on cancers, treatments and responses.

Since the discovery of the first proto-onco gene by Varmus and Bishop in 1975, [2] our knowledge of the genetic underpinnings of oncogenesis has expanded at a phenomenal rate. We now know of thousands of genetic mutations that can play a role in cancer formation and have hundreds of drugs targeting specific proteins or cellular processes. In addition, we've discovered that every cancer has a distinct complement of mutations, arises from a unique genetic background and exists in its own personalized environment. As this has become clear, the focus in oncology research has largely moved from searching for a panacea cure toward personalized or precision therapies. Precision oncology is based on the idea that the optimal treatment for any patient is one that is custom tailored to his/her genetic background and the cancer's unique complement of mutations. But how can such a treatment regimen be decided when the number of possible interactions between genetics and pharmacology is easily in the millions?

The answer may lie in machine learning, an emerging field of computer science where machines use carefully constructed algorithms and data to build representations of problems that allow them to identify patterns and make predictions. These models can synthesize of millions of responses influenced by tens of thousands of factors to predict outcomes for a wide variety of possible conditions. Often beginning with random guesses, such algorithms ‘learn’ to improve predictions by adjusting internal parameters to match desired outputs. Once trained with sufficient data, predictions for new examples can be made and the learned weights between the variables in these models can reveal underlying biological relationships that govern the response of the system. Training such models requires an explicitly defined problem with a set of inputs and outputs. For the problem of predicting treatment response in cancer, large-scale cancer cell line screening panels provide exactly this type of combinatorial information and the resulting data have yet to be fully exploited.

Cancer cell line drug screen panels consist of a large number of patient-derived cancer cell lines which can be simultaneously treated by broad range of compounds at varying doses, grown for several days and imaged to assess the effect of the drug treatment. The resulting data is inherently multi-dimensional with one measurement for each combination of cell line, compound and dose. Until now, studies that have attempted to synthesize the information produced by these experiments have largely ignored one of these dimensions by reducing the dose response data to summary measures such as the dose required to effect 50% growth inhibition (GI_{50}) or the area under a fitted dose-response curve (AUC). These studies have shown poor reproducibility with respect to predicted responses and the mechanisms identified, which is perhaps not surprising since they often follow different experimental procedures and compute summary measures differently [3].

The work presented here avoids some of these issues by developing the first model to predict the full dose-response curves for cell line/drug combinations using genomic measurements of cell lines and structural features of drugs as inputs. The traditional measures of response (e.g. GI_{50}, AUC, etc.) can then be calculated from these curves to compare with current methods. By incorporating all of the available data, we hope to improve predictive performance of traditional response outcomes, elucidate mechanisms underlying response, and reconcile findings across studies. Moreover, the model developed here is applicable to any situation where prediction of responses that vary along three independent dimensions is desired. For example, the responses may be measurements of cell lines treated with combinations of drugs, or grown in different micro-environments during treatment.

The model utilizes recently developed Bayesian statistical methods to perform a tensor factorization coupled with matrix projections. An n-dimensional tensor is simply a set of values organized into rectangular array where each entry can be specified by n indices. An ordered list of numbers (a vector) is a one-dimensional tensor, a matrix is a two-dimensional tensor and a three-dimensional tensor can be imagined as a box with numbers spaced evenly throughout. Decomposing a tensor into factor matrices gives a representation of each value in the tensor as a linear combination of latent factors along each dimension. In our model, these factors capture salient differences between drugs, cell-lines and dosage.

By combining this type of factorization with a Bayesian approach to dimensionality reduction we can link the latent factors to predictive features for each of the dimensions (i.e. genomic characteristics of the cell lines and structural features of the compounds) while controlling for model complexity. Specifically, we model the latent factors matrices as products of feature matrices and projection matrices with priors encouraging sparsity in the projection. Thus these latent matrices serve a dual purpose acting as both a reduced representations of the genomic data and factors of the response tensor. The proposed model can use genomic profiling data from cell lines such as genetic mutations, gene expression and copy number variation as well as structural and target information on the compounds to simultaneously predict the growth of each cell line when treated with each drug at each dose. By holding all of the data together in one model, similarities among cell lines and among drugs can inform predictions across all conditions, effectively leveraging all available data for the prediction of each response.

Framing the model as in a probabilistic Bayesian way allows it to handle missing response data and to encourage parsimony in the use of input features while maintaining measures of uncertainty. Predictions for new cell lines and drugs can be obtained by multiplying vectors of input data through the learned projection and latent matrices. Perhaps most impressively, a prediction can be made for both a new cell line and new drug simultaneously, pointing the way toward true precision medicine where we hope to predict responses for patients to a suite of drugs when none of them have yet been prescribed.

This novel approach we call BaTFLED (Bayesian Tensor Factorization Linked to External Data) enables the structured synthesis of large dose-response datasets produced by cell line drug screen experiments in order to make predictions and identify mechanisms of response.

In the following chapter we review background material on the problem domain, theoretical underpinnings of the model and discuss similar approaches. In chapter 2 we introduce our model, giving full details on the mathematical derivations and implementation. Chapter 3 shows results on simulated data that highlight the strengths of the method. Chapter 4 shows results on three real dose-response datasets compared with other state-of-the-art methods. Chapter 5 shows results predicting cellular counts of cancer cell lines grown in varying micro-environments and chapter 6 summarizes the work and discusses extensions and future directions.

Chapter 1

Background

Recent years have seen many studies using screening panels to predict the response of cell lines to drug treatment and a renewed interest both in tensor factorization methods and Bayesian modeling. We highlight some of this work to motivate this project and to place it in context.

1.1 Tumor derived cell lines

Though early cancer researchers sought a single cure, it has become increasingly clear that successful treatment of this incredibly heterogeneous the disease will require many cures each tailored to the individual patient. Both broadly cytotoxic compounds and carefully tailored therapies show a wide range of efficacy across patients. To understand the factors that lead to this heterogeneity of response it is essential to have a method by which a range of therapeutics can be systematically tested in diverse set of genomic backgrounds and outcomes can be assessed quantitatively. Tumor-derived cell lines have provided the opportunity to quickly test drugs in human cells and directly quantify their effects. These cells are taken directly from the patient and transformed via viral infection to prevent the cells from becoming quiescent. Thus the genome of these cells is initially almost identical to the patient's cancer and although mutations can occur over time, genetic profiling has shown that they recapitulate the genetics of patients very closely [4]. Perhaps surprisingly, despite the transformation process, the artificial growth conditions and the isolation of these cells from the *in vivo* signaling environment, many genomic indications of response found using these systems have been shown to be relevant in patients [5, 6].

1.1.1 Cancer cell line screening panels

The first large-scale cancer cell line screening panel was designed by the National Cancer Institute in 1986 and contains 60 cell lines from blood, skin, lung, colon, brain, ovary, breast, prostate, and kidney cancers [7]. To date, the so-called NCI60 have been tested with thousands of compounds and profiled genetically, epigenetically and transcriptionally by dozens of platforms. However, this panel was developed at a time when the majority of cancer therapeutics were nonspecific cytotoxic agents to which most cell lines would respond, so a complement of 6-9 cell lines per tissue type seemed sufficient to determine efficacy [4]. The broad overview provided by the NCI60 is insufficient, however, when testing targeted therapies where responses can be dramatic but rare. For example, the EGFR inhibitors getfitinib and erlotinib are effective in only about 10% of patients with non-small-cell lung cancer [8, 9]. While targeted drugs generally have many fewer side effects and good efficacy in patients that do respond, identifying factors that determine response requires that the set of cell lines being tested cover a larger segment of the possible genetic search space. Hence more recent studies have moved toward either larger screening panels or panels specific to one tissue type.

The large screening studies include the Cancer Cell Line Encyclopedia (CCLE) [10], the Genomics of Drug sensitivity in Cancer (GDSC) [11] and the Cancer Target Discovery and Development (CTD^2) effort [12]. Each of these projects contains hundreds of cell lines and compounds treated at a number of doses in replicates. While these studies share some cell lines and drugs, they are each conducted in a unique manner and have discovered different relationships between genomics and drug response.

There are many smaller tissue-specific studies studying drug response in the most common cancer types. One notable study that will be used for this work is a breast cancer panel containing 70 cell lines, 90 drugs, responses replicated in triplicate and the full datasets have been made public (Heiser et al., 2012). Other similar studies include a panel of 500 epithelial cell lines [13] and a panel of 84 non-small cell lung cancer cell lines [14].

While there is some overlap in cell lines and drugs between these studies and the genomic profiling is largely consistent between studies, the drug response findings do not easily fit into a unified framework. An analysis of the concordance between the CCLE and CGP data finds very poor correlation of summary response measures for the 471 cell lines and 15 drugs shared by both studies [3]. This discrepancy may be due to differences in the growth media, cell count measurements or the ways in which the summary response measures were calculated.

1.1.2 Response measures

Analysis of these data begins with cellular abundance measurements from each well in which cells are grown, typically at different doses and with replicates. Readouts used include the optical density of Cell Titer-Glo (the most common output measuring ATP), Syto60 stain measuring nucleic acids or resazurin dye measuring oxidation-reduction. Measurements at each dose are often normalized by subtracting readings taken simultaneously on background wells with no cells. This accounts for the variation due to illumination and other technical aspects of the measurement process. A second normalization to account for variations in the number of cells initially loaded into the well can also be done if optical density readings are taken at time zero. Some studies also attempt to normalize by the growth rate of the cell line. In this case the measurements at each dose are divided by the optical density with no drug treatment. Thus a value of one shows no effect of the drug and a value of zero indicates total cell killing.

Additionally, differences between studies in how the data is cleaned and pre-processed can be difficult to understand and reproduce. For example, the methods for obtaining drug response measures in Daemen et al. [15] cites Heiser et al. [16], which in turn cites Kuo et al. [17]. Here 20% of the drug-response data is filtered out requiring that "(i) the median SD across the nine triplicate-treated data points was <0.20; (ii) the DT [doubling time] was within 2 SD of the median DT for each cell line; (iii) the slope of the fitted curve was >0.25; (iv) we identified assays with no response by requiring inhibition at the maximum concentration $<50\%$".

The most commonly used response metric is the half maximal growth inhibition or GI_{50}. This is often used interchangeably with the half maximal inhibitory concentration (IC_{50})

assuming that the response that is being inhibited is cell growth. This is the dosage at which a drug inhibits the response of the abundance measure (and presumably the viability of the cell line) by 50%. GI50 is typically calculated by first fitting a curve to the normalized dose-response data and then finding the dose at which this curve crosses the mid-line between the maximal and minimal possible effect (0.5 if the data have been normalized to growth with no drug treatment). There are many variations on this calculation as the curve used for the fit can be a simple line, a symmetric logistic curve or a non-symmetric Gompertz curve. Moreover the minimal effect can be determined by measurements of cell counts in controls with no drug treatment or with the minimal dose. The maximal effect can be determined by the maximal amount of growth inhibition observed at tested doses (in this case the GI_{50} should be referred to as EC_{50}, the half-maximal effective concentration) or by measurements of cell counts at time zero. To complicate this further some studies choose among curves for each cell line-drug pair based on the fit to raw data and use different conventions to decide when the GI_{50} is uninformative. Finally, when training predictive models, the GI_{50} for cell lines that are unresponsive may be missing, set to the maximal dose (as in CCLE data) or inferred by extending the curve beyond measured data (as in the CGP data).

Other measures of response derived from fitted curves include Emax (the dose at which maximal effect is observed), Einf (the theoretical lower asymptote deduced from a fitted logistic curve), total growth inhibition (TGI), the hill slope coefficient (HS: a measure of the steepness of response in a logistic curve) and AUC (area under the calculated dose response curve or under the median observed response values), figure 1.1. Each of these measures has a different interpretation with respect to the expected behavior of drug treatments in patients. GI_{50}/IC_{50} is used as a general measure of potency, Einf, Emax and TGI are measures of efficacy and AUC combines aspects of both [18].

A recent study comparing several of these measures on the same raw dataset found a worrisome lack of correlation between them across cell lines and drugs [18]. This indicates that by relying on only one measure, studies are oversimplifying the true response mechanics and possibly missing important elements regulating response. For example, the slope of the response curve is not captured by the IC_{50} or AUC value. This information

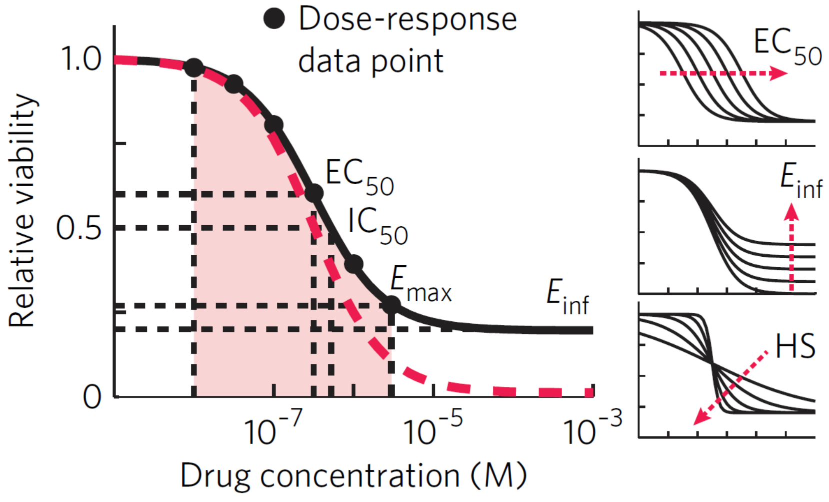


Figure 1.1: Common measures of drug response. The red dotted line shows the case where the drug is completely effective at maximal dose. Here Emax=Einf=0 and IC50=EC50. The pink area represents the AUC. [18]

may be valuable for patient responses where giving the minimal effective dose while avoiding toxicity is important. Moreover, certain sub-populations may not exhibit the typical logistic response curves at all. By routinely fitting the same class of curves to the data and extrapolating measures, the variation in response across replicates is often ignored and unexpected types of responses are missed. If genetic factors influence whether a given cell line responds to varying doses of a drug according to a logistic curve, relationships with these confounding predictive factors are also missed.

The trouble with these summary response measures becomes even more apparent when comparing results across studies. Two papers by Dr. John Quackenbush's group finds very little correlation in predicted responses for cell line-drug pairs profiled in both the CCLE and CGP projects and subsequently different genomic predictors of response [3, 19]. Specifically, Spearman rank correlation of drug-sensitivity measurements for 13 out of 15 shared drugs was less than 0.5 across the 471 shared cell lines. They suggest that the discrepancies may be due to differences in the experimental setup, normalization procedures or calculation of the outcome measures, but were unable to dig deeper because the raw response data for the CGP study was not available at the time. The groups involved in the initial studies attempt to address these issues in [20] and find better agreement when taking into account more biological knowledge of the drugs and cell lines, but this issue is still contentious [21] and clearly there is room for improvement.

One large confounding factor may be the differing growth rates of cell lines given the cell culture conditions. Hafner et al. [22] normalize responses for growth rate at each dose in a new way introducing a new set of response measures. The normalized growth rate at time t in the presence of drug at concentration c is taken to be

$$GR(c,t) = 2^{k(c,t)/k(0)} - 1 \tag{1.1}$$

where $k(c,t)$ is the growth rate of the treated cells and $k(0)$ the growth rate of untreated cells. In this paper, switching to these GR measures (GR_{50}, GRinf, etc.) reduces correlation with the number of cells seeded and dependencies on growth media.

The work presented here aims to avoid the inconsistencies caused by different curve-fitting procedures and methods of summarizing response by directly predicting the optical density or normalized growth measures at each dose. With these predictions in hand the summary response measures detailed above can be calculated and compared to current studies to assess performance. This will eliminate any bias introduced by using specific measures and allow systematic comparisons between studies. This is the first model to attempt prediction at each dose, and can be seen as an extension of previous work into a higher dimensional setting.

1.2 Current models in response prediction

While many different computational approaches have been used to predict the responses of cell lines to drug treatments with some success, recent large-scale review efforts have shown that the most successful approaches follow several clear principals.

Jang et al. [23] performed a systematic assessment of over $11,000$ different models predicting responses in both CCLE and CGP datasets. Each of these models predict responses to a single drug for 20% of the cell lines after training the models with known responses for 80% of cell lines. The researchers varied the type of input molecular feature data, the prediction algorithm, the compound being predicted, the outcome being predicted and whether the outcome data was discretized or continuous. The first principal that they discovered is that using more input molecular profiling data improved performance more than any of the other factors, including which algorithm was used. Among input data types, expression profiles had the largest effect. Second, they found that dimension reduction algorithms favoring regularized solutions like elastic net or ridge regression performed best. Third, they found that predictions of AUC were generally more accurate than IC_{50} or EC_{50} and hypothesize that this is because the AUC measure captures more of the information contained in response curves.

Another approach attempting to find the best computational methods to solve this problem was implemented by a collaboration between the National Cancer Institute (NCI) and the Dialogue on Reverse Engineering Assessment and Methods (DREAM) project [24]. This DREAM challenge released data publicly so that teams from around the world could compete in the task of predicting responses in a panel of 53 cell lines treated with 28 compounds. The best performing algorithm out of the 44 submissions used a Bayesian multitask multiple kernel learning (MKL) approach. This method combines genetic profiling data for all cell lines through kernels which measure similarities between cell lines and reduce the dimensionality of the genetic input data. The kernels are combined using weights shared across drugs and these linear combinations of kernels are given weights for each cell line that are specific to each compound. In this way, predictions for all drugs are made simultaneously so information about a cell line's response to one drug can be leveraged in the prediction of other drugs. This study supported the findings of Jang et al. [23], in that incorporation of more data and dimension reduction methods both improved prediction. Additionally it showed the benefit of combining the tasks of predicting outcomes for each drug into one model.

An extension from the same group that won the DREAM challenge expands the MKL idea to incorporate known structures and targets of drugs to improve prediction [25]. This Kernelized Bayesian Matrix Factorization (KBMF) method marries the field of Quantitative Structure-Activity Relationship (QSAR) discovery to drug response prediction. The goal of QSAR studies is to use the structural features of drugs to predict their chemical activity. By extending this to predicting the result of that activity (namely inhibiting cell growth) across cell lines, this approach moves toward true precision medicine. The central idea of this method is to factor the matrix of response values (with rows corresponding to cell lines and columns to drugs) into latent factors encoding information about similarities between cell lines or drugs respectively. These factor matrices are constrained by kernel representations of the input data (now both drug and cell line features) and the optimal factorization is learned through an efficient search of parameter space using Bayesian methods (Figure 1.2).

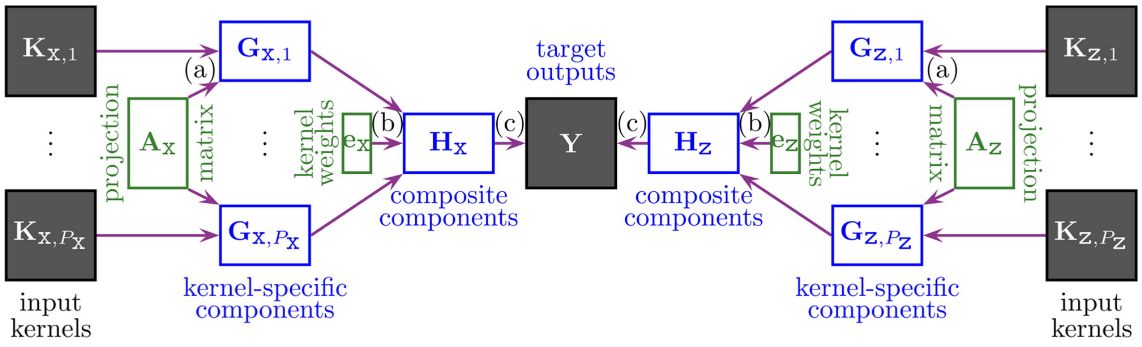

$\mathbf{K}_{\mathbf{X},1}$
$\mathbf{K}_{\mathbf{X},P_{\mathbf{X}}}$
input kernels
projection
$\mathbf{A}_{\mathbf{X}}$
matrix
(a)
$\mathbf{G}_{\mathbf{X},1}$
$\mathbf{G}_{\mathbf{X},P_{\mathbf{X}}}$
kernel-specific components
kernel weights
$\mathbf{e}_{\mathbf{X}}$
(b)
$\mathbf{H}_{\mathbf{X}}$
composite components
(c)
target outputs
$\mathbf{Y}$
(c)
$\mathbf{H}_{\mathbf{Z}}$
composite components
(b)
$\mathbf{e}_{\mathbf{Z}}$
kernel weights
$\mathbf{G}_{\mathbf{Z},1}$
$\mathbf{G}_{\mathbf{Z},P_{\mathbf{Z}}}$
kernel-specific components
(a)
matrix
$\mathbf{A}_{\mathbf{Z}}$
projection
$\mathbf{K}_{\mathbf{Z},1}$
$\mathbf{K}_{\mathbf{Z},P_{\mathbf{Z}}}$
input kernels

Figure 1.2: Kernelized Bayesian matrix factorization model. Input data are in shaded boxes, learned parameters are in green and latent representations in blue. The central matrix of IC_{50} response values is denoted by Y. Input kernels for cell lines are on the left and input kernels for drugs are on the right. Each side is multiplied by a shared projection matrix to form the latent kernel-specific components. These are then combined via the kernel weights to form composite components which multiply to form the central response matrix [25]

The method presented here is an extension of the above work into higher dimensional spaces. We expand the central matrix object into a tensor of three dimensions to capture the response of each cell line when treated with each drug at each dose. This removes the reliance on any one measure of response (IC_{50}, EC_{50}, Emax, TGI or AUC) avoiding issues due to curve fitting and choice of summary measure and will incorporate all of the available response information. Also, our method does not project the feature data through kernels feeding cell line and drug characterization data directly to the projection matrices (although we do experiment some with using kernels). This will allow for greater interpretability of both the values in the model and the learned weights placed on cell line and drug features. Additionally, as the above studies show that incorporating more predictive data sets improves model performance we incorporate extensive genomic profiling data on cell lines and structural and target data on drugs as predictors. Lastly, the use of Bayesian methods will allow for sparsity inducing priors analogous to the elastic net regression method suggested by Jang et al. [23].

Since, only a small number of studies use factorization methods for drug-response prediction, a broader view of these techniques can help to elucidate their usefulness. In the following sections we explore the use of matrix and tensor factorization techniques in fields tangentially related to the prediction task presented above.

1.3 Matrix factorization

The motivation to use tensor factorization methods stems from their capacity to organize large datasets with several independent axes of variation and from the great utility that matrix factorization methods have shown in somewhat simpler problems. For example principal component analysis (PCA) [26, 27] is often the first tool used to gain insight into relationships in data with a large number of variables. This method is based on the singular value decomposition (SVD, a type of matrix factorization) of a data matrix with rows representing each repetition of an experiment and columns each measurement. Although SVD dates to the late 19th century, it is still bearing fruit in algorithm development for data analysis as evidenced by the surrogate variable analysis (SVA) [28] which seeks to eliminate unwanted âĂIJsurrogate variablesâĂİ in gene expression data or other large matrices.

Many matrix factorization problems can be thought of in the framework of factor analysis which seeks to find a set of latent factors that explain observed relationships in data. For example a study with I individuals responding to J survey questions may seek a set of underlying factors and weights for these factors specific to each individual so that each answer can be expressed as a linear combination of the factors. Mathematically, this amounts to finding a matrix of factors F (with N rows and J columns) and a matrix of weights A (with I rows and N columns) so that the observed response matrix Y (with I rows of individuals and J columns of questions) is a product of A and F plus some minimal error matrix E.

$$Y = A \times F + E$$

component wise

$$y_{ij} = a_{i1}f_{1j} + a_{i2}f_{2j} + \; + a_{in}f_{nj} + e_{ij}$$

In the above example the responses of an individual to a survey about their job may be decomposable into factors that capture elements of age, general happiness, etc. and each person would have a different weight on how these factors affect their answer to each question. Finding this decomposition requires specifying how many factors should exist (N) as well as identifying each of the elements in A and F.

Since there are a large number of parameters in these model, a unique solution typically does not exist and there are various assumptions made to find the 'best' solution. We may prefer a solution with the smallest number of factors, least error, or one with factors that best correlate with known properties of the individuals (e.g. age, sex, etc.). Mathematically, this can be interpreted as a change of basis vectors for the space of observations (encoded by F) and is often referred to as a 'rotation'. Machine learning algorithms that employ matrix factorization (sometimes implicitly) essentially search the space of possible rotations looking for one that optimizes an objective function formulated to satisfy requirements set by the problem.

Since matrices hold values for relationships between pairs of variables and we are interested in outcomes that result from the interaction of three or more dimensions of variation, it is natural to look for higher order analogs of matrix factorization. Tensor factorization methods provide this generalization and when combined with probabilistic Bayesian estimation allow for the flexibility necessary to handle missing data and make predictions for situations that were not encountered in training data. Finding an optimal tensor factorization for a given problem similarly amounts to searching the space of possible rotations in order to optimize an objective function.

Although these approaches are young, successes in other fields combined with a good performance of probabilistic Bayesian matrix factorization methods on problems similar to the one presented here [25, 29, 30, 31] indicate that this line of research is likely to be very productive.

1.4 Tensor factorization

A tensor in this setting is simply a multidimensional array or matrix. These are not to be confused with tensors or tensor fields used in physics (such as stress tensor) which have more structure. The order (or mode) of a tensor is the number of dimensions of data. A first-order tensor is just a vector, a second-order tensor is a matrix and a third-order tensor is a rectangular prism of values. Although higher order tensors can be difficult to picture, much of the math used in third-order tensors extends naturally to any N-order tensor.

The notation throughout this paper mostly follows the conventions of the excellent review by Kolda and Bader [32]. I will use boldface capital letters to refer to tensors (e.g. $\mathbf{X}$) , capital letters (e.g. X) to refer to matrices (and index upper limits), boldface lower case letters to refer to vectors (e.g. $\mathbf{x}$) and normal letters to refer to scalars (e.g. x). Indices will typically range from 1 to their capital version so that the three modes of the third-order tensor $\mathbf{X}$ might have indices i,j and k where $i = 1 \dots I$, $j = 1 \dots J$ and $k = 1 \dots K$. A particular element of the tensor $\mathbf{X}$ will be represented by the lower case version of the tensor name subscripted by i,j and k (e.g. x_{ijk}). A one dimensional subset of a tensor obtained by fixing all other dimensions will be referred to as a fiber and represented as a vector replacing the non-fixed index with a colon (e.g. $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$ or $\mathbf{x}_{ij:}$). Similarly a two dimensional slice is a matrix and is represented with a capital letter with two dimensions replaced with colons (e.g. $X_{i::}$, $X_{:j:}$ and $X_{::k}$) Examples of these subsets for a three mode-tensor are pictured in (Figure 1.3) [32].

In order to talk about tensor factorization we must first define tensor multiplication. Beginning with first-order tensors (vectors) we can construct a higher-order tensor with the outer product (denoted by $\bigotimes$). The outer product is obtained by multiplying every element of each of the N vectors by every element of the other vectors. The order of the operation determines where the products get placed in the resulting tensor. For example the product of the third element of the first vector, the second element of the second vector and the first element of the third vector will be in the third row, second column, first frontal

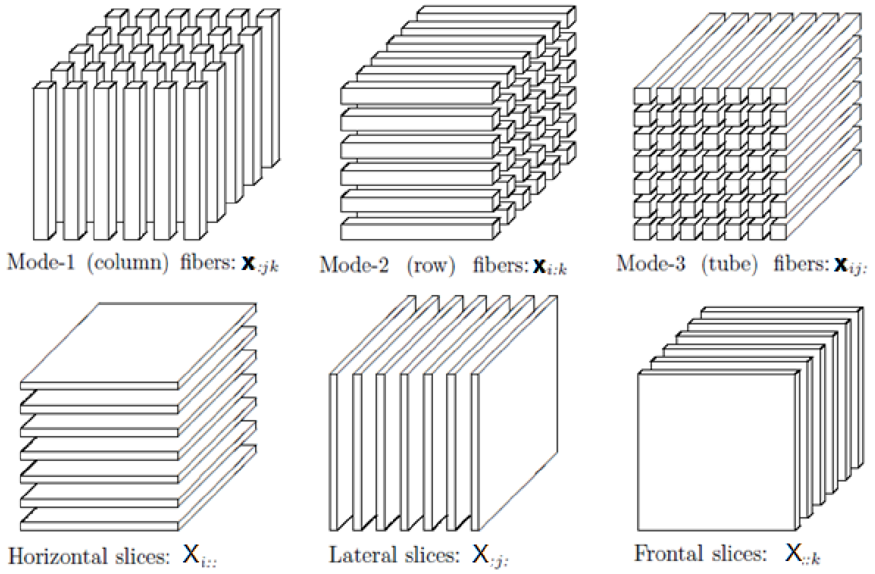

Mode-1 (column) fibers: $\mathbf{x}_{:jk}$
Mode-2 (row) fibers: $\mathbf{x}_{i:k}$
Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$
Horizontal slices: $\mathbf{X}_{i::}$
Lateral slices: $\mathbf{X}_{:j:}$
Frontal slices: $\mathbf{X}_{::k}$

Figure 1.3: Lower dimensional subsets of a third-order tensor (Kolda and Bader, 2007) [32]

slice of the resulting tensor (examples below).

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \bigotimes \begin{bmatrix}4\\5\\6\end{bmatrix} \bigotimes \begin{bmatrix}1\\2\end{bmatrix} = \mathbf{X}$$

$$\text{where } X_{::1} = \begin{bmatrix}4 & 5 & 6\\8 & 10 & 12\\12 & 15 & 18\end{bmatrix} \text{ and } X_{::2} = \begin{bmatrix}8 & 10 & 12\\16 & 20 & 24\\24 & 30 & 36\end{bmatrix}$$

The n-mode product (denoted by $\times_n$) of a tensor $\mathbf{X}$ with a matrix U is defined for matrices with the same number of columns as the n^{th} mode of $\mathbf{X}$ and is obtained by multiplying each n-mode fiber with the matrix U. For example, given the above tensor $\mathbf{X}$

and the matrix

$$U = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

$$\mathbf{X} \times_1 U = \mathbf{Y} \text{ where}$$

$$Y_{::1} = \begin{bmatrix} 88 & 110 & 132 \\ 112 & 140 & 168 \end{bmatrix} \text{ and } Y_{::2} = \begin{bmatrix} 176 & 220 & 264 \\ 224 & 280 & 336 \end{bmatrix}$$

A tensor with n modes that can be constructed from an outer product of n vectors is, in a sense, simple and is known as a rank-one tensor. The above, **X** is an example of a third order rank-one tensor. Any N-order tensor can be constructed from a sum of rank-one N-order tensors and finding this set of rank-one tensors is equivalent to the simplest type of tensor factorization.

1.4.1 CANDECOMP/PARAFAC decomposition

The CANDECOMP/PARAFAC (CP) tensor decomposition was first proposed by Hitchcock in 1927 [33], but the idea didn't gain popularity in data analysis until the early 1970s when it was used in the field of psychometrics [34] and phonetics [35]. This method breaks a given N-order tensor into a finite sum of R rank-one N-order tensors whose vector components can be rearranged to form N factor matrices (Figure 1.4). For example, given a three-mode tensor $\mathbf{Y}$ indexed by i,j and k, the CP decomposition gives R rank-one three-mode tensors. Each of these can be represented as a product of three vectors a_r, b_r and c_r of lengths I, J and K respectively. By assembling the vectors for each dimension into matrices with R columns, the original tensor can be represented as an n-mode product of a core tensor $\mathbf{D}$ and three factor matrices A, B and C with dimensions $I \times R$, $J \times R$ and $K \times R$ respectively.

$$\mathbf{Y} = \mathbf{D} \times_1 A \times_2 B \times_3 C$$

In this case the core tensor $\mathbf{D}$ is a cube with dimensions $R \times R \times R$ with ones along the diagonal (i.e. $d_{ijk} = 1$ if $i = j = k$ and 0 otherwise). Component-wise we have

$$y_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} \tag{1.2}$$

Often an exact factorization isn't needed (or desired), so less rank-one components are used, the factor matrices contain less columns, and an error tensor ($\mathbf{E}$) accounts for differences between the initial tensor and the reconstruction.

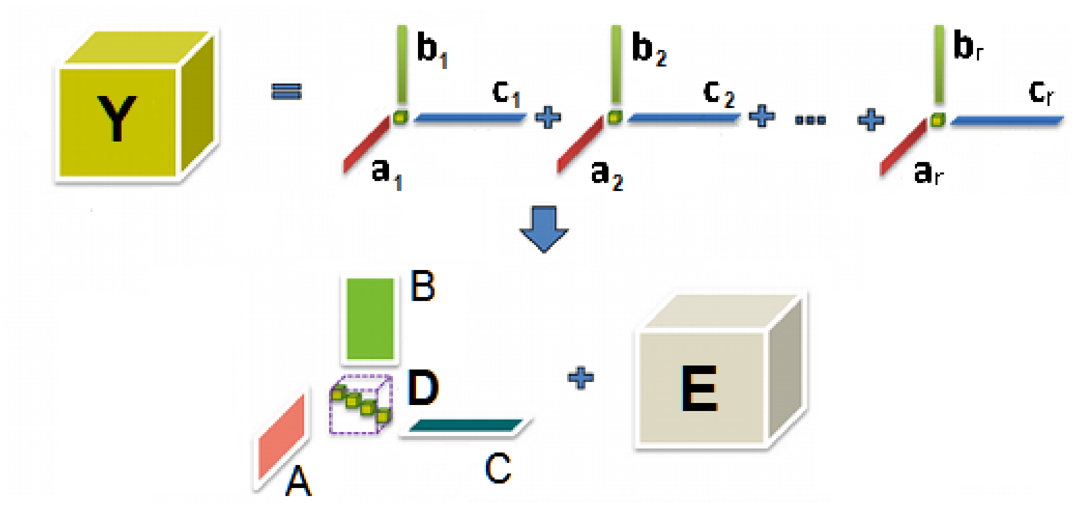


Figure 1.4: CANDECOMP/PARAFAC decomposition of a third-order tensor (adapted from `http://www.bsp.brain.riken.jp/~zhougx/tensor.html`)

While matrix factorizations are often not unique (in fact much of current cryptology relies on this fact) the CP tensor decomposition is unique (aside from scaling and permutation of the dimensions) provided the true rank of the tensor is small compared to its dimensions. This is not true of other tensor factorization methods and may be too restrictive to allow the training of models that are linked to predictor variables in an interpretable framework. Additionally, the requirement that the core tensor be square and diagonal means that each of the input data types must be reduced to projection matrices with the same number of columns and that interactions between modes can only occur between columns in the same location in each of the factor matrices.

This decomposition has been used in many fields including psychometrics [34], phonetics [35], chemometrics [36, 37, 38, 37, 39, 40], signal processing [41, 42], telecommunications [43, 44, 45] and independent component analysis [41, 46].

1.4.2 Tucker decomposition

The more general Tucker decomposition is an extension to principal component analysis [47, 48, 49] and is also known as higher-order SVD [50] and N-mode principal component analysis [51]. This factorization also breaks an N-order tensor into a core tensor and N factor matrices, but allows for a different number of columns for each factor matrix (Figure 1.5). In this case, the dimensions of the core tensor are not all the same, it is not necessarily diagonal and it encodes interactions between the different columns of the factor matrices. Again the original tensor $\mathbf{Y}$ can be expressed as a series of n-mode products between the core tensor $\mathbf{G}$ and the factor matrices A, B and C but the core $\mathbf{G}$ now has dimensions $R_1 \times R_2 \times R_3$ and the factor matrices A, B and C have dimensions $I \times R_1$, $J \times R_2$, and $K \times R_3$ respectively.

$$\mathbf{Y} = \mathbf{G} \times_1 A \times_2 B \times_3 C$$

$$y_{ijk} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} a_{i r_1} b_{j r_2} c_{k r_3}$$

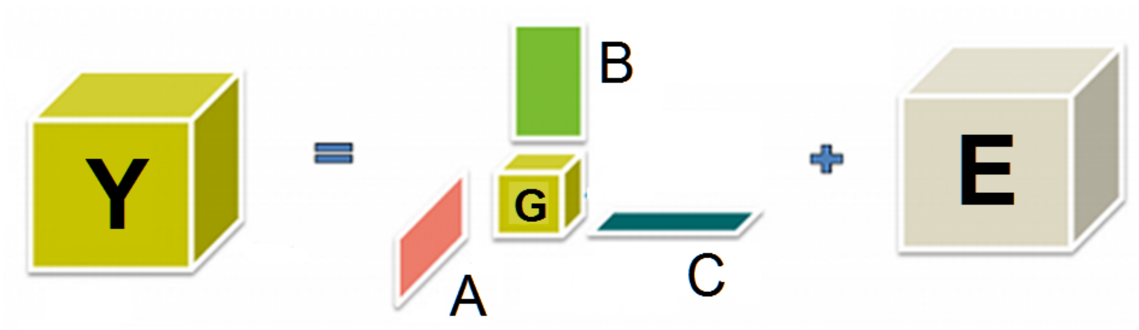


Figure 1.5: Tucker decomposition (adapted from `http://www.bsp.brain.riken.jp/~zhougx/tensor.html`)

This structure allows for much more flexibility, so while CP decompositions are often unique, infinitely many Tucker decompositions can be constructed for a given tensor.

There are several methods for choosing between these ‘rotations’, but since the core tensor encodes dependencies between the factor matrices, an obvious choice of rotation is one that maximizes the simplicity of the core. Some measures of simplicity that can be used are super-diagonality, slice-wise diagonality or minimum of variance-of-squares [52, 53]. These models have been applied in many of the same fields as CP decompositions [54, 55] as well as in facial recognition [56], the semiconductor industry [40] and to identify batch effects in experiments [57].

1.4.3 Algorithms for finding decompositions

Several algorithms have been developed to find the best factorization for a tensor derived from experimental data. In most cases the tensor will be ‘noisy’ meaning that an exact factorization is infeasible or undesirable. In a typical application the users seeks a decomposition that approximates the given data tensor within some error bounds i.e.

$$\mathbf{X} = \mathbf{D} \times_1 A_1 \times_2 A_2 \times_3 \ldots \times_N A_N + \mathbf{E}.$$

where $\mathbf{E}$ is an error tensor of the same size as $\mathbf{X}$ with a small Frobenious norm (square root of the sum of squared entries). There is a trade off here between the size of the factor matrices and the size of the norm of the error tensor. Given a certain error tolerance, one can try increasing the number of columns in the factor matrices (i.e. rank-one components in the CP model) until the desired fit is achieved. There is, however, a problem with this method as it has been shown that some tensors have arbitrarily close low-rank approximations. These can cause approximation methods to get stuck in ‘swamps’ of undesirable solutions [58] and several methods have attempted to overcome this setback [59, 60, 61].

Given a fixed number of components, the simplest and most used approach for finding a decomposition is known as alternating least squares (ALS) [34, 35]. This method begins with a random guess at the factor matrices and proceeds by fixing all but one factor matrix and solving the resulting linear least squares problem on the free matrix. Each factor matrix is solved in turn while keeping the others fixed. This method is simple to implement but lacks a real mathematical basis, has been seen to converge slowly and depends highly on the initial guess [62]. Many improvements have been proposed, some of which improve calculation times, but the solutions found are generally not better than those obtained through ALS (for comparisons see [63] and [64]).

Alternatives to the ALS method include Markov chain Monte Carlo (MCMC) and variational approximation. These methods treat the values in model as random variables and approximate the most likely joint probability distribution across these variables given the observed data. MCMC approaches sample solutions from the distribution while attempting to cover the full range of values for each variable. If this search space is sufficiently explored then the distribution of values obtained from the sampling will approximate the desired probability distribution. Although often slow to converge, this method allows one to place any type of distribution on the variables. For an example of a tensor factorization method using MCMC sampling to predict different types of links in a graph see [65]. Variational approximation is a similar technique that can be much more efficient but requires more theoretical work, can restrict the distributions in the model and only approximates a simplified version of the true probability model. Distributions for variables in the model must be conjugate and complex update equations must be derived analytically. This method was employed by Ammad-ud-din et al. for matrix factorization [25], Zhao et al. for CP decomposition with missing data [66], and Xu et al. for Tucker factorization [67]. See section 1.5 for a more detailed discussion of these methods.

1.4.4 Previous usage of tensor factorization methods

Although the basic tensors factorization methods were first mathematically defined in the 1920s [33], applications didn't begin to be designed until the 1970s [34, 35] and widespread use has only come with the advancement of Bayesian methods and computational power in the last 15 years. There are many sources detailing recent methods including Kolda and Bader [68] and the exhaustive "Three-Mode bibliography" compiled by Kroonenberg (`http://three-mode.leidenuniv.nl`).

Some interesting highlights from imaging include the following. A method for facial recognition dubbed 'Tensor Faces' that decomposes an image into factors representing the identity of the person, the angle at which they are being viewed and the illumination present (Figure 1.6) [56]. Once trained, the decomposition can be used to identify the subject of a new image by placing it in the tensor and summarizing over the view and illumination dimensions. A related method by Vlasic et al. [69] that uses a tensor representation of facial expression to map expressions from one face onto the background of another person is able to separate latent factors representing facial expressions from the individual's identity (Figure 1.7). Tensor factorization has also been used for video separation as in [70] where a low-rank tensor captures background elements of video while a sparse tensor captures elements that change between frames (Figure 1.8).

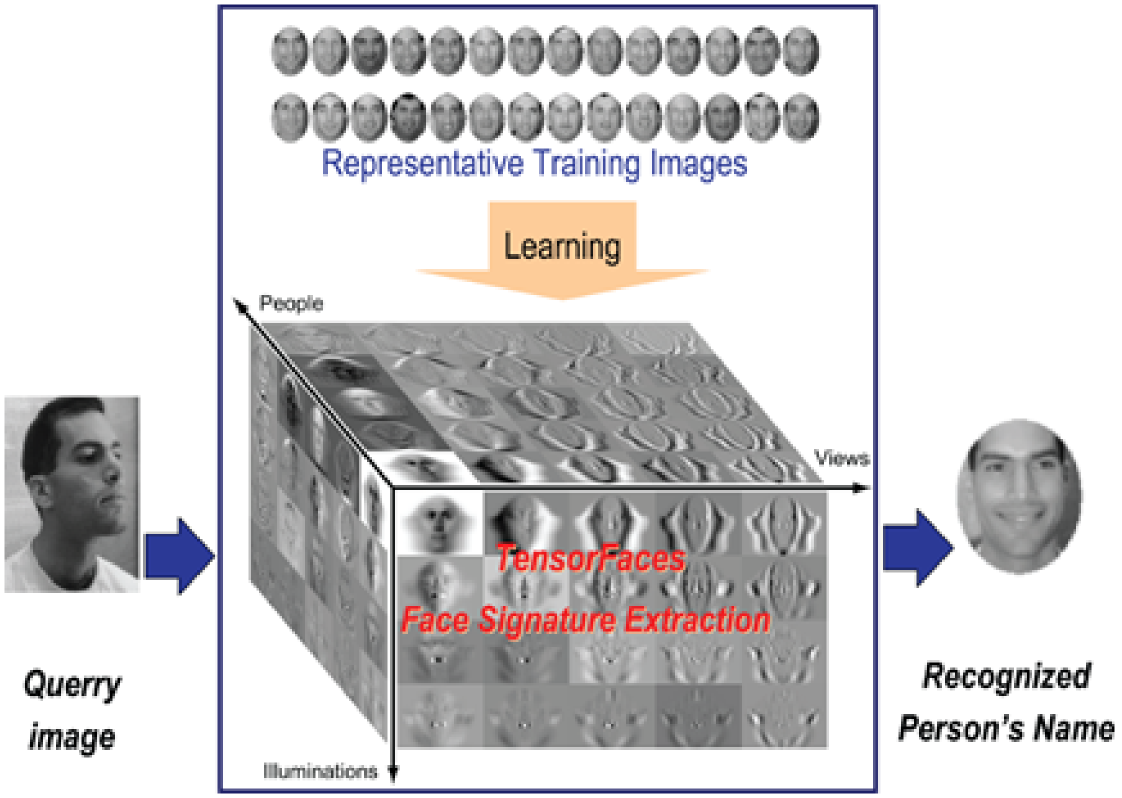


Figure 1.6: Tensor Faces. The three dimensions of the tensor capture information on the identity of the image, the angle from which he is viewed and the lighting conditions [56])

Tensor methods also tend to deal well with large amounts of missing data. Examples include two papers that predict movie preferences with up to 98% of the possible ratings missing [71, 31]. Also, Acar et al. develop a CP decomposition method to reconstruct tensors of EEG data and computer network traffic that perform well with up to 99.5% of the data missing [72]. There are also a number of methods for reconstructing video with

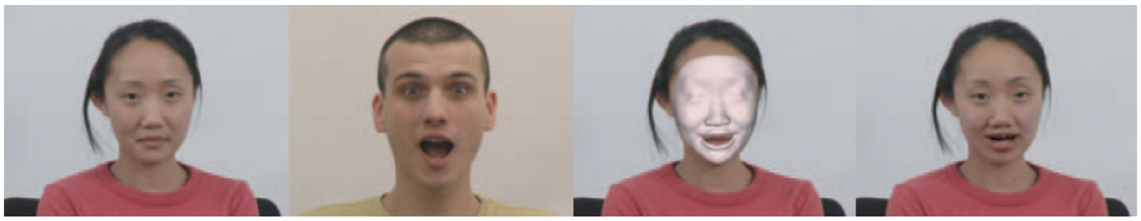


Figure 1.7: Mapping expressions from one face to another. The face of the woman in the first image is used a background on which the expression from the second image is super-imposed [69]).

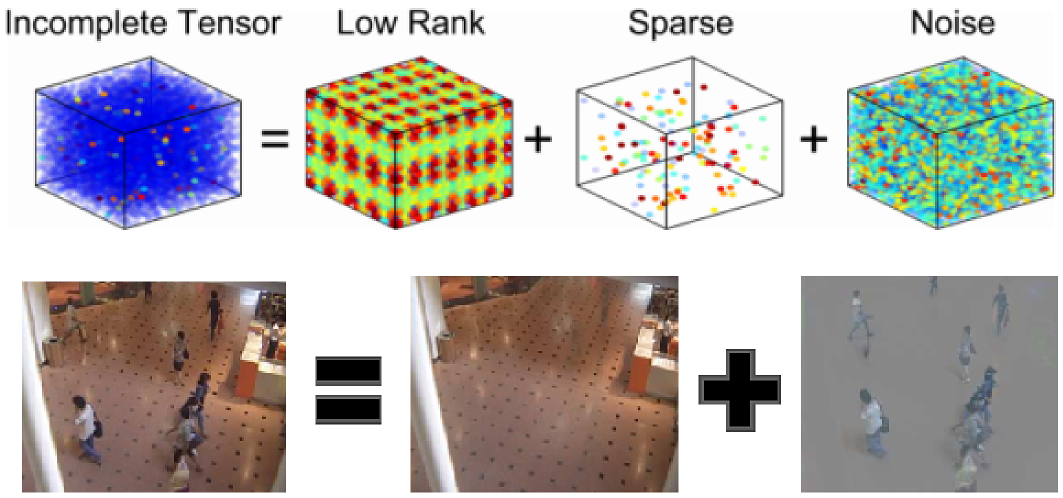


Figure 1.8: Using tensor factorization for video separation. A set of images over time possibly missing pixels is separated into low-rank, sparse and noise tensors. The low-rank approximates background while the sparse tensor reconstructs the parts of the image that change over time [70]).

many missing pixels or three-dimensional imaging stacks with missing voxels [73, 74, 75].

Other recent work involving tensor factorizations has involved incorporating tensors into machine learning techniques like neural networks. Researchers in these areas have tried adding layers of tensor factorization into neural networks [76, 77, 78] and using tensor factorization in place of the standard back-propagation algorithms to train neural networks [79]. Tensors may even help to explain how deep neural networks are able to perform so well, and indicate possible paths for improving these tools. Recent theoretical work by Drs. Shashua and Cohen shows that certain types of deep convolutional networks are equivalent to tensor factorizations and that this may explain why deeper nets perform better than shallow ones [80, 81, 82].

Finally, tensors have been used to some extent in bioinformatics. Engelen et al. use a CP decomposition to analyze fluorescence data [83] and Omberg et al. build a Tucker model to decompose a tensor of yeast gene expression microarray data from different studies across three modes: genes, time, and treatment [84]. The factorization learns reduced representations of expression (eigengenes) that correlate with known mechanisms of response. Ponnapalli et al. build on this idea to look for patterns of gene expression across multiple species [85]. Recently, Hore et al. use a CP decomposition of gene expression data from different patients and tissues to identify latent expression components [86]. Also, tensors are used quite frequently in studies of the brain since MRI and other such data are inherently three-dimensional. For example Onken et al. explore the utility of tensor factorization methods to understand spatially correlated neuron firing events [87]. Finally, Kim et al. use a CP factorization for phenotyping patient data from electronic health records [88].

1.5 Probabilistic modeling and Bayesian methods

Our method frames tensor decomposition as a statistical model where we place distributions on the unknown values of the model and look for parameters for those distributions that have the highest probability given the data and model assumptions. This is essentially a Bayesian approach since we are reversing a typical statistical question. Suppose that we have an observed tensor of responses and we know exactly how these data were generated, i.e. we have the true factorization matrices and core tensor, but we expect there to be some random component to the process of generating the responses and our observations are just representatives from a distribution of all possible responses. In this case, the values in the factor matrices and core tensor also have to come from some distribution since the multiplication process is completely deterministic. If we know the parameters (e.g. mean, variance) of these distributions we can calculate how likely our observations are given this model $p(data|model)$. The use of Bayes' rule allows us to reverse this question. Now we imagine that we have a set of responses, but we don't know how they were generated. We assume some random process (statistical model) that generates our data with unknown parameters. For any given set of parameters, we can determine the probability of our data given the model, but we now want to know the probability distribution of the model parameters given the data $p(model|data)$. This posterior distribution allows us to ask what parameter settings would give the highest probability to the data that we observe and get estimates as to how certain we are of these values. Moreover, a probabilistic framework allows us to handle missing values in the responses.

$$p(model|data) = \frac{p(data|model)p(model)}{p(data)} \tag{1.3}$$

In larger models like those used here, it is impossible to compute the posterior probability analytically and we must use approximate methods instead. We briefly present two types of approximate methods: sampling methods use random draws from the distribution to approximate its parameters and variational inference finds an optimal solution to a simpler distribution that is, in some sense, close to the true posterior.

1.5.1 Sampling methods

The most common sampling methods for this type of problem are known as Markov chain Monte Carlo or MCMC methods. The 'Monte Carlo' part refers to the random aspect of iteratively choosing samples from distributions and 'Markov chain' indicates that each sampled state depends only on the previous state and that given enough iterations, the system will converge to some equilibrium. In this case our posterior probability can be written out as a set of interlocking conditional distributions each of which we know how to sample from. We start at some random initial state and sample the next state according to some sampling schema. The Gibbs sampler is one such schema that is widely used [89]. Here, we draw a sample from the conditional distribution for each variable while holding the others fixed. After some initial 'burn in' period, the samples can be used to estimate the true posterior but with larger models with a lot of interlocking conditional distributions, it can take a long time for the samples to sufficiently explore the space. The sampling may get stuck in certain regions of high probability and not find other solutions especially if there is a region of low probability in between. This is is often the case with distributions that encourage sparsity in some aspect of the model.

Sampling approaches are used in a large number of fields and there are many good resources describing these ideas. For starters, Andreieu et al. explain sampling approaches in general, MCMC, Gibbs sampling and give a some nice historical background [90]. Other more detailed sources include books by Robert and Casella, Gelman, and Bishop. [91, 92, 93]. Several software packages have been developed to perform Gibbs sampling given a set of dependent probability distributions. These differ slightly in their implementation, with the newer tools (JAGS and Stan) being more efficient, but the specification of models is fairly standard and documentation for these packages are quite good [94, 95, 96].

MCMC approaches have been used in a few papers for tensor factorization, but most applications seem to prefer variational inference. Porteous et al. develop a probabilistic model for matrix and tensor factorization that is trained using Gibbs sampling [97]. Xiong et al. introduce "Bayesian Probabilistic Tensor Factorization (BPTF)" model which learns a CP decomposition using MCMC sampling and evaluate its performance on sales prediction and movie recommendation tasks [31]. More recently, Khan and Ammad-ud-din have released an R package that uses Gibbs sampling to perform a CP tensor decomposition of a three-mode tensor [98].

1.5.2 Variational inference

For our model, a sampling approach would be computationally intractable, so instead we use variational inference (VI) (see [93] for a more complete discussion). With this method, the goal is still to find the posterior distribution of the model parameters given then data, but instead of sampling from the this distribution, we approximate the posterior with a simpler distribution and optimize parameters of this new distribution. The optimization seeks to minimize the distance between the true posterior and this simpler version in terms of the Kullback-Leibler divergence [99]. More specifically, if we denote the set of observed variables and set parameters by Ξ and the set of all random variables by Θ, then the posterior distribution $p(\Theta|\Xi)$ can be approximated by a simpler 'q' distribution $q(\Theta)$. Typically, the posterior distribution can be written as a product of conditionally independent distributions with known forms. For example, in a CP tensor factorization model I may say that a value y_{ijk} in the tensor has a Gaussian distribution where the mean is equal to the product given in equation 1.2 and the variance is fixed at σ^2.

$$p(y_{ijk}) = \mathcal{N}(y_{ijk}; \sum_{r}^{R} a_{ir} b_{jr} c_{kr}, \sigma^2).$$

Thus y_{ijk} is independent of the other values in the model given that we known the vectors $\mathbf{a}_{i:}$, $\mathbf{b}_{j:}$ and $\mathbf{c}_{k:}$.

With variational inference we remove the dependency between the parameters by introducing a distribution $q(y_{ijk})$ in the same family as $p(y_{ijk})$ (i.e. Gaussian) but with unknown mean and variance parameters.

$$q(y_{ijk}) = \mathcal{N}(y_{ijk} | \mu(y_{ijk}), \sigma^2(y_{ijk}))$$

The new variational parameters ($\mu(y_{ijk})$ and $\sigma^2(y_{ijk})$) now implicitly depend on the structure and data in the original model and formulas for their optimal values can be found analytically. Optimal here means that they minimize the KL-divergence between q and p, a pseudo-distance between distributions ($\mathbb{E}_q$ representing the expected value over the q distribution).

$$KL(q||p) = \mathbb{E}_q \left[\log \frac{q(\Theta)}{p(\Theta|\Xi)} \right] \tag{1.4}$$

In particular the logarithm of the optimal values in the q distribution for each variational parameter θ can be obtained by finding the expectation of $\log p(\Theta|\Xi)$ with respect to the q distributions over all other variables (denoted $\Theta \backslash \theta$). This gives expressions for each of the q distributions that depend only on expected values of the other variables (and fixed parameters).

$$\log q(\theta) = \mathbb{E}_{q(\Theta \backslash \theta)} \left[\log p(\Theta|\Xi) \right] \tag{1.5}$$

Unfortunately, the expected values of the q distributions are not known *a-priori*, in fact these are our estimates for the unknown values in the original model, but an expectation maximization approach works here. We begin with a random initialization of the variational parameters and then iteratively update each variational parameter given the expectations of the other parameters in the model. At each step we find the optimal value for the chosen parameter so, other than the initialization, the process is completely deterministic and will converge to a solution. Since we are not sampling from the distributions, but updating the parameters directly, VI is often much more computationally efficient than sampling methods.

The downside is that we obtain an optimal solution to an approximate distribution as opposed to MCMC approaches that find an approximate solution to the true distribution. Also, derivation of the update equations can be tedious, and coding these complex update equations may introduce errors. One way to mitigate these is to monitor the evidence-based lower bound (ELBO, written $\mathcal{L}$) which is maximized when KL-divergence between the q and p distributions is minimized. This quantity can be computed during training both to check for convergence and to help identify errors.

$$\log p(\Theta|\Xi) = \mathcal{L}(q) + KL(q||p) \tag{1.6}$$

and

$$\mathcal{L} = \mathbb{E}[\log p(\Theta, \Xi)] - \mathbb{E}[\log q(\Theta)] \tag{1.7}$$

Do to the improved computational efficiency of these methods, variational inference has been used to train large matrix and tensor factorizations. Salakhutdinov and Mnih use VI to factor a matrix of Netflix movie ratings containing $17,770$ movies, $480,189$ users and over 100 million ratings [100]. Chu and Ghahramani implement a three-mode Tucker tensor factorization and test on a number of different datasets of varying sizes [101]. Several groups have developed methods using VI to simultaneously factor combinations of tensors and matrices where one mode is shared [102, 103, 104]. Also, VI is used to train the matrix factorization that is used to predict drug response mentioned in figure 1.2 [25] and the recent gene expression CP decomposition published by Hore et al. [86].

Our model is also trained using variational approximation but expands upon the previous work by linking the tensor factorization to predictive features for each mode.

Chapter 2

The BaTFLED model

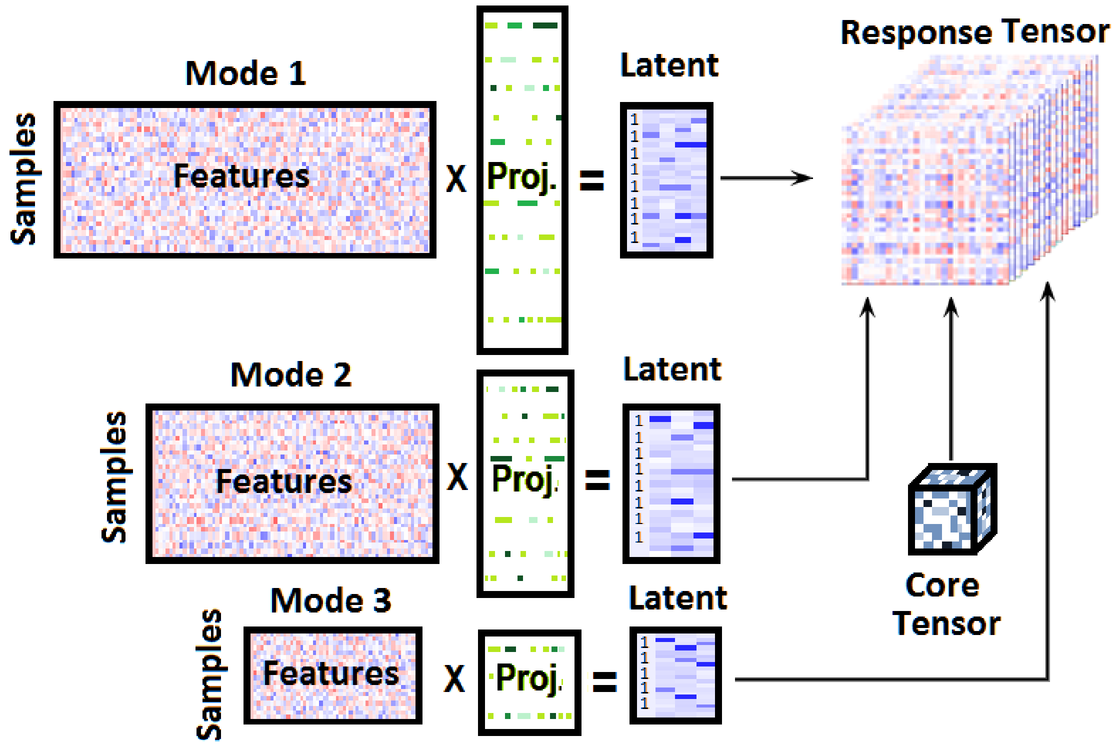


Figure 2.1: Schematic of the model. Input features are mapped through projection matrices to form latent matrices. These are then multiplied with the core tensor to form the response tensor. Parameters for the distributions on the projection matrices and core tensor can be used to encourage sparsity in these elements. Columns of ones added to the latent matrices allow BaTFLED to model lower-order interactions between the latent factors.

In this chapter we present BaTFLED (Bayesian Tensor Factorization Linked to External Data), a new machine learning model capable of predicting responses that vary across three independent dimensions. BaTFLED is formulated as a probabilistic graphic model where distributions are placed on all unknown values (figure 2.2). Estimates for the unknown values are found efficiently using variational Bayesian approximation. This is implemented in an R package which is freely available on CRAN (`https://cran.r-project.org/web/packages/BaTFLED3D/index.html`) and GitHub (`https://github.com/nathanlazar/BaTFLED3D`). There are options to implement both CP and Tucker decompositions, to use predictor features for any or none of the three modes, to encourage sparsity in the projection matrices and core tensor and to model lower-order interactions in the data by adding constant columns to the latent matrices.

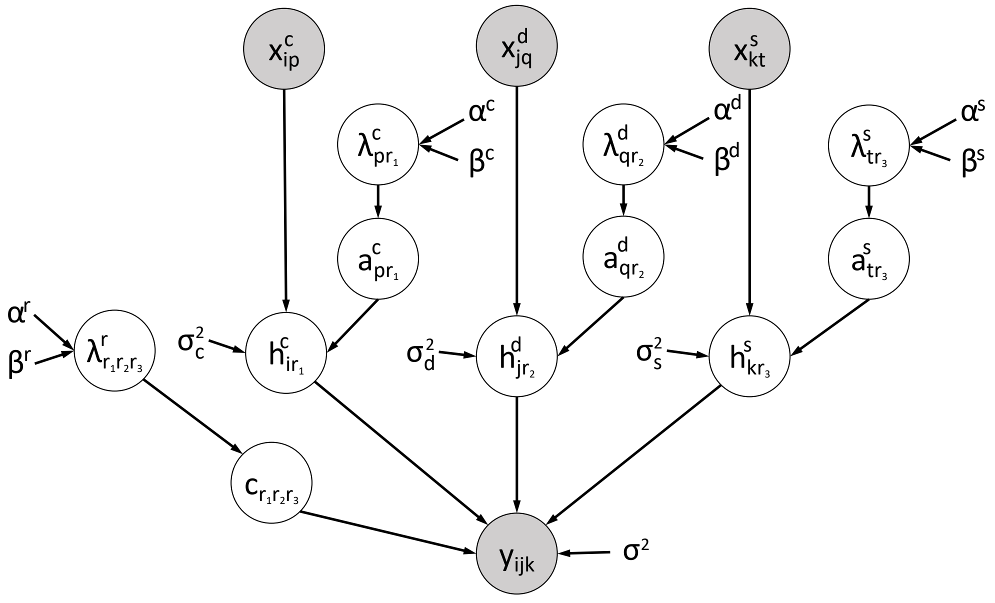


Figure 2.2: The probabilistic graphical model for the BaTFLED. Observed values are in gray, set parameters shown without circles. Plates are omitted for clarity, but can be deduced by subscripts.

2.1 Model structure

The BaTFLED model has four main levels of parameters with parallel elements for each of thee three modes. The first level consists of the input feature data for each mode and is represented by matrices X^c, X^d an X^s (the superscripts are meant to suggest cell lines c, drugs d and doses s). These matrices have one row for each sample (cell line, drug or dose) and one column for each feature associated the samples. For example, in the case of dose-response prediction, the cell lines may be characterized by binary indicator features for different subtypes or expression values for a set of genes while the drugs may have features indicating different classes of drugs, known targets or structural features. Alternatively, the input feature columns may also represent similarity between samples. In this case the columns of the X matrices would also represent samples and the X matrix would be filled with some similarity measure for the pair of samples represented by the row and column. We refer to this as 'kernelizing' the input data and, depending on the use case, this may improve performance as non-linear interactions between features can be captured and this greatly reduces the number of features. Also, in the current implementation of the algorithm, the user can optionally add a column of ones to the input data. This allows the model to learn an intercept for each mode analogous to the intercept in a linear regression. These input matrices are set before training and hence they are not represented by distributions in the model and must be fully observed.

The second level of parameters consists of the projection matrices A^c, A^d and A^s. Each row of the A matrices represents one of the input features for that mode and each column represents a latent feature. The number of latent features for each mode is set by the user. For CP decompositions, this value is the same for all three modes and for Tucker decompositions it can differ between modes. These values are unknown and are estimated by the inference procedure.

The third level of parameters consists of latent matrices H^c, H^d and H^s and a core tensor $\mathbf{C}$ in the case of a Tucker decomposition. The H matrices serve a dual purpose both as low-dimensional representations of the input feature data and as factor matrices for the response tensor. There are rows for each sample and columns for each latent feature. If input feature data (X matrices) are provided, the values in the H matrices are modeled as tight distributions centered at the product of the X and A matrices, otherwise the values are modeled by looser distributions with mean zero. With Tucker decompositions, a core tensor stores values weighting the interactions between the latent features for the different modes. For example, if the dimensions of H^c, H^d and H^s matrices are $I \times R_1$, $J \times R_2$ and $K \times R_3$ respectively, then the core tensor will have dimensions $R_1 \times R_2 \times R_3$ and the

(r_1, r_2, r_3) element will weight the interaction between the r_1 column of H^c, the r_2 column of H^d and the r_3 column of H^s. In the case of a CP decomposition, $R_1 = R_2 = R_3$ and only interactions between corresponding columns of each mode are allowed (i.e. column 1 of H^c, H^d and H^s). The core tensor does not need to be modeled explicitly as these values can be absorbed by the H matrices.

The final layer consists of the response values stored in a three-dimensional tensor. These values are modeled by distributions centered at the three-way product of the H matrices and core tensor. By using distributions and not just fixed values here, BaTFLED is able to handle datasets with missing responses. In the case of a CP decomposition the mean response values are the sum of the outer product of each of the columns of the H matrices.

$$\mu(\mathbf{Y}) = \sum_{r=1}^{R} \mathbf{h}_i^c \circ \mathbf{h}_j^d \circ \mathbf{h}_j^s$$

component-wise

$$\mu(y_{ijk}) = \sum_{r=1}^{R} h_{ir}^c h_{jr}^d h_{kr}^s$$

For Tucker models, the sum is over all combinations of columns of the H matrices weighted by the core values.

$$\mu(\mathbf{Y}) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} c_{r_1 r_2 r_3} \mathbf{h}_i^c \circ \mathbf{h}_j^d \circ \mathbf{h}_j^s$$

component-wise

$$\mu(y_{ijk}) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} c_{r_1 r_2 r_3} h_{ir_1}^c h_{jr_2}^d h_{kr_3}^s$$

In order to model the effect of lower-order interactions, the user can optionally add columns of ones to the H matrices. The corresponding elements of the core tensor serve as marginal intercepts for the responses. For example, if all three H matrices have a constant, we extend the ranges of r_1, r_2 and r_3 to include a zero index and the corresponding element of the core matrix (c_{000}) can learn the a global intercept for the response tensor; a value that is added to all responses. Similarly the c_{100} element is a coefficient for the first latent factor for the first mode and encodes the one-dimensional effects that this latent factor will have on response regardless of the values in the H matrices for modes two and three.

The coefficient c_{110} encodes interactions between the first latent factors of modes one and two (cell lines and drugs) that occur regardless of mode three (dose).

2.1.1 Distributional assumptions

In this section we give details for the distributions used in the BaTFLED model. These are chosen to maintain conjugacy and to allow the user to set hyper-parameters to encourage sparsity in the projection A matrices and the core tensor. The prior parameters for the variances of the A matrices and core tensor are represented by Λ^c, Λ^d, Λ^s and $\boldsymbol{\lambda}^r$ respectively. These have hyper-parameters α and β. Also, we use a binary indicator tensor $\boldsymbol{\Delta}$ to indicate whether each response has been observed. For simplicity we denote all the observed values and set parameters (except the responses) by Ξ.

$$\Xi = \{X^c, X^d, X^s, \sigma^2, \sigma_c^2, \sigma_d^2, \sigma_s^2, \alpha^c, \alpha^d, \alpha^s, \alpha^r, \beta^c, \beta^d, \beta^s, \beta^r, \boldsymbol{\Delta}\} \tag{2.1}$$

and all random variables by Θ

$$\Theta = \{H^c, H^d, H^s, A^c, A^d, A^s, \Lambda^c, \Lambda^d, \Lambda^s, \mathbf{C}, \boldsymbol{\Lambda}^r\} \tag{2.2}$$

Thus the full joint distribution of the the prior parameters $\Lambda^c, \Lambda^d, \Lambda^s, \boldsymbol{\Lambda}^r$, projection matrices A^c, A^d, A^s, latent factor matrices H^c, H^d, H^s, core tensor $\mathbf{C}$, and the responses $\mathbf{Y}$ given the observed data X^c, X^d, X^s and the set parameters σ^2, σ_c^2, σ_d^2, σ_s^2, α^c, α^d, α^s, α^r, β^c, β^d, β^s, β^r, $\boldsymbol{\Delta}$ is $p(\Theta, \mathbf{Y}|\Xi)$. This distribution factors as follows (dependencies on set parameters are omitted for clarity).

$$\begin{aligned} p(\Theta, \mathbf{Y}|\Xi) &= p(\mathbf{Y}, H^c, H^d, H^s, A^c, A^d, A^s, \Lambda^c, \Lambda^d, \Lambda^s, \mathbf{C}, \boldsymbol{\Lambda}^r|X^c, X^d, X^s) \\ &= p(\mathbf{Y}|\mathbf{C}, H^c, H^d, H^s)p(H^c|A^c, X^c)p(H^d|A^d, X^d)p(H^s|A^s, X^s) \\ &\quad p(\mathbf{C}|\boldsymbol{\Lambda}^r)p(A^c|\Lambda^c)p(A^d|\Lambda^d)p(A^s|\Lambda^s)p(\Lambda^c)p(\Lambda^d)p(\Lambda^s)p(\boldsymbol{\Lambda}^r) \end{aligned} \tag{2.3}$$

The distributional assumptions for each of the factors are

$$p(\mathbf{Y}|\mathbf{C}, H^c, H^d, H^s) = \prod_{i=1}^{I}\prod_{j=1}^{J}\prod_{k=1}^{K}\mathcal{N}(y_{ijk}; \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3} c_{r_1r_2r_3}h^c_{ir_1}h^d_{jr_2}h^s_{kr_3}, \sigma^2)^{\delta_{ijk}} \tag{2.4}$$

$$p(H^c|A^c, X^c) = \prod_{i=1}^{I}\prod_{r_1=1}^{R_1}\mathcal{N}(h^c_{ir_1}; \mathbf{x}^c_i\mathbf{a}^c_{r_1}, \sigma^2_c) \tag{2.5}$$

$$p(H^d|A^d, X^d) = \prod_{j=1}^{J}\prod_{r_2=1}^{R_2}\mathcal{N}(h^d_{jr_2}; \mathbf{x}^d_j\mathbf{a}^d_{r_2}, \sigma^2_d) \tag{2.6}$$

$$p(H^s|A^s, X^s) = \prod_{k=1}^{K}\prod_{r_3=1}^{R_3}\mathcal{N}(h^s_{kr_3}; \mathbf{x}^s_k\mathbf{a}^s_{r_3}, \sigma^2_s) \tag{2.7}$$

$$p(\mathbf{C}|\mathbf{\Lambda}^r) = \prod_{r_1=1}^{R_1}\prod_{r_2=1}^{R_2}\prod_{r_3=1}^{R_3}\mathcal{N}(c_{r_1r_2r_3}; 0, (\lambda^r_{r_1r_2r_3})^{-1}) \tag{2.8}$$

$$p(A^c|\Lambda^c) = \prod_{p=1}^{P}\prod_{r_1=1}^{R_1}\mathcal{N}(a^c_{pr_1}; 0, (\lambda^c_{pr_1})^{-1}) \tag{2.9}$$

$$p(A^d|\Lambda^d) = \prod_{q=1}^{Q}\prod_{r_2=1}^{R_2}\mathcal{N}(a^d_{qr_2}; 0, (\lambda^d_{qr_2})^{-1}) \tag{2.10}$$

$$p(A^s|\Lambda^s) = \prod_{t=1}^{T}\prod_{r_3=1}^{R_3}\mathcal{N}(a^s_{tr_3}; 0, (\lambda^s_{tr_3})^{-1}) \tag{2.11}$$

$$p(\mathbf{\Lambda}^r) = \prod_{r_1=1}^{R_1}\prod_{r_2=1}^{R_2}\prod_{r_3=1}^{R_3}\mathcal{G}(\lambda^r_{r_1r_2r_3}; \alpha^r, \beta^r) \tag{2.12}$$

$$p(\Lambda^c) = \prod_{p=1}^{P}\prod_{r_1=1}^{R_1}\mathcal{G}(\lambda^c_{pr_1}; \alpha^c, \beta^c) \tag{2.13}$$

$$p(\Lambda^d) = \prod_{q=1}^{Q}\prod_{r_2=1}^{R_2}\mathcal{G}(\lambda^d_{qr_2}; \alpha^d, \beta^d) \tag{2.14}$$

$$p(\Lambda^s) = \prod_{t=1}^{T}\prod_{r_3=1}^{R_3}\mathcal{G}(\lambda^s_{tr_3}; \alpha^s, \beta^s) \tag{2.15}$$

$$\delta_{ijk} = \begin{cases} 1 & if\ y_{ijk}\ is\ observed \\ 0 & otherwise \end{cases} \tag{2.16}$$

Where $\mathcal{N}(x;\mu,\sigma^2)$ is a normal distribution with mean μ and variance σ^2 and $\mathcal{G}(x;a,b)$

is a gamma distribution with shape a and scale b. We use a shared variance σ^2 for the response tensor and shared variances σ_c^2, σ_d^2 and σ_s^2 for the elements of the latent matrices H^c, H^d and H^s respectively. The gamma distributions on the inverse variance (precision) λ parameters for the projection matrices A^c, A^d and A^s and the core $\mathbf{C}$ allows the user to encourage sparsity in these parts of the model. By setting the prior shape and scale parameters to extreme values (ex. $\alpha = 10^{-10}$ and $\beta = 10^{10}$) the majority of these precision λ values move toward zero as the model is trained. Since the mean of these distributions is set at zero, pushing the precision to zero for a particular feature minimizes the contribution of that feature to the model. The λ values in the projection A matrices can optionally be shared across rows which encourages the use of the same predictor variables for all latent factors.

2.2 Inference

With the model established above, our goal is to find the posterior distribution of variables in the model given the observations $p(\Theta|\Xi, \mathbf{Y})$ as well as the marginal likelihood or model evidence $p(\mathbf{Y}|\Xi)$. Exact inference of the posterior is not analytically solvable and sampling based approaches like Markov chain Monte Carlo (MCMC) would be computationally prohibitive for a model of this size. Instead we use a variational approximation (or variational Bayes) approach that maximizes a lower bound on the log of the posterior. This ELBO (evidence based lower bound) is found by minimizing the Kullback-Leibler divergence [99] between the true posterior distribution and an approximating q distribution. See section 1.5.2 for more details.

$$\mathcal{L} = \mathbb{E}_{q(\Theta)}[\log p(\mathbf{Y}, \Theta|\Xi)] - \mathbb{E}_{q(\Theta)}[\log q(\Theta)] \tag{2.17}$$

The new q distribution is a joint distribution of the same variables as the p distribution that we assume to be fully factorable. This is known as the 'mean field' approximation.

$$q(\Theta) = q(\mathbf{C})q(H^c)q(H^d)q(H^s)q(A^c)q(A^d)q(A^s)q(\Lambda^c)q(\Lambda^d)q(\Lambda^s)q(\mathbf{\Lambda}^r) \tag{2.18}$$

Closed forms for the q distribution can be derived analytically according to equation 1.5, but they depend on expectations under the q distribution of other variables in the model.

$$\log q(\theta) = \mathbb{E}_{q(\Theta\backslash\theta)}\left[\log p(\Theta|\Xi)\right] \tag{1.5}$$

In order to find the q distribution that best approximates the p distribution, an iterative expectation maximization process is employed.

2.2.1 Update equations

As mentioned in section 1.5.2, the logarithm of the optimal q distribution for a given variable θ can be obtained by finding the expectation of $\log p(\Theta|\Xi)$ with respect to the q distributions over all other variables. Applying this process to the distributions in BaTFLED gives the update equations shown below. Equations for other modes are identical to those shown here for the first mode. All expectations are with respect to the q distributions.

Prior Λ matrices:

$$q(\lambda^c_{pr_1}) = \mathcal{G}\left(\lambda^c_{pr_1}; \alpha^c + \frac{1}{2}, \left(\frac{1}{2}\mathbb{E}[a^c_{pr_1}]^2 + \frac{1}{2}var(a^c_{pr_1}) + \frac{1}{\beta^c}\right)^{-1}\right) \tag{2.19}$$

$$q(\lambda^r_{r_1r_2r_3}) = \mathcal{G}\left(\lambda^r_{r_1r_2r_3}; \alpha^r + \frac{1}{2}, \left(\frac{1}{2}\mathbb{E}[c_{r_1r_2r_3}]^2 + \frac{1}{2}var(c_{r_1r_2r_3}) + \frac{1}{\beta^r}\right)^{-1}\right) \tag{2.20}$$

Projection A matrices

$$q(\mathbf{a}^c_{r_1}) = \mathcal{N}\left(\mathbf{a}^c_{r_1}; \boldsymbol{\mu}(\mathbf{a}^c_{r_1}), \Sigma(\mathbf{a}^c_{r_1})\right) \tag{2.21}$$

with

$$\begin{aligned}
\mu(\mathbf{a}^c_{r_1}) &= \left(\frac{1}{\sigma_c^2}\mathbb{E}[\mathbf{h}^c_{r_1}]^T X^c\right)\Sigma(\mathbf{a}^c_{r_1}) \\
\Sigma(\mathbf{a}^c_{r_1}) &= \left(\frac{1}{\sigma_c^2}(X^c)^T X^c + \mathbb{E}[\boldsymbol{\lambda}^c_{r_1}]I\right)^{-1}
\end{aligned}$$

Latent H matrices:

$$q(h^c_{ir}) = \mathcal{N}(h^c_{ir}; \mu(h^c_{ir}), \Sigma(h^c_{ir})) \tag{2.22}$$

with

$$\mu(h^c_{ir_1}) = \Sigma(h^c_{ir_1}) \Bigg(\frac{1}{\sigma^2} \sum_{j=1}^{J} \sum_{k=1}^{K} \delta_{ijk} \Bigg[y_{ijk} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathbb{E}[c_{r_1 r_2 r_3}] \mathbb{E}[h^d_{jr_2}] \mathbb{E}[h^s_{kr_3}]$$

$$- \sum_{r_1^* \neq r_1} \sum_{r_2}^{R_2} \sum_{r_3}^{R_3} \mathbb{E}[c_{r_1 r_2 r_3}] \mathbb{E}[h^c_{ir_1^*}] \Bigg[\sum_{r_2^* \neq r_2} \sum_{r_3^* \neq r_3} \mathbb{E}[c_{r_1^* r_2^* r_3^*}] \mathbb{E}[h^d_{jr_2}] \mathbb{E}[h^d_{jr_2^*}] \mathbb{E}[h^s_{kr_3}] \mathbb{E}[h^s_{kr_3^*}]$$

$$+ \sum_{r_2^* \neq r_2} \mathbb{E}[c_{r_1^* r_2^* r_3}] \mathbb{E}[h^d_{jr_2}] \mathbb{E}[h^d_{jr_2^*}] \left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3}) \right)$$

$$+ \sum_{r_3^* \neq r_3} \mathbb{E}[c_{r_1^* r_2 r_3^*}] \left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2}) \right) \mathbb{E}[h^s_{kr_3}] \mathbb{E}[h^s_{kr_3^*}]$$

$$+ \mathbb{E}[c_{r_1^* r_2 r_3}] \left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2}) \right) \left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3}) \right) \Bigg] \Bigg]$$

$$+ \frac{1}{\sigma_c^2} \mathbf{x}_i^c \mathbf{a}_{r_1}^c \Bigg)$$

and

$$\Sigma(h^c_{ir}) = \Bigg(\frac{1}{\sigma^2} \sum_{j=1}^{J} \sum_{k=1}^{K} \delta_{ijk} \sum_{r_2=1}^{R_2} \sum_{r_3}^{R_3} \Bigg[$$

$$\sum_{r_2^* \neq r_2} \sum_{r_3^* \neq r_3} \mathbb{E}[c_{r_1 r_2 r_3}] \mathbb{E}[c_{r_1 r_2^* r_3^*}] \mathbb{E}[h^d_{jr_2}] \mathbb{E}[h^d_{jr_2^*}] \mathbb{E}[h^s_{kr_3}] \mathbb{E}[h^s_{kr_3^*}]$$

$$+ \sum_{r_2^* \neq r_2} \mathbb{E}[c_{r_1 r_2 r_3}] \mathbb{E}[c_{r_1 r_2^* r_3}] \mathbb{E}[h^d_{jr_2}] \mathbb{E}[h^d_{jr_2^*}] \left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3}) \right)$$

$$+ \sum_{r_3^* \neq r_3} \mathbb{E}[c_{r_1 r_2 r_3}] \mathbb{E}[c_{r_1 r_2 r_3^*}] \left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2}) \right) \mathbb{E}[h^s_{kr_3}] \mathbb{E}[h^s_{kr_3^*}]$$

$$+ \left(\mathbb{E}[c_{r_1 r_2 r_3}]^2 + Var(c_{r_1 r_2 r_3}) \right) \left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2}) \right) \left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3}) \right) \Bigg]$$

$$+ \frac{1}{\sigma_c^2} \Bigg)^{-1}$$

Core **C** tensor

$$q(c_{r_1 r_2 r_3}) = \mathcal{N} \left(c_{r_1 r_2 r_3}; \mu(c_{r_1 r_2 r_3}), \Sigma(c_{r_1 r_2 r_3}) \right) \tag{2.23}$$

with

$$\begin{aligned}
\mu(c_{r_1r_2r_3}) = \frac{1}{\sigma^2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\Bigg[& \delta_{ijk}y_{ijk}\mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}] \\
& + \mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}]\sum_{r_1^*\neq r_1}\sum_{r_2^*\neq r_2}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1^*r_2^*r_3^*}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^d_{jr_2^*}]\mathbb{E}[h^s_{kr_3^*}] \\
& + \left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}]\sum_{r_2^*\neq r_2}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1r_2^*r_3^*}]\mathbb{E}[h^d_{jr_2^*}]\mathbb{E}[h^s_{kr_3^*}] \\
& + \mathbb{E}[h^c_{ir_1}]\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\mathbb{E}[h^s_{kr_3}]\sum_{r_1^*\neq r_1}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1^*r_2r_3^*}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^s_{kr_3^*}] \\
& + \mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_1^*\neq r_1}\sum_{r_2^*\neq r_2}\mathbb{E}[c_{r_1^*r_2^*r_3}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^d_{jr_2^*}] \\
& + \left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\mathbb{E}[h^s_{kr_3}]\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1r_2r_3^*}]\mathbb{E}[h^s_{kr_3^*}] \\
& + \left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\mathbb{E}[h^d_{jr_2}]\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_2^*\neq r_2}\mathbb{E}[c_{r_1r_2^*r_3}]\mathbb{E}[h^d_{jr_2^*}] \\
& + \mathbb{E}[h^c_{ir_1}]\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_1^*\neq r_1}\mathbb{E}[c_{r_1^*r_2r_3}]\mathbb{E}[h^c_{ir_1^*}]\Bigg]
\end{aligned}$$

and

$$\begin{aligned}
\Sigma(c_{r_1r_2r_3}) = \Bigg(\frac{1}{\sigma^2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right) \\
\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right) + \mathbb{E}[\lambda^r_{r_1r_2r_3}]\Bigg)^{-1}
\end{aligned}$$

2.2.2 The ELBO

Using equations 2.3 and 2.18, the evidence-based lower-bound (ELBO) can be written in the form below. Expectations are over all variables within each expression with respect to the q distributions.

$$\begin{aligned}\mathcal{L} = &\ \mathbb{E}[\log p(\mathbf{Y}|\mathbf{C}, H^c, H^d, H^s)] + \mathbb{E}[\log p(\mathbf{C}|\boldsymbol{\Lambda}^r)] \\ &+ \mathbb{E}[\log p(H^c|A^c, X^c)] + \mathbb{E}[\log p(H^d|A^d, X^d)] + \mathbb{E}[\log p(H^s|A^s, X^s)] \\ &+ \mathbb{E}[\log p(A^c|\Lambda^c)] + \mathbb{E}[\log p(A^d|\Lambda^d)] + \mathbb{E}[\log p(A^s|\Lambda^s)] \\ &+ \mathbb{E}[\log p(\boldsymbol{\Lambda}^r)] + \mathbb{E}[\log p(\Lambda^c)] + \mathbb{E}[\log p(\Lambda^d)] + \mathbb{E}[\log p(\Lambda^s)] \\ &- \mathbb{E}[\log q(\mathbf{C})] - \mathbb{E}[\log q(H^c)] - \mathbb{E}[\log q(H^d)] - \mathbb{E}[\log q(H^s)] \\ &- \mathbb{E}[\log q(A^c)] - \mathbb{E}[\log q(A^d)] - \mathbb{E}[\log q(A^s)] \\ &- \mathbb{E}[\log q(\boldsymbol{\Lambda}^r)] - \mathbb{E}[\log q(\Lambda^c)] - \mathbb{E}[\log q(\Lambda^d)] - \mathbb{E}[\log q(\Lambda^s)]\end{aligned} \tag{2.24}$$

The expectations of the p distributions over the q distributions are given below for each term.

$$\mathbb{E}[\log p(\Lambda^c)] = \sum_{p=1}^{P}\sum_{r_1=1}^{R_1}\left[(\alpha^c - 1)\mathbb{E}[\log(\lambda^c_{pr_1})] - \frac{1}{\beta^c}\mathbb{E}[\lambda^c_{pr_1}] - \alpha^c \log(\beta^c) - \log(\Gamma(\alpha^c))\right]$$

$$\begin{aligned}\mathbb{E}[\log p(A^c|\Lambda^c)] = &-\frac{PR_1}{2}\log(2\pi) + \frac{1}{2}\sum_{p=1}^{P}\sum_{r_1=1}^{R_1}\mathbb{E}[\log(\lambda^c_{pr_1})] \\ &- \frac{1}{2}\sum_{p=1}^{P}\sum_{r_1=1}^{R_1}\mathbb{E}[\lambda^c_{pr_1}](\mathbb{E}[a^c_{pr_1}]^2 + var(a^c_{pr_1}))\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\log p(H^c|A^c, X^c)] = &-\frac{IR_1}{2}\log(2\pi\sigma_c^2) - \frac{1}{2\sigma_c^2}\sum_{i=1}^{I}\sum_{r_1=1}^{R_1}\left(\mathbb{E}[h^c_{ir_1}]^2 + var(h^c_{ir_1})\right) \\ &+ \frac{1}{\sigma_c^2}\sum_{i=1}^{I}\sum_{r_1=1}^{R_1}\mathbb{E}[h^c_{ir_1}]\mathbf{x}^c_i\mathbb{E}[\mathbf{a}^c_{r_1}] \\ &- \frac{1}{2\sigma_c^2}\sum_{i=1}^{I}\sum_{r_1=1}^{R_1}\sum_{p_1=1}^{P}\sum_{p_2=1}^{P} x^c_{ip_1}x^c_{ip_2}\left(\mathbb{E}[a_{p_1r_1}]\mathbb{E}[a_{p_2r_1}] + Cov(\mathbf{a}_{r_1})_{p_1p_2}\right)\end{aligned}$$

$$\mathbb{E}[\log p(\mathbf{\Lambda}^r)] = \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3} \Big[(\alpha^r - 1)\mathbb{E}[\log(\lambda^r_{r_1r_2r_3})] - \frac{1}{\beta^r}\mathbb{E}[\lambda^r_{r_1r_2r_3}] - \alpha^r\log(\beta^r) - \log(\Gamma(\alpha^r))\Big]$$

$$\mathbb{E}[\log p(\mathbf{C}|\mathbf{\Lambda}^c)] = -\frac{R_1R_2R_3}{2}\log(2\pi) + \frac{1}{2}\sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\mathbb{E}[\log(\lambda^r_{r_1r_2r_3})] - \frac{1}{2}\sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\mathbb{E}[\lambda^r_{r_1r_2r_3}](\mathbb{E}[c_{r_1r_2r_3}]^2 + var(c_{r_1r_2r_3}))$$

$$\begin{aligned}
\mathbb{E}[\log p(\mathbf{Y}|\mathbf{C}, H^c, H^d, H^s)] &= -\frac{1}{2}\log(2\pi\sigma^2)\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk} \\
&-\frac{1}{2\sigma^2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\Big[y^2_{ijk} - 2y_{ijk}\sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\mathbb{E}[c_{r_1r_2r_3}]\mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}]\Big] \\
&-\frac{1}{2\sigma^2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\mathbb{E}[c_{r_1r_2r_3}]\Big[\\
&\mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}]\sum_{r_1^*\neq r_1}\sum_{r_2^*\neq r_2}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1^*r_2^*r_3^*}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^d_{jr_2^*}]\mathbb{E}[h^s_{kr_3^*}] \\
&+\left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\mathbb{E}[h^d_{jr_2}]\mathbb{E}[h^s_{kr_3}]\sum_{r_2^*\neq r_2}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1r_2^*r_3^*}]\mathbb{E}[h^d_{jr_2^*}]\mathbb{E}[h^s_{kr_3^*}] \\
&+\mathbb{E}[h^c_{ir_1}]\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\mathbb{E}[h^s_{kr_3}]\sum_{r_1^*\neq r_1}\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1^*r_2r_3^*}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^s_{kr_3^*}] \\
&+\mathbb{E}[h^c_{ir_1}]\mathbb{E}[h^d_{jr_2}]\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_1^*\neq r_1}\sum_{r_2^*\neq r_2}\mathbb{E}[c_{r_1^*r_2^*r_3}]\mathbb{E}[h^c_{ir_1^*}]\mathbb{E}[h^d_{jr_2^*}] \\
&+\left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\mathbb{E}[h^s_{kr_3}]\sum_{r_3^*\neq r_3}\mathbb{E}[c_{r_1r_2r_3^*}]\mathbb{E}[h^s_{kr_3^*}] \\
&+\left(\mathbb{E}[h^c_{ir_1}]^2 + Var(h^c_{ir_1})\right)\mathbb{E}[h^d_{jr_2}]\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_2^*\neq r_2}\mathbb{E}[c_{r_1r_2^*r_3}]\mathbb{E}[h^d_{jr_2^*}] \\
&+\mathbb{E}[h^c_{ir_1}]\left(\mathbb{E}[h^d_{jr_2}]^2 + Var(h^d_{jr_2})\right)\left(\mathbb{E}[h^s_{kr_3}]^2 + Var(h^s_{kr_3})\right)\sum_{r_1^*\neq r_1}\mathbb{E}[c_{r_1^*r_2r_3}]\mathbb{E}[h^c_{ir_1^*}]\Big]
\end{aligned}$$

The terms in the lower bound coming from $\mathbb{E}_{q(\Theta)}[\log q(\Theta)]$ are negative entropies.

$$
\begin{aligned}
\mathbb{E}_{q(\Lambda^c)}[\log q(\Lambda^c)] &= \sum_{i=1}^{I}\sum_{r_1=1}^{R_1}\Big[-a(\lambda^c_{ir_1}) - \log(b(\lambda^c_{ir_1})) \\
&\quad - \log(\Gamma(a(\lambda^c ir_1))) - (1-a(\lambda^c ir_1))\psi(a(\lambda^c ir_1))\Big] \\
\mathbb{E}_{q(\boldsymbol{\Lambda}^r)}[\log q(\boldsymbol{\Lambda}^r)] &= \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\Big[-a(\lambda_{r_1r_2r_3}) - \log(b(\lambda_{r_1r_2r_3})) \\
&\quad - \log(\Gamma(a(\lambda r_1r_2r_3))) - (1-a(\lambda r_1r_2r_3))\psi(a(\lambda r_1r_2r_3))\Big] \\
\mathbb{E}_{q(A^c)}[\log q(A^c)] &= \sum_{p=1}^{P}\sum_{r_1=1}^{R_1}\Big[-\frac{1}{2}(\log(2\pi)+1) - \frac{1}{2}\log\sigma^2(a^c_{pr_1})\Big] \\
\mathbb{E}_{q(H^c)}[\log q(H^c)] &= \sum_{i=1}^{I}\sum_{r_1=1}^{R_1}\Big[-\frac{1}{2}(\log(2\pi)+1) - \frac{1}{2}\log\sigma^2(h^c_{ir_1})\Big] \\
\mathbb{E}_{q(\mathbf{C})}[\log q(\mathbf{C})] &= \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\Big[-\frac{1}{2}(\log(2\pi)+1) - \frac{1}{2}\log\sigma^2(c_{r_1r_2r_3})\Big]
\end{aligned}
\tag{2.25}
$$

Where $\Gamma(z)$ is the gamma function, ψ is the digamma function, $a(\lambda)$ and $b(\lambda)$ are the shape and scale parameters for the gamma distributions and $\sigma^2(\theta)$ is the variance for the parameter θ.

2.3 Implementation

BaTFLED is implemented in R and the package ('BaTFLED3D') is freely available on GitHub and CRAN. Running the algorithm consists of thee major steps. First, the user must form the response and predictor data into a tensor and input matrices which are stored in one training object. Additional objects of this type can be used to store test input matrices and response tensors. Next, the user creates a training data object consisting of the projection matrices, the latent matrices, and the core tensor. Parameters are provided to specify the type of factorization to be used (CP or Tucker), the dimensions of the latent space, the meta-parameters for the sparsity priors, the variances for the latent matrices and response tensor and other options. This object is then initialized is with random values. Finally the model is trained for some number of iterations. During training, the user can monitor the ELBO, plot predictions and display images of the projection and latent matrices. Once trained the user can examine the performance in terms of the normalized root mean squared error, explained variance, Pearson correlation and Spearman correlation.

2.3.1 Package design

The main elements of the BaTFLED3D package are the objects for storing training and test data (`input_data`), the training model objects created with `mk_model()`, and the training and test functions (`train()` and `test()`). There are separate versions of the training model objects, training functions and test functions for CP and Tucker decompositions which are used internally. The data and model objects are stored as 'R6' objects which allows them to be updated in place. This greatly reduces the memory requirements and increases the speed of training. These objects are treated like environments and components are accessed using the \$ operator. All objects and functions are packaged with full documentation and test examples which can be accessed though the package manual or on the command line with the `?` operator.

2.3.2 Parameters

As BaTFLED3D is meant to be a general use package, there are a number of parameters that the user can change depending on the use case. The functions `get_data_params` and `get_model_params` allow the user to specify options which are used in generating test data or training models respectively. This is accomplished by passing a character vector of parameter names followed by `=<value>` to these functions, or by assigning values to the parameter object after generating it with default options. Some essential options are:

- `decomp`: Use either a CP or Tucker decomposition.
- `R`: Specify the size of the latent space in CP models.
- `R1`, `R2` and `R3`: Specify the size of the core for Tucker models.
- `A1.intercept`, `A2.intercept` and `A3.intercept`: Add a column of ones to the input matrices.

- `H1.intercept`, `H2.intercept` and `H3.intercept`: Add a column of ones to the latent matrices.
- `m1.alpha`, `m2.alpha`, `m3.alpha`, `m1.beta`, `m2.beta` and `m3.beta`: Prior parameters for the projection matrices.
- `core.3D.alpha`, `core.3D.beta`: Prior parameters for the core (with Tucker decompositions).
- `m1.sigma2`, `m2.sigma2` and `m3.sigma2`: Variance parameters for the latent matrices.
- `sigma2`: Variance parameter for the responses.

2.3.3 Example usage

This section walks through the use of BaTFLED on a small simulated dataset.

First we check for installation of necessary packages, install them if necessary and load them.

```
library(BaTFLED3D)
# Set up backend for parallel execution}
cores <- 1
if(.Platform$OS.type == "windows") {
  clust <- parallel::makeCluster(cores)
  doParallel::registerDoParallel(clust)
} else doMC::registerDoMC(cores)
```

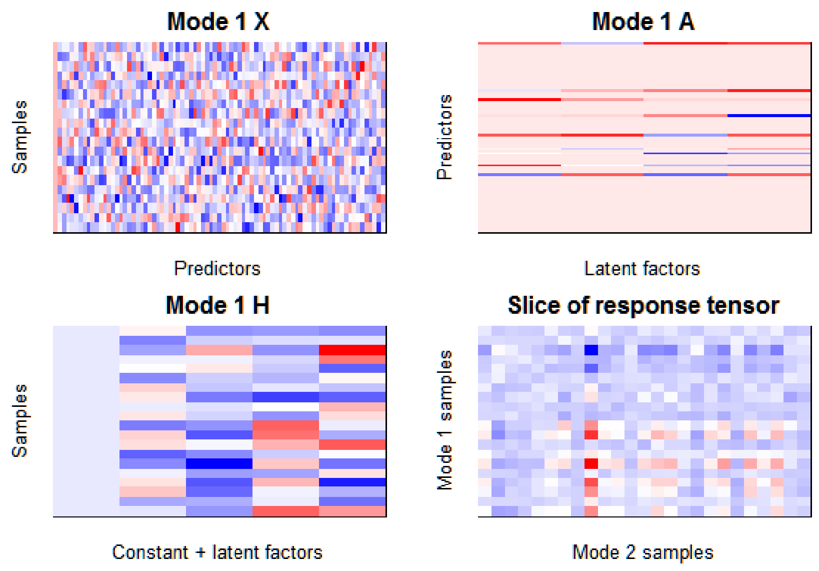
Generating a simulated dataset

Next we generate simulated data according to the assumed generative model which we will try to reconstruct using BaTFLED. Many aspects of model building can be adjusted (use `?get_data_params` to see them).

```
args <- list('decomp=Tucker',  # Either CP or Tucker factorization
  'm1.rows=20', 'm1.cols=80',  # Dimensions of input matrices
  'm2.rows=25', 'm2.cols=100',
  'm3.rows=5', 'm3.cols=0',
  'A1.intercept=T',            # Add intercepts to projection matrices
  'A2.intercept=T',
  'A3.intercept=F',
  'H1.intercept=T',            # Add intercepts to latent matrices
  'H2.intercept=T',
  'H3.intercept=T',
  'm1.true=8',                 # Number of predictors affecting output
  'm2.true=10',
  'm3.true=0',
  'R1=4', 'R2=4', 'R3=4',      # Core dimension
  'core.spar=0.1',             # Amount of sparsity in the core [0-1]
                               # 1 = no sparsity
  'noise.sd=0.2'               # Gaussian noise added to responses
)
data.params <- get_data_params(args)
toy <- mk_toy(data.params)
```

The `toy` list contains the input matrices (`mode1.X`, `mode2.X` and `mode3.X`), the sparse projection matrices (`mode1.A`, `mode2.A` and `mode3.A`), the latent matrices (`mode1.H`, `mode2.H` and `mode3.H`), the core tensor (`core` if a Tucker decomposition is used) and the response tensor (`resp`). The matrices and slices of the tensors can be view using `im_mat()`. Note that there are normally some patterns visible in the response tensor.

```
par(mfrow=c(2,2), mar=c(2,2,2,2), mgp=c(1,1,1))
im_mat(toy$mode1.X, main='Mode 1 X', ylab='Samples',
       xlab='Predictors')
im_mat(toy$mode1.A, main='Mode 1 A', ylab='Predictors',
       xlab='Latent factors')
im_mat(toy$mode1.H, main='Mode 1 H', ylab='Samples',
       xlab='Constant + latent factors')
im_mat(toy$resp[,,1], main='Slice of response tensor',
       xlab='Mode 2 samples', ylab='Mode 1 samples')
```



Training a BaTFLED model

Now that we've built a model, we use BaTFLED to learn the values in the projection and latent matrices and the core tensor given just the input data and responses. We separate two of the 'samples' from modes 1 and 2 to be used as test data and form `input_data` objects. We also set a percentage of training responses to 'NA' to be used for 'warm start' testing. If the X matrices contain constants they are removed since these is optionally added during training.

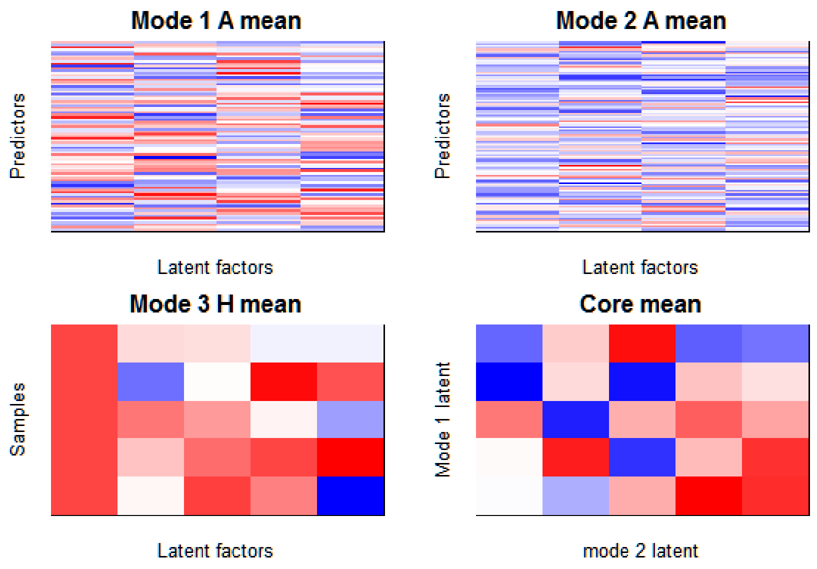
```
train.data <- input_data$new(
  mode1.X=toy$mode1.X[-c(1,2),!grepl('const', colnames(toy$mode1.X))],
  mode2.X=toy$mode2.X[-c(1,2),!grepl('const', colnames(toy$mode2.X))],
  mode3.X=toy$mode3.X[,!grepl('const', colnames(toy$mode3.X))],
  resp=toy$resp[-c(1,2),-c(1,2),])

# Remove 'warm.per' percent of responses to use as 'warm' test data
warm.per <- 0.05
warm.idx <- sample(1:prod(dim(train.data$resp)),
                   prod(dim(train.data$resp))*warm.per)
warm.resp <- train.data$resp[sort(warm.idx)]
train.data$resp[warm.idx] <- NA
train.data$delta[warm.idx] <- 0

m1.test.data <- input_data$new(
  mode1.X=toy$mode1.X[c(1,2), !grepl('const', colnames(toy$mode1.X))],
  mode2.X=toy$mode2.X[-c(1,2), !grepl('const', colnames(toy$mode2.X))],
  mode3.X=toy$mode3.X[, !grepl('const', colnames(toy$mode3.X))],
  resp=toy$resp[c(1,2),-c(1,2),])
m2.test.data <- input_data$new(
  mode1.X=toy$mode1.X[-c(1,2), !grepl('const', colnames(toy$mode1.X))],
  mode2.X=toy$mode2.X[c(1,2), !grepl('const', colnames(toy$mode2.X))],
  mode3.X=toy$mode3.X[, !grepl('const', colnames(toy$mode3.X))],
  resp=toy$resp[-c(1,2),c(1,2),])
m1m2.test.data <- input_data$new(
  mode1.X=toy$mode1.X[c(1,2), !grepl('const', colnames(toy$mode1.X))],
  mode2.X=toy$mode2.X[c(1,2), !grepl('const', colnames(toy$mode2.X))],
  mode3.X=toy$mode3.X[, !grepl('const', colnames(toy$mode3.X))],
  resp=toy$resp[c(1,2),c(1,2),])
```

To train BaTFLED we create and initialize a model object. Again there are a lot of parameters to specify, a subset of which are indicated below. To see all available parameters, their descriptions and defaults run `?get_model_params`.

```
args <- list('decomp=Tucker', 'row.share=T',
             'A1.intercept=T', 'A2.intercept=T', 'A3.intercept=F',
             'H1.intercept=T', 'H2.intercept=T', 'H3.intercept=T',
             'plot=F', 'verbose=F',
             'R1=4', 'R2=4', 'R3=4',
             'sigma2=auto',
             'm1.alpha=1e-10', 'm2.alpha=1e-10', 'm3.alpha=1',
             'm1.beta=1e10', 'm2.beta=1e10', 'm3.beta=1',
             'core.0D.alpha=1', 'core.0D.beta=1',
             'core.1D.alpha=1e-4', 'core.1D.beta=1e4',
             'core.2D.alpha=1e-8', 'core.2D.beta=1e8',
             'core.3D.alpha=1e-12', 'core.3D.beta=1e12',
             'parallel=T', 'cores=1', 'lower.bnd=T',
             'update.order=c(3,1,2)', 'show.mode=c(1,2,3)')
model.params <- get_model_params(args)
model <- mk_model(train.data, model.params)
model$rand_init(model.params)
```

The `model` is an R6 object containing A mean matrices (ex.`model$mode1.A.mean`), A covariance arrays (ex. `model$mode1.A.cov`), H mean and variance matrices (ex. `model$mode1.H.mean` and `model$mode1.H.var`), arrays for the core mean and variance (`model$core.mean` and `model$core.var`), an array of predicted responses (`model$resp`) and parameters for the prior distributions. Initially these are random values sampled from Gaussian distributions.

```
par(mfrow=c(2,2), mar=c(2,2,2,2), mgp=c(1,0,5))
im_mat(model$mode1.A.mean, main='Mode 1 A mean',
       xlab='Latent factors', ylab='Predictors')
im_mat(mode$mode2.A.mean, main='Mode 2 A mean',
       xlab='Latent factors', ylab='Predictors')
im_mat(model$mode3.H.mean, main='Mode 3 H mean',
       xlab='Latent factors', ylab='Samples')
if(model.params$decomp=='Tucker')
  im_mat(model$core.mean[,,1], main='Core mean',
         xlab='mode 2 latent', ylab='Mode 1 latent')
```



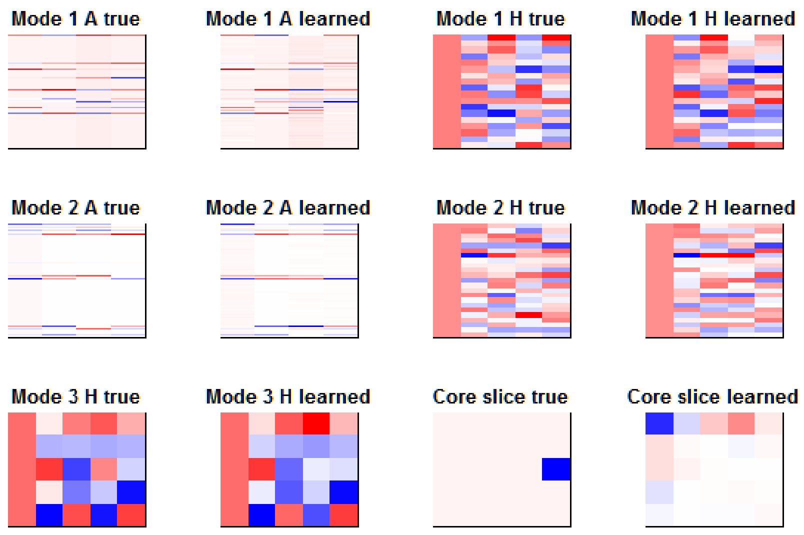
Now we are ready to train the BaTFLED model. If `params$plot=TRUE` the predictions and some of the learned matrices will be displayed during training. You can adjust which A and H matrices are displayed by changing `params$show.mode`. The observed response values for the 'warm' data are set to zero, so they appear along a vertical line in the first plot. The number of cores can be adjusted depending on how many processors are available and the number of iterations can also be adjusted by changing `reps`. With 50 iterations a Tucker decomposition of this size can take $10 - 15$ minutes.

```r
reps <- 50
test.results <- numeric(0)
trained <- model$clone()
train(d=train.data, m=trained, new.iter=reps, params=model.params)

# Get cold results
test.results <- test_results(m=trained, d=train.data,
  test.results=test.results, warm.resp=warm.resp,
  test.m1=m1.test.data, test.m2=m2.test.data,
  test.m1m2=m1m2.test.data)

if(.Platform$OS.type == "windows") parallel::stopCluster(clust)
```

Exploring the resulting model

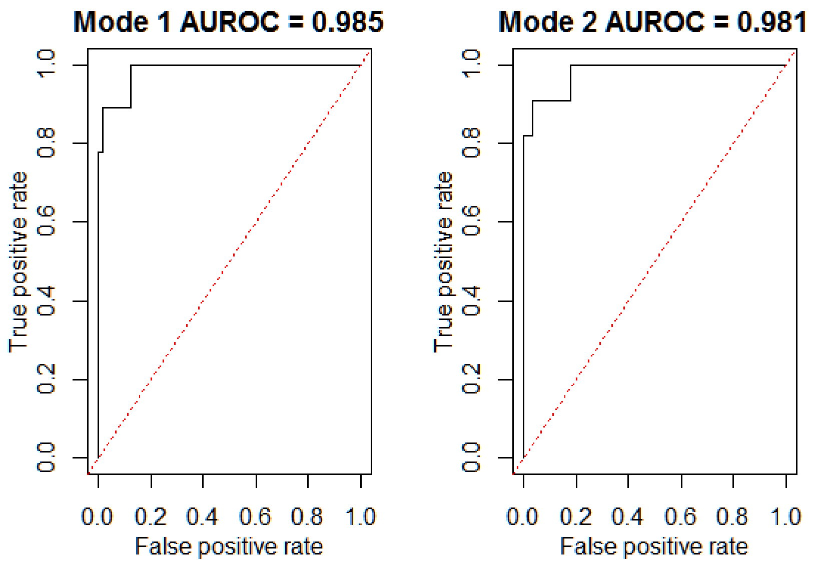
Now that the model has been trained, we can compare the learned matrices to the generating toy model. Since the latent factors (columns of the A and H matrices) can be permuted or scaled without changing predictions, the function `im_2_mat()` scales and reorders columns of the two matrices and tries to match the latent factors. The new ordering for the second matrix is returned (with negatives indicating negative scaling) so that it can be used to order the core.

```
par(mfrow=c(3,4), mar=c(2,2,2,2))
im_2_mat(toy$mode1.A, trained$mode1.A.mean,
  main1='Mode 1 A true', main2='Mode 1 A learned')
m1.H.order <- im_2_mat(toy$mode1.H[-(1:2),], trained$mode1.H.mean,
  main1='Mode 1 H true', main2='Mode 1 H learned')
im_2_mat(toy$mode2.A, trained$mode2.A.mean,
  main1='Mode 2 A true', main2='Mode 2 A learned')
m2.H.order <- im_2_mat(toy$mode2.H[-(1:2),], trained$mode2.H.mean,
  main1='Mode 2 H true', main2='Mode 2 H learned')
m3.H.order <- im_2_mat(toy$mode3.H, trained$mode3.H.mean,
  main1='Mode 3 H true', main2='Mode 3 H learned')
if(model.params$decomp=='Tucker') {
  core.reorder <- trained$core.mean[abs(m1.H.order),
    abs(m2.H.order), abs(m3.H.order)]
  core.reorder <- core.reorder * outer(sign(m1.H.order),
    outer(sign(m2.H.order), sign(m3.H.order)))
  im_2_mat(toy$core[,,1], core.reorder[,,1], sort=F,
    center=T, scale='all', main1='Core slice true',
    main2='Core slice learned')
```



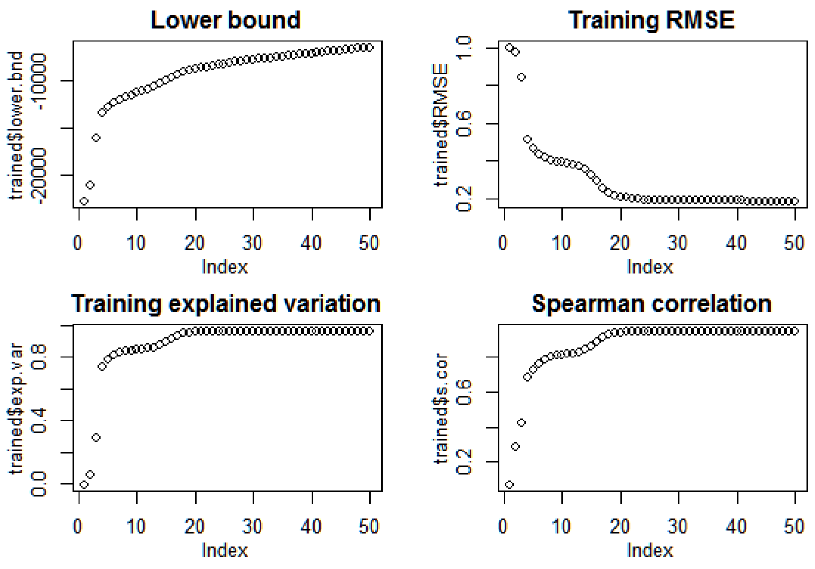
To quantify how well BaTFLED has chosen predictors we can generate receiver operating characteristic (ROC) curves for each of the modes with predictors. Predictors are 'chosen' if the sum of squared values in the corresponding row of the scaled A matrix is large.

```
par(mfrow=c(1,2), mar=c(3,3,2,2), mgp=c(2,1,0))
plot_roc(toy$mode1.A, trained$mode1.A.mean, main='Mode 1')
plot_roc(toy$mode2.A, trained$mode2.A.mean, main='Mode 2')
```



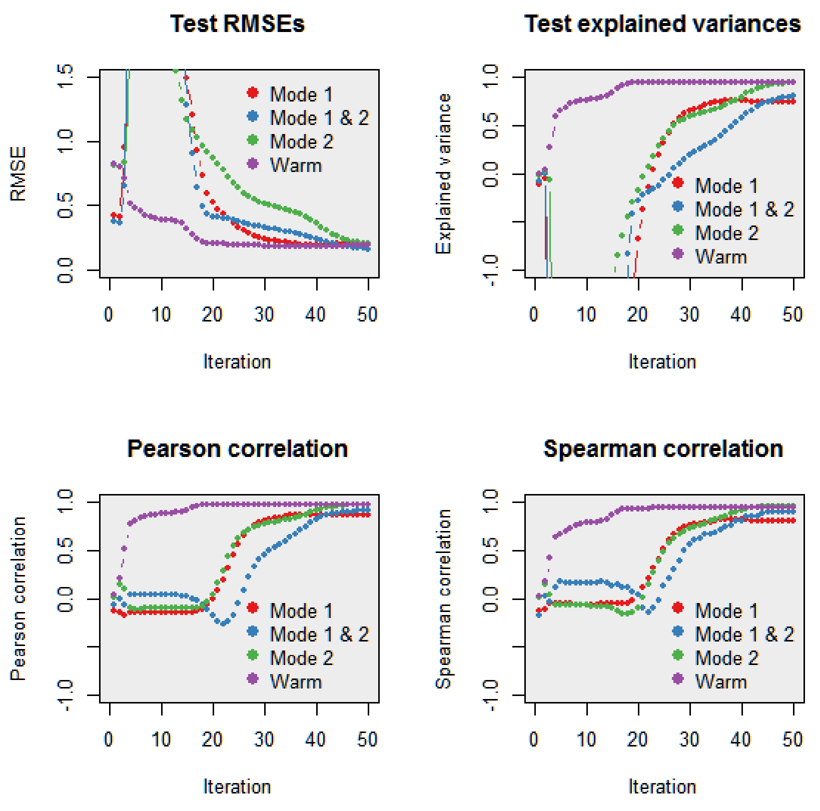
BaTFLED optionally stores values of the lower bound (`lower.bnd`), normalized root mean squared error (`RMSE`), explained variation (`exp.var`), Pearson correlation (`p.cor`) and Spearman correlation (`s.cor`) for training data.

```
par(mfrow=c(2,2), mar=c(3,3,2,2), mgp=c(2,1,0))
if(model.params$lower.bnd)
  plot(trained$lower.bnd, main='Lower bound')
if(model.params$RMSE)
  plot(trained$RMSE, main='Training RMSE')
if(model.params$exp.var)
  plot(trained$exp.var, main='Training explained variation')
if(model.params$cor)
  plot(trained$s.cor, main='Spearman correlation')
```



Also the test.results object in the training loop has stored results for the test data which can be plotted with the functions below.

```
par(mfrow=c(2,2))
plot_test_RMSE(test.results, main="Test RMSEs")
plot_test_exp_var(test.results, main="Test explained variances")
plot_test_cor(test.results, main="Pearson correlation")
plot_test_cor(test.results, main="Spearman correlation",
  method='spearman')}
```



2.3.4 Computational performance

Due to training with variational inference, fairly large models can be trained in a reasonable amount of time. Run times depend, of course, on the size of the input data, the type of decomposition, the number of iterations, and crucially, the size of the latent space. CP models run in series and are generally faster than Tucker models when run with the same number of cores, but some update operations in Tucker models can be done in parallel, so with more processors, Tucker models can be run faster. Additionally, for CP models, the latent dimension `R` may need to be much larger to model the same data. Finally, computing the ELBO is somewhat expensive, so the user can save some run time by setting the parameter `lower.bnd=FALSE`. Run-time performance for specific models will be given in the next two sections.

Chapter 3

Experiments with simulated data

In this chapter we explore the performance of the BaTFLED model on simulated datasets where the true predictive features and interactions are known. This gives insights into how the algorithm performs compared to baseline methods under various sub-optimal conditions. First we explain the performance measures used for assessment and the baseline methods used for comparison. Then we explain the different prediction and variable selection tasks and finally we show results for several different experiments. In particular, we explore the algorithm's performance when the data does or does not contain higher-order relationships, when the responses have increasing degrees of noise, when the core is not optimally sized, and when the method is underpowered.

3.1 Response measures

The performance of an predictive algorithm in can be measured using a number of different metrics each of which highlight some aspect of similarity between the known responses and the predicted responses. For this work we look at four different metrics: normalized root mean squared error, explained variance, Pearson correlation and Spearman correlation. In the equations below y_{ijk} represents the observed responses, $\widehat{y}_{ijk}$ represents the predicted responses, while y and $\widehat{y}$ represent the means of the observed and predicted responses respectively.

The normalized root mean squared error (NRMSE), given in equation 3.1, measures the average distance between predicted and observed values relative to the standard deviation of the observed values. Smaller values represent better performance with one representing a performance no better than simply guessing the global mean for all responses and zero indicating perfect prediction. This measure is sensitive to outliers, so a few very wrong predictions can have a large effect. Similarly, explained variance (given by $1 - NRMSE^2$) gives the proportion that the variance is reduced by using the predicted responses compared with just predicting the mean response. Here larger values are better with zero representing no improvement over guessing the mean and one representing a perfect prediction.

$$NRMSE = \sqrt{\frac{\sum_{ijk}(y_{ijk} - \widehat{y}_{ijk})^2}{\sum_{ijk}(y_{ijk} - y)^2}} \tag{3.1}$$

Pearson correlation (PCC), given by equation 3.2, is a measure of how well a plot of the observed vs. predicted data fit a line. The correlation ranges between negative one and one with -1 and 1 representing a perfect linear relationship with negative or positive slope respectively and 0 representing no linear relationship. The correlation is unaffected by scaling the predictions, so if the algorithm consistently under- or over-estimates values, this correlation can still be high.

$$PCC = \frac{\sum_{ijk}(y_{ijk} - y)(\widehat{y}_{ijk} - \widehat{y})}{\sqrt{\sum_{ijk}(y_{ijk} - y)^2}\sqrt{\sum_{ijk}(\widehat{y}_{ijk} - \widehat{y})^2}} \tag{3.2}$$

To compute the Spearman correlation (SCC), each y_{ijk} and $\widehat{y}_{ijk}$ value is replaced by its rank in the list of all observed or predicted values respectively and then the Pearson correlation is computed between these lists of integers. Thus this value has the same range as the PCC, but a SCC value of 1 means that the two sets have the same ordering, not necessarily a linear relationship. This measure is less sensitive to outliers than any of the methods above, but if there are many repeated values, or close similar values then the rankings may be less meaningful.

All of these measures can be computed for any prediction algorithm and task and allow for comparisons to be made across methods. As some methods tend to perform better with respect to different measures, we report all relevant measures when possible and discuss reasons for differing performances.

3.1.1 Baseline methods

In order to asses the the utility of the BaTFLED algorithm, we compare its performance to a simple baseline prediction and three classes of machine learning methods which have been shown to be optimal for different prediction tasks. The simple baseline consists of predicting the mean response across one or more of the modes. For example, if we are predicting responses for a new cell line across different drugs and doses, then we would take the mean across all training cell lines for each drug and dose combination. This baseline can actually perform quite well for cell lines that have behavior similar to the mean response. Note that this is not the global mean response that is used for calculating the NRMSE and other response measures since it varies across both drugs and doses and can therefore contain a reasonable amount of information about response behaviors.

The three classes of machine learning methods are elastic net models [105], multi-layered neural networks and random forests. These methods are designed to predict a single value given a vector of features, so in order to compare to BaTFLED models, we concatenate vectors of inputs for the different modes into one feature vector. If the task involves simultaneously predicting several values, as in the case of predicting response at each dose, then we run separate models for each task.

Elastic net models extend linear regression by adding two penalty terms to the loss function. The optimal regression coefficients $\widehat{\boldsymbol{\beta}}$ are given by equation 3.3 below. Here $\mathbf{y}$ is a vector of the observed responses, X is a matrix of input features with a row for each sample and a column for each feature, the vector $\boldsymbol{\beta}$ weights each input feature, $||$ is the Euclidean norm and $||_1$ is the L_1 norm ($||\boldsymbol{\beta}||_1 = \sum_{p=1}^{P} |\beta_p|$). The coefficient α is set by the user and controls the degree of sparsity that the model will implement. With $\alpha = 0$ this is known as ridge regression and the model which will place non-zero weights on all features. With $\alpha = 1$ this implements LASSO (least absolute shrinkage and selection operator) regression and a small number of features will be given non-zero weights. In our experiments we typically test four values for α ($\alpha = 0, .5, .9, 1$) and run models using the R package glmnet [106].

$$\widehat{\boldsymbol{\beta}} = argmin_{\boldsymbol{\beta}}(||\mathbf{y} - X\boldsymbol{\beta}||^2 + (1-\alpha)||\boldsymbol{\beta}||^2 + \alpha||\boldsymbol{\beta}||_1) \quad (3.3)$$

Neural network models are something like a stack of linear regression models where the outputs from one layer serve as the inputs for the next layer after a non-linear activation function is applied. There is a large literature on these types of models and a lot of ongoing research into their application. For this work, we use the R H2O package [107] to implement neural nets with one hidden layer of $1,500$ nodes, two hidden layers of 750 nodes each and three hidden layers with 500 nodes each. The non-linear activations are rectified linear units (ReLU) with a dropout fraction of 0.2 on the input layer and 0.5 on hidden layers. These values were chosen using suggestions from literature and to keep the models similarly sized to the BaTFLED models.

Random forest regression models use a collection of binary decision trees to split the response data given feature values. Typically a large number of small trees are generated and then combined to make predictions. Again we used the R H2O package [107] to implement these models with $1,000$ and $5,000$ trees of depth 5, and $1,000$ trees of depth 10.

3.1.2 Prediction tasks

Given a set of responses that varies along three modes, there are four different prediction tasks of varying difficulty that can be performed (figure 3.1). The simplest task consists of making predictions for the three-dimensional responses where responses for each of the given sample for each mode are known, but the response for the given combination of samples has not been observed. For example, in the cell line, drug, dose example this would entail predicting a response for a cell line, drug, dose combination where responses are known for that cell line when treated with different drugs and doses, that drug applied to different cell lines and doses and that dose for different cell lines and drugs. With the responses stored in a tensor, this amounts to filling in missing values and it can be accomplished without the use of the feature data by factoring the tensor and multiplying the factors back together. Since some responses are given for all of the samples, this is known as a ‘warm-start’ prediction task.

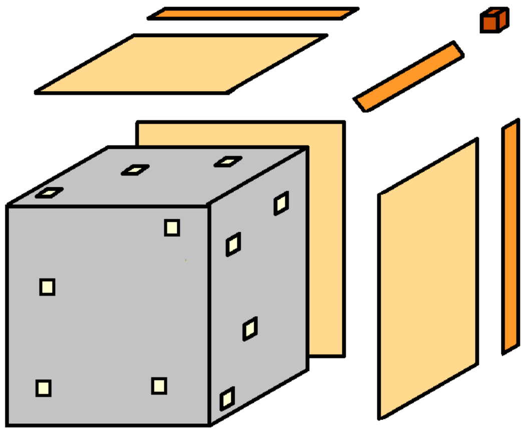


Figure 3.1: Different prediction tasks. The tensor of observed responses is represented by the central gray cube. Missing values in this cube represent warm-start prediction tasks. Lighter orange trapezoids represent prediction for one of the three modes, darker rectangles represent two-mode prediction, and the dark orange cube represents cold-start prediction for all three modes.

The other types of prediction are considered ‘cold-start’ tasks because predictions are made directly from the feature data without having seen responses for the sample from one or more of the modes. The one-mode prediction task entails making a set of predictions for a sample that is not in the training set across the training samples for the other two modes. This can be visualized as adding a slice to the response tensor corresponding to the mode being predicted. For example, in the cell line, drug, dose example, given a vector of feature values for a new cell line, response predictions would be made for that cell line when treated with each drug and dose in the training set. The two-mode prediction task adds a new fiber to the response tensor and requires feature data for a new sample from two of the modes. For example, for a new cell line and drug combination, predictions would be made at each dose. Finally, the three-mode prediction task would use features from each of the modes to predict a single value.

In addition to making predictions the BaTFLED model is also designed to perform feature selection for each of the modes. For the simulated data, the true predictors are a known *a priori* so it is also possible to assess the performance in recovering these variables. For a trained BaTFLED model, the weight for each features is taken to be the average across the rows of the projection matrices. Since this is a continuous measure of feature importance, we use the area under the receiver operator characteristic (AUROC) measure to assess the performance. This value gives the probability that a feature that is a true predictor will be ranked ahead of a feature that is not a true predictor.

3.2 Experiments

In the following subsections, we detail the setup and results for six experiments varying different aspects of the BaTFLED model. In each case the performance is compared to the methods detailed above and relevant response measures are reported.

3.2.1 One, two and three mode interactions

As BaTFLED is designed to model interactions between different modes, we first sought to assess its performance in discovering higher order interactions. To this end we generated responses from the three-mode structure shown in figure 2.1 with the three types of core tensors shown in figure 3.2. The first core tensor (referred to as 1D) only has non-zero values along fibers that correspond to the columns of ones in two of the latent matrices (i.e. c_{i00}, c_{0j0} and c_{00k}) and thus responses are governed by effects coming from each mode separately. The second core tensor (1 & 2D) has non-zero values in the faces corresponding to the columns of ones in one of the latent matrices (i.e. c_{ij0}, c_{i0k} and c_{0jk}) and so allows for interactions between at most two modes. The third core tensor (1, 2 & 3D) has values in any of the positions, but is sparse so that only $\frac{1}{2}$ of the possible interactions have a non-zero weight.

In all cases, response data is generated with 30 samples per mode and 100 features for each mode, 10 of which have non-zero weights in the projection matrices. Each mode has four latent factors, so with the added columns of ones, the core tensor is $5 \times 5 \times 5$. Zero-centered Gaussian noise with a standard deviation of $\frac{1}{10}$ of the response standard deviation is added to responses to simulate random measurement error. Before training, two examples for each mode are removed and used for cold-start testing and 1% of interactions removed for warm-start testing. Details on run parameters for the baseline methods are

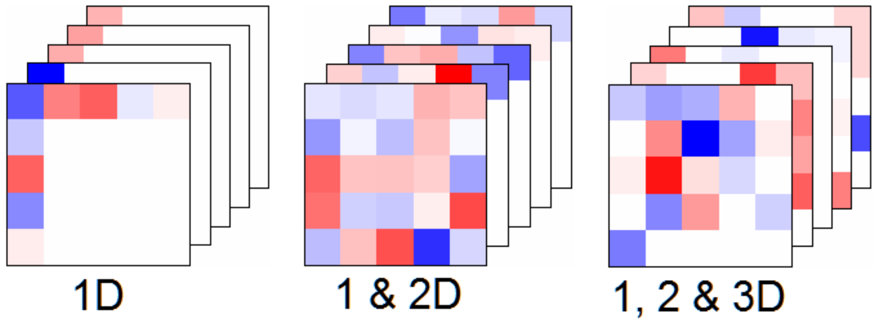


Figure 3.2: Three cores used when simulating data allowing one-mone, two-mode and three-mode interactions respectively.

given in section 3.1.1. BaTFLED models are run for 100 iterations with prior parameters $\alpha = 10^{-5}$ and $\beta = 10^{5}$. The CP models have a latent dimension of 64 to compare with the $4 \times 4 \times 4$ core of the Tucker models. Note that for CP models there is no advantage to adding columns of ones to the latent matrices since these would only be able to interact with each other. All models are run in 10 replicates with mean performance measures reported below.

From figures 3.3 and 3.4 and tables 3.2.1 and 3.2.1 we see that BaTFLED Tucker models perform as well as the best baseline models on tasks when the data is generated without higher order interactions (1D), but when these interactions are present, BaTFLED Tucker models far outperform other methods. It is interesting to note that BaTFLED CP models outperform elastic net models for the 1 & 2D warm-start task but do not perform as well as neural network and some random forest models which allow non-linear interactions between predictors. Moreover, CP models may slightly outperform other methods for the multi-mode cold-start 1 & 2D tasks. Since the majority of tensor factorizations published to date have used CP decompositions and focused on warm-start prediction, these results indicate that there is much to be gained by switching to the more flexible Tucker decomposition especially when attempting cold-start prediction for multiple modes.

When generating the data, only 10 of 100 features for each mode are ‘real’ (i.e. have non-zero weights in the projection matrices) and can influence the responses. A secondary goal of the BaTFLED model is to identify which predictors are real. We use area under the receiver operator characteristic curves (AUROC) to assess the performance of the above

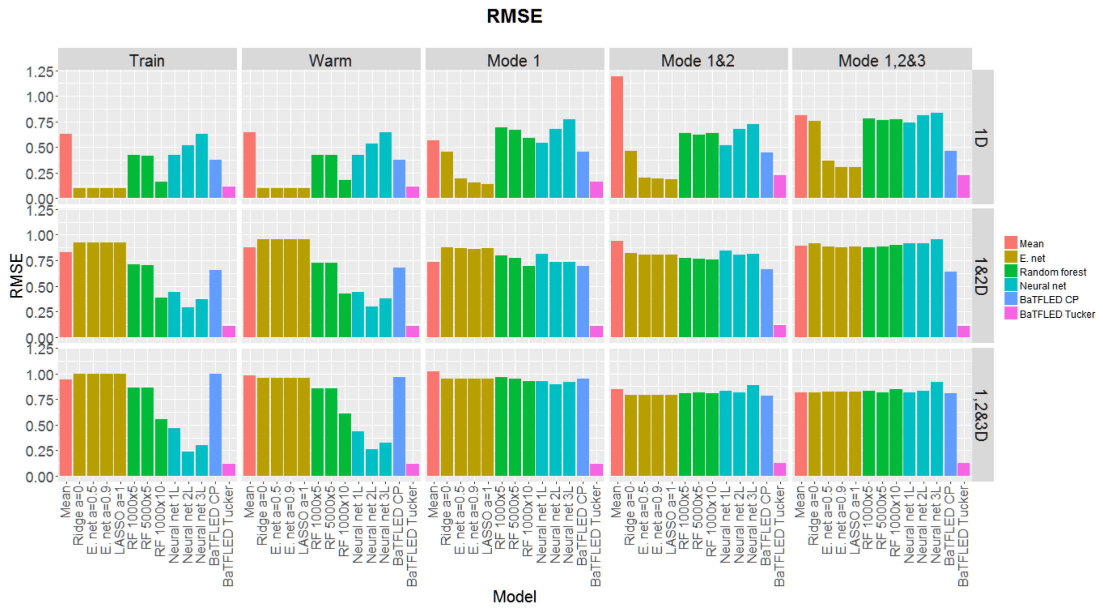


Figure 3.3: Normalized root mean squared error measures for BaTFLED and baseline methods when predicting responses with one, two and three mode interactions (means of 10 replicates). Columns represents different prediction tasks and rows show performance on data generated with no interactions between modes, interactions between at most two modes and interactions between any of the three modes. Results for cold predictions of modes 2 and 3 are similar.

models in identifying the true predictors. This requires a weight on the importance for each feature which is supplied by all of the baseline methods. For BaTFLED models, we tested several methods of weighting the predictors all of which had similar results, so we report here the simplest method of just summing the squared values across rows in the projection matrices. From the results shown in figure 3.5 and table 3.2.1, we see that BaTFLED Tucker models are able to recover the true predictors as well or better than other methods in all cases while BaTFLED CP models cannot. The advantage that Tucker models provide is especially clear when higher-order interactions are present in the data.

In the next sections we examine how this performance degrades if the simulated data have more missing values, an increasing proportion of noise, a smaller core tensor or have less samples to use for training.

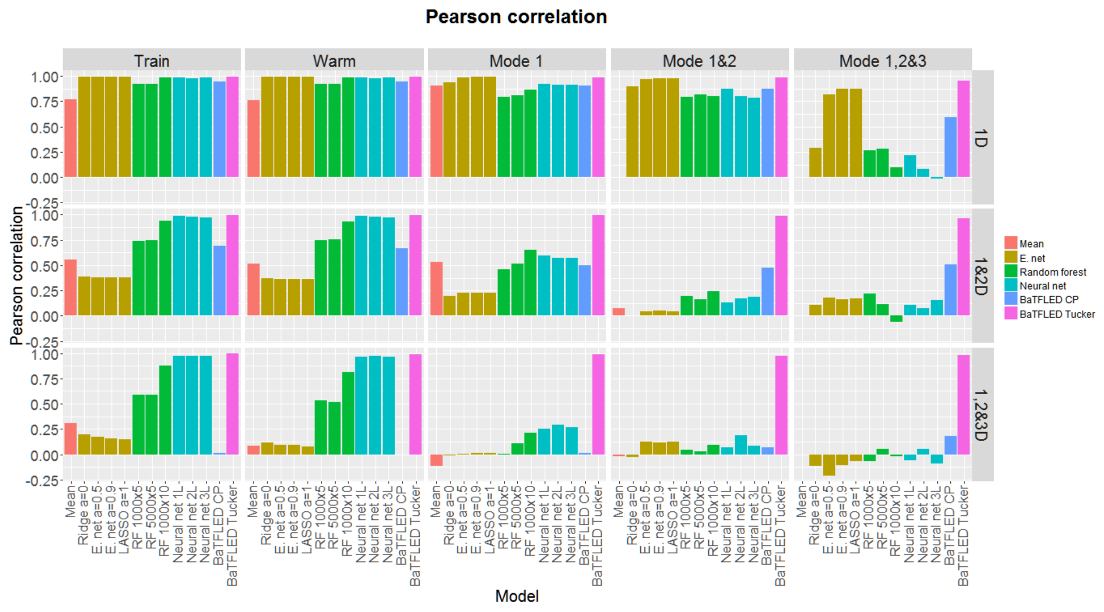


Figure 3.4: Pearson correlation measures for BaTFLED and baseline methods when predicting responses with one, two and three mode interactions (means of 10 replicates). Columns represents different prediction task and rows show performance on data generated with no interactions between modes, interactions between at most two modes and interactions between any of the three modes. Results for cold predictions of modes 2 and 3 are similar.

			Cold start prediction:						
	Train	Warm	Mode 1	Mode 2	Mode 3	Mode 1&2	Mode 1&3	Mode 2&3	Mode 1,2&3
	Data generated with only 1D interactions								
Mean	0.56(0.14)	0.59(0.15)	0.57(0.44)	0.36(0.15)	0.52(0.28)	1.19(0.50)	1.29(0.56)	1.16(0.30)	0.81(0.20)
LASSO $\alpha = 1$	**0.10**(0.00)	**0.10**(0.01)	**0.14**(0.04)	**0.18**(0.10)	0.27(0.19)	**0.18**(0.07)	0.31(0.18)	0.30(0.16)	0.30(0.16)
E. Net $\alpha = 0.9$	**0.10**(0.00)	**0.10**(0.01)	0.15(0.04)	**0.18**(0.10)	0.28(0.21)	0.19(0.06)	0.32(0.19)	0.31(0.17)	0.31(0.17)
E. Net $\alpha = 0.5$	**0.10**(0.00)	**0.10**(0.01)	0.19(0.04)	0.20(0.11)	0.32(0.28)	0.20(0.06)	0.39(0.25)	0.37(0.25)	0.37(0.25)
Ridge $\alpha = 0$	**0.10**(0.00)	**0.10**(0.01)	0.45(0.31)	0.29(0.12)	0.55(0.28)	0.46(0.18)	0.75(0.29)	0.65(0.30)	0.76(0.23)
R. Forest 1Kx5	0.42(0.04)	0.42(0.04)	0.70(0.34)	0.47(0.13)	0.73(0.29)	0.63(0.22)	0.83(0.20)	0.65(0.20)	0.78(0.14)
R. Forest 5Kx5	0.42(0.05)	0.42(0.05)	0.67(0.30)	0.48(0.14)	0.70(0.27)	0.62(0.17)	0.79(0.20)	0.66(0.25)	0.77(0.20)
R. Forest 1Kx10	0.16(0.01)	0.18(0.01)	0.59(0.35)	0.40(0.16)	0.58(0.30)	0.63(0.22)	0.76(0.22)	0.62(0.21)	0.77(0.18)
Neural net 1L.	0.43(0.05)	0.42(0.06)	0.54(0.31)	0.50(0.08)	0.58(0.25)	0.52(0.20)	0.74(0.24)	0.71(0.27)	0.74(0.19)
Neural net 2L.	0.52(0.02)	0.53(0.03)	0.67(0.31)	0.62(0.10)	0.59(0.24)	0.68(0.18)	0.78(0.25)	0.77(0.28)	0.81(0.22)
Neural net 3L.	0.63(0.03)	0.65(0.04)	0.77(0.28)	0.68(0.07)	0.72(0.21)	0.73(0.16)	0.85(0.22)	0.82(0.23)	0.84(0.18)
BaTFLED CP.	0.37(0.04)	0.38(0.04)	0.46(0.15)	0.42(0.11)	0.46(0.13)	0.45(0.20)	0.52(0.15)	0.45(0.11)	0.46(0.18)
BaTFLED Tucker	0.11(0.00)	0.11(0.01)	0.16(0.05)	0.20(0.16)	**0.15**(0.03)	0.22(0.13)	**0.18**(0.05)	**0.22**(0.14)	**0.23**(0.12)
	Data generated with 1D & 2D interactions								
Mean	0.83(0.05)	0.88(0.07)	0.74(0.26)	0.78(0.22)	0.84(0.39)	0.94(0.30)	1.09(0.31)	1.37(0.34)	0.89(0.39)
LASSO $\alpha = 1$	0.92(0.04)	0.95(0.08)	0.86(0.19)	0.93(0.15)	1.05(0.27)	0.81(0.23)	0.86(0.24)	1.13(0.36)	0.88(0.37)
E. Net $\alpha = 0.9$	0.92(0.04)	0.96(0.08)	0.86(0.19)	0.93(0.15)	1.05(0.27)	0.80(0.23)	0.86(0.23)	1.13(0.36)	0.88(0.36)
E. Net $\alpha = 0.5$	0.92(0.04)	0.95(0.08)	0.87(0.19)	0.93(0.15)	1.05(0.28)	0.81(0.23)	0.86(0.23)	1.14(0.36)	0.89(0.36)
Ridge $\alpha = 0$	0.92(0.04)	0.95(0.08)	0.87(0.19)	0.95(0.15)	1.06(0.28)	0.82(0.26)	0.85(0.20)	1.18(0.38)	0.92(0.40)
R. Forest 1Kx5	0.71(0.08)	0.73(0.09)	0.79(0.18)	0.84(0.17)	1.01(0.35)	0.77(0.22)	0.86(0.21)	1.11(0.33)	0.88(0.34)
R. Forest 5Kx5	0.71(0.08)	0.72(0.09)	0.78(0.18)	0.83(0.17)	1.01(0.36)	0.77(0.23)	0.83(0.22)	1.08(0.30)	0.88(0.33)
R. Forest 1Kx10	0.38(0.04)	0.42(0.05)	0.69(0.20)	0.73(0.18)	0.93(0.41)	0.75(0.21)	0.82(0.22)	1.05(0.31)	0.90(0.35)
Neural net 1L.	0.44(0.03)	0.44(0.04)	0.81(0.25)	0.77(0.23)	0.97(0.35)	0.84(0.28)	0.90(0.21)	1.09(0.35)	0.91(0.42)
Neural net 2L.	0.29(0.02)	0.30(0.02)	0.74(0.27)	0.71(0.22)	0.87(0.38)	0.80(0.30)	0.86(0.22)	1.09(0.33)	0.91(0.45)
Neural net 3L.	0.37(0.04)	0.37(0.04)	0.73(0.27)	0.72(0.22)	0.84(0.35)	0.81(0.32)	0.86(0.23)	1.05(0.31)	0.96(0.50)
BaTFLED CP.	0.66(0.18)	0.68(0.18)	0.70(0.17)	0.67(0.19)	0.77(0.29)	0.66(0.24)	0.71(0.15)	0.81(0.36)	0.64(0.27)
BaTFLED Tucker	**0.11**(0.01)	**0.11**(0.01)	**0.11**(0.01)	**0.11**(0.01)	**0.11**(0.01)	**0.11**(0.01)	**0.12**(0.02)	**0.12**(0.02)	**0.11**(0.04)
	Data generated with 1D, 2D & 3D interactions								
Mean	0.94(0.04)	0.98(0.05)	1.02(0.30)	0.80(0.37)	1.05(0.32)	0.85(0.43)	1.13(0.47)	0.89(0.49)	0.81(0.58)
LASSO $\alpha = 1$*	0.99(0.01)	0.96(0.04)	0.95(0.29)	0.82(0.36)	1.08(0.35)	0.79(0.44)	1.06(0.49)	0.83(0.51)	0.82(0.58)
E. Net $\alpha = 0.9$*	1.00(0.01)	0.96(0.04)	0.95(0.29)	0.82(0.36)	1.08(0.35)	0.79(0.44)	1.06(0.49)	0.83(0.51)	0.82(0.58)
E. Net $\alpha = 0.5$*	0.99(0.01)	0.96(0.04)	0.95(0.29)	0.82(0.36)	1.08(0.35)	0.79(0.43)	1.06(0.49)	0.83(0.51)	0.82(0.58)
Ridge $\alpha = 0$	1.00(0.01)	0.96(0.04)	0.95(0.29)	0.82(0.36)	1.08(0.35)	0.79(0.44)	1.06(0.49)	0.83(0.51)	0.81(0.58)
R. Forest 1Kx5	0.86(0.03)	0.85(0.04)	0.96(0.30)	0.78(0.34)	1.07(0.34)	0.81(0.40)	1.08(0.48)	0.82(0.49)	0.83(0.55)
R. Forest 5Kx5	0.86(0.02)	0.86(0.04)	0.95(0.27)	0.80(0.34)	1.07(0.33)	0.81(0.39)	1.06(0.47)	0.83(0.49)	0.81(0.56)
R. Forest 1Kx10	0.55(0.05)	0.61(0.07)	0.92(0.25)	0.78(0.32)	1.05(0.31)	0.81(0.39)	1.06(0.44)	0.84(0.46)	0.85(0.55)
Neural net 1L.	0.46(0.04)	0.43(0.03)	0.92(0.27)	0.92(0.25)	1.07(0.28)	0.83(0.42)	1.03(0.45)	0.87(0.48)	0.81(0.56)
Neural net 2L.	0.24(0.01)	0.26(0.02)	0.89(0.25)	0.81(0.32)	1.10(0.26)	0.81(0.43)	1.04(0.45)	0.86(0.47)	0.83(0.56)
Neural net 3L.	0.30(0.04)	0.33(0.03)	0.92(0.28)	0.82(0.32)	1.10(0.28)	0.89(0.44)	1.03(0.43)	0.88(0.45)	0.92(0.56)
BaTFLED CP.	1.00(0.00)	0.96(0.04)	0.95(0.29)	0.83(0.35)	1.08(0.35)	0.78(0.45)	1.05(0.50)	0.83(0.51)	0.81(0.59)
BaTFLED Tucker	**0.12**(0.03)	**0.12**(0.02)	**0.12**(0.03)	**0.13**(0.04)	**0.12**(0.02)	**0.13**(0.04)	**0.12**(0.02)	**0.13**(0.04)	**0.12**(0.04)

* Elastic net set all weights to zero in 4-5 of 10 runs.

Table 3.1: Mean normalized RMSE performance on simulated three-mode datasets relative to training set standard deviation across ten replicates (sd)

			Cold start prediction:						
	Train	Warm	Mode 1	Mode 2	Mode 3	Mode 1&2	Mode 1&3	Mode 2&3	Mode 1,2&3
	Data generated with only 1D interactions								
Mean	0.77(0.08)	0.76(0.08)	0.90(0.11)	0.97(0.03)	0.85(0.15)	0.00(0.42)	0.01(0.51)	-0.12(0.39)	NA
LASSO $\alpha = 1$	**1.00**(0.00)	**1.00**(0.00)	**0.99**(0.00)	**0.99**(0.01)	0.94(0.13)	0.97(0.03)	0.90(0.20)	0.92(0.13)	0.88(0.20)
E. Net $\alpha = 0.9$	**1.00**(0.00)	**1.00**(0.00)	**0.99**(0.01)	**0.99**(0.01)	0.93(0.13)	0.97(0.03)	0.90(0.20)	0.92(0.13)	0.87(0.21)
E. Net $\alpha = 0.5$	**1.00**(0.00)	**1.00**(0.00)	**0.99**(0.01)	**0.99**(0.01)	0.91(0.16)	0.97(0.04)	0.86(0.22)	0.89(0.16)	0.82(0.23)
Ridge $\alpha = 0$	**1.00**(0.00)	**1.00**(0.00)	0.94(0.08)	0.97(0.02)	0.83(0.19)	0.89(0.08)	0.64(0.26)	0.75(0.26)	0.29(0.55)
R. Forest 1Kx5	0.92(0.02)	0.92(0.02)	0.79(0.16)	0.93(0.04)	0.75(0.16)	0.79(0.15)	0.39(0.56)	0.74(0.32)	0.26(0.61)
R. Forest 5Kx5	0.92(0.01)	0.92(0.02)	0.81(0.11)	0.93(0.04)	0.75(0.16)	0.81(0.10)	0.42(0.59)	0.75(0.33)	0.28(0.67)
R. Forest 1Kx10	0.99(0.00)	0.99(0.00)	0.86(0.15)	0.96(0.03)	0.84(0.15)	0.80(0.13)	0.48(0.46)	0.76(0.26)	0.10(0.63)
Neural net 1L.	0.99(0.00)	0.99(0.00)	0.92(0.08)	0.96(0.03)	0.85(0.15)	0.87(0.08)	0.64(0.26)	0.78(0.19)	0.21(0.56)
Neural net 2L.	0.98(0.00)	0.98(0.00)	0.91(0.15)	0.94(0.05)	0.84(0.15)	0.80(0.21)	0.56(0.44)	0.69(0.33)	0.08(0.63)
Neural net 3L.	0.99(0.00)	0.99(0.00)	0.91(0.13)	0.95(0.04)	0.84(0.15)	0.78(0.23)	0.38(0.50)	0.64(0.35)	-0.02(0.69)
BaTFLED CP.	0.95(0.01)	0.95(0.01)	0.90(0.09)	0.94(0.03)	0.88(0.09)	0.87(0.13)	0.76(0.26)	0.86(0.11)	0.59(0.55)
BaTFLED Tucker	0.99(0.00)	0.99(0.00)	**0.99**(0.01)	**0.99**(0.00)	**0.99**(0.01)	**0.98**(0.02)	**0.97**(0.02)	**0.98**(0.01)	**0.95**(0.06)
	Data generated with 1D & 2D interactions								
Mean	0.55(0.08)	0.51(0.09)	0.53(0.29)	0.62(0.14)	0.63(0.23)	0.08(0.34)	0.00(0.23)	-0.08(0.15)	NA
LASSO $\alpha = 1$	0.38(0.09)	0.36(0.13)	0.23(0.29)	0.26(0.35)	0.26(0.25)	0.04(0.31)	0.17(0.44)	0.25(0.35)	0.17(0.56)
E. Net $\alpha = 0.9$	0.38(0.09)	0.36(0.13)	0.23(0.29)	0.26(0.35)	0.26(0.24)	0.05(0.30)	0.17(0.44)	0.26(0.35)	0.17(0.57)
E. Net $\alpha = 0.5$	0.38(0.09)	0.36(0.13)	0.22(0.28)	0.26(0.35)	0.26(0.25)	0.04(0.30)	0.17(0.44)	0.25(0.35)	0.18(0.56)
Ridge $\alpha = 0$	0.39(0.09)	0.37(0.13)	0.20(0.30)	0.26(0.34)	0.25(0.26)	0.01(0.30)	0.23(0.39)	0.21(0.39)	0.11(0.32)
R. Forest 1Kx5	0.74(0.06)	0.75(0.08)	0.46(0.20)	0.46(0.34)	0.40(0.28)	0.19(0.36)	0.24(0.36)	0.33(0.37)	0.22(0.46)
R. Forest 5Kx5	0.75(0.06)	0.75(0.07)	0.51(0.20)	0.44(0.41)	0.38(0.35)	0.16(0.39)	0.35(0.30)	0.34(0.43)	0.12(0.51)
R. Forest 1Kx10	0.94(0.01)	0.93(0.02)	0.65(0.22)	0.64(0.29)	0.53(0.29)	0.24(0.53)	0.34(0.41)	0.47(0.45)	-0.06(0.66)
Neural net 1L.	0.98(0.00)	0.98(0.00)	0.60(0.19)	0.69(0.16)	0.62(0.22)	0.13(0.38)	0.28(0.21)	0.40(0.40)	0.11(0.45)
Neural net 2L.	0.98(0.00)	0.98(0.00)	0.57(0.18)	0.71(0.16)	0.63(0.20)	0.17(0.34)	0.22(0.27)	0.51(0.23)	0.07(0.36)
Neural net 3L.	0.97(0.00)	0.97(0.01)	0.57(0.19)	0.70(0.17)	0.64(0.20)	0.19(0.40)	0.24(0.25)	0.54(0.27)	0.16(0.43)
BaTFLED CP.	0.69(0.26)	0.67(0.31)	0.50(0.34)	0.67(0.28)	0.66(0.27)	0.47(0.35)	0.48(0.28)	0.70(0.26)	0.51(0.50)
BaTFLED Tucker	**0.99**(0.00)	**0.99**(0.00)	**0.99**(0.00)	**0.99**(0.00)	**0.99**(0.00)	**0.99**(0.01)	**0.99**(0.01)	**0.99**(0.01)	**0.96**(0.08)
	Data generated with 1D, 2D & 3D interactions								
Mean	0.31(0.11)	0.08(0.12)	-0.11(0.50)	0.22(0.40)	0.23(0.25)	-0.02(0.16)	-0.05(0.14)	0.03(0.08)	NA
LASSO $\alpha = 1$*	0.15(0.05)	0.08(0.08)	0.02(0.17)	0.20(0.12)	0.01(0.17)	0.12(0.17)	-0.19(0.23)	0.16(0.25)	-0.07(0.35)
E. Net $\alpha = 0.9$*	0.16(0.06)	0.09(0.09)	0.01(0.19)	0.17(0.12)	0.00(0.19)	0.12(0.19)	-0.23(0.25)	0.12(0.26)	-0.11(0.37)
E. Net $\alpha = 0.5$*	0.17(0.05)	0.10(0.09)	0.00(0.20)	0.18(0.12)	0.01(0.20)	0.13(0.20)	-0.29(0.25)	0.18(0.24)	-0.21(0.34)
Ridge $\alpha = 0$	0.20(0.05)	0.11(0.10)	-0.01(0.22)	0.11(0.18)	0.01(0.25)	-0.02(0.30)	-0.20(0.35)	0.03(0.36)	-0.11(0.31)
R. Forest 1Kx5	0.59(0.06)	0.53(0.11)	0.01(0.30)	0.27(0.31)	0.13(0.20)	0.05(0.25)	-0.04(0.26)	0.18(0.34)	-0.07(0.43)
R. Forest 5Kx5	0.59(0.06)	0.51(0.11)	0.11(0.35)	0.23(0.33)	0.14(0.17)	0.03(0.32)	0.04(0.26)	0.16(0.30)	0.05(0.43)
R. Forest 1Kx10	0.88(0.03)	0.81(0.06)	0.22(0.49)	0.33(0.34)	0.24(0.24)	0.09(0.28)	0.04(0.37)	0.17(0.32)	-0.02(0.46)
Neural net 1L.	0.97(0.00)	0.96(0.01)	0.25(0.50)	0.39(0.39)	0.21(0.27)	0.07(0.35)	0.12(0.37)	0.05(0.26)	-0.06(0.26)
Neural net 2L.	0.97(0.00)	0.97(0.00)	0.30(0.45)	0.38(0.38)	0.22(0.19)	0.19(0.36)	0.14(0.31)	0.09(0.33)	0.06(0.30)
Neural net 3L.	0.97(0.00)	0.96(0.00)	0.27(0.47)	0.35(0.44)	0.21(0.24)	0.09(0.36)	0.19(0.37)	0.12(0.34)	-0.09(0.37)
BaTFLED CP.	0.01(0.03)	0.00(0.07)	0.01(0.06)	-0.02(0.05)	0.02(0.07)	0.07(0.20)	0.02(0.19)	0.00(0.16)	0.18(0.36)
BaTFLED Tucker	**0.99**(0.00)	**0.99**(0.00)	**0.99**(0.01)	**0.98**(0.01)	**0.99**(0.00)	**0.97**(0.04)	**0.99**(0.02)	**0.98**(0.01)	**0.98**(0.03)

* Elastic net set all weights to zero in 4-5 of 10 runs.

Table 3.2: Mean Pearson correlation performance on simulated three-mode datasets across ten replicates (sd)

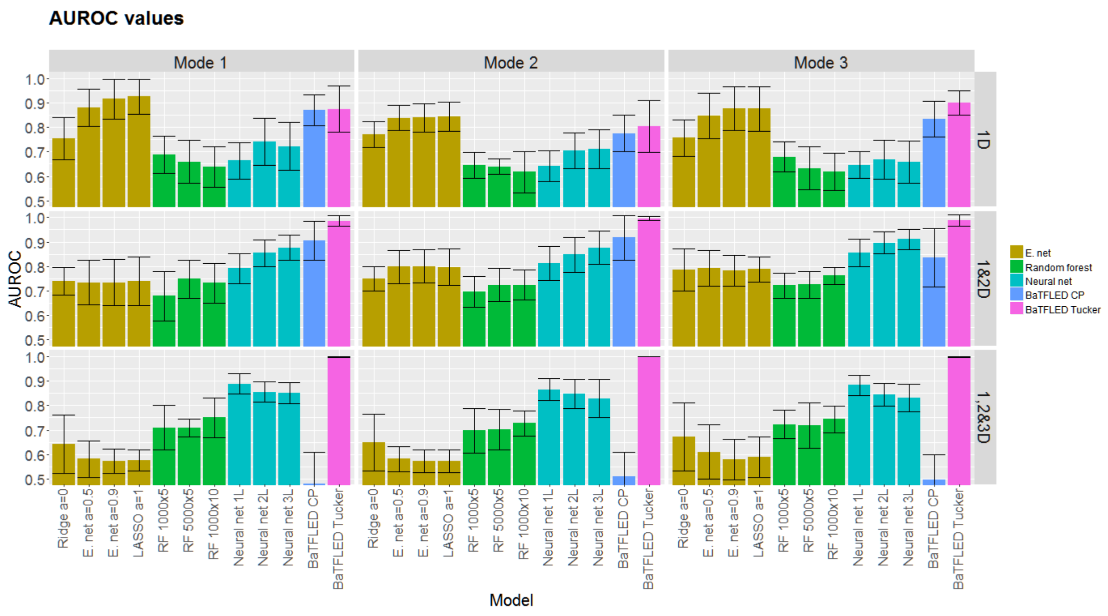


Figure 3.5: Area under the receiver operator characteristic curves for BaTFLED and baseline methods when predicting responses with one, two and three mode interactions (means of 10 replicates, sd shown in error bars). Columns show the three modes and rows show performance on data generated with no interactions between modes, interactions between at most two modes and interactions between any of the three modes.

	LASSO $\alpha = 1$	E. Net $\alpha = .9$	E. Net $\alpha = .5$	Ridge $\alpha = 0$	RF 1Kx5	RF 5Kx5	RF 1Kx10	Neural net 1L.	Neural net 2L.	Neural net 3L.	BaTFLED CP.	BaTFLED Tucker
	Data generated with only 1D interactions											
Mode 1	**0.93**(0.07)	0.92(0.08)	0.88(0.08)	0.75(0.09)	0.69(0.08)	0.66(0.09)	0.64(0.08)	0.66(0.07)	0.74(0.10)	0.72(0.10)	0.87(0.06)	0.87(0.09)
Mode 2	**0.84**(0.06)	**0.84**(0.06)	**0.84**(0.05)	0.77(0.05)	0.65(0.05)	0.64(0.03)	0.62(0.08)	0.64(0.06)	0.70(0.07)	0.71(0.08)	0.78(0.08)	0.80(0.11)
Mode 3	0.88(0.09)	0.88(0.09)	0.85(0.09)	0.76(0.07)	0.68(0.06)	0.63(0.09)	0.62(0.08)	0.65(0.05)	0.67(0.08)	0.66(0.09)	0.83(0.07)	**0.90**(0.05)
	Data generated with 1D & 2D interactions											
Mode 1	0.74(0.10)	0.73(0.09)	0.73(0.09)	0.74(0.06)	0.68(0.10)	0.75(0.08)	0.73(0.08)	0.79(0.06)	0.85(0.06)	0.88(0.05)	0.90(0.08)	**0.99**(0.02)
Mode 2	0.80(0.08)	0.80(0.07)	0.80(0.07)	0.75(0.05)	0.70(0.06)	0.72(0.07)	0.72(0.06)	0.81(0.07)	0.85(0.07)	0.88(0.07)	0.92(0.09)	**1.00**(0.01)
Mode 3	0.79(0.05)	0.78(0.06)	0.79(0.07)	0.79(0.09)	0.72(0.05)	0.72(0.05)	0.76(0.04)	0.86(0.06)	0.90(0.04)	0.91(0.04)	0.84(0.12)	**0.99**(0.02)
	Data generated with 1D, 2D & 3D interactions											
Mode 1	0.58(0.04)	0.57(0.05)	0.58(0.07)	0.64(0.12)	0.71(0.09)	0.71(0.04)	0.75(0.08)	0.89(0.04)	0.86(0.04)	0.85(0.04)	0.48(0.13)	**1.00**(0.00)
Mode 2	0.57(0.05)	0.57(0.05)	0.58(0.05)	0.65(0.12)	0.70(0.09)	0.70(0.08)	0.73(0.05)	0.87(0.05)	0.85(0.06)	0.83(0.08)	0.51(0.10)	**1.00**(0.00)
Mode 3	0.59(0.08)	0.58(0.08)	0.61(0.11)	0.67(0.14)	0.72(0.06)	0.72(0.09)	0.74(0.06)	0.88(0.04)	0.84(0.05)	0.83(0.06)	0.50(0.10)	**1.00**(0.00)

Table 3.3: Mean area under the receiver operator characteristic curve (AUROC) for selecting predictors in simulated three-mode data over 10 replicates (sd)

3.2.2 Missing responses

In real datasets, a large portion of the responses are often missing. In order to ascertain the resilience of BaTFLED to missing data we repeated the above tests removing 10, 25, 50, 75 and 90 percent of responses and using these as warm-test data. For these tests we only compared to ridge and LASSO regression and results for RMSE and Pearson's correlation are shown in figures 3.6 and 3.7. From these plots it is clear that BaTFLED Tucker models outperform other methods and that there is little degradation of performance with up to 75% of responses missing in the case of 1 & 2D interactions and up to 50% of responses missing with 1, 2 & 3D interactions. For the case where the core allows no interactions between modes (1D), BaTFLED Tucker performs similarly to LASSO models but slightly better in the case of predicting responses for all modes simultaneously (M123).

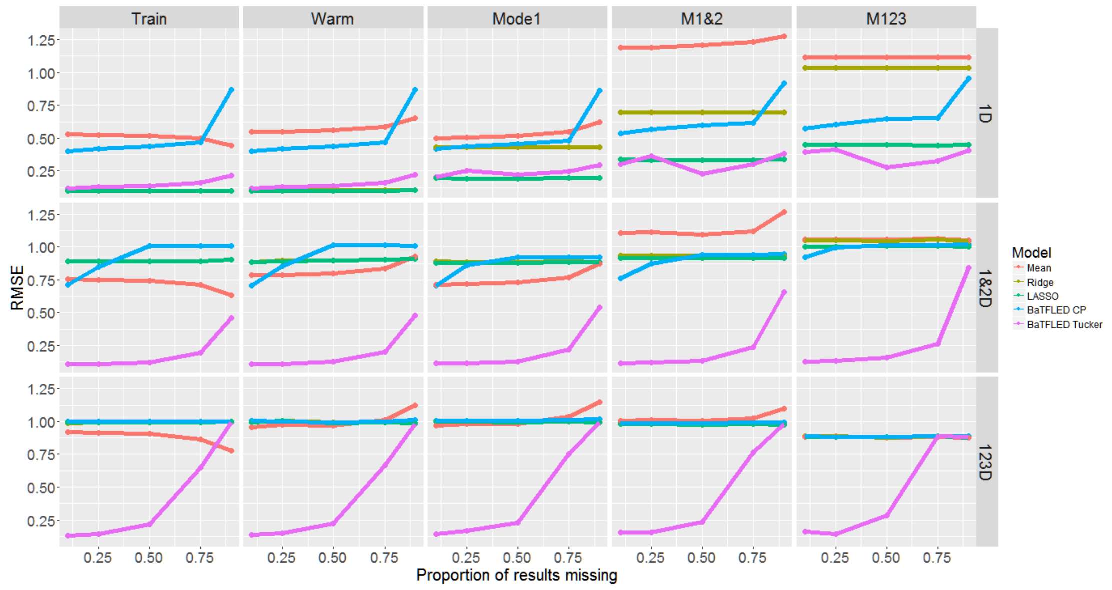


Figure 3.6: Normalized root mean squared error for BaTFLED, ridge and LASSO regression methods when predicting responses with increasing sparsity of responses for one, two and three mode interactions. We tested removing 10%, 25%, 50%, 75% and 90% of responses and using these for warm-start testing. Values shown represent means across 10 replicates.

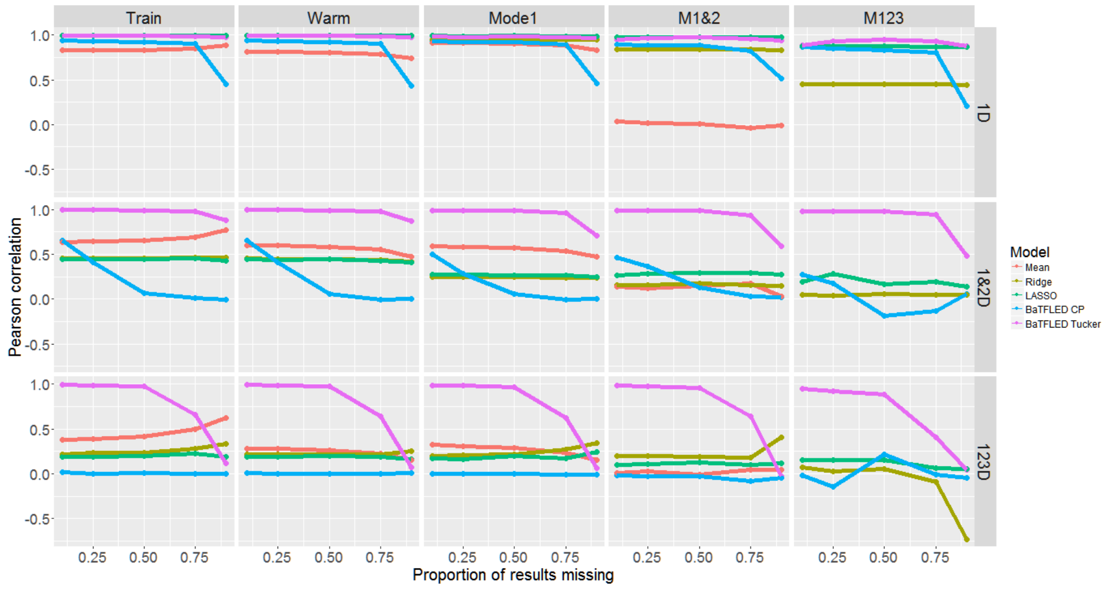


Figure 3.7: Pearson correlation for BaTFLED, ridge and LASSO regression methods when predicting responses with increasing sparsity of responses for one, two and three mode interactions. We tested removing 10%, 25%, 50%, 75% and 90% of responses and using these for warm-start testing. Values shown represent means across 10 replicates.

3.2.3 Increasinge response noise

Real datasets often contain a large degree of noise in responses that is not related to the underlying structure in the data. To ascertain the resilience of BaTFLED to this type of noise we repeated the above tests with 10% of responses missing while adding an increasing amount of zero-mean Gaussian noise to responses. We tested noise generated with standard deviations 0.1, 0.25, 0.5, 0.75 and 0.9 times the standard deviation of the responses. From figures 3.8 and 3.9 we see again that BaTFLED Tucker models outperform other methods, but that performance degrades approximately linearly with increasing noise for all tasks.

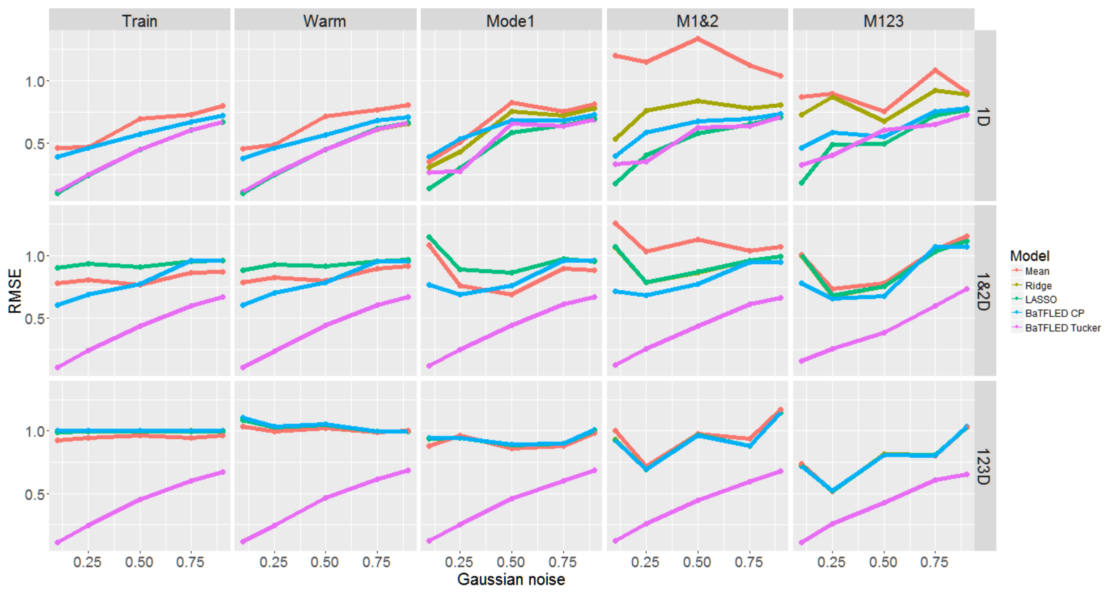


Figure 3.8: Normalized root mean squared error for BaTFLED, ridge and LASSO regression methods when making predictions with increasing noise added to responses for one, two and three mode interactions. We tested adding zero-mean Gaussian noise with standard deviation 0.1, 0.25, 0.50, 0.75 and 0.90 times the standard deviation of the responses. Values shown represent means across 10 replicates.

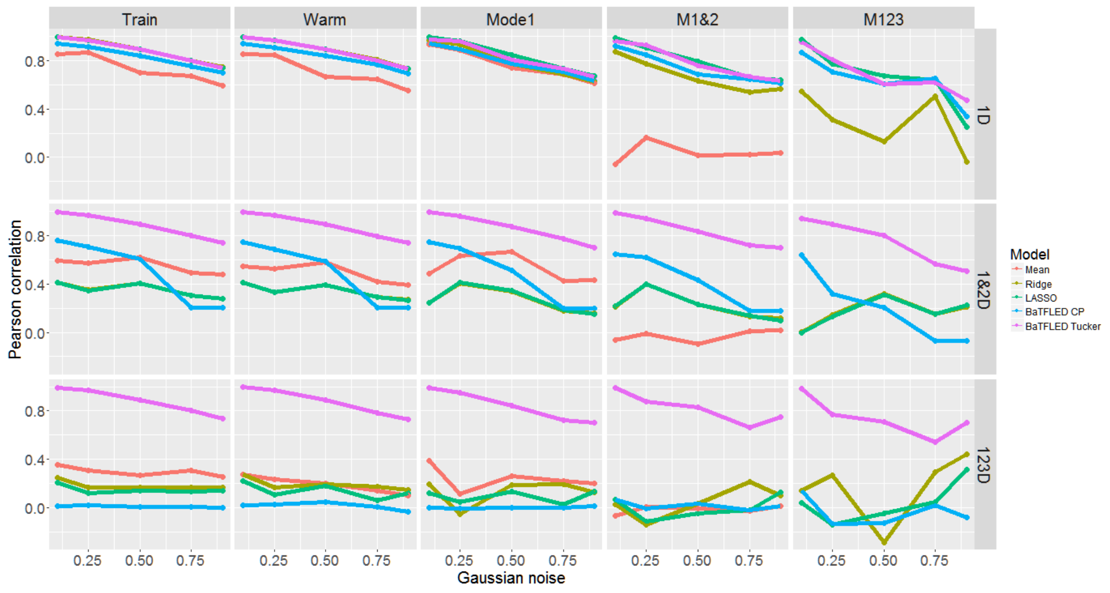


Figure 3.9: Pearson correlation for BaTFLED, ridge and LASSO regression methods when making predictions with increasing noise added to responses for one, two and three mode interactions. We tested adding zero-mean Gaussian noise with standard deviation 0.1, 0.25, 0.50, 0.75 and 0.90 times the standard deviation of the responses. Values shown represent means across 10 replicates.

3.2.4 Decreasing core size

In all of the above simulations the core used during training Tucker models is the same size as the core used to generate the data ($4 \times 4 \times 4$). Since the optimal core size is generally not known *a priori* we investigated how the performance changes as we decreased the size of the core used during training. In order to make fair comparisons with CP decompositions, we also decreased the number of latent features in these models maintaining the same total number of unknown values. So, for example, if a Tucker model is trained with a $3 \times 3 \times 3$ core we would train the CP model with a latent dimension of $3 * 3 * 3 = 27$. Figures 3.10 and 3.11 show that the performance of BaTFLED Tucker models degrades roughly linearly as the size of the core decreases while CP models fluctuate. Interestingly, when predicting responses generated without higher order interactions (1D), CP models seem to improve slightly as the core size decreases. One explanation for this may be that there are only 13 values in the true core for these models so CP models with more than 13 latent factors may either be over-fitting the training data or have less power to learn the larger number of variables.

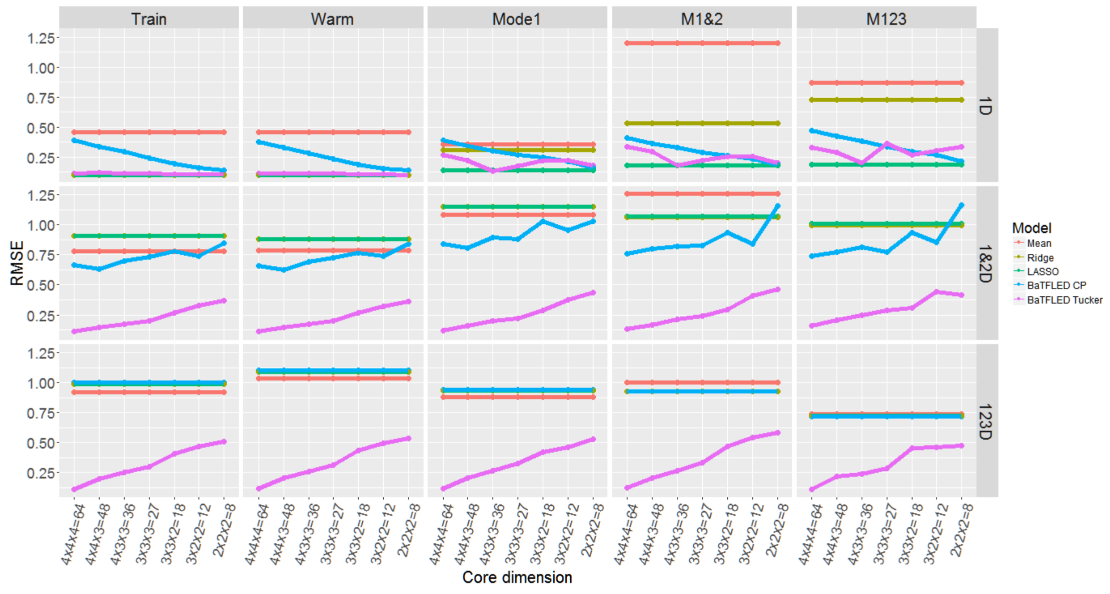


Figure 3.10: Normalized root mean squared error for BaTFLED, ridge and LASSO regression methods when making Tucker model predictions with decreasing core sizes for one, two and three mode interactions. We tested core sizes ranging from the size used to generate responses ($4 \times 4 \times 4$) down to $2 \times 2 \times 2$. For BaTFLED CP models, the number of latent factors was set to the product of the dimensions of the core in the Tucker models. Values shown represent means across 10 replicates.

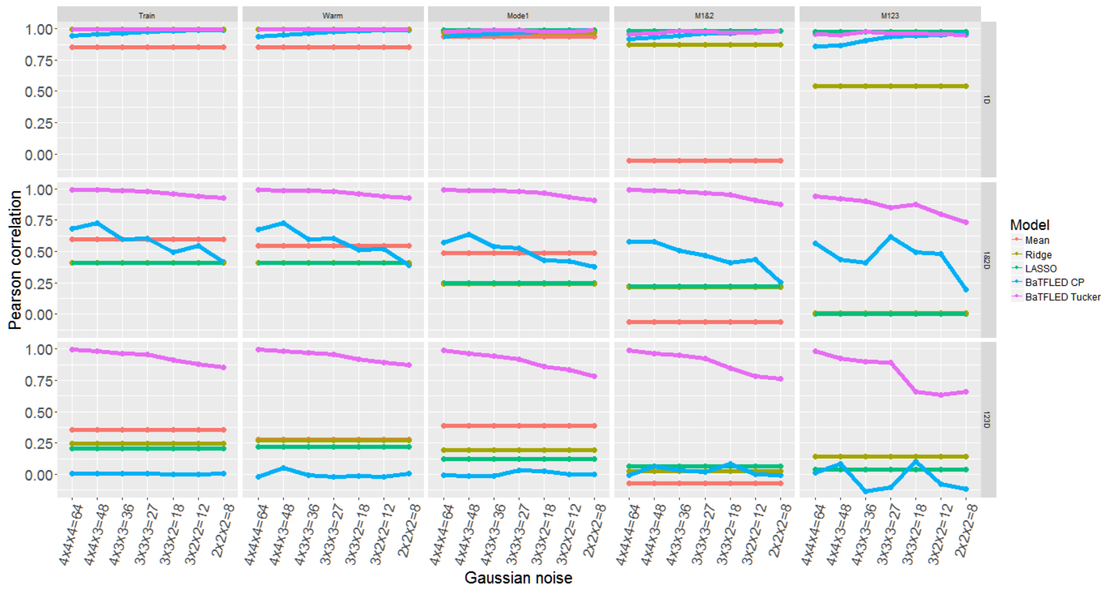


Figure 3.11: Pearson correlation for BaTFLED, ridge and LASSO regression methods when making Tucker model predictions with decreasing core sizes for one, two and three mode interactions. We tested core sizes ranging from the size used to generate responses ($4 \times 4 \times 4$) down to $2 \times 2 \times 2$. For BaTFLED CP models, the number of latent factors was set to the product of the dimensions of the core in the Tucker models. Values shown represent means across 10 replicates.

3.2.5 Insufficient power

Finally we wondered how BaTFLED would perform in a situation where there was insufficient power to learn the relationships in the data. Thus we generated responses using a model with 30 samples for each mode and removed two samples from each mode as test data, but now we trained using only a subset of the remaining 28 samples for each mode. We test models with 28, 25, 22, 19, 16, 13 and 10 samples for each mode.

From figures 3.12 and 3.13 we see that when predicting responses generated without higher-order interactions (1D), both BaTFLED models are more resilient to being underpowered than elastic net models. As may be expected, the improvement of LASSO models over ridge regression models vanishes when there are not enough samples to identify the true predictors. When two-dimensional interactions are present (1&2D), BaTFLED Tucker models outperform other methods with little degradation of performance until using just 13 samples. When interactions between all three modes are present, again BaTFLED Tucker consistently outperforms other methods, but more degradation is seen as the number of training samples decreases.

Recall that for this example there were 10 true predictors and 90 nuisance variables. Figure 3.14 shows the performance of these models in identifying these predictors. It is encouraging that BaTFLED is able to identify the majority of the correct features with as few as 16 samples, but generally the number of true features is unknown. While the increased structure in BaTFLED models allows for an improved performance over elastic net methods, it seems that in order to robustly identify predictors and make good predictions, training samples should number at least twice the number of true predictors.

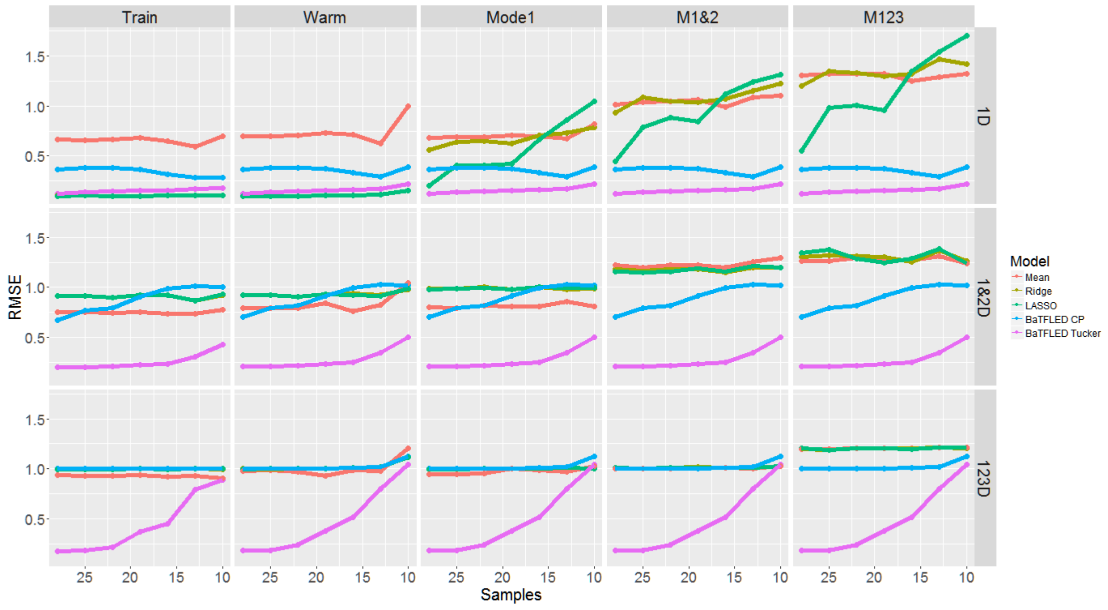


Figure 3.12: Normalized root mean squared error for BaTFLED, ridge and LASSO regression methods when making predictions with decreasing number of training samples. Values shown represent means across 10 replicates.

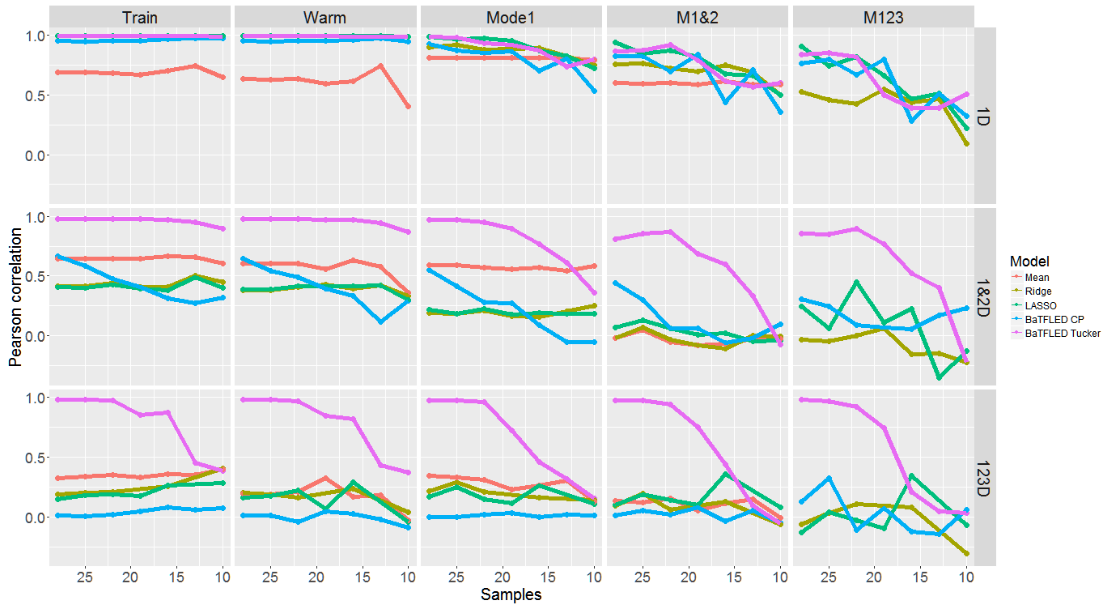


Figure 3.13: Pearson correlation for BaTFLED, ridge and LASSO regression methods when making predictions with decreasing number of training samples. Values shown represent means across 10 replicates.

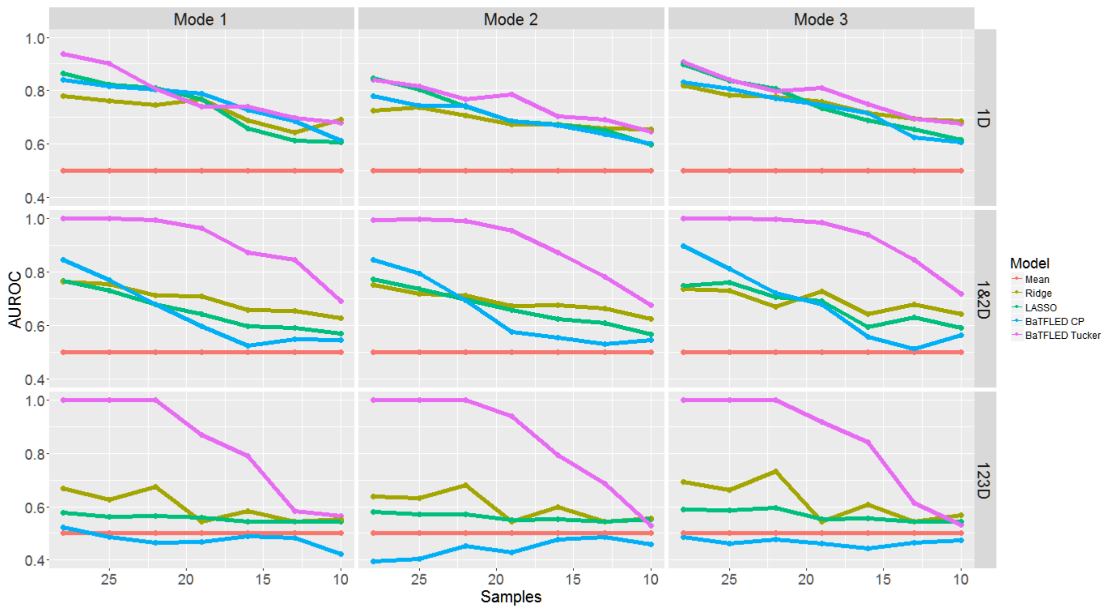


Figure 3.14: Area under the receiver operator characteristic curve for BaTFLED, ridge and LASSO regression methods with decreasing number of training samples. The theoretical value for random guessing of 0.5 is shown for mean prediction. There were 10 true and 90 nuisance variables for each mode. Values shown represent means across 10 replicates.

3.2.6 Conclusions

Using simulated data, we are able to obtain a clear understanding of the strengths of the BaTFLED algorithm. First, we see that while other methods are able to perform well when there are only one-dimensional interactions in the data, for cold-start tasks, allowing higher order interactions causes these models to perform on par with the baseline of simply predicting the mean across training samples. Second, on warm-start tasks, the non-linear models are able to significantly outperform mean prediction and elastic net models, but BaTFLED Tucker models are the best performing in all cases. Third, BaTFLED CP models do not show any significant advantages over other methods which is particularly interesting since most tensor factorization work to date has used CP the decomposition. Fourth, the performance of Tucker models is very robust to missing responses, added response noise, sub-optimal core size and insufficient power when compared to elastic net models. Finally we see that given sufficient training data, BaTFLED models are able to correctly identify the features influencing response.

However, for a given dataset, it is unclear whether there are higher-order interactions influencing responses. These experiments give some insight into what performance would be expected for these different algorithms if higher-order interactions are not present. Namely, we would expect that elastic net models would perform significantly better than mean prediction and that BaTFLED Tucker models would perform similarly to elastic net models and better than BaTFLED CP models. If instead we find that elastic net models perform no better than mean prediction, and we are not missing more than half of the responses, we would expect BaTFLED Tucker models to perform better than these models regardless of the core size provided that the noise is Gaussian and we have sufficient training data.

Chapter 4

Drug response datasets

In this chapter we explore the performance of the BaTFLFED algorithm on three different datasets assessing the growth of cancer cell lines when treated with drugs at varying doses. The first dataset was used in a public competition as part of a dialog for reverse engineering (DREAM) challenge (specifically the DREAM7 NCI-DREAM Drug Sensitivity Prediction Challenge [24]). In this competition teams sought to predict the GI_{50} response measure for 52 breast cancer cell lines when treated with 26 drugs. Here, we instead predict the response at each of 9 doses. The second dataset, referred to as the 'Heiser data' is similar in that it consists of a panel of breast cancer cell lines, was generated by the same group and has 9 measured doses. Although this data contains some of the same responses measured for the DREAM challenge, the data have been cleaned and expanded to include 71 cell lines and 108 drugs. Also, the responses for this datasets are normalized to control for the growth rate of the cell lines using a recent algorithm from Hafner et al. [22]. Additionally on this dataset we show results for predicting summary response measures like GI_{50}. The third dataset, produced by the Cancer Target Discovery and Development (CTD^2) effort, is the largest publicly available cell line/drug response dataset that has been produced to date [108] and contains 907 cell lines across 24 cancer types and 545 different compounds measured at 8 doses.

4.1 Prediction tasks

For each of these datasets we measure performance for four different prediction tasks. For warm-start prediction we predict the responses for cell line/drug combinations where responses are known for the given cell line and drug, but not the particular combination. Like with the simulated data, this is generally the easier than cold-start prediction tasks since information can be leveraged from known responses rather than relying only on the input features for cell lines and drugs. However, the setup differs slightly from the warm-start case shown in chapter 3 since responses are missing at all doses for a given cell line/drug combination. The three cold-start tasks are: 1) predict a suite of responses for a new cell line across all training drugs and all doses, 2) predict a suite of responses for a new drug across all training cell lines and doses and 3) predict the response at each dose for a cell line/drug combination where no responses for cell line or drug are present in the training data.

4.2 Passing features through kernel functions

Since BaTFLED cannot model interactions between features of the same mode, the predictive performance can often be improved by using kernel functions. Here instead of feeding the input features directly into the model, the sample$\times$feature matrix is first transformed into a matrix of sample$\times$sample similarity scores in the $0-1$ range. If features are binary then we use the Jaccard similarity coefficient. This measure gives the proportion of positive indicators shared between two samples out of the total number of positive indicators in either sample. If $\mathbf{x}$ and $\mathbf{y}$ are indicator row vectors, then

$$J(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\mathbf{y}^T}{\mathbf{x}\mathbf{x}^T + \mathbf{y}\mathbf{y}^T - \mathbf{x}\mathbf{y}^T} \tag{4.1}$$

If the features are continuous measures then we use the the Gaussian kernel which is defined as

$$G(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\right) \tag{4.2}$$

where σ can be adjusted to improve performance. When choosing σ parameters, we begin with σ equal to the average of the euclidean distances between vectors in the matrices and scale this value by a parameter s. Larger values of s increase the similarity between vectors in a distance-dependent fashion. When multiple different types of data are used as input features, we calculate kernels separately for each data type. These kernels can be combined by concatenation, averaging or by taking their product.

4.3 DREAM7 data

The NCI-DREAM Drug Sensitivity Prediction Challenge challenge [24] provided contestants with GI_{50} response data for 35 cell lines treated with 28 drugs. Models trained on this data were used to predict GI_{50} rankings (for each drug) on an additional 18 cell lines. Rather than predicting this summary measure, we compare the performance of BaTFLED to baseline methods when predicting normalized responses at each dose. We show results for 10-fold cross-validation runs on the 35 training cell lines when predicting the four warm and cold start scenarios listed above. Additionally, we show results for the final 18 cell lines designated as test data in the original competition. We also report on the importance of input features for both cell lines and drugs for the various methods. In the subsections below we give details on how the data were cleaned and processed. Since the task of predicting at each dose is new, the cell lines were not uniformly characterized, and our method utilizes features for drugs, we had to make several choices in processing the data that would likely have effects on the results.

4.3.1 Cell line data

The dream challenge provided information on the subtype for each cell line as well as mutation, expression, methylation, reverse phase protein array (RPPA), and single nucleotide polymorphism array data. The full DREAM dataset contained 83 cell lines with the following subtype categories: basal (27), claudin low (10), luminal (35), matched normal (2), normal like (7) and unknown(3) which were encoded as binary indicators. In addition, the mutational status of ERBB2 was also given and encoded as a binary indicator (19 of cell lines were positive).

Whole-exome DNA sequencing data assessing $32,293$ mutations in 50 of the cell lines was provided. We coded these data a a binary matrix indicating whether each of $11,995$ genes were found to be mutated in each of the cell lines. The number of mutated genes per cell line ranges from 92 in LY2 to $1,880$ in MDAMB361. We removed 12 genes with names that look like dates (perhaps due to auto-conversions in excel) and $4,788$ genes that are mutated in only one cell line. The mutation profile for several of the cell lines are highly correlated (Pearson correlation for MDAMB175VII and MDAMB157: 0.999, MCF10A and MCF10F: 0.817).

Expression profiles for $18,632$ genes in 46 cell lines was also provided through the competition. The expression values range from 1.37 to 13.92 and these values are z-scaled for each gene before use. Pearson correlations between cell lines range from -0.424 to 0.566.

Additionally the competition provided methylation data ranging from 0 to 1 for $27,551$ CpGs and 41 cell lines. Again a few of the cell lines have high Pearson correlations (HCC2185 and HCC1187: 0.974, LY2 and MCF7: 0.960, SKBR3 and AU565: 0.904).

To assess protein expression, reverse phase protein array (RPPA) data was provided for 42 cell lines and 66 proteins. The values range from -4.9 to 8.9 and were z-scaled across each protein before use. Correlations of the scaled data range from -0.474 to 0.757.

Finally, single nucleotide polymorphism array data (Affymetrix SNP6) was provided for $27,234$ loci and 47 cell lines in order to assess copy number changes. These values range from -6.44 to 3.58 and were also z-scaled before use. There were $1,436$ (0.11%) missing values which were set to zero after scaling. Pearson correlation between cell lines ranged from -0.344 to 0.748.

All together there is are $80,690$ potential features that could be used for prediction. Of the 84 total cell lines, only 31 have data from all platforms. Since, the current implementation BaTFLED algorithm cannot handle missing values in the input matrices and setting a large portion of the features to mean values for cell lines without data will substantially increase the similarity of these cell lines for non-biological reasons, we choose to only use a subset of this data for our tests.

We restrict the input features to use only the subtype, mutation and expression datasets which leaves us with 52 cell lines and $30,617$ features. Of these, 17 cell lines are in the final test set as split by the DREAM organizers. Among all cell lines, 7 do not have expression data (2 of which are in the test set) and 2 do not have mutation data (both are in the training data). This is still a very large number of features given the small number of cell lines available, so we further restricted the mutation and expression data to genes known to be involved in cancer. To this end, we downloaded a list of 572 cancer-related genes from the Catalogue Of Somatic Mutations In Cancer (COSMIC) database [109]. By restricting the above data to just these genes, we reduce the number of of mutation features to 387 and expression to 554. To obtain our final set of features for cell lines, we add to these features, indicators of subtype and ERBB2 mutation status, the mean and standard deviation of the doubling time for untreated cells and the total mutation counts. We remove any indicators that are not present in at least two samples and, in order to reduce the redundancy of features, we combined highly correlated features (Pearson correlation $> .95$) by averaging their values across cell lines (after z-scaling). This leaves us with a final matrix of 52 cell lines and 733 features (6 subtype indicators, 177 mutation features and 550 expression features).

4.3.2 Drug data

In the initial DREAM challenge competition the names of the drugs were not given, so it was not possible to get features to use for predicting responses to new drugs. After the contest was complete, the names of the 28 drugs were released along with the publication [24].

We used known drug targets and structural features computed from the chemical formula as features for these compounds when available. Two of these drugs (Cetuximab and Trastuzumab) are antibodies, so chemical formulas are not available and we removed these from our analysis. One additional drug is a combination of two other drugs in the study (4-HC+Dox), so we could not compute structural features and these were set to mean values. Target indicators and fingerprint values for this drug were taken to be the union of the indicators for the two drugs separately. For all drugs, we used the given CAS drug identifier to look up the CID identifier in PubChem. When there were multiple CID identifiers for a given CAS number, we use the CID with the most bioassay records (AIDs). The CID identifiers were used to get PubChem fingerprints and SMILES (Simplified Molecular-Input Line-Entry system) strings. The fingerprints are vectors of 881 binary values indicating whether specific chemical structures are present in the drug. Indicators with little discriminatory power (present in 0, 1, 25 or 26 drugs) were removed. The SMILES string were fed into Padel software [110] to generate a suite of $1,444$ chemical features detailing bond counts, angles, aromaticity measures and others. Features with no variation across drugs (419), those containing 'Inf' or 'NA' values (349), only one non-zero value (7) or values over 10^{10} (4) were removed. Of the remaining 665 features, non-binary features were z-scaled and highly correlated features (Pearson's $r^2 > 0.9$) were combined.

Additionally, we built binary indicator features of drug classes, mode of action and gene targets for the compounds when possible. There were 7 drug classes including regulation, cell cycle, metabolism, proteasome, signaling/growth, signaling NFkB and signaling RTK and 22 modes of action primarily listing pathway targets. Gene targets (29 total) were obtained by searching information in DrugBank, TTD (Therapeutic Targets Database), and GDSC (Genomics of Drug Sensitivity in Cancer) databases but as the majority of these are only targeted by one drug, only 6 are useful as features (ABL, KIT, PDGFR, BCR-ABL, IKK and PTS). Overall, these produced 36 indicators and after removing 1 class and 20 mode of action features due to lack of discriminatory power we are left with 15. Again, we merged together predictors that were highly correlated (Pearson correlation greater than 0.9 or less than -0.9) and were left with a final set of 433 features for the 26 drugs.

4.3.3 Response data

Although the Costello et al. study did provide the raw response data at each dose, this data was not in a form that could be directly used in the BaTFLED model. The majority of cell line/drug combination were tested at 9 doses plus a zero-drug control in three replicates. But, of the $53 \times 28 = 1,484$ cell line/drug combinations 556 (37.5%) were tested in multiple times (in up to 6 different runs) and 192 (12.9%) were not tested. The GI_{50} values that were used in the competition were geometric mean among repeated experiments that passed quality control measures or were set to the highest dose if no response was observed. In order to get data that was treated uniformly across the whole set, we selected raw data for each cell line/drug combination from one experiment attempting to match the raw data with the reported GI_{50} values when possible. To this end, we first normalized the raw responses to the no-drug control using the formula cited in Costello et al. [24] and given in the equation below. N_i is the normalized value at dose i, T_i is the raw optical density (od) response at dose i and time 72 hours minus the background od at 72 hours, T_z is the median od at time zero minus the background od at time zero and C is the median od with no drug at 72 hours.

$$N_i = \begin{cases} \frac{T_i - T_z}{C - T_z}, & \text{if } T_i \geq T_z \\ \frac{T_i - T_z}{T_z}, & \text{if } T_i < T_z \end{cases} \tag{4.3}$$

We then fit four-parameter logistic curves to the normalized responses and chose the data that had the best fit value provided that the median normalized responses across replicates were within the range $(-2, 2)$. About 70% of curves were able to be fit $(1,440/2,075)$. When no curves were could be fit for a given combination, we used the data with the highest plate number, assuming these were more recently run. Four drugs were tested with different doses depending on the cell line. For these, we fit Loess curves to these data and translated the responses along the curve to the more common set of doses. In all, we are able to collect raw responses at each of 9 doses for $1,138$ of the $1,484$ combinations (76.7%). For this dataset, we tested the performance of BaTFLED and other models when predicting the normalized responses (median across replicates). Box plots of responses at each dose are shown in figure 4.1.

4.3.4 Cross-validation results

For experiments on the DREAM data, we compare BaTFLED CP and Tucker models to elastic nets, random forests and neural networks on both warm start and cold start tasks. For the 10-fold cross-validation runs, each fold contained 31 cell lines, 23 drugs and all 10 doses (including the zero-dose). Cold-start predictions were made for 4 cell lines, 3 drugs and a random warm-start set of 1% of cell line/drug combinations across all doses. In this case, all types of testing were performed simultaneously. CP models were run with 200 columns in the latent matrices while Tucker models were given $10 \times 10 \times 10$ cores.

Separate models are trained for each dose for elastic net, random forest, and neural net models. The inputs for each method are vectors containing the 733 features for cell lines and 433 features for drugs concatenated together. Elastic net models were run using the R package ‘glmnet’ and while the ‘h2o’ package was used for random forest and neural net

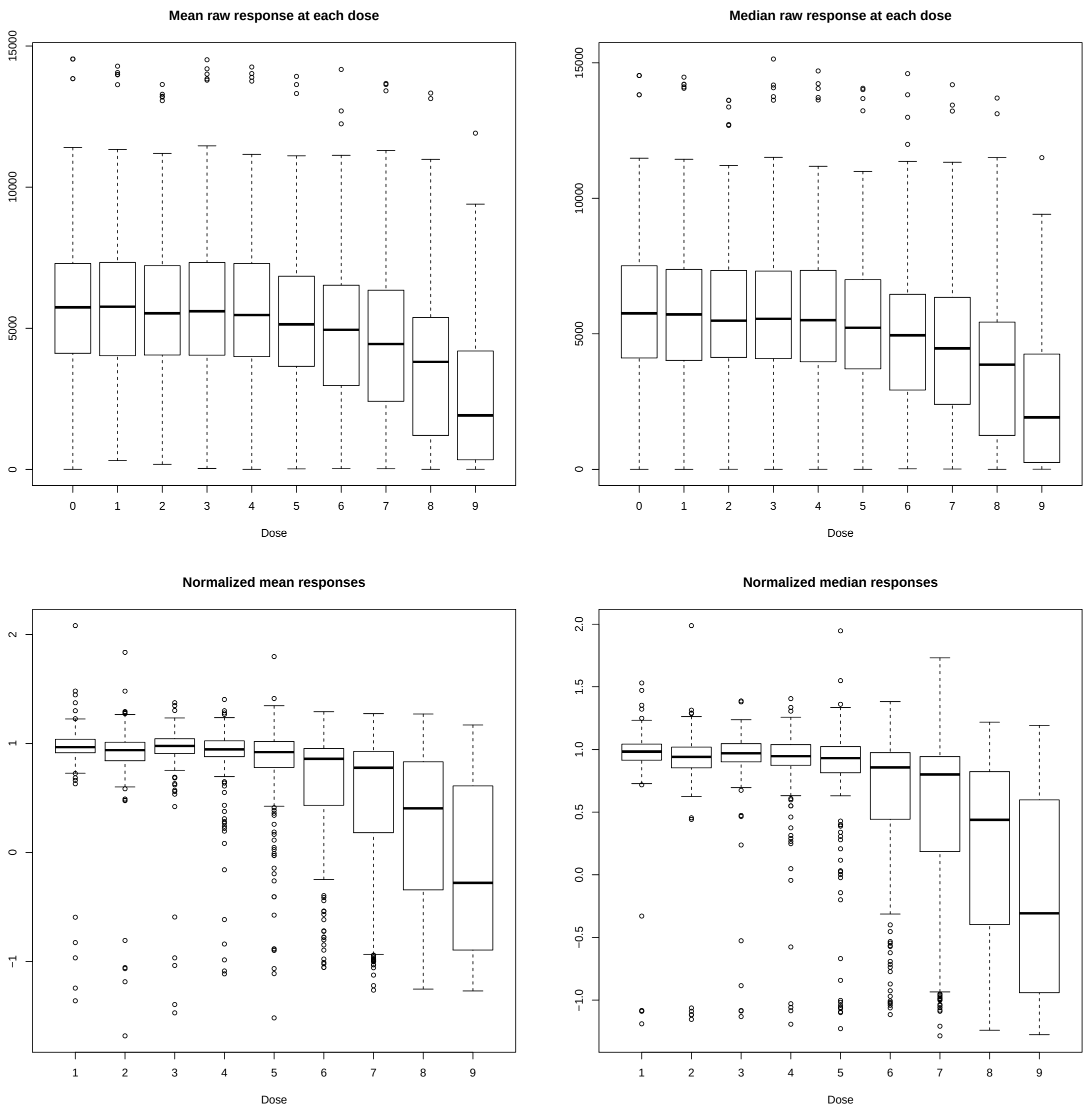


Figure 4.1: Box plots showing raw and normalized responses at each dose for the DREAM data.

training. The same model parameters that were used on the simulated data were tested here (see section 3).

Since this data has more input features and fewer total responses than the simulated data, we believe that it is somewhat underpowered for this prediction task. Experimentation with artificially underpowered simulated datasets suggest that this can be partially overcome by performing multiple rounds of training to reduce the number of features. For the the cross-validation runs we found the best performance by first training BaTFLED models for 200 iterations with strong sparsity priors ($\alpha = 10^{-10}$ and $\beta = 10^{10}$), keeping the union top 15% of predictors across folds (278 and 152 cell line features for CP and Tucker methods and 163 and 105 drug features for CP and Tucker methods), and retraining for another 200 iterations without encouraging sparsity ($\alpha = 1$ and $\beta = 1$). Since the features used in the second round of training for a given cross-validation fold are selected based on folds that include the current test data in the training set, there is some information leakage in this approach, however, no such leakage occurs in the results for the final test set shown below. The average performance across folds in terms of normalized root mean squared error, Pearson correlation and Spearman correlation are given in tables 4.1 and 4.2, and figure 4.2.

While these methods appear mostly comparable with the retrained Tucker model showing some advantages, we are also interested in how the methods perform when predicting individual cell lines, drugs or combinations of the two. Figure 4.3 shows the performance measures for each task. Each point represents one cell line, drug or cell line/drug combination when predicted in a cold-start cross-validation run. Additionally, we show Pearson correlation results for the best performing method from each group compared to mean prediction in figures 4.4, 4.5 and 4.6. These plots show that a greater number of individual cell lines and drugs are predicted better by BaTFLED than by other methods and that for individual cell lines and drugs, the predictions represent a larger improvement over mean prediction than other methods. For example, when predicting for cold-start cell lines, LASSO performs better than mean prediction for 31% of cell lines, but these are only slightly off the diagonal and represent a minimal improvement. The retrained BaTFLED Tucker model performs better for 43% of cell lines and these are further from the diagonal, showing a larger improvement. This indicates that BaTFLED may be advantageous in particular settings, likely when higher order interactions are present. Moreover, since BaTFLED models the data in a fundamentally different way than other methods, it may also confer an advantage when combined with other methods in a consensus method.

Table 4.1: Mean normalized RMSE on DREAM dataset over ten cross-validation runs (sd).

	Training	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.72(0.01)	0.86(0.28)	0.83(0.11)	0.78(0.07)	0.97(0.10)
LASSO a=1	0.62(0.01)	0.70(0.19)	0.81(0.09)	0.78(0.09)	0.96(0.13)
E. net a=0.9	0.62(0.01)	0.67(0.15)	0.81(0.09)	0.78(0.09)	0.96(0.13)
E. net a=0.5	0.62(0.01)	0.61(0.12)	0.81(0.09)	0.78(0.09)	0.97(0.13)
Ridge a=0	0.64(0.01)	0.59(0.08)	0.85(0.08)	0.78(0.09)	0.99(0.12)
R.F. 1Kx5	0.55(0.01)	0.57(0.21)	0.85(0.21)	0.76(0.07)	0.89(0.21)
R.F. 5Kx5	0.55(0.01)	**0.53**(0.20)	0.85(0.21)	0.76(0.07)	0.89(0.21)
R.F. 1Kx10	0.50(0.01)	0.57(0.17)	0.85(0.22)	0.75(0.07)	0.89(0.21)
N.N. 1L	0.38(0.03)	0.69(0.18)	1.04(0.20)	0.86(0.12)	1.14(0.27)
N.N. 2L	0.39(0.02)	0.59(0.15)	0.87(0.20)	0.82(0.07)	0.89(0.23)
N.N. 3L	0.33(0.02)	0.56(0.21)	0.89(0.20)	0.82(0.05)	1.02(0.19)
CP	0.68(0.03)	0.77(0.30)	0.91(0.18)	0.78(0.06)	0.96(0.21)
CP retrain	0.68(0.02)	0.68(0.15)	0.84(0.17)	0.95(0.18)	1.02(0.45)
Tucker	0.29(0.02)	1.24(0.79)	0.83(0.19)	0.97(0.12)	0.99(0.25)
Tucker retrain	**0.27**(0.02)	0.90(0.56)	**0.75**(0.16)	**0.74**(0.07)	**0.81**(0.24)

Table 4.2: Mean Pearson correlation on DREAM dataset over ten-fold cross-validation (sd).

	Training	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.69(0.01)	0.66(0.14)	0.63(0.10)	0.66(0.06)	0.43(0.12)
LASSO a=1	0.79(0.01)	0.77(0.12)	0.62(0.09)	0.66(0.09)	0.48(0.14)
E. net a=0.9	0.79(0.01)	0.77(0.13)	0.62(0.09)	0.66(0.09)	0.48(0.13)
E. net a=0.5	0.79(0.01)	0.82(0.11)	0.62(0.09)	0.66(0.09)	0.47(0.14)
Ridge a=0	0.78(0.01)	**0.83**(0.07)	0.59(0.10)	0.65(0.09)	0.42(0.13)
R.F. 1Kx5	0.84(0.01)	0.78(0.11)	0.58(0.13)	0.65(0.09)	0.42(0.17)
R.F. 5Kx5	0.84(0.01)	0.82(0.10)	0.57(0.12)	0.66(0.07)	0.42(0.16)
R.F. 1Kx10	0.87(0.01)	0.82(0.06)	0.58(0.13)	0.66(0.08)	0.42(0.16)
N.N. 1L	0.93(0.01)	0.78(0.08)	0.49(0.10)	0.62(0.08)	0.33(0.15)
N.N. 2L	0.94(0.01)	0.78(0.05)	0.53(0.10)	0.58(0.08)	0.39(0.20)
N.N. 3L	**0.97**(0.01)	0.80(0.12)	0.51(0.13)	0.60(0.08)	0.29(0.20)
CP	0.73(0.03)	0.67(0.11)	0.48(0.09)	0.64(0.06)	0.34(0.13)
CP retrain	0.74(0.01)	0.63(0.14)	0.58(0.16)	0.57(0.13)	0.43(0.31)
Tucker	0.96(0.01)	0.62(0.36)	0.58(0.11)	0.56(0.10)	0.40(0.20)
Tucker retrain	0.96(0.00)	0.54(0.46)	**0.69**(0.08)	**0.70**(0.07)	**0.55**(0.24)

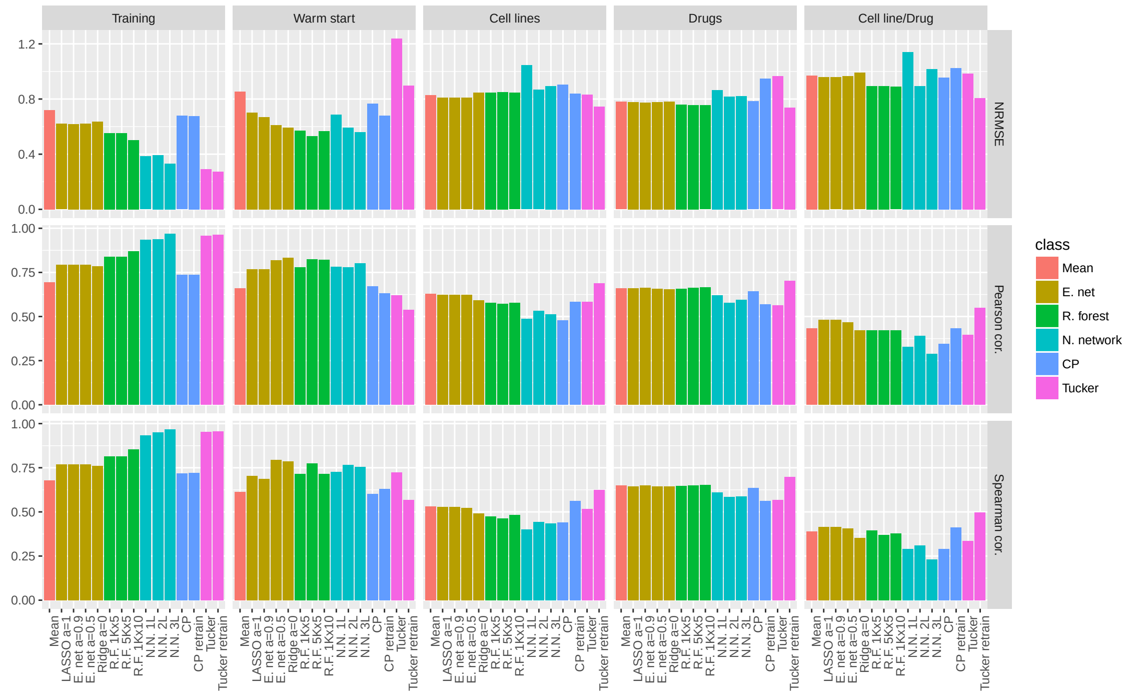


Figure 4.2: Performance measures for cross-validation runs on DREAM data (average across 10 cv runs). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown.

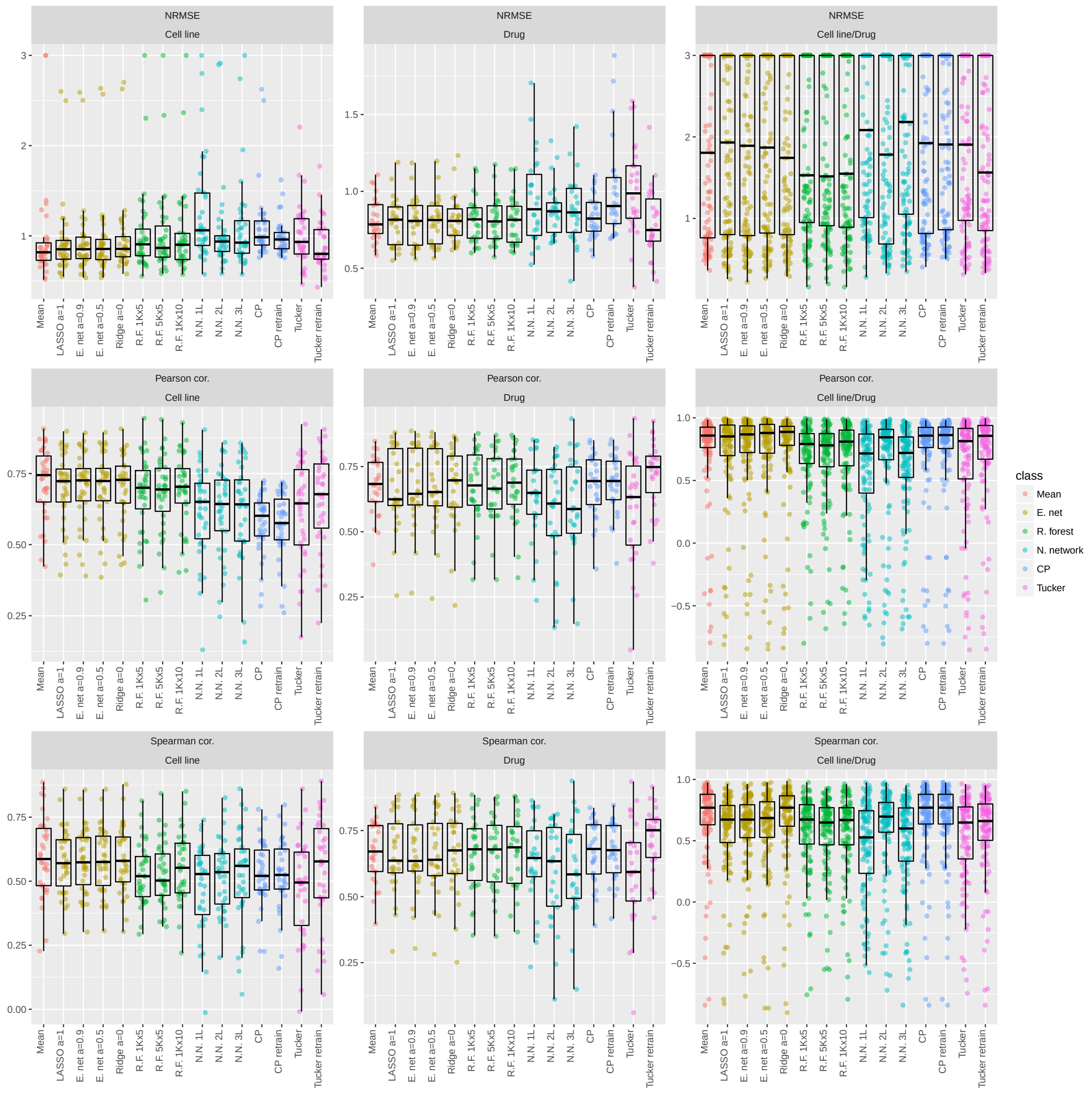


Figure 4.3: Performance measures for cross-validation runs on DREAM data. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

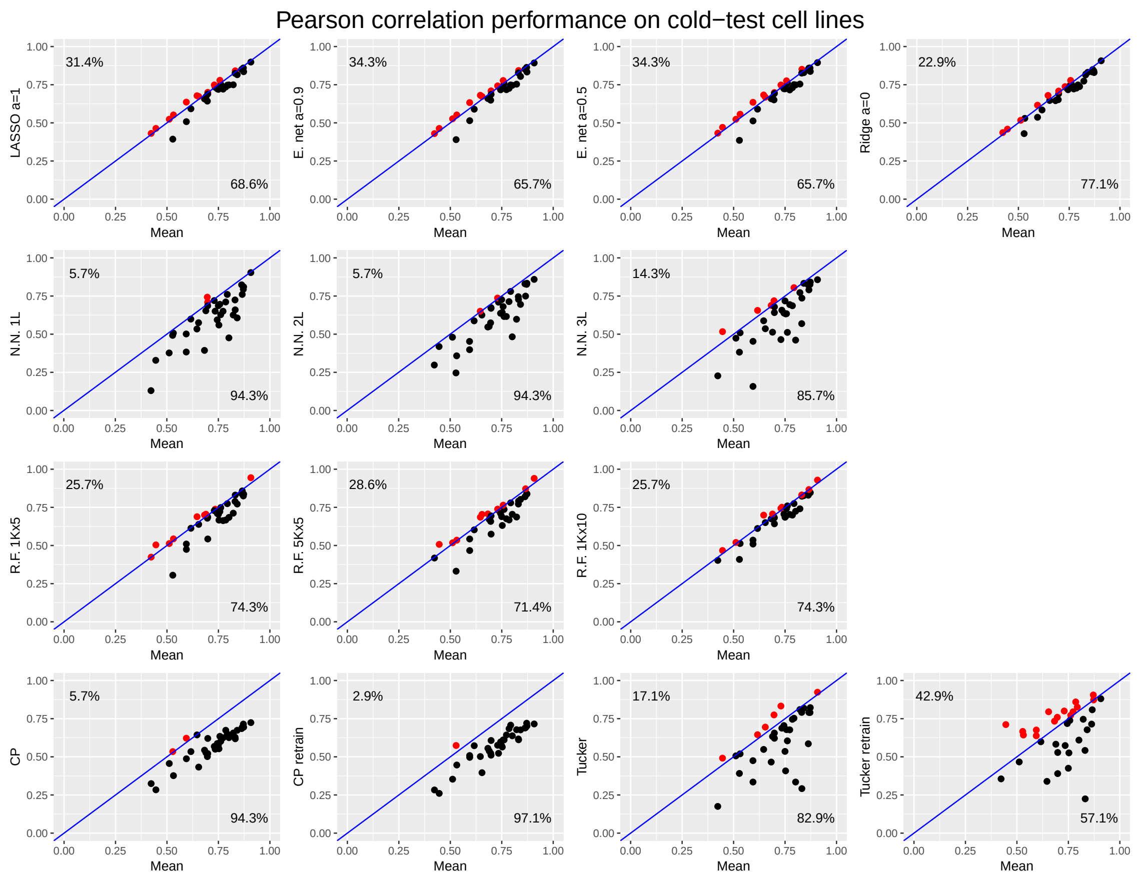


Figure 4.4: Comparison between mean prediction and other methods on the cold-start cell line task. Each point represents one cell line with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell lines on either side of the diagonal are annotated.

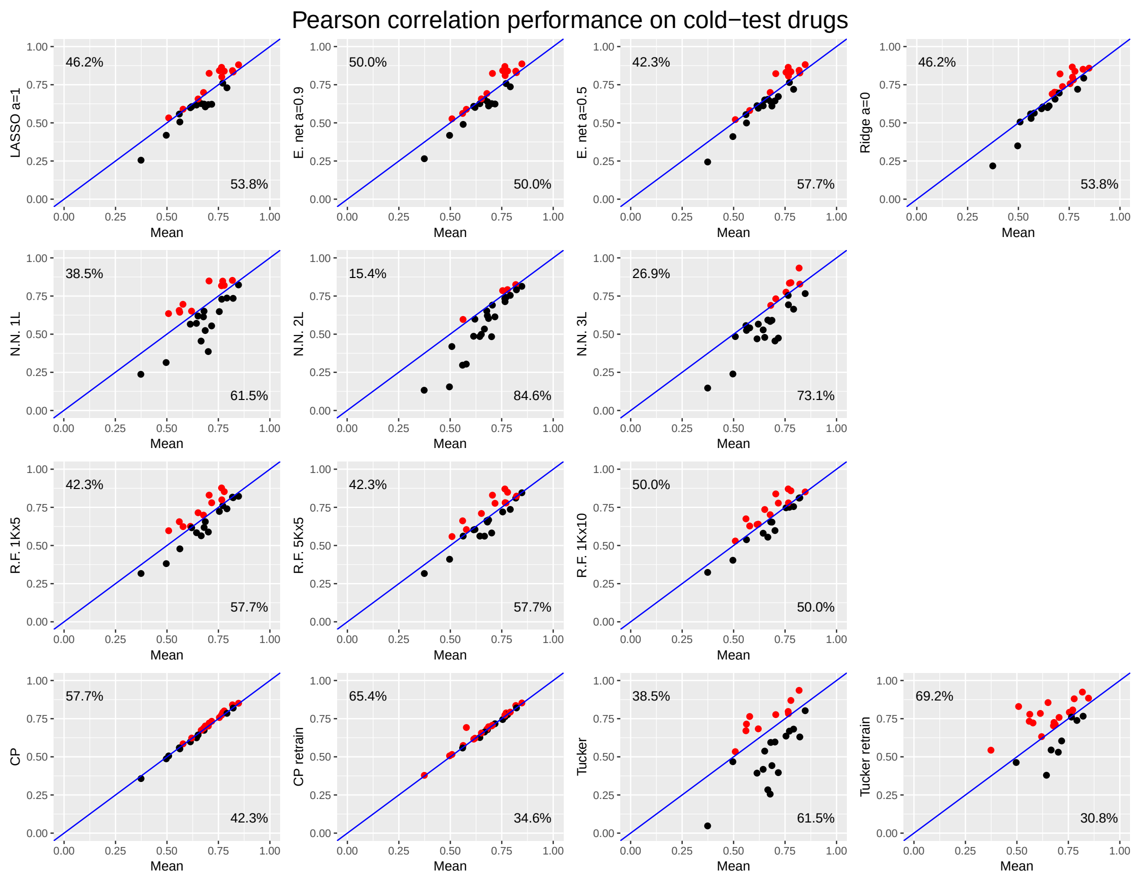


Figure 4.5: Comparison between mean prediction and other methods on the cold-start drug task. Each point represents one drug with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of drugs on either side of the diagonal are annotated.

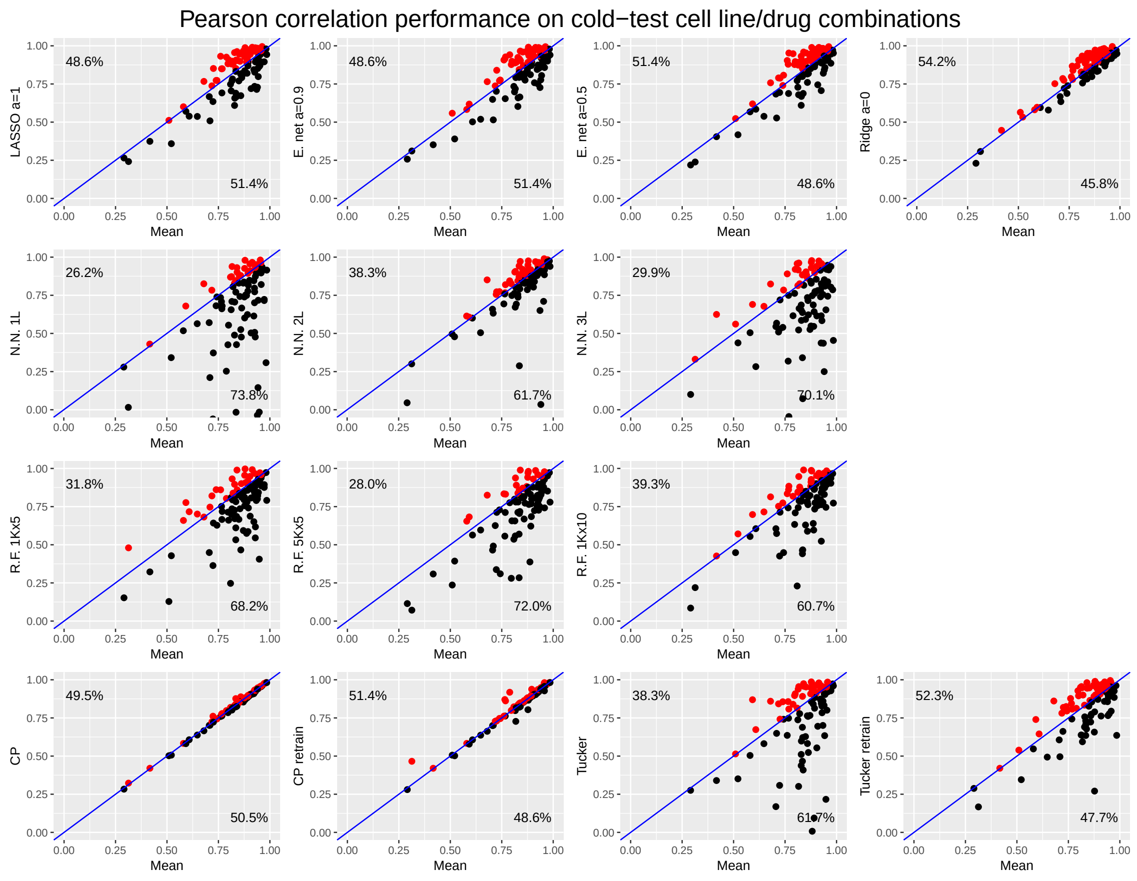


Figure 4.6: Comparison between mean prediction and other methods on the cold-start cell line/drug combination task. Each point represents one cell line/drug combination with its x coordinate showing the Pearson correlation of mean prediction (across cell lines) with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell line/drug combinations on either side of the diagonal are annotated.

4.3.5 Final test set results

As mentioned above, methods in the DREAM competition were judged by their prediction performance on a final test set of 17 cell lines. For this comparison, we trained BaTFLED and other models on the full set of 35 cell lines and 26 drugs and measured the cold-start performance on the test set cell lines. In this case, the two-round scheme for BaTFLED models consisted of training 10 models for 120 rounds with strong sparsity priors ($\alpha = 10^{-10}$ and $\beta = 10^{10}$), with different random initializations, selecting the union of the top 15% of predictors for the second round (330 and 191 cell line features for CP and Tucker models and 172 and 102 drug features for CP and Tucker models respectively), and training again for 120 rounds without encouraging sparsity ($\alpha = 1$ and $\beta = 1$). Figure 4.7 shows the mean performance across 10 replicates for BaTFLED and other models. Figure 4.8 shows the same data broken down by the predictive performance for the 17 individual cell lines and acoss these cell lines for each of the 26 drugs in the training data. BaTFLED models perform slightly better than other models on this very difficult challenge.

Table 4.3: Normalized RMSE on final 17 testing cell lines from DREAM dataset. Means over ten replicates (sd).

	Training	Warm	Cell lines
Mean	0.79(0.00)	0.81(0.17)	0.83(0.00)
LASSO a=1	0.61(0.00)	0.69(0.17)	0.83(0.00)
E. net a=0.9	0.61(0.00)	0.71(0.22)	0.83(0.00)
E. net a=0.5	0.62(0.00)	0.61(0.15)	0.82(0.00)
Ridge a=0	0.66(0.00)	0.70(0.14)	0.81(0.00)
R.F. 1Kx5	0.57(0.00)	0.69(0.17)	0.82(0.01)
R.F. 5Kx5	0.57(0.00)	0.65(0.17)	0.83(0.01)
R.F. 1Kx10	0.50(0.00)	0.63(0.14)	0.82(0.00)
N.N. 1L	0.40(0.02)	0.66(0.23)	0.83(0.01)
N.N. 2L	0.37(0.02)	0.68(0.08)	0.85(0.00)
N.N. 3L	0.31(0.01)	0.56(0.14)	0.88(0.01)
CP	0.30(0.01)	**0.51**(0.17)	**0.78**(0.00)
CP retrain	**0.16**(0.00)	0.59(0.19)	0.83(0.01)
Tucker	0.29(0.00)	0.79(0.61)	0.80(0.02)
Tucker retrain	0.29(0.00)	0.85(0.57)	0.80(0.02)

Table 4.4: Pearson correlation on final 17 testing cell lines from DREAM dataset. Means over ten replicates (sd).

	Training	Warm	Cell lines
Mean	0.61(0.00)	0.62(0.20)	0.57(0.00)
LASSO a=1	0.79(0.00)	0.75(0.14)	0.59(0.00)
E. net a=0.9	0.79(0.00)	0.75(0.18)	0.59(0.00)
E. net a=0.5	0.79(0.00)	0.80(0.14)	0.60(0.00)
Ridge a=0	0.77(0.00)	0.75(0.12)	0.60(0.00)
R.F. 1Kx5	0.83(0.00)	0.75(0.18)	0.57(0.01)
R.F. 5Kx5	0.83(0.00)	0.76(0.14)	0.57(0.01)
R.F. 1Kx10	0.87(0.00)	0.78(0.12)	0.58(0.00)
N.N. 1L	0.93(0.01)	0.77(0.20)	0.60(0.01)
N.N. 2L	0.94(0.01)	0.76(0.08)	0.57(0.00)
N.N. 3L	0.97(0.00)	0.84(0.10)	0.54(0.01)
CP	0.95(0.00)	**0.88**(0.08)	**0.62**(0.01)
CP retrain	**0.99**(0.00)	0.83(0.11)	**0.62**(0.00)
Tucker	0.96(0.00)	0.76(0.28)	0.61(0.02)
Tucker retrain	0.96(0.00)	0.72(0.30)	0.61(0.02)

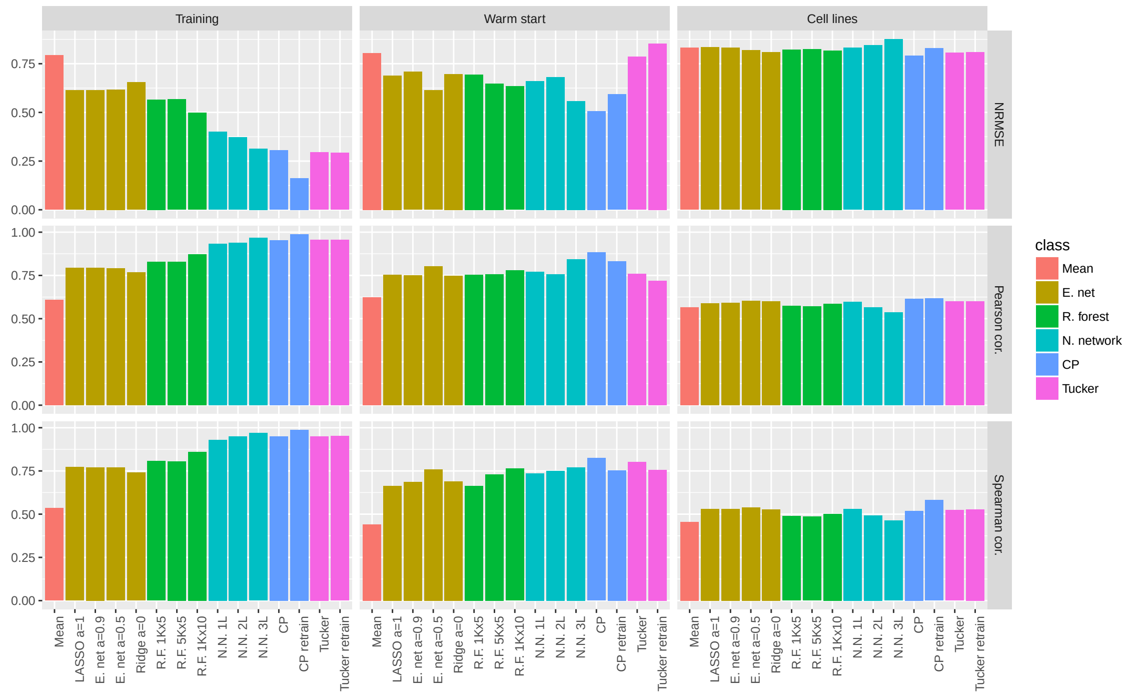


Figure 4.7: Performance measures on the final test cell lines from the DREAM data. Normalized root mean squared error, Pearson correlation and Spearman correlation for training data, warm-start and final test cell lines are shown. Bar height represents the mean over 10 replicates.

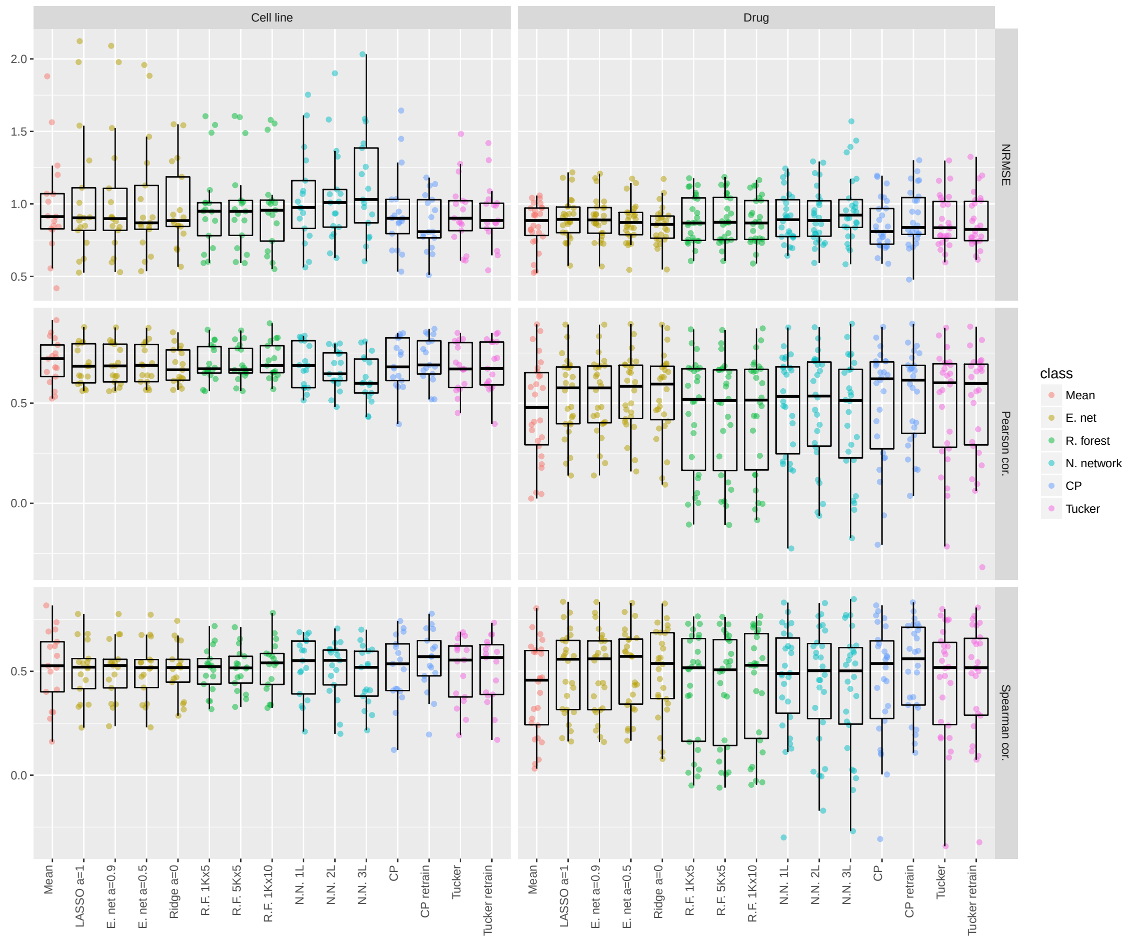


Figure 4.8: Performance measures on the final test cell lines for the DREAM data (mean across 10 replicates). Normalized root mean squared error, Pearson correlation and Spearman correlation for the 17 test cell lines are shown in the first column. The same measures for the predictive performance on these cell lines for each drug in the training data is shown in the second column.

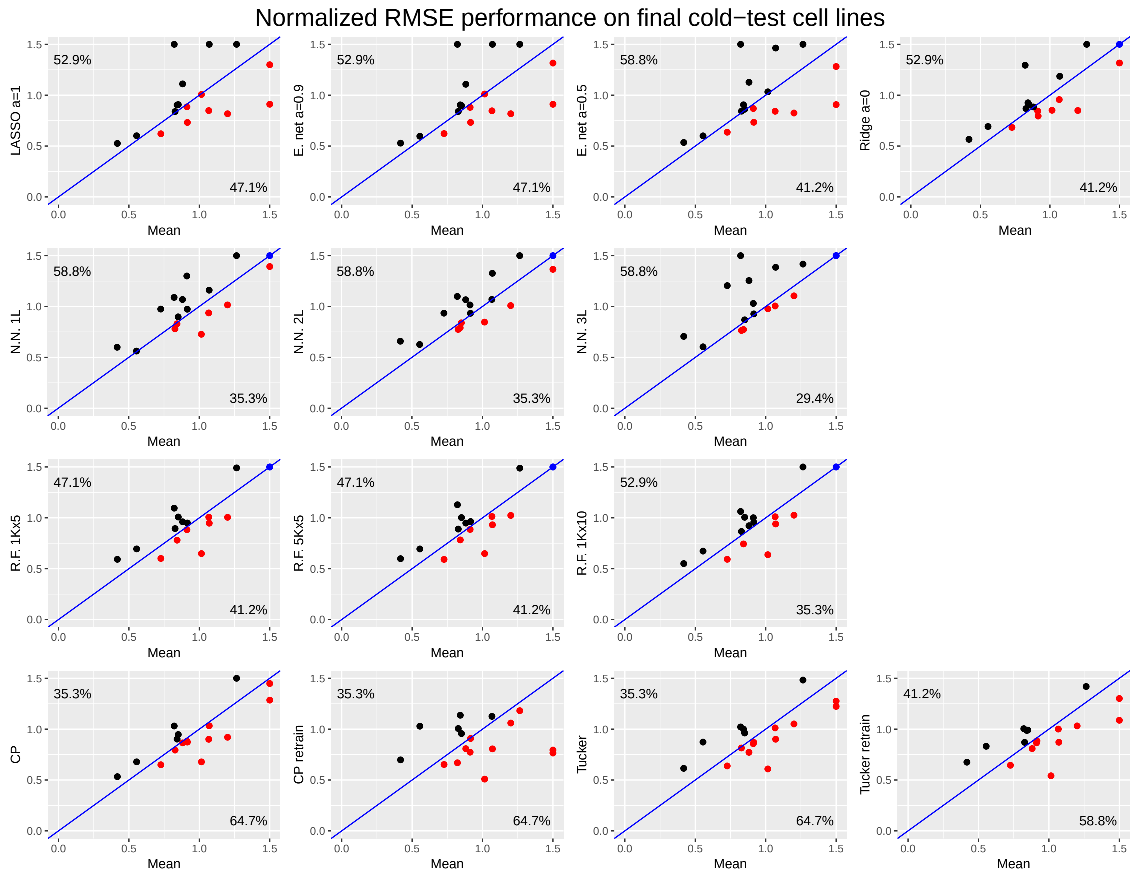


Figure 4.9: Comparison between mean prediction and other methods when predicting for 17 test cell lines. Each point represents one of the final test cell lines with its x coordinate showing the normalized root mean squared error (NRMSE) for mean prediction and the y coordinate showing the NRMSE for an alternate method. Red points are predicted better by the alternate method and the percentage of cell lines on either side of the diagonal are annotated. Values greater than 1.5 are displayed at 1.5

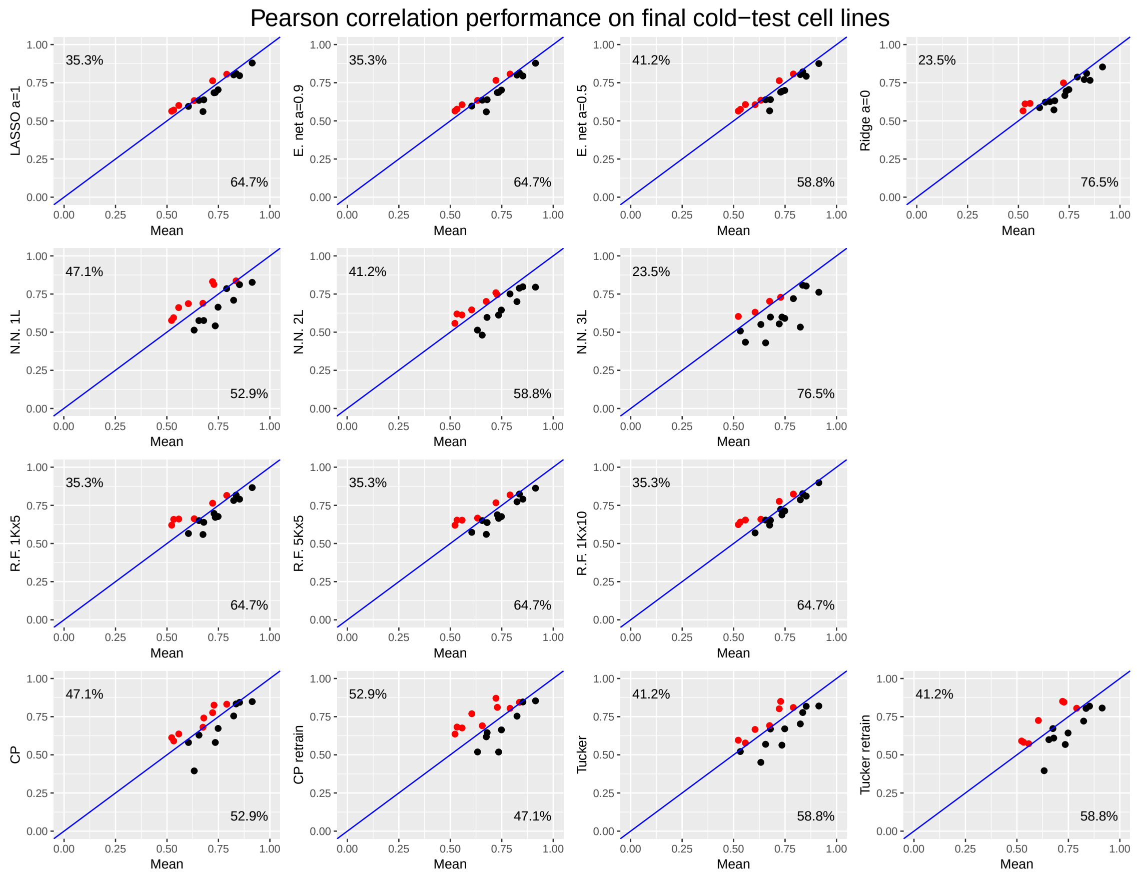


Figure 4.10: Comparison between mean prediction and other methods when predicting for 17 test cell lines. Each point represents one cell line with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of combinations predicted better by either method is shown.

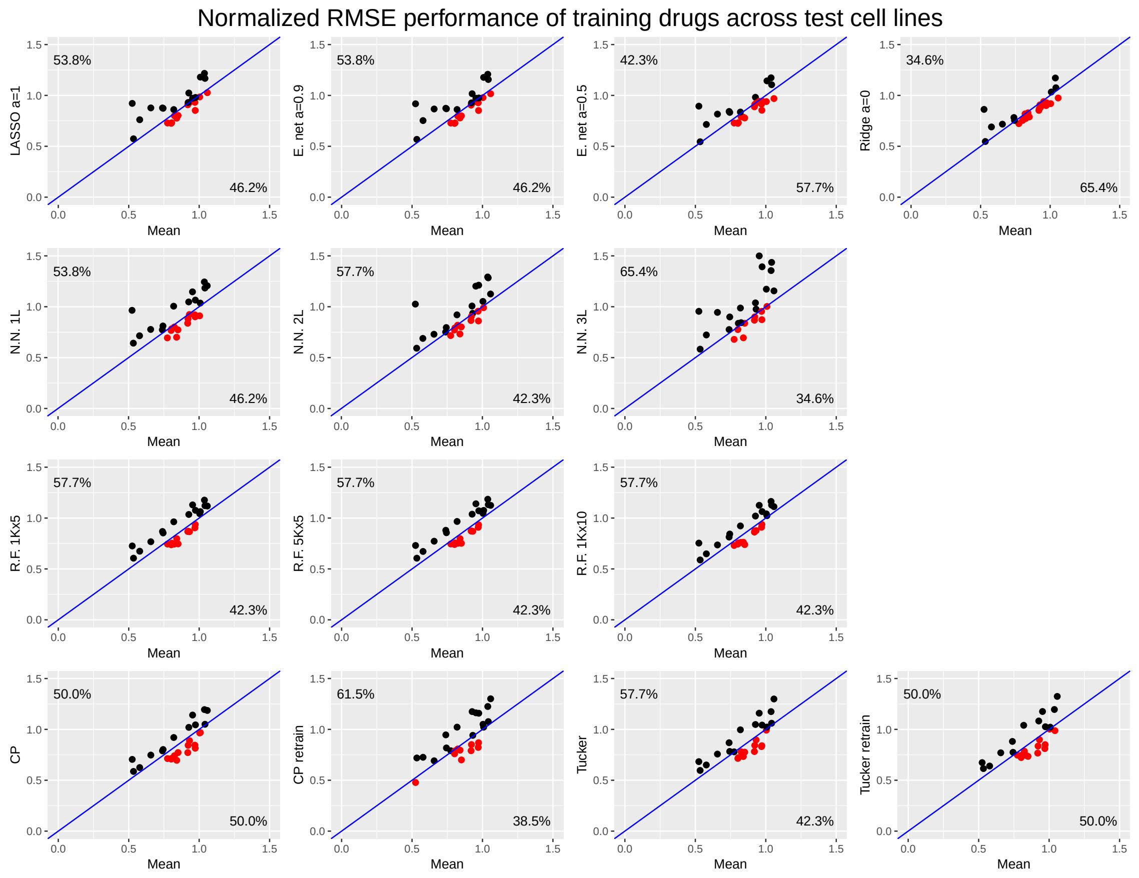


Figure 4.11: Comparison between mean prediction and other methods when predicting for 17 test cell lines. Each point represents one of the training set drugs with its x coordinate showing the normalized root mean squared error (NRMSE) for mean prediction across the test cell lines and the y coordinate showing the NRMSE for an alternate method. Red points are predicted better by the alternate method and the percentage of drugs on either side of the diagonal are annotated. Values greater than 1.5 are displayed at 1.5

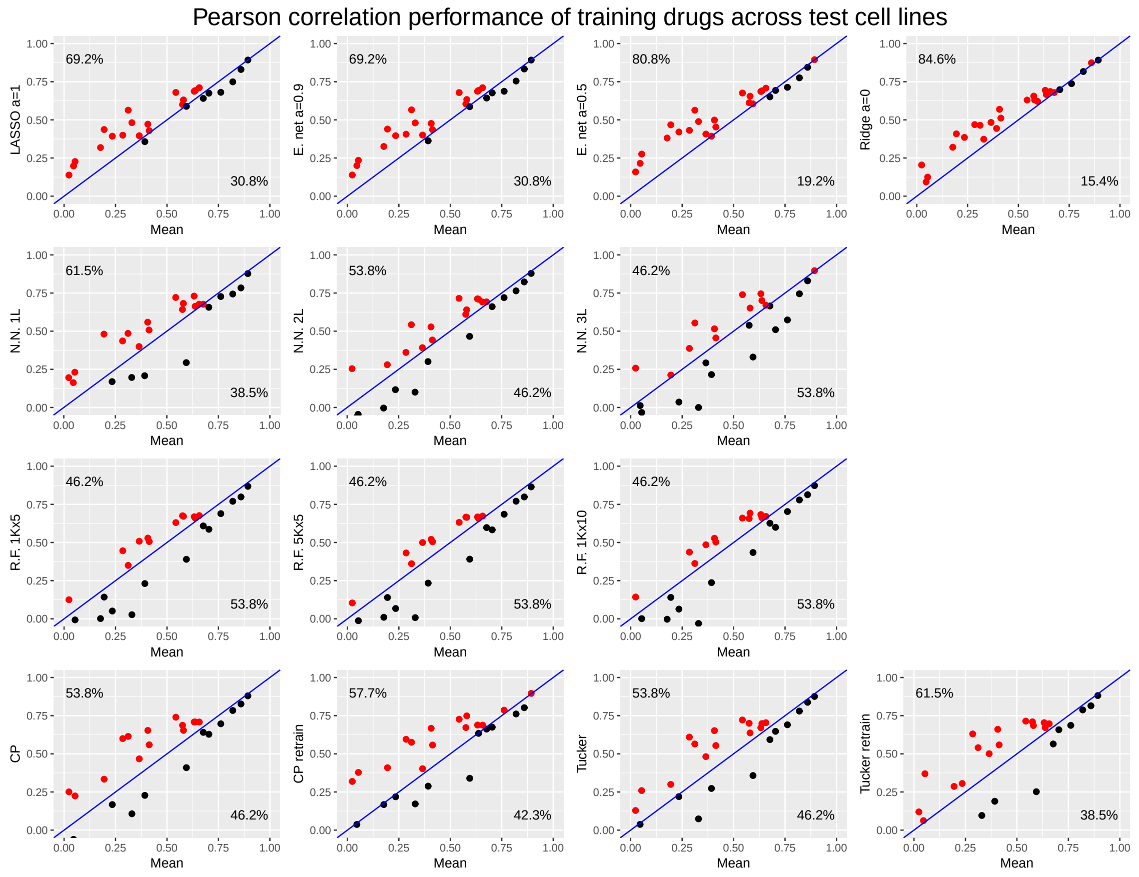


Figure 4.12: Comparison between mean prediction and other methods when predicting for 17 test cell lines. Each point represents one of the training set drugs with its x coordinate showing the Pearson correlation of mean prediction with the true response across the test cell lines and the y coordinate showing the Pearson correlation for the alternate method. Red points are predicted better by the alternate method and the percentage of combinations predicted better by either method is shown.

4.3.6 Feature selection

The majority of the methods tested above perform some features selection while training. Elastic net models (with the exception of ridge regression) choose features to include in the model explicitly, while continuous measures of feature importance can be extracted from neural networks and random forests. We compare which features were selected by the model from each category that performed best in terms of Pearson correlation on the final 17 cell lines. Among elastic net models, the run with $\alpha = 0.5$ performed best. The random forest model with $1,000$ trees of depth 10 performed best in its group and the single-layer neural network with $1,500$ nodes performed best in its group. Figures 4.13 and 4.14 show the proportion of features that were selected in the top 15% of features for each method grouped by the type of feature while figure 4.15 highlights these features on heatmaps. The elastic net model selected less than 15% of predictors in any run, so all selected features are shown.

For the cell line features, there are clear differences in which features were used by each method. The total number of features in the union of the top 15% across runs was highest for the BaTFLED Tucker and random forest models (329 and 230 respectively), indicating that these methods were using different features depending on the random initialization. For comparison, CP models selected 190 features, neural nets 130 and the elastic net models 59 features. Elastic net and random forest methods used almost no mutation features, while the neural net, CP and Tucker methods selected almost half of these. All methods use a relatively large portion of the 6 ‘other’ features (including subtype indicators and doubling time).

For drug features, all methods except for the Tucker model selected a similar number of features (103 for elastic net, 129 for random forest, 80 for neural net, 101 for CP and 171 for Tucker models). The types of features chosen differed greatly between methods, with the Tucker model choosing fairly evenly from the different types of features while the random forest model entirely excluded the 9 ‘other’ features.

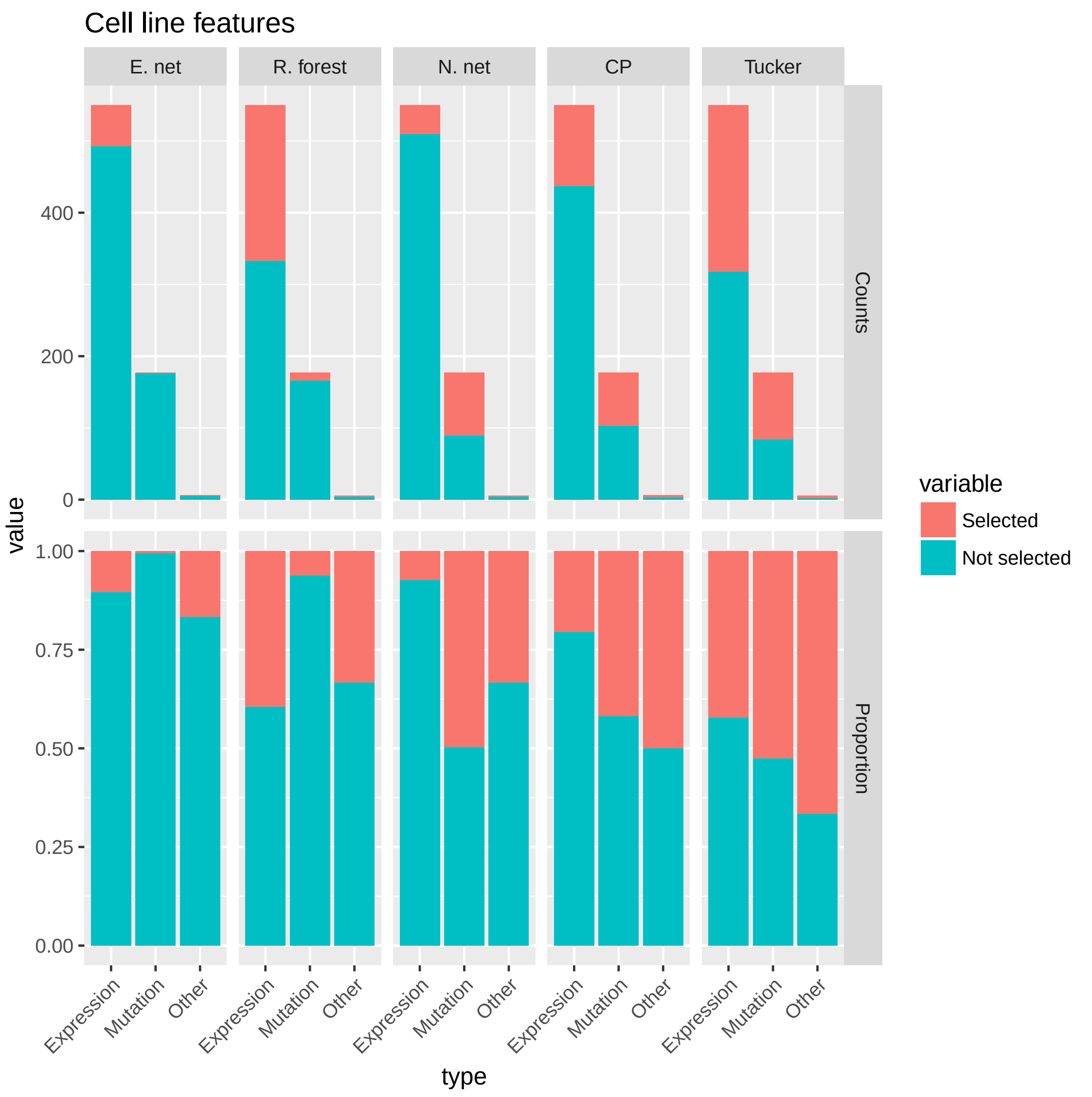


Figure 4.13: Top row: Number of cell line features selected by different methods for the DREAM data. Bar height shows the total number of features of the given type, red portion shows the number of these features in the union of the top 15% of features across replicates. Note that for the elastic net model, only about 5% of features had non-zero values in each replicate. Bottom row: Proportion of total features from each category that are chosen.

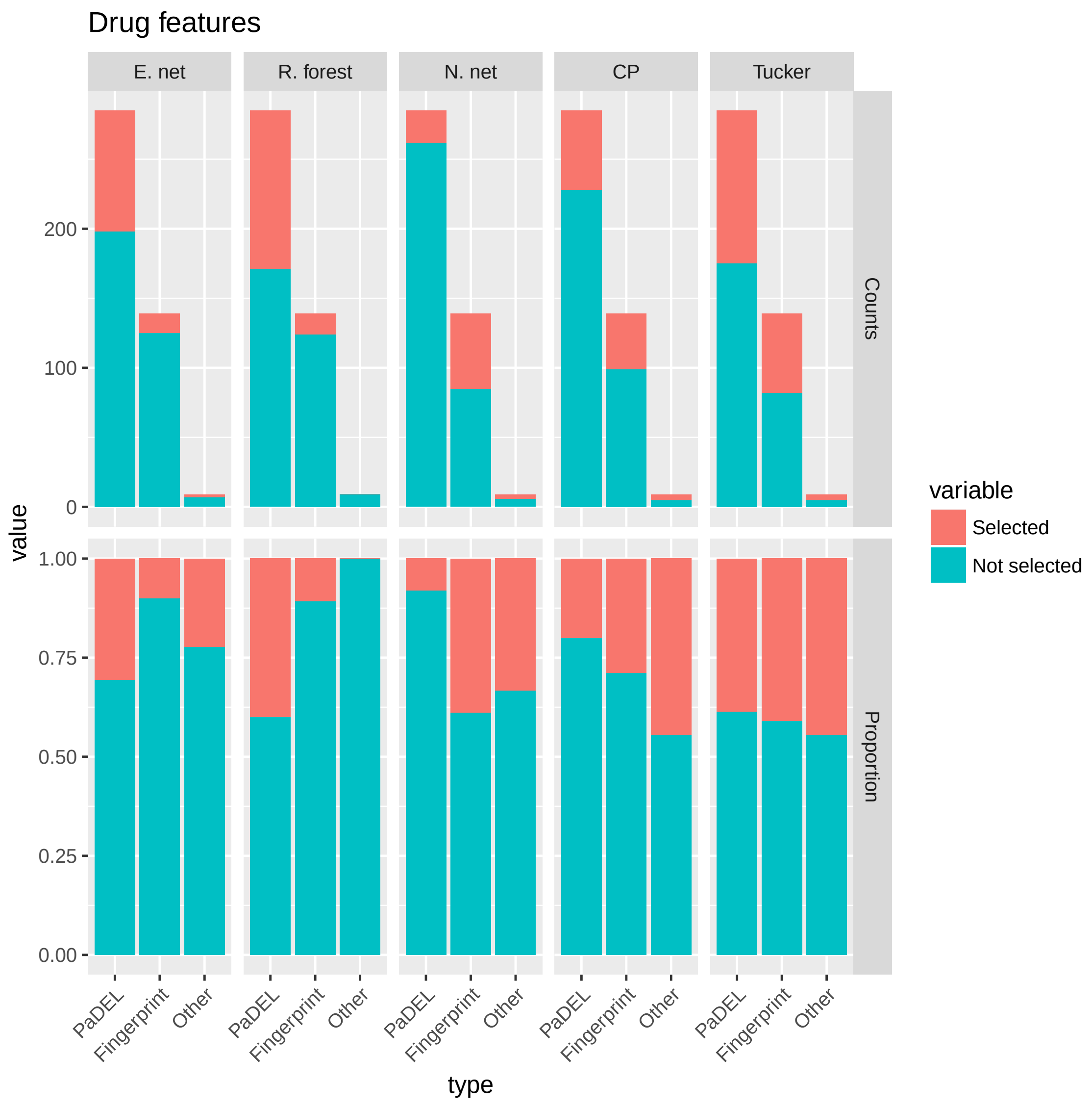


Figure 4.14: Top row: Number of drug features selected by different methods for the DREAM data. Bar height shows the total number of features of the given type, blue portion shows the number of these features in the union of the top 15% of features across replicates. Note that for the elastic net model, only about 5% of features had non-zero values in each replicate. Bottom row: Proportion of total features from each category that are chosen

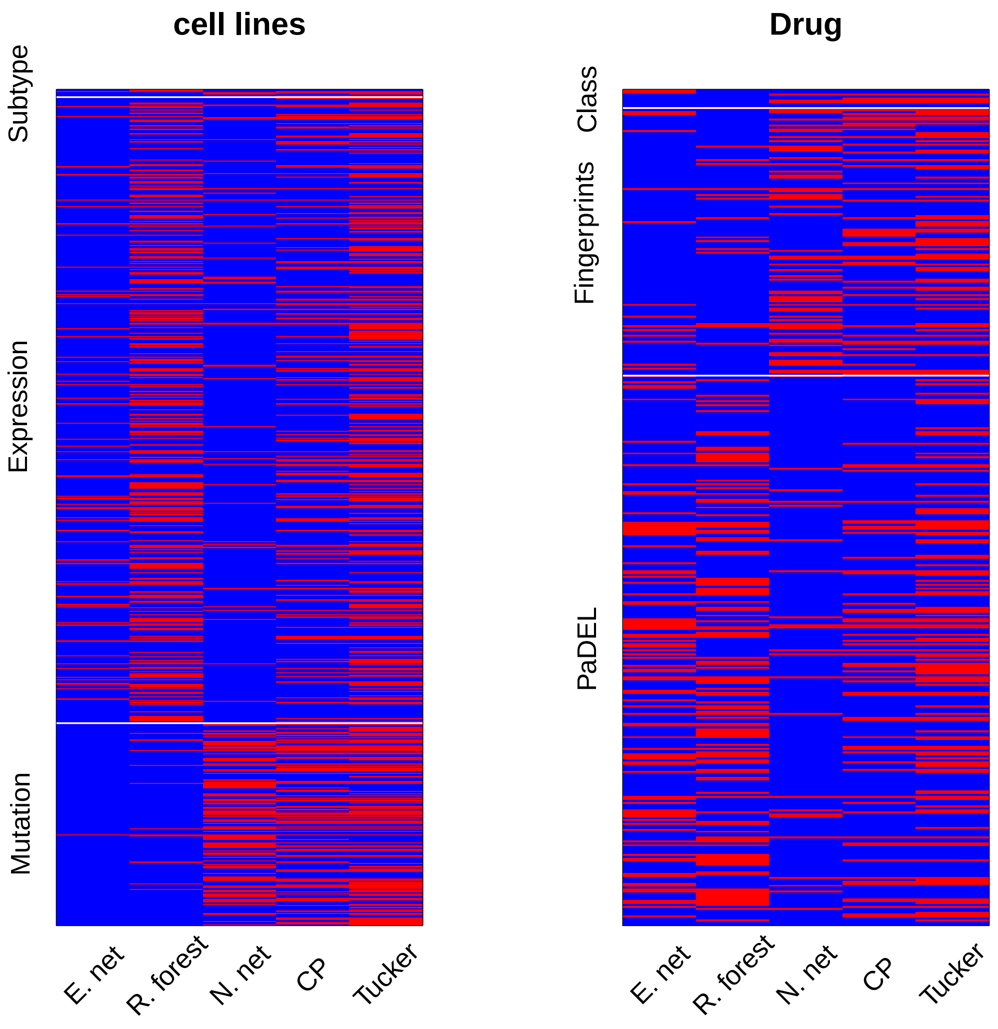


Figure 4.15: Features selected by different methods for the DREAM data. Each row is a feature, features selected in the top 15% of features for any replicate are colored red. Note that different methods tend to choose different features.

4.4 Heiser data

After the DREAM challenge was run, the group that generated the data continued refining their protocols and testing new drugs and cell lines [16] [15]. In this section we compare BaTFLED to other methods on an updated version of the breast cancer drug response data used in the previous section. This dataset contains 71 cell lines 107 drugs and is normalized using a method that accounts for the growth rate of the cell line [22].

4.4.1 Cell line features

All cell line data was downloaded from JWGray Breast Cancer Cell Line Panel project on the data sharing cite synapse.org [111]. Seven different data types were examined for use as features including subytypes, median doubling times across replicates, exon array expression, binarized expression, copy number, reverse phase protein lysate arrays (RPPA) and methylation data. Subtypes and the doubling time were annotated for 60 of the cell lines. Afffymetrix exon array expression data (continuous values for $18,632$ genes and binary values for $36,953$ genes) was available for 51 of the cell lines. DNA copy number assessed by the Affymetrix Genome-Wide Human SNP Array 6.0 was available for 63 cell lines and $27,234$ genes. We removed 752 genes that had 'NA' values. RPPA data generated by the Gordon Mills lab was available for 46 cell lines and 146 genes. Finally, methylation data (Illumina HumanMethylation27) was available for 47 cell lines, $27,578$ CpGs and $14,477$ genes. When we only consider cell lines that have responses measured for 80% of drugs, we're left with subtype annotations for 50 cell lines, continuous expression data for 48, binary expression data for 40, copy number data for 50, RPPA data for 44, and methylation data for 44 cell lines. Figure 4.16 shows which data were available for which cell lines.

To reduce the total size of this input data, we selected features for 609 genes identified as being related to cancer by the Catalogue of Somatic Mutations in Cancer (COSMIC) effort (Downloaded Jan 16 2017) [109] for further use. Additionally, for each data type, we averaged together features that had a Pearson correlation coefficient greater than 0.9 or less than -0.9. This left a total of 583 expression, 486 binary expression, 381 copy number, 33 RPPA and 1, 110 methylation features.

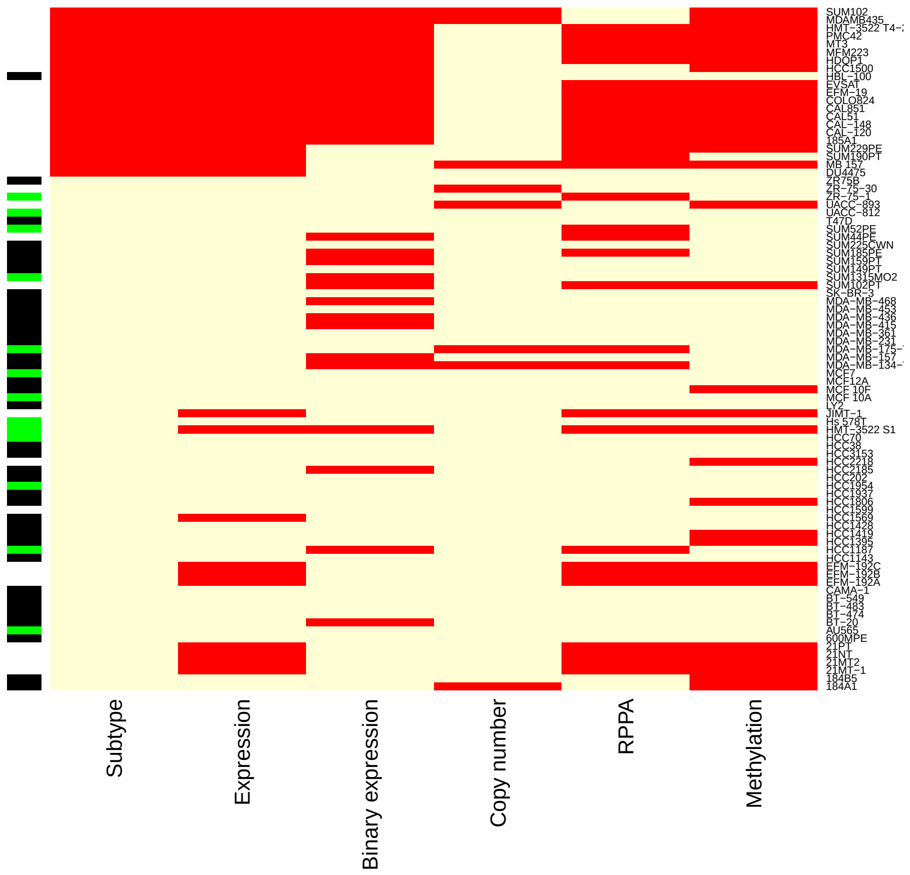


Figure 4.16: Data available for cell lines. Columns show different data types, red indicates missing data. Colors on the left indicate which cell lines were in the training and test sets (black and green respectively).

Since only 29 cell lines have all data types and the algorithms tested here cannot handle missing data in the input features, we tried two approaches to deal with missing data. First, we ran models using only the 48 cell lines for which subtype annotations, doubling time and expression data were available. In this case, the final test set contained 12 cell lines. Second, we converted the subtype, expression, copy number, RPPA and methylation features to similarity matrices between 51 cell lines using kernel functions. For binary data we used the Jaccard coefficient (equation 4.1) and for continuous data we used Gaussian kernels (equation 4.2). The width of the Gaussian kernels were set to .75

times the mean distance between vectors in the training set. This value was chosen based on visual inspection of heatmaps of kernel matrices. To combine these kernels, we took the weighted mean of the kernels across 5 datasets where the weights were the reciprocal of the standard deviation of the feature matrix for that data type. We kept the final training set of 13 cell lines separate when computing these kernels, so the kernel features consist of a similarity measure for each of the 38 training cell lines. A heatmap of the kernelized matrix is show in figure 4.17. While using kernels allows us to utilize all of the available data, we are not able to do meaningful feature selection.

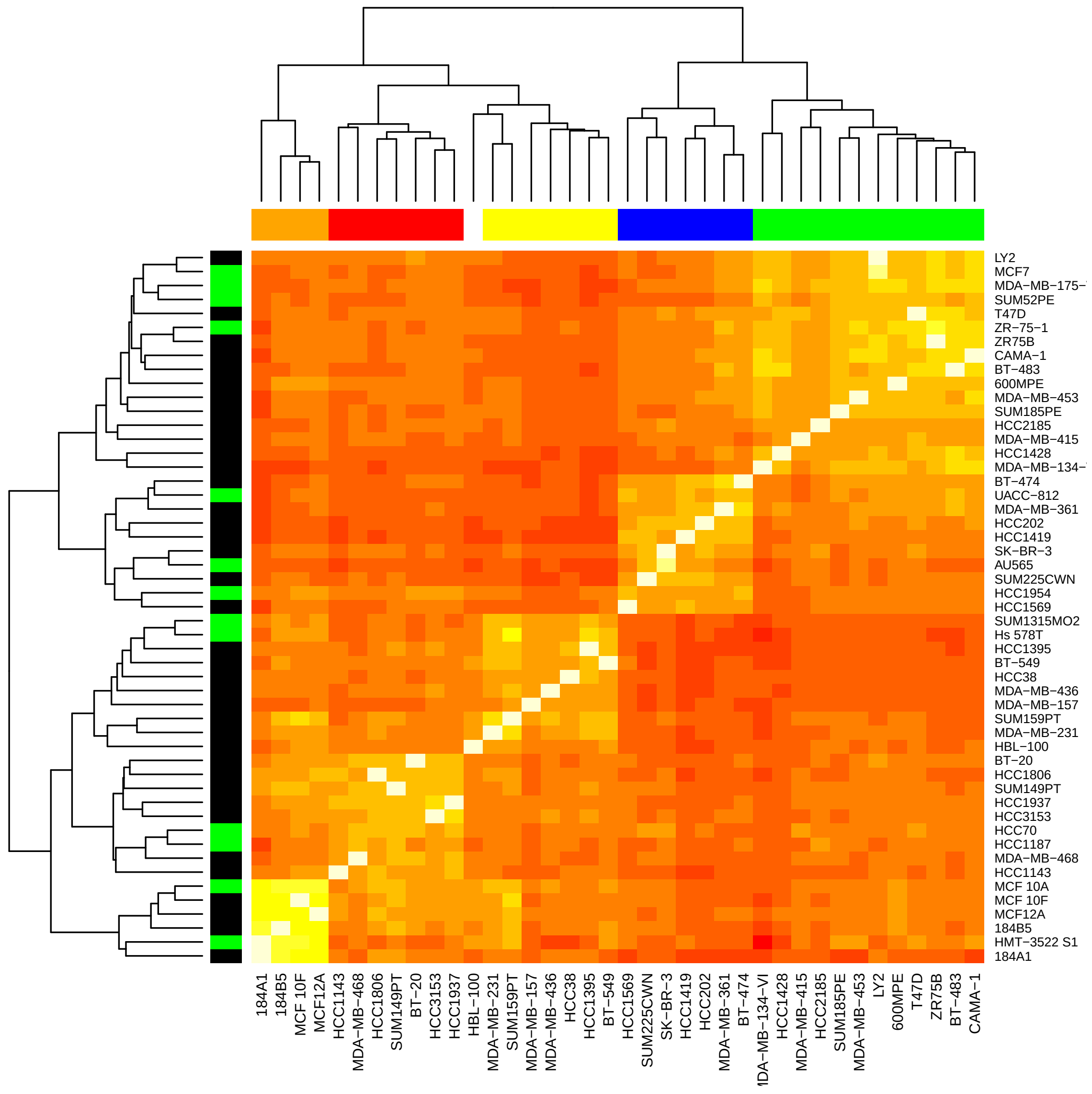


Figure 4.17: Heatmap of kernelized cell line features. Row colors indicate training and test set cell lines (black and green respectively). Column colors show subtypes for the training set cell lines that are used as features: red: basal, yellow: claudin low, blue: ERBB2 amplified, green: luminal, orange: non-malignant, white: not annotated. Dendrogram by hierarchical clustering based on Euclidean distance.

4.4.2 Drug features

Names and annotations for 107 drugs tested against the above cell lines were downloaded from the LINCS data portal (experiment 20269) [112]. Three of the drugs are antibodies and were discarded because structural information could not be obtained. The remaining drugs had 61 targets and 31 class annotations, but as many of these were unique to particular drugs or redundant, they would not be useful in a predictive setting. After removing targets and classes that were annotated for only one drug or were redundant, we were left with 4 targets and 23 classes (figure 4.18). Next we used the SMILES (simplified molecular-input line-entry system) strings and PaDEL software (v2.21) [110] to obtain 881 PubChem fingerprints and $1,875$ structure descriptors. Structural descriptors that were more than 75% 'NA' values, had no variance across drugs or only had one non-zero value were removed leaving $1,115$ features. Also, 116 features with a range larger than $1,000$ were removed and highly correlated features were combined (Pearson correlation greater than 0.9 or less than -0.9) leaving 428 structural features. The resulting feature matrix had 470 (1.08%) values that were 'NA', these were set to the mean across drugs. Next, fingerprint features present in only one drug were removed, and correlated features were combined leaving 271 fingerprint features. After this data cleaning, we were left with a set of 726 drug features.

As with the cell line features, we explored both using these features directly and using kernelized versions. We computed Jaccard kernels for the target/class data, and Gaussian kernels for the fingerprint and PaDEL data. Because we combined correlated fingerprint features, these were not all binary. Inspecting heatmaps of kernel matrices produced using different widths for the fingerprint and PaDEL kernels we chose a width of 0.75 times the average distance between vectors for both of these datasets. As with the cell line data, we combined these kernels using a weighted average where the weights were equal to the reciprocal of the standard deviations (figure 4.19).

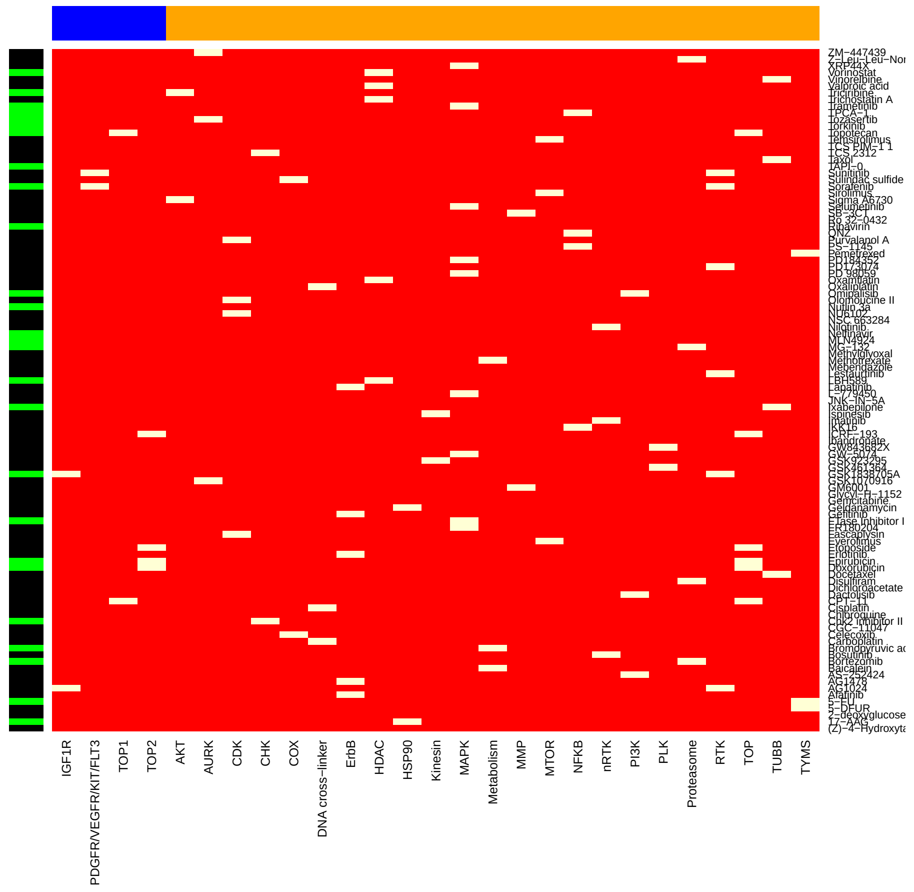


Figure 4.18: Target and class indicators for drugs. White rectangles represent positive annotation for the target or class. Column colors indicate target (blue) or class (orange) indicators. Row colors indicate training and test sets (black and green respectively).

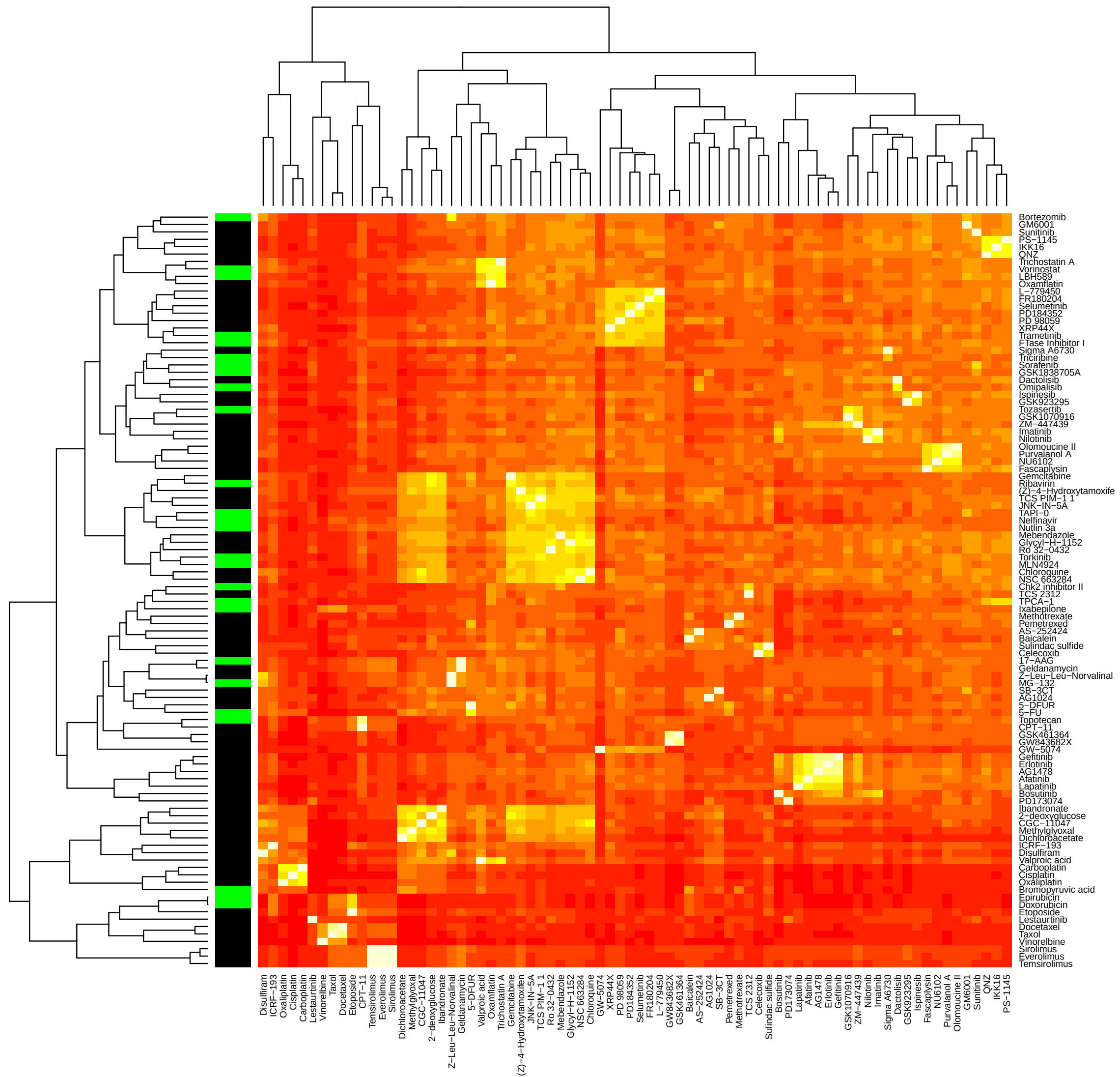


Figure 4.19: Heatmap of kernelized features for drug data. Rows colors indicate training and test set drugs (black and green respectively). Columns show the training set drug similarities that are used as features. Dendrogram by heirarchal clustering based on Euclidean distance.

4.4.3 Response data

As mentioned in section 1.1.2 studies of cell line drug response may be confounded by the growth rate of the cell line. To account for this, the data for this section was characterized using the growth rate 50 (GR_{50}) response metric instead of the common growth inhibition (GI_{50}) measure. These values, as well as the raw growth response measurements at 9 doses, were downloaded from the LINCS data portal (experiments 20268 and 20269). GR_{50} and raw data was available for 71 cell lines and 110 drugs. Of the total $7,810$ combinations, $4,864$ had these response values (62.3%). Three of the drugs are antibodies and were removed. Of the remaining $7,597$ combinations $2,618$ (34.5%) had multiple sets of raw responses and $2,809$ (37.0%) had no raw responses. Since our methods assumes that the same doses are used for each drug, we chose the 9 most common doses for each drug, checked that these were evenly spaced and removed results with different doses. Five drugs had small annotation issues with the drug doses that were corrected manually. One drug (FR180204) was tested at 18 doses, we chose the upper 9 as the active region after examining response curves. Six drugs had less than 9 doses and so have missing values in the response tensor. After these corrections $2,279$ combinations (30.0%) have multiple sets of raw responses and $3,014$ (39.7%) have no raw responses.

Next we selected one set of raw data for each cell line/drug combination by 1) choosing the replicate with the GR_{50} closest to the median across replicates, 2) breaking ties by using the replicate with the larger GRAOC, 3) breaking further ties (4 combinations) by using the larger GR_{50} and 5) breaking a final tie by taking the replicate with more total cells before treatment. These raw data were then normalized for growth rate using the GR_{50} formula given in equation 1.1. Finally cell lines that were missing responses for more than 80% of drugs and drugs that were missing responses for more than 80% of cell lines were removed leaving 51 cell lines and 102 drugs in the final set.

4.4.4 Experiments

For this dataset, instead of performing training for all prediction tasks simultaneously like was done in section 4.3, we remove final training sets for each task separately. For the warm-start task, we randomly remove responses for 20% of cell line/drug combinations that have data (800 of $3,978$ and 818 of $4,128$ for the kernelized data). For cold start tasks, we split roughly 25% of the cell lines and drugs into a final testing set (12/48 cell lines with the expression data runs, 13/51 cell lines when using kernelized features and 26/102 drugs). We report results both on these final test sets and on five-fold cross validation runs on just the training data. When running models to predict for cell lines, we use all 102 drugs and perform 5-fold cross-validation on the 36 (38 for kernelized runs) training cell lines. When predicting for drugs, we use all 48 cell lines (51 for kernelized runs), and perform 5-fold cross-validation on the 76 training drugs. When predicting for cell line/drug combinations, we only use the training set cell lines and training set drugs and perform cross-validation on both cell lines and drugs.

As with the DREAM data and the simulated data, we compared performance on each task to predicting mean responses, four elastic net models, three random forest models and three neural net models. We used the same settings for these as with previous tests. BaTFLED CP models were trained with a latent dimension of 200 and Tucker models given $10 \times 10 \times 9$ cores. Due to the normalization of responses, we didn't expect there to be effects that would apply regardless of dose, so we only added columns of 1s to the cell line and drug latent matrices. Also, we found that training for 50 iterations was sufficient for these models to converge. For the cross-validation results, we tested a two-round training scheme for BaTFLED models, which consisted of training initial models strongly encouraging sparsity in the features ($\alpha = 10^{-10}$ and $\beta = 10^{10}$), selecting the union of the top 15% of features across folds for the second round and retraining without encouraging sparsity ($\alpha = \beta = 1$). In Tucker models, sparsity was not encouraged in the core. On the final test set, we characterized the performance of BaTFLED models using all features and encouraging sparsity and separately, using just those features selected in two-round cross-validation testing and not encouraging sparsity. Note that although this two-round strategy does represent some data leakage for the cross-validation results, the results on the final test set do not have this contamination.

4.4.5 Performance

In CV runs, the retrained BaTFLED Tucker model performs the best on all cold-start tasks in terms of average NRMSE across folds (Tables 4.5, 4.6 and Figure 4.20). While mean prediction and elastic net models perform comparably on most tasks, random forest and neural net models perform poorly on the cold-start cell line task. These two methods show differences on the cold-start drug task and combination cell line/drug task where Random forest performs better than neural nets. When considering the results from the runs that use kernelized features, mean prediction and LASSO models generally perform the best while random forest and neural net models are particularly bad at cold-start cell line prediction. Interestingly, none of the models using kernelized features outperforms the models that use the features directly except for elastic net regression for cell line/drug combination prediction. The best NRMSE performance for this task without kernels is 0.756 for the retrained Tucker model while the kernelized elastic net model has a slightly better NRMSE of 0.751 (Tables 4.7, 4.8 and Figure 4.21).

While many of these methods appear comparable on average, examining performance on individual cell lines, drugs and cell line/drug combinations suggests that they methods have different strengths. Figures 4.22 and 4.23 show box plots for predicting individual examples and figures 4.24, 4.25, and 4.26 show head-to-head comparisons of Pearson correlation performance for the non-kernelized models. Each panel compares one of the models with the baseline of mean prediction.

Table 4.5: Mean normalized RMSE on the Heiser dataset over five cross-validation folds (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.46(0.00)	0.47(0.01)	0.49(0.01)	0.77(0.04)	0.78(0.03)
LASSO a=1	0.46(0.00)	0.48(0.01)	0.49(0.01)	0.82(0.06)	0.84(0.07)
E. net a=0.9	0.46(0.00)	0.47(0.01)	0.49(0.01)	0.81(0.06)	0.84(0.06)
E. net a=0.5	0.46(0.00)	0.48(0.01)	0.49(0.01)	0.80(0.05)	0.82(0.06)
Ridge a=0	0.45(0.00)	0.48(0.01)	0.49(0.01)	0.81(0.05)	0.79(0.05)
N.N. 1L	0.32(0.01)	0.54(0.02)	0.80(0.01)	0.93(0.07)	0.92(0.08)
N.N. 2L	**0.18**(0.00)	0.48(0.01)	0.80(0.01)	0.83(0.04)	0.80(0.05)
N.N. 3L	0.21(0.01)	0.48(0.01)	0.98(0.05)	0.93(0.07)	0.93(0.05)
R.F. 1000x5	0.46(0.00)	0.49(0.01)	0.86(0.01)	0.79(0.04)	0.81(0.04)
R.F. 5000x5	0.46(0.00)	0.50(0.01)	0.85(0.03)	0.80(0.05)	0.81(0.03)
R.F. 1000x10	0.37(0.00)	**0.46**(0.01)	0.89(0.03)	0.80(0.04)	0.80(0.03)
CP	0.35(0.01)	0.67(0.04)	0.51(0.01)	0.94(0.15)	0.91(0.14)
Tucker	0.43(0.01)	0.47(0.01)	0.49(0.01)	0.88(0.10)	0.89(0.11)
CP retrain	0.36(0.00)	0.59(0.01)	0.51(0.02)	0.94(0.09)	0.95(0.08)
Tucker retrain	0.43(0.01)	0.47(0.01)	**0.48**(0.01)	**0.72**(0.03)	**0.76**(0.06)

*Training results on warm-start task.

Table 4.6: Mean Pearson correlation on the Heiser dataset over five cross-validation folds (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.89(0.00)	0.88(0.01)	**0.88**(0.00)	0.64(0.04)	0.64(0.02)
LASSO a=1	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.60(0.06)	0.59(0.06)
E. net a=0.9	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.61(0.05)	0.59(0.06)
E. net a=0.5	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.62(0.05)	0.61(0.06)
Ridge a=0	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.62(0.05)	0.64(0.03)
N.N. 1L	0.96(0.00)	0.86(0.01)	0.62(0.01)	0.52(0.08)	0.53(0.08)
N.N. 2L	**0.99**(0.00)	0.88(0.01)	0.62(0.02)	0.59(0.04)	0.62(0.05)
N.N. 3L	0.98(0.00)	0.87(0.00)	0.48(0.06)	0.51(0.08)	0.50(0.07)
R.F. 1000x5	0.89(0.00)	0.87(0.01)	0.61(0.02)	0.63(0.04)	0.61(0.03)
R.F. 5000x5	0.89(0.00)	0.87(0.01)	0.61(0.02)	0.61(0.06)	0.61(0.02)
R.F. 1000x10	0.93(0.00)	**0.89**(0.01)	0.61(0.02)	0.62(0.05)	0.62(0.03)
CP	0.94(0.00)	0.77(0.02)	0.87(0.00)	0.51(0.11)	0.52(0.11)
Tucker	0.90(0.01)	0.88(0.01)	0.87(0.01)	0.55(0.10)	0.54(0.11)
CP retrain	0.93(0.00)	0.81(0.01)	0.86(0.01)	0.53(0.10)	0.53(0.06)
Tucker retrain	0.90(0.01)	**0.89**(0.01)	**0.88**(0.01)	**0.70**(0.03)	**0.68**(0.07)

*Training results on warm-start task.

Table 4.7: Mean normalized RMSE on the Heiser dataset over five cross-validation folds when using kernelized features (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.46(0.00)	0.48(0.01)	**0.49**(0.02)	0.77(0.04)	0.78(0.03)
LASSO a=1	0.46(0.00)	0.48(0.01)	**0.49**(0.02)	0.79(0.08)	0.80(0.08)
E. net a=0.9	0.46(0.01)	0.48(0.02)	**0.49**(0.02)	0.79(0.08)	0.80(0.08)
E. net a=0.5	0.46(0.00)	**0.47**(0.01)	**0.49**(0.02)	0.79(0.09)	0.79(0.09)
Ridge a=0	0.60(0.00)	0.62(0.01)	0.62(0.01)	**0.74**(0.05)	**0.75**(0.05)
N.N. 1L	**0.27**(0.00)	0.57(0.01)	5.07(0.48)	4.87(0.80)	0.81(0.05)
N.N. 2L	0.28(0.01)	0.51(0.01)	1.00(0.12)	0.78(0.05)	0.85(0.07)
N.N. 3L	0.29(0.01)	0.49(0.01)	0.93(0.06)	0.85(0.04)	0.88(0.09)
R.F. 1000x5	0.46(0.00)	0.49(0.01)	0.88(0.06)	0.81(0.07)	0.82(0.06)
R.F. 5000x5	0.50(0.03)	0.54(0.03)	0.88(0.05)	0.81(0.07)	0.82(0.07)
R.F. 1000x10	0.43(0.00)	**0.47**(0.01)	0.90(0.05)	0.83(0.08)	0.83(0.07)
CP	0.57(0.00)	0.59(0.01)	0.60(0.01)	1.01(0.09)	1.00(0.11)
Tucker	0.50(0.01)	0.52(0.01)	0.53(0.02)	1.13(0.15)	1.13(0.15)

*Training results on warm-start task.

Table 4.8: Mean Pearson correlation on the Heiser dataset over five cross-validation folds when using kernelized features (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.89(0.00)	**0.88**(0.01)	**0.88**(0.01)	0.64(0.04)	0.64(0.03)
LASSO a=1	0.89(0.00)	**0.88**(0.00)	0.87(0.01)	0.65(0.05)	0.64(0.05)
E. net a=0.9	0.89(0.00)	**0.88**(0.01)	0.87(0.01)	0.65(0.05)	0.64(0.05)
E. net a=0.5	0.89(0.00)	**0.88**(0.00)	0.87(0.01)	0.65(0.05)	0.65(0.05)
Ridge a=0	0.80(0.00)	0.79(0.01)	0.79(0.01)	**0.69**(0.04)	**0.68**(0.04)
N.N. 1L	**0.97**(0.00)	0.85(0.01)	-0.27(0.10)	-0.25(0.16)	0.62(0.04)
N.N. 2L	0.96(0.00)	0.86(0.01)	0.62(0.01)	0.64(0.05)	0.59(0.06)
N.N. 3L	0.96(0.00)	0.87(0.00)	0.62(0.01)	0.57(0.04)	0.55(0.09)
R.F. 1000x5	0.89(0.00)	**0.88**(0.00)	0.63(0.01)	0.60(0.06)	0.60(0.06)
R.F. 5000x5	0.87(0.02)	0.84(0.02)	0.63(0.01)	0.60(0.06)	0.60(0.06)
R.F. 1000x10	0.90(0.00)	**0.88**(0.00)	0.63(0.01)	0.59(0.07)	0.60(0.06)
CP	0.82(0.00)	0.81(0.01)	0.81(0.01)	0.52(0.08)	0.51(0.09)
Tucker	0.87(0.00)	0.86(0.01)	0.85(0.01)	0.46(0.10)	0.45(0.12)

*Training results on warm-start task.

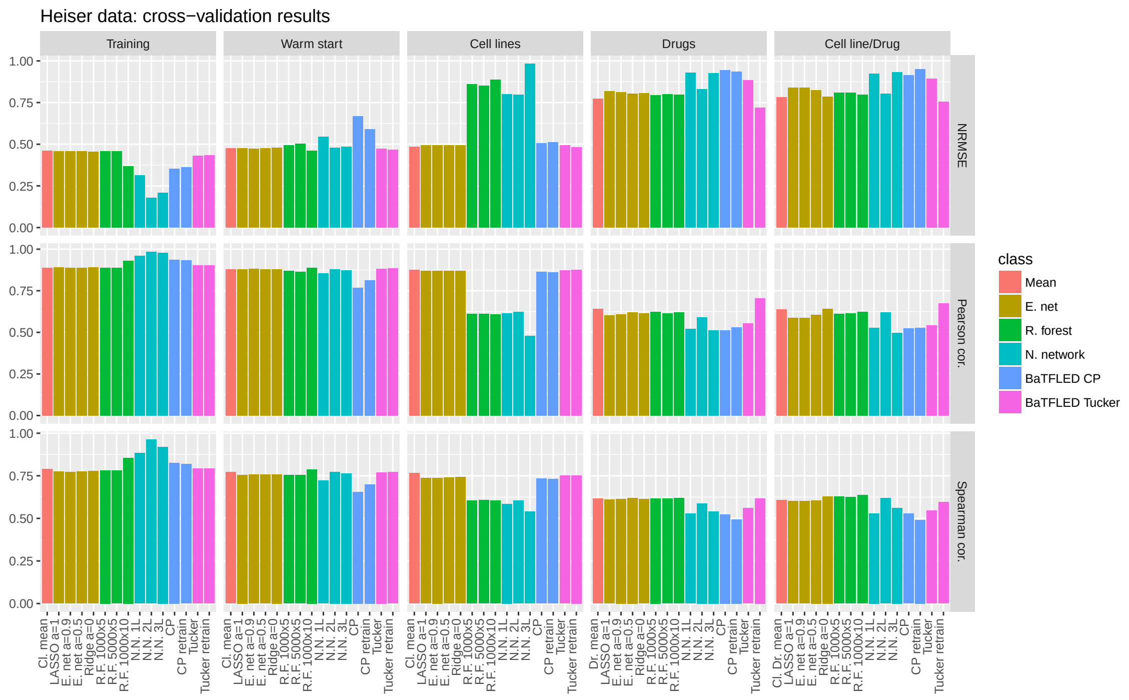


Figure 4.20: Performance measures for cross-validation runs on Heiser data (average across 5 cv runs). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown.

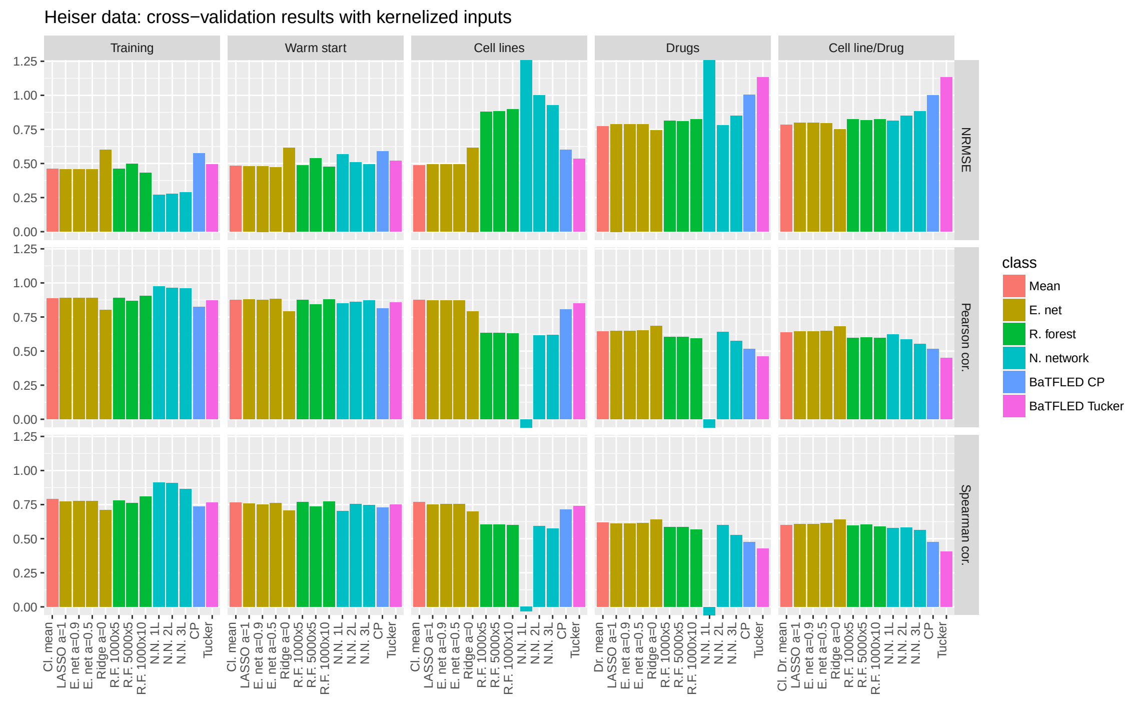


Figure 4.21: Performance measures for cross-validation runs on Heiser data when using 'kernelized' features (average across 5 cv runs). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown.

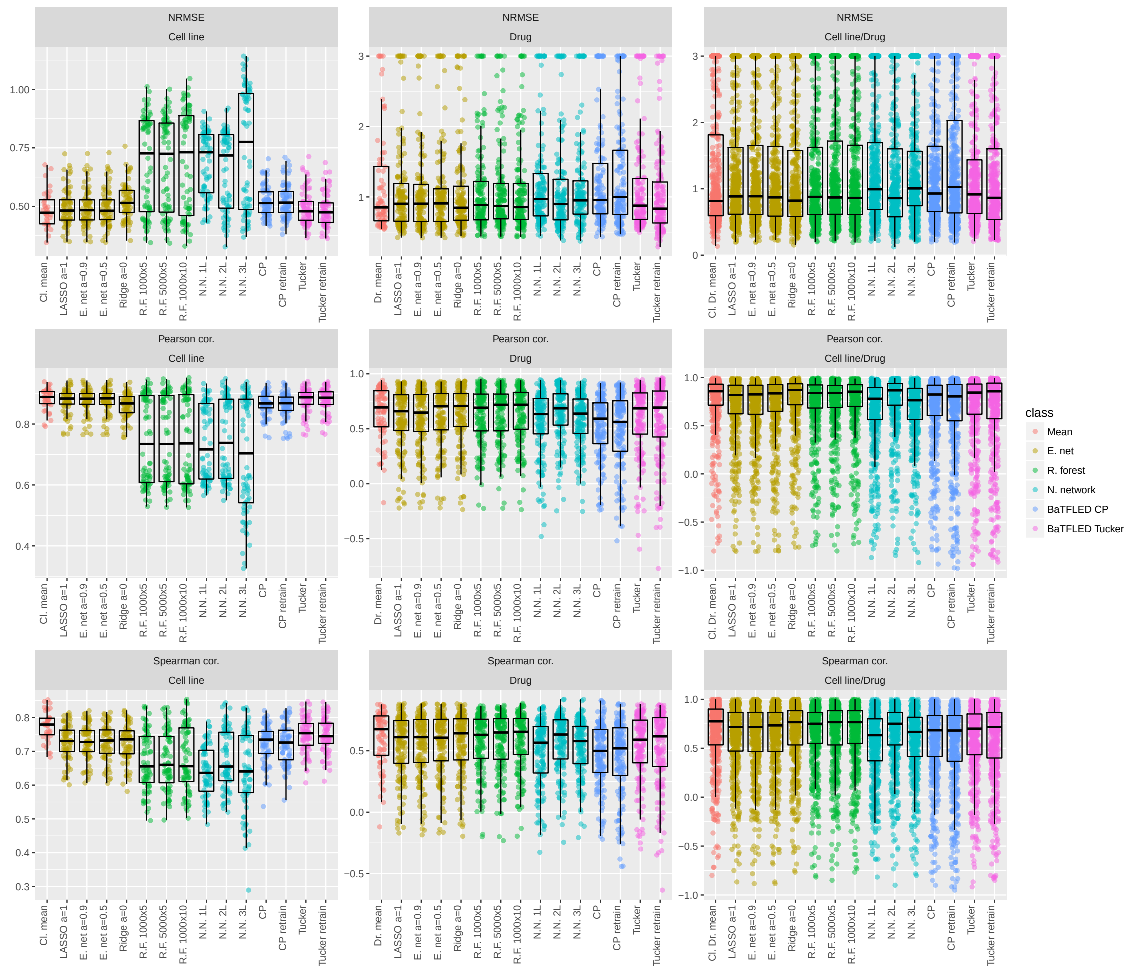


Figure 4.22: Performance measures for cross-validation runs on Heiser data. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

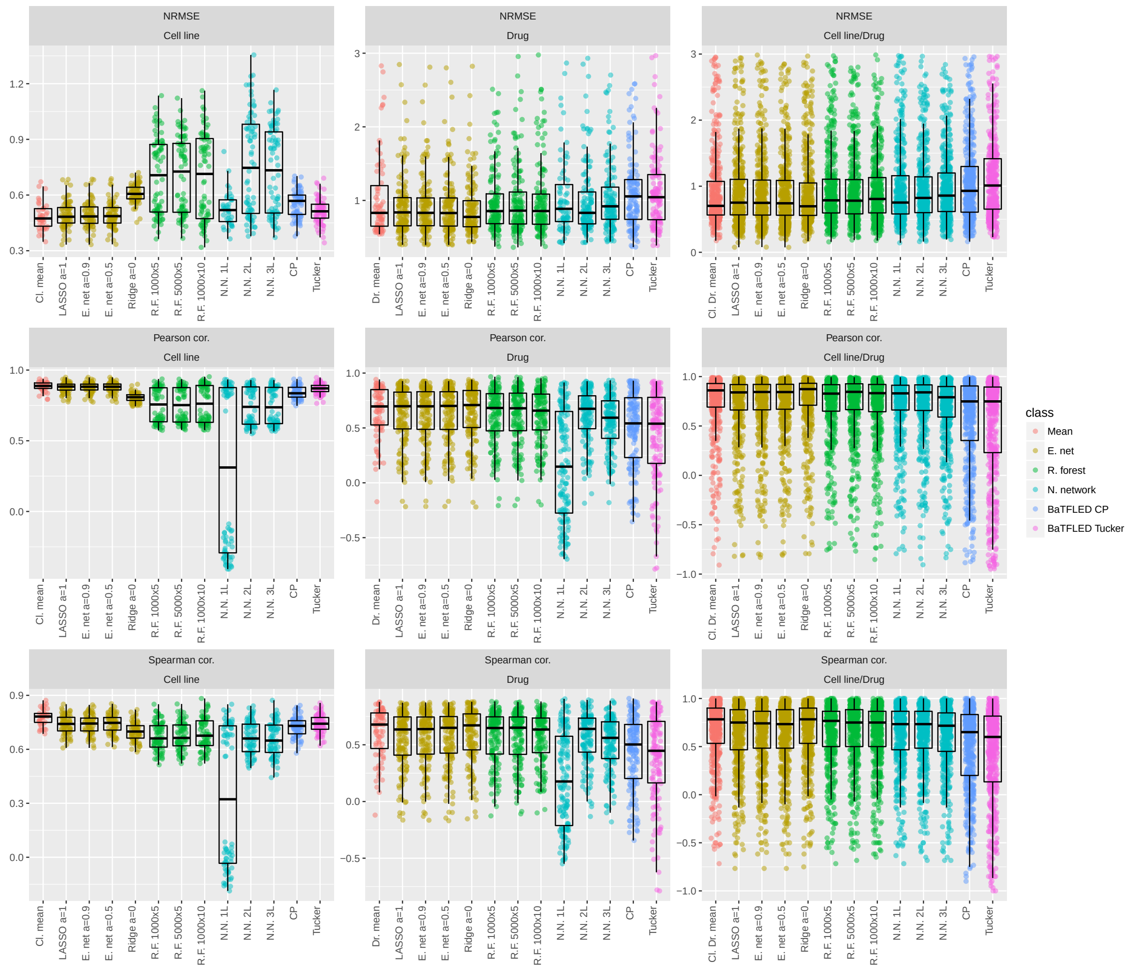


Figure 4.23: Performance measures for cv runs on Heiser data with kernelized features. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

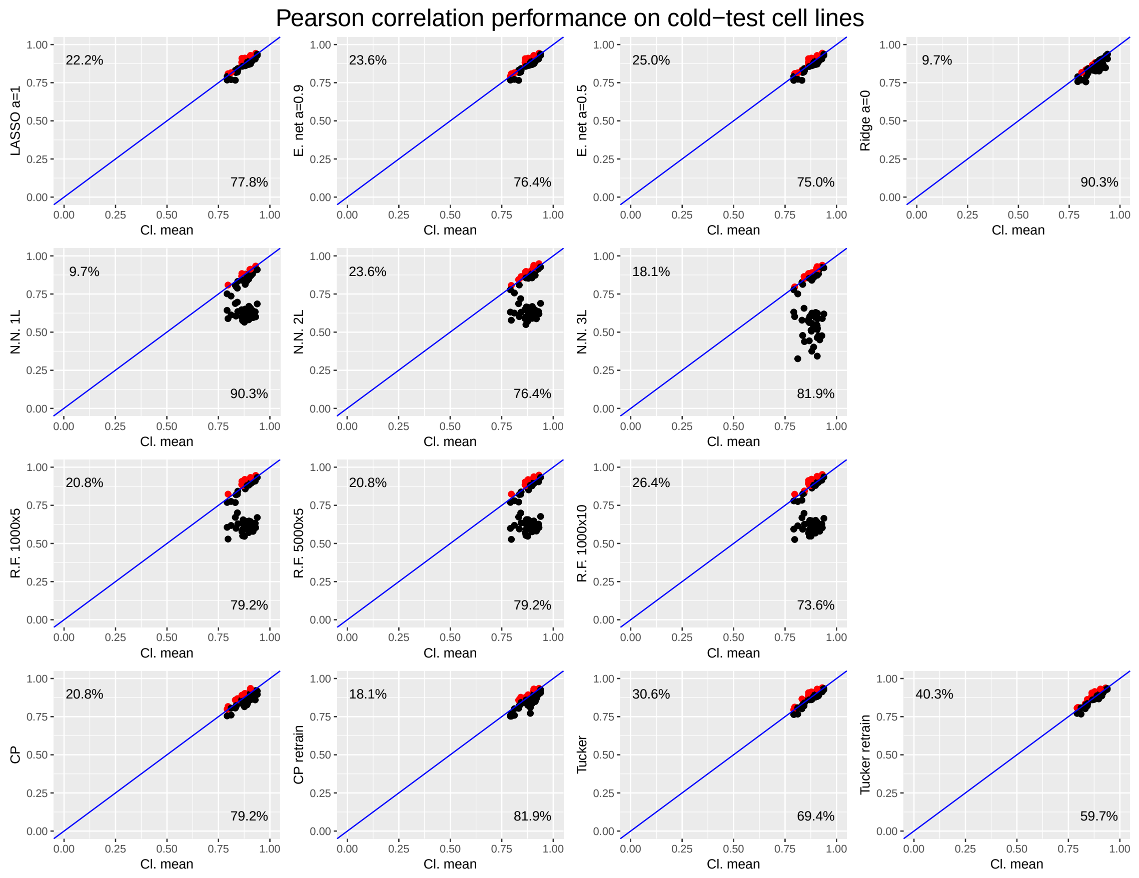


Figure 4.24: Comparison between mean prediction and other methods on the cold-start cell line task from cross-validation runs. Each point represents one cell line with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell lines on either side of the diagonal are annotated.

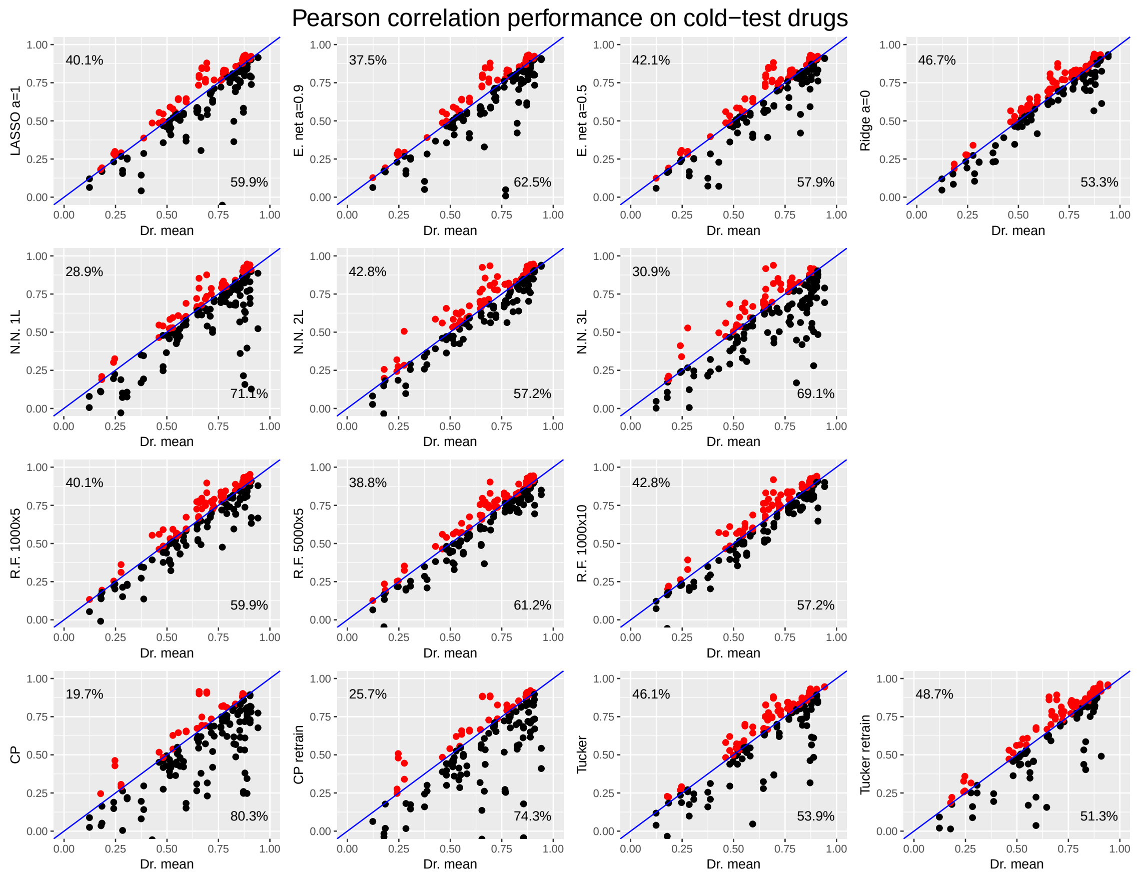


Figure 4.25: Comparison between mean prediction and other methods on the cold-start drug task from cross-validation runs. Each point represents one drug with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of drugs on either side of the diagonal are annotated.

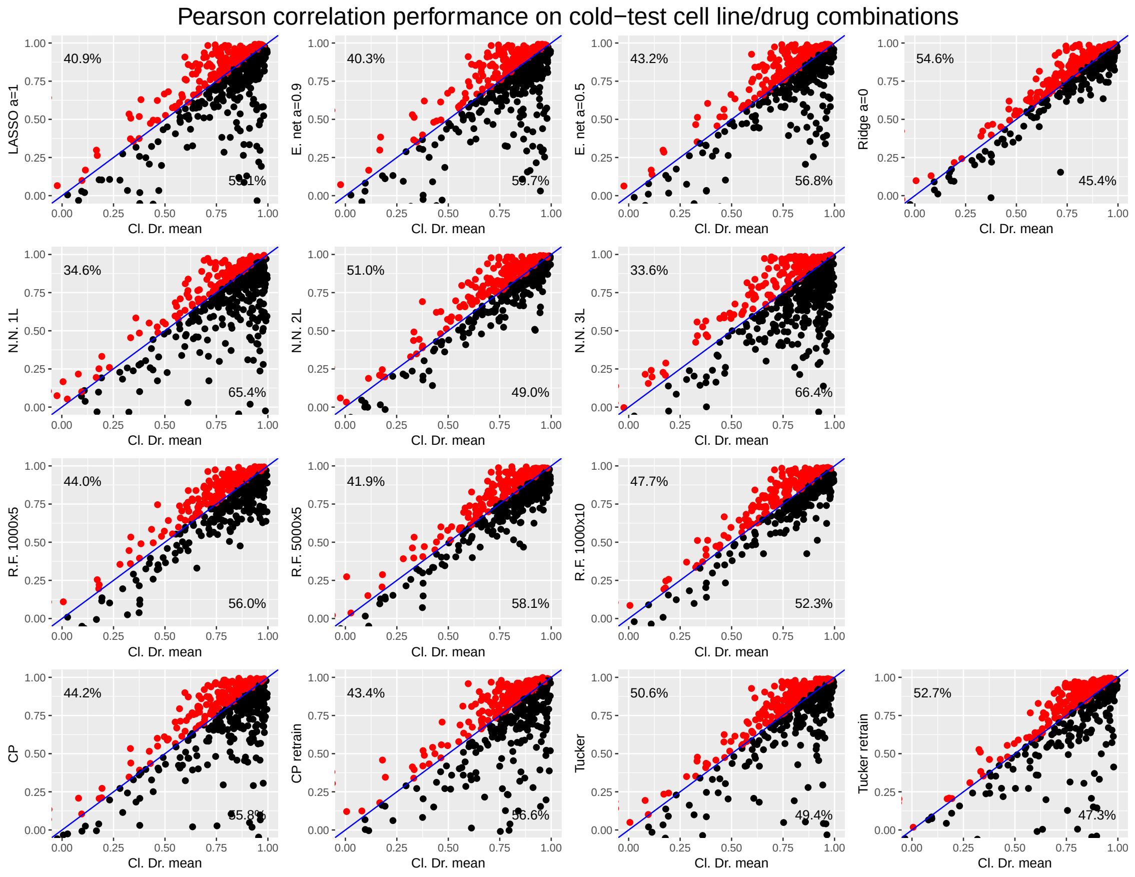


Figure 4.26: Comparison between mean prediction and other methods on the cold-start cell line/drug combination task from cross-validation runs. Each point represents one cell line/drug combination with its x coordinate showing the Pearson correlation of mean prediction (across cell lines) with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell line/drug combinations on either side of the diagonal are annotated.

For the final test data, the BaTFLED Tucker models are among the best performing algorithms at warm-start prediction and cold-start cell line prediction (Tables 4.9, 4.10 and Figure 4.27). However, these models perform similarly to simple mean prediction which highlights both the difficulty of the problem and the importance of feature engineering and cleaning for all machine learning algorithms. Runs that use features selected by the cross-validation runs do not perform significantly better on any tasks and have a worse performance at the cold-start drug and cell line/drug combination task. This indicates either that the predictive power of the drug features is diffusely distributed among a large number of features or that the feature selection performed by the BaTFLED models does not easily identify strong predictors. Finally, Tucker models outperform the CP models in almost all cases, which may be due to over-fitting since the NRMSE on the training data is smaller for CP models than for Tucker models.

Interestingly, by using kernelized features, several methods are able to outperform mean prediction on the cold-start drug and cell line/drug combination task. BaTFLED CP is top method in terms of Pearson correlation for both of these tasks (Tables 4.11, 4.12 and Figure 4.28. This highlights the potential of Tensor methods when combined with non-linear features and indicates that CP models may have advantages in certain situations, particularly when feature selection is not one of the goals. Additionally, the observation that feature selection seems to hurt cold-start drug prediction, and that several methods can outperform mean prediction on this task indicates that non-linear interactions may be more important in the drug features. We chose kernel widths based on visual inspection of kernel matrices and attempted to balance the kernels from different data types equally which is probably not optimal. A thorough exploration of the kernel widths and combination weighting schemes would likely produce better results. Our preliminary tests indicate that the kernel parameters, the algorithm used for prediction, and the parameters of that algorithm can all be inter-related in unpredictable ways suggesting that a larger Bayesian model incorporating all of these factors may yield prediction improvements.

As with the cross-validation results, different methods have advantages for specific cell lines, drugs and cell line/drug combinations. Plots of head-to-head comparisons of these methods with mean prediction on the final test data are shown in figures 4.31, 4.32 and

4.33.

Table 4.9: Mean normalized RMSE on the final test data for the Heiser dataset (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.46(0.00)	0.48(0.00)	**0.48**(0.00)	0.80(0.00)	0.82(0.00)
LASSO a=1	0.46(0.00)	0.48(0.00)	0.49(0.00)	1.10(0.00)	1.08(0.00)
E. net a=0.9	0.46(0.00)	0.48(0.00)	0.49(0.00)	1.10(0.00)	1.07(0.01)
E. net a=0.5	0.46(0.00)	0.48(0.00)	0.49(0.00)	1.09(0.00)	1.07(0.00)
Ridge a=0	0.50(0.00)	0.52(0.00)	0.53(0.00)	0.92(0.00)	0.91(0.00)
N.N. 1L	0.34(0.01)	0.53(0.00)	2.20(0.54)	1.43(0.28)	0.85(0.01)
N.N. 2L	**0.20**(0.01)	0.48(0.00)	0.84(0.05)	0.91(0.10)	0.82(0.00)
N.N. 3L	0.22(0.00)	0.48(0.00)	1.01(0.03)	0.82(0.02)	0.83(0.01)
R.F. 1000x5	0.46(0.00)	0.50(0.00)	0.80(0.01)	0.78(0.01)	**0.80**(0.00)
R.F. 5000x5	0.46(0.00)	0.50(0.00)	0.80(0.01)	0.78(0.01)	**0.80**(0.01)
R.F. 1000x10	0.37(0.00)	**0.46**(0.00)	0.80(0.00)	**0.77**(0.00)	**0.80**(0.00)
CP	0.37(0.00)	0.57(0.01)	**0.48**(0.00)	0.99(0.02)	0.95(0.01)
Tucker	0.44(0.01)	0.47(0.00)	**0.48**(0.00)	0.95(0.01)	0.92(0.01)
CP selected	0.38(0.00)	0.54(0.00)	0.49(0.00)	3.87(0.06)	3.22(0.22)
Tucker selected	0.43(0.01)	0.47(0.01)	**0.48**(0.00)	1.01(0.02)	1.25(0.04)

*Training results on warm-start task.

Table 4.10: Mean Pearson correlation on the final test data for the Heiser dataset (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.89(0.00)	0.88(0.00)	**0.88**(0.00)	0.65(0.00)	0.63(0.00)
LASSO a=1	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.42(0.00)	0.43(0.00)
E. net a=0.9	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.42(0.00)	0.43(0.00)
E. net a=0.5	0.89(0.00)	0.88(0.00)	0.87(0.00)	0.42(0.00)	0.43(0.00)
Ridge a=0	0.87(0.00)	0.85(0.00)	0.85(0.00)	0.50(0.00)	0.51(0.00)
N.N. 1L	0.95(0.00)	0.86(0.00)	0.53(0.08)	0.33(0.11)	0.59(0.02)
N.N. 2L	**0.98**(0.00)	0.88(0.00)	0.62(0.01)	0.50(0.12)	**0.64**(0.00)
N.N. 3L	**0.98**(0.00)	0.88(0.00)	0.61(0.01)	0.62(0.02)	0.62(0.01)
R.F. 1000x5	0.89(0.00)	0.87(0.00)	0.62(0.00)	0.65(0.01)	**0.64**(0.01)
R.F. 5000x5	0.89(0.00)	0.87(0.00)	0.62(0.01)	0.64(0.01)	0.63(0.01)
R.F. 1000x10	0.93(0.00)	**0.89**(0.00)	0.63(0.00)	**0.66**(0.00)	**0.64**(0.00)
CP	0.93(0.00)	0.82(0.00)	**0.88**(0.00)	0.54(0.01)	0.53(0.01)
Tucker	0.90(0.00)	0.88(0.00)	**0.88**(0.00)	0.56(0.00)	0.55(0.01)
CP selected	0.93(0.00)	0.85(0.00)	0.87(0.00)	0.34(0.00)	0.31(0.01)
Tucker selected	0.90(0.01)	**0.89**(0.00)	**0.88**(0.00)	0.47(0.01)	0.26(0.02)

*Training results on warm-start task.

Table 4.11: Mean normalized RMSE on the final test data for the Heiser dataset when using kernelized features (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.46(0.00)	**0.48**(0.00)	**0.48**(0.00)	0.80(0.00)	0.82(0.00)
LASSO a=1	0.46(0.00)	**0.48**(0.00)	0.49(0.00)	0.76(0.00)	0.77(0.00)
E. net a=0.9	0.46(0.00)	**0.48**(0.00)	0.49(0.00)	0.76(0.00)	0.77(0.00)
E. net a=0.5	0.46(0.00)	**0.48**(0.00)	0.49(0.00)	**0.75**(0.00)	**0.76**(0.00)
Ridge a=0	0.60(0.00)	0.62(0.00)	0.62(0.00)	**0.75**(0.00)	0.77(0.00)
N.N. 1L	**0.29**(0.01)	0.55(0.01)	4.04(0.36)	4.22(0.51)	**0.76**(0.00)
N.N. 2L	0.30(0.00)	0.50(0.00)	1.00(0.04)	0.79(0.02)	0.79(0.00)
N.N. 3L	0.32(0.01)	0.49(0.00)	0.90(0.02)	0.90(0.02)	0.81(0.00)
R.F. 1000x5	0.53(0.00)	0.56(0.00)	0.85(0.01)	0.76(0.00)	0.78(0.01)
R.F. 5000x5	0.53(0.00)	0.56(0.00)	0.86(0.01)	0.76(0.01)	0.78(0.00)
R.F. 1000x10	0.43(0.00)	**0.48**(0.00)	0.87(0.01)	0.76(0.01)	0.79(0.00)
CP	0.62(0.00)	0.64(0.00)	0.64(0.00)	**0.75**(0.01)	0.77(0.00)
Tucker	0.69(0.01)	0.69(0.01)	0.69(0.01)	0.77(0.00)	0.79(0.00)

*Training results on warm-start task.

Table 4.12: Mean Pearson correlation on the final test data for the Heiser dataset when using kernelized features (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.89(0.00)	**0.88**(0.00)	**0.88**(0.00)	0.65(0.00)	0.64(0.00)
LASSO a=1	0.89(0.00)	**0.88**(0.00)	0.87(0.00)	0.68(0.00)	0.68(0.00)
E. net a=0.9	0.89(0.00)	**0.88**(0.00)	0.87(0.00)	0.68(0.00)	0.68(0.00)
E. net a=0.5	0.89(0.00)	**0.88**(0.00)	0.87(0.00)	0.68(0.00)	0.68(0.00)
Ridge a=0	0.80(0.00)	0.79(0.00)	0.79(0.00)	0.70(0.00)	0.69(0.00)
N.N. 1L	**0.97**(0.00)	0.85(0.00)	-0.27(0.06)	-0.19(0.04)	0.68(0.00)
N.N. 2L	0.96(0.00)	0.87(0.00)	0.62(0.01)	0.66(0.01)	0.65(0.00)
N.N. 3L	0.95(0.00)	0.87(0.00)	0.61(0.01)	0.52(0.03)	0.62(0.00)
R.F. 1000x5	0.85(0.00)	0.83(0.00)	0.64(0.00)	0.68(0.00)	0.66(0.01)
R.F. 5000x5	0.85(0.00)	0.83(0.00)	0.64(0.00)	0.68(0.01)	0.66(0.00)
R.F. 1000x10	0.90(0.00)	**0.88**(0.00)	0.64(0.00)	0.67(0.01)	0.66(0.01)
CP	0.79(0.00)	0.77(0.00)	0.78(0.00)	**0.71**(0.00)	**0.70**(0.00)
Tucker	0.73(0.01)	0.73(0.01)	0.73(0.01)	0.69(0.00)	0.68(0.00)

*Training results on warm-start task.

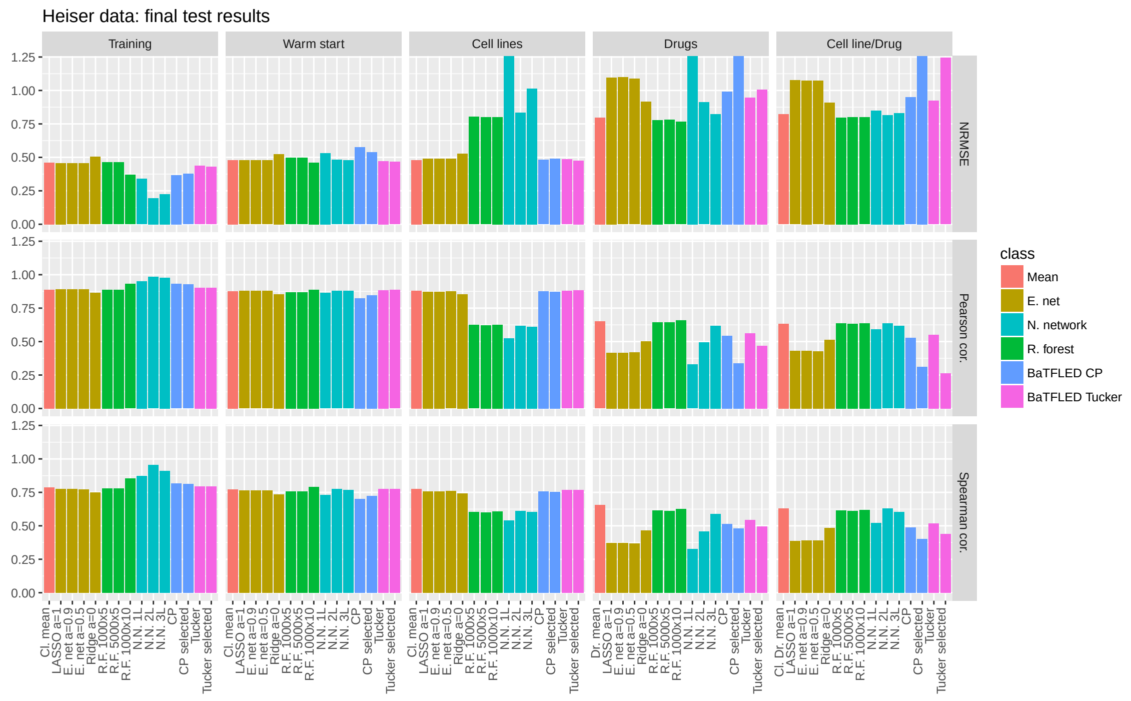


Figure 4.27: Performance measures for final test runs on Heiser data (average across 5 replicates with different random starts). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown.

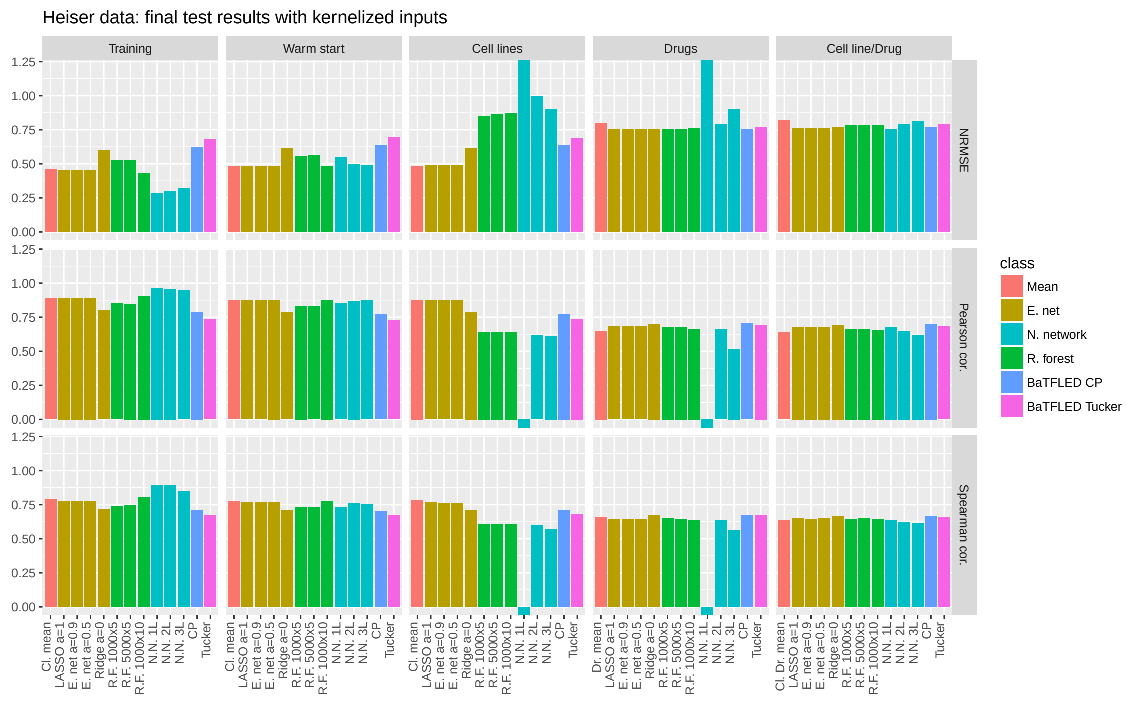


Figure 4.28: Performance measures for final test runs on Heiser data when using kernelized features (average across 5 replicates with different random starts). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown.

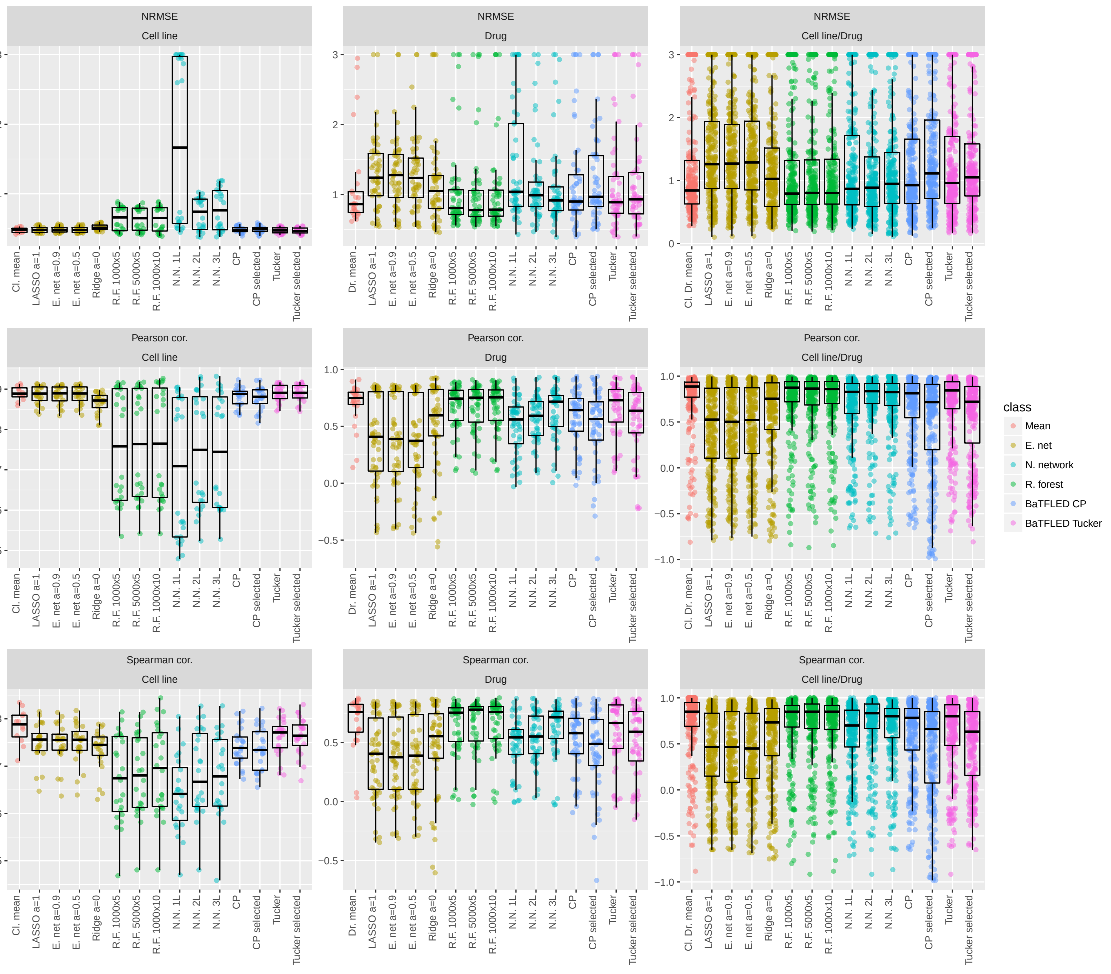


Figure 4.29: Performance measures for final test runs on Heiser data from one randomly selected replicate. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

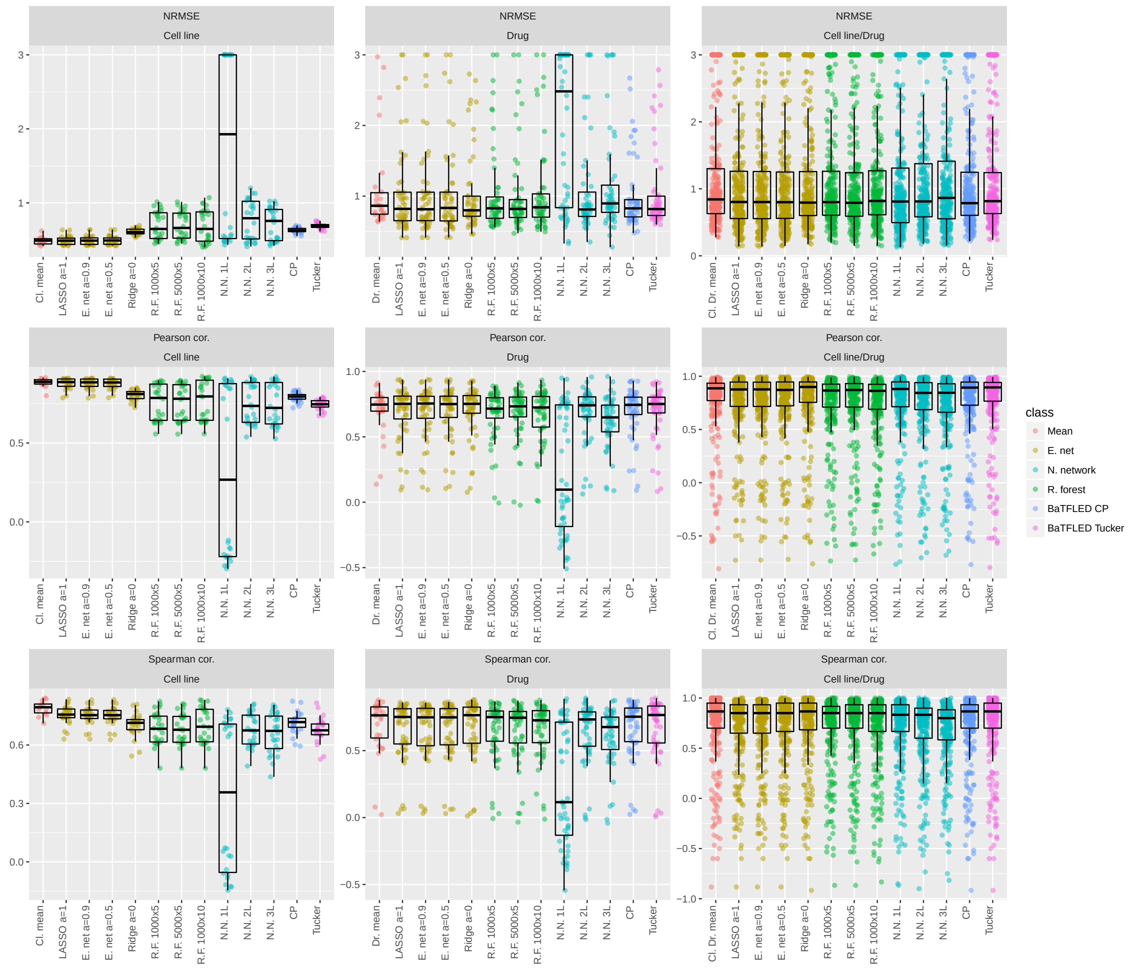


Figure 4.30: Performance measures for final test runs on Heiser data with kernelized features from one randomly selected replicate. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

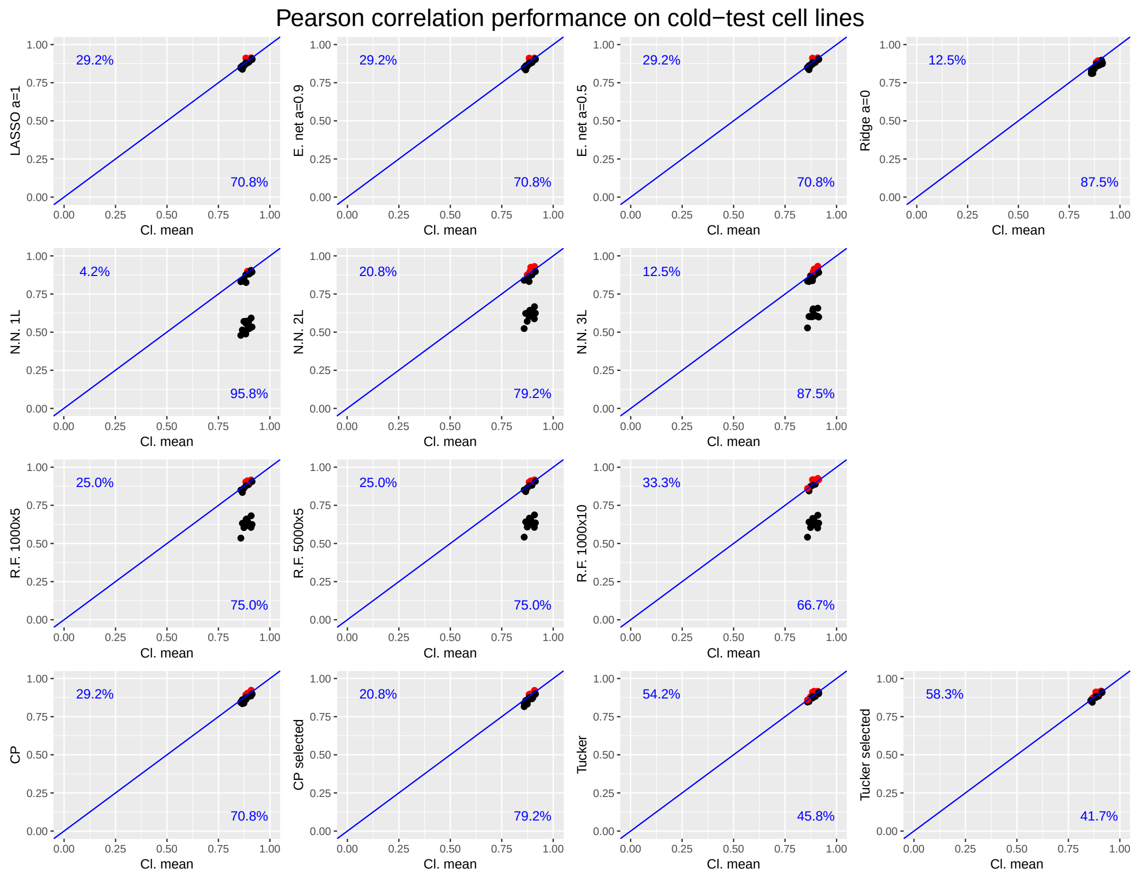


Figure 4.31: Comparison between mean prediction and other methods on the cold-start cell line task for the final Heiser test data. Each point represents one cell line with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell lines on either side of the diagonal are annotated.

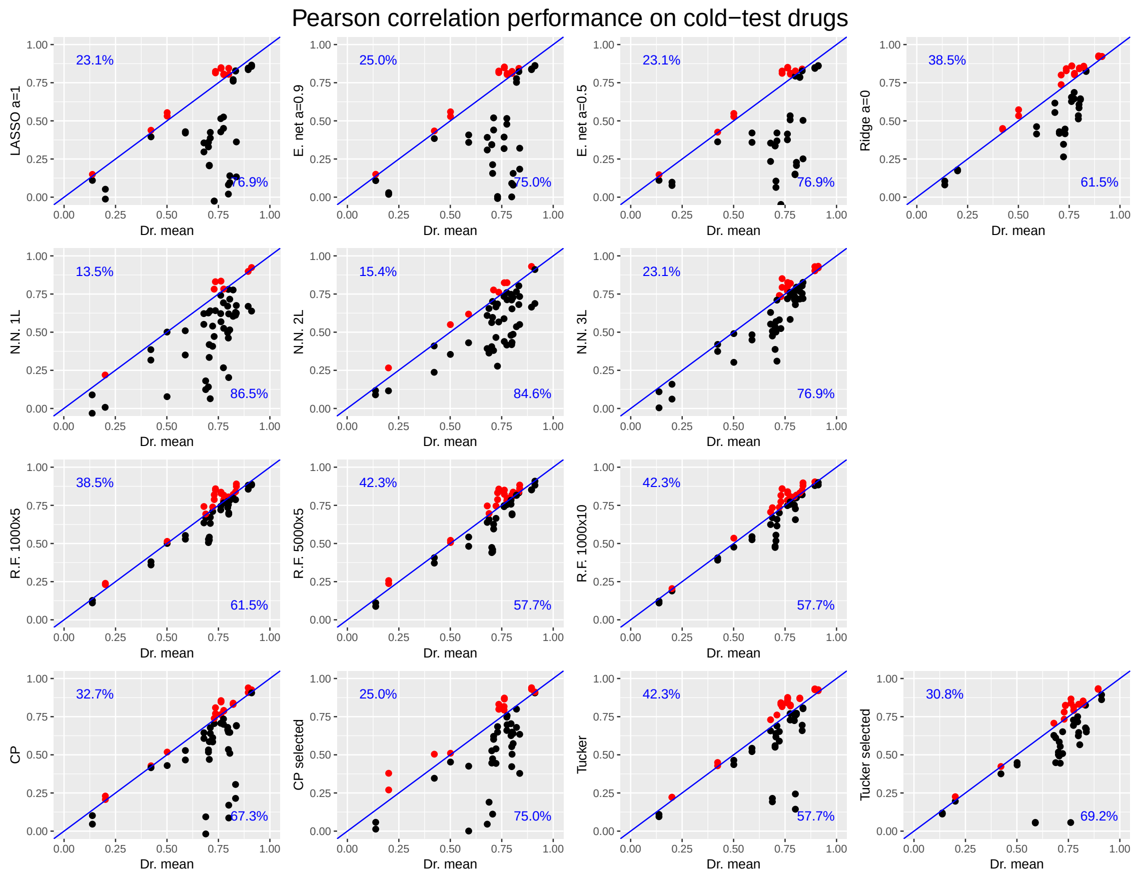


Figure 4.32: Comparison between mean prediction and other methods on the cold-start drug task for the final Heiser test data. Each point represents one drug with its x coordinate showing the Pearson correlation of mean prediction with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of drugs on either side of the diagonal are annotated.

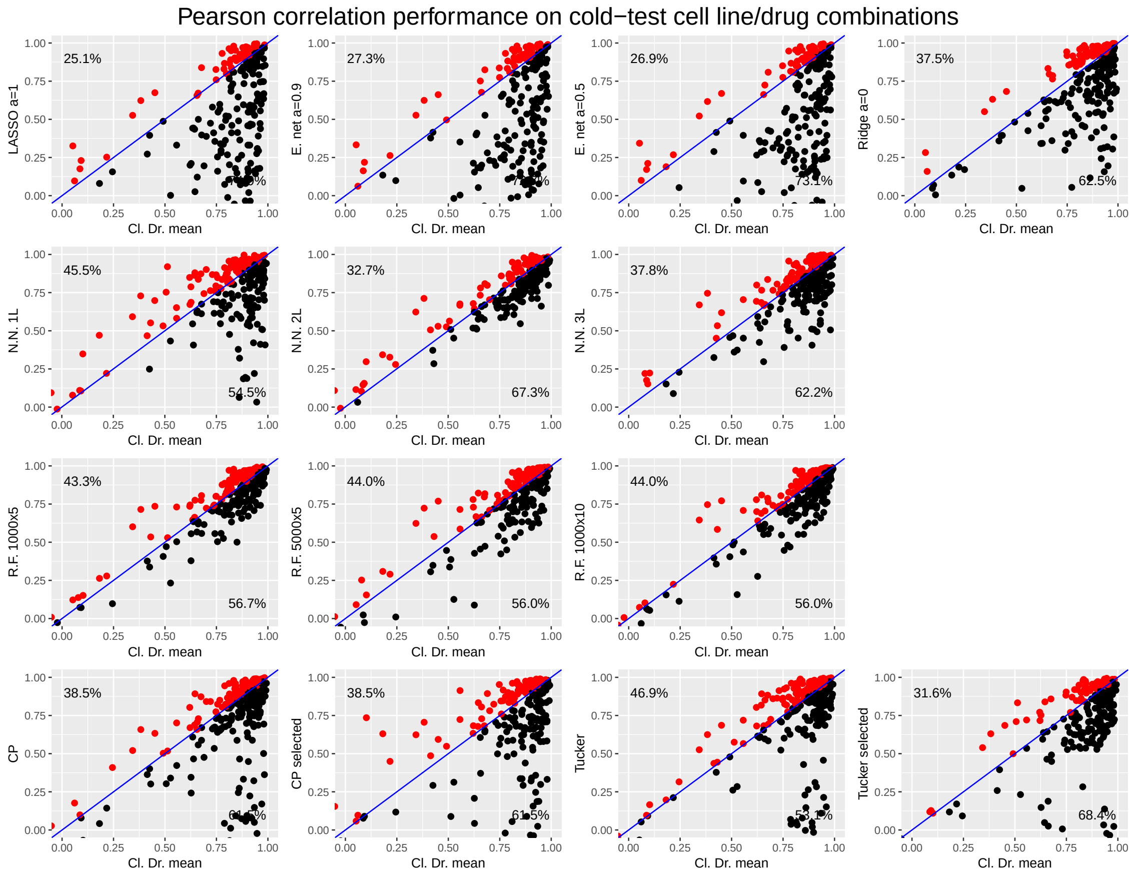


Figure 4.33: Comparison between mean prediction and other methods on the cold-start cell line/drug combination task for the final Heiser test data. Each point represents one cell line/drug combination with its x coordinate showing the Pearson correlation of mean prediction (across cell lines) with the true response and the y coordinate showing the Pearson correlation between the alternate method and the true response. Red points are predicted better by the alternate method and the percentage of cell line/drug combinations on either side of the diagonal are annotated.

4.4.6 GR_{50} prediction

Since the majority of studies have used the GI_{50} response response measure to judge drug sensitivity, we also compared the performance of the above methods in terms of this response. For both the observed and predicted responses, we fit four-parameter logistic functions to responses at each dose using the R package 'drc' [113]. The responses have been normalized to the growth rate for this data set, so this is what the authors of [22] refer to as the GR_{50}. GR_{50} measures were annotated relative to the doses (1..9) and inferred GR_{50} values below 1 were set to 'NA' and above the maximum dose were set to 10. We then measured the NRMSE, Pearson correlation and Spearman correlation between the observed and predicted GR_{50} values for each method. Additionally we calculated these measures for curves fit to the mean response (across cell lines or drugs) at each dose. Tables 4.13 and 4.14 and figure 4.34 show the average performance for these three metrics on the cross-validation runs while tables 4.15 and 4.16 and figure 4.35 show the same for the final test set responses.

Table 4.13: Mean normalized root mean squared error for cross-validation runs for the Heiser dataset when predicting GR_{50} values extrapolated from fitted curves (sd. over five cross-validation folds).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.63(0.01)	0.65(0.03)	0.68(0.08)	1.04(0.04)	1.26(0.14)
LASSO a=1	0.63(0.01)	0.66(0.03)	0.70(0.06)	0.97(0.16)	0.99(0.21)
E. net a=0.9	0.63(0.01)	0.64(0.03)	0.70(0.06)	0.97(0.16)	1.02(0.23)
E. net a=0.5	0.63(0.01)	0.66(0.04)	0.70(0.06)	1.03(0.13)	1.03(0.14)
Ridge a=0	0.62(0.01)	0.68(0.04)	**0.67**(0.06)	1.01(0.11)	0.99(0.11)
R.F. 1000x5	0.61(0.02)	0.64(0.01)	1.02(0.02)	1.03(0.09)	1.06(0.20)
R.F. 5000x5	0.61(0.01)	0.66(0.02)	1.02(0.01)	1.00(0.13)	1.02(0.15)
R.F. 1000x10	0.49(0.00)	**0.61**(0.05)	1.02(0.02)	1.05(0.13)	1.01(0.15)
N.N. 1L	0.40(0.02)	0.68(0.01)	1.04(0.03)	1.13(0.17)	1.07(0.17)
N.N. 2L	**0.26**(0.01)	0.62(0.03)	1.06(0.03)	1.04(0.12)	0.98(0.09)
N.N. 3L	0.28(0.01)	0.62(0.03)	1.82(0.55)	1.24(0.25)	1.13(0.19)
CP	0.44(0.03)	0.93(0.08)	0.68(0.05)	1.15(0.19)	1.04(0.12)
CP retrain	0.45(0.02)	0.80(0.05)	0.68(0.06)	1.15(0.12)	1.13(0.15)
Tucker	0.60(0.02)	0.66(0.02)	0.68(0.07)	1.04(0.14)	1.00(0.16)
Tucker retrain	0.60(0.04)	0.66(0.05)	**0.67**(0.05)	**0.94**(0.10)	**0.89**(0.13)

*Training results on warm-start task.

Table 4.14: Mean Pearson correlation for cross-validation runs for the Heiser dataset when predicting GR_{50} values extrapolated from fitted curves (sd. over five cross-validation folds).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.77(0.00)	0.76(0.02)	0.74(0.05)	0.08(0.04)	-0.01(0.28)
LASSO a=1	0.77(0.00)	0.75(0.02)	0.72(0.05)	0.37(0.32)	0.34(0.34)
E. net a=0.9	0.77(0.01)	0.76(0.03)	0.72(0.05)	0.37(0.32)	0.31(0.37)
E. net a=0.5	0.77(0.01)	0.75(0.03)	0.72(0.05)	0.33(0.28)	0.29(0.32)
Ridge a=0	0.78(0.00)	0.73(0.03)	0.73(0.05)	0.37(0.16)	0.35(0.21)
R.F. 1000x5	0.79(0.01)	0.76(0.01)	0.01(0.04)	0.31(0.16)	0.32(0.23)
R.F. 5000x5	0.79(0.01)	0.75(0.02)	0.03(0.02)	0.36(0.16)	0.32(0.24)
R.F. 1000x10	0.87(0.00)	**0.78**(0.04)	0.01(0.04)	0.33(0.12)	0.36(0.20)
N.N. 1L	0.92(0.01)	0.76(0.01)	0.02(0.03)	0.30(0.16)	0.36(0.16)
N.N. 2L	**0.97**(0.00)	**0.78**(0.03)	0.05(0.03)	0.29(0.27)	0.37(0.22)
N.N. 3L	0.96(0.00)	**0.78**(0.02)	NA(NA)	0.23(0.30)	0.39(0.19)
CP	0.90(0.01)	0.55(0.07)	0.74(0.04)	0.28(0.19)	0.35(0.18)
CP retrain	0.89(0.01)	0.65(0.03)	0.74(0.04)	0.23(0.17)	0.25(0.20)
Tucker	0.80(0.02)	0.76(0.02)	0.75(0.05)	0.31(0.24)	0.31(0.32)
Tucker retrain	0.80(0.03)	0.76(0.04)	**0.76**(0.03)	**0.42**(0.13)	**0.43**(0.23)

*Training results on warm-start task.

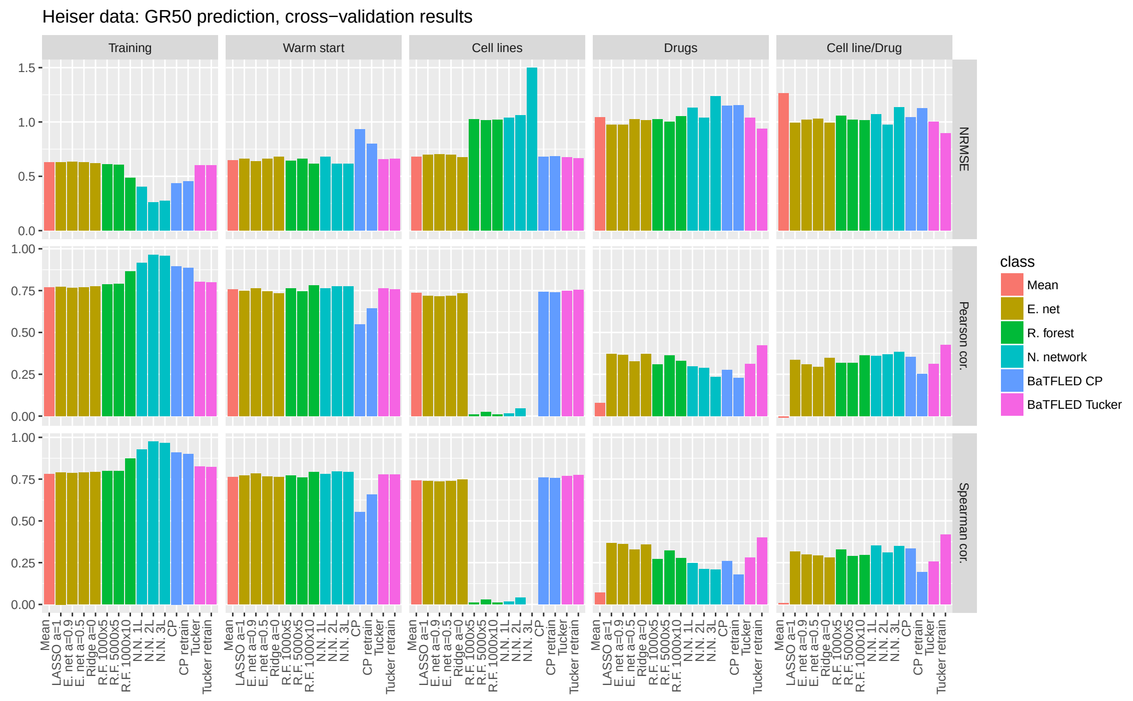


Figure 4.34: Performance measures for cross-validation runs on Heiser data when predicting GR_{50} values extracted from curves. Mean normalized root mean squared, Pearson correlation and Spearman correlation across 5 cross-validation folds for training data and four prediction tasks. Note: scales differ for each response measure and NRMSE values above 1.5 are set to 1.5 for visualization purposes.

Table 4.15: Mean normalized root mean squared error on the final test data for the Heiser dataset when predicting GR_{50} values extrapolated from fitted curves (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.63(0.00)	0.65(0.00)	**0.61**(0.00)	1.10(0.00)	1.47(0.00)
LASSO a=1	0.64(0.00)	0.64(0.00)	0.63(0.00)	1.23(0.04)	1.23(0.09)
E. net a=0.9	0.64(0.00)	0.65(0.00)	0.63(0.00)	1.24(0.03)	1.19(0.05)
E. net a=0.5	0.64(0.00)	0.65(0.00)	0.63(0.00)	1.11(0.02)	1.14(0.05)
Ridge a=0	0.67(0.00)	0.70(0.00)	0.67(0.00)	1.56(0.01)	1.52(0.01)
R.F. 1000x5	0.60(0.00)	0.64(0.01)	1.01(0.02)	1.09(0.02)	1.09(0.03)
R.F. 5000x5	0.61(0.01)	0.64(0.01)	1.03(0.02)	1.10(0.04)	1.08(0.02)
R.F. 1000x10	0.49(0.00)	0.62(0.01)	1.01(0.00)	**1.06**(0.01)	**1.07**(0.01)
N.N. 1L	0.44(0.03)	0.72(0.03)	1.59(0.10)	2.04(0.20)	1.09(0.10)
N.N. 2L	**0.28**(0.02)	**0.59**(0.00)	1.03(0.01)	1.22(0.13)	1.26(0.03)
N.N. 3L	0.29(0.01)	0.61(0.01)	1.08(0.03)	1.07(0.07)	1.18(0.07)
CP	0.47(0.02)	0.80(0.02)	**0.61**(0.01)	1.25(0.01)	1.29(0.04)
CP selected	0.48(0.01)	0.74(0.01)	0.63(0.02)	1.23(0.03)	1.13(0.06)
Tucker	0.59(0.03)	0.62(0.01)	**0.61**(0.01)	1.26(0.03)	1.35(0.03)
Tucker selected	0.59(0.04)	0.64(0.02)	0.63(0.02)	1.36(0.02)	1.50(0.05)

*Training results on warm-start task.

Table 4.16: Mean Pearson correlation on the final test data for the Heiser dataset when predicting GR_{50} values extrapolated from fitted curves (sd. over five replicates with different random starts).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.77(0.00)	0.76(0.00)	0.78(0.00)	0.08(0.00)	-0.08(0.00)
LASSO a=1	0.77(0.00)	0.76(0.00)	0.77(0.00)	**0.22**(0.05)	0.25(0.06)
E. net a=0.9	0.77(0.00)	0.76(0.00)	0.77(0.00)	0.20(0.03)	0.22(0.06)
E. net a=0.5	0.77(0.00)	0.77(0.00)	0.77(0.00)	**0.22**(0.02)	0.22(0.05)
Ridge a=0	0.73(0.00)	0.71(0.00)	0.73(0.00)	-0.11(0.00)	-0.06(0.00)
R.F. 1000x5	0.79(0.00)	0.77(0.01)	-0.01(0.03)	0.11(0.02)	0.16(0.04)
R.F. 5000x5	0.79(0.00)	0.77(0.01)	0.01(0.03)	0.09(0.05)	0.16(0.02)
R.F. 1000x10	0.87(0.00)	0.78(0.00)	0.03(0.01)	0.12(0.02)	0.20(0.02)
N.N. 1L	0.90(0.01)	0.74(0.01)	0.07(0.01)	-0.09(0.12)	**0.30**(0.10)
N.N. 2L	**0.96**(0.00)	**0.81**(0.00)	0.08(0.03)	0.16(0.11)	0.09(0.01)
N.N. 3L	**0.96**(0.00)	0.79(0.01)	0.06(0.01)	0.13(0.06)	0.27(0.03)
CP	0.88(0.01)	0.64(0.01)	**0.79**(0.01)	0.18(0.02)	0.14(0.02)
CP selected	0.87(0.01)	0.70(0.01)	0.78(0.01)	0.17(0.03)	0.27(0.05)
Tucker	0.81(0.02)	0.79(0.01)	0.78(0.01)	0.16(0.03)	0.10(0.02)
Tucker selected	0.81(0.03)	0.78(0.01)	0.78(0.01)	-0.03(0.03)	-0.12(0.03)

*Training results on warm-start task.

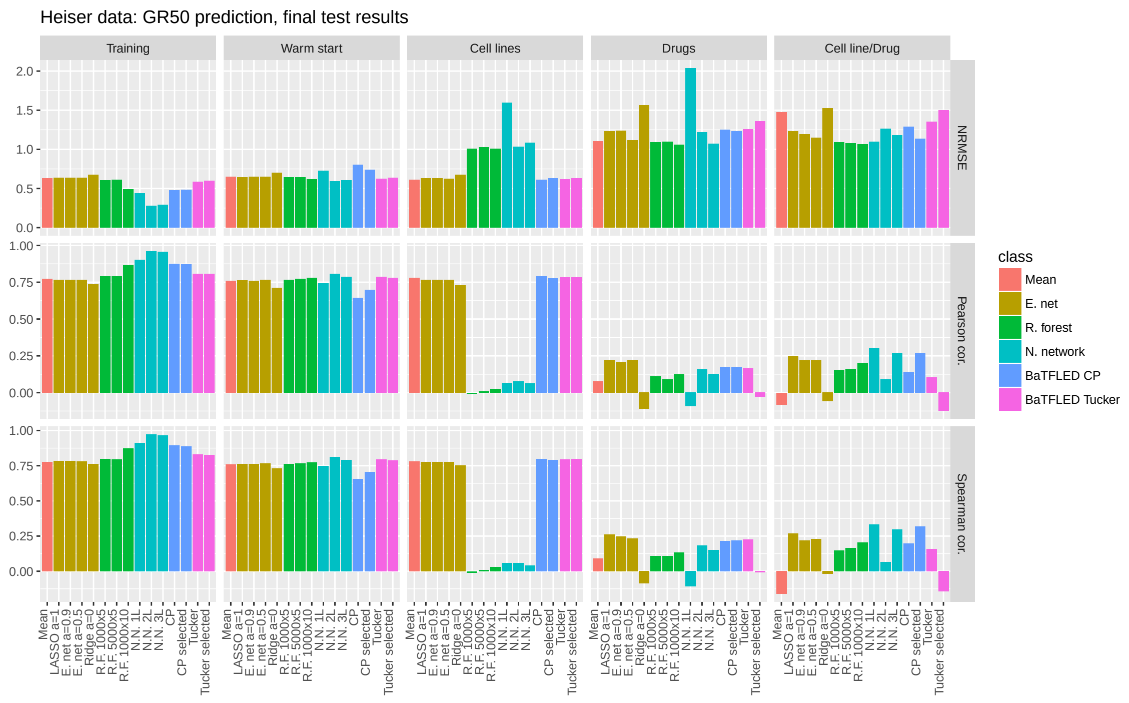


Figure 4.35: Performance measures for final test runs on Heiser data when predicting GR_{50} values extracted from curves. Mean normalized root mean squared, Pearson correlation and Spearman correlation across 5 replicates with different random starts for training data and four prediction tasks. Note: scales differ for each response measure.

We also measured the performance of the baseline algorithms when predicting the GR$_{50}$ values directly rather than predicting response at each dose, fitting curves and calculating GR$_{50}$ values. In general, we find a worse performance in almost all cases, supporting the hypothesis that there is some benefit to dealing with responses directly. Figure 4.36 shows the performance for mean prediction, elastic net, random forest and neural net models on this task when using the same run parameters as above.

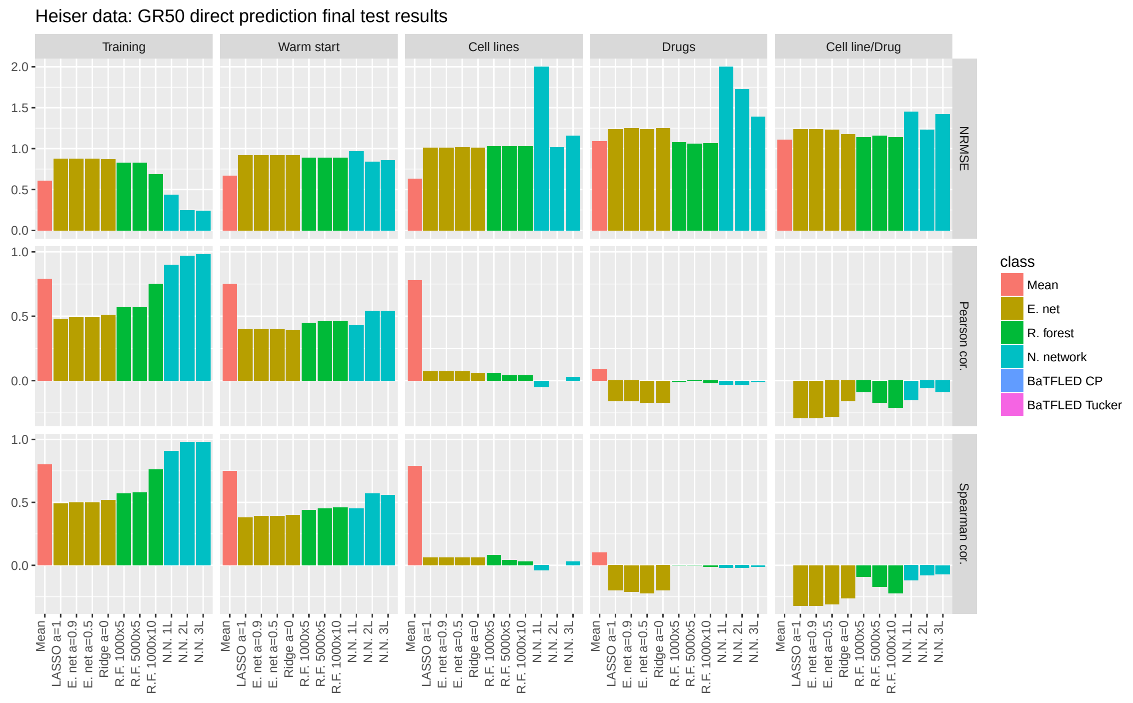


Figure 4.36: Performance measures on final test cases for Heiser data when predicting GR$_{50}$ values directly. Mean normalized root mean squared, Pearson correlation and Spearman correlation across 5 replicates with different random starts for training data and four prediction tasks. Note: scales differ for each response measure and NRMSE values above 2 have been set to 2 for visualization purposes.

Taken together, we see from the above tests that the BaTFLED models perform as well or better than completing methods at most tasks. In particular, on the cross-validation runs, the retrained Tucker model is the best performing model at all the cold-start tasks, while for the final test data, BaTFLED models perform best on the cold-start cell line task without kernelized features and are tied for best on the cold-start drug task with kernelized features. Interestingly, the feature selection step does not seem to confer much of an advantage in this evaluation suggesting that the predictive value of the input features may be broadly spread across the input data. Additionally we find that BaTFLED methods perform comparably when predicting GR_{50} responses extracted from curves and that all methods do better by first predicting responses at each dose rather than predicting $GR_5 0$ values directly.

4.4.7 Selected features

As with the DREAM data we explored which features were selected by the different prediction algorithms that we tested. We consider the union of the top 15% of features across the five replicate runs on the final test data for LASSO, two layer neural net, and random forest with 1000 trees at a depth of 10 nodes. For CP and Tucker models, we consider the same number of top features in the 'selected' runs (after the first round of selection by the cross-validation runs). For cell lines, these features consist of 5 subtype indicators, median doubling time without treatment and expression values for 583 genes. For drugs, features include 27 target and class indicators, 271 PubChem fingerprint features and 432 PaDEL structural indicators. The LASSO model only selected $10-11\%$ of cell line features during each run and these were fairly consistent so the total number of these features is less than 15%.

The methods differed widely in which features they chose with the LASSO method selecting the fewest features and the Tucker model selecting the most (Table 4.17). For cell line features, LASSO, random forest, and Tucker models chose both the subtype and expression features about equally, while the neural net and CP model preferred subtype features (Figure 4.37). For drugs, LASSO and random forest models chose almost exclusively PaDEL structural features while the neural net model preferred class/target features (Figure 4.38). BaTFLED models chose almost entirely fingerprint and PaDEL features (about equally) and were fairly consistent in which features they chose (Figure 4.39). In fact, the CP and Tucker models share about half of their chosen features (72 of 153 and 154 features respectively).

Table 4.17: Number of features selected in the union of the top 15% across replicates for Heiser data (percentage).

	LASSO	Neural net	Random forest	CP	Tucker
Cell lines	71 (12.1)	98 (16.6)	155 (26.3)	163 (27.7)	201 (34.1)
Drugs	113 (15.5)	116 (15.9)	145 (19.9)	153 (21.0)	154 (21.1)

No cell line features were selected by all methods and 15 features were selected by four of the methods. These included the median doubling time and expression values for the following genes: AR, BIRC3, CBLB, EGFR, ETV5, FAM22A, FAM46C, FLT4, KIT, NOTCH1, PPARG, TMPRSS2, TRIM24 and VTI1A. The only method that outperformed mean prediction for the cold-start cell line task was the BaTFLED Tucker model. For this model, the cell line features that were in the top 15% of features across all the five replicates were expression values for ARHGAP26, FAM46C, FLT4, HOXC11, KLF4, NFATC2, SDHA and SPOP.

Looking at the cold-start drug prediction runs, no drug features are selected by all methods and four PaDEL features were selected by four methods: AATS5m (an autocorrelation descriptor), maxHBint4 (a hydrogen bond strength measure), MDEC.22 (molecular distance between secondary carbons), and a combination feature of several values measuring the coefficient sum of the last eigenvector from Barysz matrix. The best performing method for cold-start drug prediction was the random forest model which had 76 PaDEL and two footprint features in the top 15% across all replicates. The best performing BaTFLED run was the Tucker model using all features. This model had 51 features consistently selected among replicates, the indicator of the HDAC targeting class, 34 PaDEL and 16 fingerprint features.

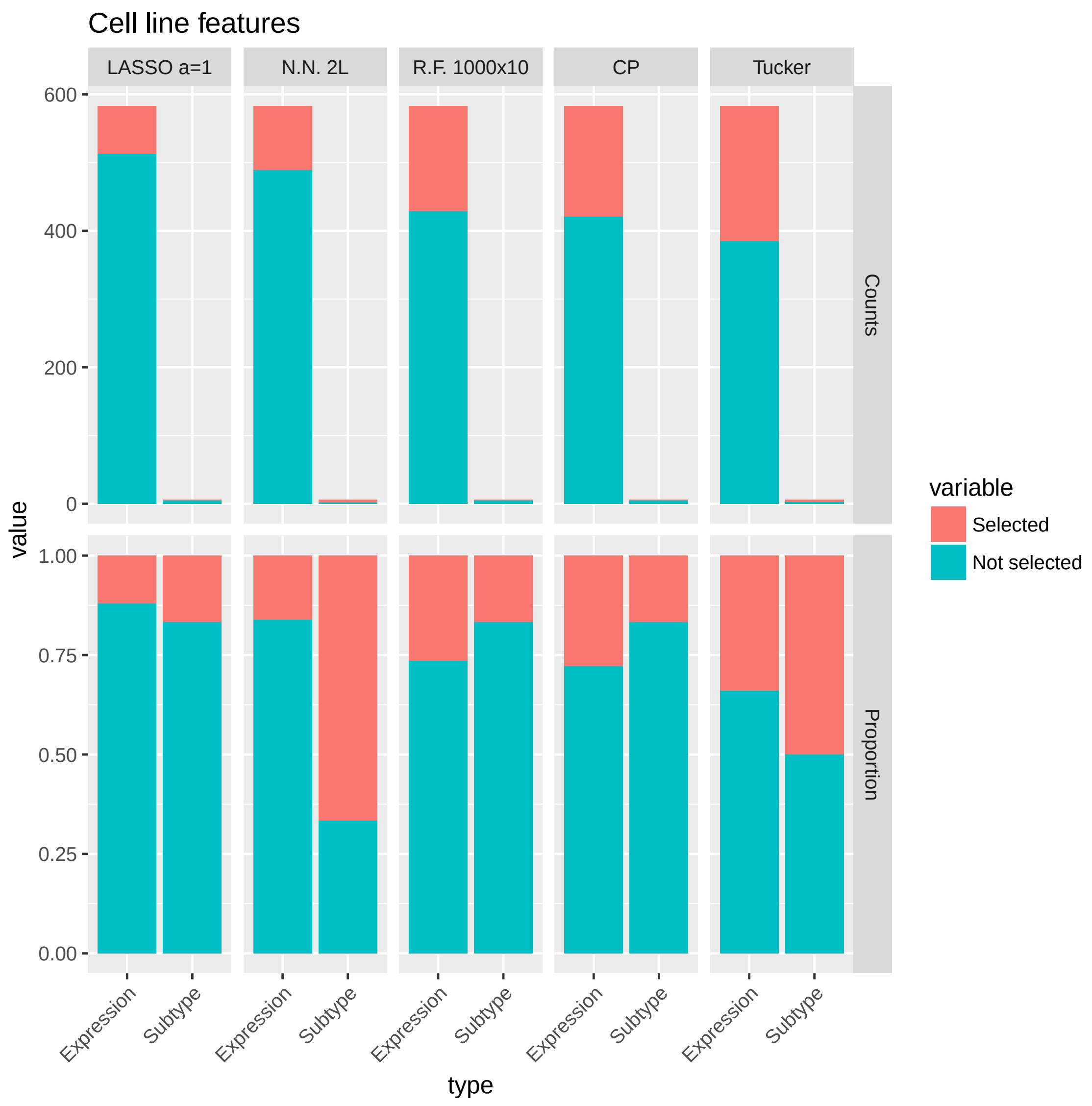


Figure 4.37: Top row: Number of cell line features selected by different methods. Bar height shows the total number of features of the given type, red portion shows the number of these features in the union of the top 15% of features across replicates. Note that for the LASSO model, only about $10 - 11\%$ of features had non-zero values in each replicate. Bottom row: Proportion of total features from each category that are chosen.

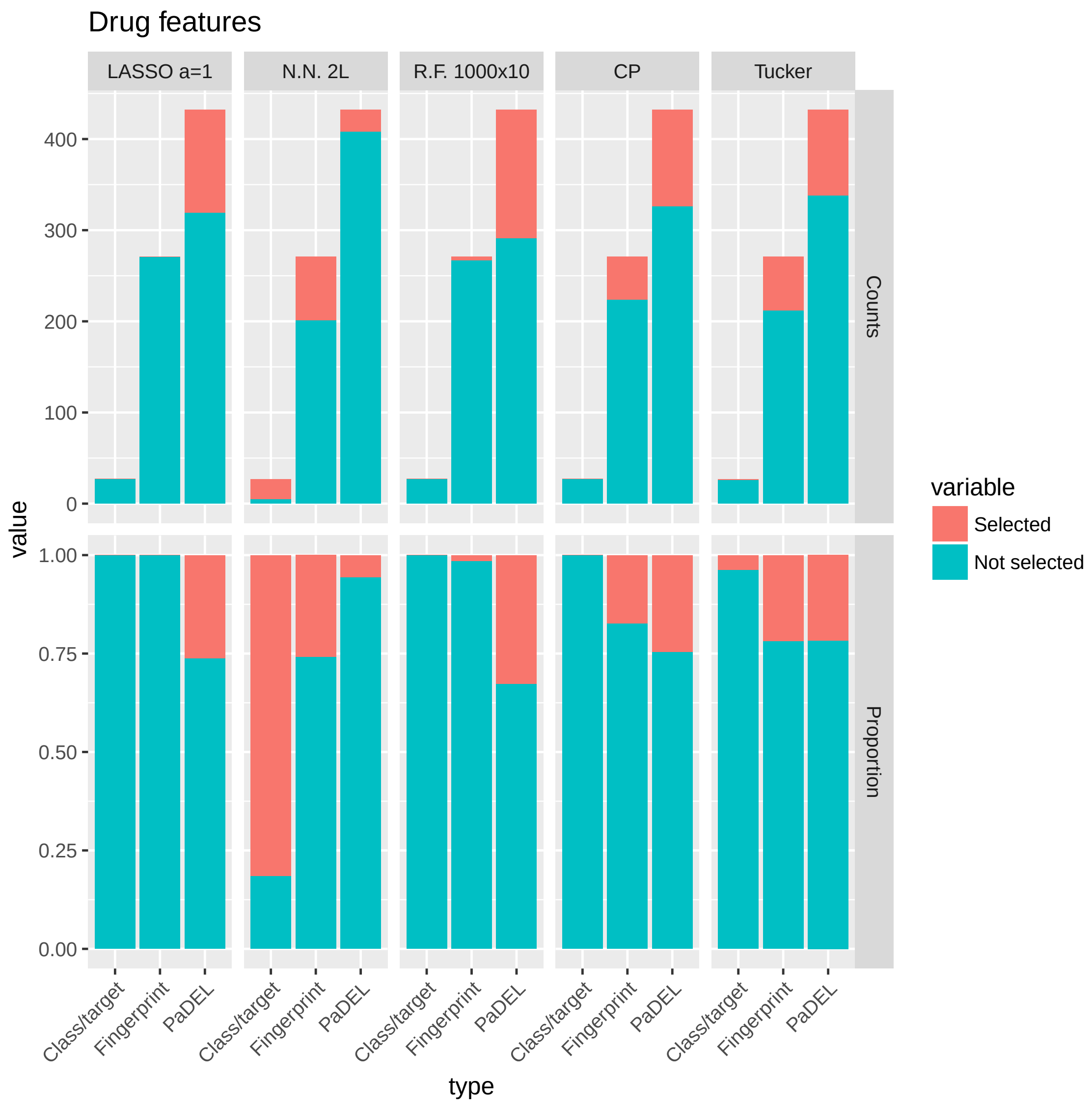


Figure 4.38: Top row: Number of drug features selected by different methods. Bar height shows the total number of features of the given type, blue portion shows the number of these features in the union of the top 15% of features across replicates. Bottom row: Proportion of total features from each category that are chosen

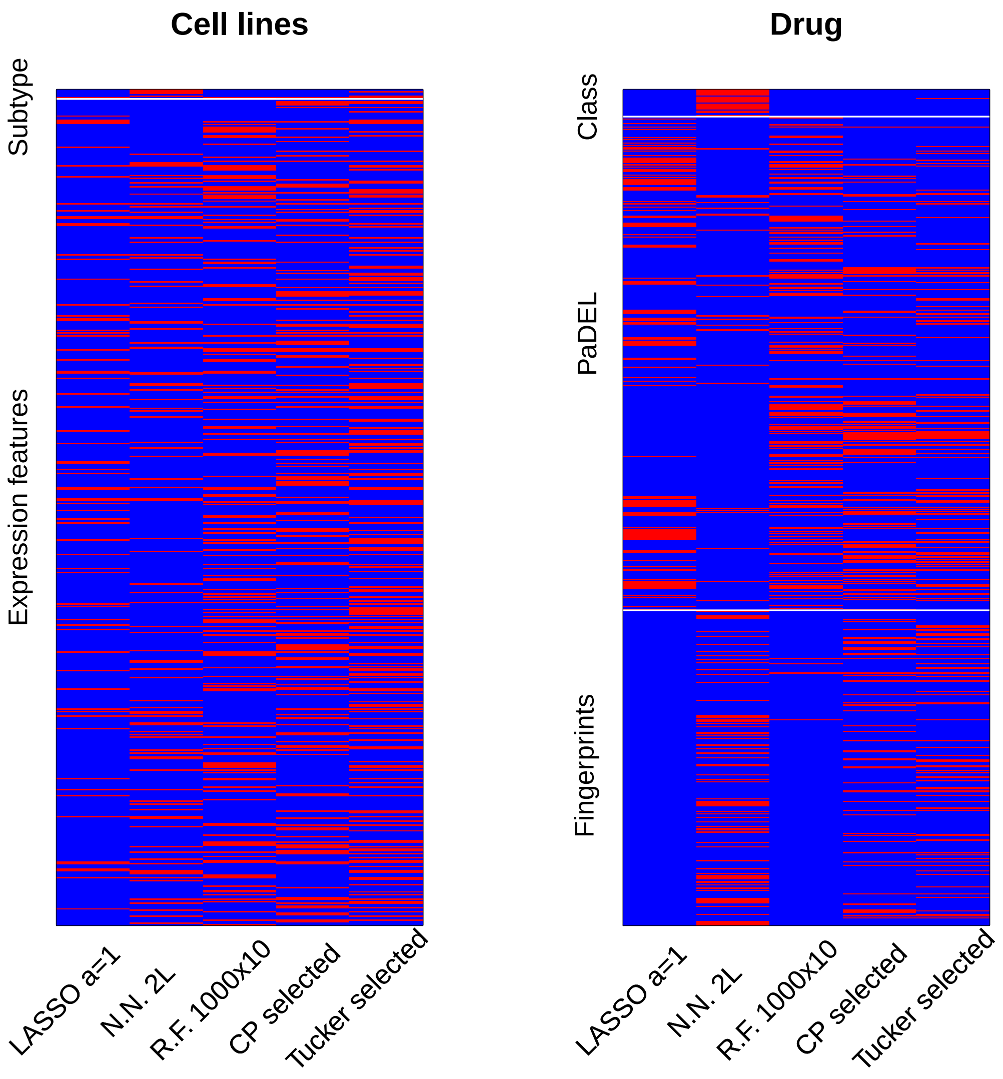


Figure 4.39: Features selected by different methods for the Heiser data. Each row is a feature, features selected in the top 15% of features for any replicate are colored red. Note that different methods tend to choose different features.

4.5 CTD^2 data

The Cancer Target Discovery and Development (CTD^2) effort is the largest cancer cell line screening study with publicly available data that has been published to date. There have been two iterations of this effort, the first, CTRP1 (Cancer Therapeutics Response Portal) having 242 cell lines, 185 drugs and 8 doses was published in 2013 [108] and a the second CTRP2 with 860 cell lines and 481 drugs measured at 16 doses published in 2014 [114, 115]. The cell lines are obtained from a broad range of cancers and the compounds were chosen as an 'informer set' of different drug classes.

4.5.1 Response data

Raw and normalized response data for 907 cell lines, 545 drugs was downloaded from the CTD^2 web portal (https://ocg.cancer.gov/programs/ctd2/data-portal). These values consist of CellTiter-Glo (www.promega.com) luminescence measurements quantifying the amount of ATP present in wells as a proxy for live cell counts. The responses are normalized to percent viability relative to DMSO controls as described in [114]. Of these cell lines, 887 have meta-data associated with them describing the experiments in which they were tested with the vast majority (868) being tested in only one experiment (18 have 2 experiment ids and 1 has 3). All but one have a unique growth mode (adherent: 794, mixed: 18, suspension 233 or spheroid 16) and we excluded the results from the less common growth mode for this cell line (id 1095). Five cell lines were grown in two different media and for these we only kept results from the more common media. Normalized response data was provided in a 'post-qc' dataset which had only 13 cell lines with replicate experiments. We keep the replicate with the higher experiment number assuming these were re-run due to some technical issue. Most cell line/drug combinations ($313,105$ of $463,901$) were tested at 16 doses. We removed two drugs (ids 290356 and 418166) with more than 16 doses and matched doses where possible when combinations had less than 16. Additionally we removed cell lines which had been tested with less than 20% of drugs and drugs which were tested with less than 20% of cell lines and we removed responses for cell lines and drugs for which we could not obtain features (below). This left a final tensor of normalized responses for 783 cell lines 493 drugs and 16 doses with 1,278,816 missing values (20.7%).

4.5.2 Cell line features

All feature data for 1,107 cell lines was downloaded from the CTD^2 web portal with the exception of the mutation data which came from the CCLE effort [10] (downloaded March 1 2016). Overall these data contain, copy number values for 23,174 genes in 823 cell lines, gene expression data for 18,543 genes in 823 cell lines, and mutation data for 1,638 genes in 1,107 cell lines. To reduce the number of features, we again focused on COSMIC genes (557 copy number features, 542 expression features and 458 mutation features). As with the other datasets we combined features correlated above 0.9 or below -0.9 which reduced the copy number features to 475. Additionally, we removed mutation features that were found in less than three cell lines leaving 387 features. We note that 76 cell lines (9.6%) had no mutations annotated.

We used the metadata provided with the responses to construct Binary indicators of growth mode, culture media, primary site, histology and subtype. There were four growth modes (adherent 673, mixed 17, spheroid 16 and suspension 182) and 72 different culture media. However, only 13 of these have more than 10 cell lines with the most common being DMEM001: 81, EMEM001: 54, RPMI001: 408 and RPMI003: 49. There were 24 different primary sites listed including autonomic ganglia (17), biliary tract (8), bone (29), breast (60), central nervous system (69), endometrium (28), haematopoietic and lymphoid tissue (181), kidney (36), large intestine (62), liver (27), lung (187), oesophagus (27), ovary (52), pancreas (46), pleura (11), prostate (8), salivary gland (2), skin (62), small intestine (1), soft tissue (21), stomach (38), thyroid (12), upper aerodigestive tract (33) and urinary tract (28). Histology annotation was also provided, with the vast majority (642) being carcinomas and only five others with more than 10 cell lines (glioma: 62, haematopoietic neoplasm: 51, malignant melanoma 61, neuroblastoma 17 and mesothelioma 11). Finally there were 67 subtypes listed, but only 20 that had more than 10 cell lines: acute lymphoblastic B cell leukaemia (15), acute lymphoblastic T cell leukaemia (15), acute myeloid leukaemia (35), adenocarcinoma (152), astrocytoma (11), astrocytoma Grade IV: (38), chronic myeloid leukaemia (12) chronic myeloid leukaemia (11), clear cell renal cell carcinoma (12), diffuse large B cell lymphoma (18), ductal carcinoma (61), hepatocellular carcinoma (24), Hodgkin lymphoma (12), large cell carcinoma (14), non small cell carcinoma (24), plasma cell myeloma (30), renal cell carcinoma (17), small cell carcinoma (54), squamous cell carcinoma (81), and transitional cell carcinoma (22). In total, we generated 188 indicators, removed those with less than 10 cell lines (leaving 58 indicators) and removed 5 more that were highly correlated (greater than 0.9 or lesss than -0.9) consisting of tissue indicators matching subtypes. Additionally, we made a binary indicator of whether average of log2 DMSO readout in untreated cells was above or below 14.5. This cutoff was decided based on visual examination of histograms in which there were two clear populations with 65 cell lines (7.33%) in the upper group. Bar plots of counts for these indicators are given in figure 4.40 and a heatmap of indicators is shown in figure 4.41.

Removing features with no variation or less than two non-zero values (after removing the cell lines without one or more datatype) leaves 783 cell lines, $43,332$ total features and $1,456$ COSMIC features (52 annotations, 542 gene expression values, 387 gene mutation values and 475 gene copy number values).

4.5.3 Drugs features

For the drug data, there were 514 compounds with response data and we used gene targets, PubChem fingerprints and PaDEL structural descriptors as features. Starting with a total of 324 gene targets listed in the annotations downloaded from CTD^2, we removed all but 77 that were annotated for at least three drugs. Next we used the SMILES string and the PaDEL software ([110]) to obtain $1,441$ structual features and 881 PubChem fingerprints for the 493 drugs which were able to be processed. We removed features with no variation and features where less than 10 drugs were non-zero or non-one (leaving $1,619$). We then applied a $\log_{10}$ transformation to 325 non-binary features if the transformed version was closer to a normal distribution (i.e. had a larger Shapiro-Wilk test statistic). We also z-scaled all features and set values more than 4 standard deviations from the mean to 4 or -4. There were $1,103$ (0.18%) values changed on the low end of the distribution and $3,228$ (0.53%) on the high end. Finally we set $8,382$ (2.25%) 'NA' values to zero, combined

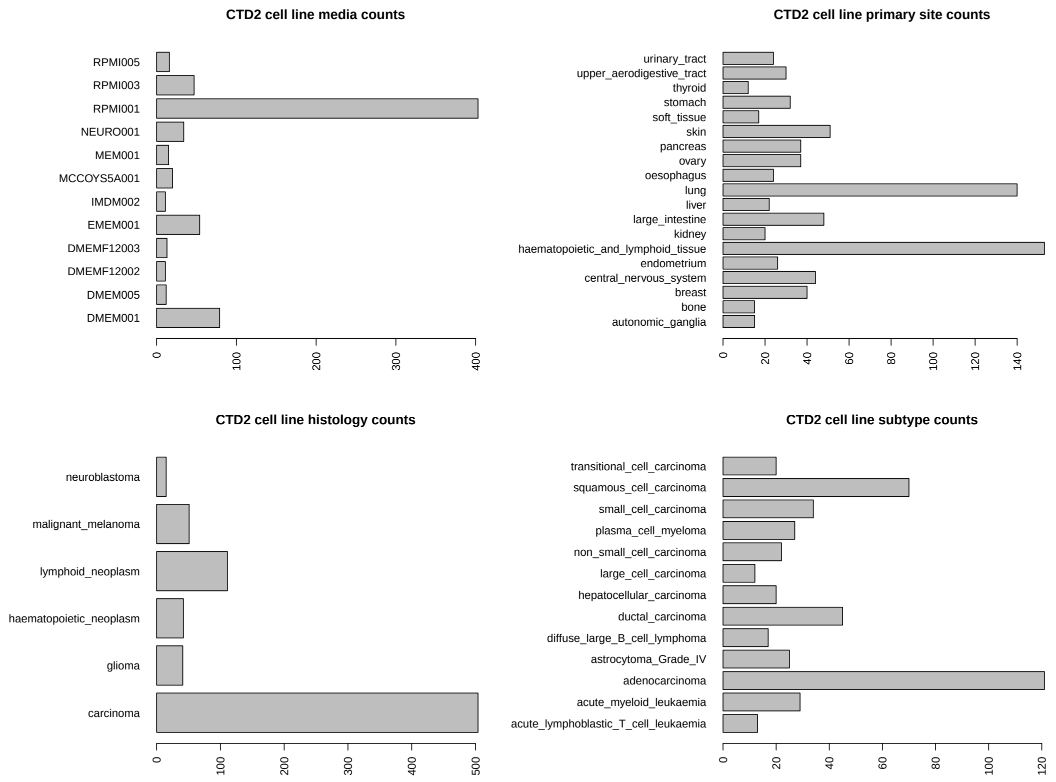


Figure 4.40: Barplots of counts for cell line annotations in 1,107 cell lines in the CTD^2 dataset.

865 highly correlated features (Pearson correlation above 0.9 or below -0.9) and re-scaled. The final set has 493 drugs and 831 features (77 targets, 161 pubchem indicators and 593 1D and 2D structural features).

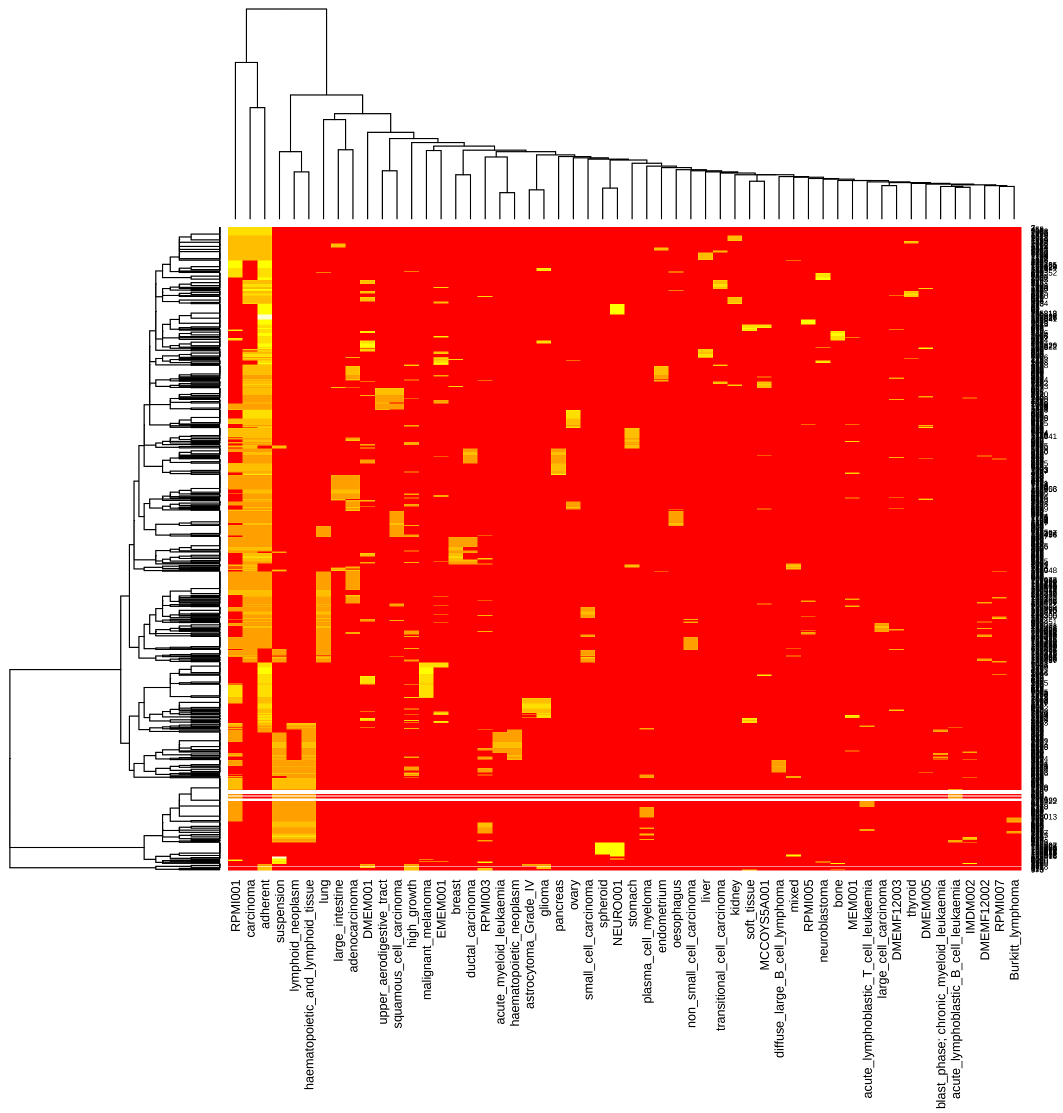


Figure 4.41: Heatmap of 63 binary indicators of 1,107 cell lines in the CTD^2 dataset. Each row represents one cell line and each column an indicator. Colors show whether the cell line has a given annotation, red means no, yellow means yes with darker colors indicating more positive indicators for that cell line.

4.5.4 Experiments

With the CTD^2 data, as with the Heiser data, we perform separate train/test splits for each task holding out about 20% of the data and ran a 5-fold cross validation to select hyper-parameters and features. For the final warm start task, we removed $62,802$ cell line/drug combinations from the $314,011$ combinations that had response data and for cross-validation runs held out an additional randomly selected 20%. For cold-start cell line prediction tasks, we held out a final test set of 156 of the 783 cell lines and removed an additional 126 during each cross-validation fold. For cold-start drug prediction tasks, we held out 98 drugs of the 493 drugs as the final test set and removed an additional 79 drugs during each cross-validation run. For testing on the cold-start cell line/drug task, we used the same training and test splits as above.

Due to size of this dataset and computational costs, we only compared the performance of BaTFLED models to mean prediction and the four elastic net models used previously. Although there were a few tasks where neural net and random forest methods outperformed elastic net models (namely cold-start drug prediction with unkernelized features on the Heiser data), most of the time the elastic net models were comparable the top performers. BaTFLED CP models were trained with a latent dimension of 200 and BaTFLED Tucker models with a $10 \times 10 \times 10$ core. All BaTFLED models were run for 50 iterations with strong sparsity priors on the projection matrices ($\alpha = 10^{-10}$, and $\beta = 10^{10}$) when using all features and weak sparsity priors ($\alpha = \beta = 1$) when using features selected by the cross-validation runs.

4.5.5 Performance

In the cross-validation runs, BaTFLED models perform about as well as elastic net models for warm-start and cold-start cell line prediction. Although BaTFLED models appear worse than elastic net models in terms of NRMSE and Pearson correlation when predicting for drugs and cell line/drug combinations, they have a slight advantage in terms of Spearman correlation (tables 4.18, 4.19 and 4.20 and figure 4.42. If we look at the performance for individual cell lines and drug instead of measuring responses in aggregate, we see that the median performance of BaTFLED models is actually better than elastic net models for these tasks with all three measures (Figure 4.43). This may indicate that BaTFLED models are more prone to making larger errors than elastic net models since the average results shown in figure 4.42 are more sensitive to outliers than the medians in figure 4.43. Alternatively, there may be some Simpson's paradox effects when considering all predictions versus considering predictions for each cell line or drug. Ultimately, mean prediction performs best on all cold-start tasks, indicating that the features are not very useful in this dataset. This may be due to the way that the feature data was cleaned and processed or due to the size and heterogeneity of this pan-cancer data.

Table 4.18: Mean normalized root mean squared error NRMSE on the CTD^2 dataset over five cross-validation folds (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.59(0.00)	0.59(0.00)	**0.59**(0.01)	**0.82**(0.02)	**0.85**(0.02)
LASSO a=1	**0.55**(0.00)	**0.55**(0.00)	0.61(0.01)	1.13(0.06)	1.15(0.07)
E. net a=0.9	**0.55**(0.00)	**0.55**(0.00)	0.61(0.01)	1.13(0.06)	1.15(0.06)
E. net a=0.5	**0.55**(0.00)	**0.55**(0.00)	0.61(0.01)	1.12(0.06)	1.15(0.06)
Ridge a=0	0.58(0.00)	0.58(0.00)	0.62(0.01)	1.00(0.03)	1.03(0.03)
CP	0.56(0.01)	0.56(0.01)	0.62(0.01)	11.47(23.52)	17.09(36.05)
Tucker	0.59(0.02)	0.59(0.02)	0.66(0.01)	5.53(10.09)	13.73(28.46)

*Training results on warm-start task.

Table 4.19: Mean Pearson correlation on the CTD^2 dataset over five cross-validation folds (sd).

	Training	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.81(0.00)	0.81(0.00)	**0.81**(0.01)	**0.57**(0.03)	**0.53**(0.03)
LASSO a=1	**0.84**(0.00)	**0.84**(0.00)	0.80(0.01)	0.40(0.03)	0.38(0.04)
E. net a=0.9	**0.84**(0.00)	0.83(0.00)	0.80(0.01)	0.41(0.03)	0.38(0.04)
E. net a=0.5	**0.84**(0.00)	0.83(0.00)	0.80(0.01)	0.41(0.03)	0.38(0.04)
Ridge a=0	0.82(0.00)	0.82(0.00)	0.78(0.01)	0.45(0.02)	0.42(0.03)
CP	0.83(0.00)	0.83(0.00)	0.78(0.01)	0.40(0.22)	0.38(0.21)
Tucker	0.81(0.01)	0.81(0.01)	0.75(0.01)	0.36(0.21)	0.35(0.20)

*Training results on warm-start task.

Table 4.20: Mean Spearman correlation on the CTD^2 dataset over five cross-validation folds (sd).

	Training	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.64(0.00)	**0.64**(0.00)	**0.64**(0.01)	**0.49**(0.03)	**0.45**(0.03)
LASSO a=1	**0.65**(0.00)	**0.64**(0.00)	0.59(0.01)	0.34(0.03)	0.31(0.03)
E. net a=0.9	**0.65**(0.00)	**0.64**(0.00)	0.59(0.01)	0.34(0.03)	0.31(0.03)
E. net a=0.5	**0.65**(0.00)	**0.64**(0.00)	0.59(0.01)	0.34(0.03)	0.31(0.03)
Ridge a=0	0.63(0.00)	0.62(0.00)	0.58(0.00)	0.38(0.02)	0.35(0.03)
CP	0.64(0.01)	**0.64**(0.00)	0.58(0.01)	0.40(0.01)	0.38(0.02)
Tucker	0.62(0.01)	0.62(0.01)	0.57(0.01)	0.38(0.04)	0.37(0.02)

*Training results on warm-start task.

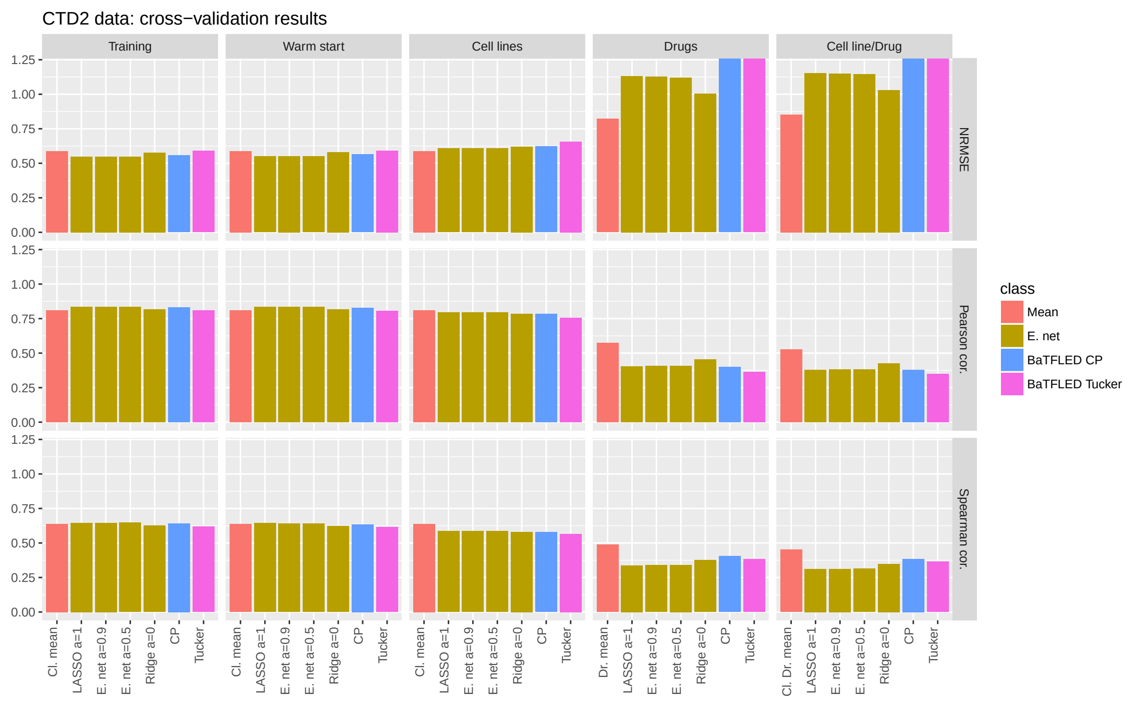


Figure 4.42: Performance measures for cross-validation runs on CTD^2 data (average across 5 runs). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown. Note scales differ for each response measure and NRMSE values above 1.25 have been set to 1.25 for visualization purposes.

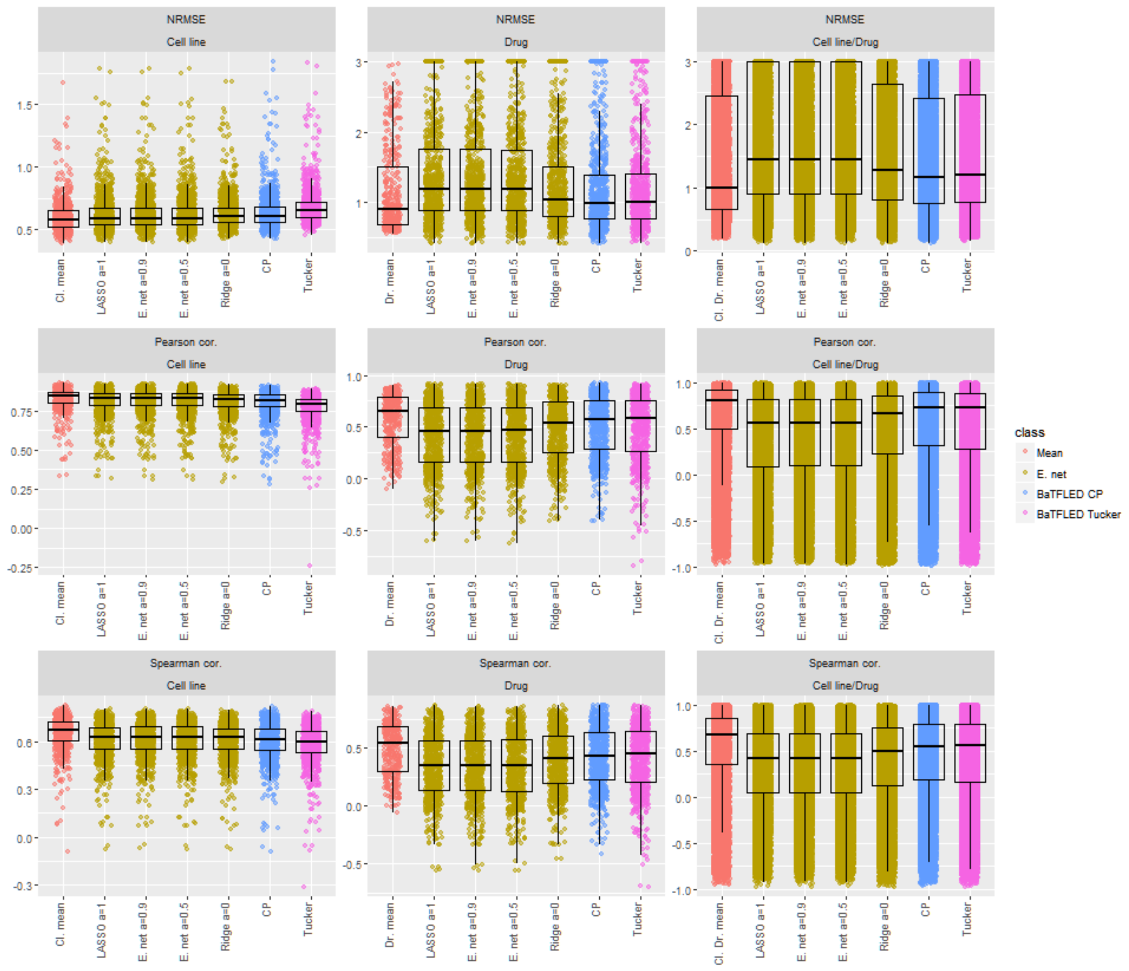


Figure 4.43: Performance measures for cross-validation runs on CTD^2 data. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

On the final test set, BaTFLED models perform similarly to other methods on the warm-start task, slightly worse on cold-start cell line prediction and better than elastic net models on cold-start drug and cell line/drug prediction (Tables 4.21, 4.22 and figures 4.44 and 4.45). As with the cross-validation results, mean prediction performs better than other methods at all cold-start tasks. Using features selected by the cross-validation runs for BaTFLED models gives worse performance at all tasks, indicating that, for this large dataset, many features are useful. In figures 4.46, 4.47 and 4.48 we show head-to-head comparisons between mean prediction, elastic net models and BaTFLED models for individual cell lines, drugs and cell line/drug combinations. In particular, the comparison between BaTFLED CP and ridge regression shows that the CP model performs better for 55.4% of cell lines, 68.6% of drugs and 62.0% of cell line/drug combinations.

Table 4.21: Normalized root mean squared error NRMSE on the final CTD^2 test data. Mean five replicates with different random starts (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.59(0.00)	0.59(0.00)	**0.60**(0.00)	**0.81**(0.00)	**0.84**(0.00)
LASSO a=1	0.55(0.00)	**0.55**(0.00)	0.62(0.00)	1.19(0.00)	1.23(0.00)
E. net a=0.9	0.55(0.00)	**0.55**(0.00)	0.62(0.00)	1.19(0.00)	1.23(0.00)
E. net a=0.5	0.55(0.00)	**0.55**(0.00)	0.62(0.00)	1.18(0.00)	1.22(0.00)
Ridge a=0	0.58(0.00)	0.58(0.00)	0.63(0.00)	1.00(0.00)	1.04(0.00)
CP	**0.54**(0.01)	**0.55**(0.00)	0.62(0.00)	0.90(0.01)	0.93(0.01)
Tucker	0.60(0.01)	0.60(0.01)	0.66(0.01)	0.90(0.06)	0.96(0.05)
CP selected	0.57(0.00)	0.57(0.00)	0.80(0.00)	1.11(0.00)	1.53(0.02)
Tucker selected	0.57(0.01)	0.57(0.01)	0.75(0.01)	1.58(0.04)	1.88(0.05)

*Training results on warm-start task.

Table 4.22: Pearson correlation on the final CTD^2 test data. Mean five replicates with different random starts (sd).

	Training*	Warm	Cell lines	Drugs	Cell lines & drugs
Mean	0.81(0.00)	0.81(0.00)	**0.80**(0.00)	**0.59**(0.00)	**0.55**(0.00)
LASSO a=1	**0.84**(0.00)	0.83(0.00)	0.79(0.00)	0.38(0.00)	0.34(0.00)
E. net a=0.9	**0.84**(0.00)	0.83(0.00)	0.79(0.00)	0.38(0.00)	0.34(0.00)
E. net a=0.5	**0.84**(0.00)	0.83(0.00)	0.79(0.00)	0.38(0.00)	0.35(0.00)
Ridge a=0	0.82(0.00)	0.82(0.00)	0.78(0.00)	0.45(0.00)	0.41(0.00)
CP	**0.84**(0.00)	**0.84**(0.00)	0.78(0.00)	0.54(0.00)	0.50(0.00)
Tucker	0.80(0.01)	0.80(0.01)	0.75(0.01)	0.54(0.04)	0.49(0.02)
CP selected	0.82(0.00)	0.82(0.00)	0.66(0.00)	0.48(0.00)	0.33(0.00)
Tucker selected	0.82(0.00)	0.82(0.00)	0.70(0.00)	0.34(0.01)	0.27(0.01)

*Training results on warm-start task.

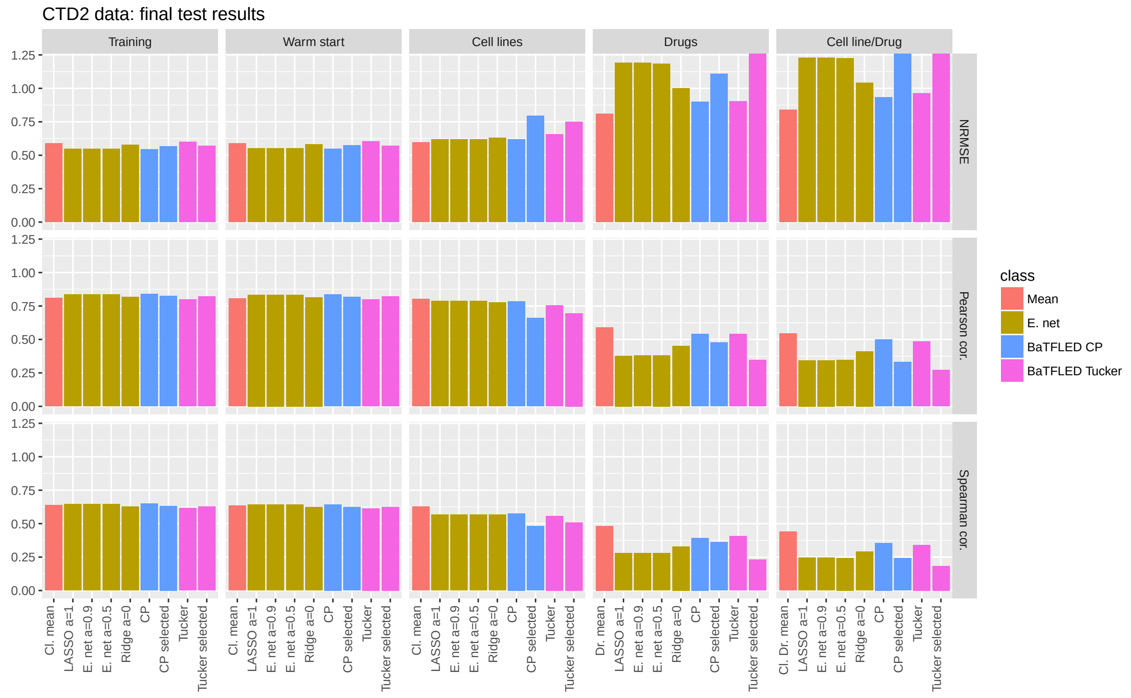


Figure 4.44: Performance measures for final test runs on CTD^2 data (average across 5 replicates with different random starts). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown. Note scales differ for each response measure and NRMSE values above 1.5 have been set to 1.5 for visualization purposes.

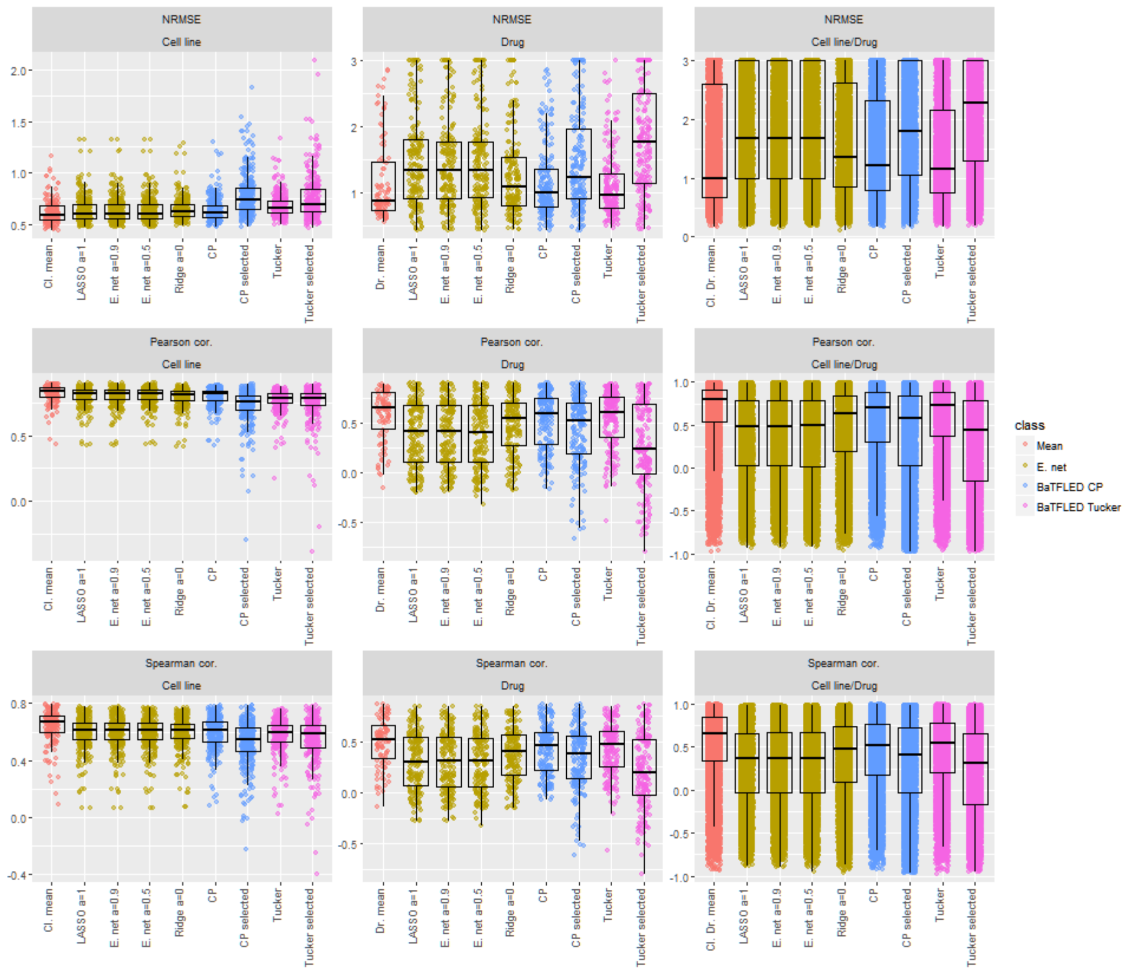


Figure 4.45: Performance measures for final test runs on CTD^2 data from one randomly selected replicate. Points and boxplots show the normalized root mean squared error (NRMSE), Pearson correlation and Spearman correlation for individual cell lines, drugs and cell line/drug combinations for cold start prediction tasks. Note: scales differ for each plot and NRMSE values above 3 have been set to 3 for visualization purposes.

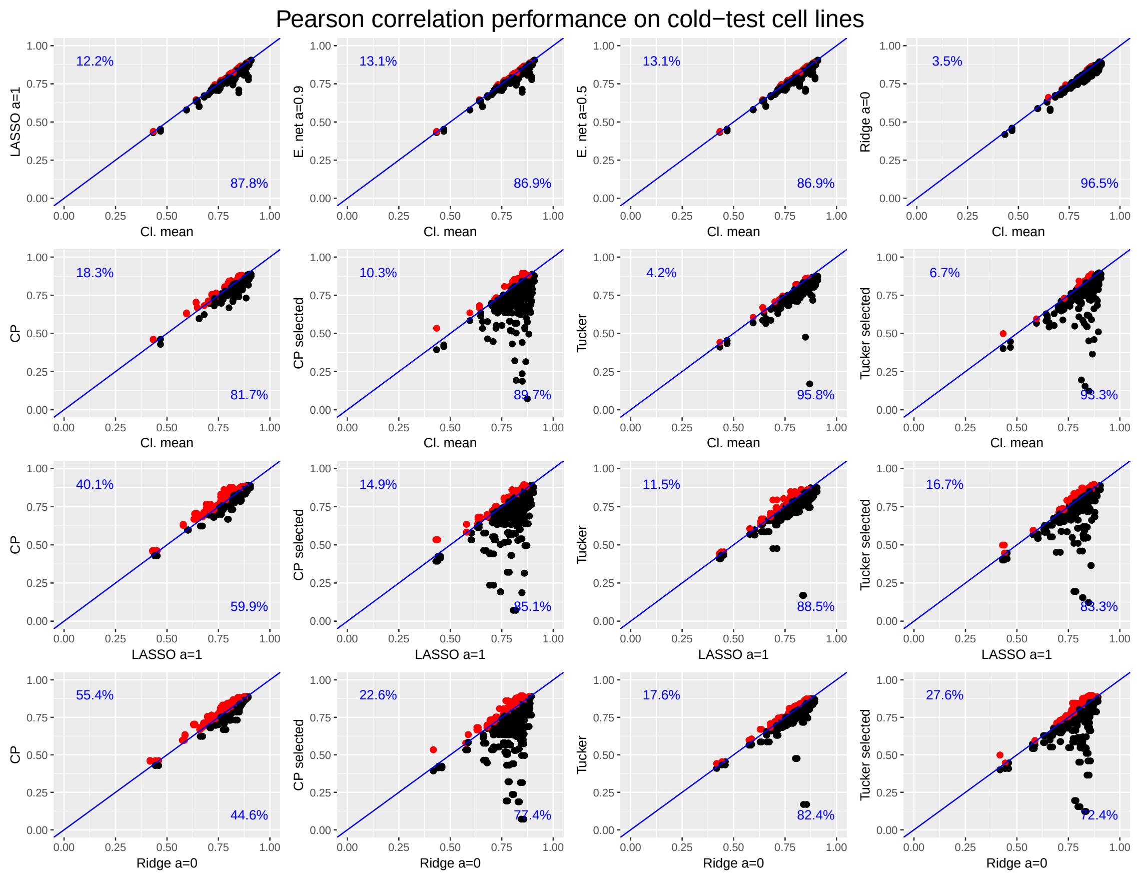


Figure 4.46: Comparison between mean prediction and other methods on the cold-start cell line task for the final CTD^2 test data. Each point represents one cell line with x and y coordinates showing the Pearson correlation with the true responses for the two different methods. Red points are predicted better by the method on the y-axis and the percentage of cell lines on either side of the diagonal are annotated.

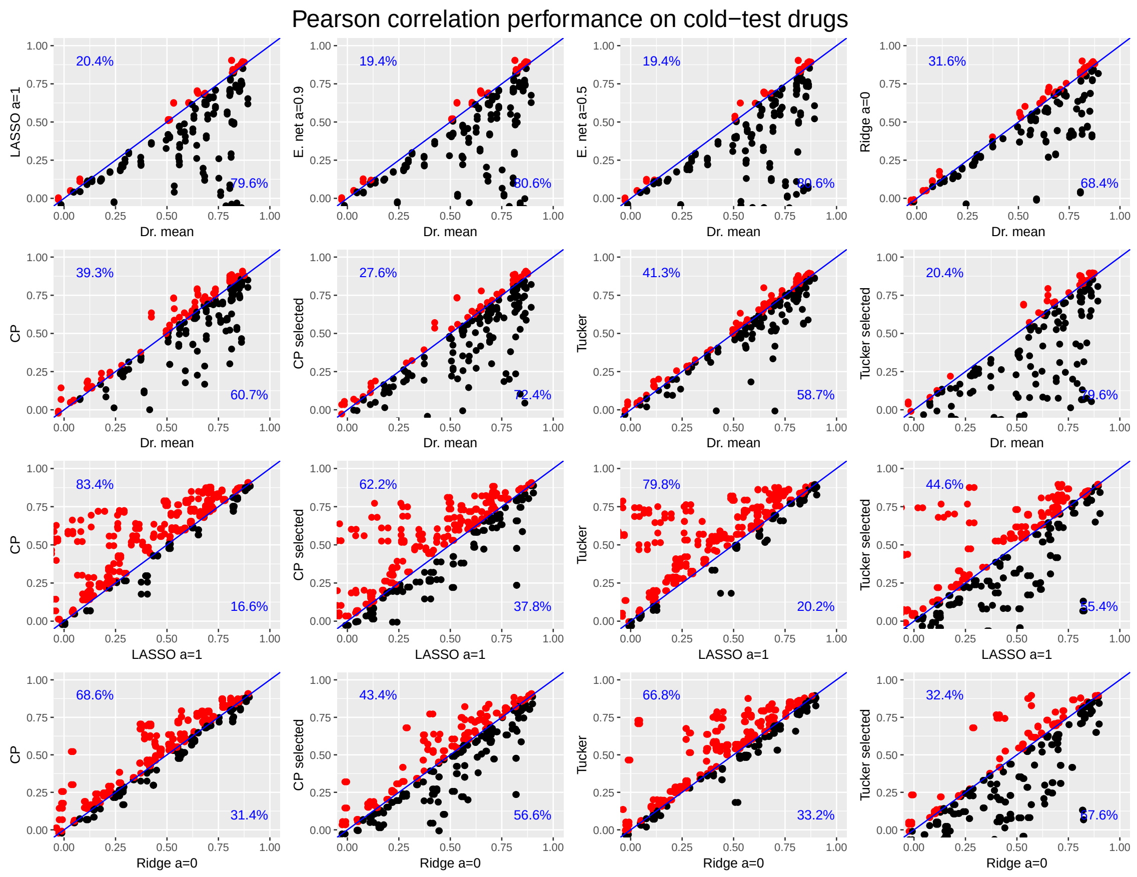


Figure 4.47: Comparison between mean prediction and other methods on the cold-start drug task for the final CTD^2 test data. Each point represents one cell line with x and y coordinates showing the Pearson correlation with the true responses for the two different methods. Red points are predicted better by the method on the y-axis and the percentage of cell lines on either side of the diagonal are annotated.

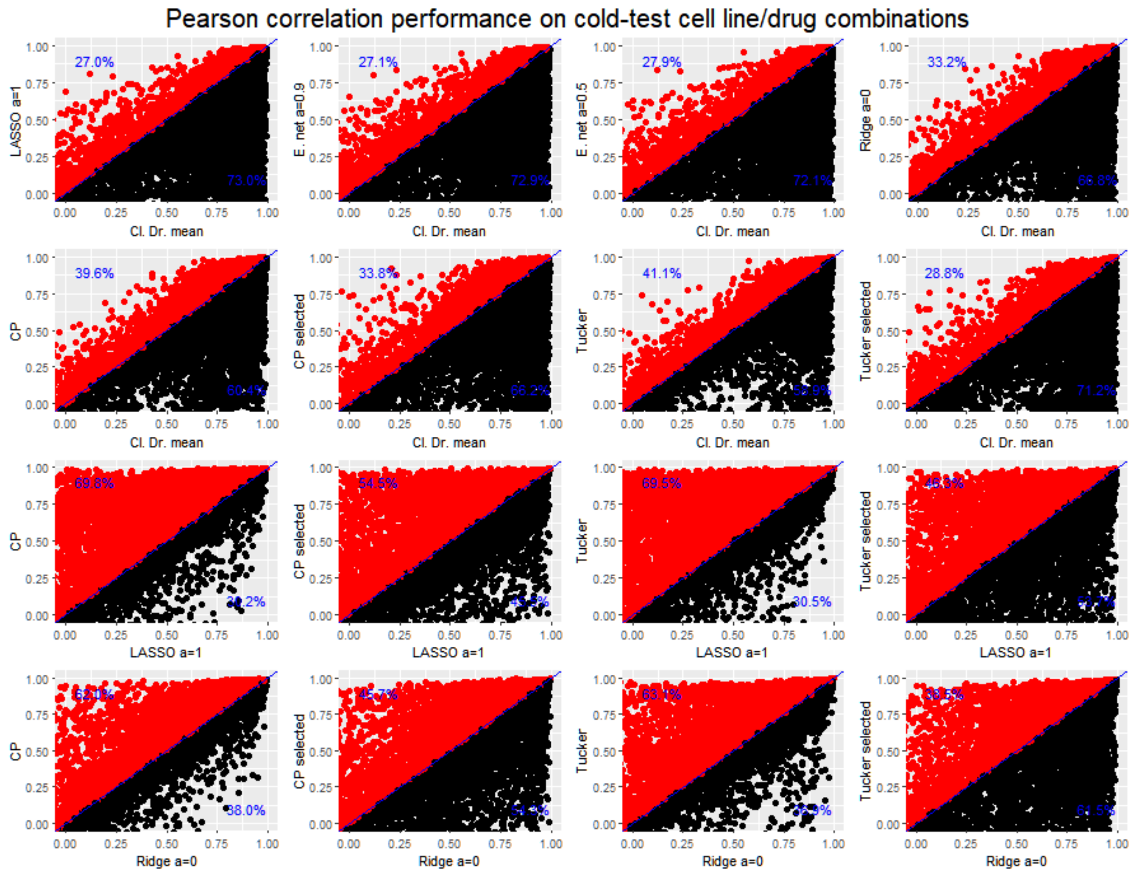


Figure 4.48: Comparison between mean prediction and other methods on the cold-start cell line/drug combination task for the final CTD^2 test data. Each point represents one cell line with x and y coordinates showing the Pearson correlation with the true responses for the two different methods. Red points are predicted better by the method on the y-axis and the percentage of cell lines on either side of the diagonal are annotated. Note: mean predictions are across cell lines as these performed better than taking the mean across drugs.

4.5.6 Selected features

For the CTD^2 data, we examine which features were selected by LASSO, elastic net with $\alpha = 0.5$, ridge regression, BaTFLED CP and BaTFLED Tucker models. We consider the features selected during cold-start prediction runs for each task, i.e. we look at the cell line features for runs trained with all drugs predicting for the final test cell lines. Table 4.23 shows counts of the total number of features in the top 15% of features in at least one replicate run. Elastic net models are very consistent regardless of the random initialization, so the features tend to be the same across replicate runs and the total percentages are around 15%. With the run parameters used here, BaTFLED models are much more sensitive to the initialization and hence have a greater proportion of features selected across runs. The number and percentage of each type of feature selected by each method are shown in figures 4.49 and 4.50. For both the cell line and drug data, all methods choose about equally from the different data types. Figure 4.51 shows which features were chosen by each method and the consistency among elastic net methods is apparent.

We find that 70 cell line features and 44 drug features were selected by all methods. For cell lines, this set includes 4 annotation features (indicators for growth media RPMI003, malignant melanoma, oesophagus and upper aerodigestive tract), 18 copy number features, 33 expression features and 15 mutation features. The selection of the growth media indicator, which was used for 49 cell lines, suggests that using this media may have an effect of drug response. For drugs, we find 2 target features (ALK and PLK1), 10 fingerprint features and 32 PaDEL features 13 of which combine correlated features.

We also looked at which features were selected in the top 15% by all replicate BaTFLED CP and Tucker runs. For the cell line prediction task, all CP runs chose 5 features including an indicator of the RPMI001 media (408 cell lines) one copy number feature (for gene RNF43) and three gene expression features (BCL11A, IKZF1 and P2RY8). In the Tucker run, no cell line features other than the constant term were selected by all replicates. Looking at the drug features, the CP runs consistently chose 31 features and the Tucker runs only 2. The 31 features chosen by CP runs include 5 gene targets (CDK2, MAP2K1, MAP2K2, PLK1 and SRC), 4 fingerprint indicators and 22 PaDEL descriptors. The

Tucker runs only consistently choose two PaDEL descriptors (C1SP2: "doubly bound carbon bound to one other carbon" and a combination feature of 16 correlated features).

Table 4.23: Number of features selected in the union of the top 15% across replicates for CTD^2 data (percentage).

	LASSO	Neural net	Random forest	CP	Tucker
Cell lines	224 (15.4)	223 (15.3)	219 (15.0)	722 (49.6)	783 (53.8)
Drugs	126 (15.2)	129 (15.5)	125 (15.0)	306 (36.8)	421 (50.7)

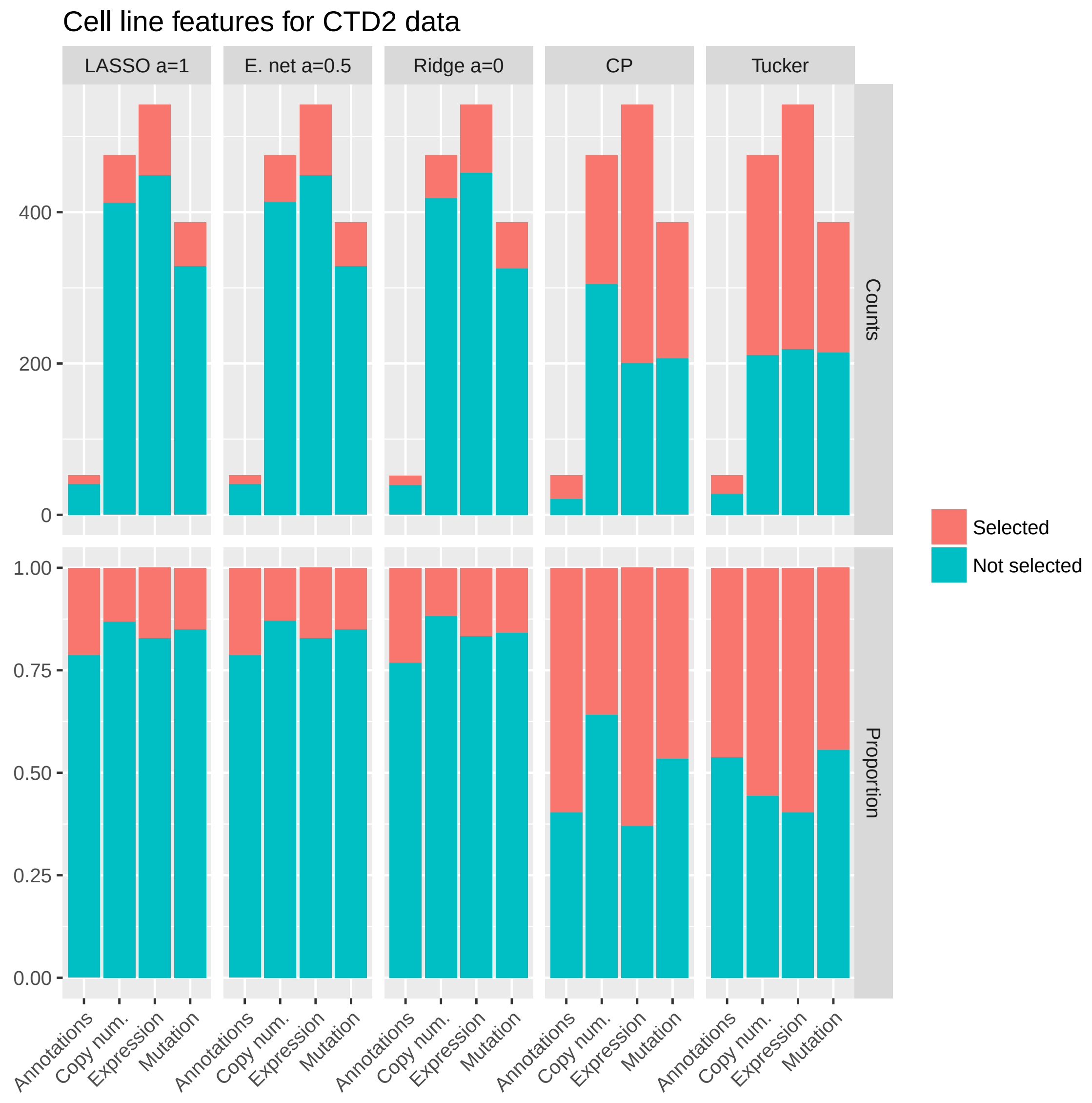


Figure 4.49: Top row: Number of cell line features selected by different methods for the CTD^2 data. Bar height shows the total number of features of the given type, red portion shows the number of these features in the union of the top 15% of features across replicates. Bottom row: Proportion of total features from each category that are chosen.

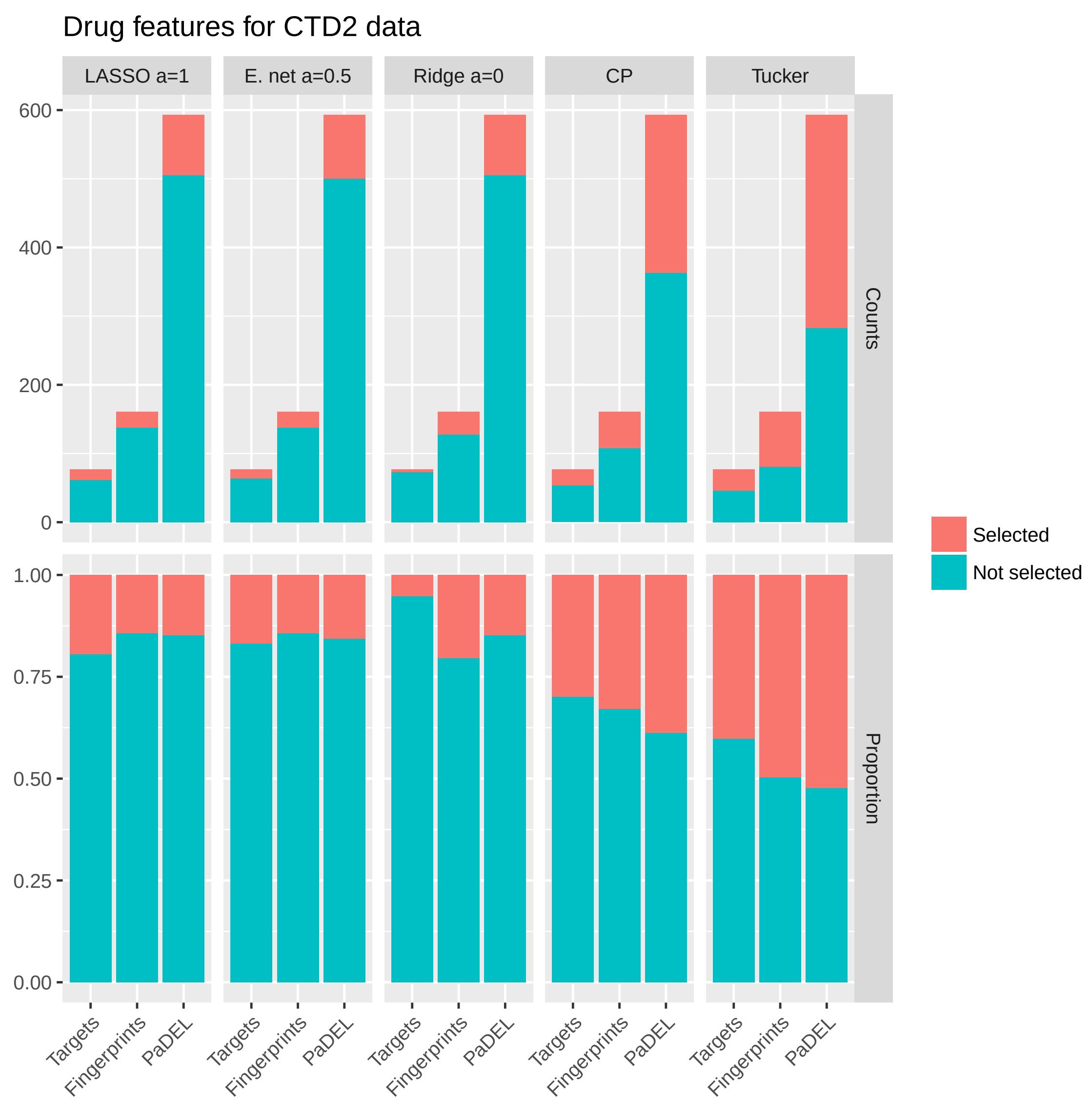


Figure 4.50: Top row: Number of drug features selected by different methods for the CTD^2 data. Bar height shows the total number of features of the given type, blue portion shows the number of these features in the union of the top 15% of features across replicates. Bottom row: Proportion of total features from each category that are chosen

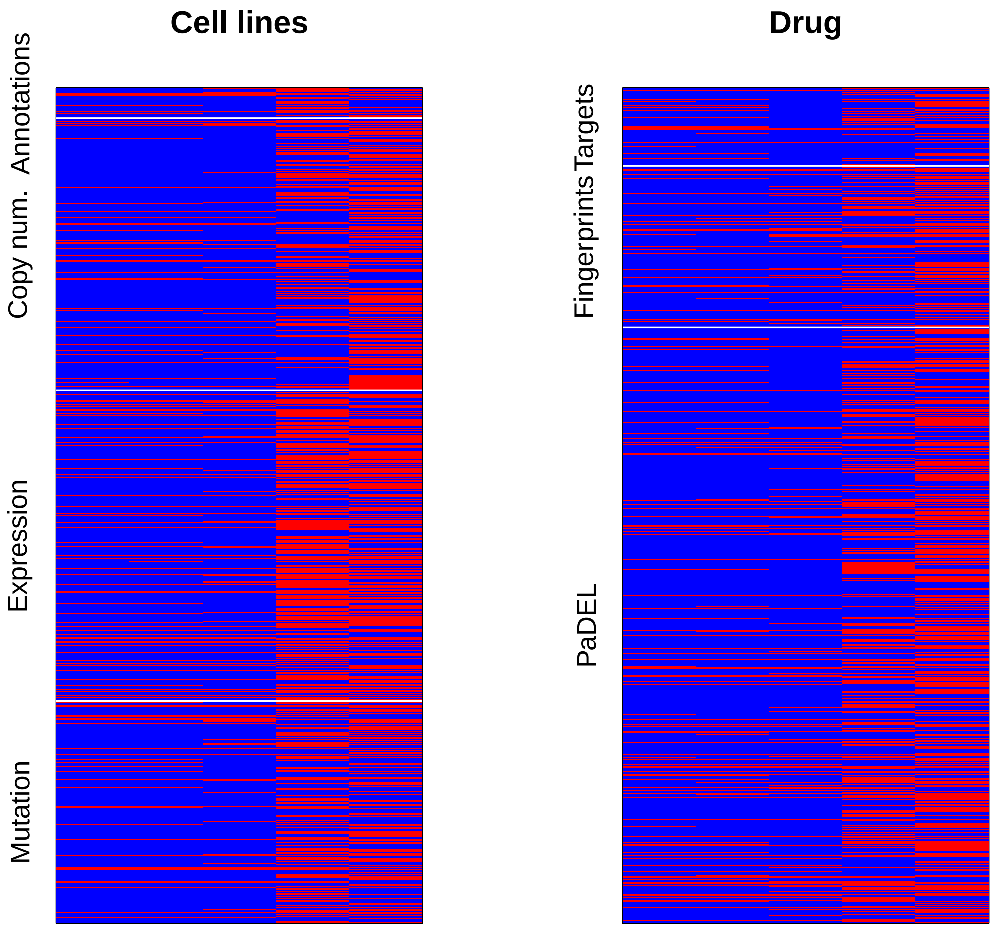


Figure 4.51: Features selected by different methods for the CTD^2 data. Each row is a feature, features selected in the top 15% of features for any replicate are colored red. Note that different methods tend to choose different features and that there are more features chosen by BaTFLED methods due to a greater variance across runs.

4.6 Themes across datasets

In this section, we discuss the results between the different datasets and elucidate trends and differences between them. While each dataset is processed differently and contains different cell lines and drugs, some patterns can be seen which may be informative both of the datasets and the algorithms tested.

While the models that were tested were fairly consistent, the three datasets were normalized differently and different features were used. For the DREAM data, we trained for all of the testing scenarios simultaneously during cross-validation and only performed final cold-start testing for cell lines while for the Heiser and CTD^2 datasets, we split test sets separately for each task and had final test sets for each task. The DREAM data responses were normalized according to the formula 4.3 from Costello et al. [24], the Heiser responses were normalized to account for growth rate according to equation 1.1 from Hafner et al. [22] and for the CTD^2 data we downloaded normalized responses according to [114]. For the DREAM dataset we used subtype, expression and mutation cell line features, for the Heiser data we used only subtype and expression features and for the CTD^2 data we used annotations, copy number, expression and mutation data. For all datasets we used target/class indicators, PubChem fingerprints and PaDEL descriptors for the drug data, but there were differences in which targets were annotated, which features were removed or combined do to information content and correlation structure. Additionally, we tested using kernel functions on the Heiser data which allowed us to include cell line features describing copy number, methylation and protein abundance. Also for the Heiser dataset, we tested predicting GR_{50} both directly and calculated from predicted curves. Table 4.24 recaps the size of each of the datasets and what features were used.

Pulling together results across these datasets, some general trends are visible (table 4.25). BaTFLED models are among the top performers for all datasets at warm-start prediction. For the cold-start cell line task, BaTFLED models perform well for DREAM and Heiser datasets. For cold-start drug and cell line/drug combination prediction, BaTFLED models do well on the Heiser data when using kernelized features and better than any method that uses the features directly. Our methods are also among the best performers

Table 4.24: Summary of data sets

	Responses	Test C.L.	Test Dr.	C.L. features	C.L. types	Dr. features	Dr. types
DREAM	$52 \times 26 \times 9$	17	0	733	annot. (6) expression (177) mutation (550)	433	class (6) fingerprint (139) PaDEL (285)
Heiser	$48 \times 102 \times 9$	12	26	589	annot. (6) expression (583)	726	annot. (27) fingerprint (271) PaDEL (428)
Heiser (kernelized)	$51 \times 102 \times 9$	13	26	38	annot. (6) expression (583) copy num. (381) RPPA (33) methylation (1,110)	76	annot. (27) fingerprint (271) PaDEL (428)
CTD^2	$783 \times 493 \times 16$	156	98	1,456	anno. (52) expression (542) mutation (387) copy num. (475)	831	anno. (77) fingerprint (161) PaDEL (593)

for the CTD^2 data on these tasks, but do not perform as well as simple mean prediction. Looking at the Pearson correlation values we see that these fall in a similar range $(0.95-0.99)$ for the training data with the exception of the CTD^2 dataset. This dataset is much larger and heterogeneous (pan-cancer), so it may be less prone to over fitting. The warm start correlation values are also fairly consistent ranging from 0.84 to 0.89 across all datasets. Larger differences are apparent on the cold-test tasks where, for cell lines, the methods we tested seem to have a harder time predicting the DREAM responses. This may be due to the treatment of the cell line features (we set some missing data to zero for these features) or because the normalization did not take into account the growth rate and there were not many training cell lines to learn from. On cold start drug tasks, the CTD^2 data has lower correlations than the Heiser data, perhaps due to a greater diversity of drugs or a feature set with more PaDEL descriptors and target/class annotations. The results for cold-start cell line/drug combination prediction are similar to those for drug prediction and only slightly worse, which is perhaps surprising considering that we are now predicting based entirely on the feature data of both the cell lines and drugs.

Finally, we consider the run time of each of these algorithms when run in parallel on 5 2.7GHz Intel®Xeon®processors. The elastic net algorithms are the fastest, taking around 12 minutes on the smaller DREAM and Heiser datasets and about $20-22$ hours on the CTD^2 data. Ridge regression runs are significantly quicker than the other elastic

Table 4.25: Summary of best performing algorithms on final test data (Pearson correlation).

		Train	Warm	Test C.L.	Test Dr.	Test C.L/Dr.
DREAM	1st	**CP** re (0.99)	**CP** (0.88)	**CP** (0.62)		
	2nd	NN 3L (0.97)	NN 3L. (0.84)	**CP** re (0.62)		
	3rd	**Tucker** (0.96)	**CP** re (0.83)	**Tucker** (0.61)		
Heiser	1st	NN 2L. (0.98)	RF 1000x10 (0.89)	**Tucker** sel. (0.88)	RF 1000x10 (0.66)	RF 1000x10 (0.64)
	2nd	NN 3L. (0.98)	**Tucker** sel. (0.89)	Mean (0.88)	Mean (0.65)	RF 1000x5 (0.64)
	3rd	NN 1L. (0.95)	**Tucker** (0.88)	**Tucker** (0.88)	RF 1000x5 (0.65)	NN 2L (0.64)
Heiser (kernels)	1st	NN 1L (0.97)	RF 1000x10 (0.88)	Mean (0.88)	**CP** (0.71)	**CP** (0.70)
	2nd	NN 2L (0.96)	CL Mean (0.88)	LASSO (0.87)	Ridge (0.70)	Ridge (0.69)
	3rd	NN 3L (0.95)	LASSO (0.88)	E.net a=.9 (0.87)	**Tucker** (0.69)	**Tucker** (0.68)
CTD^2_*	1st	**CP** (0.84)	**CP** (0.84)	Mean (0.80)	Mean (0.59)	Mean (0.55)
	2nd	LASSO (0.84)	LASSO (0.84)	LASSO (0.79)	**Tucker** (0.54)	**CP** (0.50)
	3rd	E.net a=.9 (0.84)	E.net a=.9 (0.84)	E.net a=.5 (0.79)	**CP** (0.54)	**Tucker** (0.49)

re: second round training, sel: features selected during cross-validation rounds.
* Neural net and random forest models were not run on the CTD^2 data.

net models, taking about half of that time. Run time for random forest models depends on the number of trees and their depth. Smaller models ($5,000$ trees at depth 5) take around 15 minutes for the Heiser data, while larger models ($1,000$ trees at depth 10) take about 2.5 hours on the Heiser data. Run time for neural net models also varies with their size. On the Heiser data, the one layer neural net models take about $35 - 40$ hours, two layer nets about 80 hours and three layer nets about 48 hours. For BaTFLED, with the current implementation, CP models can only be run in serial. On Heiser data with the parameters tested here, they take around 7 hours and on the CTD^2 data take about 6 days. Some of the updates in the Tucker models can be run in parallel and so can be faster given that the resources are available. Using 5 cores, Heiser models take around 15 hours and CTD^2 models take around 4.5 days.

Chapter 5

BaTFLED performance on micro-environment microarray data

In this chapter we show results for BaTFLED and comparison methods when predicting the cell line growth in different environmental settings. There is strong evidence that the specific cellular milieu in which a tumor develops has a large impact on how these cells behave and this may help inform treatment in patients. A recently developed technology dubbed 'micro-environment microarrays' (MEMA) provides one avenue for interrogating these relationships [116]. Figure 5.1 shows an overview of this technology. Arrays are printed with spots containing combinations of extra-cellular matrix proteins and ligands, cells are added, grown and imaged at high resolution using different staining sets. For this analysis, we look at cell counts extracted from these images.

5.1 Response data

The current MEMA dataset contains response data for four cell lines, 47 ligands and 41 extra-cellular matrix proteins (ECMPs). Response data were downloaded from the hosting platform Synapse (`www.synapse.org` projects: syn9838910 and syn9838911). We focused on $\log_2$ counts of cell nuclei segmented using CellProfiler [117], summarized across replicates and normalized using a reduction of unwanted variance (RUV) and LOESS smoothing approach (level 4 data). We used cell counts for each of three staining sets with response data for ligands and ECMPs measured in at least 3/4 cell lines (47 ligands and 41 ECMPs). In the sections below, we remove 2 ligands and 3 ECMPs because we were

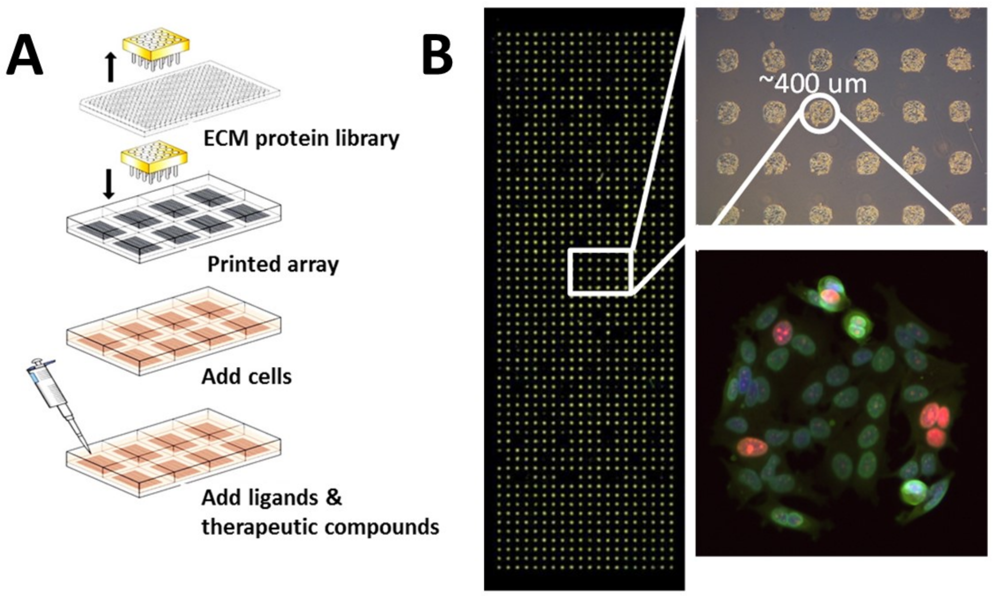


Figure 5.1: An overview and examples of MEMA production and usage. A Protein libraries are printed into well plates upon which cells are cultured. Ligands are then added to each well and cells are allowed to grow for 48-72h. B Examples of low resolution scans, phase contrast, and a single high content image from a MEMA. Cells are stained with DAPI, cell mask and EdU for assessment of proliferation in the example. `http://www.ohsu.edu/xd/education/schools/school-of-medicine/departments/basic-science-departments/biomedical-engineering/bme-labs/korkola-lab/research/microenvironment-and-cancer.cfm`

unable to obtain features for them so the final response dataset is $4 \times 45 \times 38$. Normalized $\log_2$ cell counts for one of the three staining sets are shown in figures 5.2, 5.3 and 5.4.

As this technology is still somewhat in the development phase, one of the cell lines (MCF10A) was grown with a higher density of cells than the other two cell lines. To make values comparable between cell lines, we simplified the responses to discrete growth levels using the following steps. First, we z-scaled responses for each cell line and staining set and quantized responses into three three types of growth. We set the top and bottom 10% of responses to -1 and 1 respectively and the center 80% to zero for each cell line and staining set. We then took the mean across the three staining sets so the response values have the discrete values $(\pm 1, 2/3, 1/2, 1/3$ or $0)$. Figure 5.5 shows heatmaps of the resulting values.

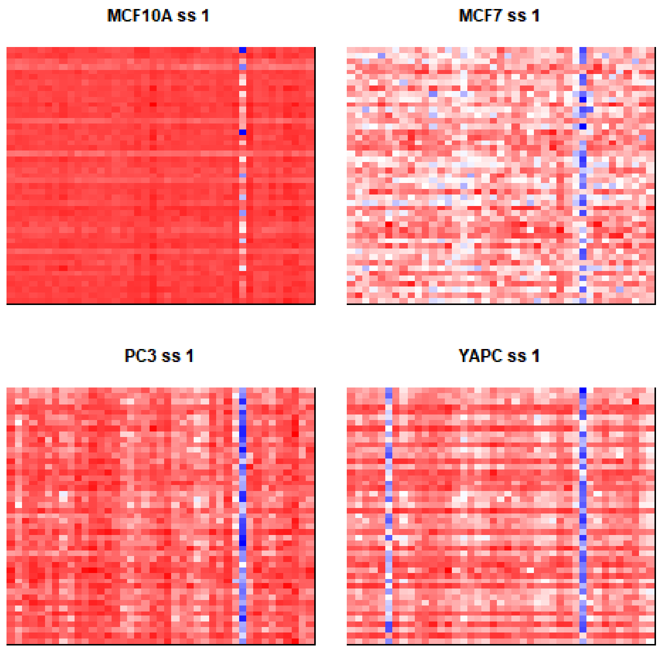


Figure 5.2: Heatmap showing the $\log_2$ cell count response values from each cell line after RUV-Loess normalization before scaling and discritization for the first staining set. Each row is a ligand and each column an extra cellular matrix protein. Red is high (with a maximum value of 9.5) and blue low (minimum value of -2.1).

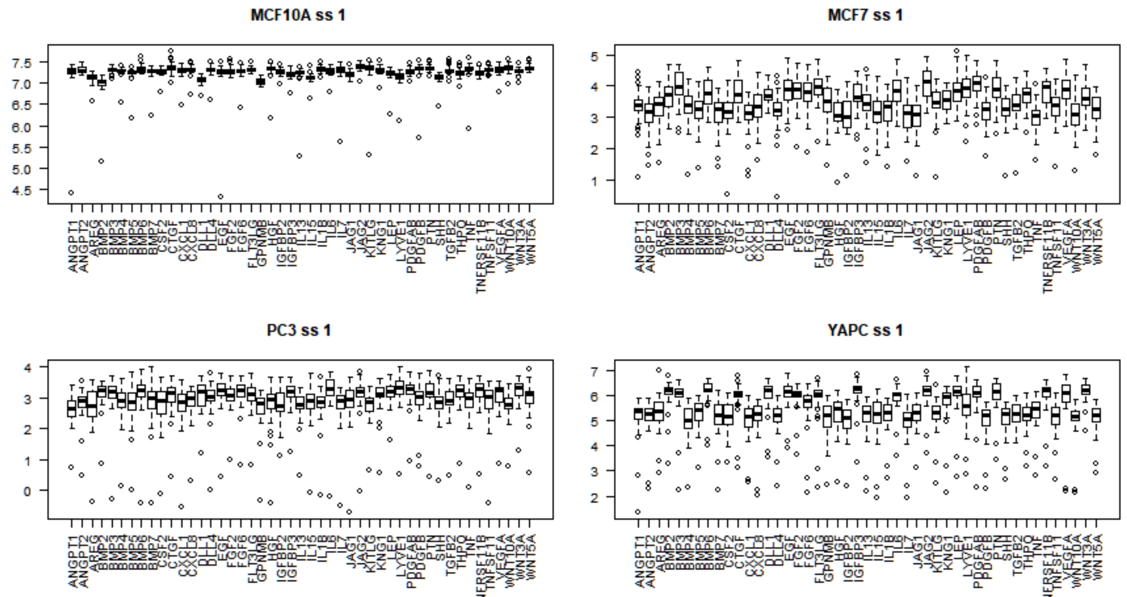


Figure 5.3: Boxplots of the $\log_2$ cell count response values from each cell line after RUV-Loess normalization before scaling and discritization for the first staining set. Separate boxes are shown for each ligand ranging across extra-cellular matrix proteins.

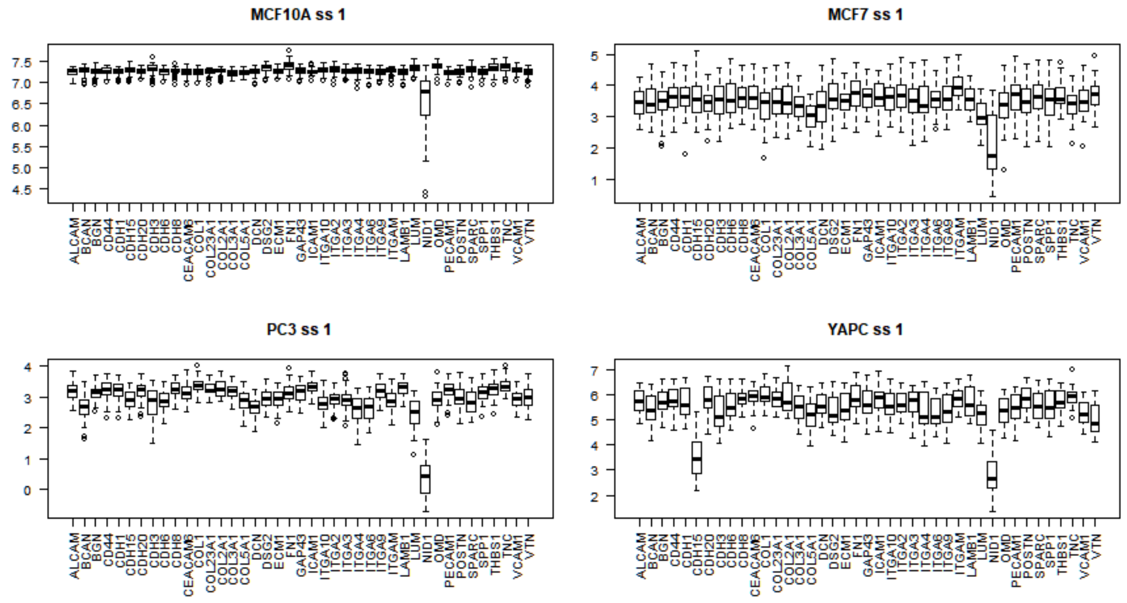


Figure 5.4: Boxplots of the $\log_2$ cell count response values from each cell line after RUV-Loess normalization before scaling and discritization for the first staining set. Separate boxes are shown for each extra-cellular matrix protein ranging across the different ligands.

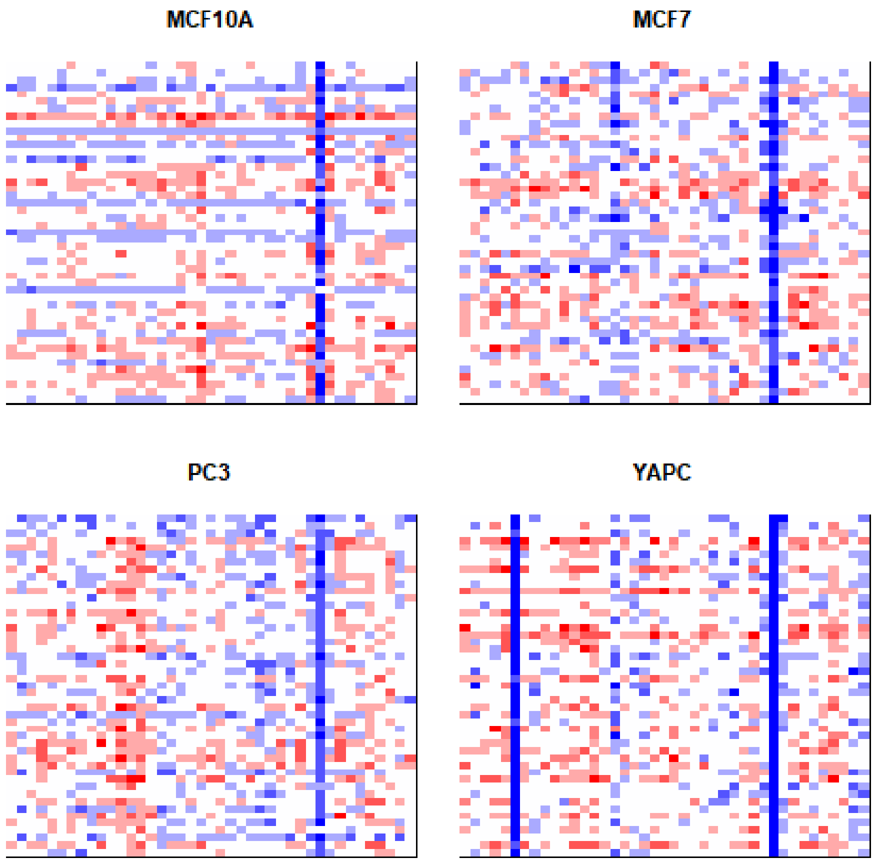


Figure 5.5: Heatmap showing the discritized $\log_2$ cell count response values from each cell line. Each row is a ligand and each column an extra cellular matrix protein. Red is high (with a maximum value of 1) and blue low (minimum value of -1)

5.2 Input feature data

We used gene expression data from the L1000 platform [118] as input features for the cell lines. Level 3 summary data were downloaded from the gene expression omnibus (GEO accession number GSE70138 from LINCs study LDG-1227). We obtained expression values for 821 genes from two breast cancer lines (MCF10A and MCF7) one prostate cancer line (PC3) and one pancreatic cancer line (YAPC). These values were then z-scaled across cell lines and converted to a 4×4 similarity kernel matrix using a Gaussian kernel with width equal to the mean Eucledian distance between cell lines (figure 5.6).

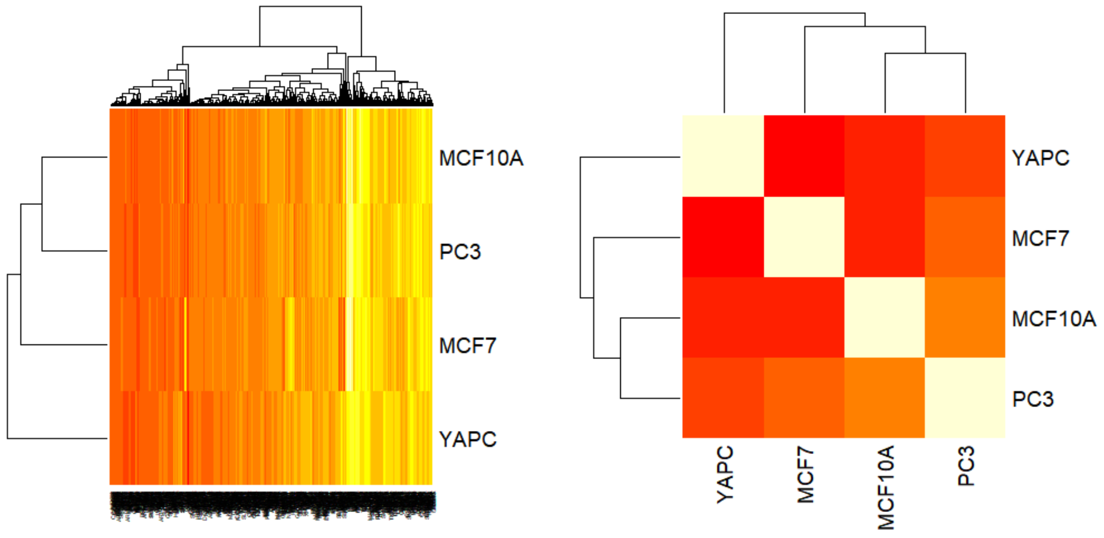


Figure 5.6: Heatmap of L1000 gene expression data and kernel feature matrix for the cell line data. Hotter colors represent higher values (higher expression or more similar cell lines). Dendrograms produced using heirachical clustering.

For ligand features, we used a gene-gene interaction network from InnateDB [119] downloaded with Cytoscape [120]. The idea here is that ligands with similar interactions in the network may have similar effects on cellular growth, so information in the network may be used to predict the growth of cells for new ligands. To this end, we coded the the network as a binary adjacency matrix A with rows and columns for each of 849 genes that were connected to one or more of the ligand genes. A one in this symmetric matrix means the genes corresponding to that row and column have an edge in the network. We then multiplied this matrix by itself to obtain power matrices A^2, A^3, ..., A^6 and binarized the results. The matrix A^n encodes whether two genes are connected by a path of with n edges. We combined these matrices by taking their weighted sum with weights corresponding to the geometric series 1, 1/2, 1/4, 1/8, 1/16, 1/32. This way, two genes that are adjacent have the strongest weights, but information is preserved on which genes are connected out to paths of length 6. We then restricted the rows to only those ligands in used in the MEMA studies and two ligands without any interactions in the network were removed (WNT10A and PDGFAB). Figure 5.7 shows this 45×849 matrix after applying heirachical clustering to the rows and columns. Finally we again transformed this matrix to a kernel matrix using a Gaussian kernel with width equal to the mean Euclidean distance between ligands.

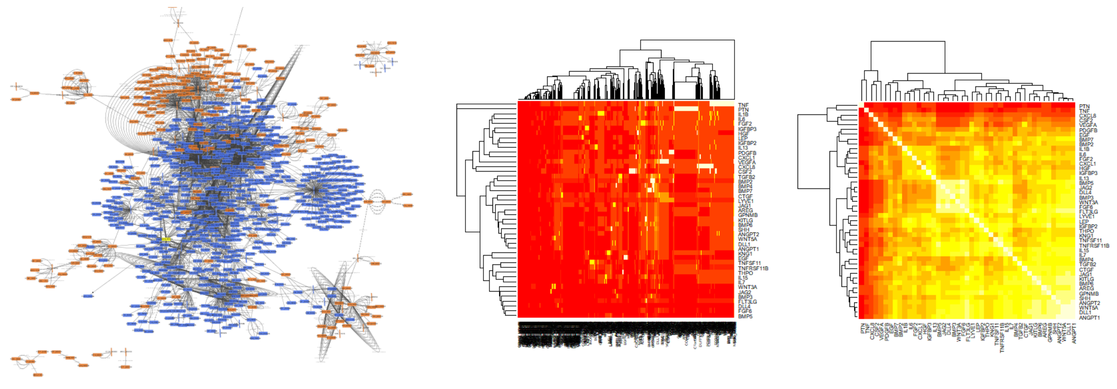


Figure 5.7: InnateDB network, feature matrix and kernel matrix for the ligand data. Center plot has rows for each of the 45 ligands used in the MEMA study and columns for all 849 genes connected to ligands in the network. Right plot has rows and columns for the 45 ligands. Hotter colors represent higher values (stronger connections between genes or more similarly connected ligands). Dendrograms produced using heirachical clustering.

We treated the extra-cellular matrix proteins (ECMPs) in the same way as the ligands, except that we started with a gene interaction network from the Agile Protein Interactions Dataserver APID [121] as this had information for more of the ECMPs. Three ECMPs (CDH20, COL1 and OMD) were not included in the network and were removed from the study. Figure **??** shows the feature matrix before applying the Gaussian kernel.

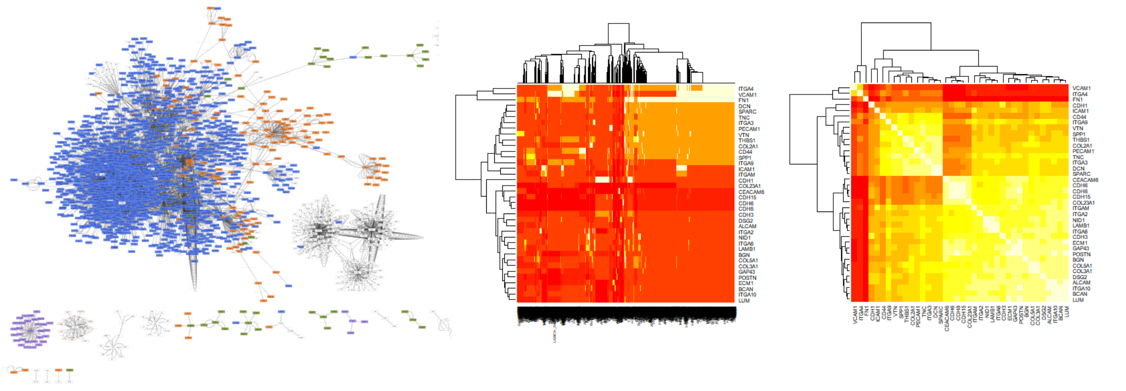


Figure 5.8: APID network, feature matrix and kernel matrix for the ECMP data. Center plot has rows for each of the 38 ECMPs used in the MEMA study and columns for all 1,593 genes connected to ECMPs in the network. Right plot has rows and columns for the 38 ligands. Hotter colors represent higher values (stronger connections between genes or more similarly connected ECMPs). Dendrograms produced using heirachical clustering.

5.3 Results

For this dataset we compared to the same methods as with other datasets (elastic net, random forests and neural networks) with the exception that we only ran ridge regression from the elastic net family. Since we used kernel matrices as input features, feature selection was unlikely to improve performance. We used the same parameters as with the Heiser data with the exception that we did not encourage sparsity in the BaTFLED models, CP models had a latent dimension of 100 and Tucker models have a core of $4 \times 10 \times 10$. When running comparison methods with the dose-response datasets, we ran separate models to predict at each dose, but in this case we concatenate features from all three modes and run a single model to make predictions.

We did not do any parameter tuning for this task and we present results for cross-validation runs leaving out about 20% of responses for each mode in each fold (1 of 4 cell lines, 9 of 45 ligands and 8 of 38 ECMPs). For combined predictions tasks (e.g. predicting for a left-out cell line and left-out ligand simultaneously) we ran models for all possible combinations of CV-folds. This means we ran 4 folds for left-out cell lines, 5 folds for left-out ligands and ECMPs, 20 folds for left-out cell line/ligand and cell line/ECMPs, 25 folds for left-out ligand/ECMPs and 100 folds for left-out cell line/ligand/ECMP runs. Kernel features for test cell lines were removed from the input matrices for each CV fold.

The results shown in tables 5.1 and 5.2 and figure 5.9 indicate that the BaTFLED Tucker model has an advantage over other models when predicting warm-start responses and cold-start cell line responses. BaTFLED performs about as well as other methods at the other tasks and none of these outperform mean prediction. This indicates that the network-based features used for ligands and ECMPs were not very informative of cell growth. Also BaTFLED models have some positive correlation for the harder tasks where most random forest and neural net models do not. Additionally, for this dataset we differentiate the performance of mean prediction on training and warm data depending on whether we predict the mean response across cell lines, ligands or ECMPs. In the dose-response data, the best performing mean was always the mean of the training cell lines, but for this data, the mean across ligands performs best on the warm start data in terms of RMSE and Pearson correlation. For the cold-start tasks, the choice of mean is determined by the task and we use the most informative mean possible for each task.

Table 5.1: Mean normalized RMSE on MEMA dataset cross-validation runs (sd).

	Training	Warm	Cell lines	Ligands	ECMPs	Cell lines & Ligands	Cell lines & ECMPs	Ligands & ECMPs	All
Cl. mean	**0.69**(0.00)	0.99(0.02)	-	-	-	-	-	-	-
Lig. mean	0.79(0.00)	**0.80**(0.02)	-	-	-	-	-	-	-
ECMP mean	0.88(0.00)	0.91(0.01)	-	-	-	-	-	-	-
Mean	-	-	0.96(0.10)	**0.81**(0.00)	**0.90**(0.04)	**0.93**(0.05)	**0.99**(0.05)	**1.02**(0.02)	**1.03**(0.04)
Ridge a=0	0.88(0.00)	0.92(0.01)	0.92(0.03)	0.96(0.04)	0.99(0.07)	0.98(0.05)	1.01(0.08)	1.07(0.07)	1.08(0.10)
N.N. 1L	0.82(0.04)	0.99(0.04)	0.96(0.07)	0.99(0.02)	1.03(0.08)	0.99(0.09)	1.11(0.15)	1.16(0.08)	1.18(0.17)
N.N. 2L	0.76(0.03)	0.90(0.02)	0.90(0.05)	0.95(0.02)	0.95(0.05)	0.97(0.06)	1.01(0.06)	1.07(0.05)	1.08(0.09)
N.N. 3L	0.82(0.02)	0.92(0.03)	0.94(0.07)	0.95(0.02)	1.00(0.08)	0.97(0.06)	1.02(0.07)	1.09(0.06)	1.10(0.11)
R.F. 1000x5	0.80(0.01)	0.85(0.03)	0.90(0.05)	0.90(0.01)	0.97(0.04)	0.95(0.04)	1.01(0.06)	1.05(0.05)	1.07(0.08)
R.F. 5000x5	0.80(0.01)	0.84(0.02)	0.89(0.04)	0.90(0.02)	0.96(0.04)	0.95(0.04)	1.00(0.06)	1.04(0.03)	1.06(0.07)
R.F. 1000x10	0.70(0.00)	0.82(0.02)	0.90(0.06)	0.89(0.01)	0.94(0.06)	0.95(0.04)	1.01(0.07)	1.07(0.04)	1.08(0.07)
CP	0.81(0.00)	0.86(0.02)	0.91(0.03)	0.93(0.04)	0.99(0.07)	0.96(0.05)	1.02(0.08)	1.08(0.06)	1.07(0.09)
Tucker	0.80(0.01)	**0.80**(0.01)	**0.88**(0.05)	0.91(0.04)	1.01(0.07)	0.95(0.05)	1.03(0.09)	1.09(0.07)	1.09(0.10)

Table 5.2: Mean Pearson correlation on MEMA dataset cross-validation runs (sd).

	Training	Warm	Cell lines	Ligands	ECMPs	Cell lines & Ligands	Cell lines & ECMPs	Ligands & ECMPs	All
Cl. mean	**0.73**(0.00)	0.41(0.02)	-	-	-	-	-	-	-
Lig. mean	0.62(0.01)	0.60(0.02)	-	-	-	-	-	-	-
ECMP mean	0.48(0.00)	0.41(0.02)	-	-	-	-	-	-	-
Mean	-	-	0.43(0.08)	**0.60**(0.01)	**0.46**(0.08)	**0.43**(0.08)	**0.27**(0.08)	-0.09(0.08)	NA(NA)
Ridge a=0	0.51(0.01)	0.40(0.01)	0.41(0.07)	0.31(0.09)	0.27(0.14)	0.28(0.11)	0.23(0.15)	0.04(0.13)	0.06(0.19)
N.N. 1L	0.59(0.06)	0.36(0.08)	0.38(0.11)	0.32(0.07)	0.34(0.09)	0.34(0.11)	0.22(0.11)	0.00(0.11)	-0.04(0.22)
N.N. 2L	0.65(0.03)	0.45(0.04)	0.46(0.07)	0.38(0.05)	0.37(0.09)	0.36(0.08)	0.23(0.10)	**0.05**(0.11)	**0.07**(0.16)
N.N. 3L	0.58(0.04)	0.43(0.04)	0.38(0.14)	0.38(0.03)	0.30(0.09)	0.31(0.10)	0.22(0.14)	0.01(0.11)	0.05(0.15)
R.F. 1000x5	0.61(0.01)	0.53(0.05)	0.45(0.07)	0.46(0.02)	0.30(0.10)	0.38(0.06)	0.21(0.14)	-0.01(0.10)	-0.01(0.16)
R.F. 5000x5	0.61(0.01)	0.55(0.02)	0.46(0.07)	0.45(0.04)	0.31(0.09)	0.38(0.06)	0.21(0.14)	-0.00(0.10)	-0.01(0.16)
R.F. 1000x10	**0.73**(0.00)	0.57(0.03)	0.47(0.08)	0.48(0.03)	0.38(0.10)	0.38(0.07)	0.24(0.12)	-0.03(0.10)	-0.01(0.14)
CP	0.63(0.01)	0.53(0.03)	0.42(0.07)	0.38(0.08)	0.26(0.12)	0.31(0.10)	0.21(0.15)	-0.01(0.11)	0.05(0.19)
Tucker	0.61(0.01)	**0.61**(0.01)	**0.50**(0.09)	0.44(0.06)	0.25(0.12)	0.37(0.09)	0.22(0.14)	0.03(0.11)	0.04(0.16)

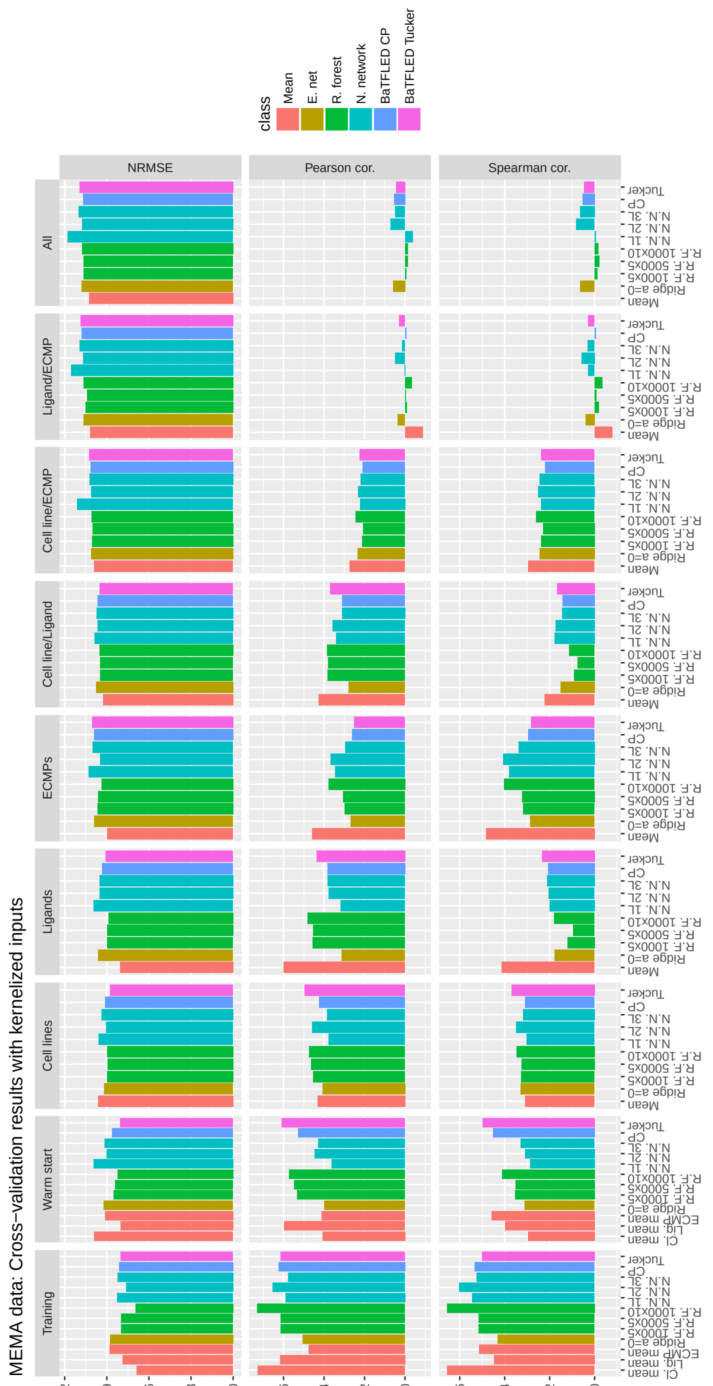


Figure 5.9: Performance measures for cross-validation runs on MEMA data (average across cv runs). Normalized root mean squared error, Pearson correlation and Spearman correlation for training data and four prediction tasks are shown. Note: y-axis differs between response measures.

Chapter 6

Conclusions and future directions

In this final chapter we review the BaTFLED model developed and tested here, discuss other possible applications of the method and propose possible improvements and extensions.

In the first chapter we introduced the problem of predicting drug response in cancer cell lines and discuss tensor factorization methods as a possible avenue to improve these studies. Previous studies extrapolate summary measures from dose-response curves which are then used to make connections with the cell line genetics and drug features. We proposed modeling the responses at each dose and introduce tensor methods as one framework to accomplish this goal. Tensor factorization techniques have been used in a broad array of fields including video separation and completion, image de-noising, predicting consumer preferences and explaining gene expression patterns.

Next we reviewed work utilizing tensor factorization to understand multi-dimensional datasets. This work was primarily focused on two tasks 1) using a factorization to fill in missing values in a tensor and 2) finding a factorization for a given tensor where the factor matrices can be interpreted relative to the problem domain. Two major types of factorization were highlighted, the simpler CANDECOMP/PARAFAC (CP) decomposition and the more flexible Tucker decomposition. Most previous methods focused on the CP decomposition as it is easier to compute and often uniquely determined. These decompositions can be found using direct algorithms when all of the responses are observed, but require a probabilistic framework when there are missing values and uncertainty. Probabilistic methods utilizing Bayesian techniques have been employed for similar tasks in a lower dimensional settings (matrix factorization) and we expand upon these ideas for this work.

Due to the size and complexity of tensor models, we propose using approximate variational inference to train the models and explain how this technique is implemented.

With this background established, we move to introducing the BaTFLED model (Bayesian Tensor Factorization Linked to External Data). This method is the first algorithm to use tensor factorization in a predictive framework that allows the user to make predictions for test examples not seen in the training data using features that characterize new examples. BaTFLED uses a Bayesian framework to decompose a three-dimensional tensor into three latent matrices and a smaller core tensor (in the case of a Tucker decomposition). The latent matrices are linked to input feature data through projection matrices that weight the importance of the input features. This model structure is assumed and the values in the projection matrices, latent matrices and core tensor are learned using variational inference. Missing response values can be filled in by multiplying the latent matrices together in a warm-start prediction task. Cold-start prediction for new examples from one, two or three of the modes are made by multiplying new feature vectors through the projection matrices and latent matrices. In addition, the values learned within the model can highlight feature importance and interactions between latent representations of the input data.

In describing the BaTFLED model, full details are provided on the model structure, the update equations and implementation in the R programming language. All the code for the algorithm is collected in a freely available package complete with full help documentation and a vignette showing an example use case on simulated data. Both the CP and Tucker decomposition are implemented and many options controlling the size of the factorization, the amount of sparsity and various run parameters are included.

In chapter 3 we examine the performance of BaTFLED on simulated data. Data is generated according to the structured assumed in the model and we show how BaTFLED is able to recover true interactions and make accurate predictions in cases where other methods do not perform well. Specifically, if there are higher order interactions (between two or more modes) present in the data, then elastic net, neural net and random forest models are not able to make good predictions or identify the salient features. In addition, we test the performance of our algorithm in comparison to these methods in several common settings: missing response data, random noise added to responses, fewer latent dimensions than the generating data and insufficient sample power. In each of these cases, BaTFLED is shown to be very robust with the performance degrading only in extreme cases.

In chapter 4 we explore the performance of BaTFLED and other methods on three real drug-response datasets. We first show results for a relatively small dataset of breast cancer cell lines that was used in an open challenge hosted by the DREAM (Dialogue on Reverse Engineering Assessment and Methods) effort. We explore both cross-validation and final test set results demonstrating that, in the task of predicting response at each dose, BaTFLED methods have an advantage over other methods both at the warm-start task and at predicting for cold-start cell lines. We also explore the features selected by each method and show performance for individual cell lines and drugs. Next we test BaTFLED on a similar but larger breast cancer dataset that was normalized to account for the growth rate of the cell lines. Again we see some advantages for the BaTFLED method at warm-start and cold-start cell line prediction. We also test a non-linear extension of the method using kernel functions to encode the input feature data. We find that this extension improves performance for cold-start drug prediction and cold-start cell line/drug combination prediction for BaTFLED models allowing them to outperform competitors. Next we consider the performance of these algorithms when predicting the GR_{50} curve summary measures extracted from the predictions at each dose. Here we find that BaTFLED outperforms other methods in the cold-start cell line prediction task and performs comparably to other methods at other tasks. Moreover we see that there is an advantage for all methods in predicting responses at each dose and then calculating GR_{50} values rather than predicting the GR_{50} values directly. Finally we test BaTFLED and elastic net methods on the largest dataset of this type that has been generated to date from the Cancer Target Discovery and Development (CTD^2) effort. Here we find that the BaTFLED CP method has the best performance at the warm-start task and although no method is able to outperform a simple baseline at the cold-start prediction tasks, BaTFLED methods outperform elastic net algorithms.

Finally, chapter 5 shows results for BaTFLED, ridge regression, random forest and neural network algorithms on a new task, predicting growth response for cell lines in varying micro-environments. This dataset measures cell counts for different cell lines grown in the presence of combinations of protein ligands and extra-cellular matrix proteins. For this task predictions are made for the warm-start task and for each of the three modes and combinations thereof using kernelized feature data for each of the three modes. This demonstrates the versatility and power of the method, showing superior performance for warm-start and cold-start cell line tasks.

6.1 Algorithmic improvements and extensions

Although the performance of the BaTFLED algorithm presented above is often better than baseline methods, there are still many areas in which algorithmic advances could improve its usefulness. We summarize a few ideas that we've had for improvements and extensions below, beginning with the more concrete ideas to improve the current model and moving toward larger ideas for research that build on this work. This is far from a complete list, but it demonstrates that the ideas presented here are more of a jumping off point for a large possible area of research.

Firstly, the algorithm is currently implemented in the statistical language R and most operations are done in series. Implementing the full package or some sub-routines in a lower-level language would likely improve run time performance. Also, using a graphical processing unit (GPU) for the operations that can be done in parallel would likely give a large boost in speed although this would primarily apply only to Tucker decompositions.

Secondly, the optimal number of iterations for training is often unclear and depends on the size of the data and the initialization of the variables. We explored several stopping criteria based on monitoring the lower bound or the training error, but we did not find a method that consistently avoided over-fitting and so we opted to run models for a set number of iterations. We did monitor the performance on test data at each iteration during cross-validation rounds to see which iteration produced the lowest error and this varied widely with different initialization, data and tasks. With further research on varied datasets, it may be possible to devise a stopping criterion that would be more optimal and this would likely improve predictive performance.

Third, depending on the data and task, the relative magnitude of values in the latent matrices may vary and this may cause some issues in balancing the tensor decomposition. For both the CP and Tucker methods, the latent matrices could be scaled relative to each other while still maintaining the same response values. Although this didn't seem to be a large problem in practice, our data were of similar sizes and we always used the same sparsity criteria for all modes. If this were not the case, the balance between modes may become a larger issue. One possibility to address this would be to add a step during the update process that re-scales these values.

Fourth, depending on the problem, it may be advantageous to use different prior structures when selecting features or interactions in the core tensor. Currently the user can only specify α and β values to be used for all features in a given mode. We briefly experimented with customizing these priors based on gene pathways or drug features that are known to work together. For example, if we wanted to encourage the first latent factor (column) in the projection matrix for cell line data to capture a specific gene pathway activation, we would encourage sparsity strongly ($\alpha = 10^{-10}$, $\beta = 10^{10}$) for genes outside of that pathway (in that column) and not encourage sparsity ($\alpha = 1$, $\beta = 1$) for genes in the pathway. Similarly for the priors on the core tensor, we could encourage sparsity less strongly for interactions that we wanted to favor. We experimented with this also, encouraging the core to prioritize lower-order interactions (between 1 or 2 modes), but did not notice an improvement in performance.

Fifth, it may be advantageous to combine various BaTFLED models or BaTFLED models with other models using ensemble methods. From the boxplots in figures 4.8 4.29 and 4.45 and the accompanying head-to-head scatter plots, we see that that different models have advantages for specific cell lines or drugs. By combining different models in an ensemble, for example, by averaging the predictions from several runs, performance may be improved. Since BaTFLED models seem to vary based on the random initialization and there are many different parameters to tune when running (CP vs Tucker, different sparsity, core sizes, etc.) these methods may be particularly amenable to using ensemble methods. Also, from the results on the Heiser dataset, there is an advantage to using kernelized features for some tasks and runs using kernels with different widths and weights could also be combined.

Sixth, since we have seen that using kernelized features can give an advantage, but since it is not obvious what kernel widths and weights would be best for any given dataset, a model that could learn these parameters during training would likely perform better. The probabilistic framework in the BaTFLED model could be extended to include such parameters which would be updated in the same way as the current model values.

Seventh, different distributions on the projection matrices may capture the latent factors better in some situations. We use the automatic relevance determination (ARD) prior for these values, and we explored using spike-and-slab priors and Laplace priors to encourage sparsity more strongly. We found that these more complex models were harder to train and didn't show much stronger sparsity, but they may be useful in other situations. For gene expression data, the three parameter beta prior presented by Gao et al. [122] and other similar factor models may be more useful.

Eighth, when testing BaTFLED models we tried using different number of latent factors for CP models and different size cores for Tucker models, it was unclear whether there was an optimal sized model to use. Rather than running separate models for each size, the Bayesian framework could be expanded to automatically determine the number of latent factors to use. Several publications have use this type of technique with matrix and tensor factorizations ([102], [103]), but they have not, to our knowledge, been applied to tensor factorization in a predictive setting yet.

Ninth, the projection framework the the current model only allows for linear transformations of the input data. While kernel methods can provide non-linearity, another option would be to replace the projection part of the model with a structure like a neural network. Here the input data would be passed through several layers of projection and non-linear transformations applied at each layer. This could allow for more flexibility in fitting data and may provide better predictions.

Tenth, an obvious extension to the model presented here is to expand the number of dimensions in the tensor. One could imagine that this would be useful when looking at drug combination data with more than two drugs, modeling cellular environments or when trying to combine data from different studies. In the latter case, we imagine combining data from the three studies analyzed here into one four dimensional model. The fourth mode would have responses for the same cell line and drug in different studies and features could encode differences in how the experiments were run and processed. This could help to explain differences between studies and identify the influence of different processing steps. In order to expand BaTFLED to higher dimensional settings, it would be preferable to avoid deriving the update equations directly. There have been some recent advancements in this area including the ‘black-box‘ variational inference developed by Ranganath et al. [123]

6.2 Other application directions

The BaTFLED algorithm was developed as a general use method and may be applied in any setting were responses are measured on combinations of different input data. Beginning with the case of drug responses in cancer cell lines, this method could be applied to make predictions for combination therapies. In this case, the responses would be some curve summary measure like GI_{50}, the first mode would correspond to cell lines and the other two modes would correspond to two different drugs being applied in combination. Similarly, aspects of the cell line micro-environment could be used in combination with drug treatments. With extensions to larger dimensional settings, models similar to BaTFLED could be used to explore and predict responses of cell lines to drug combinations at various doses, in different microenvironments and across different studies simultaneously.

Other possible areas where BaTFLED models may be useful include predicting protein or compound binding affinities to different substrates at different concentrations, modeling the spread of infectious diseases over space and time, modeling animal and plant distributions while accounting for genetic variations or even predicting stock movements in varying conditions over time. We are currently exploring some of these ideas and hope to expand the reach of these methods to many disparate fields.

Finally we hope that models of this type will ultimately be useful in predicting drug response in patients. Although we cannot test the same patient with different drugs in different conditions, we could imagine a sparse tensor capturing the responses for a large number of patients or groups of patients with varying environmental factors and genetics, but similar diseases and treatments. This could be used to recommend treatment for new cases, to predict progression and to identify important factors contributing to disease.

6.3 Conclusion

In this work we have developed a new machine learning algorithm 'BaTFLED' which predicts responses in a three-dimensional tensor given feature data for one or more of the dimensions. This model uses a probabilistic Bayesian framework and variational approximation to efficiently learn values in the model and can make predictions for new samples for one or or more of the modes. We test BaTFLED on simulated data comparing to state-of-the-art machine learning techniques and show that it outperforms these when higher dimensional interactions are present and that it is remarkably robust missing data, noise and insufficient power. We also test this method on four real datasets. Three measuring the growth response of cancer cell lines when treated with drugs at different doses and one measuring cell growth in differing micro-environments. We show that in many cases this method outperforms the other machine learning algorithms. In addition to performing prediction, the algorithm models interactions between the data from the three different modes which may be informative of the problem domain. Finally we discuss many possible improvements, extensions and new applications for this method. The algorithm is packaged for use by a general audience and freely available on CRAN (`https://cran.r-project.org/web/packages/BaTFLED3D/index.html`) and GitHub (`https://github.com/nathanlazar/BaTFLED3D`) complete with help documentation and running instructions. We believe that this work represents a significant advance in modeling and predicting multi-dimensional data and that it represents a new direction of research that is likely to play a large role in the future given the explosion in the production of large multi-dimensional datasets in biology and other fields.

Bibliography

[1] Bernard. W. Stewart and Christopher. P. Wild. *World Cancer Report 2014*. World Health Organization, 2014.

[2] D. Stehelin, H. E. Varmus, J. M. Bishop, and P. K. Vogt. DNA related to the transforming gene(s) of avian sarcoma viruses is present in normal avian DNA. *Nature*, 260(5547):170–173, March 1976.

[3] Benjamin Haibe-Kains, Nehme El-Hachem, Nicolai Juul Birkbak, Andrew C. Jin, Andrew H. Beck, Hugo J. W. L. Aerts, and John Quackenbush. Inconsistency in large pharmacogenomic studies. *Nature*, 504(7480):389–393, December 2013.

[4] Sreenath V. Sharma, Daniel A. Haber, and Jeff Settleman. Cell line-based platforms to evaluate the therapeutic efficacy of candidate anticancer agents. *Nature Reviews Cancer*, 10(4):241–253, April 2010.

[5] Richard M. Neve, Koei Chin, Jane Fridlyand, Jennifer Yeh, Frederick L. Baehner, Tea Fevr, Laura Clark, Nora Bayani, Jean-Philippe Coppe, Frances Tong, Terry Speed, Paul T. Spellman, Sandy DeVries, Anna Lapuk, Nick J. Wang, Wen-Lin Kuo, Jackie L. Stilwell, Daniel Pinkel, Donna G. Albertson, Frederic M. Waldman, Frank McCormick, Robert B. Dickson, Michael D. Johnson, Marc Lippman, Stephen Ethier, Adi Gazdar, and Joe W. Gray. A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes. *Cancer cell*, 10(6):515–527, December 2006.

[6] Eric A. Collisson, Anguraj Sadanandam, Peter Olson, William J. Gibb, Morgan Truitt, Shenda Gu, Janine Cooc, Jennifer Weinkle, Grace E. Kim, Lakshmi Jakkula, Heidi S. Feiler, Andrew H. Ko, Adam B. Olshen, Kathleen L. Danenberg, Margaret A. Tempero, Paul T. Spellman, Douglas Hanahan, and Joe W. Gray. Subtypes of pancreatic ductal adenocarcinoma and their differing responses to therapy. *Nature Medicine*, 17(4):500–503, April 2011.

[7] Robert H. Shoemaker. The NCI60 human tumour cell line anticancer drug screen. *Nature Reviews Cancer*, 6(10):813–823, October 2006.

[8] Martin H. Cohen, Grant A. Williams, Rajeshwari Sridhara, Gang Chen, W. David McGuinn, David Morse, Sophia Abraham, Atiqur Rahman, Chenyi Liang, Richard Lostritto, Amy Baird, and Richard Pazdur. United States Food and Drug Administration Drug Approval Summary Gefitinib (ZD1839; Iressa) Tablets. *Clinical Cancer Research*, 10(4):1212–1218, February 2004.

[9] Frances A. Shepherd, José Rodrigues Pereira, Tudor Ciuleanu, Eng Huat Tan, Vera Hirsh, Sumitra Thongprasert, Daniel Campos, Savitree Maoleekoonpiroj, Michael Smylie, Renato Martins, Maximiliano van Kooten, Mircea Dediu, Brian Findlay, Dongsheng Tu, Dianne Johnston, Andrea Bezjak, Gary Clark, Pedro Santabárbara, and Lesley Seymour. Erlotinib in previously treated nonâĂŞsmall-cell lung cancer. *New England Journal of Medicine*, 353(2):123–132, July 2005.

[10] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A. Margolin, Sungjoon Kim, Christopher J. Wilson, Joseph Lehar, Gregory V. Kryukov, Dmitriy Sonkin, Anupama Reddy, Manway Liu, Lauren Murray, Michael F. Berger, John E. Monahan, Paula Morais, Jodi Meltzer, Adam Korejwa, Judit Jane-Valbuena, Felipa A. Mapa, Joseph Thibault, Eva Bric-Furlong, Pichai Raman, Aaron Shipway, Ingo H. Engels, Jill Cheng, Guoying K. Yu, Jianjun Yu, Peter Aspesi, Melanie de Silva, Kalpana Jagtap, Michael D. Jones, Li Wang, Charles Hatton, Emanuele Palescandolo, Supriya Gupta, Scott Mahan, Carrie Sougnez, Robert C. Onofrio, Ted Liefeld, Laura MacConaill, Wendy Winckler, Michael Reich, Nanxin Li, Jill P. Mesirov, Stacey B. Gabriel, Gad Getz, Kristin Ardlie, Vivien Chan, Vic E. Myer, Barbara L. Weber, Jeff Porter, Markus Warmuth, Peter Finan, Jennifer L. Harris, Matthew Meyerson, Todd R. Golub, Michael P. Morrissey, William R. Sellers, Robert Schlegel, and Levi A. Garraway. The Cancer Cell Line Encyclopedia enables predictive modeling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, March 2012.

[11] Ming Yang, Yingming Li, and Zhongfei Zhang. Multi-Task Learning with Gaussian Matrix Generalized Inverse Gaussian Model. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 423–431. JMLR Workshop and Conference Proceedings, May 2013.

[12] Drew J. Adams, Daisuke Ito, Matthew G. Rees, Brinton Seashore-Ludlow, Xiaoling Puyang, Alex H. Ramos, Jaime H. Cheah, Paul A. Clemons, Markus Warmuth, Ping Zhu, Alykhan F. Shamji, and Stuart L. Schreiber. NAMPT Is the Cellular Target

of STF-31-Like Small-Molecule Probes. *ACS Chemical Biology*, 9(10):2247–2254, October 2014.

[13] Ultan McDermott, Sreenath V. Sharma, Lori Dowell, Patricia Greninger, Clara Montagut, Jennifer Lamb, Heidi Archibald, Raul Raudales, Angela Tam, Diana Lee, S. Michael Rothenberg, Jeffrey G. Supko, Raffaella Sordella, Lindsey E. Ulkus, A. John Iafrate, Shyamala Maheswaran, Ching Ni Njauw, Hensin Tsao, Lisa Drew, Jeff H. Hanke, Xiao-Jun Ma, Mark G. Erlander, Nathanael S. Gray, Daniel A. Haber, and Jeffrey Settleman. Identification of genotype-correlated sensitivity to selective kinase inhibitors by using high-throughput tumor cell line profiling. *Proceedings of the National Academy of Sciences of the United States of America*, 104(50):19936–19941, December 2007.

[14] Martin L. Sos, Kathrin Michel, Thomas Zander, Jonathan Weiss, Peter Frommolt, Martin Peifer, Danan Li, Roland Ullrich, Mirjam Koker, Florian Fischer, Takeshi Shimamura, Daniel Rauh, Craig Mermel, Stefanie Fischer, Isabel Stückrath, Stefanie Heynck, Rameen Beroukhim, William Lin, Wendy Winckler, Kinjal Shah, Thomas LaFramboise, Whei F. Moriarty, Megan Hanna, Laura Tolosi, Jörg Rahnenführer, Roel Verhaak, Derek Chiang, Gad Getz, Martin Hellmich, Jürgen Wolf, Luc Girard, Michael Peyton, Barbara A. Weir, Tzu-Hsiu Chen, Heidi Greulich, Jordi Barretina, Geoffrey I. Shapiro, Levi A. Garraway, Adi F. Gazdar, John D. Minna, Matthew Meyerson, Kwok-Kin Wong, and Roman K. Thomas. Predicting drug susceptibility of non-small cell lung cancers based on genetic lesions. *Journal of Clinical Investigation*, 119(6):1727–1740, June 2009.

[15] Anneleen Daemen, Obi L Griffith, Laura M Heiser, Nicholas J Wang, Oana M Enache, Zachary Sanborn, Francois Pepin, Steffen Durinck, James E Korkola, Malachi Griffith, Joe S Hur, Nam Huh, Jongsuk Chung, Leslie Cope, Mary Jo Fackler, Christopher Umbricht, Saraswati Sukumar, Pankaj Seth, Vikas P Sukhatme, Lakshmi R Jakkula, Yiling Lu, Gordon B Mills, Raymond J Cho, Eric A Collisson, Laura J van't Veer, Paul T Spellman, and Joe W Gray. Modeling precision treatment of breast cancer. *Genome Biology*, 14(10):R110, 2013.

[16] Laura M. Heiser, Anguraj Sadanandam, Wen-Lin Kuo, Stephen C. Benz, Theodore C. Goldstein, Sam Ng, William J. Gibb, Nicholas J. Wang, Safiyyah Ziyad, Frances Tong, Nora Bayani, Zhi Hu, Jessica I. Billig, Andrea Dueregger, Sophia Lewis, Lakshmi Jakkula, James E. Korkola, Steffen Durinck, Francois Pepin, Yinghui Guan, Elizabeth Purdom, Pierre Neuvial, Henrik Bengtsson, Kenneth W. Wood, Peter G. Smith, Lyubomir T. Vassilev, Bryan T. Hennessy, Joel Greshock, Kurtis E.

Bachman, Mary Ann Hardwicke, John W. Park, Laurence J. Marton, Denise M. Wolf, Eric A. Collisson, Richard M. Neve, Gordon B. Mills, Terence P. Speed, Heidi S. Feiler, Richard F. Wooster, David Haussler, Joshua M. Stuart, Joe W. Gray, and Paul T. Spellman. Subtype and pathway specific responses to anticancer compounds in breast cancer. *Proceedings of the National Academy of Sciences*, 109(8):2724–2729, February 2012.

[17] Wen-Lin Kuo, Debopriya Das, Safiyyah Ziyad, Sanchita Bhattacharya, William J. Gibb, Laura M. Heiser, Anguraj Sadanandam, Gerald V. Fontenay, Zhi Hu, Nicholas J. Wang, Nora Bayani, Heidi S. Feiler, Richard M. Neve, Andrew J. Wyrobek, Paul T. Spellman, Laurence J. Marton, and Joe W. Gray. A systems analysis of the chemosensitivity of breast cancer cells to the polyamine analogue PG-11047. *BMC Medicine*, 7(1):77, December 2009.

[18] Mohammad Fallahi-Sichani, Saman Honarnejad, Laura M. Heiser, Joe W. Gray, and Peter K. Sorger. Metrics other than potency reveal systematic variation in responses to cancer drugs. *Nature Chemical Biology*, 9(11):708–714, November 2013.

[19] C. Hatzis, P. L. Bedard, N. J. Birkbak, A. H. Beck, H. J. W. L. Aerts, D. F. Stern, L. Shi, R. Clarke, J. Quackenbush, and B. Haibe-Kains. Enhancing Reproducibility in Cancer Drug Screening: How Do We Move Forward? *Cancer Research*, 74(15):4016–4023, August 2014.

[20] The Cancer Cell Line Encyclopedia Consortium and The Genomics of Drug Sensitivity in Cancer Consortium. Pharmacogenomic agreement between two cancer cell line data sets. *Nature*, 528(7580):84–87, December 2015.

[21] Zhaleh Safikhani, Petr Smirnov, Mark Freeman, Nehme El-Hachem, Adrian She, Quevedo Rene, Anna Goldenberg, Nicolai Juul-Birkbak, Christos Hatzis, Leming Shi, Andrew H. Beck, Hugo J.W.L. Aerts, John Quackenbush, and Benjamin Haibe-Kains. Revisiting inconsistency in large pharmacogenomic studies. *F1000Research*, 5:2333, September 2016.

[22] Marc Hafner, Mario Niepel, Mirra Chung, and Peter K. Sorger. Growth rate inhibition metrics correct for confounders in measuring sensitivity to cancer drugs. *Nature Methods*, 13(6):521–527, June 2016.

[23] In Sock Jang, Elias Chaibub Neto, Juistin Guinney, Stephen H. Friend, and Adam A. Margolin. Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 63–74, 2014.

[24] James C. Costello, Laura M. Heiser, Elisabeth Georgii, Mehmet Gönen, Michael P. Menden, Nicholas J. Wang, Mukesh Bansal, Muhammad Ammad-ud din, Petteri Hintsanen, Suleiman A. Khan, John-Patrick Mpindi, Olli Kallioniemi, Antti Honkela, Tero Aittokallio, Krister Wennerberg, Nci Dream Community, James J. Collins, Dan Gallahan, Dinah Singer, Julio Saez-Rodriguez, Samuel Kaski, Joe W. Gray, and Gustavo Stolovitzky. A community effort to assess and improve drug sensitivity prediction algorithms. *Nature Biotechnology*, 32(12):1202–1212, December 2014.

[25] Muhammad Ammad-ud din, Elisabeth Georgii, Mehmet Gönen, Tuomo Laitinen, Olli Kallioniemi, Krister Wennerberg, Antti Poso, and Samuel Kaski. Integrative and Personalized QSAR Analysis in Cancer by Kernelized Bayesian Matrix Factorization. *Journal of Chemical Information and Modeling*, 54(8):2347–2359, August 2014.

[26] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[27] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, November 1901.

[28] Jeffrey T Leek and John D Storey. Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis. *PLoS Genet*, 3(9):e161, September 2007.

[29] Jean-Philippe Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, 101(12):4164–4169, March 2004.

[30] Yuan Gao and George Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, January 2005.

[31] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SDM*, volume 10, pages 211–222. SIAM, 2010.

[32] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications, November 2007.

[33] Frank Lauren Hitchcock. *The Expression of a Tensor Or a Polyadic as a Sum of Products.* sn., 1927.

[34] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[35] Richard A. Harshman. Foundations of the parafac procedure: models and conditions for an “explanatory” multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[36] C. J. Appellof and E. R. Davidson. Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056, November 1981.

[37] Rasmus Bro. Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis. *Chemometrics and Intelligent Laboratory Systems*, 46(2):133–147, March 1999.

[38] John M. Henshaw, Lloyd W. Burgess, Karl S. Booksh, and Bruce R. Kowalski. Multicomponent Determination of Chlorinated Hydrocarbons Using a Reaction-Based Chemical Sensor. 1. Multivariate Calibration of Fujiwara Reaction Products. *Analytical Chemistry*, 66(20):3328–3336, October 1994.

[39] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis with applications in the chemical sciences.* John Wiley & Sons, Ltd, 2004.

[40] Barry M. Wise, Neal B. Gallagher, Stephanie Watts Butler, Daniel D. White, and Gabriel G. Barna. A comparison of principal component analysis, multiway principal component analysis, trilinear decomposition and parallel factor analysis for fault detection in a semiconductor etch process. *Journal of Chemometrics*, 13(3-4):379–396, May 1999.

[41] Pierre Comon. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, April 1994.

[42] Pieter M. Kroonenberg. *Applied multiway data analysis.* John Wiley & Sons, Inc., 2008.

[43] Lieven De Lathauwer and Joséphine Castaing. Tensor-based techniques for the blind separation of DS-CDMA signals. *Signal Processing*, 87(2):322–336, February 2007.

[44] N.D. Sidiropoulos, R. Bro, and G.B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388, August 2000.

[45] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro. Blind PARAFAC receivers for DS-CDMA systems. *IEEE Transactions on Signal Processing*, 48(3):810–823, March 2000.

[46] Joos Lievende Moor de Lathauwer, BartVandewalle. On the Best Rank-1 and Rank-(r1, R2, . . . , Rn) Approximation of Higher-Order Tensors. *SIAM Journal on Matrix Analysis & Applications*, 21(4):1324–1342, April 2000.

[47] L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology.*, pages 110–127. Holt, Rinehart and Winston, New York, 1964.

[48] L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. In C. W. Harris, editor, *Problems in measuring change.*, pages 122–137. University of Wisconsin Press, Madison WI, 1963.

[49] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, September 1966.

[50] L. De Lathauwer, B. De Moor, and J. Vandewalle. A Multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, January 2000.

[51] Pieter M. Kroonenberg and Jan de Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, March 1980.

[52] Claus A. Andersson and Rene Henrion. A general algorithm for obtaining simple structure of core arrays in< i> N</i>-way PCA with application to fluorometric data. *Computational statistics & data analysis*, 31(3):255–278, 1999.

[53] Claus A. Andersson René Henrion. A new criterion for simple-structure transformations of core arrays in N-way principal components analysis. *Chemometrics and Intelligent Laboratory Systems*, (2):189–204, 1999.

[54] Paul J. Gemperline, Kevin H. Miller, Terry L. West, John E. Weinstein, J. Craig Hamilton, and John T. Bray. Principal component analysis, trace elements, and blue crab shell disease. *Analytical Chemistry*, 64(9):523A–532A, May 1992.

[55] P. M. Kroonenberg. *Three-mode principal component analysis : theory and applications.* Doctoral thesis, April 1983. Promotor: J. de Leeuw.

[56] M. Alex O. Vasilescu and Demetri Terzopoulos. Multilinear image analysis for facial recognition. In *Pattern Recognition, International Conference on*, volume 2, pages 20511–20511. IEEE Computer Society, 2002.

[57] Ricard Boqué and Age K. Smilde. Monitoring and diagnosing batch processes with multiway covariates regression models. *AIChE Journal*, 45(7):1504–1520, July 1999.

[58] Ben C. Mitchell and Donald S. Burdick. Slowly converging parafac sequences: Swamps and two-factor degeneracies. *Journal of Chemometrics*, 8(2):155–168, March 1994.

[59] Rasmus Bro and Henk A. L. Kiers. A new efficient method for determining the number of components in PARAFAC models. *Journal of Chemometrics*, 17(5):274–286, May 2003.

[60] Pentti Paatero. Construction and analysis of degenerate PARAFAC models. *Journal of Chemometrics*, 14(3):285–299, May 2000.

[61] William S. Rayens and Benjamin C. Mitchell. Two-factor degeneracies and a stabilization of PARAFAC. *Chemometrics and Intelligent Laboratory Systems*, 38(2):173–181, October 1997.

[62] Pierre Comon, Xavier Luciani, and André LF De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405, 2009.

[63] Nicolaas (Klaas) M. Faber, Rasmus Bro, and Philip K. Hopke. Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. *Chemometrics and Intelligent Laboratory Systems*, 65(1):119–137, January 2003.

[64] Giorgio Tomasi and Rasmus Bro. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis*, 50(7):1700–1734, April 2006.

[65] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. Probabilistic Latent Tensor Factorization Model for Link Pattern Prediction in Multi-relational Networks. *arXiv:1204.2588 [cs, stat]*, April 2012. arXiv: 1204.2588.

[66] Qibin Zhao, Guoxu Zhou, Liqing Zhang, Andrzej Cichocki, and Shun-ichi Amari. Robust Bayesian Tensor Factorization for Incomplete Multiway Data. *arXiv:1410.2386 [cs]*, October 2014. arXiv: 1410.2386.

[67] Zenglin Xu, Feng Yan, Yuan, and Qi. Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis. *arXiv:1108.6296 [cs]*, August 2011. arXiv: 1108.6296.

[68] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, September 2009.

[69] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 426–433. ACM, 2005.

[70] Miao Zhang and C. Ding. Robust Tucker Tensor Decomposition for Effective Image Representation. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 2448–2455, December 2013.

[71] Yanwei Pang, Zhao Ma, Jing Pan, and Yuan Yuan. Robust Sparse Tensor Decomposition by Probabilistic Latent Semantic Analysis. In *2011 Sixth International Conference on Image and Graphics (ICIG)*, pages 893–896, August 2011.

[72] Evrim Acar, Tamara G. Kolda, Daniel M. Dunlavy, and Morten Morup. Scalable Tensor Factorizations for Incomplete Data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011. arXiv: 1005.2197.

[73] Silvia Gandy, Benjamin Recht, and Isao Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[74] Heng Huang and C. Ding. Robust tensor factorization using R1 norm. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, June 2008.

[75] Ji Liu, P. Musialski, P. Wonka, and Jieping Ye. Tensor Completion for Estimating Missing Values in Visual Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, January 2013.

[76] Wenzhe Pei, Tao Ge, and Baobao Chang. Max-Margin Tensor Neural Network for Chinese Word Segmentation. In *ACL (1)*, pages 293–303, 2014.

[77] Dong Yu, Li Deng, and Frank Seide. The Deep Tensor Neural Network With Applications to Large Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):388–396, February 2013.

[78] J. T. Chien and Y. T. Bao. Tensor-Factorized Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–14, 2017.

[79] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods. *arXiv:1506.08473 [cs, stat]*, June 2015. arXiv: 1506.08473.

[80] Nadav Cohen and Amnon Shashua. Inductive Bias of Deep Convolutional Networks through Pooling Geometry. *arXiv:1605.06743 [cs]*, May 2016. arXiv: 1605.06743.

[81] Nadav Cohen and Amnon Shashua. Convolutional Rectifier Networks as Generalized Tensor Decompositions. *arXiv:1603.00162 [cs]*, March 2016. arXiv: 1603.00162.

[82] Nadav Cohen, Or Sharir, and Amnon Shashua. On the Expressive Power of Deep Learning: A Tensor Analysis. *arXiv:1509.05009 [cs, stat]*, September 2015. arXiv: 1509.05009.

[83] Sanne Engelen, Stina Frosch, and Bo Munk Jørgensen. A fully robust PARAFAC method for analyzing fluorescence data. *Journal of Chemometrics*, 23(3):124–131, March 2009.

[84] Larsson Omberg, Gene H. Golub, and Orly Alter. A tensor higher-order singular value decomposition for integrative analysis of DNA microarray data from different studies. *Proceedings of the National Academy of Sciences*, 104(47):18371–18376, November 2007.

[85] Sri Priya Ponnapalli, Michael A. Saunders, Charles F. Van Loan, and Orly Alter. A Higher-Order Generalized Singular Value Decomposition for Comparison of Global mRNA Expression from Multiple Organisms. *PLoS ONE*, 6(12):e28072, December 2011.

[86] Victoria Hore, Ana Viñuela, Alfonso Buil, Julian Knight, Mark I. McCarthy, Kerrin Small, and Jonathan Marchini. Tensor decomposition for multiple-tissue gene expression experiments. *Nature Genetics*, 48(9):1094–1100, September 2016.

[87] Arno Onken, Jian K. Liu, P. P. Chamanthi R. Karunasekara, Ioannis Delis, Tim Gollisch, and Stefano Panzeri. Using Matrix and Tensor Factorizations for the Single-Trial Analysis of Population Spike Trains. *PLOS Comput Biol*, 12(11):e1005189, November 2016.

[88] Yejin Kim, Robert El-Kareh, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Discriminative and Distinct Phenotyping by Constrained Tensor Factorization. *Scientific Reports*, 7(1):1114, April 2017.

[89] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE transactions on pattern analysis and machine intelligence*, 6(6):721–741, June 1984.

[90] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[91] Christian P. Robert and George Casella. *Introducing Monte Carlo methods with R.* Use R! Springer, New York, 2010. OCLC: ocn462920377.

[92] Andrew Gelman. *Bayesian data analysis.* Chapman & Hall/CRC texts in statistical science. CRC Press, Boca Raton, third edition edition, 2014.

[93] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[94] David Lunn. *The BUGS book: a practical introduction to Bayesian analysis.* Texts in statistical science. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2013.

[95] Martyn Plummer and others. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd international workshop on distributed statistical computing*, volume 124, page 125. Vienna, 2003.

[96] Stan Development Team. Stan Modeling Language Users Guide and Reference Manual, Version 2.14.0., 2016.

[97] Ian Porteous, Evgeniy Bart, and Max Welling. Multi-HDP: A Non Parametric Bayesian Model for Tensor Factorization. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

[98] Suleiman A. Khan and Muhammad Ammad-ud din. tensorBF: an R package for Bayesian tensor factorization. *bioRxiv*, page 097048, December 2016.

[99] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[100] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[101] Wei Chu and Zoubin Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. 2009.

[102] Suleiman A. Khan and Samuel Kaski. Bayesian Multi-view Tensor Factorization. *Machine Learning and Knowledge Discovery in Databases*, pages 656–671, 2014.

[103] Suleiman A. Khan, Eemeli Leppäaho, and Samuel Kaski. Bayesian multi-tensor factorization. *Machine Learning*, 105(2):233–253, November 2016.

[104] Adrian R. Groves, Christian F. Beckmann, Steve M. Smith, and Mark W. Woolrich. Linked independent component analysis for multimodal data fusion. *NeuroImage*, 54(3):2198–2217, February 2011.

[105] Hui Zou. *Some perspectives of sparse statistical modeling.* PhD thesis, Citeseer, 2005.

[106] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 2010.

[107] Spencer Aiello, Tom Kraljevic, Petr Maj, and with contribution from the H2O.ai team. h2o: R Interface for H2o, 2016. R package version 3.10.0.8.

[108] Amrita Basu, Nicole E. Bodycombe, Jaime H. Cheah, Edmund V. Price, Ke Liu, Giannina I. Schaefer, Richard Y. Ebright, Michelle L. Stewart, Daisuke Ito, Stephanie Wang, Abigail L. Bracha, Ted Liefeld, Mathias Wawer, Joshua C. Gilbert, Andrew J. Wilson, Nicolas Stransky, Gregory V. Kryukov, Vlado Dancik, Jordi Barretina, Levi A. Garraway, C. Suk-Yee Hon, Benito Munoz, Joshua A. Bittker, Brent R. Stockwell, Dineo Khabele, Andrew M. Stern, Paul A. Clemons, Alykhan F. Shamji, and Stuart L. Schreiber. An Interactive Resource to Identify Cancer Genetic and Lineage Dependencies Targeted by Small Molecules. *Cell*, 154(5):1151–1161, August 2013.

[109] Nidhi Bindal, Simon A. Forbes, David Beare, Prasad Gunasekaran, Kenric Leung, Chai Y. Kok, Mingming Jia, Sally Bamford, Charlotte Cole, Sari Ward, Jon Teague, Michael R. Stratton, Peter Campbell, and Andrew P. Futreal. COSMIC: the catalogue of somatic mutations in cancer. *Genome Biology*, 12(Suppl 1):P3, September 2011.

[110] Chun Wei Yap. PaDEL-descriptor: An open source software to calculate molecular descriptors and fingerprints. *Journal of Computational Chemistry*, 32(7):1466–1474, May 2011.

[111] Matthew Furia. synapseClient: Synapse R Client from Sage Bionetworks. 2016.

[112] Amar Koleti, Vasileios Stathias, Raymond Terryn, Michele Forlin, Dusica Vidović, Caty Chung, Wen Niu, Caroline Monteiro, Christopher Mader, Avi Ma'ayan, Mario Medvedovic, and Stephan Schürer. The LINCS Data Portal and FAIR LINCS Dataset Landing Pages, November 2016. DOI:10.6084/m9.figshare.4244210.v2.

[113] Christian Ritz, Florent Baty, Jens C. Streibig, and Daniel Gerhard. Dose-Response Analysis Using R. *PLOS ONE*, 10(12):e0146021, December 2015.

[114] Brinton Seashore-Ludlow, Matthew G. Rees, Jaime H. Cheah, Murat Cokol, Edmund V. Price, Matthew E. Coletti, Victor Jones, Nicole E. Bodycombe, Christian K. Soule, Joshua Gould, Benjamin Alexander, Ava Li, Philip Montgomery, Mathias J. Wawer, Nurdan Kuru, Joanne D. Kotz, C. Suk-Yee Hon, Benito Munoz, Ted Liefeld, Vlado Dancik, Joshua A. Bittker, Michelle Palmer, James E. Bradner, Alykhan F. Shamji, Paul A. Clemons, and Stuart L. Schreiber. Harnessing Connectivity in a Large-Scale Small-Molecule Sensitivity Dataset. *Cancer Discovery*, October 2015.

[115] Matthew G. Rees, Brinton Seashore-Ludlow, Jaime H. Cheah, Drew J. Adams, Edmund V. Price, Shubhroz Gill, Sarah Javaid, Matthew E. Coletti, Victor L. Jones, Nicole E. Bodycombe, Christian K. Soule, Benjamin Alexander, Ava Li, Philip Montgomery, Joanne D. Kotz, C. Suk-Yee Hon, Benito Munoz, Ted Liefeld, Vlado Dancik, Daniel A. Haber, Clary B. Clish, Joshua A. Bittker, Michelle Palmer, Bridget K. Wagner, Paul A. Clemons, Alykhan F. Shamji, and Stuart L. Schreiber. Correlating chemical sensitivity and basal gene expression reveals mechanism of action. *Nature Chemical Biology*, 12(2):109–116, February 2016.

[116] Mark A. LaBarge, Bahram Parvin, and James B. Lorens. Molecular deconstruction, detection, and computational prediction of microenvironment-modulated cellular responses to cancer therapeutics. *Advanced Drug Delivery Reviews*, 69âĂŞ70:123–131, April 2014.

[117] Anne E. Carpenter, Thouis R. Jones, Michael R. Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A. Guertin, Joo Han Chang, Robert A. Lindquist, Jason Moffat, Polina Golland, and David M. Sabatini. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, 7(10):R100, October 2006.

[118] Justin Lamb, Emily D. Crawford, David Peck, Joshua W. Modell, Irene C. Blat, Matthew J. Wrobel, Jim Lerner, Jean-Philippe Brunet, Aravind Subramanian, Kenneth N. Ross, Michael Reich, Haley Hieronymus, Guo Wei, Scott A. Armstrong, Stephen J. Haggarty, Paul A. Clemons, Ru Wei, Steven A. Carr, Eric S. Lander, and

Todd R. Golub. The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. *Science*, 313(5795):1929–1935, September 2006.

[119] Karin Breuer, Amir K. Foroushani, Matthew R. Laird, Carol Chen, Anastasia Sribnaia, Raymond Lo, Geoffrey L. Winsor, Robert E. W. Hancock, Fiona S. L. Brinkman, and David J. Lynn. InnateDB: systems biology of innate immunity and beyond-recent updates and continuing curation. *Nucleic Acids Research*, 41(D1):D1228–D1233, January 2013.

[120] Melissa S. Cline, Michael Smoot, Ethan Cerami, Allan Kuchinsky, Nerius Landys, Chris Workman, Rowan Christmas, Iliana Avila-Campilo, Michael Creech, Benjamin Gross, Kristina Hanspers, Ruth Isserlin, Ryan Kelley, Sarah Killcoyne, Samad Lotia, Steven Maere, John Morris, Keiichiro Ono, Vuk Pavlovic, Alexander R. Pico, Aditya Vailaya, Peng-Liang Wang, Annette Adler, Bruce R. Conklin, Leroy Hood, Martin Kuiper, Chris Sander, Ilya Schmulevich, Benno Schwikowski, Guy J. Warner, Trey Ideker, and Gary D. Bader. Integration of biological networks and gene expression data using Cytoscape. *Nature Protocols*, 2(10):2366–2382, October 2007.

[121] Diego Alonso-López, Miguel A. Gutiérrez, Katia P. Lopes, Carlos Prieto, Rodrigo Santamaría, and Javier De Las-Rivas. APID interactomes: providing proteome-based interactomes with controlled quality for multiple species and derived networks. *Nucleic Acids Research*, 44(W1):W529–W535, July 2016.

[122] Chuan Gao, Christopher D. Brown, and Barbara E. Engelhardt. A latent factor model with a mixture of sparse and dense factors to model gene expression data with confounding effects. *arXiv:1310.4792 [q-bio, stat]*, October 2013. arXiv: 1310.4792.

[123] Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black Box Variational Inference. In *AISTATS*, pages 814–822, 2014.