Improving Maximum Daily Salinity Regressor Performance in the Columbia River Estuary Project

Rafael de Jesús Fernández Moctezuma
I.S.C., ITESM Campus San Luis Potosí, 2001

A thesis submitted to the faculty of the
Oregon Graduate Institute of Science and Technology
in partial fulfillment of the
requirements for the degree
Master of Science
in
Computer Science and Engineering

October 2005

© Copyright 2005 by Rafael de Jesús Fernández Moctezuma
All Rights Reserved

The thesis “Improving Maximum Daily Salinity Regressor Performance in the Columbia River Estuary Project” by Rafael de Jesus Fernandez Moctezuma has been examined and approved by the following Examination Committee:

Dr. Todd K. Leen, Adviser
Professor

Dr. Miguel A. Carreira-Perpinan
Assistant Professor

Dr. Deniz Erdogmus
Assistant Professor

Dedication

To my parents, Rafael de Jesús and Rosa Imelda, and to my grandmother Rosa María.

A mis padres, Rafael de Jesús y Rosa Imelda, y a mi abuela Rosa María.

Acknowledgements

This thesis is the result of Prof. Todd K. Leen's advising. Thank you for giving me the opportunity to work with you, Todd. You have been a true mentor, and I owe you a great deal of professional growth.

Thank you Prof. Jesús Leyva Ramos for encouraging me to pursue an advanced degree, and for nurturing my interest in Science.

Thank you to all my family, friends, and loved ones for all your support and encouragement.

Thank you Zhengdong Lu and Haiming Zheng, it was a pleasure to work with you. You always had important comments, ideas, and suggestions.

Special thanks to Bob and Miguel, true scholars and great friends. Thank you for your guidance. I guess we could say "Thanks for showing me *the way.*"

This work was supported by a scholarship granted by the Consejo Nacional de Ciencia y Tecnología (CONACyT), México. I am grateful and honored to have received your support.

Contents

List of Tables

List of Figures

Abstract

Improving Maximum Daily Salinity Regressor Performance in the Columbia River Estuary Project

Rafael de Jesús Fernández Moctezuma

Supervising Professor: Dr. Todd K. Leen

The goal of this research is to improve the performance of the Maximum Daily Salinity regressor used in the fault detection mechanism deployed in the Columbia River estuary (CORIE). The Center for Coastal and Land-Margin Research is developing an Environmental Observation and Forecasting System. The goal of the CORIE project is to gain a better understanding of the estuary. The team has deployed sensors in the estuary to measure salinity, temperature, pressure, and velocity. Of these sensors, salinity sensors are subject to bio-fouling, an event that results in data loss over time. Previous work in fault detection helped prevent data loss.

Our work improves the performance of the regressor used as part of the detector architecture. We looked at temperature measurements as inputs for the salinity regressor. We used the Gaussian Mixture Model to build a new salinity regressor. In addition to the Gaussian Mixture Model, we attempted to include historical

information into our regressor, explored the use of single-layer neural networks, and considered incorporating measurements from nearby stations to improve regressor performance. We also considered incorporating numerical predictions for salinity from SELFE, a numerical model of the estuary developed by the CORIE team. We show a performance comparison of the original and new regressors.

Chapter 1

Introduction

A common problem in statistics is to try to determine the relationship between several random variables. *Regression* accomplishes the task of determining the relationship. Bishop [3], Duda et al. [6], and Mitchell [7] define regression as a method to find a description of the data in terms of a function. To illustrate this concept, consider a set of measurements of height and weight of individuals from a population (see Table 1.1.) We assume these two variables are dependent. We express the relation as a *target function*. We will fit the model function's parameters to minimize a *measurement of error*. This error measurement provides performance information – the relation between the estimated target function's value at a given point and the observed (true) value.

Table 1.1: Sample measurements of height and weight from a population. Height is expressed in inches and weight is expressed in pounds. The measurements are indexed with the variable i.

Index (i)	Height (Inches)	Weight (Pounds)
1	75	163
2	73	171
3	67	151
4	64	141

If we assume the relation between height and weight is linear, we can write a

model function to predict weigth with the form

$$\hat{f}(x) = ax + b \tag{1.1}$$

where a and b are the function's parameters, and x is height. To measure the error, we can use the mean squared error (MSE)

$$MSE \equiv \frac{1}{D}\sum_{i=1}^{D}(f(x_i) - \hat{f}(x_i))^2 \tag{1.2}$$

where D is the number of observed data points, $f(x_i)$ is the observed weight value for a given point (indexed by i), and $\hat{f}(x_i)$ is the value of the model function (in this example, the estimated weight) for a given height x_i. We can also define a different model function, such as a quadratic:

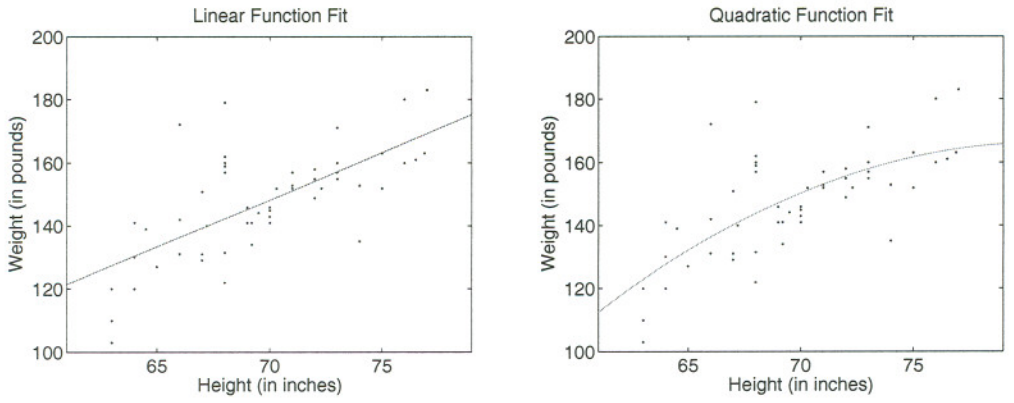
$$\hat{f}(x) = ax^2 + bx + c. \tag{1.3}$$

For any choice of model function, the learning task is to fit its parameters (a and b for Equation 1.1, a, b, and c for Equation 1.3.) Figure 1.1 illustrates a linear fit and a quadratic fit on height and wight data created to illustrate regression.

The results from the example (shown in Figure 1.1) show how the choice of a different model function produced a different regressor. The MSE dropped when we used a quadratic fit instead of a linear fit. It is clear that a critical step when regressing a variable in terms of other(s) is the choice of the model function. Not everything can be represented as a straight line.

We have reviewed the basic concepts of regression and showed an example with two choices of model functions. This brings us to a comfortable point to describe the work of this research.

As part of an Environmental Observation and Forecasting System, sensors deployed in the Columbia River estuary (CORIE) collect salinity, temperature, pressure, and velocity information. The salinity sensors are subject to bio-fouling, an event that causes decay of the measured maximum diurnal salinity. This leads to



(a) Linear fit

(b) Quadratic fit

Figure 1.1: Weight as a function of height. There are 52 available observations. Part (a) shows a linear fit superimposed on available data. Part (b) shows a quadratic fit superimposed on available data. The MSE for the linear fit is 151.5 and for the quadratic fit is 146.6.

undesirable data loss. Archer et al. [1, 2] developed an automatic fault detection mechanism to alert the CORIE staff at early stages of bio-fouling. The fault detection architecture uses a sequential likelihood ratio test. As part of this test, they built a regressor for maximum daily salinity using temperature information.

This research improves the performance of the regressor used in the CORIE project. Chapter 2 presents the Gaussian Mixture Model, our choice for regression. Chapter 3 details the problem scenario, reviews the previous work on CORIE and presents our choices for improving the regressor performance. Chapter 4 shows experimental results. The Appendix describes alternative approaches we tried to improve performance, and should provide documentation of our experience with these approaches for future reference.

Chapter 2

Regression with the Gaussian Mixture Model

Our intention to improve the performance of the maximum daily salinity regressor deployed on the CORIE project led us to consider the use of the Gaussian Mixture Model. This chapter reviews the Gaussian Probability distribution, presents properties amicable for regression, introduces the Mixture Model, and discusses the EM algorithm for fitting Gaussian Mixture parameters.

2.1 The Gaussian Probability Distribution

The French mathematician Abraham de Moivre developed the *Normal Distribution* in the early 18th century. In the early 19th century, Carl Friedrich Gauss used it to analyze the distribution of errors in astronomical observations. The distribution bears Gauss' name extensively in the engineering and physics literature[1]. The shape of this distribution resembles a bell (see Figure 2.1.)

When an input x is only one variable, we define this *univariate* probability distribution as

$$p(x) \equiv \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)\right) \tag{2.1}$$

[1]For interesting historical information about Gauss, visit the MacTutor History of Mathematics archive, http://www-history.mcs.st-andrews.ac.uk/history/

where μ is the *mean* and σ^2 is the *variance*. If we have D samples, we can calculate the mean and variance [2] as

$$\mu = \frac{1}{D}\sum_{i=1}^{D} x_i \tag{2.2}$$

$$\sigma^2 = \frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2. \tag{2.3}$$

The square root of the variance, σ, is called the *standard deviation*. Notice how we fully describe the Gaussian distribution by its mean and variance. If we sample from a Gaussian distribution, we expect to obtain μ, since this point has the highest probability. If a random variable x is distributed as Gaussian, we say that the *expected value* of x is the mean of the Gaussian distribution:

$$E[x] = \mu. \tag{2.4}$$

When the input is an n-dimensional vector $\bar{x}$, the Gaussian distribution is

$$p(\bar{x}) \equiv \mathcal{N}(x; \bar{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} exp\left(-\frac{1}{2}(\bar{x} - \bar{\mu})^T \Sigma^{-1} (\bar{x} - \bar{\mu})\right). \tag{2.5}$$

For n-dimensional inputs, the mean $\bar{\mu}$ is an n-dimensional vector, and the covariance Σ is an $n \times n$ matrix. In equation 2.5, $|\Sigma|$ denotes the determinant of the covariance matrix, Σ^{-1} is the *inverse* of the covariance matrix, and $(\bar{x} - \bar{\mu})^T$ is the transpose of the vector difference.

2.2 Joint, Marginal, and Conditional Gaussian Probability Distributions

Gaussian distributions have a great advantage: joints, marginals, and conditionals are also Gaussian. Recall that a *joint probability distribution* involves two or more variables, that may be dependent or independent from each other. Consider two

[2] Wolfram's Mathworld provides interesting discussion on mean and variance, at http://mathworld.wolfram.com/Mean.html and http://mathworld.wolfram.com/Variance.html

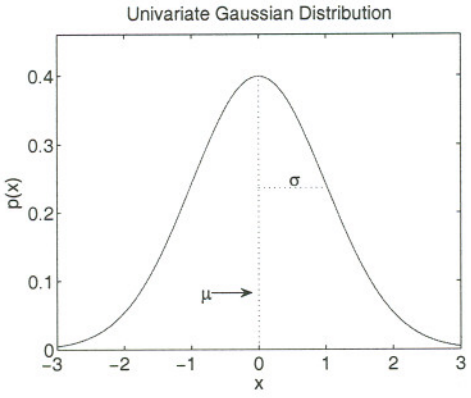


Figure 2.1: A Univariate Gaussian Distribution with $\mu = 0$ and $\sigma^2 = 1$. Notice how the "bell" is centered at the mean. The standard deviation σ is also illustrated.

random variables x and y which are outputs of a given process. The joint distribution tells us the probability of a given *pair* of values $\{x_i, y_i\}$ occurring together [8]. We write joint probability distributions as either

$$p(x, y) = p(x)p(y) \tag{2.6}$$

$$p(x, y) = p(x)p(y|x) \tag{2.7}$$

where the expression in equation 2.6 denotes statistical independence between x and y and equation 2.7 denotes the probability of observing y given the ocurrence of x, a desirable situation for regression, since we can express one variable in terms of the other. Joint Gaussians have the form

$$p(x, y) \equiv \mathcal{N}(\bar{\mu}, \Sigma), \text{with } \bar{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}. \tag{2.8}$$

Each random variable has a contribution in the mean vector and covariance matrix (denoted with subindices in equation 2.8)[3].

Consider the task of regressing x on y, and z. We can easily group y and z as a new variable, R, to manipulate the expressions in convenient block form.

[3]These expressions are frequently extended to more than two variables representing the mean vector and covariance matrix in block form.

Starting with a joint probability distribution of those two random variables, we can obtain a *conditional probability.* Following Bayes Theorem, we obtain the conditional probability as

$$p(x|R) = \frac{p(x,R)}{p(R)} \tag{2.9}$$

We *marginalize* the joint $p(x,y)$ to obtain $p(y)$ by integrating the joint distribution over x. The result is Gaussian, and equivalent to removing the contribution of x. The conditional density is

$$p(x|R) = \mathcal{N}(\mu_{x|R}, \sigma_{x|R}) \tag{2.10}$$

with conditional mean and variance:

$$\mu_{x|R} = \mu_x + \bar{\sigma}_{xR}^T \Sigma_{RR}^{-1}(R - \bar{\mu}_R) \tag{2.11}$$

$$\sigma_{x|R}^2 = \sigma_x^2 - \bar{\sigma}_{xR}^T \Sigma_{RR}^{-1} \bar{\sigma}_{xR}. \tag{2.12}$$

In Equation 2.11 we have a linear expression to estimate x in terms of R. This is a linear regressor.

2.3 Gaussian Mixtures

The Gaussian distribution is not necessarily a good way to estimate the true distribution of the random variables we observe. Observation may come from more than one distribution. Stephenson [9] gives a great motivating example, which we summarize as follows: Suppose we draw the following observations:

$$X = \{-20, -19, -19, -18, 7, 8, 9, 9, 10, 11\}.$$

A Gaussian would use -2.2 as the mean value of this distribution, and the numbers drawn are not even close to that estimate. It is better to have two subsets, $X_1 = \{-20, -19, -19, -18\}$, and $X_2 = \{7, 8, 9, 9, 10, 11\}$, with a Gaussian distribution for each subset. Even better, we can build a distribution over both Gaussians

as a weighted sum. The weights would be $\frac{4}{10}$ and $\frac{6}{10}$ respectively, corresponding to the fraction of the data points that belong to each Gaussian component. Gaussian mixtures are expressed as

$$p(\bar{x}) = \sum_{i=1}^{c} w_i p_i(\bar{x}|\bar{\mu}_i, \Sigma_i) \tag{2.13}$$

where c is the number of Gaussian components. For each component i, the associated weight, mean, and covariance matrix are w_i, $\bar{\mu}_i$, and Σ_i respectively. The sum of all weights must be unity ($\sum_{i=1}^{c} w_i = 1$). In Section 2.2 we derived an expression for a conditional Gaussian distribution. Following the same rationale, we can write an expression for a *conditional Gaussian Mixture*. The final expression is

$$p(x|y) = \frac{1}{p(y)} \sum_{i=1}^{c} w_i p_i(y) p_i(x|y) \tag{2.14}$$

where every item indexed by i corresponds to a Gaussian component. There is a closed form expression for the expected value of the conditional,

$$\begin{aligned} E[x|y] &= \frac{1}{p(y)} \sum_{i=1}^{c} w_i p_i(y) E_i[x|y] \\ &= \sum_{i=1}^{c} p(i|y) E_i[x|y] \end{aligned} \tag{2.15}$$

which is a regressor function for x with input y.

2.4 Fitting the model parameters

When working with Gaussian Mixtures, one faces the task of fitting the model parameters (weights, means, and covariances) as well as deciding the number of components to use. We used the *Expectation-Maximization* (EM) algorithm [5] to fit the weights, means, and covariances, and *cross-validation* [7] to determine an appropriate number of Gaussian components.

The Expectation-Maximization (EM) algorithm (Dempster, 1977 [5]) is generally used to fit the parameters of Gaussian Mixtures, as shown in a popular tutorial by Bilmes [4]. Dempster et al. [5], and later Xu and Jordan [10] provide proof of convergence to local optima of this algorithm. Recall that we fit the parameters of a proposed model to minimize a measurement of error. For EM, we find parameters that maximize *likelihood* of the data under the proposed distribution. Think of this as "maximizing the benefit", comparable to "minimizing the error." In summary, this algorithm iterates over two steps: the E-step and the M-step. We start with a random selection of parameters (weights, means, covariances) for a mixture of c components. The E-Step computes the expected complete-data likelihood values as a function of the current proposed set of parameters given the previous ones used. The M-Step finds new values of parameters that maximize the expected complete-data likelihood function. The two steps are repeated until convergence according to a given criteria.

We used cross-validation [7] to determine the number of Gaussian components. First, we separated the available data in two disjoint sets: *fitting* ($\mathcal{F}$) and *hold-out* ($\mathcal{H}$). Then, we proposed a range of the number of components to consider (2,3,...,10). For each choice of the number of components, we used EM to fit a Gaussian Mixture[4] with the set $\mathcal{F}$. For every Mixture, we evaluated its performance on the set $\mathcal{H}$. We were then able to decide which number of components was best suited for our data by looking at the mean square error on the set $\mathcal{H}$.

Suppose we have modelled the joint density of two variables, x and y as a Gaussian Mixture. If we wish to regress x in terms of y, we must find the conditional density $p(x|y)$. This process is detailed in Chapter 3, where we show how we built a regressor for salinity..

[4]EM converges to local optima. It is wise to use several restarts and pick the parameters that yield the best performance (i,e., minimize the MSE of the mixture evaluated on the set $\mathcal{F}$.)

Chapter 3

Building a Regressor for the CORIE data

This chapter reviews the previous work by Archer et al. [1, 2], presents a summary of the characteristics of the problem we try to solve, and details our proposed regression mechanism using the Gaussian Mixture Model.

3.1 Bio-fouling

As part of an environmental observation and forecasting system, sensors deployed in the Columbia River estuary (CORIE)[1] collect salinity, pressure, temperature, and velocity measurements. Figure 3.1 shows stations deployed in the estuary. Measurements have been archived since 1996. The CORIE team uses the measurements to gain a better understanding of the estuary. Of the sensors deployed, salinity sensors are subject to bio-fouling, an event that causes a decay of the measured maximum daily salinity. Bio-fouling is caused by the growth of biological material on the sensor. A CORIE expert will take a look at the salinity measurements and identify a monotonic decay of the maximum diurnal salinity measurement. The expert will then estimate when bio-fouling started and discard the corrupted data. Figure 3.2 shows clean and bio-fouled timeseries.

[1]For up to date information on the CORIE project, visit www.ccalmr.ogi.edu/CORIE/

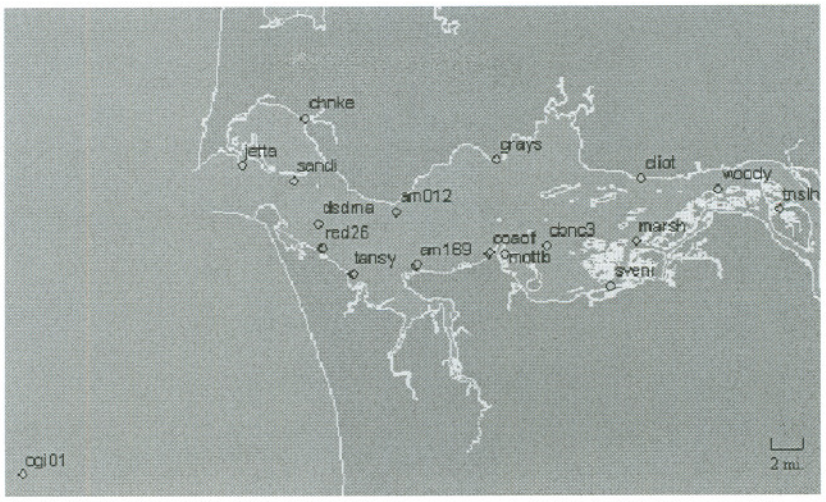


Figure 3.1: CORIE Stations map. Stations are marked with circles. We refer to stations in this text by their name.

Detection of bio-fouling can take weeks or months. Degradation can either be a very slow process or occur within a couple of weeks. It is not until a sensor is substantially compromised that the experts start analyzing the timeseries. Even then, determining the day at which bio-fouling starts is an uncertain estimation. We refer to the time at which bio-fouling occurs as *on-set time*.

3.2 Detection of Bio-fouling

In order to detect bio-fouling, Archer et al. [1, 2] looked at sources of correlated information. They found that temperature sensors are not subject to bio-fouling, and assumed there is a correlation between temperature and salinity measurements. This assumption follows from the fact that salinity and temperature at a given station result from the same mixing process of ocean and river waters. They proposed that the measured salinity and temperature at a given station results from a linear mixing of ocean and river waters, expressed as

$$S_m = \alpha(t)S_o + (1 - \alpha(t))S_r \tag{3.1}$$

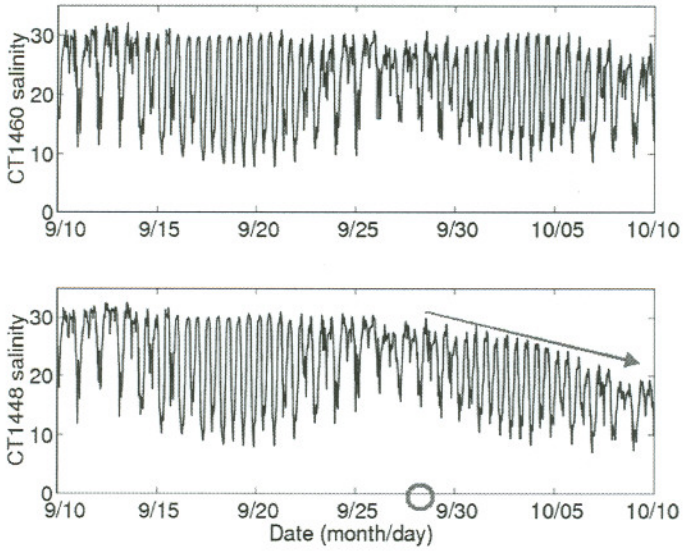


Figure 3.2: Clean and bio-fouled salinity time series examples from sensors in the Columbia River estuary. The top timeseries shows a clean signal, with its natural variability. The bottom plot shows a bio-fouled sequence. Notice how the maximum salinity value is decreasing over time after 9/28, the estimated time at which bio-fouling began. Image from Archer et al.[2]

$$T_m = \alpha(t)T_o + (1 - \alpha(t))T_r. \tag{3.2}$$

In the above formulation, the observed salinity and temperature at a given time t are S_m and T_m, the ocean salinity and temperature are S_o and T_o, and the river salinity and temperature are S_r and T_r. The linear mixing coefficient is $\alpha(t)$. Archer et al. estimated α by solving the temperature equation 3.2

$$\alpha(t) = \frac{T_r - T_m}{T_r - T_o}. \tag{3.3}$$

In upstream river there is almost no salinity penetration, hence equation 3.1 with $S_r = 0$ implies $S_m = \alpha(t)S_o$, i.e., $\alpha(t)$ is well correlated with salinity measurements, which means S_m is a linear function of $\alpha(t)$. With this information, they modeled salinity s and the mixing coefficient α as jointly Gaussian,

$$p(s, \alpha) \equiv \mathcal{N}(\mu, \Sigma) \tag{3.4}$$

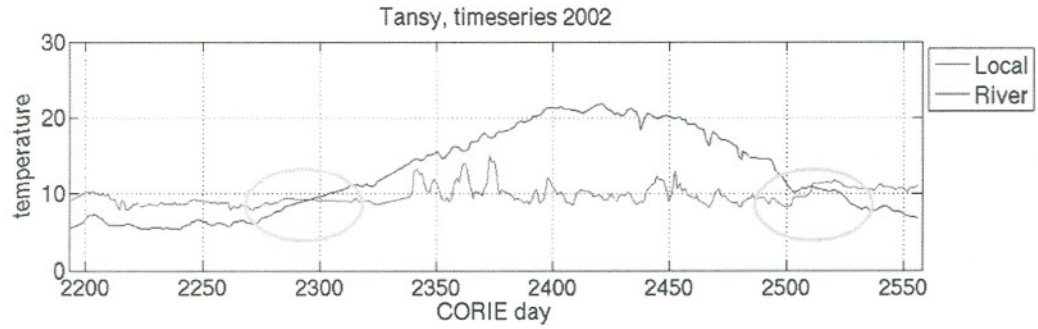


Figure 3.3: Temperature timeseries. Yearly, local and river temperatures cross over during Spring and Fall. This is intuitive since upstream river is warmer during the Summer than during the Winter when compared to the temperature observed mid-estuary.

which can be used to regress salinity on the mixing coefficient as discussed in Chapter 2. At this point, with a regressor that gives an expected value for salinity observing upstream river temperature, ocean temperature, and local temperature, they used a sequential likelihood ratio test to detect bio-fouling [1, 2]. This approach reduced data-loss by half.

Although the detection architecture yielded satisfactory results, we observed that during transition periods (spring and fall) the regressor performed poorly. This follows from the fact that during this transition periods the local and river temperatures cross-over. Figure 3.3 illustrates a year-long timeseries of temperature data. In this research, we looked at improving the overall performance of the regressor, aiming specifically to the performance during transition periods.

3.3 Feature selection

As we mentioned earlier, the clearest indication of bio-fouling is a decrease of maximum daily salinity, Archer et al. [1] describe how to extract this feature. The maximum daily salinity occurs near the tidal flood, a time at which the water depth

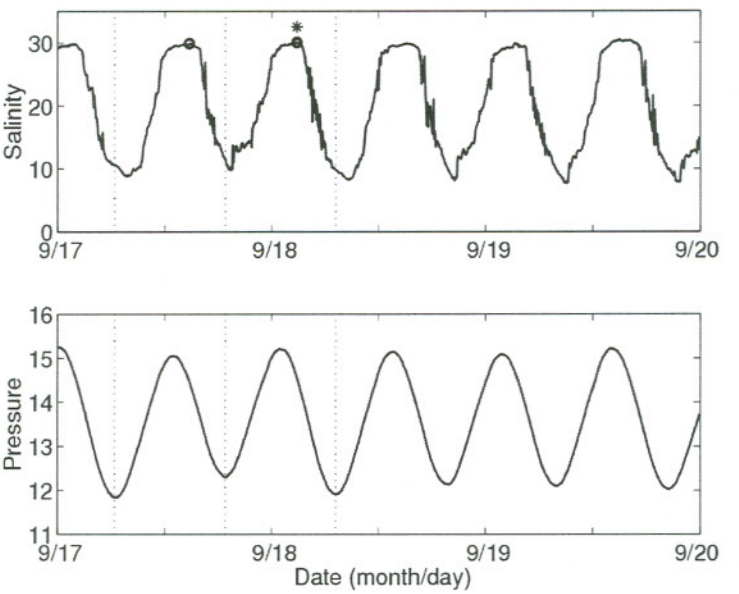


Figure 3.4: Feature selection looking at tidal cycles. There are two tidal cycles per day, identified looking at the pressure signal, which is clean. After identifying the maximum salinity between ebbs, we pick the maximum of those two values as the maximum daily salinity. mage from Archer et al. [1]

is highest. By looking at the pressure signal, which is clean and not subject to bio-fouling, they identify tidal ebbs by simply looking at the lowest points of the pressure signal. Then, they identify the maximum salinity measurement between ebs. In one day, there are two ebbs, so they choose the maximum salinity of those two values. Figure 3.4 shows a sample pressure signal and a salinity timeseries.

Once the maximum daily salinity is extracted, we look at the matching temperature for the time at which the maximum occured. The river temperature is obtained from stations located upstream (usually Eliot or Woody). Ocean temperature is assumed constant at 10.43 Celsius based on observations from stations close to the ocean (usually jetta or sandi).

Local temperature and upstream river temperature were candidate input features

to replace the mixing coefficient originally proposed. Projections of the temperature space are shown in figure 3.5. In addition, we noted that by normalizing the measured and river temperature by the denominator of the mixing coefficient, we gained a natural separation of winter and summer datapoints as shown in Figure 3.6. The normalized temperatures are calculated as

$$T_{mN} = \frac{T_m}{T_o - T_r} \tag{3.5}$$

$$T_{rN} = \frac{T_r}{T_o - T_r} \tag{3.6}$$

where T_{mN} and T_{rN} refer to normalized local and river temperature, T_o is the ocean temperature, T_m is the local temperature, and T_r is the upstream river temperature. We then proceeded to model the joint density

$$p(s, T_m, T_r, T_{mN}, T_{rN}) \tag{3.7}$$

as a Gaussian Mixture. The regressor function after fitting the model parameters, $E[s|T_m, T_r, T_{mN}, T_{rN}]$, has the form of equation 2.15.

Figures 3.5 and 3.6 allow a quick visualization of the temperature spaces. It can be seen, for instance, that Summer points have a different functional form than Winter points. The Gaussian Mixture Model, when properly fitted, should capture these different forms with its various Gaussian components.

We also decided that data points for which the difference between ocean and river temperatures was less or equal to 1.5 Celsius were unnormalizable. This proved useful in experimentation and is discussed in more detail in Chapter 4.

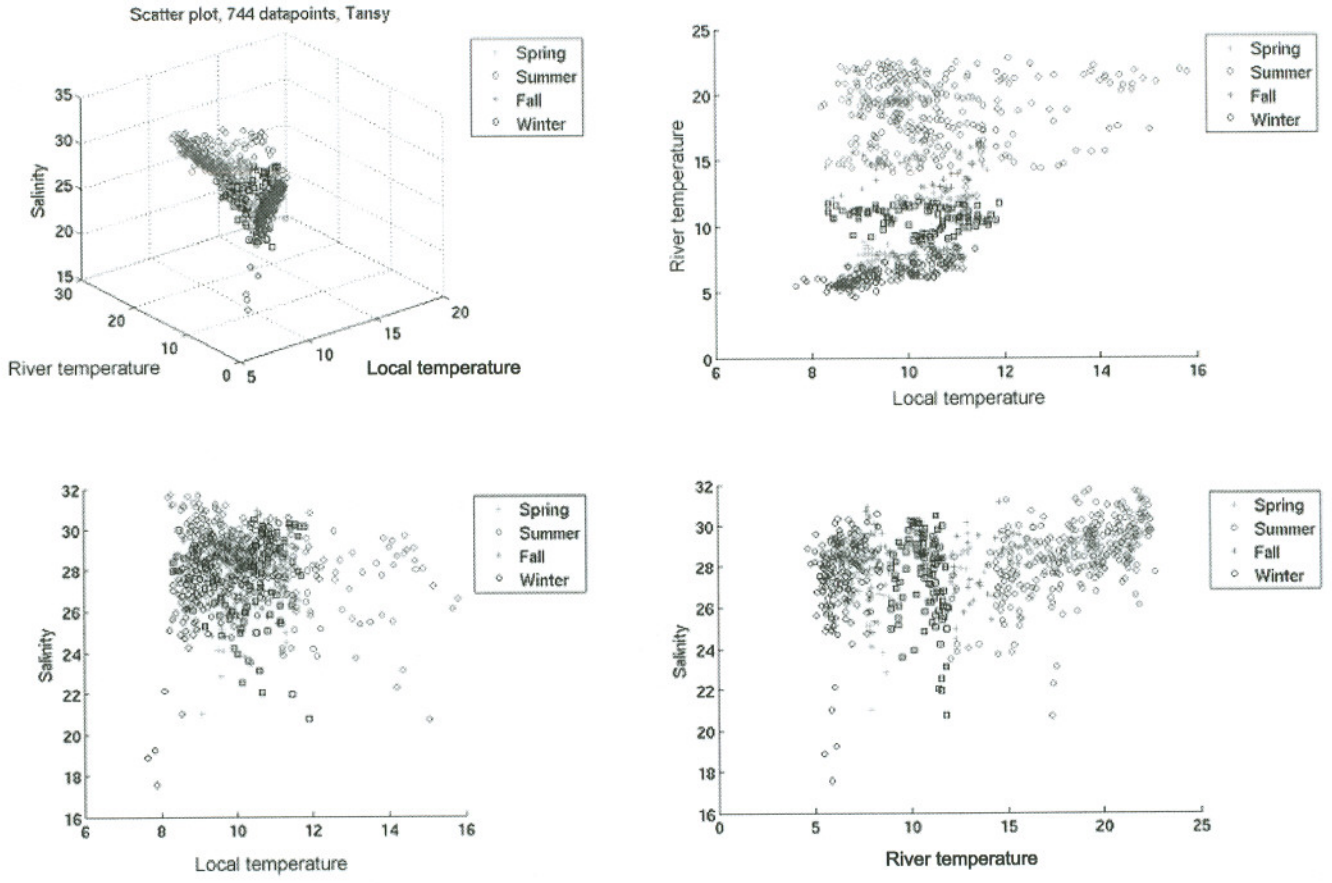


Figure 3.5: Tansy feature space. Notice how different seasons have different functional forms. The unnormalizable points are shown in black squares.

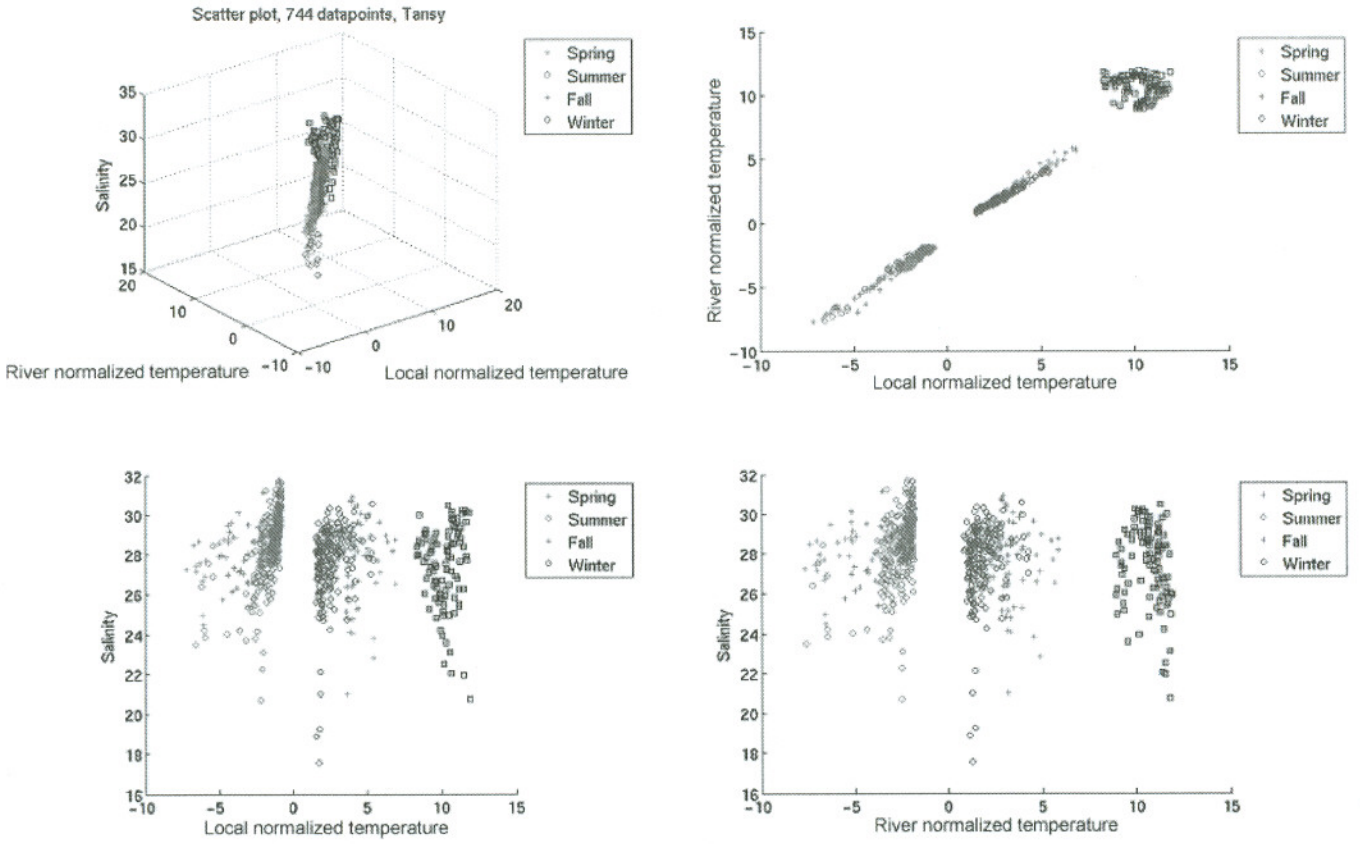


Figure 3.6: Tansy normalized feature space. Notice how the normalized features provide a clear separation. Unnormalizable points are shown inside a black square..

Chapter 4

Experimental results

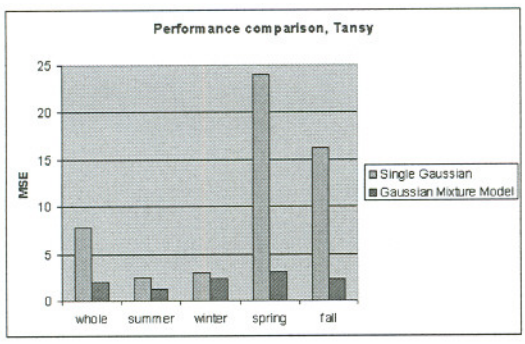
This chapter presents and discusses results obtained using the Gaussian Mixture Model for regression. We compare our performance to the Single Gaussian regressor originally deployed by Archer et al.[1, 2]

We experimented with data from two stations: Tansy and am169. Refer to Figure 3.1 to visualize their location in the estuary. Station am169 has several sensors. We will refer to the sensor located at 11.3m below mean sea level as "am169_middle" and the sensor located at 14.3m below sea level as "am169_bottom".

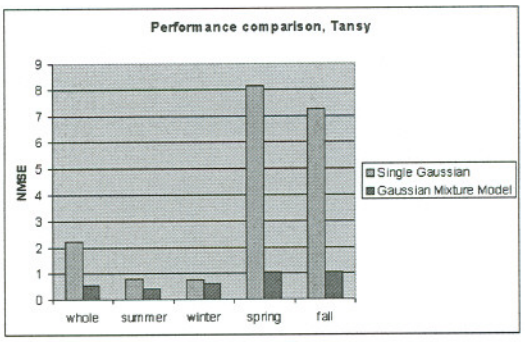
There was enough historical data archived for these stations, and a reasonable number of data points available. Data from 1999 to 2003 was available for Tansy, and data from 2001 to 2003 for am169. We had 744 data points available for Tansy. For am169_middle, we had 533 data points, and for am160_bottom, 586.

We split the data to make sure we had a "full year" for training. This means we covered all four seasons with points from the various years. We produced 10 different shuffles, so that we could estimate the regressor performance. Every shuffle always tried to set aside a full year for training.

In Chapter 3 we discussed the feature selection process. Recall that after normalizing the temperatures we could visually identify, by looking at the datapoints, a clear division between summer, winter, and unnormalizable data points. This motivated us to experiment with four different configurations for our input vectors: (1) Only raw features, (2) Only normalized features, (3) Both raw and normalized,



(a) Comparison on Tansy, MSE



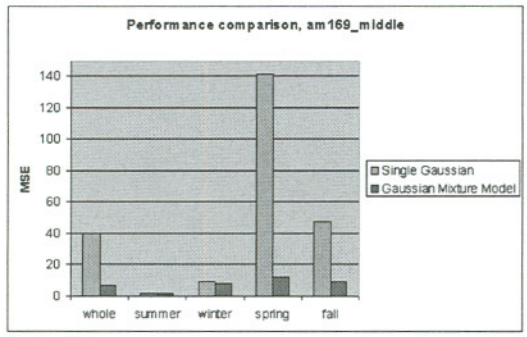
(b) Comparison on Tansy, NMSE

Figure 4.1: Performance comparison of original and new regressor on Tansy. The blue bars represent the error from the original $p(s|\alpha)$ regressor, the red bars represent the error from the new $p(s|T)$ regressor, which uses 10 Gaussian components. Notice the improvement in performance during the transition periods. "NMSE" refers to the normalized mean-square error. Figures show average of ten splits.

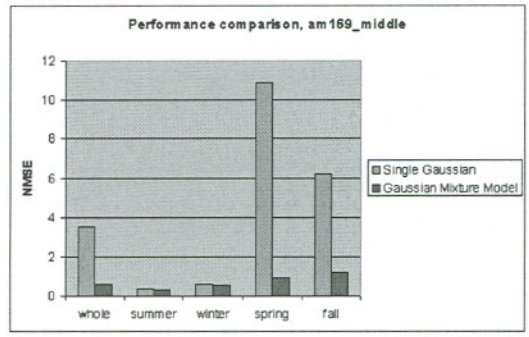
and (4) Both raw and normalized with a hard-rule that separated normalizable from unnormalizable datapoints. Configuration (4) required having two separate architectures, each trained with combination features, but with one architecture devoted exclusively to the input vector of raw features for points that cannot be normalized. Section A.5 in the appendix shows training and test errors for these different choices of inputs.

Results are presented in Figures 4.1–4.3.. The mean square error and the variance normalized mean square error ($nmse$) are shown. Results are averaged over the 10 splits. Significant improvement was observed during the transition periods, which results in an overall performance improvement of the regressor.

In summary, our proposed new regressor which uses local, river, and normalized temperatures as inputs performed considerably better than the original model that used the mixing coefficient as input. The overall performance improves, most notable in the transition periods.

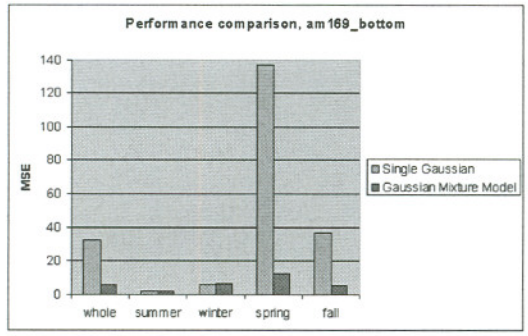


(a) Comparison on am169_middle, MSE

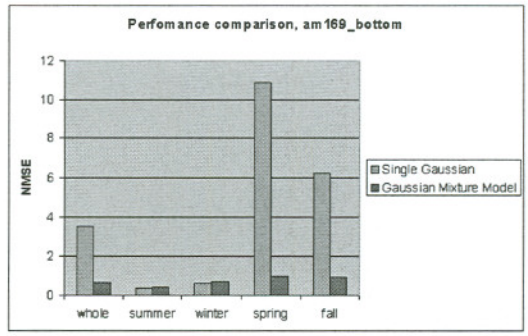


(b) Comparison on am169_middle, NMSE

Figure 4.2: Performance comparison of original and new regressor on am169_middle. The blue bars represent the error from the original $p(s|\alpha)$ regressor, the red bars represent the error from the new $p(s|T)$ regressor, which uses 9 Gaussian components. Notice the improvement in performance during the transition periods. "NMSE" refers to the normalized mean-square error. Figures show average of ten splits.



(a) Comparison on am169_bottom, MSE



(b) Comparison on am169_bottom, NMSE

Figure 4.3: Performance comparison of original and new regressor on am169_bottom. The blue bars represent the error from the original $p(s|\alpha)$ regressor, the red bars represent the error from the new $p(s|T)$ regressor, which uses 7 Gaussian components. Notice the improvement in performance during the transition periods, but in this station we do not observe improvement in Summer and Winter data. "NMSE" refers to the normalized mean-square error. Figures show average of ten splits.

Chapter 5

Conclusion, Related, and Future Work

We observed that the selection of new input features improved the performance of the salinity regressor. Using a nonlinear model (Gaussian Mixture Model) helped identify different functional forms for the feature space. We observed significant improvement during transition periods, which yielded overall performance enhancement when compared to the original regressor.

Parallel to this research, Haiming Zheng experimented with the Mixture of Experts model. This model uses a gating function (neural network) to direct inputs to several linear components. The Gaussian Mixture Model performed comparably to this approach, which is currently deployed in the CORIE project.

After trying several regression approaches and observing similar results (refer to the Appendix), it is reasonable to assume that we have achieved the best possible performance with the input features we selected.

Future work may explore the use of additional input features, such as windforcings. It may also be possible to incorporate information from nearby stations. It may also be convenient to extend the current mixing process assumption.

Bibliography

[1] Archer, C., Baptista, A., and Leen, T. K. Fault Detection for Salinity Sensors in the Columbia Estuary. *Water Resources Research, 39(3)* 10.1029/2002WR001376, 2003.

[2] Archer, C., Leen, T. K., and Baptista, A. Parameterized Novelty Detection for Environmental Sensor Monitoring. *Advances in Neural Information Processing Systems 16*, pp. 619-624, 2004.

[3] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[4] Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gausian Mixture and Hidden Markov Models. *Technical Report*, UC Berkeley. ICSI-TR-97-021, 1997.

[5] Dempster, S.P., Laird, N.M., and Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm *Journal of the Royal Statistical Society, Series B, 39.* pp.1–38, 1977.

[6] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification* (2nd Edition). Wiley-Interscience, 2001.

[7] Mitchell, T. M. *Machine Learning.* McGraw-Hill, 1997.

[8] Rozanov, Y. A. *Probability Theory: A Concise Course.* Dover, 1969.

[9] Stephenson, T. A. Conditional Gaussian Mixtures. *Technical Report*, Institute for Perceptual Artificial Intelligence, Valais, Switzerland. IDIAP-RR 03-11, 2003.

[10] Xu, L. and Jordan, M. I. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural Computation 8*, pp. 129-151, 1996.

Appendix A

Additional Approaches

As described before, the goal of this research is to improve the performance of the maximum daily salinity regressor. We tried several approaches, documented in this appendix.

A.1 Use of historical temperature information

When we looked at historical timeseries of maximum daily salinity, we concluded it was a reasonable assumption to expect historical temperature information from past days to help improve the predictor's performance. We decided to conduct a pilot study and build linear regressors using current day flood and ebb temperatures, as well as previous days' information. We built several linear regressors of this form:

$$\hat{S}_1(T_f^0, T_e^0)$$
$$\hat{S}_3(T_f^{-2}, T_e^{-2}, T_f^{-1}, T_e^{-1}, T_f^0, T_e^0)$$
$$\hat{S}_5(T_f^{-4}, T_e^{-4}, T_f^{-3}, T_e^{-3}, T_f^{-2}, T_e^{-2}, T_f^{-1}, T_e^{-1}, T_f^0, T_e^0)$$

Regressor $\hat{S}_1$ uses current day information, regressor $\hat{S}_3$ incorporates the previous two days' measurements, and so on. We trained on Tansy data from Summer 2002 and tested on Summer 2003. We observed no significant improvement when adding historical information. Table A.1 shows the performance of the regressors.

Figure A.1 shows timeseries of measured maximum daily salinity and predicted values according to regressors $\hat{S}_1$, $\hat{S}_3$, and $\hat{S}_5$.

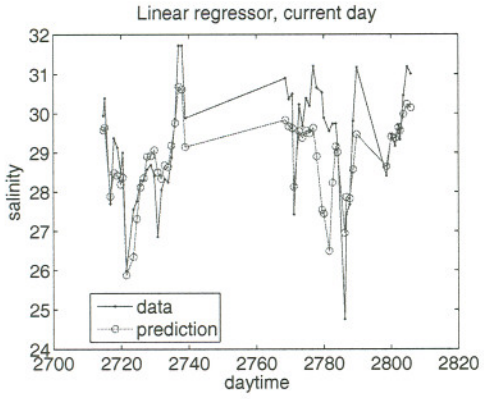
Table A.1: Performance of linear regressors using historical data. Notice how there is no significant improvement when adding information from previous days. The measure of performance is the Mean Square Error.

Regressor	MSE
$\hat{S}_1$	1.1248
$\hat{S}_2$	1.0913
$\hat{S}_3$	1.0886
$\hat{S}_4$	1.0700
$\hat{S}_5$	1.1041

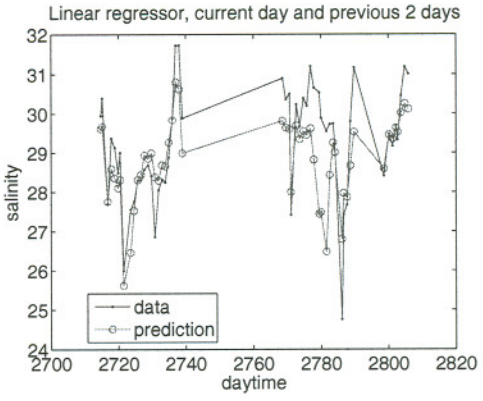
We were convinced that there was no useful information in historical data by examining the linear model's parameters. The values assigned to m were more significant, in all cases, for the current day. Table A.2 shows the model parameters for regressors $\hat{S}_1$, $\hat{S}_3$, and $\hat{S}_5$.

Table A.2: Model parameters for historical data linear regressors. Vector m contains the weights, b is the constant. Notice how the last two entries of m (corresponding to the current day) are significantly greater than the others.

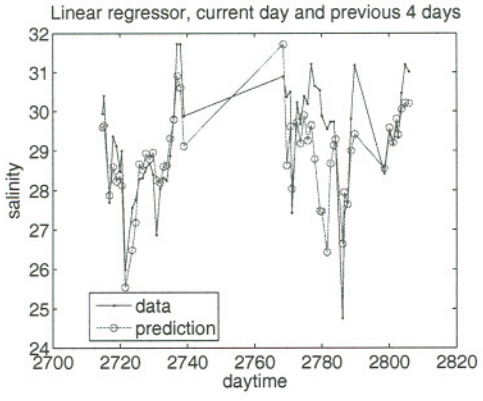
Regressor	Model Parameters
$\hat{S}_1$	$m = (10.1, -11.4)'$, $b = -18.6$
$\hat{S}_2$	$m = (-0.9, 0.4, 10.8, -11.2)'$, $b = -19.2$
$\hat{S}_3$	$m = (0.1, -1.5, -1.1, 1.9, 10.9, -11.4)'$, $b = -19.1$
$\hat{S}_4$	$m = (-1.2, -0.6, 1.1, 0.5, -1.3, 0.6, 10.7, -10.5)'$, $b = -20.2$
$\hat{S}_5$	$m = (1.3, -2.2, -2.3, 1.2, 1.2, 0.8, -1.3, -0.4, 10.9, -10.2)'$, $b = -19.2$



(a) $\hat{S}_1$ linear regressor



(b) $\hat{S}_3$ linear regressor



(c) $\hat{S}_5$ linear regressor

Figure A.1: Prediction using historical information. The timeseries reveal that there is no significant improvement when incorporating historical information.

A.2 Incorporating SELFE salinity numerical prediction

The CORIE team has developed a numerical model of the estuary called "SELFE', which simulates the dynamics of the river. We considered incorporating the SELFE salinity predictions as an independent observation to our maximum daily salinity regressor. We used the same criteria applied to measurement time series to determine tidal cycles.

Unfortunately, the data from SELFE did not help improve the regressor performance. The salinity predictions from SELFE are not as reliable as we expected at this point. We believe the numerical predictions might be useful as the CORIE team continues to make progress in their system.

A.3 Neural Network based regressor

As we described earlier, the transition periods represent a challenge for the regressor. It was a reasonable complementary experiment to use Neural Networks to see if this non-linear architecture could better pick up the seasonality changes.

We set up pilot studies using two architectures: 30 hidden nodes and 50 hidden nodes, both single-layered, searching for an optimal regularizer α. We were trying to beat the performance of the Mixture of Experts (ME) model (mentioned in Chapter 5.) Results were comparable but not superior to the ME regressor, a situation that discouraged us from further pursuing this approach. Further exploration of different architectures and regularizer parameter values would have been too time-consuming, and the Gaussian Mixture Model had already performed better than the original Single Gaussian regressor. Table A.3 shows training and test errors on data from Tansy station using the ME model based regressor. Table A.4 shows training and test errors on the same data using the NN based regressor.

Table A.3: Mixture of Experts performance on Tansy data. Results are averaged over 10 splits of the available data. Performance is presented in terms of mean square error (*mse*) and variance normalized mean square error (*nmse*).

	whole	summer	winter	spring	fall
training-mse	1.38	0.86	1.37	2.39	1.78
training-nmse	0.44	0.29	0.47	0.83	0.91
test-mse	1.81	1.21	2.07	2.75	2.11
test-nmse	0.51	0.38	0.54	0.94	0.95

Table A.4: Neural Network performance on Tansy data. The network had 50 hidden inputs, and was trained for 10,000 iterations with the regularizer $\alpha = 1.4$. This setting that yielded the best performance for this station. Results are averaged over 10 splits of the available data. Performance is presented in terms of mean square error (*mse*) and variance normalized mean square error (*nmse*).

	whole	summer	winter	spring	fall
training-mse	1.42	0.94	1.46	2.34	1.86
training-nmse	0.46	0.32	0.50	0.81	0.95
test-mse	1.86	1.29	2.14	2.78	2.21
test-nmse	0.53	0.41	0.56	0.95	0.99

The Neural Network regressor performed comparably to the Mixture of Experts regressor. Transition periods (spring and fall) are still difficult. Notice how the normalized mean square error for both approaches is close to unity, which means the regressors are performing just as well as predicting the mean value for those periods.

A.4 Incorporating salinity information from nearby stations

We considered the possibility of looking at salinity measurements from nearby stations to either improve the regressor performance or provide additional information to alert the CORIE staff of potential failures. Consider the following scenario: Three stations (A, B, and C) are located close to each other. We have temperature based regressors for each station. We wish to also regress salinity on the measurements from the other stations. This setting is detailed in Figure A.2. Every connection in the graph represents an available and reliable regressor. We can visualize the information building a matrix as the one shown in Table A.5.

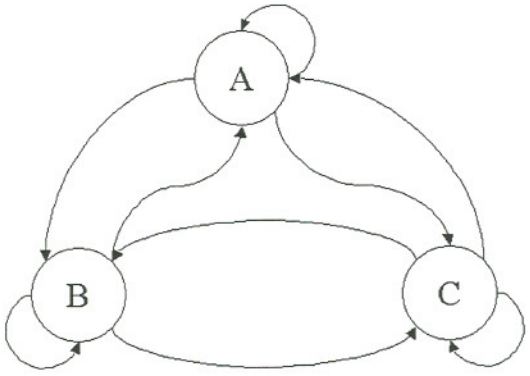


Figure A.2: Diagram of nearby station relations. Connections represent a reliable source of information. Self-links represent the temperature based regressor.

Suppose no station bio-fouls, and assume salinity at nearby stations provides useful information for regression. We can build regressors incorporating salinity measurements from other stations. For station A, considering stations B and C as relevant sources of information:

$$\hat{S}_A(T_A, S_B, S_C) \tag{A.1}$$

Suppose we are trying to predict station A's salinity and only station C bio-fouls.

Table A.5: Matrix of regressors from nearby stations. The matrix represents a connected graph, where every entry corresponds to a connection. In a fully-connected graph, we have regressors for all stations based on local temperature measurements and other stations salinity measurements. The diagonal elements are the temperature based regressors. $\hat{S}_X$ is the expected salinity for station X, S_X is the measured salinity, and T_X is the vector of temperature measurements (local and upstream river.)

	A	B	C
A	$\hat{S}_A(T_A)$	$\hat{S}_A(S_B)$	$\hat{S}_A(S_C)$
B	$\hat{S}_B(S_A)$	$\hat{S}_B(T_B)$	$\hat{S}_B(S_C)$
C	$\hat{S}_C(S_A)$	$\hat{S}_C(S_B)$	$\hat{S}_C(T_C)$

First, we face the problem of determining if C is no longer reliable. Assuming a detector based on $\hat{S}_C(T_C)$ has fired an alarm, and detectors based on $\hat{S}_C(S_A)$ and $\hat{S}_C(S_B)$ agree with the alarm, a given detection system would have to modify the connectivity of the graph and discard information from station C. The result of this decision is shown in Figure A.3 and Table A.6.

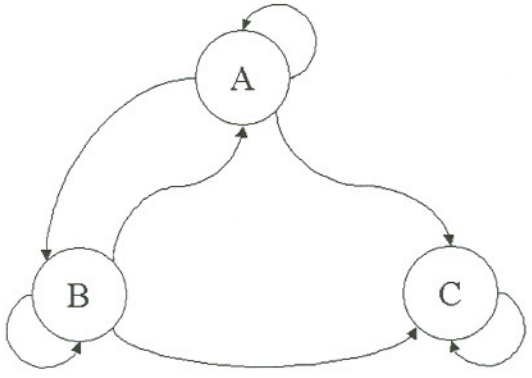


Figure A.3: Diagram of nearby station relations after station C bio-fouls. Connections from station C are removed.

What happens if stations B and C bio-foul at the same time? Likely, regressors $\hat{S}_B(S_C)$ and $\hat{S}_C(S_B)$ cannot be trusted to provide reliable information to the detectors, a situation that provides additional difficulty for the system to determine

Table A.6: Matrix of regressors from nearby stations after station C bio-fouls. Regressors based on S_C are removed from the detection system.

	A	B	C
A	$\hat{S}_A(T_A)$	$\hat{S}_A(S_B)$	
B	$\hat{S}_B(S_A)$	$\hat{S}_B(T_B)$	
C	$\hat{S}_C(S_A)$	$\hat{S}_C(S_B)$	$\hat{S}_C(T_C)$

which stations should be dropped from the connection graph.

We think it is not viable to further pursue the use of salinity measurements from nearby stations, since their salinity measurements are also unreliable, and a system that attempts to work around possible scenarios would end up being a collection of special ad-hoc conditions.

A.5 Comparison of features in Gaussian Mixture Model

This section includes a comparison of performance when using: (1) local measurements, (2) normalized features, (3) combination features, and (4) combination features with a hard-rule. We observed comparable performance when using combination features or combination features with a hard-rule separating unnormalizable points. When using a hard-rule, the performance during summer and winter was slightly worse.

Table A.7: Gaussian Mixture Model on Tansy, 8 components, raw features only

	whole	summer	winter	spring	fall
train-mse	1.44	0.93	1.33	2.63	1.73
train-nmse	0.46	0.31	0.45	0.91	0.88
test-mse	1.92	1.32	2.10	3.14	1.97
test-nmse	0.55	0.42	0.55	1.01	0.88

Table A.8: Gaussian Mixture Model on Tansy, 6 components, normalized features only

	whole	summer	winter	spring	fall
train-mse	1.52	1.01	1.59	2.42	1.82
train-nmse	0.49	0.34	0.54	0.84	0.93
test-mse	2.08	1.35	2.43	3.25	2.21
test-nmse	0.59	0.43	0.63	1.11	0.99

Table A.9: Gaussian Mixture Model on Tansy, 10 components, combination features

	whole	summer	winter	spring	fall
train-mse	1.37	0.87	1.31	2.34	1.87
train-nmse	0.44	0.29	0.45	0.81	0.95
test-mse	2.01	1.26	2.40	3.08	2.31
test-nmse	0.57	0.40	0.62	1.05	1.04

Table A.10: Gaussian Mixture Model on Tansy, 4 components for the Normalizable points, 2 components for the Unnormalizable points, combination features with hard rule

	whole	summer	winter	spring	fall
train-mse	1.18	0.80	1.29	2.39	1.54
train-nmse	0.38	0.27	0.44	0.83	0.79
test-mse	1.76	1.06	2.43	3.01	1.86
test-nmse	0.49	0.34	0.63	1.02	0.83

Table A.11: Gaussian Mixture Model on am169_middle, 11 components, raw features only

	whole	summer	winter	spring	fall
train-mse	5.38	1.57	6.74	9.18	6.51
train-nmse	0.47	0.29	0.50	0.62	0.88
test-mse	6.81	1.49	9.86	10.31	9.08
test-nmse	0.60	0.29	0.67	0.79	1.19

Table A.12: Gaussian Mixture Model on am169_middle, 7 components, normalized features only

	whole	summer	winter	spring	fall
train-mse	5.39	1.98	6.93	8.29	6.48
train-nmse	0.47	0.36	0.52	0.56	0.88
test-mse	7.03	1.90	7.59	12.21	10.36
test-nmse	0.62	0.38	0.52	0.94	1.36

Table A.13: Gaussian Mixture Model on am169_middle, 9 components, combination features

	whole	summer	winter	spring	fall
train-mse	5.34	1.86	6.72	8.36	6.73
train-nmse	0.47	0.39	0.50	0.57	0.92
test-mse	6.83	1.76	8.03	11.88	9.38
test-nmse	0.60	0.35	0.55	0.91	1.23

Table A.14: Gaussian Mixture Model on am169_middle, 4 components for the Normalizable points, 7 components for the Unnormalizable points, combination features with hard rule

	whole	summer	winter	spring	fall
train-mse	4.76	1.91	6.52	6.97	6.58
train-nmse	0.42	0.35	0.49	0.47	0.89
test-mse	6.22	1.77	8.45	12.51	9.22
test-nmse	0.55	0.35	0.58	0.96	1.21

Table A.15: Gaussian Mixture Model on am169_bottom, 9 components, raw features only

	whole	summer	winter	spring	fall
train-mse	5.26	2.03	6.18	10.65	4.78
train-nmse	0.55	0.34	0.59	0.85	0.90
test-mse	5.99	2.11	6.55	12.89	6.14
test-nmse	0.66	0.35	0.71	1.02	1.04

Table A.16: Gaussian Mixture Model on am169_bottom, 6 components, normalized features only

	whole	summer	winter	spring	fall
train-mse	4.77	2.22	6.45	7.70	4.29
train-nmse	0.50	0.38	0.62	0.62	0.81
test-mse	5.64	2.61	6.41	11.02	4.92
test-nmse	0.61	0.44	0.69	0.87	0.84

Table A.17: Gaussian Mixture Model on am169_bottom, 7 components, combination features

	whole	summer	winter	spring	fall
train-mse	4.41	2.03	5.65	7.58	4.05
train-nmse	0.46	0.35	0.54	0.60	0.77
test-mse	5.88	2.47	6.27	12.54	5.33
test-nmse	0.64	0.41	0.68	0.99	0.91

Table A.18: Gaussian Mixture Model on am169_bottom, 6 components for the Normalizable points, 7 components for the Unnormalizable points, combination features with hard rule

	whole	summer	winter	spring	fall
train-mse	3.85	2.21	5.50	6.16	2.97
train-nmse	0.40	0.37	0.52	0.49	0.56
test-mse	4.96	2.51	6.48	11.13	4.35
test-nmse	0.54	0.42	0.70	0.88	0.74

Biographical Note

Rafael de Jesús Fernández Moctezuma was born in San Luis Potosí, SLP, México, on February 15, 1979. He obtained his B.S. in Computer Systems Engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) Campus San Luis Potosí in 2001. As a full-time undergraduate student, he joined the Campus' Dirección de Informática (IT Department), part-time, from 1997 to 1998. He also worked as a part-time assistant for the Departamento de Computación (Department of Computer Science), setting up the Linux-based architecture of the Computer Network Laboratory from 1999 to 2001. He graduated with honors, and was additionally recognized for leadership in extra-curricular activities.

In 2001, he joined the Applied Mathematics and Computer Science Division at the Instituto Potosino de Investigación Científica y Tecnológica (IPICyT), a public research center funded by the Consejo Nacional de Ciencia y Tecnología (CONACyT), México. His work at IPICyT included support for the Division's IT infrastructure, as well as writing articles that popularize science. He was awarded a full scholarship for graduate studies by CONACyT in 2003.

His areas of interest include topics in applied Machine Learning, Knowledge Discovery in Databases, and Data Mining.