# EECE 5644 Homework 2s

Emily Costa (costa.em@northeastern.edu)

March 14, 2021

# 1 Maximum Likelihood and Maximum A Posteriori 3rd Order Linear Regression Estimators

First, I derived the two estimators, Maximum Likelihood (MLE) and Maximum A Posteriori (MAP). I assume that scalar-real y and two-dimensional read vector x are related to each other according to $y = c(x, w) + v$, where

$$v \sim \mathcal{N}(0,\, \sigma^2),$$

$$w = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \end{bmatrix},$$

$$\phi(x_n) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1^3 & x_2^3 \end{bmatrix}.$$

I also assume a data set $D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ with $N$ sample of $(x, y)$ pairs where the samples are independent and identically distributed according to the model. To implement the MLE estimate for a Gaussian distribution, I derived the following estimator expression:

$$\mathbf{arg\,max}_w \prod_{n=1}^{N} p(t|\phi(x_n), w, \beta) = \mathbf{arg\,max}_w \prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{t_n - w^T\phi(x_n)}{\sigma}\right)^2\right), \text{where}$$

$$\beta = 1/\sigma^2$$

Next, I take the natural log of the function in order to reduce it. Additionally, I drop the constants, which are unnecessary the finding the maximum. I did not show this step for brevity.

$$\mathbf{arg\,max}_w \ln \prod_{n=1}^{N} p(t|\phi(x_n), w, \beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(w), \text{where}$$

$$\beta = 1/\sigma^2,$$

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2, \text{which is squared error.}$$

Finally, we set the gradient to zero and solve for the w vector to calculate the maximum probability.

$$\hat{w}_{ML} = 0 = (\phi^T\phi)^{-1}\phi^T\hat{t}$$

To derive the MAP estimate for a Gaussian distribution, it is similar to MLE except we use a posterior distribution instead of only likelihood.

$$\mathbf{arg\,max}_w \prod_{n=1}^{N} p(t|\phi(x_n), w, \beta) * p(\phi(x_n), w, \beta)$$

After completing a similar derivation and setting the gradient to zero, we get the following expression (note: $\lambda$ is the $MxM$ matrix with values equal to gamma, $\gamma$):

$$\hat{w}_{MAP} = 0 = \mathbf{arg\,max}_w -\frac{\lambda}{2}w^Tw - \beta E_D(w) = (\lambda I + \phi^T\phi)^{-1}\phi^T\hat{t}$$

$$\beta = 1/\sigma^2,$$

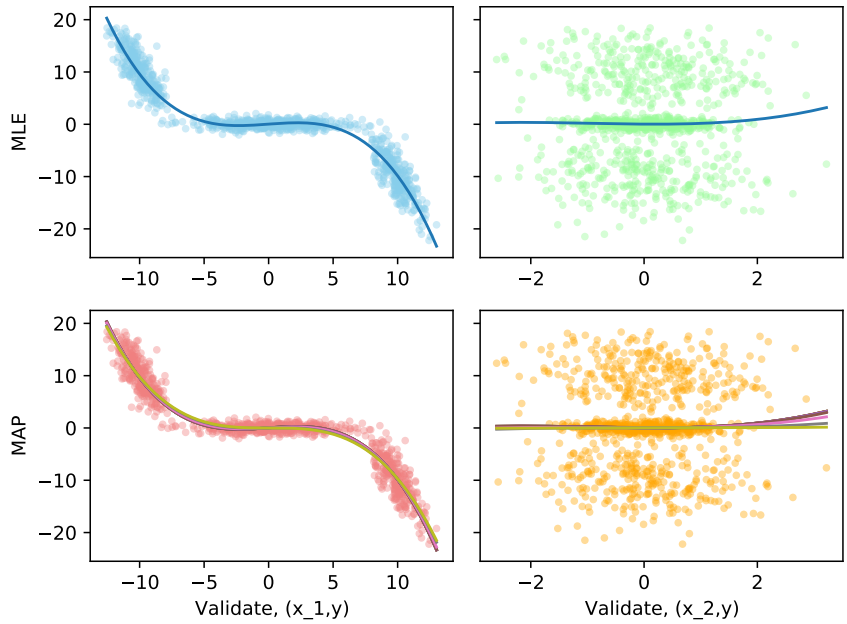


Figure 1: *The 1,000 samples from the validation data set scattered. The left plots are the first x dimension and the right plots are the second x dimension. The top plots have the linear regression lines created by the Maximum-Likelihood Estimator (MLE) and the bottom plots have the linear regression lines created by the Maximum A Posteriori (MAP) using varying gammas between* $10^{-4}$ *and* $10^{4}$.

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2, \text{which is squared error.}$$

Next, I implemented my derived estimator expressions. I used the test and validate data sets generated by the provided Python code to, respectively, obtain and evaluate the estimators. Figure 1 shows the samples from the validation data set along with the regression lines fitted using the two estimators. The mean squared error for the MLE-trained model is 5.2126. Table 1 shows how the mean squared errors changed for the MAP-trained model as gamma was varied. We observe that the MLE-trained model performed as poorly as the worst performing MAP-trained model. This is because the MLE estimate assumes a uniform prior, which is equivalent to a gamma of 0. Hence, as gamma approaches 0 for the MAP estimate, the performance becomes similar to the MLE estimate. Additionally, we observe that a higher gamma results in a better performing MAP-trained model. This is because regularization to the prior is increased.

## 2 Theoretically Optimal Classifier and Varying Training Data Set Sizes with Maximum Likelihood Parameter Estimation

For this section, I generated 4 data sets of samples from the specified distributions.

| MAP-trained Model Performance | |
|---|---|
| $\gamma$ | Mean Squared Error |
| $10^{-4}$ | 5.2126 |
| $10^{-3}$ | 5.2126 |
| $10^{-2}$ | 5.2119 |
| $10^{-1}$ | 5.2054 |
| $10^{0}$ | 5.1436 |
| $10^{1}$ | 4.7543 |
| $10^{2}$ | 4.2140 |
| $10^{3}$ | 4.0819 |
| $10^{4}$ | 4.0909 |

Table 1: How gamma, $\gamma$, affects the performance of the MAP-trained model.

## 2.A Theoretically Optimal Classifier that Achieves Minimum Probability of Error

The following is the classification rule that I used:

$$\frac{p(x|L=1)}{p(x|L=0)} = \frac{g(x|m_1, C_1)}{p(x|m_0, C_0)} > \gamma = \frac{p(x|L=0)}{p(x|L=1)} * \frac{\lambda_{01} - \lambda_{00}}{\lambda_{10} - \lambda_{11}}$$

where the threshold $\gamma$ is a function of class priors and fixed (nonnegative) loss values for each of the four cases $D = i|L = j$ where D is the decision label that is either 0 or 1, like L. $\lambda_{ij}$ is defined in Table 2. $\lambda_{ij}$ values range from 0 to 1, where 1 is the highest cost. In order to minimize expected risk, the costs for incorrect results (false negatives and positives), $\lambda_{01}$ and $\lambda_{10}$, are set to the highest cost possible, 1, while the costs for correct results (true negatives and positives), $\lambda_{00}$ and $\lambda_{11}$, are set to the lowest cost possible, 0.

$$\therefore \gamma = \frac{0.65}{0.35} * \frac{1-0}{1-0} = 1.857$$

$$\therefore \frac{p(x|L=1)}{p(x|L=0)} > 1.857$$

| Conditional Loss Function | |
|---|---|
| $\lambda_{ij}$, where $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$ | Cost Type |
| $\lambda_{00}$ | True negative |
| $\lambda_{01}$ | False negative |
| $\lambda_{10}$ | False positive |
| $\lambda_{11}$ | True positive |

Table 2: The type of cost given by the conditional loss functions, $\lambda_{ij}$, in the likelihood-ratio test.

Once the classifier was implemented and used for the validation data set, the experimental probability of error was 0.35. The ROC curve is shown in Figure 2. Additionally, the incorrectly (in red) and correctly (in green) classified labels are shown in Figure 3.

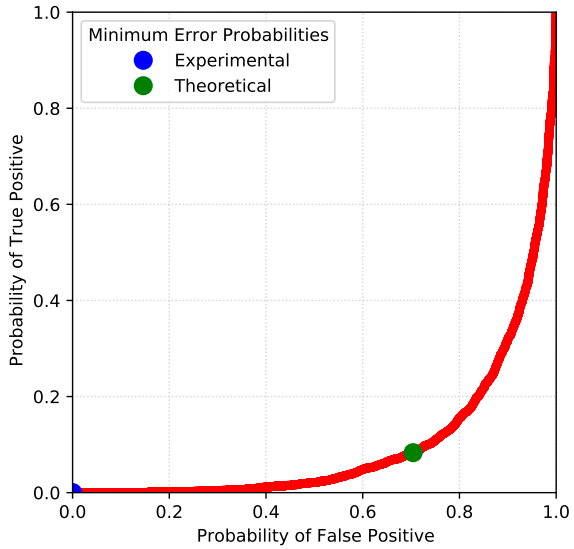


Figure 2: *ROC Curve.*

## 2.B Varying Training Data Set Sizes with Maximum Likelihood Parameter Estimation

### 2.B.a Logistic-Linear-Function-Based Approximations

Next, I used the maximum likelihood parameter estimation technique (MLE) to train three separate logistic-linear-function-based approximations of class label posterior functions. This was implemented similarly to the MLE in Question 1, where I used gradient descent as the numerical optimization approach.

$$\hat{w}_{ML} = 0 = (\phi^T \phi)^{-1} \phi^T \hat{t}$$

$$\hat{w} = \begin{bmatrix} w_0 & w_1 \end{bmatrix},$$

$$\phi(x_n) = \begin{bmatrix} 1 & x \end{bmatrix}.$$

The main difference is I used a 1st order linear regression instead of using a 3rd order linear regression. I then classified based on if the sample validation data was on the correct side of the linear regression. Figure 4 shows how this classifier classified the validation data set based on the various training data set sizes. The probability of errors were not significantly different as the training data sets changed in size. The probability of errors from the plots in Figure 4 from left to right were 0.384, 0.388, and 0.387, respectively.

### 2.B.b Logistic-Quadratic-Function-Based Approximations

To implement this, I simply added additional coefficients to the previous expression, as such:

$$\hat{w}_{ML} = 0 = (\phi^T \phi)^{-1} \phi^T \hat{t}$$

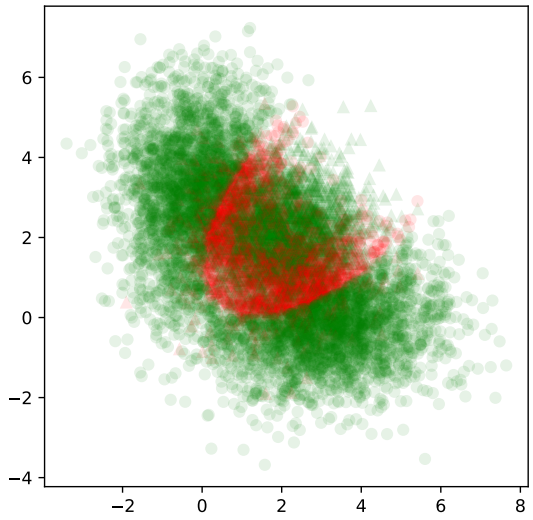


Figure 3: *The classified data set samples.*

$$\hat{w} = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 \end{bmatrix},$$

$$\phi(x_n) = \begin{bmatrix} 1 & x & x^2 & x^3 & x^4 \end{bmatrix}.$$

Figure 5 shows how this classifier classified the validation data set based on the various training data set sizes. The probability of errors were not significantly different as the training data sets changed in size. The probability of errors from the plots in Figure 4 from left to right were 0.280, 0.388, and 0.379, respectively. They did not significantly improve from the previous classifier.

# 3 Categorical Distribution

The ML estimator for $\theta$ is as follows:

$$p(z_n = c|\theta) = \theta_c$$

$$\sum_{c=1}^{k} \theta_c = 1$$

The MAP estimator, assuming that the prior $p(\theta)$ is used for the parameters is a Dirichlet distribution with hyperparameter $\alpha$, is as follows:

$$Cat(\theta)Dir(\alpha) = p(z|\theta)p(\theta|\alpha) = \theta_c \frac{1}{\beta(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k=1} = \frac{\beta(\alpha)}{\beta(\alpha)} Dir(\alpha)$$

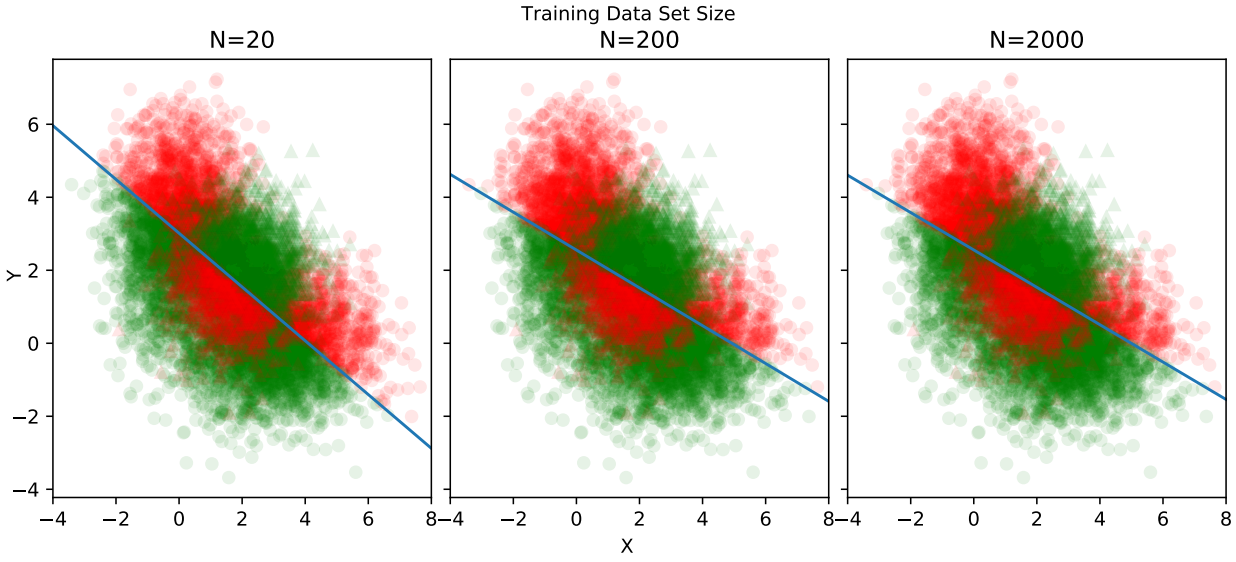


Figure 4: *The classified data sets with MLE classification trained by a Gaussian mixture of size N.*

# 4 Appendix

See my GitHub for all the source code: Click Me!
or copy and paste:

`https://github.com/emilyjcosta5/machine_learning/tree/main/homework_1`

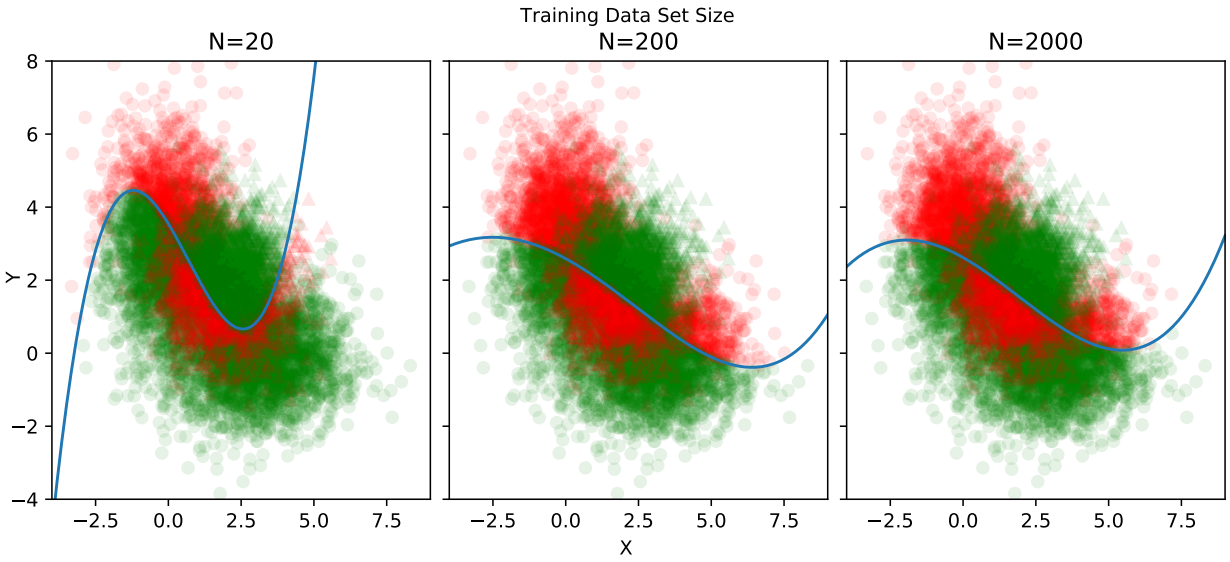


Figure 5: *The classified data sets with MLE classification trained by a Gaussian mixture of size N.*