EECE 5644: Assignment 1

Jeffrey Chen

October 30, 2024

https://github.com/jeffreychen159/EECE-5644-Machine-Learning-and-Pattern-Recognition

---

1 Question 1

Plugging in the parameters of the class-conditional pdf, I generated 4 different samples, D^{20}_{train}, D^{200}_{train}, D^{2000}_{train}, and used them for both part 1 and part 2.

1.1 Part 1

We are given that

$$P(L=0) = 0.6$$
$$P(L=1) = 0.4$$
$$p(x|L=0) = w_{01}g(x|m_{01}, C_{01}) + w02g(x|m02, C_{02})$$
$$p(x|L=1) = w_{11}g(x|m_{11}, C11 = w_{12}g(x|m12, C_{12})$$

We know that $g(x|m, C)$ is a multivariate Gaussian probability density function with mean vector m and covariance matrix C where there parameters are: $w_{i1} = w_{i2} = 1/2$ for $i \in 1, 2$.

To find the theoretically optimal classifier, we would need to start off with the following equation

$$\frac{P(x|L=1)}{P(x|L=0)} \overset{?}{>=} \gamma$$

we also know that

$$\gamma = \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda 11} \cdot \frac{P(x|L=0)}{P(x|L=1)}$$

We can plug the class conditional to get the function where when solving, we get

$$\gamma = 1.5$$

We also got a Theoretical Minimum P(error) of 0.113 and a Classifier Minimum P(error) of 0.0928.

1.2 Part 2

Using Matlab's fminsearch, I trained three datasets D^{20}_{train}, D^{200}_{train}, D^{2000}_{train} in where I generated 10000 class posterior labels. From each of the 10000 class labels, since they were not fully rounded, I rounded them to the nearest class posterior label and compared them to the actual labels generated in $D^{10K}_{validate}$. Given this, I was able to get an error rate and a success rate. Part 2a's and part 2b's process was the same but just with a different activation function where 2a used an ISRU activation layer while 2b used a Logistic function layer.

	$d20$	$d200$	$d2000$
accuracy	0.5369	0.5127	0.5310
error	0.4631	0.4873	0.4690

Figure 1: Table showing the training accuracy for all training samples compared to the data for an ISRU activation layer.

	$d20$	$d200$	$d2000$
accuracy	0.5487	0.5091	0.5299
error	0.4513	0.4909	0.4701

Figure 2: Table showing the training accuracy for all training samples compared to the data for a Logistic Function activation layer.

From the tables, we can see that the training data doesn't really make the training more or less accurate. All training sets do seem to do better than 50% which is good as it shows that there is a little bit more than luck that is involved in the data. Also on top of this, it seems that the data only requires us to do 1 epoch which is probably the reason why the data is only slightly better than 50%. I believe with more layers and epochs, the data can be a lot more accurate.

Comparing the difference between the logistic function and the ISRU function, there isn't a significant difference either. While there is a difference, given the number of samples, the different activation layer functions seem to be insignificant to the data.

Comparing the difference between the classifiers trained and the expected error from the $D^{10K}_{validate}$, we can see that there is a huge error in the training data where $P(error)$ was around 0.4 to 0.5 for most of the data compared to the theoretical $P(error)$ of 0.113 or classifier $P(error)$ of 0.0928.

2 Question 2

To start this question, I first derived two estimators, an ML estimator and a MAP estimator. Assuming that scalar-real y and two-dimensional real vector x are related to each other according to $y = c(x, w) + v$, where $c(., w)$ is a cubic polynomial in x with coefficients w and v is a random Gaussian random scalar with mean zero and σ^2-variance.

2.1 Finding ML Estimator

Let's start with the ML Estimator

From this, we can say that

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

From this, given that $y = c(x, w) + v$ and $v_i \sim \mathcal{N}(0, \sigma^2)$, we have

$$y_i | x_i, w \sim \mathcal{N}(c(x_i, w), \sigma^2)$$

From this function, I got the following estimator

$$p(y_i|x_i, w, \beta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\left(\frac{y_i - w^T c(x_i)}{2\sigma^2}\right)^2\right)$$

Applying the argmax to the estimator

$$\arg\max \prod_{n=1}^{N} p(y_i|x_i, w, \beta) = \arg\max \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\left(\frac{y_i - w^T c(x_i)}{2\sigma^2}\right)^2\right)$$

Taking the log and cleaning up the function

$$\log p(y_i|x_i, w, \beta) = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2} \sum_{n=1}^{N} (y_i - w^T c(x_i))^2$$

We can discard the sum as this is our squared error. After setting the gradient to 0 and solving, we get

$$\hat{w}_{ML} = \frac{\hat{y}_i}{c}$$

2.2 Finding the MAP Estimator

To find the MAP Estimator, it is similar to finding the ML Estimator but we use a posterior distribution on top of a likelihood distribution.

Starting at the application of the arg max, the MAP Estimator would look like this

$$\arg\max \prod_{n=1}^{N} p(y_i|x_i, w, \beta) * p(x_i, w, \beta)$$

From here, since a MAP Estimator applies a regularization constant, we can derive that as $\lambda\mathcal{I}$ where λ is a matrix where all values are set to γ and $\mathcal{I}$ is the identity matrix. This would give the MAP Estimator to be

$$\hat{w}_{ML} = \frac{\hat{y_i}}{c}\lambda\mathcal{I}$$

2.3 Coding and Results

gamma	0.0001	0.001	0.01	0.1	1	10	100	1000	10000
MLE Error	4.6294	4.62934	4.6294	4.6294	4.6294	4.6294	4.6294	4.6294	4.6294
MAP Error	4.62939	4.62939	4.62929	4.62839	4.61997	4.57373	4.56589	4.61244	4.64570

Figure 3: Table of the gamma and both errors.

When γ was varied, we can see that when looking at the ML error compared to the MAP error, when the values got larger, there seemed to be a larger difference in error. This is due to the regularization that is done to the function whereas γ increases, the function is more regularized.

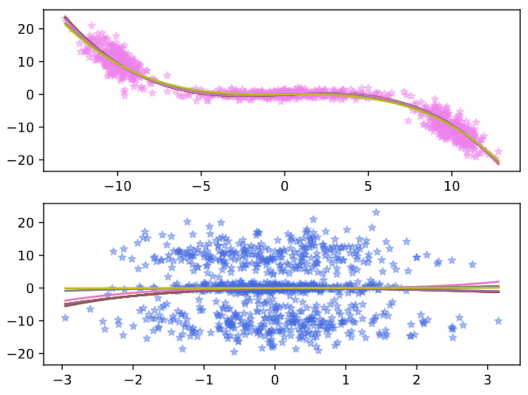

Figure 4: The graphs of the MAP Estimator

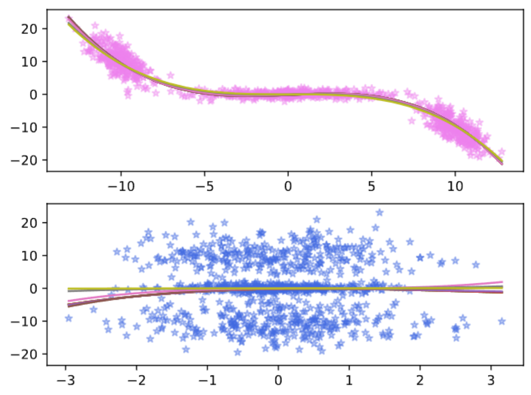

Figure 5: The graphs of the ML Estimator

Looking at figures 4 and 5, there isn't really much visual difference in the estimators. The differences are small and are better observed by looking at the average squared error earlier than by looking at the graphs. While the MAP estimate may not get closer to the true position, it does get more certain as it is regularized with $\gamma\mathcal{I}$.

3 Question 3

3.1 Optimization Problem

Simplifying the prior information, we get

$$p(x,y) = e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})}$$

Applying the MAP likelihood function, we get the follow function

$$\sum_{k=1}^{K} \frac{(r_i - d_{T_i})^2}{2\sigma_i^2} + \frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}$$

From this, after discarding unnecessary values, we can define the MAP as

$$\hat{x}, \hat{y} = \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}$$

The code first generates a random position for the true position in the circle. After that, it then places K landmarks around the origin where they are split evenly. Each of the land mark calculates the distance to the true value and it adds a preset noise term. Afterwards, it evaluates the MAP and plots the contors.

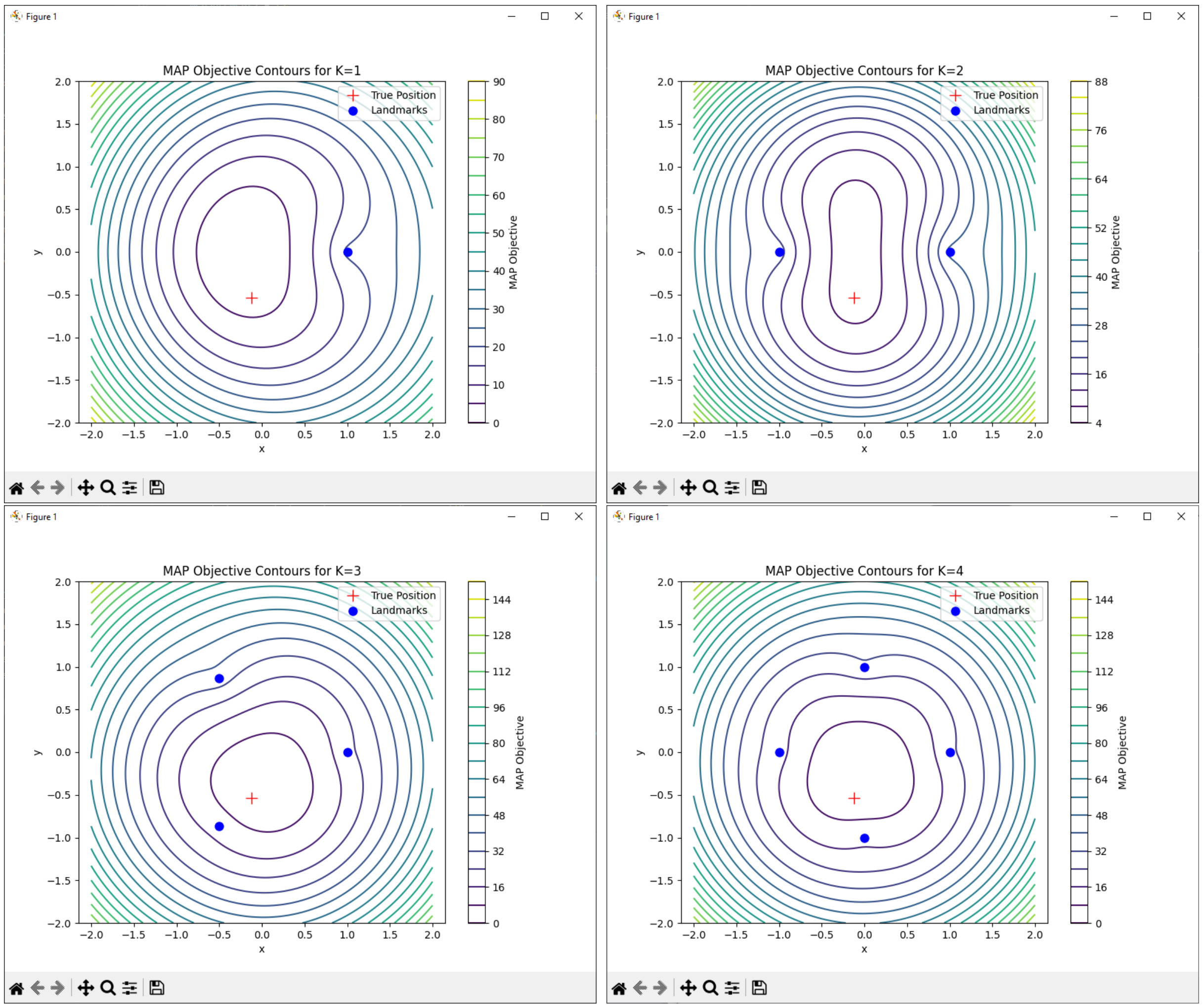

Figure 6: These plots here show the MAP Objective contors of each value K.

Looking at the plots, we can see that the smallest map always circles the true position. While hard to tell with only 4 K values, increasing the K value and testing up to $K = 10$ has given a more accurate MAP estimator. But in general, while some K values may be inconsistent, a higher K value doesn't always get closer to the true position but it is more certain to where the true position is.

The contors justify the solution since the MAP seems to capture the true position. While higher K values don't imply it gets closer to the true position, it does imply that it does get more certain.

4 Question 4

4.1 Minimum Risk Proof

To show that minimum risk is obtained if we decide ω_i if $P(\omega_i|x) \geq P(\omega_j|x)$ for all j and if $P(\omega_i|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$ and reject otherwise, let's start out by using the definition of risk using α_i

$$R(\alpha_i|x) = \sum_{i=1}^{c} \lambda(\alpha_i|w_j)P(w_j|x)$$

From this, where $\lambda(\alpha_i|\omega_j)$ is used to define choosing class α_i where the true class is ω_j, we get the simplified function

$$R(\alpha_i|x) = \lambda_s(1 - P(\omega_i|x))$$

Now to define the definition of risk for rejection, we have

$$R(\alpha_{c+1}|x) = \lambda_r$$

To minimize risk, we would need to reject where

$$R(\alpha_{c+1}) \leq R(\alpha_j|x)$$

By substituting, we get

$$\lambda_r \leq \lambda_s(1 - P(\omega_i|x))$$

This can be rearranged as

$$P(\omega_i|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$$

Given in the beginning that we decide that ω_i if $P(\omega_i|x) \geq P(\omega_j|x)$ for all j and if $P(\omega_i|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$ and reject otherwise, by applying the definition of risk, minimum risk is obtained where it is shown by proving equality.

4.2 If $\lambda_r = 0$

If $\lambda_r = 0$, this means that there is no loss for rejection. Applying this to the function $\lambda(\alpha_i|\omega_j)$, we should always reject as it does't affect our risk as to minimze risk in this situation. We would prefer to take no loss due to rejection rather than a substitution error.

4.3 If $\lambda_r > \lambda_s$

If $\lambda_r > \lambda_s$, this means that there is more loss from a rejection compared to a misclassification error. This would mean we would never reject as to minimize risk, a misclassification loss would be better than a rejection loss.

5 Question 5

5.1 ML Estimator for Θ

Looking at the question, we are given $D = \{z_1, ..., z_N\}$ where it is drawn independently from $Cat(\Theta)$ where $\Theta = [\theta_1, ..., \theta_K]^T$.

Given this, we can represent the likelihood function as

$$P(D|\Theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} \theta_K^{z_{n,k}}$$

To get $\hat{\theta_k}$, we would need to simplify this function where we can isolate θ. To do this, we can take the log of both sides which gives us

$$\log(P(D|\Theta)) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \log \theta_k$$

where $z_{n,k}$ is the k-th component of z_n.

Given the function of the ML Estimator is to maximize the log-likelihood with respect to Θ, given the constraint $\sum_{k=1}^{K}$, we can apply the Langrange multipliers, which gets us the following **ML Estimator**

$$\hat{\theta_k} = \frac{N_k}{N}$$

where $N_K = \sum_{n=1}^{N} z_{n,k}$ is the count of observations in k and $N = \sum_{k=1}^{K} N_k$ is the total number of samples.

5.2 MAP Estimator for Θ

Assuming that the prior distribution for Θ is a Dirchlet distribution with hyperparameter $\alpha = [\alpha_1, ..., \alpha_K]^T$, the Dirichlet distribution with parameter α is given by

$$p(\Theta|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

where $B(\alpha)$ is the normalization constant

$$B(\alpha) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)}$$

The MAP estimator maximizes the posterior distribution when

$$p(\Theta|D) \propto p(D|\Theta) p(\Theta|\alpha)$$

Combining the likelihood and the Dirichlet prior, the posterior distribution for Θ is

$$p(\Theta|D) \propto \prod_{k=1}^{K} \theta_k^{N_k+\alpha_k-1}$$

To maximize this with respect to Θ, given the problem parameters of $\sum_{k=1}^{K} \theta_k = 1$, we get the following **MAP estimate**

$$\hat{\theta} = \frac{N_k + \alpha_k - 1}{N + \sum_{k=1}^{K} \alpha_k - K}$$