COMP 540 Homework 3

Tony Ren (netid: lzr1), Hanyang Li (netid: hl43)

February 16, 2018

Problem 1: MAP and MLE parameter estimation

Please see the scanned pages at the end.

Problem 2: Logistic regression and Gaussian Naive Bayes

Problem 2.1

$$P(y=1|X) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$$

$$P(y=0|X) = 1 - g(\theta^T X) = \frac{e^{-\theta^T X}}{1+e^{-\theta^T X}}$$

Problem 2.2

$y \sim Bernoulli(\gamma)$
$x_j|y=0 \sim N(\mu_j^0, \sigma_j^2)$
$x_j|y=1 \sim N(\mu_j^1, \sigma_j^2)$
According to the Bayes Theorem,

$$P(y=1|X) = \frac{P(X|y=1)P(y=1)}{P(X)}$$

$$
\begin{aligned}
P(y=1|X) &= \frac{P(X|y=1)P(y=1)}{P(X)} \\
&= \frac{P(X|y=1)P(y=1)}{P(X,y=1)+P(X,y=0)} \\
&= \frac{P(X|y=1)P(y=1)}{P(X|y=1)P(y=1)+P(X|y=0)P(y=0)} \\
&= \frac{P(y=1)\prod_{j=1}^{d}P(x_j|y=1)}{P(y=1)\prod_{j=1}^{d}P(x_j|y=1)+P(y=0)\prod_{j=1}^{d}P(x_j|y=0)} \\
&= \frac{\gamma\prod_{j=1}^{d}f(x_j|\mu_j^1,(\sigma_j)^2)}{\gamma\prod_{j=1}^{d}f(x_j|\mu_j^1,(\sigma_j)^2)+(1-\gamma)\prod_{j=1}^{d}f(x_j|\mu_j^0,(\sigma_j)^2)}
\end{aligned} \tag{1}
$$

, where the probability density function $f(x|\mu,\sigma^2)$ of Gaussian Distribution is

$$
f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}
$$

Similarly, we can derive the class posterior probability for $y=0$:

$$
P(y=0|X) = \frac{\gamma\prod_{j=1}^{d}f(x_j|\mu_j^0,(\sigma_j)^2)}{\gamma\prod_{j=1}^{d}f(x_j|\mu_j^0,(\sigma_j)^2)+(1-\gamma)\prod_{j=1}^{d}f(x_j|\mu_j^1,(\sigma_j)^2)}
$$

, where the probability density function $f(x|\mu,\sigma^2)$ of Gaussian Distribution is

$$
f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}
$$

Problem 2.3

Assume $P(y=1)=P(y=0)=0.5$, then $\gamma=0.5$.
Under this condition, the simplified Naive Bayes for $P(y=1|X)$ is

$$
\begin{aligned}
P(y=1|X) &= \frac{\prod_{j=1}^{d}f(x_j|\mu_j^1,(\sigma_j)^2)}{\prod_{j=1}^{d}f(x_j|\mu_j^1,(\sigma_j)^2)+\prod_{j=1}^{d}f(x_j|\mu_j^0,(\sigma_j)^2)} \\
&= \frac{1}{1+\prod_{j=1}^{d}\frac{f(x_j|\mu_j^0,(\sigma_j)^2)}{f(x_j|\mu_j^1,(\sigma_j)^2)}}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
\prod_{j=1}^{d} \frac{f(x_j|\mu_j^0,(\sigma_j)^2)}{f(x_j|\mu_j^1,(\sigma_j)^2)} &= \prod_{j=1}^{d} \frac{\frac{1}{\sqrt{2\pi(\sigma_j)^2}} e^{-\frac{(x-\mu_j^0)^2}{2(\sigma_j)^2}}}{\frac{1}{\sqrt{2\pi(\sigma_j)^2}} e^{-\frac{(x-\mu_j^1)^2}{2(\sigma_j)^2}}} \\
&= \prod_{j=1}^{d} e^{\frac{(x-\mu_j^1)^2}{2(\sigma_j)^2} - \frac{(x-\mu_j^0)^2}{2(\sigma_j)^2}} \\
&= e^{\sum_{j=1}^{d} [\frac{(x-\mu_j^1)^2}{2(\sigma_j)^2} - \frac{(x-\mu_j^0)^2}{2(\sigma_j)^2}]} \\
&= e^{\sum_{j=1}^{d} [\frac{(\mu_j^1)^2-(\mu_j^0)^2}{2(\sigma_j)^2} + \frac{\mu_j^0-\mu_j^1}{(\sigma_j)^2} x_j]} \\
&= e^{-\sum_{j=1}^{d} [\frac{(\mu_j^0)^2-(\mu_j^1)^2}{2(\sigma_j)^2} + \frac{\mu_j^1-\mu_j^0}{(\sigma_j)^2} x_j]} \\
&= e^{-[\sum_{j=1}^{d} \frac{(\mu_j^0)^2-(\mu_j^1)^2}{2(\sigma_j)^2} + \sum_{j=1}^{d} \frac{\mu_j^1-\mu_j^0}{(\sigma_j)^2} x_j]}
\end{aligned} \tag{3}
$$

Gaussian Naive Bayes with uniform class priors:

$$
P(y=1|X) = \frac{1}{1+e^{-[\sum_{j=1}^{d} \frac{(\mu_j^0)^2-(\mu_j^1)^2}{2(\sigma_j)^2} + \sum_{j=1}^{d} \frac{\mu_j^1-\mu_j^0}{(\sigma_j)^2} x_j]}}
$$

Logistic Regression:

$$
P(y=1|X) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}} = \frac{1}{1+e^{-[\theta_0 + \sum_{j=1}^{d} \theta_j x_j]}}
$$

Parameterize Gaussian Naive Bayes:

$$
\theta_0 = \sum_{j=1}^{d} \frac{(\mu_j^0)^2-(\mu_j^1)^2}{2(\sigma_j)^2}
$$

$$
\theta_j = \frac{\mu_j^1-\mu_j^0}{(\sigma_j)^2} for j = 1, ..., d
$$

Thus, a Gaussian Naive Bayes model with uniform class priors is equivalent to a Logistic Regression model.

Problem 3: Reject option in classifiers

Problem 3.1

To minimize the risk, we need to make sure that (1) class j is the most possible class of x among all possible classes, and (2) the misclassification cost of choosing $y = j$ is not greater

than the cost of rejection.
The first condition is true when class j has the highest posterior probability among all class k (k=1,...,C). i.e. $P(y=j|x) \geq P(y=k|x)$ for all possible class k.
For the second condition to be true, we need to calculate the expected misclassification cost of choosing $y=j$, which is $(1-P(y=j|x))\lambda_s$
We should choose $y=j$ only if $(1-P(y=j|x))\lambda_s \leq \lambda_r$
i.e. $P(y=j|x) \geq 1-\frac{\lambda_r}{\lambda_s}$
Therefore, the minimum risk is obtained if the two conditions stated in the problem are met.

Problem 3.2

When $\frac{\lambda_r}{\lambda_s}=0$, we only choose $y=j$ when $P(y=j|x) \geq 1$, even if $P(y=j|x) \geq P(y=k|x)$ for all possible class k. This probability is very low, so in this case, we are very likely to take the reject action.
When $\frac{\lambda_r}{\lambda_s}=1$, we only choose $y=j$ when $P(y=j|x) \geq 0$, which means the probability that we choose $y=j$ is just the posterior probability $P(y=j|x)$. Under this condition, the probability that we choose $y=j$ is the biggest.
As $\frac{\lambda_r}{\lambda_s}$ increases, the relative cost of rejection increases. Therefore, in general, the bigger $\frac{\lambda_r}{\lambda_s}$ is, the less likely we are going to take the reject action.

Problem 4: Kernelizing k-nearest neighbors

For two points $x^{(i)}$ and $x^{(j)}$ from set D, the Euclidean Distance of them is:

$$d(x^{(i)},x^{(j)})=\sqrt{-2(x^{(i)})^Tx^{(j)}+(x^{(i)})^Tx^{(i)}+(x^{(j)})^Tx^{(j)}}$$

To kernelize a KNN classifier, we can define the kernel function to be the square of this Euclidean Distance function. Thus, by applying this kernel to the dataset, we can obtain a m by m Gram matrix K, where m is the number of data points, and each element of K is

$$k(x^{(i)},x^{(j)})=-2(x^{(i)})^Tx^{(j)}+(x^{(i)})^Tx^{(i)}+(x^{(j)})^Tx^{(j)}$$

Then, we can apply the decision rule of KNN to make predictions. To do this, we should find the labels of the k nearest neighbors (the k training points that have smallest $k(x^{(i)},x^{(j)})$ values with the test point) for each test data point and take the majority vote as the predicted class of the test point.

Problem 5: Constructing kernels

Problem 5.1

To prove a kernel is valid, we can show that its corresponding Gram matrix K is positive definite.

Since $k(x,x^{'}) = ck_1(x,x^{'})$, the relationship of their corresponding Gram matrices is $K = cK_1$
Since k_1 is a valid kernel, K_1 is positive definite. Therefore, for any non-zero vector z

$$z^T K z = z^T c K_1 z = c(z^T K_1 z)$$

Since $c > 0$ and $z^T K_1 z > 0$, $z^T K z > 0$
Therefore, K is positive definite and k is valid.

Problem 5.2

Since k_1 is a valid kernel, $\exists \phi$ such that $k_1(x,x^{'}) = \phi(x)^T \phi(x^{'})$

$$k(x,x^{'}) = f(x) k_1 f(x^{'}) = f(x)\phi(x)^T \phi(x^{'}) f(x^{'}) = \phi^{'}(x)^T \phi^{'}(x^{'})$$

Therefore, $k(x,x^{'})$ is a valid kernel.

Problem 5.3

Since $k(x,x^{'}) = k_1(x,x^{'}) + k_2(x,x^{'})$, the relationship of their corresponding Gram matrices is $K = K_1 + K_2$
Therefore, for any non-zero vector z

$$z^T K z = z^T (k_1 + K_2) z = z^T K_1 z + z^T K_2 z > 0$$

Therefore, $k(x,x^{'})$ is a valid kernel.

Problem 6: One vs all logistic regression

My OVA classifier and the sklearns implementation give the same results. They have the accuracy (0.361300), confusion matrix, and visualization of he learned weights for each class.

Problem 7

Please see the attached scanned pages.

HW 3 Problem 1

Lzr1
HL43

1.1 MLE: $\theta = \text{argmax}_\theta (1-\theta)^{m-h}\theta^h$

$\uparrow$ Bernoulli Distribution

m = total data points
h = total heads in data

$$\log\theta = (m-h)\log(1-\theta) + h\log\theta$$

$$\frac{\partial}{\partial\theta}\log\theta = -\frac{m-h}{1-\theta} + \frac{h}{\theta} = 0$$

$$\frac{h}{\theta} = \frac{m-h}{1-\theta}$$

$$h - h\theta = m\theta - h\theta$$

$$h = m\theta$$

$$\boxed{\theta_{MLE} = \frac{h}{m} = \frac{\text{heads in dataset}}{\text{total pts in dataset}}}$$

1.2 MAP: $\theta_{MAP} = \text{argmax}_\theta P(\theta|D)$

using Bayes Theorem

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$\uparrow$ ignore b/c no θ term

$$P(\theta) = \theta^{a-1}(1-\theta)^{b-1} \quad \text{(Beta distribution)}$$

$$\theta_{MAP} = \text{argmax}_\theta P(D|\theta)P(\theta)$$

$$= \text{argmax}_\theta \theta^h(1-\theta)^{m-h}(\theta^{a-1})(1-\theta)^{b-1}$$

$$= (\theta^{a+h-1})(1-\theta)^{b+m-h-1}$$

$$\log\theta_{MAP} = (a+h-1)\log\theta + (b+m-h-1)\log(1-\theta)$$

$$\frac{\partial}{\partial\theta}\log\theta_{MAP} = \frac{a+h-1}{\theta} + -\frac{b+m-h-1}{1-\theta} = 0$$

$$\frac{a+h-1}{\theta} = \frac{b+m-h-1}{1-\theta}$$

$$\theta(b+m-h-1) = (a+h-1)(1-\theta)$$

$$\theta b + \theta m - \theta h - \theta = a+h-1 - \theta a - \theta h + \theta$$

$$\theta a + \theta b + \theta m - 2\theta = a+h-1$$

$$\boxed{\theta_{MAP} = \frac{a+h-1}{a+b+m-2} \Rightarrow \text{if } a=b=1,\ \theta_{MAP} = \frac{1+h-1}{1+1+m-2} = \frac{h}{m} = \theta_{MLE}}$$

Problem 7

7.1

The reason the loss should be about $-ln(0.1)$ is because of the probability function. There are 10 classes in this dataset and therefore a random sampling should contain about all 10 classes with the probability of a specific class showing up to be 1/10. Therefore, when calculating the loss function, the inside of the log_e will be 1/10.

7.9

It seems like the OVA has a higher accuracy at 0.361 compared to our Softmax regression's 0.274. Softmax regression should be able to attain greater than 0.35 but it seems like the parameters we used did not show that. Our conclusion is that OVA is better at classifying for the CIFAR-10 dataset in this case but in general they should behave around the same amount.

Columns for Confusion Matrix:

$['plane','car','bird','cat','deer','dog','frog','horse','ship','truck']$

Confusion Matrix for Softmax

$$\begin{bmatrix} 588 & 8 & 2 & 3 & 7 & 31 & 88 & 6 & 201 & 66 \\ 186 & 87 & 0 & 5 & 8 & 44 & 294 & 12 & 161 & 203 \\ 307 & 11 & 19 & 3 & 30 & 84 & 403 & 24 & 62 & 57 \\ 211 & 8 & 6 & 22 & 11 & 177 & 381 & 29 & 53 & 102 \\ 155 & 7 & 12 & 6 & 45 & 105 & 520 & 28 & 53 & 69 \\ 212 & 2 & 10 & 11 & 17 & 271 & 310 & 32 & 80 & 55 \\ 155 & 4 & 2 & 3 & 9 & 72 & 638 & 23 & 19 & 75 \\ 198 & 8 & 1 & 6 & 34 & 107 & 265 & 89 & 81 & 211 \\ 265 & 17 & 0 & 1 & 3 & 69 & 64 & 3 & 461 & 117 \\ 190 & 25 & 1 & 2 & 8 & 19 & 138 & 9 & 191 & 417 \end{bmatrix}$$

$$\text{Confusion Matrix for OVA} \begin{bmatrix} 465 & 59 & 22 & 24 & 19 & 35 & 26 & 60 & 200 & 90 \\ 67 & 463 & 18 & 34 & 23 & 31 & 44 & 51 & 96 & 173 \\ 123 & 64 & 194 & 77 & 96 & 89 & 151 & 88 & 70 & 48 \\ 67 & 86 & 78 & 161 & 48 & 193 & 171 & 51 & 62 & 83 \\ 65 & 38 & 102 & 64 & 234 & 90 & 194 & 129 & 36 & 48 \\ 47 & 63 & 81 & 127 & 81 & 272 & 114 & 89 & 71 & 55 \\ 31 & 53 & 67 & 102 & 86 & 78 & 457 & 51 & 29 & 46 \\ 53 & 62 & 51 & 46 & 69 & 85 & 66 & 406 & 47 & 115 \\ 149 & 78 & 8 & 25 & 9 & 34 & 22 & 19 & 541 & 115 \\ 59 & 208 & 14 & 22 & 23 & 29 & 60 & 56 & 109 & 420 \end{bmatrix}$$

By looking at the confusion matrices, one can get a good idea of what classifications worked and which did not. Looking at the diagonal row shows the categories that were obtained correctly by each classification algorithm. It appears that Softmax is very good at classifying some types of images while being very poor for others while OVA seems pretty decent regardless of the image. There are still ups and downs but much less drastic as softmax. For example, planes are classified well (588) in Softmax but birds, cats and deer all perform much worse(19, 22, and 45). OVA is much better in these categories(194,161,234), but classify planes more poorly(465)