CSCI 5525: Machine Learning

Homework 1

Jingxiang Li

October 1, 2015

Problem 1.a

The optimal $f(x)$ is $f(x) = E(y|x)$

Proof. The expected loss is

$$\begin{aligned} E_{(x,y)}[l(f(x),y)] &= \int_x \int_y l(f(x),y)p(y|x)dyp(x)dx \\ &= \int_x \int_y (f(x)-y)^2 p(y|x)dyp(x)dx \end{aligned}$$

Here we minimize the expected loss

$$\min_f \mathrm{E}_{(x,y)}[l(f(x),y)] \Leftrightarrow \min_f \int_y (f(x)-y)^2 p(y|x)dy$$

To minimize the expected loss, we take the derivative with respect to f, and set it to 0.

$$\begin{aligned} &\frac{\partial}{\partial f}\int_y (f(x)-y)^2 p(y|x)dy = 0 \\ \Rightarrow &\int_y \frac{\partial}{\partial f}(f(x)-y)^2 p(y|x)dy = 0 \\ \Rightarrow &\int_y (f(x)-y)p(y|x)dy = 0 \\ \Rightarrow &\int_y f(x)p(y|x)dy = \int_y yp(y|x)dy \\ \Rightarrow &f(x) = E(y|x) \end{aligned}$$

Q.E.D

Problem 2.a

Proof. The expected loss for f is

$$L(f) = p(f(x) \neq y) = \mathrm{E}_{(x,y)}[I(f(x) \neq y)] = \int_x \sum_y I(f(x) \neq y) p(y|x) p(x) dx$$

Here we minimize the expected loss with respect to f

$$\begin{aligned} &\min_f \int_x \sum_y I(f(x) \neq y) p(y|x) p(x) dx \\ \Leftrightarrow &\min_f \sum_y I(f(x) \neq y) p(y|x) \\ \Leftrightarrow &\min_f I(f(x) \neq 1) p(1|x) + I(f(x) \neq -1) p(0|x) \end{aligned}$$

Notice that $p(0|x) > 0, p(1|x) > 0$ and $p(0|x) + p(1|x) = 1$.

Considering $I(f(x) \neq 1)$ and $I(f(x) \neq 0)$, there will be exactly one has value 1, and another one 0.

Hence once $p(1|x) > p(0|x)$, the optimal f^* must satisfy $f^*(x) = 1$; otherwise $f^*(x) = -1$

i.e.

$$f^*(x) = \begin{cases} +1, & \text{if } p(1|x) > 1/2 \\ -1, & \text{otherwise} \end{cases}$$

Since f^* is optimal, $L(f^*) \leq L(f)$

Q.E.D.

Problem 3.a

Fisher's Linear Discriminant

Similar to Fisher's Linear Discriminant for two classes problem, we first construct the within-class covariance matrix S_W and between class covariance matrix S_B.

Let K be the number of classes, C_k be the indicator of class k, N_k be the number of observations in class k. Then we can derive S_W and S_B by:

$$S_W = \sum_{k=1}^{K} S_k$$

$$S_B = \sum_{k=1}^{K} N_k(m_k - m)(m_k - m)^T$$

where

$$S_k = \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T$$

$$m_k = \frac{1}{N_k} \sum_{n \in C_k} x_n$$

Let W be the resulting weight matrix for Fisher's Linear Discriminant, we can derive it by maximizing the following quantity

$$J(W) = \operatorname{Tr}\left\{(WS_WW^T)^{-1}(WS_BW^T)\right\}$$

The weight matrix are determined by those eigenvectors of $S_W^{-1}S_B$ that correspond to the D' largest eigenvalues[1].

Note that S_B is composed from K rank 1 matrices, and only $(K - 1)$ of these matrices are independent, thus S_B has rank at most equal to (K - 1) and so there are at most (K - 1) nonzero eigenvalues. Hence we are unable to find more than $(K - 1)$ linear "features" by this means[2].

Sometimes calculating eigenvectors for $S_W^{-1}S_B$ results in all complex vectors, which is due to the bad condition of S_W. For this case we simply find eigenvectors for $(S_W + I)^{-1}S_B$ as a substitute.

After obtaining the weight matrix W, we project X by $X^{\text{new}} = XW$ and treat X^{new} as the new design matrix for future processing.

Multivariate Gaussian Generative Model

Multivariate Gaussian Generative Model assumes that each class of observations come from a multivariate Gaussian distribution. To make prediction for single observations, it applies the following Bayes rule

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}, \quad k = 1, \ldots, K$$

Note that $p(x)$ is equal for all k, hence we can ignore the denominator of the posterior probability. To make predictions for a single observation x, we simply calculate "prediction score" $log(p(x|C_k)) + log(p(C_k))$, which is simply the logarithm of the above formula, for all k and classify it to the class with the largest "prediction score".

It's easy to derive $p(x|C_k)$ and $p(C_k)$. for $p(C_k)$, it can be pre-specified or use the class proportion:

$$p(C_k) = \frac{\#\{x_n | x_n \in C_k\}}{\#\{x_n\}}$$

For $p(x|C_k)$, since we assume C_k has a multivariate normal distribution, it's sufficient to estimate the mean μ_k and the covariance matrix Cov_k of C_k, where

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n \in C_k} x_n$$
$$\hat{\mathrm{Cov}}_k = \frac{1}{N_K - 1} \sum_{n \in C_k} (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^T$$

Then it's straightforward to estimate $p(x|C_k)$.

Least Squares Linear Discriminant

Let $X \in \mathbb{R}^{n \times p}$ be the design matrix, $y \in \mathbb{R}^{n \times K}$ be the response matrix in bit-vector representation. For each class C_k, the model assumes

$$y_{i,k} = w_k^T x_i, \quad k = 1, \ldots, K$$

whose matrix form is

$$y = WX$$

To estimate W, we consider least square estimate, which means to minimize the following quantity

$$E_D(\tilde{W}) = \frac{1}{2} \mathrm{Tr} \left\{ (X\tilde{W} - y)^T (X\tilde{W} - y) \right\}$$

The solution for the above optimization problem is

$$\tilde{W} = (X^T X)^{-1} X^T y$$

Notice that $X^T X$ might be invertible, in such case we use $X^T X + I$ as a substitute.

To make prediction for new observation x, we project it to $x^T W$ and predict it as class k if the kth element of the resulting projected vector has the largest value among others.

Simulation Result

The simulation is based on MNIST-1378 dataset for 4-class classification of hand written digits: 1, 3, 7 and 8. We use 10 fold cross-validation to evaluate the performance of the following two modeling procedures:

1. Fisher's Discriminant Analysis + Multivariate Gaussian Generative Model (Fisher)
2. Least Squares Linear Discriminant (LS)

The simulation result is summarized in table 1. Note that they all perform well in terms of prediction, Fisher seems to be slightly better than LS method, but the difference in prediction accuracy is almost negligible.

Table 1: Fisher + Multivariate Gaussian V.S. Least Squares Discriminant

	Mean	Std	fold0	fold1	fold2	fold3	fold4	fold5	fold6	fold7	fold8	fold9
Fisher Train Erros	4.66%	0.06%	4.78%	4.60%	4.61%	4.65%	4.64%	4.69%	4.74%	4.63%	4.63%	4.60%
LS Train Erros	5.20%	0.05%	5.21%	5.17%	5.19%	5.32%	5.19%	5.21%	5.19%	5.18%	5.24%	5.15%
Fisher Test Erros	5.37%	0.37%	5.04%	5.66%	5.53%	5.28%	5.25%	4.84%	5.18%	5.12%	5.87%	5.97%
LS Test Erros	5.99%	0.50%	5.90%	6.38%	6.66%	4.91%	6.11%	5.83%	6.39%	5.66%	5.70%	6.32%

Problem 4.a

Logistic Regression

Logistic Regression assumes

$$p(C_1|x) = \sigma(w^T x)$$

where $\sigma(\cdot)$ is the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Then for a data set$\{x_n, y_n\}$, where $y_n \in \{0, 1\}$, the likelihood function is

$$p(y|x) = \prod_{n=1}^{N} \sigma(w^T x_n)^{y_n}(1 - \sigma(w^T x_n))^{1-y_n}$$
$$\ln(p(y|x)) = \sum_{n=1}^{N} \left\{ y_n \ln(\sigma(w^T x_n)) + (1 - y_n) \ln(\sigma(w^T x_n)) \right\}$$

To get an estimate of w, we maximize the regularized log likelihood function, which is equivalent to

$$\arg\min_{w} L(w) = -\ln(p(y|x)) + \frac{\lambda}{2}||w||^2$$

To solve the above optimization problem, we consider gradient descent.

It's easy to derive the first order derivative for the loss function

$$\nabla L(w) = \sum_{n=1}^{N}(\sigma(w^T x_n) - y_n)x_n + \lambda w$$

Then the iterative equation is

$$w := w - \alpha \nabla L(w)$$

where α is step size.

By iteratively update w till convergence (or maximum number of iteration is reached), we can obtain an estimate for w.

Naive Bayes with Marginal Univariate Gaussian Approximation

Naive Bayes assumes features are independent with each other, which suggests that

$$p(x_1, \ldots, x_p|C_k) = \prod_{i=1}^{p} p(x_i|C_k)$$

By univariate Gaussian approximation, we assume each $p(x_i|C_k),\ i = 1, \ldots, N,\ k = 1, \ldots, K$ has a Gaussian distribution, whose parameters can be estimated from sample.

For prediction, simply apply Bayes rule

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{\prod_{i=1}^{p} p(x_i|C_k)p(C_k)}{p(x)}$$

By taking the logarithm of $p(C_k|x)$, we can simply derive a score for prediction

$$\text{score}_k = \sum_{i=1}^{p} \log p(x_i|C_k) + \log p(C_k)$$

Then we can predict new observation as class k if the prediction score for class k has the largest value among others.

Simulation Result

The simulation is based on Spam dataset for 2-class classification. For each iteration, we first split the whole dataset into 80% for training and 20% for test. Then, within training set, we randomly select 5%, 10%, ..., 30% for training the classifier, and record test error for each of the inner-split. The whole procedure will be repeated 100 times. We apply this validation procedure to evaluate the prediction performance of the following two models:

1. Logistic Regression
2. Naive Bayes with Marginal Univariate Gaussian Approximation

The simulation result is summarized in table 2 and figure 1. According to the mean test errors, Logistic Regression dominates Naive Bayes in all scenarios. As the size of training set increases, the test error of logistic regression drops significantly. However, the test error of naive Bayes seems to be insensitive to the increasing size of training set.

Table 2: Logistic V.S. Naive Bayes

	5 percent	10 percent	15 percent	20 percent	25 percent	30 percent
Logistic Test Error Mean	11.69%	10.04%	9.36%	8.78%	8.58%	8.38%
Logistic Test Error Std	1.60%	1.15%	1.11%	1.02%	1.03%	0.95%
Naive Bayes Test Error Mean	18.49%	18.33%	18.55%	18.63%	18.29%	18.66%
Naive Bayes Test Error Std	1.86%	1.62%	1.66%	1.50%	1.43%	1.46%

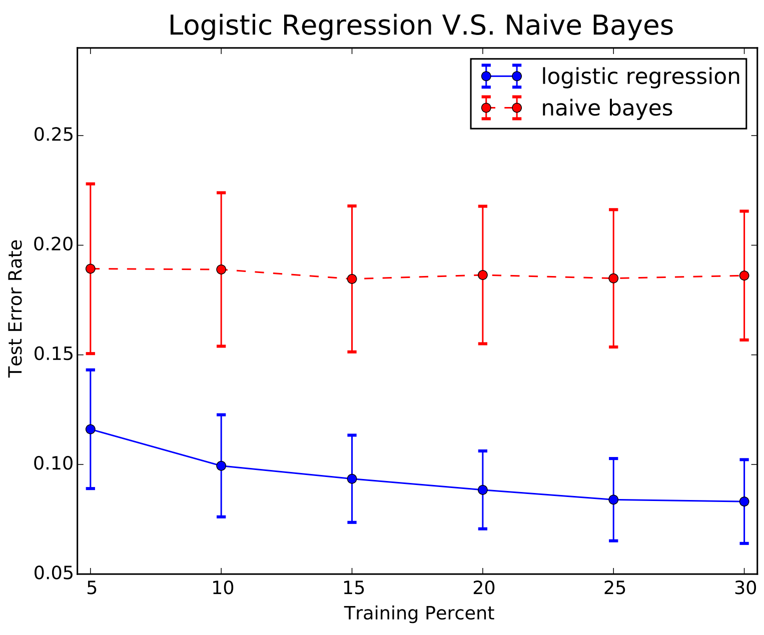

Figure 1: Logistic V.S. Naive Bayes

References

1. Bishop, C. M. *Pattern recognition and machine learning* (springer, 2006).
2. FUKUNAGA, R. Statistical pattern recognition (1990).