CSC2515: Assignment 2

Due on Sunday, November 12, 2017

Professor Ethan Fetaya

Zhaoyu Guo (999008069)

Question One

In this question, we will derive the maximum likelihood estimates for class-conditional Gaussians with independent features(diagonal covariance matrices) i.e. Gaussian Naive Bayes, with shared variances.The discrete class label $y \in (1, 2, ..., K)$ and a real valued vector of d features $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_d)$

Part One

Based on the question, we know $p(y = k) = \alpha_k$ and

$$p(\mathbf{x}|y = k, \boldsymbol{\mu}, \boldsymbol{\sigma}) = (\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2\}$$

Then, by using Baye's rule and law of total probability, we can derive the expression for $p(y = k|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma})$

$$\begin{aligned}
p(y = k|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}) &= \frac{p(\mathbf{x}|y = k, \boldsymbol{\mu}, \boldsymbol{\sigma})p(y = k)}{p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\sigma})} \\
&= \frac{p(\mathbf{x}|y = k, \boldsymbol{\mu}, \boldsymbol{\sigma})p(y = k)}{\sum_{k=1}^{K}[p(\mathbf{x}|y = k, \boldsymbol{\mu}, \boldsymbol{\sigma})p(y = k)]} \\
&= \frac{(\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2\}\alpha_k}{\sum_{k=1}^{K}[(\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2\}\alpha_k]} \\
&= \frac{exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2\}\alpha_k}{\sum_{k=1}^{K}[(exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2\}\alpha_k]}
\end{aligned}$$

Part Two

In this part, we will write down the expression for the negative likelihood function(NLL)

$$l(\boldsymbol{\theta}; D) = -\log p(y^{(1)}, \mathbf{x}^{(1)}, y^{(2)}, \mathbf{x}^{(2)}, ..., y^{(N)}, \mathbf{x}^{(N)}|\boldsymbol{\theta})$$

Then, we can separate the NLL into two parts:

$$\begin{aligned}
l(\boldsymbol{\theta}; D) &= -\log p(y^{(1)}, \mathbf{x}^{(1)}, y^{(2)}, \mathbf{x}^{(2)}, ..., y^{(N)}, \mathbf{x}^{(N)}|\boldsymbol{\theta}) \\
&= -[\log p(y^{(1)}, y^{(2)}, ..., y^{(N)}|\boldsymbol{\theta}) + \log p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}...\mathbf{x}^{(N)}|y^{(1)}, y^{(2)}, ..., y^{(N)}, \boldsymbol{\theta}]
\end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$, and we assume that the data are iid.

$$\begin{aligned}
\log p(y^{(1)}, y^{(2)}, ..., y^{(N)}|\boldsymbol{\theta}) &= \log[p(y^{(1)}|\boldsymbol{\theta}) \times p(y^{(2)}|\boldsymbol{\theta}) \times ... \times p(y^{(N)}|\boldsymbol{\theta})] \\
&= \log[\alpha_{y^{(1)}} \times \alpha_{y^{(2)}} \times ... \times \alpha_{y^{(N)}}] \\
&= \sum_{n=1}^{N} \log \alpha_{y^{(n)}}
\end{aligned}$$

$$\begin{aligned}
\log p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}...\mathbf{x}^{(N)}|y^{(1)}, y^{(2)}, ..., y^{(N)}, \boldsymbol{\theta}) &= \log[\prod_{n=1}^{N}[(\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(n)} - \mu_{y^{(n)}i})^2\}]] \\
&= \sum_{n=1}^{N} \log[(\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} exp\{-\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(n)} - \mu_{y^{(n)}i})^2\}] \\
&= \sum_{n=1}^{N}[-\frac{1}{2}\sum_{i=1}^{D} \log[2\pi\sigma_i^2] - \sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(n)} - \mu_{y^{(n)}i})^2]
\end{aligned}$$

Combine those two expressions together, we will have:

$$\begin{aligned} l(\boldsymbol{\theta}; D) &= -\log p(y^{(1)}, \mathbf{x}^{(1)}, y^{(2)}, \mathbf{x}^{(2)}, ..., y^{(N)}, \mathbf{x}^{(N)}|\boldsymbol{\theta}) \\ &= -[\log p(y^{(1)}, y^{(2)}, ..., y^{(N)}|\boldsymbol{\theta}) + \log p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}...\mathbf{x}^{(N)}|y^{(1)}, y^{(2)}, ..., y^{(N)}, \boldsymbol{\theta}] \\ &= -\sum_{n=1}^{N}[\log \alpha_{y^{(n)}} - \frac{1}{2}\sum_{i=1}^{D}\log[2\pi\sigma_i^2] - \sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i^{(n)} - \mu_{y^{(n)}i})^2] \end{aligned}$$

Part Three

In this part, we want to take partial derivatives of the likelihood with respect to each of the parameters μ_{ki} and with respect to the shared variances σ_i^2.
First, let's take partial derivatives of the likelihood with respect to each of the parameters μ_{ki}:

$$\frac{\partial l(\boldsymbol{\theta}; D)}{\partial \mu_{ki}} = \sum_{n=1}^{N}\mathbb{1}(y^{(n)} = k)[-\frac{{x_i}^{(n)} - \mu_{y^{(n)}i}}{\sigma_i^2}]$$

Then, let's take partial derivatives of the likelihood with respect to each of the parameters σ_i^2:

$$\begin{aligned} \frac{\partial l(\boldsymbol{\theta}; D)}{\partial \sigma_i^2} &= \sum_{n=1}^{N}[[-\frac{1}{2} \times \frac{1}{2\pi\sigma_i^2} \times 2\pi] - [\frac{({x_i}^{(n)} - \mu_{y^{(n)}i})^2}{2} \times (-1) \times (\sigma_i^2)^{-2}]] \\ &= \sum_{n=1}^{N}[-\frac{1}{2\sigma_1^2} + \frac{1}{2}(\frac{{x_i}^{(n)} - \mu_{y^{(n)}i}}{\sigma_i^2})^2] \end{aligned}$$

Part Four

Setting the partial derivatives equal to 0, and we will find the maximum likelihood estimates for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$:

$$\frac{\partial l(\boldsymbol{\theta};D)}{\partial \mu_{ki}} = \sum_{n=1}^{N}\mathbb{1}(y^{(n)} = k)[-\frac{{x_i}^{(n)} - \mu_{y^{(n)}i}}{\sigma_i^2}] = 0$$

Then, we will get MLE of μ_{ki}

$$\mu_{ki} = \frac{\sum_{n=1}^{N}\mathbb{1}(y^{(n)} = k)x_i^{(n)}}{\mathbb{1}(y^{(n)} = k)}$$

$$\frac{\partial l(\boldsymbol{\theta};D)}{\partial \sigma_i^2} = \sum_{n=1}^{N}[-\frac{1}{2\sigma_1^2} + \frac{1}{2}(\frac{{x_i}^{(n)} - \mu_{y^{(n)}i}}{\sigma_i^2})^2] = 0$$

Then, beacause we are using the shared variances for each feature, so we will get MLE of σ_i^2

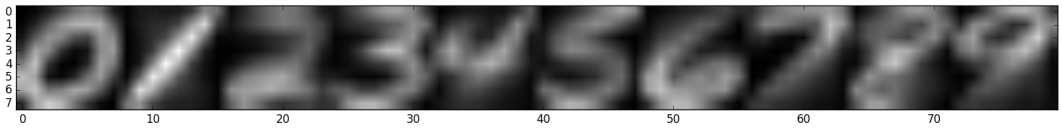
$$\sigma_i^2 = \frac{1}{N}\sum_{n=1}^{N}[x_i^{(n)} - \mu_{y^{(n)}i}]^2$$

The matrix $\boldsymbol{\mu}$ will be 10 by 64 matrix, and $\boldsymbol{\sigma}$ will be 64 by 64 diagonal matrix with all off-diagonal elements equal to 0.

Question Two

2.0: Load Data and Visualization

Please see ***q2_0.py*** for the implementation.



2.1: K-NN Classifier

Part 1

For K = 1, the training data classification accuracy is 100%, the test data classification accuracy is 96.88%.
For K = 15, the training data classification accuracy is 96.37%, the test data classification accuracy is 96.1%

Part 2

For $K > 1$, K-NN might encounter ties that need to be broken in order to make a decision. When more than one classes with the same number of data points closest to the test point $\mathbf{x}$, we will compare the distance from the most closest point in all tie classes, if the tie still exist, we will compare the distance from the second closest point, and until we found an unique class. Then $\mathbf{x}$ will be assigned to that class.

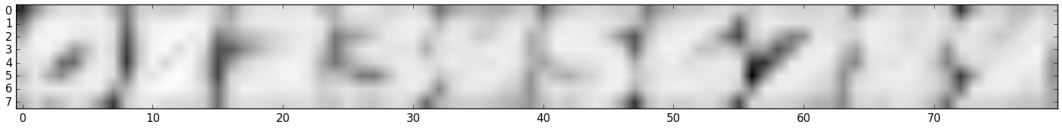
Part 3

When we used 10 fold cross validation, we found that the optimal K is 4.
When K = 4, the training data classification accuracy is 98.64%, the test data classification accuracy is 97.28%. The average accuracy across folds is 96.56%.

2.2: Conditional Gaussian Classifier Training

Part 1

Plot an 8 by 8 image of the log of the diagonal elements of each covariance matrix Σ_k. Plot all ten classes side by side using the same gray-scale.



Part 2

The average conditional log-likelihood for training data is -0.1246244
The average conditional log-likelihood for test data is -0.1966732

Part 3

The accuracy for training data is 98.14%
The accuracy for test data is 97.275%

2.3: Naive Bayes Classifier Training

Part 1

Please see ***q2_3.py*** for the implementation.

Part 2

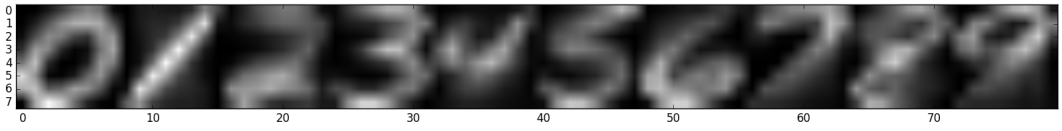
Let N_k represent the number of data points of the kth class, and let N_{kj} be the number of times $b_j = 1$ in the kth class. Then, by Bayes rule, we will have:

$$\begin{aligned}
p(\eta_{kj}|y=k,\mathbf{b}) &\propto \frac{p(\mathbf{b}|y=k,\eta_{kj})p(\eta_{kj})}{p(\mathbf{b},y=k)} \\
&\propto p(\mathbf{b}|y=k,\eta_{kj})p(\eta_{kj}) \\
&\propto [\prod_{k=1}^{N_k}\prod_{j=1}^{d}(\eta_{kj})^{b_j^{(k)}}(1-\eta_{kj})^{1-b_j^{(k)}}] \times \frac{\eta_{kj}^{2-1}(1-\eta_{kj}^{2-1})}{Beta(2,2)} \\
&\propto [\eta_{kj}^{N_{kj}}(1-\eta_{kj})^{N_k-N_{kj}}] \times [\eta_{kj}(1-\eta_{kj})] \\
&\propto \eta_{kj}^{N_{kj}}(1-\eta_{kj})^{N_k-N_{kj}} \\
&\propto Beta(2+N_{kj}, 2+N_k-N_{kj})
\end{aligned}$$

Since the mode of a Beta(a,b) distribution is $\frac{a-1}{a+b-2}$, we have MAP of η_{kj} is $\frac{1+N_{kj}}{2+N_k}$

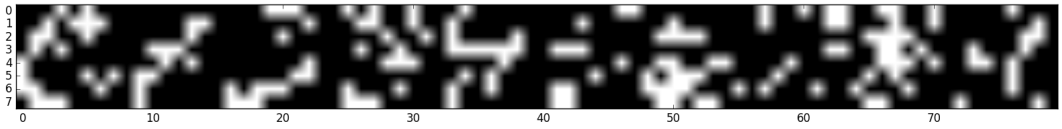
Part 3

Plot each of our η_k vectors as an 8 by 8 gray-scale image.



Part 4

Given our parameters, sample one new data point using our generative model for each of the 10 digit classes. Plot these new data points as 8 by 8 gray-scale images.



Part 5

The average conditional log-likelihood for training data is -0.94375386
The average conditional log-likelihood for test data is -0.9872704337

Part 6

The accuracy for training data is 77.414%
The accuracy for test data is 76.425%

2.4: Model Comparison

When comparing the result of those classifiers, we found that all of three classifiers can get test classification accuracy rate greater than 75%. K-NN classifier with 97.28% test classification accuracy, Conditional Gaussian classifier with 98.7% test classification accuracy, and Naive Bayes classifier with 76.425% test classification accuracy. We can see Conditional Gaussian has the best performance and Naive Bayes has the worst performance. The conditional Gaussian classifier uses a probabilistic process to calculate the probability of a new test data point to each digit class by using a prior and evidence. In addition, conditional Gaussian classifier does not need the assumption that features are independent. K-NN classifier also get a very high test classification accuracy. K-NN is the non-parametric mode. In K-NN, we just need to choose k, then we can get a result with that k, so K-NN is almost the simplest classifier. However, in k-NN, there might encounter ties that need to be broken in order to make a decision. In Naive Bayes classifier, we need a strong assumption: when class is given, each feature is independent. However, this assumption may not be hold in our dataset since the pixels in each digit class are not independent because they may interact with other pixels. Therefore, the results match our expectations, especially we get the lowest test classification accuracy in Naive Bayes as expected.