PRML Assignment 1

April 15, 2020

Introduction

In this assignment, we are going to generate a data set with three labels, whose data is sampled from three different gaussian distribution. Then we will construct 2 linear classification model(namely generative model and discriminative model) to classify the data set. Finally, we will reorganize the data set, sample it from different distributions setting, comparing the classification results, and find out the factors that affect classification results.

Data generation

As we all know, a variable that follows gaussian distribution can be represented as $X \sim \mathcal{N}(\mu, \delta^2)$, and its probability density function is:

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

So we can know that we can determine a gaussian distribution by two parameter: the mean μ, and the variance of the distribution δ^2.
In this assignment, we first generate data in the following setting:

$$\mathcal{N}_A = \mathcal{N}(0,1) \quad \mathcal{N}_B = \mathcal{N}(4,1) \quad \mathcal{N}_C = \mathcal{N}(-4,1)$$

$$Dataset = \{((x,y), label) | x \sim \mathcal{N}_{label}, y \sim \mathcal{N}_{label}\}$$

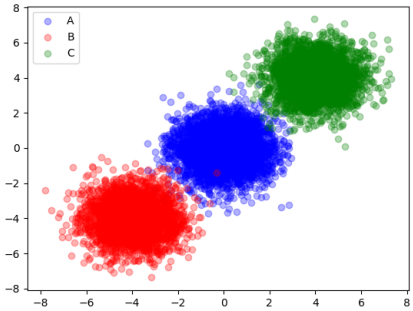
$$DatasetSize = 10000$$

$$P_{label} = \{A : 0.4, B : 0.3, C : 0.3\}$$

You can generate the data by command:

```
python source.py generate
```

to generate this data set, and write it to a.data in the folder. Furthermore, we can visualize the data set to coordinate axis, the result is as follow:



Implement method: we use **numpy.random.normal(loc, scale, size)** to sample from a gaussion distribution, and we use **matplotlib.pyplot.scatter(x,y)** to visualize the data.

Generative model

In this model, we assume that data of each label is normally distributed, then we can estimate a gaussion distribution for each label. Then we can optimize the distribution parameter to maximize the likelihood function. We can solve this problem by mathematical method as following:
Target: find $\hat{\theta}$ that:

$$\hat{\theta} = arg \max_{\theta} P(\mathbf{X}|\theta)$$

and $P(\mathbf{x}|\theta)$ is likelihood function. Specifically in this assignment:

$$\begin{aligned} P(\mathbf{X}|\theta) &= \prod_{j=1}^{3} \prod_{x_i \in j} pr(x_i|y_i) \\ &= \prod_{j=1}^{3} \prod_{x_i \in j} \frac{1}{2\pi\delta_{y_i}} e^{-\frac{1}{2}(\frac{x-\mu_{y_i}}{\delta})^2} \end{aligned}$$

It's hard to calculate its calculus, we can change our goal to maximize the $lnP(\mathbf{x}|\theta)$, then we have:

$$
\begin{aligned}
ln\ P(\mathbf{X}|\theta) &= ln \prod_{j=1}^{3} \prod_{x_i \in j} \frac{1}{2\pi\delta_{y_i}} e^{-\frac{1}{2}(\frac{x-\mu_{y_i}}{\delta})^2} \\
&= -\sum_{j=1}^{3} n_j ln(2\pi\delta_j) \\
&\quad -\sum_{j=1}^{3} \frac{1}{2\delta^2} \sum_{x_i \in j} \|x_i - \mu_i\|^2
\end{aligned}
$$

then we can calculate the partial derivatives for it:

$$
\frac{\partial\ lnP}{\partial \mu_i} = -\frac{1}{\delta_i} \sum_{x_j \in i} (x_j - \mu_i)
$$

$$
\frac{\partial\ lnP}{\partial \delta_i} = -\frac{n_i}{\delta_i} + \frac{1}{\delta_i^3} \sum_{x_j \in i} \|x_j - \mu_i\|^2
$$

let the partial derivatives be 0, then we have:

$$
\mu_i = \frac{1}{n_i} \sum_{x_j \in i} x_j
$$

$$
\delta_i^2 = \frac{1}{n_i} \sum_{x_j \in i} \|x_j - \mu_i\|^2
$$

and n_i stands for the appear times for each label in the data set. So we can directly calculate the optimal parameters in this way.
For prediction, we can calculate the probability by Bayes' theorem:

$$
P(y_i|x_i) = P(x_i|y_i) * P(y_i)
$$

and y_i stands for the probability label y_i appear in the data set. And we classify the example to the label with highest probability.
P.S. we need to mention here that the data for each dimension are generated respectively, so there is no need to consider covariance, we can easily find that in this situation, the results are same with the formula above if we consider the covariance.

Discriminative model

For this model, we use a linear classification algorithm, we train a weight matrix and W and a constant b, then we build the $f(x)$:

$$
f(x) = Wx + b
$$

and W is a $n * dim$ matrix, x is a $dim * 1$ vector, where n stands for the label number, and dim stands for the dimensions of x, and b is a $n * 1$ bias vector. we can also simplify the formula by adding a bonus 1 to the end of x vector, and turn W in to $n * (dim + 1)$, then we can simply calculate the formula as $\hat{y}(x) = Wx$.
After the calculation, we can predict the label by apply the softmax function to the output:

$$
\begin{aligned}
p(x) &= softmax(\hat{y}(x)) \\
&= \frac{e^{y_i}}{\sum e^{y_i}}
\end{aligned}
$$

then we choose the label with max value in $\hat{y}(x)$ as its label, Here we finish our prediction.
Then we can consider how to train the weight matrix W to get the best performance, here we use a loss function called **Cross Entropy Error Function**, it can be represented as:

$$
L = -\sum_{i=1}^{N} log(p(x_i)_{y_i})
$$

Our mission is to minimize the loss function. Here we can use the **gradient descent** method, we can calculate the partial derivatives from loss to W, and change it following the direction of the gradient, which will make the loss decrease fastest. Details as follow:

$$
\begin{aligned}
\frac{\partial L}{\partial W} &= \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W} \\
&= (\frac{e^{y_i}}{\sum e^{y_j}} - Onehot_k) * x^T
\end{aligned}
$$

the $Onehot_k$ is a vector that have only one 1 on the kth dimension, and k is its truth label. then, we can decrease the loss function by:

$$
W' = W - \theta * \frac{\partial L}{\partial W}
$$

where θ is a super parameter called learning rate. After gradient descent for times, we can have a good weight matrix W for classification.

Experiment Results

During the experiment, we will split the dataset into three part: 80% for train set, 10% for validate set, and 10% for test set. And the learning rate for

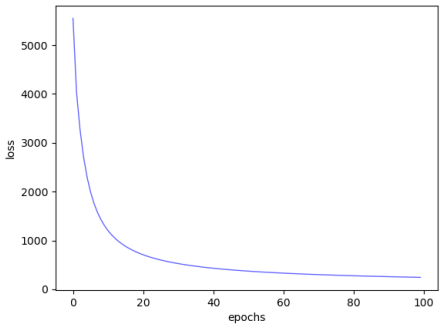
discriminative model is $1e-2$, the train epoch for discriminative model is 1000.
Here are the results:

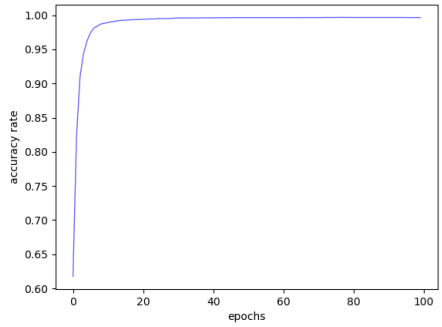
Accuracy	Train set	Test set
Generative model	0.996875	0.9975
Discriminative model	0.996375	0.9965

the detailed information is as follow:

Accuracy	A	B	C
GenerativeModel	0.9987	0.9968	0.9967
DiscriminativeModel	0.9964	0.9965	0.9965

And the loss and accuracy plot are as follows:





we can see that in this assignment, there are no big difference between generative model and discriminative model, both of them can finish the task almost perfectly.
You can recurrance the result by running following command in the terminal:

```
python source.py generate
python source.py generative_model
python source.py discriminative_model
```

and the result will be saved to the *gm_results.txt* and *dm_results.txt* respectively.

Generative vs Discriminative

From the method and result we mentioned above, we can conclude that the generative model assume that the data must follow some distributions, and then we calculate the best parameter by statistics methods. And the discriminative model transform our goal from maximize the likelihood function to minimize the loss function, it uses gradient descent to decrease the loss function, and increase the accuracy rate during the training.

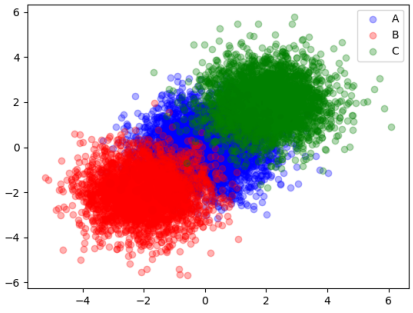
Reorganize data

In this section, we will reorganize the data by changing its distributions, prior probabilities, and figure out how it will infect the classification results.

First, we can the μ of our distribution: we change it to:

$$\mathcal{N}_A = \mathcal{N}(0,1) \quad \mathcal{N}_B = \mathcal{N}(2,1) \quad \mathcal{N}_C = \mathcal{N}(-2,1)$$

the plot figure and the results are as follow:

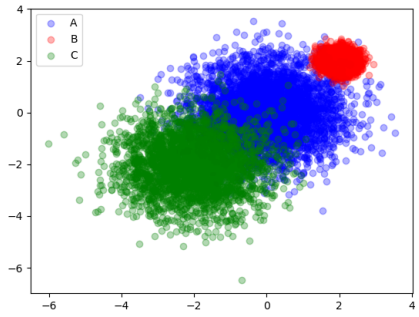


Accuracy	Train set	Test set
Generative model	0.894	0.889
Discriminative model	0.8935	0.89

We can find that since we move the , μ of distributions closer, their distribution will have some overlapping, so it's harder for the classification decision. So both of the train accuracy and test accuracy decrease.
Now we consider to change the δ^2 of the distributions. We change the distributions to:

$$\mathcal{N}_A = \mathcal{N}(0,1) \quad \mathcal{N}_B = \mathcal{N}(2,0.25) \quad \mathcal{N}_C = \mathcal{N}(-2,1)$$

the plot figure and the results are as follow:



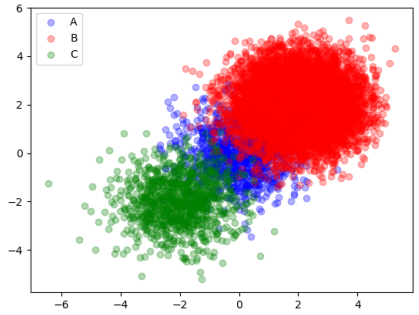
Accuracy	Train set	Test set
Generative model	0.921125	0.9285
Discriminative model	0.932375	0.9405

We can find that since we smaller the δ^2 of N_C, this measure decrease the overlapping of $\mathcal{N}_A$ and $\mathcal{N}_C$, so the classifier can discriminate label A and label C easier, so the accuracy increase comparing to the last experiment.
And at last, we can modify the proportion for each label in the data set, we change the setting to:

$$\mathcal{N}_A = \mathcal{N}(0,1) \quad \mathcal{N}_B = \mathcal{N}(2,1) \quad \mathcal{N}_C = \mathcal{N}(-2,1)$$

$$P_{label} = \{A : 0.1, B : 0.8, C : 0.1\}$$

the data plot and results are as follow:



Accuracy	Train set	Test set
Generative model	0.943875	0.95
Discriminative model	0.943875	0.952

We can find that after we change the proportion of the labels, the accuracy increases. That's because the label B occupies most of the data set, then even the classifier directly classify the example to label B, it will get the high accuracy. Under this situation, the accuracy rate increase, but that's not means the classifier becomes stronger. In that case, the classifier may predict worse comparing to the original parameter under the real world data.

Data reorganize conclusion

During the data reorganization, we can find that the data features is so important to machine learning method, since the model has few preset parameters, most of the parameters learned from the data itself. Both of the generative and discriminative algorithm are good, and their performances are almost the same in this assignment, so the performance is mainly depending on the data quality. Take the gaussian distribution as an example, we can find that if the data overlapping is small(in various senses, not only on the two dimension) and the data label proportion is relatively balanced, the performance will be better.

Reference

https://docs.scipy.org/doc/numpy/reference/
https://en.wikipedia.org/wiki/Normal_distribution