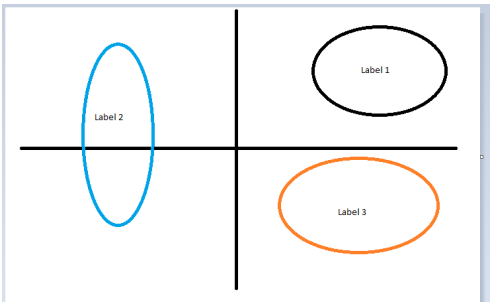
PRML Assignment1

1. Data Collection

1.1 Design

When designing the gaussian distribution , it is necessary that each can be seperated from others. Taking this into consideration, **2-D gaussian distribution** is suitable to construct the dataset. As is shown in the figure, each class compiled to a 2-D gaussian distribution with specific mu and sigma. We can modify these parameters in the follow-up experiment to dig deeper into the influence.



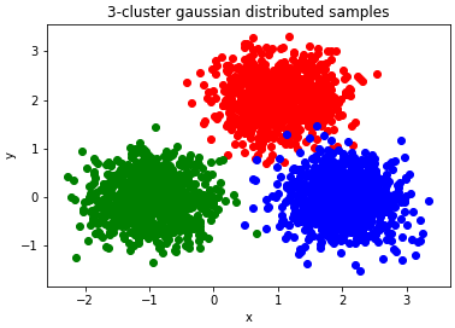
1.2 Preprocessing

First, generate three classes of gaussian distributed samples with specific expectation, covariance matrix and number(*gen_gaussian_clusters(ns,mus,sigma)*). Create dataset based on the gaussian distributions generated and shuffle it(*create_dataset(mus,sigma,ns)*). Then, write back the data and labels to the file and store it. Data and labels can be loaded from the file as arrays, with shape of (n,2) and (n,) respectively(*load_dataset(filename)*). Finally, encode labels into one-hot vectors so that computing loss and scoring are made easier.

1.3 Visualization

As is pointed in the figure, each class is marked with a unique color.

(mus=[[1,2],[-1,0],[2,0]], sigma=[[.2,0],[0,.2]], n=1000*3)



1.4 Link

https://pan.baidu.com/s/18jSoM6nGUzguMsRB19JSHQ (extract code: 1r1y)

2. Model

2.1 Discriminative Classifier

2.1.1 Principle

Discriminative classifiers model the posterior p(ylx) directly, or model the **decision boundary** between the classes. **Softmax Regression** works on multi-class classification. To train this linear discriminative classifier, **mini-batch gradient descent** is employed to minimize loss. In every iteration, choose batch_size of samples to compute loss and update the weight matrix. In every epoch, the dataset is reshuffled and similar procedures are redone. The decision boundary can be learned through this process.

2.1.2 Softmax

Softmax function is used to score each class. The higher the score, the more likely the sample is in this class. We have function $softmax(w^T x) = \frac{e^{w_c^T x}}{\sum_{\hat{c}} e^{w_{\hat{c}}^T x}}$. To avoid overflow and underflow, the formula can be improved as $softmax(w^T x) = \frac{e^{w_c^T x - max}}{\sum_{\hat{c}} e^{w_{\hat{c}}^T x - max}}$. ($W_c$ *is the weight vector for class c.*)

```
scores = X.dot(self.theta)
shift_scores = scores - np.max(scores, axis = 1).reshape(-1,1)
return np.exp(shift_scores) / np.sum(np.exp(shift_scores),
axis=1).reshape(-1,1)`
```

2.1.3 Gradient Descent

Cross-entropy loss is employed to calculate loss: $R(W) = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)})^T log\hat{y}^{(n)}$

```
softmax_output = self.softmax(X_batch)
loss = -np.sum(np.log(softmax_output[range(n), list(y_batch)]))
loss /= n
```

And the gradient is: $\frac{\partial(R(W))}{\partial(W)} = -\frac{1}{N}\sum_{n=1}^{N} x^{(n)}(y^{(n)} - \hat{y}^{(n)})^T$

```
y_actual = get_one_hot(y_batch,k).reshape(n,k)
dw = np.dot(X_batch.T,(y_actual-softmax_output.reshape(n,k)))
dw /= n
```

Update parameter: $W_{t+1} \leftarrow W_t + \alpha(\frac{1}{N}\sum_{n=1}^{N} x^{(n)}(y^{(n)} - \hat{y}_{w_t}{}^{(n)})^T)$

```
loss,grad = self.cal_cost_and_grad(mini_dataset,mini_labels)
self.theta += lr * grad
```

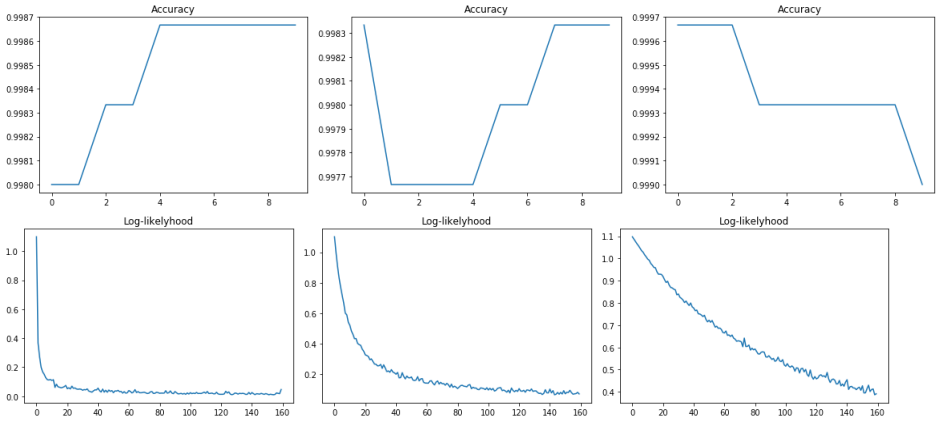
2.1.4 Hyper-parameters

- Learning rate

  Learning rate determines how fast or slow we will move towards the optimal weights.

  (batch_size=128, epoch=10, mus=[[0,2],[-2,0],[2,0]], sigma=[[.2,0],[0,.2]], n=1000*3 train : test=2 : 1)

  lr=0.9, 0.09, 0.009



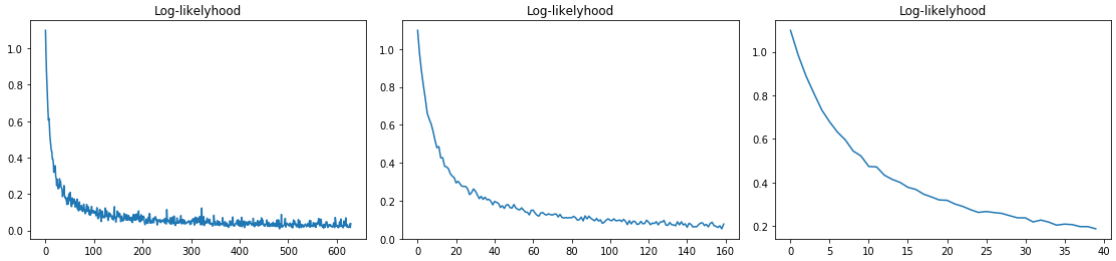
It can be concluded that learning rate should correspond to the capacity of dataset and epoch. Too small learning rate on a not that large training set will hinder convergence of loss function, thus decreasing accuracy.

- Batch_size

  The dataset is divided into numbers of batches to get trained and update, called iteration. One batch size is the sample capacity of each iteration.

  (lr=0.1, epoch=10, mus=[[0,2],[-2,0],[2,0]], sigma=[[.2,0],[0,.2]], n=1000*3 train : test=2 : 1)

  batch_size=32, 128, 512



batch_size	accuracy
32	99.7%
128	100%
512	99.9%

It can be concluded that as batch size increases, the training vibration decreases. The loss curve tends to be smoother. However, too big batch size decreases the iteration time simultaneously and causes lower accuracy with the same epoch.

- Epoch

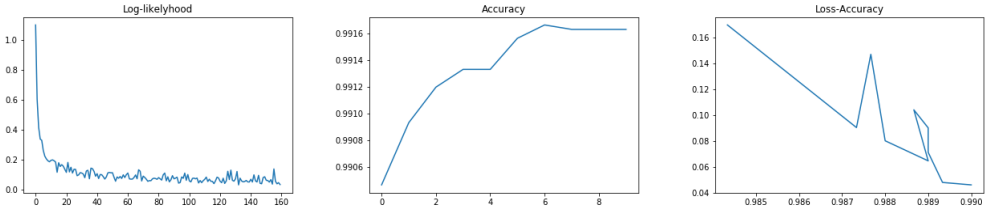
  One epoch is when an entire dataset is passed. If the learning rate is small or the batch size is big, epoch should be increased to get trained fully. But too big epoch may result in over-fitting and long training time.

.

2.1.5 Visualization

(lr=0.9, batch_size=128, epoch=10, mus=[[1,2],[-1,0],[2,0]], sigma=[[.2,0],[0,.2]], n=1000*3 train : test=2 : 1)

As is shown in the figure, log loss decreases as iteration increases. The trend of accuracy seems more obscure. Generally, it increases as iteration increases. The accuracy on the test dataset is 98.93%. Log loss slowly decreases when the predicted probability approaches 1.



2.1.6 Summary

It can be concluded that discriminative classifier tackles classification problem directly. The input dataset yields a decision boundary as output through training. The drawback is it can't generate new samples and relationships between variables are not explicit.

2.2 Generative Classifier

2.2.1 Principle

Generative classifiers learn a model of the **joint probability**, p(x, y), of the inputs x and the label y, and make their predictions by using **Bayes rules** to calculate p(ylx), and then picking the most likely label y. First, the dataset is assumed to follow gaussian distribution. Then, **parametric density estimation** is employed to calculate mu, sigma. Finally, use the **probability density function** to calculate the probability for each class.

2.2.2 μ

$$\mu = \overline{X} = \frac{1}{N}\sum X_{(a)}$$

```
for i in range(n):
    self.ps[labels[i]] += 1
    self.mus[labels[i]] += data[i]
for i in range(k):
    self.mus[i] /= self.ps[i]
    self.ps[i] /=n
```

2.2.3 Σ

$$S = \frac{1}{N}\sum (X_{(a)} - \overline{X})^T (X_{(a)} - \overline{X})$$

```
for i in range(n):
    self.sigma += np.dot(np.asmatrix(data[i]-
self.mus[labels[i]]).T,np.asmatrix(data[i]-self.mus[labels[i]]))
    self.sigma /= n
```

2.2.4 Bayes Formula

Since μ and Σ have been estimated, Bayes rule can be used to calculate P(Y |X). **Maximum likelihood estimation** is employed during this process.

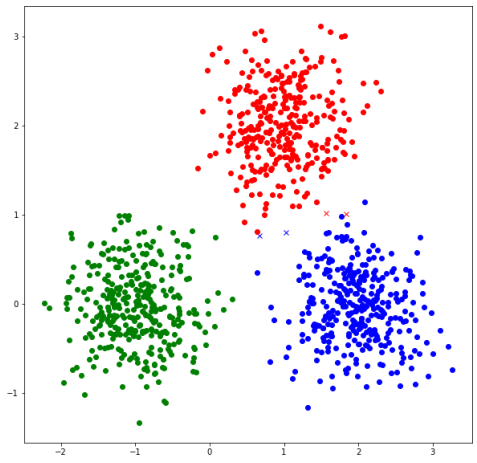
$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

```
for i in range(n):
    for j in range(k):
        prob[j]
=stats.multivariate_normal.pdf(data[i],self.mus[j],self.sigma)*self.ps[j]
    y_pred = np.argmax(prob)
```

2.2.5 Visualization

As is shown in the figure, the wrong cases account for a very small percentage. The accuracy is 99.4%.

(mus=[[1,2],[-1,0],[2,0]], sigma=[[.2,0],[0,.2]], n=1000*3)



2.2.6 Summary

It can be concluded that generative classifier explicitly models the actual distribution of each class. By substituting samples into the estimated distributions, it can find out the most likable class.

2.3 Comparison

2.3.1 Math

From the perspective of math, training classifiers involve estimating P(Y|X). To achieve this, the methods differ in the two models.

> **Generative classifiers**
>
> - Assume some functional form for **P(Y), P(X|Y)**
> - Estimate parameters of **P(X|Y), P(Y)** directly from training data
> - Use Bayes rule to calculate **P(Y |X)**
>
> **Discriminative Classifiers**
>
> - Assume some functional form for **P(Y|X)**
> - Estimate parameters of **P(Y|X)** directly from training data

2.3.2 Training

For discriminative classifiers, they learn a decision boundary by employing mini-batch gradient descent. Each iteration contributes to the optimization of the decision boundary.

For generative classifiers, they first assume the distribution form. Then, they learn sigma, mu of the distribution by employing parametric density estimation.

Based on the differences, it can be concluded that:

- When the generative **modeling assumptions do not hold**, the performance of the generative model will decrease sharply. In this case, discriminative classifiers perform better.
- When training **data is scarce**, generative models are preferred. Discriminative classifiers needs great amounts of data to learn the decision boundary, while generative classifiers requires fewer if the assumption holds.
- Naive Bayes is a learning algorithm with **greater bias, but lower variance**, than Logistic Regression. Because it pays less attention to the training data and oversimplifies the model by asserting a strong assumption. On the contrary, Logistic Regression only learns from training data and does not generalize on the data which it hasn't seen before. Each update of weight matrix depends only on previous label, and not future labels.

2.3.3 Classification

For discriminative classifiers, they use decision boundary and softmax function (**softmax(**$w^T x$**)**) to find the class with max probability.

For generative classifiers, they adopt probability dense function for gaussian distribution to classify the samples. Since we have:

P($c_k|X) = N(X, \mu, \Sigma) = \frac{P(X|c_k)P(c_k)}{\sum_j P(X|c_j)P(c_j)}$)=softmax($a_k$), $a_k = lnP(X|c_k)P(c_k) = w^T X + w_0$

Thus, they can be seen as using **softmax(**$w^T x + w_0$**)** to find the class with max probability.

2.3.4 Parameters

For generative classifiers, we have 2*M(μ_1, μ_2)+$\frac{M \cdot (M+1)}{2}(\Sigma) + 1(\pi)$ parameters. M is the dimension of samples.

For discriminative classifiers, we have M+1 parameters.

It can be concluded that **generative classifiers needs more parameters**, but they **can generate synthetic example data** with them accordingly.

2.3.5 Accuracy

In the experiment, the performance of generative classifier (99.4%) is a little bit better than the discriminative one (98.93%). The reasons may be:

- We already know the distribution and assume the form to be gaussian. That **correct assumption** promotes the performance of the generative classifier.
- There are some overlaps among the three classes. So, **it may be difficult for the discriminative classifier to learn the exact decision boundary**. That problem may be relived for generative classifiers because they learn the parameters for the three classes.

But in general, both of them do well in the test dataset.

2.3.6 Cost

For discriminative classifiers, the **time complexity for training** the model is very high.

For generative classifiers, it **takes time to estimate the parameters of O(**M^2**) scale.** M is the dimension of samples.

2.3.7 Application

Generative model involve modeling, discriminative models directly solve classification. Thus, generative models are more elegant, have explanatory power. They are more suitable for situations where samples are scarce, distribution form is known and generating new samples is required. Discriminative models are more desirable for pure classification problem with clear boundary and large training samples.

3. Analysis

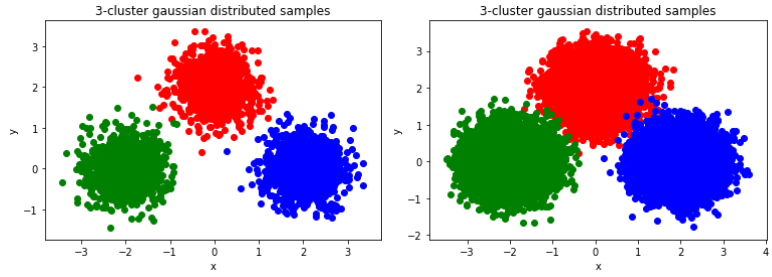
3.1 Overlap

3.1.1 n

If μ and Σ are fixed, increasing the scale of dataset may result in more overlaps.

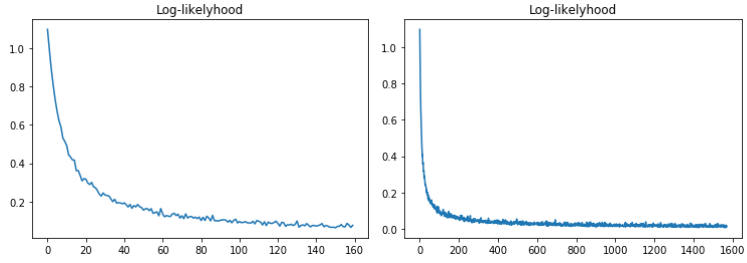
(lr=0.1, epoch=10, mus=[[0,2],[-2,0],[2,0]], sigma=[[.2,0],[0,.2]], train : test=2 : 1)

n=1000*3, 10000*3



For discriminative models, the accuracy drops from 100% to 99.9%. As is shown in the data distribution, lager scale results in blurred boundaries between classes. It's difficult for discriminative models to learn the decision boundary and tell the samples apart. The model can't learn an exact direction for gradient descent, **rendering the loss curve to vibrate**.

n=1000*3, 10000*3



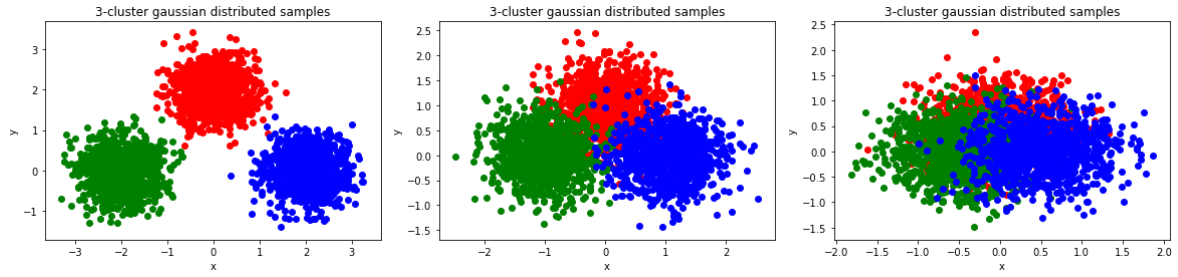
For generative models, the accuracy drops from 100% to 99.9% as well. As it adopts maximum likelihood estimation to classify, the blurred boundary among classes also disturbs classification.

3.1.2 μ

The mean for each class determines the position for it. If n and Σ are fixed, making μ for each class closer may result in more overlaps.

(lr=0.1, epoch=10, sigma=[[.2,0],[0,.2]], n=1000*3,training set =2000, test set=1000)

mus=[[0,2],[-2,0],[2,0]], [[0,1],[-1,0],[1,0]], [[0,.5],[-.5,0],[.5,0]]



mu	Discriminative model	Generative model
[[0,2],[-2,0],[2,0]]	99.8%	99.7%
[[0,1],[-1,0],[1,0]]	95.2%	92.9%
[[0,.5],[-.5,0],[.5,0]]	80.1%	70.5%

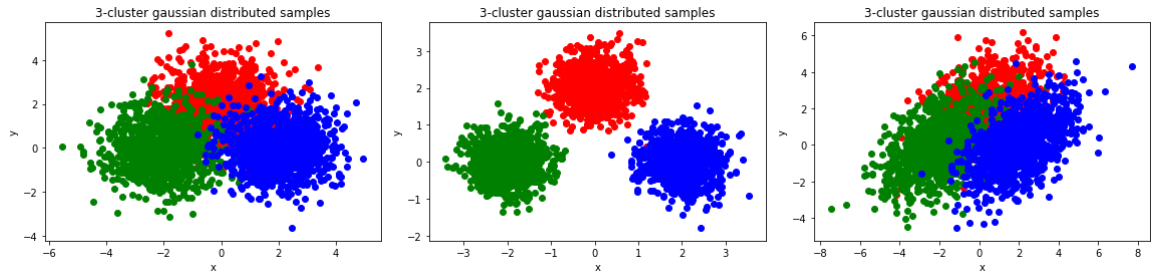
Accuracy for both models drop as overlap expands. **The expanding overlap caused by mu exerts more influence on generative models. Because it directly intervene the process of parametric density estimation of mu, which is decisive for generative models to classify.**

3.1.3 Σ

Covariance matrix is a symmetric matrix, which determines the shape of gaussian distribution. If n and μ are fixed, changing Σ to make classes more relevant may result in more overlaps.

(lr=0.1, epoch=10, mus=[[0,2],[-2,0],[2,0]], n=1000*3,training set =2000, test set=1000)

sigma=[[1,0],[0,1]],[[.2,0],[0,.2]],[[2,1],[1,2]]



sigma	Discriminative model	Generative model
[[1,0],[0,1]]	93.6%	89.8%
[[.2,0],[0,.2]]	99.9%	99.9%
[[2,1],[1,2]]	85.4%	77.5%

The analysis is quite same as expanding overlap caused by mu. **Generative models are more vulnerable to the expanding overlap caused by sigma.**

3.2 Scale

(lr=0.1, epoch=10, mus=[[0,2],[-2,0],[2,0]], sigma=[[.2,0],[0,.2]], n=10000*3,test set=2000)

train set	Discriminative model	Generative model
1000	99.7%	99.85%
5000	99.9%	99.85%
10000	99.8%	99.8%

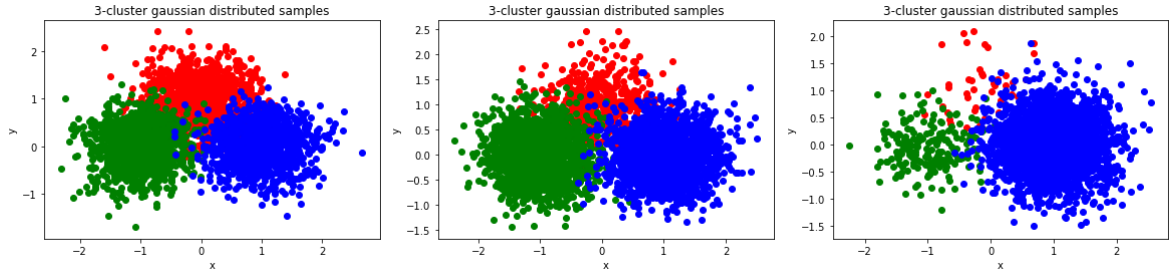
From these experiments, it can be concluded that **Discriminative Classifier depends more on the number of training samples.** It is improved more significantly as training samples grow. And this also coincides with the conclusion in *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes(2001)*, that is ***the generative model may approach its asymptotic error much faster than the discriminative model***. However, **too many training samples may result in over-fitting**, thus decreasing accuracy.

3.3 Proportion

In the previous experiments, the number for each class is the same. The uneven proportion for each class can also make a difference.

(lr=0.1, epoch=10, mus=[[0,1],[-1,0],[1,0]], sigma=[[.2,0],[0,.2]], n=3000,training set =2000, test set=1000)

1:1:1,1:2:3,1:4:55



proportion	Discriminative model	Generative model
1:1:1	95.2%	92.9%
1:2:3	95.6%	93.9%
1:4:55	98.2%	98.7%

As the gap scale grow striking, the accuracy on the test set is improved. A reasonable explanation may be the less the class account for in the training set, the less the class appears in the test set. On one hand, the under trained class contributes less error rate since it account for a small part in the test set. On the other hand, the majority class is well-trained. The conclusion especially holds for generative model.

4. Discussion

In the previous experiments, the covariance matrix for the three gaussian distributions is expected to be the same. This assumption is based on the irrelevance between covariance matrix difference among classes and the classification problem. It remains to be proved further.

5. Demo

```
python source.py
```

6. Reference

[1] 邱锡鹏《神经网络与深度学习》https://nndl.github.io/

[2] Andrew Y. Ng and Michael Jordan. On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes,2001

[3] Chapter 3: GENERATIVE AND DISCRIMINATIVE CLASSIFIERS: NAIVE BAYES AND LOGISTIC REGRESSION, Copyright c2017. Tom M. Mitchell. All rights reserved. *DRAFT OF April 9, 2020*

[4] Machine Learning: Generative and Discriminative Models, Machine Learning Course: http://www.cedar.buffalo.edu/~srihari/CSE574/index.html

[5] https://medium.com/@raghavan99o/discriminative-vs-generative-models-c416e53317a7