2020 Pattern Recognition and Machine Learning Technical Report

Ruian He, 16307110216,

I. Task Description

USING the Bayesion Decision Theory for handwritten recognition based on MNIST dataset available at mnist website[1]. Here, assume that data follow Gaussion distribution.

II. Data Description

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. [2]

The data is stored in a very simple file format designed for storing vectors and multidimensional matrices.

A. Label File

The label file is like the following.The labels values are 0 to 9.

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
...	...	...	...
xxxx	unsigned byte	??	label

B. Image File

And the image file is as followed.Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
...	...	...	...
xxxx	unsigned byte	??	pixel

III. Data Preprocessing

First of all, we uncompress the `.gz` files to get ubyte files described as above. And we use struct module in python to extract the magic number, the image number, the row number and col number. Then we read $num*row*col$ numbers stored in unsigned bytes from the file which is the data. We can visualize using `matpoltlib` module.

Fig. 1. Sample Digit

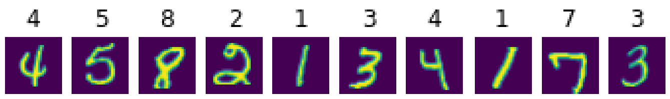
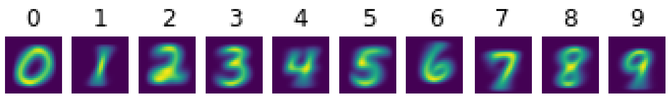


Fig. 2. Average Digit



We also can plot the average image for every digit, and get a general view over all training data.

Because we will run bayesian decision for $row*col$ features in a image, we also need to flat the matrix of the image to get a vector of $row*col$ length.

IV. Algorithm Introduction

A. Algorithm principle

In statistics, Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features.

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

Next, we start to build Bayesian Decision model from scratch. As the data follow Gaussion distribution, we can calculate the miu and sigma of the Gaussion distribution for every digit and we know the prior for each digit from the training set. [1]

Actually we can choose monovariable or multivariable gaussian distribution to describe. For the monovariable model, we assume that the features are irrelevant, and we can simply multiply the possibility $p_i(x_j)$ of every feature to get the likelihood L_i of every digit.

$$L_i = \prod_{i=1}^{784} p_i(x_j) = \prod_{i=1}^{784} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{x_j-\mu_i}{2\sigma_i}}$$

As for multivariable model, we use covariance to replace variance in monovariable model to describle the connection between features. And we only need to calculate one possibility for one point. For the 2π term is same among all likelihood, we can ignore it when calculating the posterior.

[1]http://yann.lecun.com/exdb/mnist/.

$$L_i = p_i(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{2}} e^{-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)}$$

When we predict, according to the bayesian theorem, we can get $posterior = likelihood * prior/evidence$ and use log on both sides. As we only need to compare the relative size, we can ignore that evidence which is the same for all posteriors. Then the final one to compare is $\log(prior) + \log(likelihood)$.The digit class which get the largest posterior will be the choice.

B. Algorithm implementation

When implementing the algorithm, we first extract the prior, mean, variance and covariance for each digit, and predict the label by calculating the likelihood of each digit using the above formulas.

For some reason, the model is overfitting on test set, so we must add some hyperparameter to the variance in order to smooth the gaussian distribution we predicted. Just like $\sigma' = \sigma + smooth$.

Listing 1. Bayesian Decision

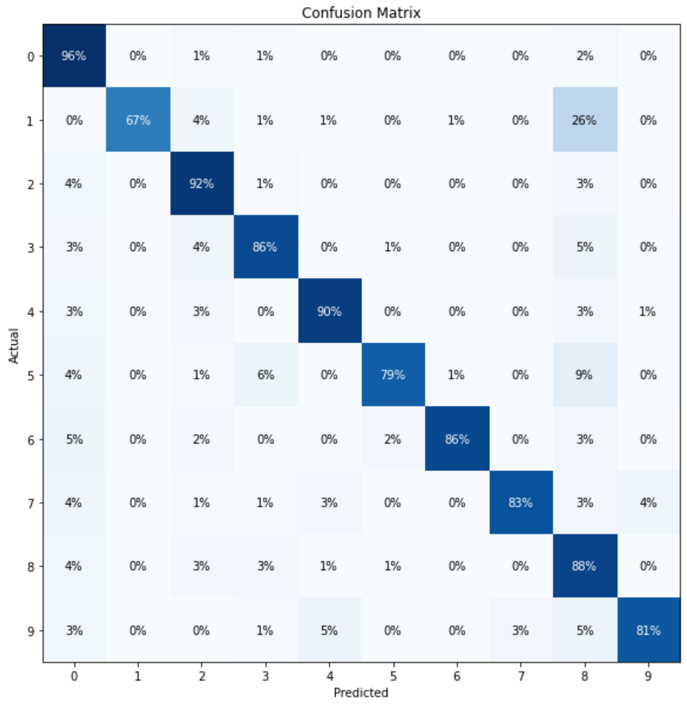
```
class BayesianDecision(object):
  def __init__(self):
    self.eps = 1e-5
    self.smooth = 1000

  def train(self, X, y):
    n_features = X.shape[1]
    self.prior = np.bincount(y)/y.shape
        [0]
    self.miu = np.zeros((10,n_features))
    self.var = np.zeros((10,n_features))
    self.cov = np.zeros((10,n_features,
        n_features))
    for i in range(10):
      select = X[np.where(y == i)]
      self.miu[i] = np.mean(select,axis
          =0)
      self.var[i] = np.var(select,axis=0)
           + self.smooth
      self.cov[i] = np.cov(select.T) +
          self.smooth

  def predict(self, X, mode):
    likelihood = np.zeros((10,X.shape[0])
        )
    for i in range(10):
    diff = X - self.miu[i]
    if mode == 'multi':
      det_sqrt = np.sqrt(np.linalg.det(
          self.cov[i]))
      likelihood[i] = np.log(np.exp(np.
          diag(-1/2*diff.dot(np.linalg.
          pinv(self.cov[i])).dot(diff.T)))
          /(det_sqrt+1e-5))
    else:
      possibility = np.exp(-1/2*np.square
          (diff)/self.var[i])/np.sqrt(self
          .var[i])

    likelihood[i] = np.sum(np.log(
        possibility),axis=1)
    posterior = likelihood + np.log(self.
        prior+self.eps)[:,np.newaxis]
    return np.argmax(posterior,axis = 0)
```

V. EXPERIMENTAL RESULTS AND ANALYSIS

Now we get the model and the data,then we can start training and testing. We finally choosed $smooth = 1000$ in 0-255 gray scale space and get 0.815 on the monovariable model, and 0.8459 on the multivariable model.

Moreover, we can look into accuracies in every digit class and plot the confusion matrix. The following is the confusion matrix of the result the multivariable model produce. Comparing to the average image in 3, we can find that the zero(0) is far different from other digit and get the highest accuracy but the distribution of one(1) is so similar to that of the eight(8) that many ones were mistaken as eight. The result matched our expectations.

Fig. 3. Confusion Matrix



REFERENCES

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.