CS5487 Course Project Report

Huankang Guan
SID:56260695
EID:huankguan2
huankguan2-c@my.cityu.edu.hk

Xianbin Zhu
SID:56493233
EID:xbzhu5
xbzhu5-c@my.cityu.edu.hk

1. Introduction

Handwritten classification has been studied in the past decades extensively due to their important applications. For example, recognize handwritten digits on cheque. In this project, we mainly apply some existing techniques to address handwritten digit classification. First, We apply and implement various methods, *i.e*. Bayesian classifier, KMeans, KNN, SVM, regression, LDA, Second, We evaluate the selected methods on the provided dataset (a subset of MNIST), containing 4000 images. After that, we have a further analysis on the results by completing a series of studies and test the selected methods on the challenges dataset, which is provided by our course instructor.

2. Methodology

We choose a line of methods, *i.e*. Bayesian classfier, kmeans, KNN, SVM, regression, and LDA, to address this classification task. Also, we apply LDA and PCA as dimensionality reduction algorithm. In each following subsection, we mainly talk about the selected method's key technical points and the key ideas we adopt in our experiments.

2.1. Bayesian Classifier

Bayesian classifier using Bayes Theorem can be used to perform classification task fastly. We assume the distribution of each class is a multivariate Gaussian distribution. Thus, we have

$$p(x|y=j) = N(x|u_j, \Sigma_j)$$

where x is the input features, y is the corresponding label, u_j is the samples mean of class j, and Σ_j is the covariance matrix of the samples with label j. Based on Bayes Theorem, we have

$$\begin{aligned} p(y=j|x) &= \frac{p(x|y=j)\cdot p(y=j)}{p(x)} \\ &\propto p(x|y=j)\cdot p(y=j) \\ &= N(x;u_j,\Sigma_j)\cdot\frac{n\_sample_j}{n\_sample} \end{aligned} \tag{1}$$

where n_sample_j means the number of samples with lable j, and n_sample means the total number of all samples.

Since the probabilities obtained by above manner are small and need to handle exponent arithmetic, we take the log likelihood:

$$\begin{aligned} &ln(p(y=j|x)) \propto ln(N(x;u_j,\Sigma_j)\cdot\frac{n\_sample_j}{n\_sample}) \\ &= -\frac{d}{2}ln(2\pi) - \frac{1}{2}ln|\Sigma_j| - \frac{1}{2}(x-u_j)^T\Sigma_j^{-1}(x-u_j) \\ &+ ln(\frac{n\_sample_j}{n\_sample}) \end{aligned} \tag{2}$$

where d means the number of features dimensionality. When training, we need to calculate the u_j, Σ_j for each class, and in order to make the inferring process faster, we preserve the determant and the inverse of convariance matrix. When inferring, we feed the input features into each multivariate Gaussian distribution and select the one with largest possibility as the predict label.

2.2. kmeans

kmeans is an unsupervised clusting algorithm. It firstly choose K random points as centers and then assign each data points to one of the K clusters *i.e*. assign the closest cluster center to the data point. After the assignment process, update the cluster centers based on the existing data points and their labels. Keep updating the cluster centers iteratively and, finally, we divide all the data points into K clusters.

But in this classification task, we do not run the kmeans algorithm to divide all the training points into K clusters, instead, we explicitly calculate each class center as the final cluster center. When inferring, we assign the data point with one of the K clusters according its Euclidean distance to each cluster centers. Such method can run really fast with time complexity O(K).

2.3. KNN

K-Nearest Neighbor (KNN) classifier is a very simple classification algorithm. KNN does not take much time to train as it only preserves all the training data in training stage. When inferring, given a testing data point (denoted as x_*), it goes through all the training data points to find K closest traning data points (denoted as $x_1, x_2, ..., x_k$).

the label of point x_* depends on these K training data points. Formally, we also need to define a decision function:

$$\hat{y_*} = f(x_1, x_2, ..., x_k) \tag{3}$$

where $\hat{y_*}$ is the predict label of x_*. A very common way to determine the testing data point's label is based on the amount of samples of each class among these K points. We go a step further by considerring both the number of samples of each class and the average distance. Mathematically,

$$\hat{y_*} = \arg\min_{j \in C} \frac{\left(\sum_{i=1}^{i=K} d_i \cdot I[y_i == j]\right)^{\alpha}}{\left(\sum_{i=1}^{i=K} I[y_i == j]\right)^{\beta}} \tag{4}$$

where C contains all the labels appearing in $x_1, x_2, ..., x_k$, y_i is the label of x_i, d_i is the distance between x_i and x_*, I[*] is an indicator function whose value equal to 1.0 when * is TRUE, otherwise return 0.0 and, α, β are the hyper-parameters. We can control α, β to add/reduce weights to distance/amount. When $\alpha = 0, \beta = 1$, it means that we only consider the number of samples of each class among these K points, ignoring distance. When $\alpha = 1, \beta = 1$, it means that we consider the average distance. While increasing β, we shift much attention to the amount rather than the distance.

2.4. SVM

Support Vector Machines(SVM) is a type of supervised learning for regression and classification of data. SVM can be used in linear classification and non-linear classification. Non-linear SVMs in this project include BRF-kernel SVM and Poly-kernel SVM.

Linear SVM The goal of linear SVM is to obtain a hyper-plane to divide the training data into two classes with a maximum margin and thereafter use this model to predict the test data with acceptable bias and covariance. Generally, if the margin between the two classes and the hyperplane is larger, the error of recognizing the classes incorrectly is smaller(In this project, the classes are the digits from 0 to 9). The linear function is $f(x) = \omega^T x + b$. Let x denote the data point. If $f(x) > 0$, the data point would be in one side of the hyperplane and if $f(x) < 0$, the data point would be in another side. And thus two classes are separated.

In order to recognize ten digits, we use the one-versus-rest strategy. We train ten SVMs for ten digits. Then by transforming this problem to binary-classes recognition and repeating the above mentioned method ten times, we can recognize ten digits. For the feature vector that might be labelled more than once, the label with the highest confidence is chosen.

RBF-kernel SVM Radial basis function(RBF)-kernel as known as Gaussian kernel is a good way to solve the classification problem with SVM when the data is not linearly separable. The RBF is defined as follows[2], where a and b are two sample feature vectors.

$$k(a, b) = e^{-\frac{||a-b||^2}{2\sigma^2}}$$

Using the RBF-kernel we can transform the non-linear separable data into linear separable data. After this operation, we can treat this problem as Linear SVM.

Poly-kernel SVM Poly-kernel SVM is similar to RBF-kernel SVM. For this type of SVM, the kernel chosen is poly-kernel which can be defined in the following.

$$k(a, b) = (a^T b + c)^d$$

The rest of process follows the operations in Linear SVM.

2.5. Regression

2.5.1 Least Square Regression & Regularized Least Square Regression

Both least square regression (LS) and regularized least Square regression (RLS) can be used for classification. We will use RLS as explanation. Given a dataset D = $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ (assume that $y_i \in \{-1, +1\}$), we need to optimize the following objective function:

$$\hat{\omega} = \arg\min_{\omega} ||y - \Phi^T \omega||^2 + \lambda ||\omega||^2 \tag{5}$$

where $\Phi = [\phi(x_1), \phi(x_2), ..., \phi(x_n)]$, $y = [y_1, y_2, ..., y_n]^T$. We can derive $\omega = (\Phi\Phi^T + \lambda I)^{-1}\Phi y$. When inferring, if $\hat{y} = \omega^T x_* > 0$, then label the new instance x_* as +1. Otherwise, label x_* with -1.

Now, turning to multi-classes classification. On one hand, we can train multiple classifiers by applying one-vs-rest technic to address multi-classes classification problem. On the other hand, note that the groundtruth y can be extend to onehot form *i.e.* digit 0 can be expressed as [1,0,0...,0]. Therefore, we are able to predict a onehot output directly. The new objective function have the similar form as equation 5. But the shape of ω is $d \times 10$ instead of $d \times 1$. After the derivation, we found that it has the same solution as two-classes RLS.

Similarly, we can derive the solution for LS, $\omega = (\Phi\Phi^T)^{-1}\Phi y$. We note that RLS is LS when $\lambda = 0.0$. Thus, we only implement RLS in our experiment and implement RLS with one-vs-rest(or one-vs-all) technic and RLS with extending y to onehot form, separately.

2.5.2 Logistic Regression

In this part, we would first introduce the principle of logistical regression and then give the description on how to recognize digits by this method.

Logistic Regression also known as Logit Regression is a very important classifier in the field of machine learning. It uses the probability to predict a certain class (e.g., win/lose, good/bad, 0/1). In another word, the Logistical Regression can be seen as a function between the variable and the likelihood that it belongs to a certain class. This function is sigmoid function:

$$\text{sigmod}(x) = \frac{1}{1 + e^{-f(x)}}$$

In the above function, $f(x) = \omega^T X$, where $\omega = [\omega_1; \omega_2; \ldots; \omega_n]$, $X = [x_1; x_2; \ldots; x_n]$. It is clear that $f(x)$ is a linear function and ω is the coefficients. The sigmoid function is used to predict the probability.

To simplify the computation, taking the log of the probability is a general method. To avoid overfitting, there are two main methods L_1 and L_2 regularization.

$$L_1(\omega) = \Sigma_{i=1}^{N} \log(1 + \exp(-y_i \omega^T x_i)) + \frac{\lambda}{2}||\omega||$$

$$L_2(\omega) = \Sigma_{i=1}^{N} \log(1 + \exp(-y_i \omega^T x_i)) + \frac{\lambda}{2}||\omega||_2$$

Using Logistic Regression can only classify two classes. To recognize ten different classes(digits), the one-versus-rest strategy is applied.

2.6. PCA

Principal component analysis (PCA) can be used for dimensionality reduction. It can be shown that the principal components are eigenvectors of the data's convariance matrix. Thus, we implement PCA by solving for eigenvectors and eigenvalues of data's covariance matrix. Formally, given data D = $\{(x_1, y_1), ..., (x_n, y_n)\}$, we derive covariance matrix Σ and solve for eigen pairs (v_i, λ_i). Finally, the output features y = $[v_1^T, v_1^T, ..., v_k^T]^T \cdot x$.

2.7. LDA

Linear discriminant analysis (LDA) is a supervised linear classifier, which could be also used for dimensionality reduction. Therefore, we not only consider LDA as a classifier, but also use it to reduce feature dimensionality. LDA try to minimize the intra-class scatter and maximize the inter-class distance. For multi-classes classification [1], LDA need to maximize an objective function with following form:

$$J = \frac{det(\phi^T \Sigma_b \phi)}{det(\phi^T \Sigma_w \phi)} \tag{6}$$

where ϕ is the transform function we need to solve, Σ_b is the inter-class scatter ($\Sigma_b = \sum_{i=1}^{n} m_i(\bar{x_i} - \bar{x})(\bar{x_i} - \bar{x})^T$), and Σ_w is the intra-class matrix ($\Sigma_w = \sum_{i=1}^{n} \sum_{x \in c_i} (x - \bar{x_i})(x - \bar{x_i})^T$). It could be obtain ϕ by solving a generalized eigenvalue problem ($\Sigma_b \phi = \lambda \Sigma_w \phi$).

After getting ϕ, it can perform classification task by solve for $\arg\min_k d(z\phi, \bar{x_k}\phi)$, given a new instance z, where $\bar{x_k}$ is the centroid of k-th class.

2.8. Our Solution

We observe that some digits are really difficult for models to classified. Looking at the Figure 1, we can find some pair digits are confused. For example, model is prone to predict digit 5 as digit 3, and predict digit 3 as digit 5. According to this observation, we divide 10 digits into 5 pairs, *i.e.* (3,5), (4,9), (2,8), (1,7) and (0,6). Thus, we perform digit classification task in two steps. In the first step, we train a logistic regression classifier to determine which pair the new instance should belong to. In the second step, we train five logistic regression classifiers for each pair. Then we could be able to know the label of the new instance.

The reason for our pipeline is that two similar digits may be classified easily if we have a particular classifier for it. It is different from one-vs-rest and one-vs-one tricks, that our pipeline need some extra prior information(which pair of digits are similar).

We also change the logistic regression to KNN in our solution to see if the performance would become better or not.

	0	1	2	3	4	5	6	7	8	9
0	187	0	6	0	0	5	1	1	0	0
1	0	191	0	0	0	1	1	0	7	0
2	2	5	171	7	2	1	2	3	7	0
3	1	0	10	167	2	12	2	1	3	2
4	1	0	2	0	165	2	0	1	2	27
5	3	3	1	21	5	149	4	1	9	4
6	1	0	4	0	4	11	179	0	1	0
7	0	1	6	0	3	0	0	183	1	6
8	4	1	9	6	5	8	1	3	156	7
9	3	1	4	4	5	3	0	4	9	167

Figure 1. Predict results of Logistic regression. ceil (i,j) means the number of digit i is predicted as j.

3. Experiments

3.1. Benchmark datasets

The provided dataset is a subset of MNIST dataset. The dataset has 10 classes (digit 0 through 9) and each class has 400 images (totally 4000 images). Each sample is a small image with shape of 28x28.

3.2. Implemention Details

We follow the course project's intruction dividing the whole dataset into two groups. For each trial, one of these two groups will be divided into training dataset and validation dataset. We adopt k-fold cross validation method to tune the model. Empirically, we set k = 10. Then test the trained model on the test dataset.

3.3. Quantitative Comparison

We compare the selected methods using accuracy. The results are shown in Table 1. Among the compared methods, SVM with RBF kernel work best and applying PCA on SVM with polynomial kernel can boost the performance. Regard to the LDA as feature transform function, it seems that it does not help to improve the performance. In our opinions, LDA classifier achieve 78.20% averagely on these two trials. So it fails to help other classifiers to improve the performance if other classifiers can work better than 78.20% in the original space. To prove our asumption, we adopt manhattan distance in kmeans (denoted as kmeans with Manhattan distance), and compare the performance under different settings (Table 2).

3.4. Further Analysis

We draw the mean images of each digit in Figure 2.

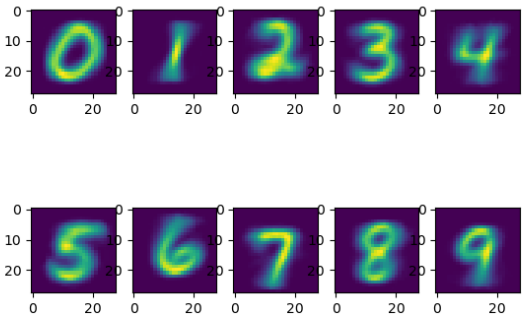

Figure 2. The mean image of each digit from 0 through 9

3.4.1 Bayesian Classifier

Bayesian classifier treats each class as a Gaussian distribution and select the one with highest possibility as predicted result. Figure 3 shown per-class-error-rate. Digit 5, 8 and 3 confound the Bayesian model very much. It makes sense since digit 5, 3 and 8 are much similar than other digits (Figure 2), while digit 0 and 1 are discriminative.

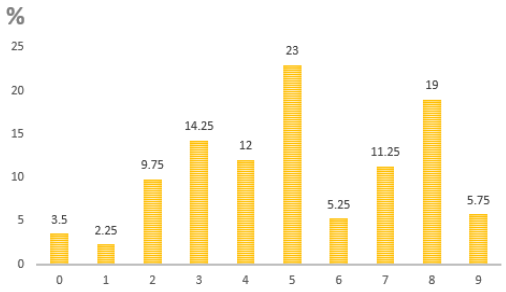

Figure 3. Per-class-error-rate of Bayesian Classifier

3.4.2 kmeans

kmeans algorithm assign the new instance with cluster which is closest to it. As shown in Figure 4 and 5, we can see that the error rate of digit 5 and digit 8 are really high and these two digits have a very similar characteristic in first 2 pricipal components. Obviously, it would not be a good choice to only utilize the center of each class to distinguish the new instances. We should consider much more information, like Bayesian classifier also take covariance into consideration.

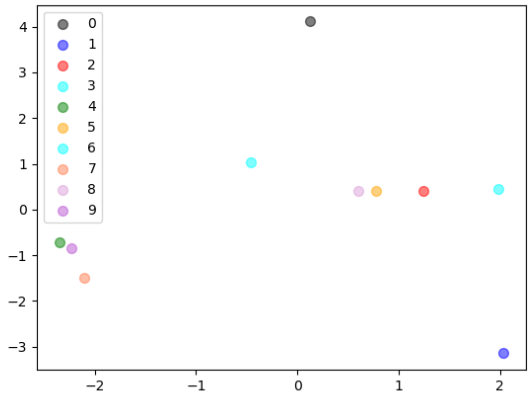

Figure 4. The cluster centers of kmeans algorithm (Apply PCA=2).

3.4.3 KNN

We try different α and β to see if the performance would become better or not. We can obtain the best results if taking both average distance and number of each class into account. And shift much attention to the number of each class has little effects (Table 3).

Plus, we consider the how K affect the performance. We try different K, and show the results in Table 4. It seems that K does not affect results a lot. And larger K may hurt

Table 1. Quantitative Comparison Using Accuracy (times 100). Each method adopt the best settings by tunning on the validation dataset

Method	trial-1	trial-2	mean(std)	PCA(n_component=200)	LDA(n_component=9)
Bayesian	90.05	88.75	89.40(0.65)	64.33	79.55
kmeans	81.80	78.85	80.38(1.42)	80.33	78.20
KNN($K=5, \alpha=1, \beta=2$)	92.55	91.05	91.80(0.75)	92.05	79.80
LDA	78.45	77.95	78.20(0.25)	84.83	78.20
SVM(Linear)	90.10	89.95	90.03(0.08)	89.65	78.40
SVM(RBF)	94.10	93.40	93.75(0.35)	94.40	78.33
SVM(Poly)	91.55	89.20	90.38(1.17)	94.20	76.75
Logistics Regression	88.10	87.50	87.80(0.30)	87.85	78.15
RLS(one-vs-rest)	85.15	84.20	84.68(0.48)	84.68	76.68
RLS(onehot)	85.15	84.20	84.68(0.48)	84.68	76.68
ours(logistic regression)	83.45	80.70	82.08(1.38)	82.15	76.20
ours(KNN)	91.95	91.20	91.58(0.375)	91.85	79.80

Table 2. kmeans-with different distance metrics. Quantitative Comparison Using Accuracy (times 100). kmeans with Manhattan distance perform really bad in original space, but its performance get a significant improvement when applying LDA since LDA can obtain a accuracy of 78.45%.

Method	trial-1	trial-2	mean(std)	PCA(n_component=200)	LDA(n_component=9)
kmeans(Euclidean distance)	81.80	78.85	80.38(1.42)	80.33	78.20
kmeans(Manhattan distance)	65.65	63.15	64.40(1.25)	80.13	76.93

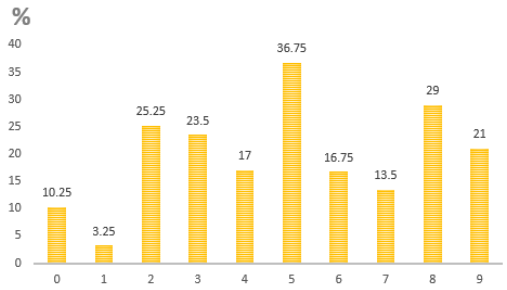

Figure 5. Per-class-error-rate of kmeans model.

Table 3. KNN algorithm. Fix K=20 and we try different α and β as mentioned in section 2.3. N means the number of each class, avg distance is the average distance of each class.

method	accuracy	description
$\alpha=0, \beta=1$	88.13%	N
$\alpha=1, \beta=1$	83.03%	Avg Distance
$\alpha=1, \beta=2$	88.88%	$\propto \frac{avg\ distance}{N}$
$\alpha=1, \beta=3$	88.70%	$\propto \frac{avg\ distance}{N^2}$
$\alpha=1, \beta=4$	88.65%	$\propto \frac{avg\ distance}{N^3}$
$\alpha=1, \beta=5$	88.65%	$\propto \frac{avg\ distance}{N^4}$

the performance, since larger K may introduce some wrong points.

Table 4. KNN algorithm. Fix $\alpha=1, \beta=2$, try different K

K	accuracy
1	91.60%
2	91.60%
3	91.65%
5	91.80%
10	91.00%
20	88.88%

3.4.4 SVM

SVM adopts different kernels may lead to a large difference. It seems that RBF kernel work better, while linear kernel perform worst. It is worthnoted that, as for RBF and polynomial kernels, applying PCA=200 may help to improve the performance. The similar situation can be seen in RLS and LDA. In our view, that is because some components could be a distraction for the classifiers. Thus, reducing some trivial components can lead to a better performance.

We compare the per-class-error-rate with different kernels. As shown in Figure 6, we can see that applying PCA on SVM with polynomial kernel can help the model to discriminate digit 5, 8 better. We explore the impact of number of component of PCA by trying different number of components on SVM with polynomial kernel (Figure 7).

3.4.5 Regression

RLS performs classification task by predicting a target value. If such value larger than 0.0, then it would be clas-

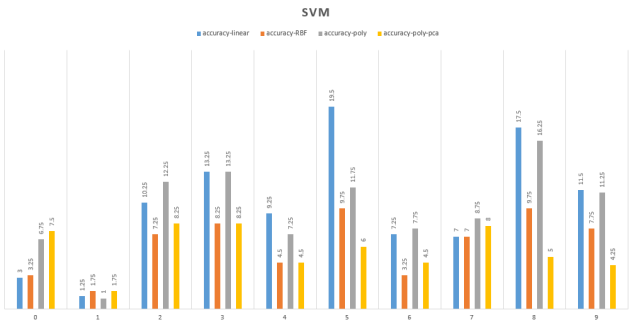

Figure 6. Per-class-error-rate of SVM with different kernels.

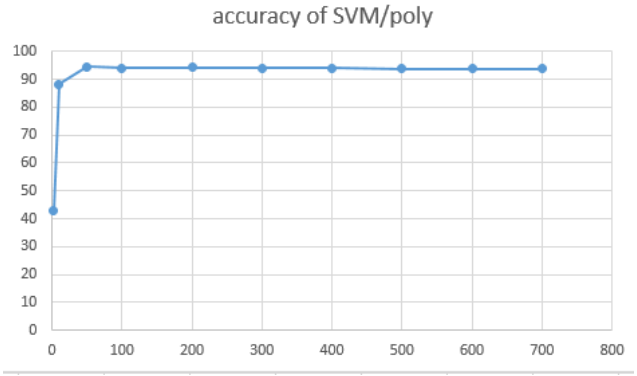

Figure 7. The accuracy of SVM with poly kernel under different number of components (PCA).

sified to positive cluster. Or it would label the new instance as negative. Since the objective function would penalize the "too correct" points, it may mis-regconize the points near the boundary. To address such problem, logistic regression use a sigmoid function to transform target value to [0.0,1.0], which avoids to penalize the "too correct" points. And the results show that logistic regression does perform better.

Besides, we have a further exploration on how regulization term affect the performance (Table 5). For RLS, regulization term affects performance a lot. But we do not observe a large fluctuation in logistic regression.

4. Bonus Challenge

We test the selected methods and our method on the challenging dataset and Table 6 shows the results.

Table 5. RLS & logistic regression. Different lambda values affect the performance.

lambda	RLS-accuracy	Logistic regression-accuracy
1e-3	77.13	85.83
1	80.53	87.80
10	83.15	88.45
50	84.53	87.10
80	84.68	86.75
100	84.68	86.43
150	84.63	85.90
200	84.28	85.50
1000	83.25	80.85
10000	78.93	74.73

Table 6. Quantitative Comparison Using Accuracy on challenging dataset

Method	Accuracy
Bayesian	44.00
KMeans	57.33
KNN	70.67
LDA	54.67
SVM(linear)	68.00
SVM(RBF)	72.00
SVM(Poly)	70.67
logistic regress	55.33
RLSovr	57.33
RLSonehot	57.33
ours(LR)	61.33
ours(KNN)	67.33

References

[1] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. Using discriminant analysis for multi-class classification: an experimental investigation. *Knowl. Inf. Syst.*, 10(4):453–472, 2006. 3

[2] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70, 2004. 2