MATH 4432 Mini Project 1: Comparative analysis of 4 models on NIPS Dataset

Zizheng Lin[1] and Yanbang Wang[2]

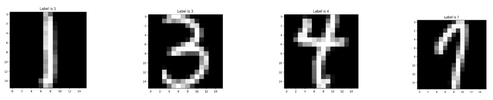
[1]Department of Mathematics, HKUST
[2]Department of Computer Science and Engineering, HKUST

1. Introduction

Within the realm of machine learning, MNIST has long been a classic data set that could effectively test many learning algorithms for computer vision. This set of data introduces a very fundamental classification task of identifying the handwritten digit formatted in bitmap images. More rigorously, given a 16x16 grayscale image (in fact the most popular version is sized 28x28), the algorithm is expected to correctly identify it into one of the ten categories of decimal digit, namely 0,1,2,...9. While the state-of-the-art method nowadays primarily employs convolutional neural network to extract the features and make classification, it is also worth noting that traditional classification algorithms could also make their own contribution. In this study, we tested a couple of different learning algorithms on this MNIST data set, carefully compare and the empirical study results, and make comment about the direction for future improvement.

2. MNIST Dataset

The given data is in the form of .csv file, where each row (I.e. labeled image) starts with a label, followed by an array of length 256 (16 * 16 pixels). Each individual value is a normalized grayscale value falling between -1.0 and 1.0, which maps to the 0-255 grayscale uniformly. Therefore, basically pre-processing phase only involves reading the data and transforming them into appropriate format and data structure. OpenCV was employed as a supplementary visualization tool to draw the image and make sense of it.



As has been introduced by MNIST documentation, there are 7291 training observations and 2007 test observations, distributed as follows:

	0	1	2	3	4	5	6	7	8	9
Train	1194	1005	731	658	652	556	664	645	542	644
Test	359	264	198	166	200	160	170	147	166	177

Note that the 256 dimensions of each sample are directly treated as features (i.e. predicators) in the following analysis. There is no need to apply computer vision techniques like pre-whitening and noise removal, since the grayscale images are already formatted in a noise-free and low-resolution manner.

3. Metodology

3.1 Linear Discriminant Analysis(LDA)

This classification method bases itself on the Bayesian probability theory, picking the class that enable the sample to appear with highest probability, by estimating class distribution and the conditional probability of the sample's appearance given the class label.

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l)\cdot P(y=l)}$$

An assumption is made that the data should follow a multi-dimensional Gaussian Distribution, with all classes share a common covariant matrix of predicators:

$$p(X|y=k) = \frac{1}{(2\pi)^n|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t\Sigma_k^{-1}(X-\mu_k)\right)$$

While there is, generically, not much to tune about this method, there conveniently exists a shrinkage algorithm (Ledoit and Wolf) that estimates the covariant matrix in a more effective manner than that of the empirical sample estimation. In empirical study, "auto" parameter was set for this method, indicating an automatic search for the optimal shrinkage coefficient.

3.2 Quadratic Discriminant Analysis(QDA)

Similar to LDA, QDA also employs the Bayesian Probability Theory and makes the assumption that the data generically follow Gaussian distribution. A fundamental modification lies in that QDA does not share a common covariance matrix among all predicators, which LDA does. The matrix is, instead, estimated separately for each class, which results a quadratic decision surfaces.

$$p(X|y=k) = \frac{1}{(2\pi)^n|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t\Sigma_k^{-1}(X-\mu_k)\right)$$

4. Logistic Regression

Logistic Regression models the probability odds ratio of belong to certain class as linear combination of covariates, which gives rise to the following prediction function:

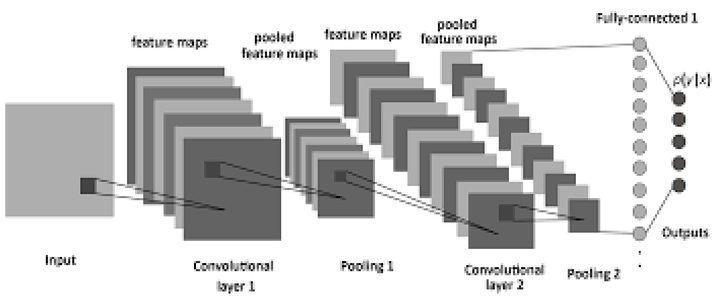
$$\Pr(Y_i = y \mid \mathbf{X}_i) = {p_i}^y(1-p_i)^{1-y} = \left(\frac{e^{\beta\cdot\mathbf{X}_i}}{1+e^{\beta\cdot\mathbf{X}_i}}\right)^y\left(1-\frac{e^{\beta\cdot\mathbf{X}_i}}{1+e^{\beta\cdot\mathbf{X}_i}}\right)^{1-y} = \frac{e^{\beta\cdot\mathbf{X}_i\cdot y}}{1+e^{\beta\cdot\mathbf{X}_i}}$$

Since there are 10 different classes, we adopt one-versus-rest approach to make the classification. Specifically, for each sample we compute its probability of belonging to certain class and classify it to be the one that maximize the probability value.
After performing hyperparameter search, we figured out the optimal hyperparameter settings to be follows: inverse of regularization strength C as 0.3, penalty option as L2 and optimization algorithm to be limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS). Hence the objective function of each class would be the followings:

$$\min_{w,c} \frac{1}{2}w^Tw + C\sum_{i=1}^{n}\log(\exp(-y_i(X_i^Tw+c))+1).$$

4.1 Convolutional Neural Network

We utilize one of the most popular and effective model for image recognition which is Convolutional Neural Network(CNN) [1] on MNIST dataset. The deep learning algorithms generally depend heavily on the architecture design, which can be conceptually described as following graph. In terms of CNN, its impressive performance in classifying images may be attributed to its feature extraction capability in convolutional layers, and then feature reconstruction capability in pooling layers. The trick of randomly dropping out specific input data during each iteration also enhance the possibility of removing noise of input images.



Careful experiments and tuning lead to our final architecture of the CNN model: 2 consecutive convolutional layers followed by 1 max-polling layer, and another 2 consecutive convolutional layers with 1 max-polling layer, and we also introduce artificial noise to input feature so as to create more data to feed model. Other hyperparameter setting may be referenced in our source code.

5. Result and Analysis

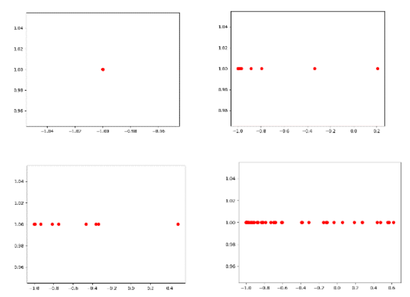
The performance of each model is included in following table:

Model	Test Accuracy
Logistic Regression	0.91330
CNN	0.97608
LDA	0.88341
QDA	0.81714

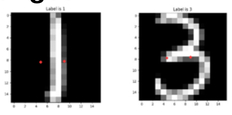
5.1 QDA and LDA

As has been manifested by empirical study, CNN prevails in the task with overwhelming lead in classification algorithm. Meanwhile, traditional methods like LR and LDA, have also achieved a generally satisfying result, though they still exhibit in this task some problems that might lie within the very basis of their theory assumptions. First off, the assumption held by LDA and QDA that each pixel in the handwritten image should follow (approximately) normal distribution seems not soundly based. We pick a pixel, (indexed 128, which pins right at the center of the image) and visualize its grayscale distribution in 4 classes (1,2,3,8). Obviously, the majority of the pixel's values sampled does not follow the normal distribution even in the roughest sense.

First off, the assumption held by LDA and QDA that each pixel in the handwritten image should follow (approximately) normal distribution seems not soundly based. We pick a pixel, (indexed 128, which pins right at the center of the image) and visualize its grayscale distribution in 4 classes (1,2,3,8). Obviously, the majority of the pixel's values sampled does not follow the normal distribution even in the roughest sense.



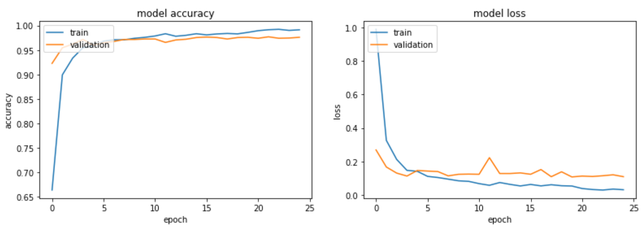
Second, it also remains highly questionable as to whether the covariant matrix could really be shared among classes. An intuitive analysis would give rise to the sense that, in different digits, the correlation between the grayscale of two fixed pixels would vary a lot. For example, in digit "3", two pixels at the red positions might have very strong correlation, meaning that the increase in intensity in one pixel would very significantly indicate an accompanied increase in intensity of the other pixel. On the other hand, the same positions in digit "1" do not have such strong correlation, because the left pixel value at left position might well remain 0 in any digit "1", whereas the right pixel could vary a lot because it is more close to the body of the digit and thus more subjected to the grayscale changes.



To address the vulnerability of QDA, we look at its covariant matrix derivation process. Indeed, to derive the covariance matrix for 256 predicators (which is super high dimension), the available data is way far from being enough to achieve a decent estimation – there are in total $2^{16} \div 2 = 17768$ coefficients to be determined from a single class of training samples numbered only a few hundreds [2].

5.2 CNN

We trained the CNN model for 25 epochs and record the accuracy as well as its softmax loss value in following 2 graphs, where the validation refers to the result obtained on test data set.



The confusion matrix for classification is also included, which shows that the model has difficulty in distinguishing between digit '3' and '5', and between '4' and '9'.

```
[[355   0   1   0   0   0   1   1   0   1]
 [  0 256   0   0   2   0   3   3   0   0]
 [  1   0 196   0   0   0   0   1   0   0]
 [  1   0   2 159   0   4   0   0   0   0]
 [  0   1   0   0 190   1   0   1   0   7]
 [  3   0   0   3   0 154   0   0   0   0]
 [  0   0   0   0   1   0 168   0   1   0]
 [  0   0   2   0   2   0   0 143   0   0]
 [  1   0   0   0   1   0   1   0 163   0]
 [  1   0   0   1   0   0   0   0   0 175]]
```

As shown in the accuracy history, the accuracy of CNN model quickly converges above 0.95 within first 3 epochs, with a few fluctuation in the following epochs. One strange phenomenon is that training accuracy turns out to be lower than test accuracy in first 4 epochs. This might be explained by following reasonings: The adoption of noise data generation expands the training data set and also introduce some images that are more difficult to classify than the original one, thus lowering the performance of model on training data. However, such additional difficulties posed on training data actually effectively prevent the model from overfitting since the noise counteracts the original noise. This corresponds to reducing variance $E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$ of estimator in expected prediction error $EPE(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$

6. Conclusion

Based on the experiment results, we may conclude that CNN model is more suitable than the other 3 classifiers in terms of coping with image data. Such advantage is gained through its capability of extracting, abandoning and reconstructing hidden features through high dimensional image data. In contrast, the other 3 classifiers utilize all of the 256 pixels to compute the decision boundary, which is very prone to overfitting.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365 – 411, 2004.