STAT 535 Final Project Report

Anh Huynh

November 29th, 2013

Abstract. We report on application of two supervised learning models to the task of recognizing written digits. The first model is the k-nearest-neighbor, and the second one is the Gaussian discriminant analysis. We describe the feature set used, as well as complete model description and parameterization. We explain the training strategy and provide experimental results, which include the learning curve and validation losses for both models. Lastly, an estimate of the classification error is given. The best classification error attained by the k-nearest-neighbor is 5.76%, whereas for the Gaussian discriminant analysis, it is 15.5%. The second method, however, is markedly faster than the first one. All codes for this project is written by the author.

1. Project Objective

The goal of this project is to use two supervised learning models, namely, k-nearest-neighbor and Gaussian discriminant analysis, to learn the task of recognizing written digits. The available data include two sets trainingImagesA.npy and trainingImagesB.npy of 20000 images each. These are subsets of the MNIST handwritten digits. The images have size 28×28 and pixel-range between 0 and 255. The labels to these images, which explains which digit the image is supposed to depict, are given in two sets trainingLabelA.npy and trainingLabelB.npy.

2. Preprocessing

We can regard each image in the training set as a $28^2 = 784$-dimensional vector, whose components have range $[0, 255]$. As is famously known in the case of the k-nearest-neighbor model, at this scale, the curse of dimensionality is too great for the model to work well. We therefore seeked to reduce the dimensions naively as follow. We took 7×7 subsquares within the 28×28 image and computed the average value of the pixels within those 7×7 subsquares. To be precise, we used the subsquares whose sides project onto the axes as one of the following intervals:

$$[0, 6], [4, 10], [7, 13], [10, 16], [14, 20], [17, 23], [21, 27].$$

Thus, the 28×28 images are reduced to 7×7 images, or 49-dimensional vectors. In our experience this turns out to be adequate.

Example 1.

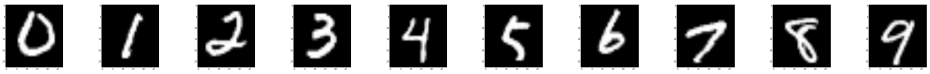


Figure 1. Sample images from trainingImagesA.npy

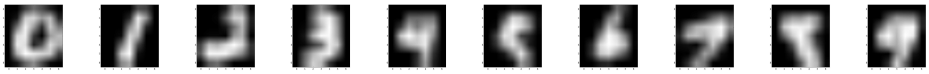


Figure 2. The same images, reduced.

The reason for choosing this feature set is as follows. First of all, even though this reduction blurs the images somewhat, it still preserves the relative position of the brighter pixels versus the darker pixels. Furthermore, observe that the area closed to the boundary of the images are invariably black. In our reduction of the images, there is a certain amount of overlapping of the subsquares on the middle part of the images, thus giving more weights to the more relevant part of the images. In Section 6, we will detail the effect of using the original images, using nonoverlapping 4×4 subsquares, and using overlapping 7×7 subsquares.

In the case of the Gaussian discriminant analysis model, we are faced with a different difficulty. Here we need to compute the covariance matrices and their determinants. It often turns out that with the original images, the determinants are too big for numerical computation. Therefore, we used the same image reduction procedure as for the k-nearest-neighbor model. Furthermore, we divide the value of every pixel in the picture by 4. This preserves the relative different between pixels, but reduce the value of the determinants that we need to compute. In our experience, a factor of 4 is sufficient for our classification task.

3. Predictors

3.1. k-nearest-neighbor model.

Our k-nearest-neighbor model relies on the Euclidean metric on the images. In other words, we flatten all images to 49-dimensional vectors and use the Euclidean norm

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_{49}^2}$$

as the metric. The model does not require any training time, but does the following at classification time.

Algorithm 1

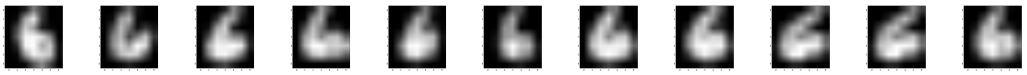
- Suppose given an image $v \in \mathbb{R}^{49}$.
- Run through all images in the training set, compute the distance between v and these images. Every time, update the set of k images whose distance to v is smallest.
- The result of the last step is the set of k nearest neighbors to v.
- Each image $u^1, \ldots, u^k$ has a vote for its own label. The label with most votes wins.
- In case of tie, choose the smallest label.

The best result is when we choose $k = 5$ and choose the training set to be 10000 randomly chosen images from trainingImagesA.npy. See Section 6 for more details.

Example 2. Take the following image trainingImagesB.npy, which has label 6.

Figure 3.

Its 11 nearest neighbors from the training set are



They all have label 6, therefore the model predicts the label of the test image to be 6.

3.2. Gaussian determinant analysis model.

The Gaussian determinant analysis model assumes that the images with the same label follow the multivariate Gaussian distribution. Namely, for each $i = 0, 1, ..., 9$,

$$\text{Prob}(v \,|\, \text{label} = i) = \mathcal{N}(\mu_i, \Sigma_i)(v)$$

where $\mu_i \in \mathbb{R}^{49}$ is the mean and $\Sigma_i \in \mathbb{R}^{49\times 49}$ is the covariance matrix. These parameters are estimated using the training set, see Section 4.

When predicting the label of a test image v, we follow the Bayes rule:

$$\text{Prob}(\text{label} = i \,|\, v) \sim \text{Prob}(v \,|\, \text{label} = i) = \mathcal{N}(\mu_i, \Sigma_i)(v) = \frac{\exp\left(-\frac{1}{2}(v-\mu_i)^T\Sigma^{-1}(v-\mu_i)\right)}{(2\pi)^{49/2}|\Sigma_i|^{1/2}} \qquad (1)$$

and gives v the label of the mode of $\text{Prob}(\text{label} \,|\, v)$. The best result we get is when using all 20000 images of trainingImagesA.npy for training. See Section 6 for more details.

Example 3. Take the following image from trainingImageB.npy.

Figure 4.

From the model we estimated using the training set, we have

$$\begin{aligned}
\text{Prob}(v \,|\, \text{label} = 0) &= 2.82 \times 10^{-94} \\
\text{Prob}(v \,|\, \text{label} = 1) &= 8.69 \times 10^{-176} \\
\text{Prob}(v \,|\, \text{label} = 2) &= 2.34 \times 10^{-83} \\
\text{Prob}(v \,|\, \text{label} = 3) &= 2.46 \times 10^{-79} \\
\text{Prob}(v \,|\, \text{label} = 4) &= 4.28 \times 10^{-137} \\
\text{Prob}(v \,|\, \text{label} = 5) &= 3.58 \times 10^{-77} \\
\text{Prob}(v \,|\, \text{label} = 6) &= 2.47 \times 10^{-234} \\
\text{Prob}(v \,|\, \text{label} = 7) &= 3.18 \times 10^{-278} \\
\text{Prob}(v \,|\, \text{label} = 8) &= 1.47 \times 10^{-51} \\
\text{Prob}(v \,|\, \text{label} = 9) &= 0
\end{aligned}$$

Therefore, we predict that the label of the test image is 8.

4. Training Algorithms

The models we use are relatively simple, there is straightforward and deterministic rules for prediction. There is no training for the k-nearest-neighbor. The Gaussian Discriminant Analysis requires the following training procedure. We need to estimate the mean μ_i and the covariance matrix Σ_i for each label $i = 0, 1, ..., 9$.

Algorithm 2

- Divide the training images into 10 sets S_i, $i = 0, 1, ..., 9$ where S_i consists of images with label i.
- For each i, we compute the mean by
$$\mu_i = \frac{1}{|S_i|} \sum_{v \in S_i} v$$
and the covariance matrix by
$$\Sigma_i[l, k] = \frac{1}{|S_i| - 1} \sum_{v \in S_i} v_l v_k$$

However, it might happen that there are components in all v that are the same, which will result in Σ_i with rows and columns of 0, whereas formula 1 can only be used for positive definite Σ_i. We deal with it by making the following change.

Algorithm 3

- Run through the rows of Σ_i to find any that consists entirely of 0.
- Suppose row j consists of entirely 0, remember j, and delete both row j and column j from Σ_i.
- When computing $\mathcal{N}(\mu_i, \Sigma_i)(v)$, first we compare v^j with μ_i^j for the j that we remember from the last step. If they are different for any j, return 0. Otherwise, we compute the probability density with the modified Σ_i.

5. Training Strategy

5.1. k-nearest-neighbor model.

The parameters in this model then consists of k, which is the number of neighbors we consider, and the training set. As for the training set, once its size is specified, we use randomization to pick images from trainingImagesA.npy. We run the algorithm for $k = 3, 5, 7$, 11 and training size $500, 1000, 5000, 10000$ to classify 5000 images from trainingImagesB.npy, record the classification error

$$L_{01} = \sum_{i=1}^{5000} 1_{\{\text{predict}(u^i) \neq \text{label}(u^i)\}}. \tag{2}$$

Then we decide on the best performer, as presented in Section 6. The reason for the choice of k is that odd prime numbers help avoid ties, and small numbers to keep bias low. The reason for the choice of training size and test size is time constraint. Testing 5000 images with 10000 training images requires 800 seconds.

5.2. Gaussian discriminant analysis model.

The only choice we have to make is the training set. As for the k-nearest-neighbor model, we choose the size among $500, 1000, 5000, 10000, 20000$, and then use randomization to choose images from trainingImagesA.npy. For each, we use the fitted μ_i, Σ_i to classify 5000 test images from trainingImagesB.npy and compare the classification error as in (2).

6. Experimental Results

First we illustrate the effect of our feature set. We ran 3-nearest-neighbor on 5000 test images from trainingImagesB.npy, and report the loss in the following table.

training size ($k = 3$)	nonreduced images	reduced images
500	85.76%	17.36%
1000	81.10%	14.36%
5000	77.20%	9.86%
10000	77.14%	7.50%

The reason for the dramatic improvement is the curse of dimensionality, which is known to plague the k-nearest-neighbor model.

Now we report the result of our model selection for the k-nearest-neighbor model.

size/k	k=3	k=5	k=7	k=11
size = 500	17.36%	17.34%	18.44%	19.56%
size = 1000	14.36%	12.60%	13.76%	14.78%
size = 5000	9.86%	8.94%	10.00%	9.22%
size = 10000	7.50%	7.28%	8.08%	8.66%

Table 1.

A graph is provided for easy comparison.

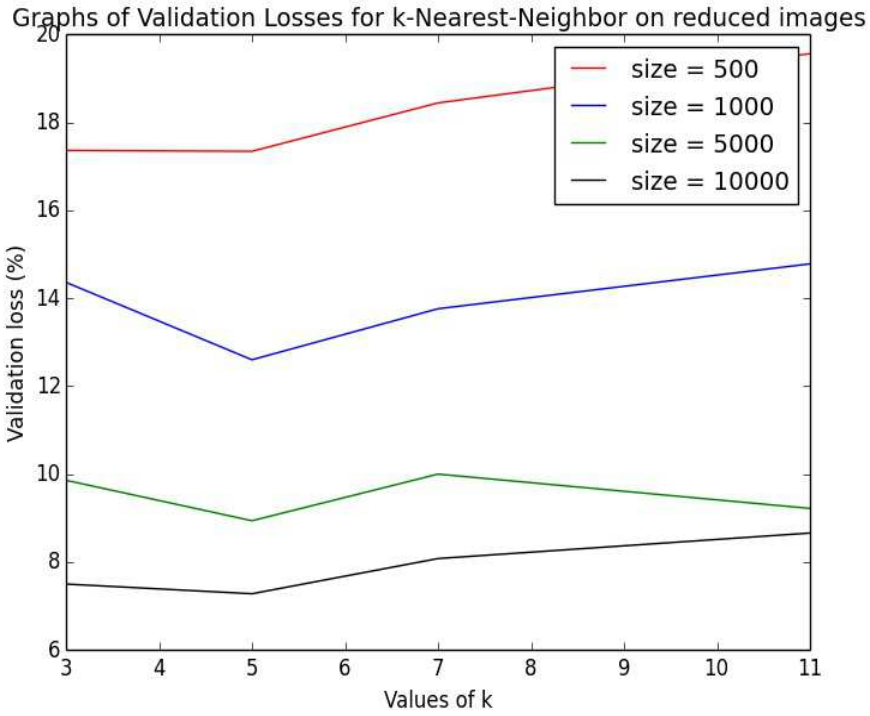


Figure 5.

Surprisingly, $k = 5$ is always better than $k = 3$. It may be due to the nature of the images that there are always noises, i.e. images of different digits that look very alike. In that case, having more images allow for the correct label to outvote the incorrect ones. Clearly, $k = 5$ is also a more stable choice than $k = 3$. We should note that this method is very slow. Since there is no training and the work is done at classification time, the time it took to classify 5000 test images when $k = 5$ and size=10000 is about 800 seconds.

We now report on the result of model selection for the Gaussian discriminant analysis model.

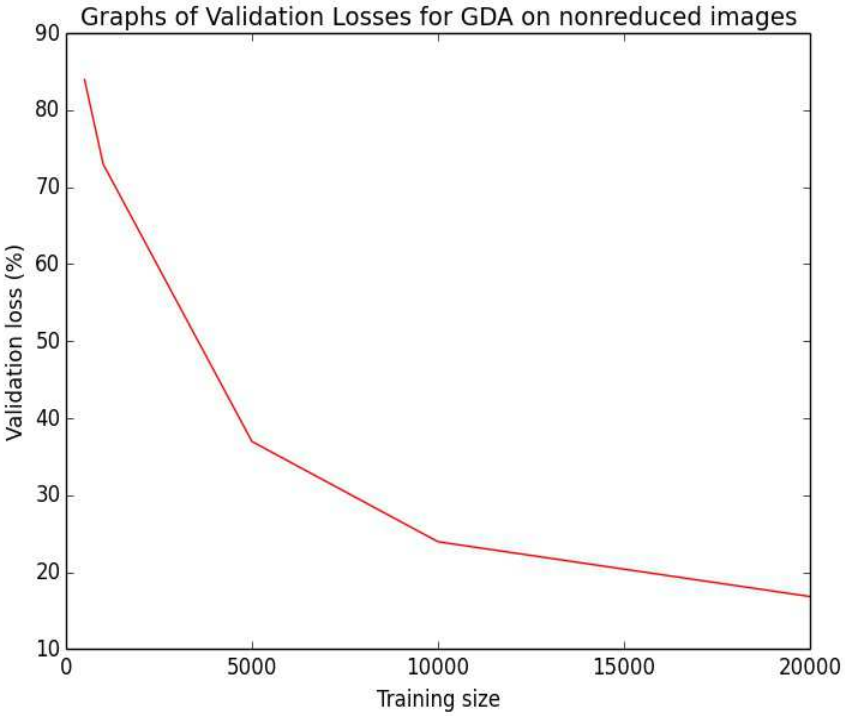


Figure 6.

It makes sense that the more training images we have, the better the prediction power, since μ_i and Σ_i can be estimated more accurately. In this case, the improvement is exponential, and one can hope that when combining both trainingImagesA.npy and trainingImagesB.npy for training (which increases the still very short training time, but not classification time), classification error will be as low as 10%, making it a viable alternative to the k-nearest-neighbor model. This method is very fast, taking 40 seconds to classify the same amount of test images given to the k-nearest-neighbor model.

7. Classification Error

Estimate of the classification error can be given by the statistics we summarized in Section 6. Essentially, for the 5-nearest-neighbor model with 10000 training images, we expect $L_{01} \sim 7.2\%$. For the Gaussian discriminant analysis model with 20000 training images, we expect $L_{01} \sim 16.82\%$.