CPS 571 HW1

Shengxin Qian

1. Perceptron Algorithm

(a). The algorithm made 2 mistakes if we start with $\overrightarrow{x_1}$. The convergence process is:

[0, 0] -> [-1, 1] -> [0.5, 2]

The algorithm made 1 mistakes if we start with $\overrightarrow{x_2}$. The convergence process is:

[0, 0] -> [0, 1]

The diagram showing the progression of the plane is:

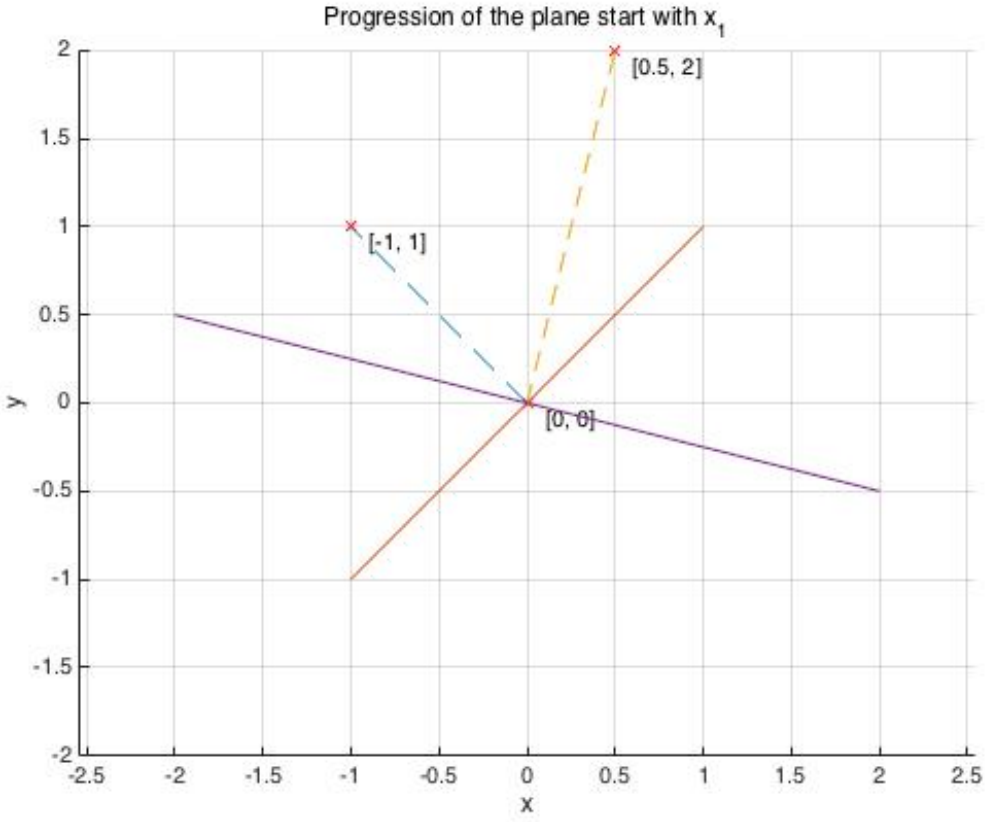

Figure 1: The progression of the plane starting with $\overrightarrow{x_1}$

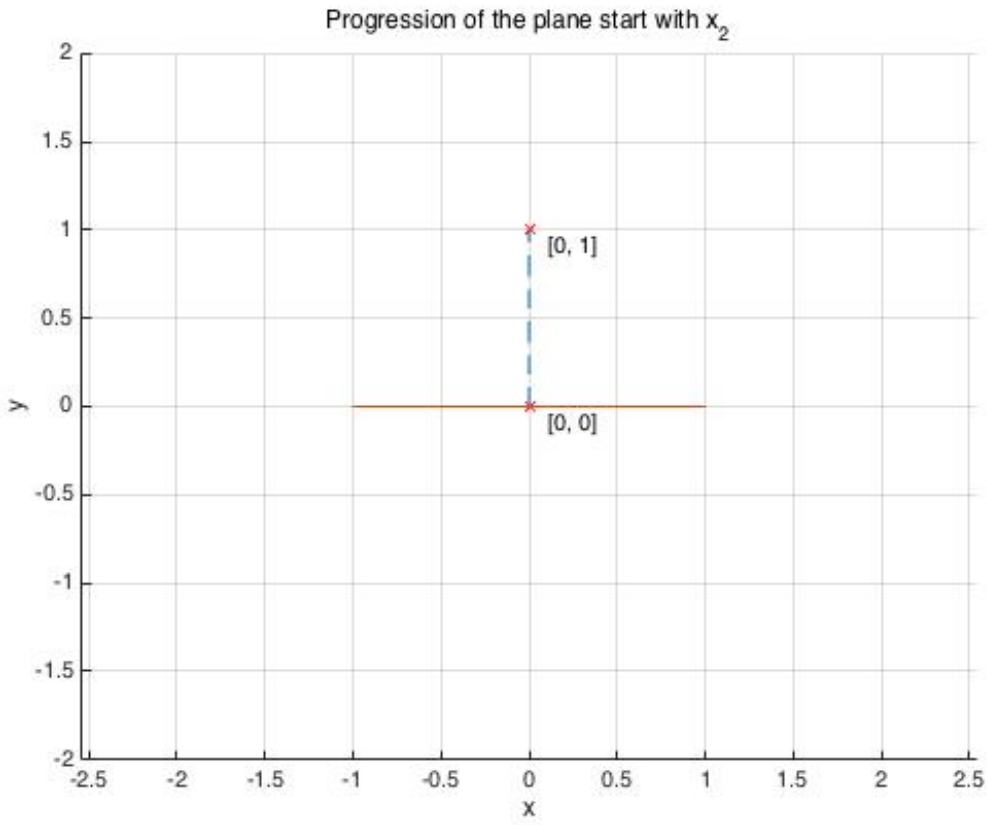

Figure 2: The progression of the plane starting with $\overrightarrow{x_2}$

(b). The algorithm made 6 mistakes if we start with $\overrightarrow{x_1}$. The convergence process is:

[0, 0] -> [-1, 1] -> [9, 2] -> [8, 3] -> [7, 4] -> [6, 5] -> [5, 6]

The algorithm made 1 mistakes if we start with $\overrightarrow{x_2}$. The convergence process is:

[0, 0] -> [0, 1]

The diagram showing the progression of the plane is:

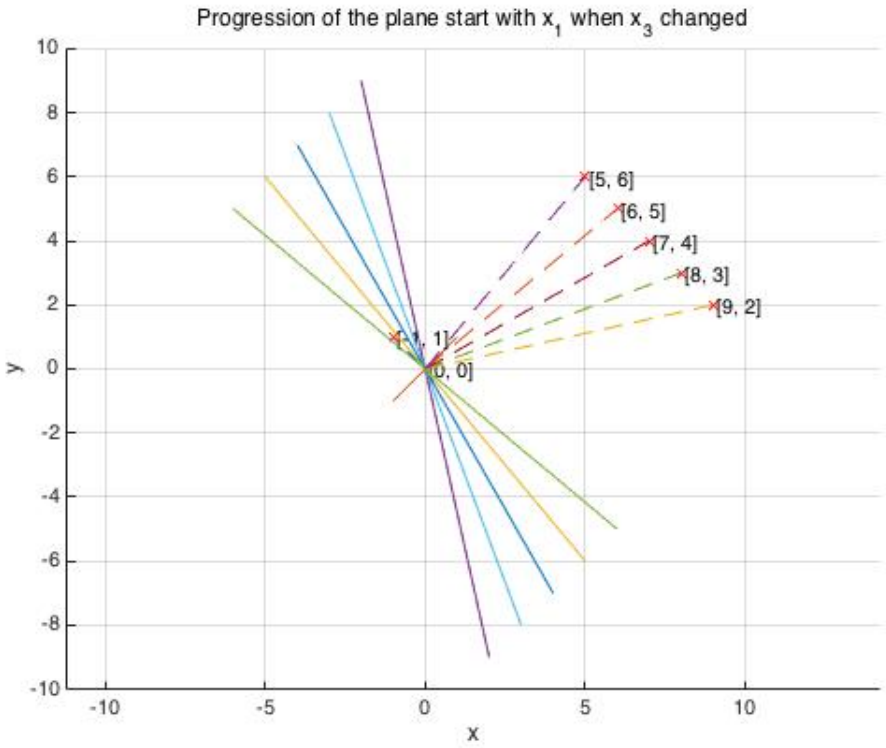

Figure 3: The progression of the plane starting with $\overrightarrow{x_1}$ when $\overrightarrow{x_3}$ changed

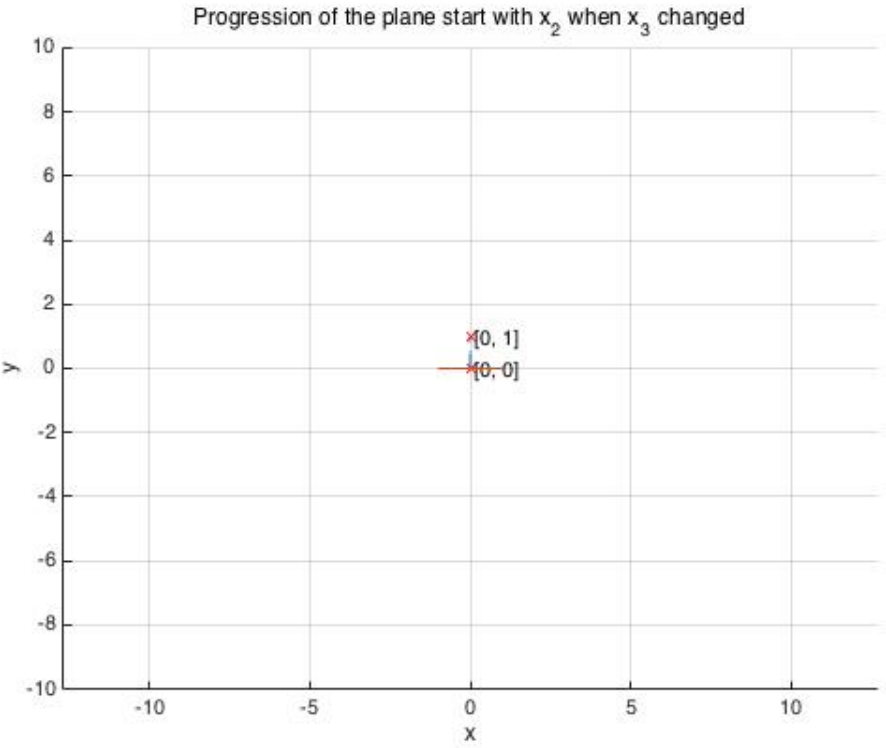

Figure 4: The progression of the plane starting with $\overrightarrow{x_2}$ when $\overrightarrow{x_3}$ changed

(c). The difference between the number of mistakes in (a) and (b) was caused by the difference of $\overrightarrow{x_3}$ norm in (a) and (b). Because the norm of $\overrightarrow{x_3}$ in (b) is larger than that of $\overrightarrow{x_3}$ in (a), when the algorithm meets a mistake at $\overrightarrow{x_3}$, the addition caused by $\overrightarrow{x_3}$ would make the plane deviate farther on the opposite side. Then, the plane needs to be restored by $\overrightarrow{x_1}$ and that takes more steps because $\overrightarrow{x_3}$ norm is much larger than $\overrightarrow{x_1}$ norm.

The difference between $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ is caused by the planes whose perpendicular vectors are $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$. The plane which is perpendicular to $\overrightarrow{x_1}$ has 1 misclassification. On the contrary, the plane which is perpendicular to $\overrightarrow{x_2}$ has 0 misclassification. The state of initial misclassification describes how far away from the right region. That is why starting from $\overrightarrow{x_1}$ will cause one more mistakes.

(d). As we discussed in question(c), if we want to maximize the number of mistakes, we need to choose the vector which has larger norm and more number of misclassifications when the partition plane is the plane perpendicular to the starting vector.

2. Handwritten Digits Classification

(a). When we simply run the perceptron algorithm without any feature reduction, the accuracy is 0.9991. The relation between the number of epochs and the number of mistakes in each epoch is shown below. As we can see in the Figure 5, because the plane needs to be adjusted in high dimension space, each epoch does not always converge. Sometimes, divergence happens.

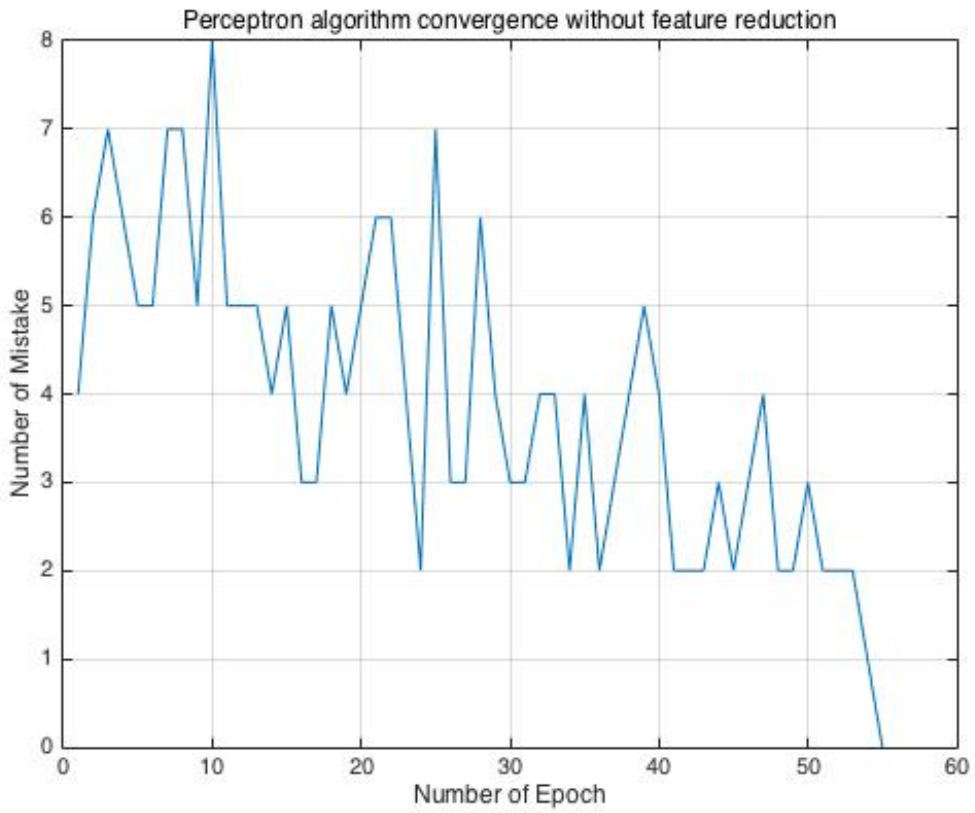

Figure 5. Perceptron algorithm convergence with no feature reduction

(b). One naïve feature reduction technique is to use the average normalization values of each feature in the training set to drop some trivial dimensions. We can reduce the number of features to approximate 220 and maintain similar accuracy. After normalization, the values in the training set are all between [0, 1]. If one feature is significant, the average normalization values would close to 1. On the other hand, that of trivial features would close to 0. So, the naïve feature reduction technique is to set a threshold and drop all the feature which has lower average normalization values then the threshold. This reduction technique works pretty well, Figure 6 shows the relation between the number of features we choose and the corresponding recognition accuracy.

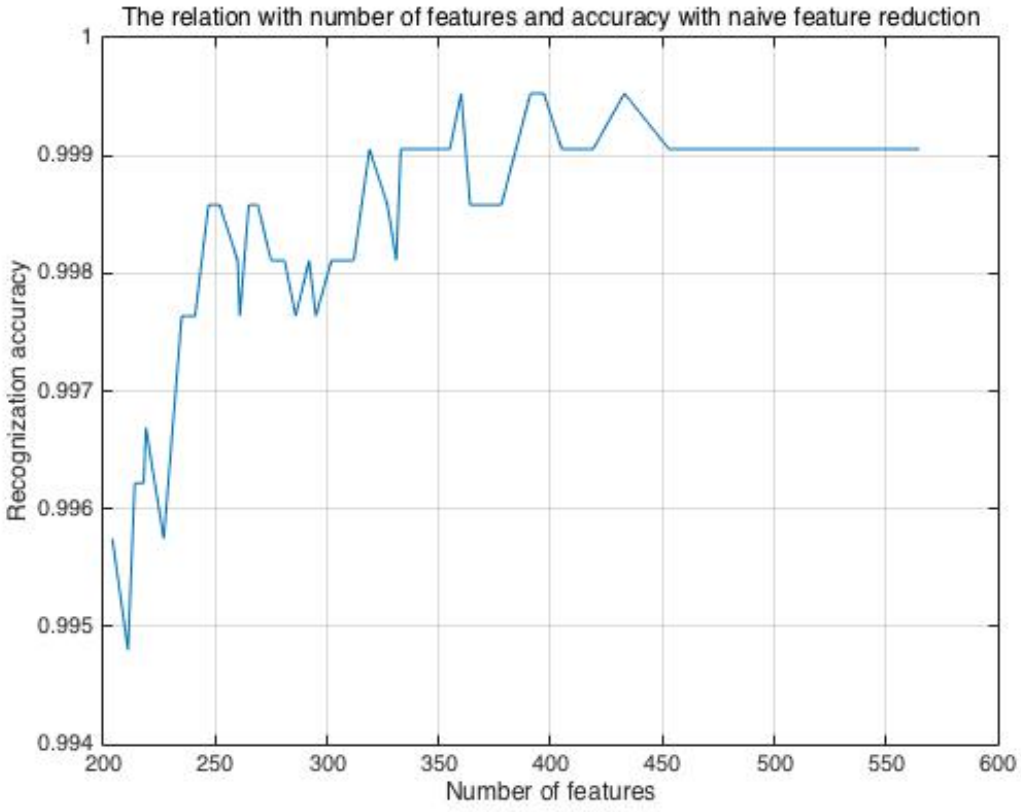

Figure 6. Number of chosen features and accuracy with naïve reduction

As we can see in the Figure 6, even we reduced the number of features to nearly 200 (we have 784 features in raw data), the recognition accuracy is still larger than 0.995. The correlation between recognition accuracy and number of features are not strictly positive, but the trend is the more features we have, the more accuracy the recognition is. However, the less features we chosen, the convergence consumes more time. So, it is better to choose more than 200 features.

(c) Another good feature reduction technique is using Principle component analysis (PCA). PCA is really useful in computer vision because it could compress the high dimension and sparse data into lower dimension data which is suitable for image data preprocessing. Figure 7 shows the relation between the number of chosen features and recognition accuracy. With PCA, we can

reduce the number of features to nearly 50 (raw data has 784 features) and obtain even better accuracy. However, when we reduce the number of feature to less than 40, it may not converge or takes too long to converge. So, we would better choose more than 40 features. The sweet spot is a bit more than 50 features. The algorithm of PCA is:

We assume the raw data is represented as matrix X, each sample is represented as a row vector.

$$\mathrm{COV} = X^T X$$
$$[\mathrm{U}, \mathrm{S}, \mathrm{V}] = \mathrm{SVD}(\mathrm{COV})$$
$$X_{new} = X U_K$$

U is a transform matrix which contains column vectors. U_K is a selected transform matrix which transform original space into K-dimension space. The choice of K is depending on the singular value in S matrix.

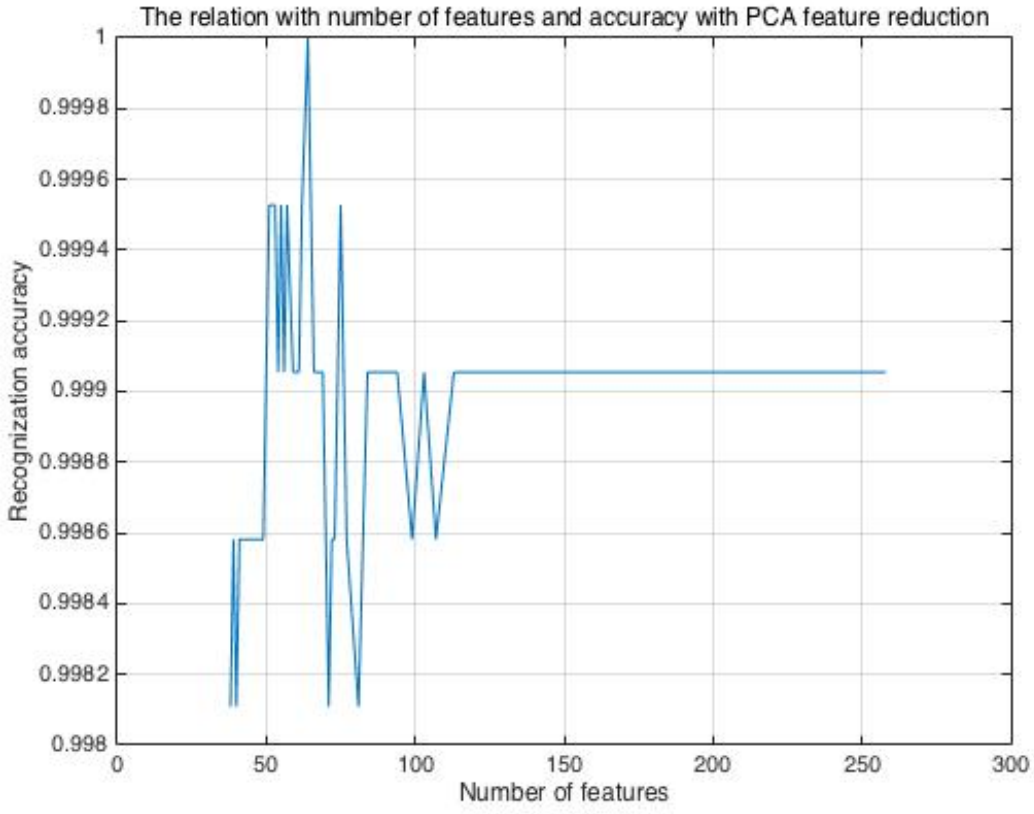

Figure 7. Number of chosen features and accuracy with PCA

As we can see in the graph, when we chose some specific numbers of features and transform them into other spaces, the accuracy of recognition could be even more accuracy than the raw data. This is because the PCA does not just drop trivial features but also increase the variance of each feature which makes two classes easier to be separated and decrease the covariance between each feature which reduces some noise.