# Comparison of Standard Machine Learning Methods

Erik Sandström

March 15th 2018

## Introduction

The objective of this project is to implement and compare four standard approaches in machine learning. It is assumed that the reader is well aquanited with the theory behind these methods. The code is publicly available for further details. All methods were trained and tested on a subset of the MNIST digit dataset containing 5000 training images and 500 tests images of 28x28 sized grayscale images. All algorithms were implemented without the use of any libraries in MatLab except the Support Vector Machine, where the LIBSVM library was used. This project is a summary of some of the material covered in ECE175A at University of California, San Diego. The dataset can be downloaded via http://www.svcl.ucsd.edu/courses/ece175/.

## Nearest Neighbor Classifier

### Class Conditional Error Rate

The conditional probability error rates for each digit were calculated by first calculating the number of digits within each class from the test set and then, for each class, calculating the number of correct classifications. The probability rate is then given by the quotient between the number of misclassified pictures within a class and the total number of pictures within that class.

Table 1: Conditional probability error rates for the different classes.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| P(Error\|Class = i) | 0 | 0 | 0.13 | 0.16 | 0.13 | 0.06 | 0.05 | 0.02 | 0.25 | 0.19 |

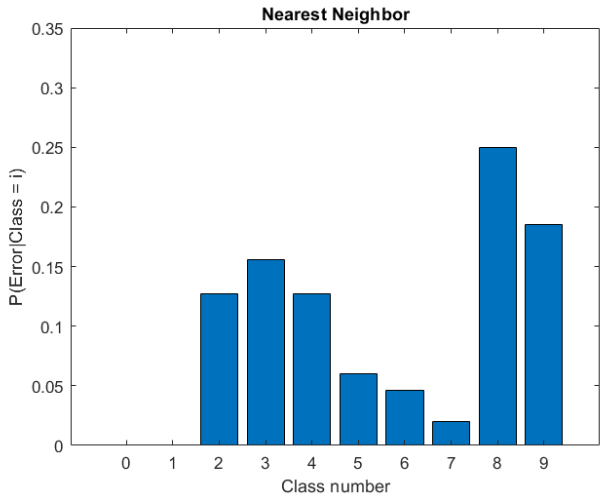


Figure 1: Plot of the conditional probability error for each class.

## Total Error Rate

The total error rate was calculated by first couting the number of nonzero elements in the vector formed by the difference between the classifier-produced label vector and the vector called `labelTest`. The total error rate is then given by the quotient between this number and the total number of classified pictures i.e. 500. The result is

$$P(Error) = 0.0940. \tag{1}$$

## Examples of Misclassified Digits

Five examples of misclassified digits are displayed in figure 2, 3, 4, 5 and 6. It is clear that the misclassified digits look very much similar to the closest neighboring digit from the training set which explains why the misclassification occured.


Figure 2: Example 1 of misclassified digit. The misclassified digit is classified according to the label of the training digit.


Figure 3: Example 2 of misclassified digit. The misclassified digit is classified according to the label of the training digit.

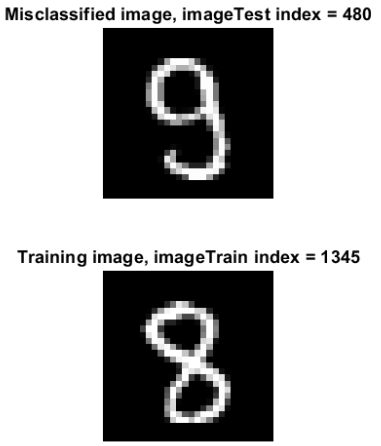


Figure 4: Example 3 of misclassified digit. The misclassified digit is classified according to the label of the training digit.


Figure 5: Example 4 of misclassified digit. The misclassified digit is classified according to the label of the training digit.

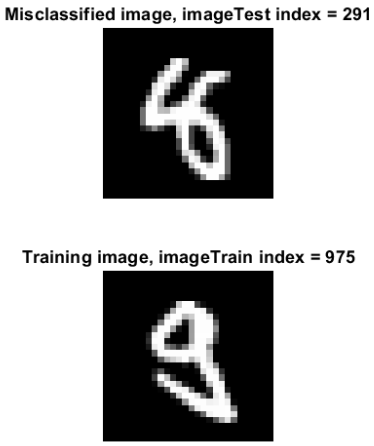


Figure 6: Example 5 of misclassified digit. The misclassified digit is classified according to the label of the training digit.

# Gaussian Classifier

## Calculating the Sample Mean over Each Class

The sample mean for each class is given by the figure 7.

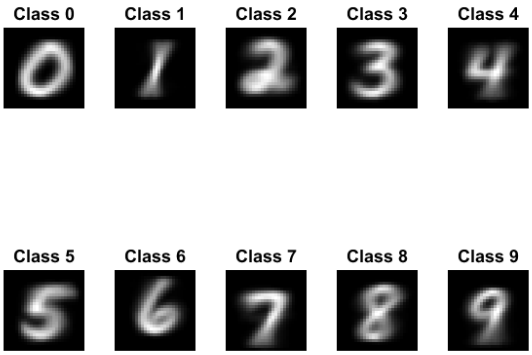


Figure 7: Sample mean of each class.

## Class Conditional Error Rate

The conditional probability error rates were calculated by first calculating the number of pictures within each class from `labelTest` and then, for each class, calculating the number of correct classifications. The probability rate is then given by the quotient between the number of misclassified pictures within a class and the total number of pictures within that class. The error rates for each class are given by table 2 and figure 8.

Table 2: Conditional probability error rates for the different classes.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| P(Error\|Class = i) | 0.14 | 0.03 | 0.35 | 0.2 | 0.22 | 0.34 | 0.23 | 0.24 | 0.25 | 0.26 |

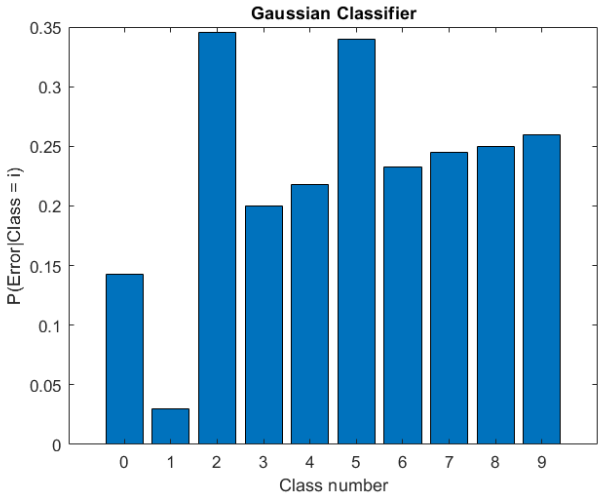


Figure 8: Plot displaying the conditional error rate for each class.

## Total Error Rate

The total error rate was calculated by first couting the number of nonzero elements in the vector formed by the difference between the classifier-produced label vector and the vector called `labelTest`. The total error rate is then given by the quotient between this number and the total number of classified pictures i.e. 500. The result is

$$P(Error) = 0.2220 \tag{2}$$

## Covariance Matrix

The covariance matrix for each class is calculated and the resulting matrices are symmetric as expected and are displayed below. The covariance matrix is

always positive semidefinite (and in some case positive definite). A positive semidefinite covariance matrix has eigenvalues that are non-negative. If only positive eigenvalues are present, then the covariance matrix is invertible and positive definite, but in our case the determinant is calculated to be zero which means that the covariance matrix is positive semidefinite. This means that the matrix is non invertible. None of the covariance matrices belonging to the classes are concluded to be invertible and thus we cannot evaluate the log term containing the determinant which becomes log(0), which is infinity. The reason why the covariance matrix is noninvertible is due to the fact that we have a too small data training set compared to the size of the covariance matrix. With more data, we could mitigate this error. In the next section, dimensionality reduction will be used in order to make use of the covariance data.

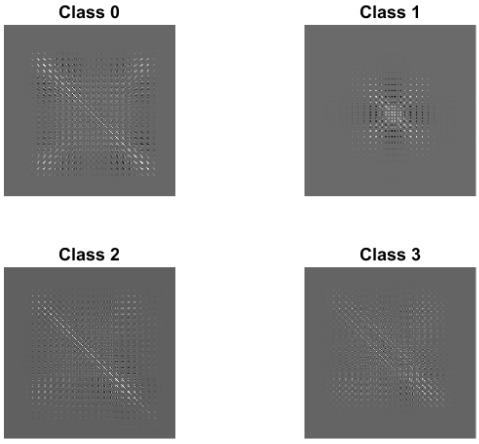


Figure 9: Covariance matrices for classes 1-4.

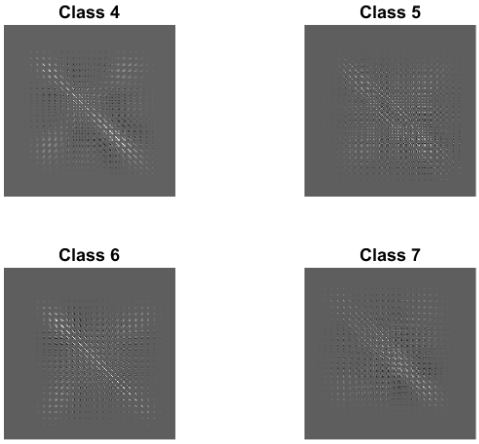


Figure 10: Covariance matrices for classes 5-8.

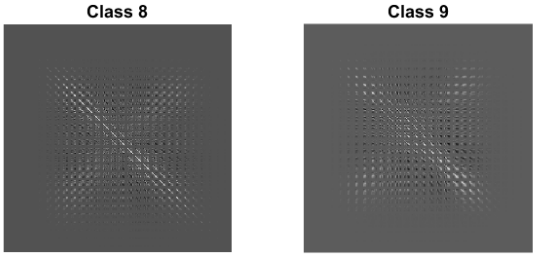


Figure 11: Covariance matrices for classes 9-10.

# Gaussian Classifier with PCA

## Principal Components and Eigenvalues

The top 10 principal components over all classes are plotted in figure 12

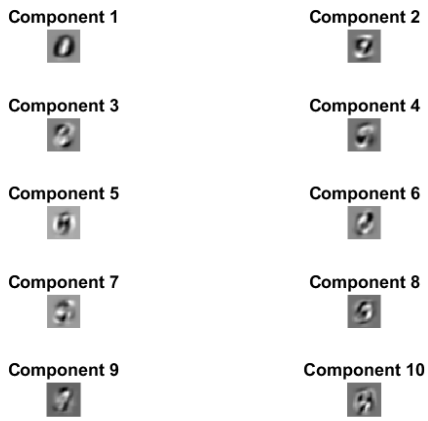


Figure 12: The first 10 principal components over all classes as 28x28 images.

The eigenvalues are plotted in figure 13 as a function of the principal component number.

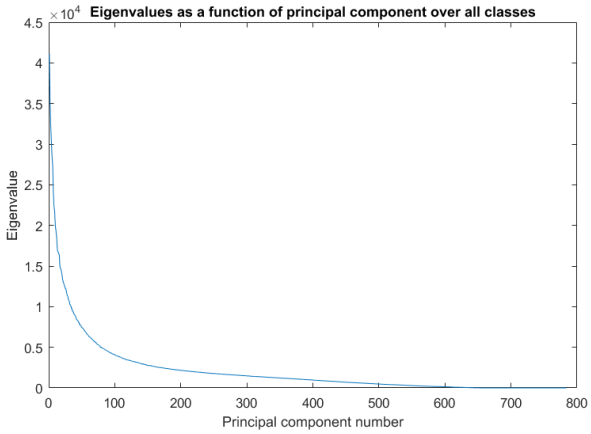


Figure 13: The eigenvalues plotted in decreasing order.

## Classification using PCA

According to figure 13, the "knee" of the plot can be seen to be around 90. A rule of thumb is that the "knee" indicates a good starting point in regards to the choice of the number of principal components to use.

The total error rate for different choices of subspace dimension is plotted in

figure 14.

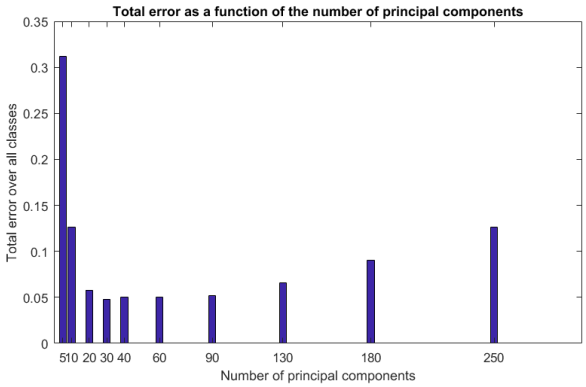


Figure 14: The total error rate as a function of the dimension of the subspace when performing PCA.

The result is also given by table 3.

Table 3: Total error rate as a function of the number of principal components used

| #PC | 5 | 10 | 20 | 30 | 40 | 60 | 90 | 130 | 180 | 250 | 350 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P(Error) | 0.312 | 0.126 | 0.058 | 0.048 | 0.050 | 0.050 | 0.052 | 0.066 | 0.090 | 0.126 | N/A |

The total error rate when not performing PCA, but doing Gaussian classification directly on the data is

$$P(Error) = 0.2220. \tag{3}$$

The results can be greatly improved by performing PCA and removing non discriminant features to reduce the complexity of the model and thus being able to gain a better result with fewer samples. The best result at and error of 4.8 % is gained for 30 principal components which is considered to be fairly in range of the approximative optimal number given by the position of the "knee" in the scree plot for the eigenvalues. The conditional error rate when 30 principal components are used is given by figure 15.

Finally, it can be determined that the covariance matrix for the case when using 350 principal components is positive semidefinite and thus non-invertible. This is a result of the fact that too few samples are used in relation to the complexity of the model.

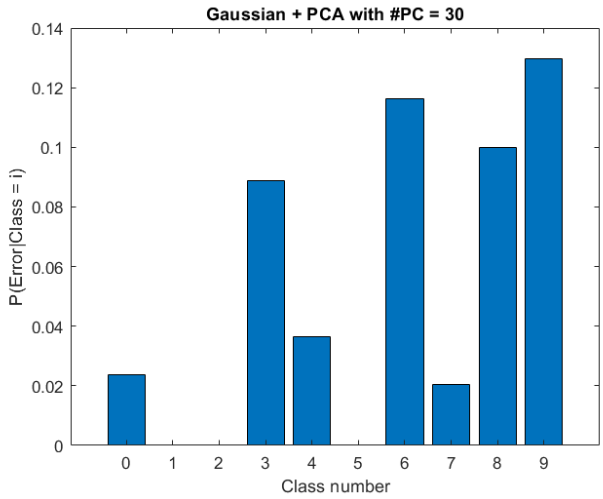


Figure 15: Class conditional error rate when 30 principal components are used.

## Digit Least Like Five

The top 10 principal components over class 5 are plotted in figure 16.

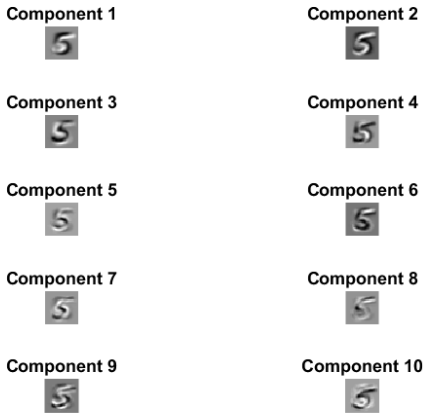


Figure 16: The first 10 principal components over class 5 as 28x28 images.

To find the image that looks least like digit 5, each test image is first transformed to the space determined by all the eigenvectors to digit 5 i.e. the space of all eigenvectors is 784-dimensional. Call the transformed test image vector X. The 40 first components of X is the projection of X onto the subspace of

dimension 40 belonging to the 40 first principal components of digit 5. Call this 40-dimensional vector Y. The euclidean distance of Y is calculated and the difference $||Z||^2$ between X and Y is calculated according to

$$||Z||^2 = ||X||^2 - ||Y||^2. \tag{4}$$

Equation 4 holds since the Pythagoream theorem holds in all spaces, regardless of dimension. The vector Z is the projection of X to all vectors orthogonal to Y which means that the norm of Z describes the energy orthogonal to the most important directions to digit 5. The resulting image that looks least like a 5 according to this optimum is of index 26 in the matrix of test digits and is given by figure 17.


Figure 17: The digit that looks least like a 5 is a 0. Index 26 denotes the position in the matrix of test digits.

# Support Vector Machine

## Binary Classificataion on Digits 6 and 8

When the SVM with a linear kernel and $C = 2^{-4}$ was trained on only digits from class 6 and 8 and tested on these digits from the test set, 100 % classification accuracy was gained.

Figure 18 depicts the training images that are closest to the decision boundary, i.e. they are examples of support vectors to the classifier.

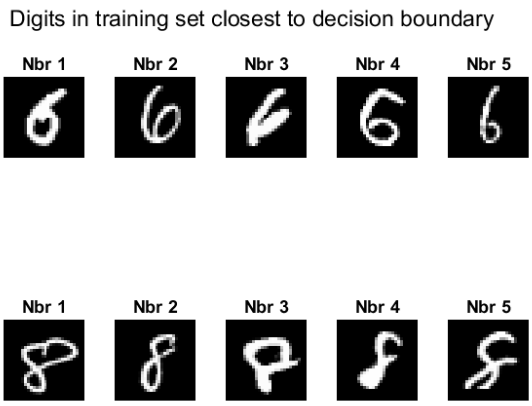


Figure 18: Five support vectors from each class are plotted as 28x28 images.

The normal to the separating hyperplane defined by the SVM is given by the formula

$$\omega^* = \sum_i \alpha_i y_i x_i, \tag{5}$$

where $\alpha_i$ are the Lagrange multipliers, $y_i$ the label of an input $x_i$. This results in the normal vector displayed in figure 19 as a 28x28 image.

Figure 19: The normal vector to the hyperplane of the SVM displayed as a 28x28 image.

For each test example, the absolute value of its distance to the classification boundary was computed. A histogram with 10 bins was used to illustrate the distribution given by figure 20.


Figure 20: Histogram showing how the test digits are spread around the decision boundary.

The five test examples from each class that are farthest from the boundary are given by figure 21.

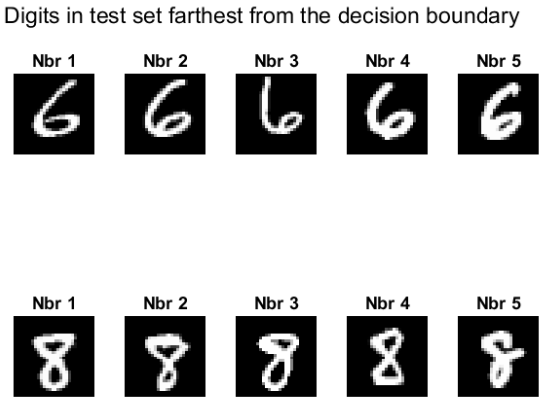


Figure 21: The five digits from each class that are farthest from the decision boundary.

The five test examples from each class that are closest to the boundary are given by figure 22.

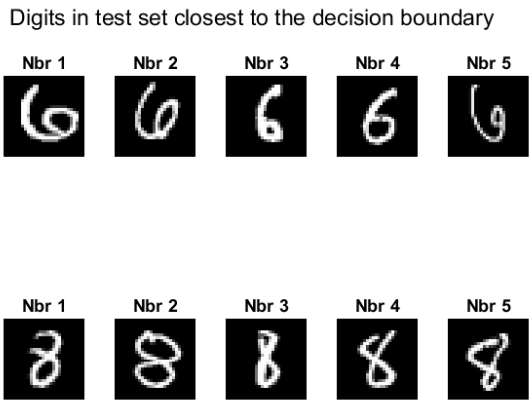


Figure 22: The five digits from each class that are closest to the decision boundary.

One can conclude that the digits that are far from the decision boundary are more ideal digits than the digits close to the decision boundary. This makes sense since we expect digits that are not ideal to have traits from several digits.

## Multiclass Classification on All Digits

2-fold cross validation on the training set and grid search was performed to find the optimal values of C and $\gamma$. This resulted in a maximum accuracy of 95.06 % when $\gamma = 2^{-5}$ and $C = 2^5$, $2^7$, $2^9$, or $2^{11}$.

When using the optimal values of $\gamma$ and $C$ and testing on the entire digit test set yielded 95.8 % accuracy. It is quite remarkable that higher accuracy was achieved when testing compared to training. The class conditional error rate is given by figure 23.

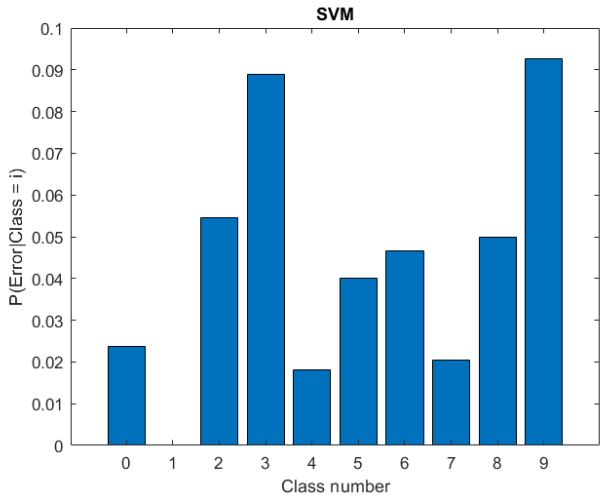


Figure 23: Class conditional error rates for the SVM.

# Comparison

The total classification accuracy for the nearest neighbor classifier, the Gaussian classifier, the Gaussian classifier with PCA and the SVM classifier is given by table 4.

Table 4: Error of classification for different techniques.

| Classifier | Total error |
|---|---|
| NN | 9.4 % |
| Gaussian | 22.2 % |
| Gaussian + PCA | 4.8 % |
| SVM | 4.2 % |

Best performance is gained by performing SVM classification with an error rate of 4.2 %.

The class conditional error rates for each classifier is given by figure 24.

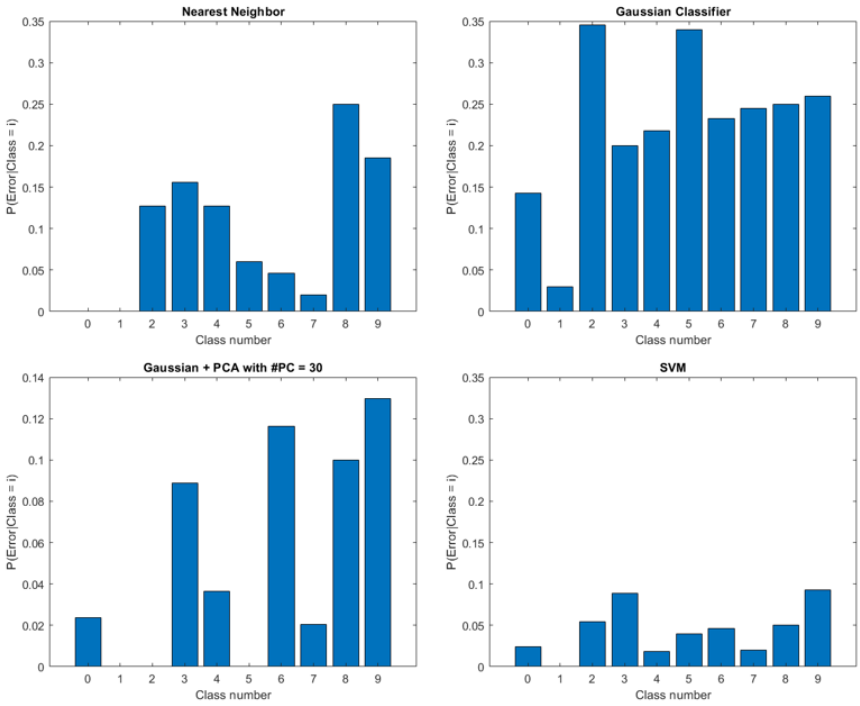


Figure 24: Class conditional error rates for the nearest neighbor, Gaussian, Gaussian with PCA and SVM classifier.

The NN classifier is surprisingly sophisticated despite being very simple. It is able to create non linear decision boundaries which is an advantage. It does not assume any distribution of the data which is considered positive. It is, however not very efficient due to the fact that each test point has to be compared with every training point in order to make a decision. It is also a hard classifier since small changes among the training examples can affect classification drastically. This can be a disadvantage to the robustness of the classifier. Another disadvantage is that we need to make sure we are using a rational distance metric. The optimal distance metric can vary between problems.

The Gaussian classifier assumes that the data is normally distributed. If this does not hold, the classifier will perform terribly. In order to utilize the entire formula of the decision function for the Gaussian classifier, enormous amounts of data needs to be provided during training in order to create positive definite covariance matrices. The curse of dimensionality hits this classifier hard. This is a disadvantage. An advantage is that training can be done first and testing will be performed very efficiently since each testing image is compared to only the ideal" digit of each class.

To increase efficiency of the Gaussian classifier, PCA can be performed. This yields much higher accuracy since classification is performed in a subspace with much lower dimension where not as much data is needed to fill the space. This method is very effective, but PCA does not always work and one must take this into account. Depending on the problem, the principal components might consist of vectors where the classes are non discriminant. If the first principal component is a vector along which the classes are non discriminant and only this vector is used during classification, the classifier will have an accuracy of 50 % which is the worst classifier you could ever create.

The SVM performs the best. It is a very robust method for classifying data since only the support vectors determine the outcome of the decision. Therefore, data points that are not support vectors do not matter for the decision. This, in turn, makes the method effective when the training set is not very large. If the training set becomes large, there are better classifiers like neural networks, but for small training sets, the SVM will outperform most other classifiers. The classifier is also versatile since it can use kernel functions to separate data that is not linearly separable. A drawback of this classifier is the complexity of the optimization problem.