Pattern Recognition in image analysis: an example of written digits

**Background:**

## Pattern recognition is one of the most practical and promising research fields in terms of machine learning. Related algorithms are widely applied in robotic vision, facial recognition and automatic monitoring etc. Image recognition, a sub-discipline of pattern recognition, is a branch focusing on how to detect and extract information from image data. The goal of this project is to build models which take an image of a handwritten single digit, and can correctly determine what that digit is. Obviously, it is an application of pattern recognition.

The data used in this project were taken from the MNIST dataset. MNIST dataset is a large database of handwritten digits that is commonly used for training various image processing systems. Each image is with 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. In the original dataset each pixel of the image is represented by a value between 0 and 255, where 0 is black, 255 is white and anything in between is a different shade of grey.

Kaggle is the world's largest community of data scientists. It provides a online platform for data scientists to compete with each other and solve complex data science problems. Its training set and testing set is provided by Kaggle (<https://www.kaggle.com/>) with the format of Comma Separated Value(CSV). The training set contains more than 70000 digit pictures and testing set contains 28000 images needed to classify. After participants upload their classifications to the website, Kaggle will provide the accuracy of classification, or the proportion of test images that are correctly classified.

In this report, totally four supervised learners will be trained and tested by three group members. The four models are k-nearest neighbour’s (modeler: Ziyuan Wang), decision tree (modeler: Yiwei Zhang) and random forest and support vector machine (modeler: Hongliang Chi). The related parts of presentation and the related parts of this report are mainly completed by corresponding modelers.

**K-nearest neighbors algorithm**

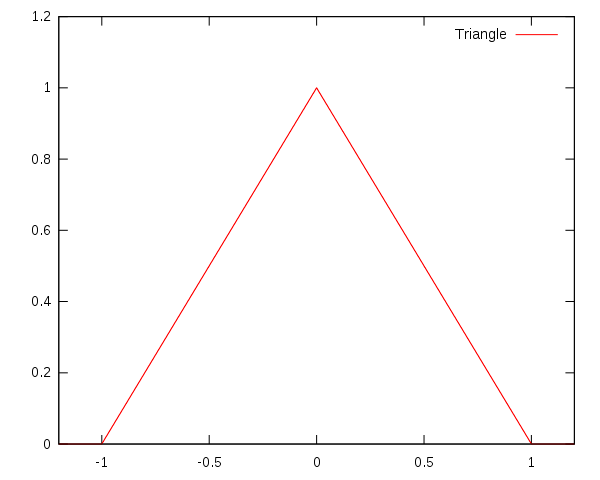
Method:

Algorithms:

In general, KNN is a famous non-parametric method used for classification and regression. The input consists of the *k* closest training examples in the [feature space](http://en.wikipedia.org/wiki/Feature_space). In k-NN classification, the result of classification is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class which has majority among its k nearest neighbors (k can be chosen from 1 to infinity but often very small). For example when k = 1, the object is simply assigned to the class of its single nearest neighbor. According to the elements of statistical learning written by Trevor Hastie, Robert Tibshirani and Jerome Friedman (2013, p.17), K-nearest neighbors algorithm dominate the market for low dimensional problem. Curse of dimensionality exists as a barrier for KNN to solve the high dimensional problem. Kernel method (also known as weighted KNN) is a revised version of KNN with a smoothing weight which is determined by a chosen type of kernel function (in my case is triangular).The kernel function is graphed and presented as below.

Triangular kernel function

x



*Figure1 of KNN*

From the above picture we can see that the weights die off with distance.

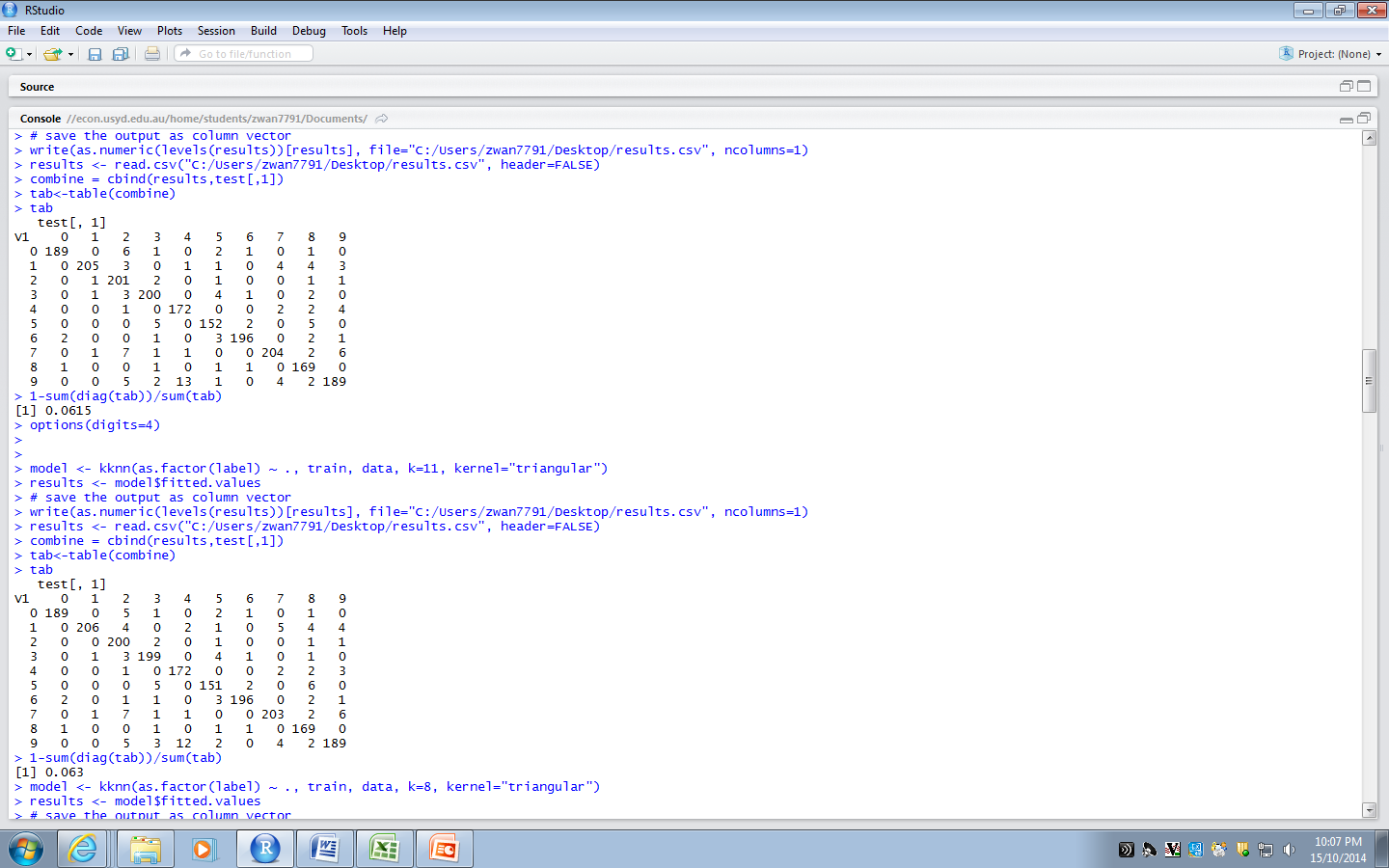
Friedman (1997) explains what is curse of dimensionality. The bias variance trade-off regarding to classification error in KNN generally is driven by the bias in high dimensional space (too input many columns). Because of the geometry of Euclidean spaces, the radius of a neighborhood grows as the nth root of its volume. Whilst the number of training points in the region K grows roughly linearly with the volume. Thus, even the smallest possible number of neighbors (K = 1) gives rise to large area in terms of radius and consequently cause high bias even for K = 1.

Optimal Model Selection:

The way to select the optimal number of neighbors is to use training set to train model and testing set to test whether the model is over fitting. Since the training set provided is too large (73.22 megabyte) to run the model selection code. The original training set is split into a sub-training set (the first 4000 observations) and the testing set (the 4001st to 6001st observations). Also columns with low variance have been deleted to reduce the dimensionality of this digit recognizing problem and also save the running time of the algorithm. Error rate are computed and then compared between different numbers of neighbors.

Findings (with error correction of the misclassification curves)

Confusion Matrix of K=1 Confusion Matrix of K=6 Confusion Matrix of k=11

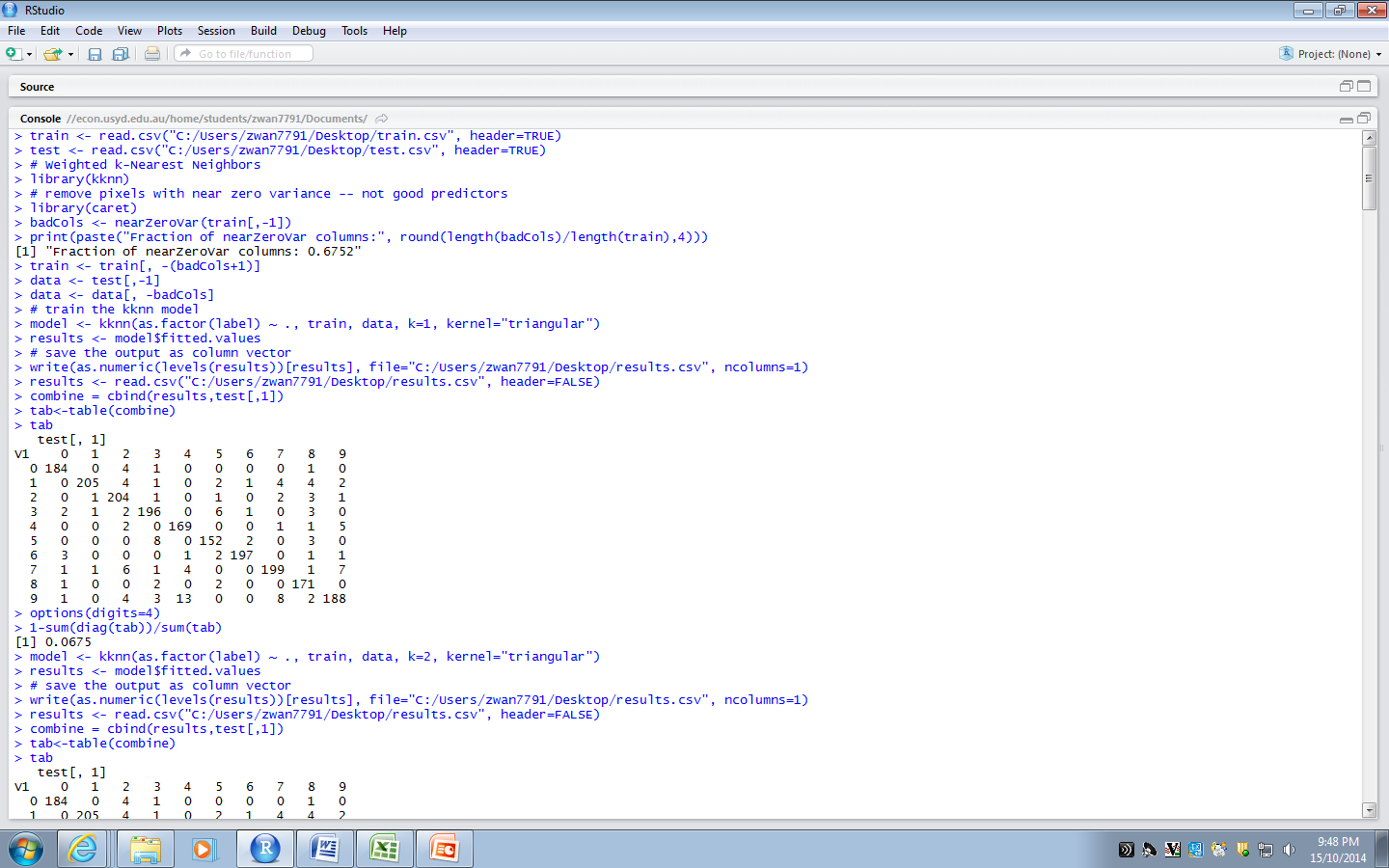
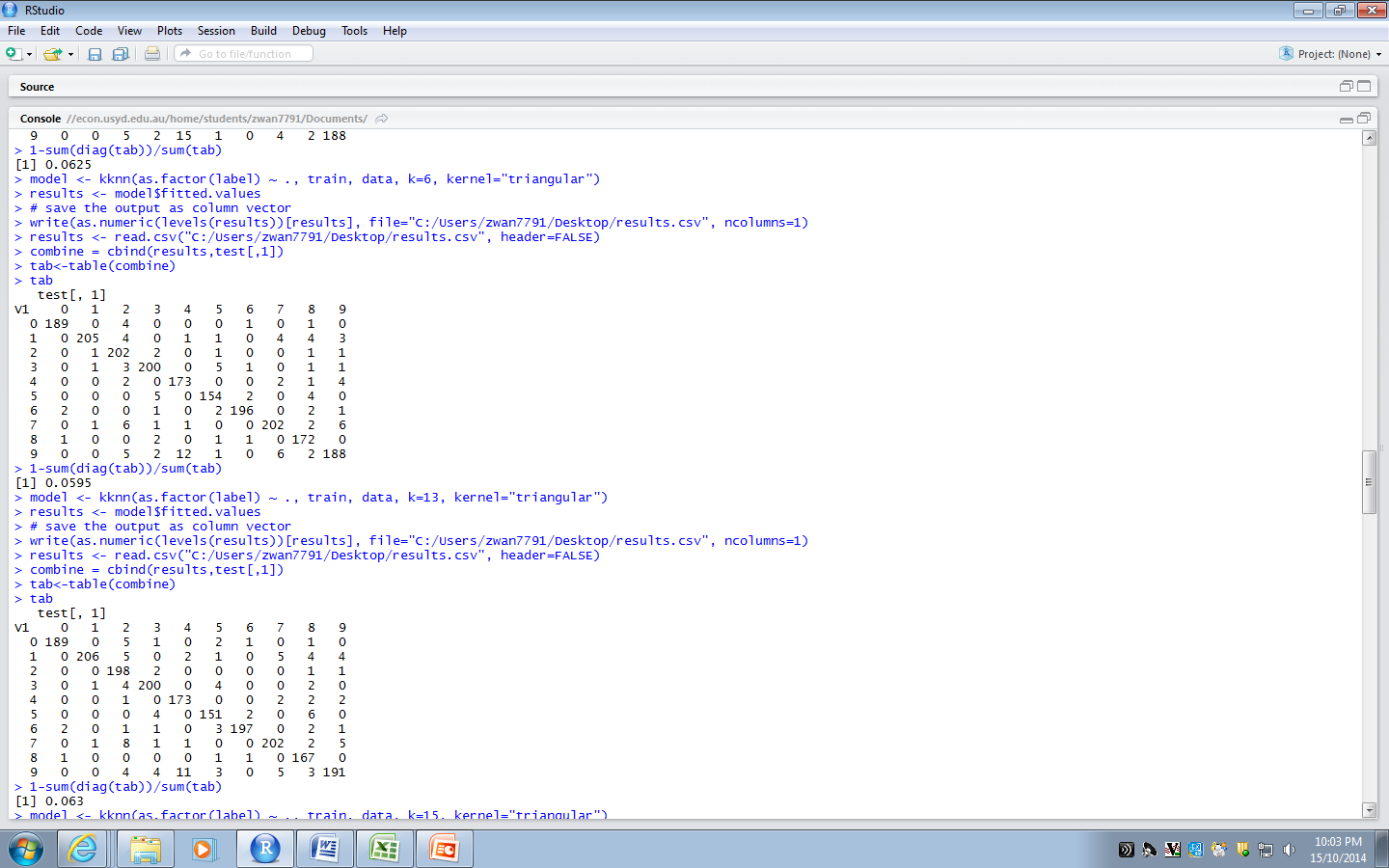


**93.70%**

**94.05%**

**94.5%**

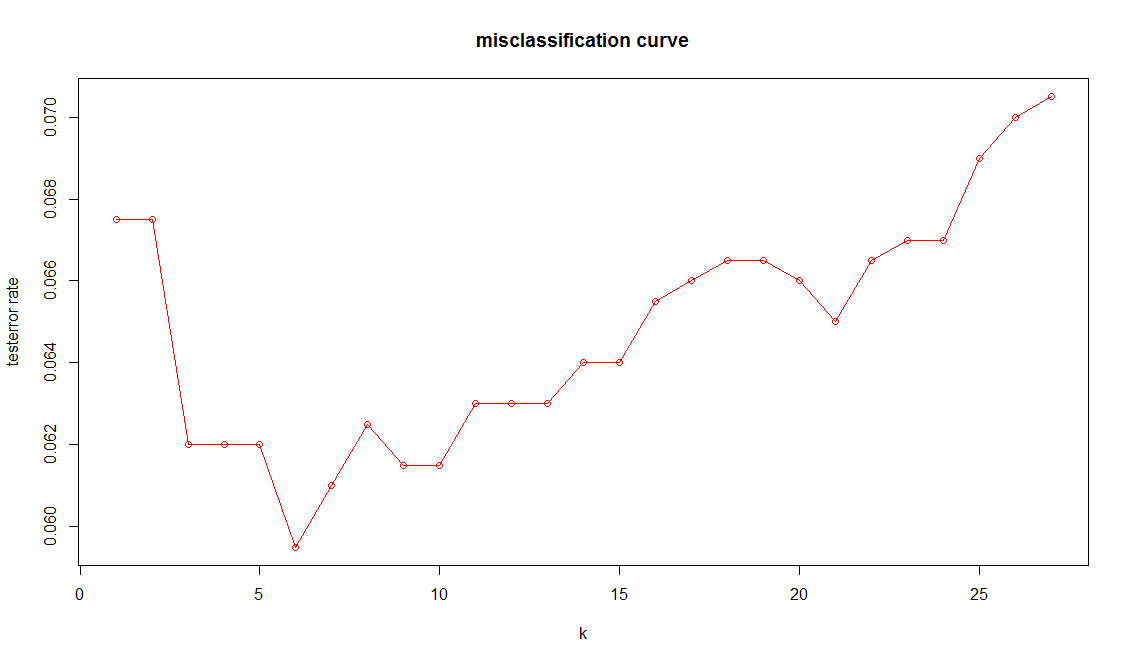
**93.25%**



*Figure2 of KNN Figure3 of KNN Figure4 of KNN*

In order to present before the audience, confusion matrix was generated for K=1, k=6 and K=11 respectively. Number in the diagonal of confusion matrix means correct classifications. In confusion matrix we can observe that KNN often confuse 7 with 9.

Misclassification curve used in presentation Misclassification curve for k=1 to k=27



**k**

*Figure5 of KNN Figure 6 of KNN*

There is a typo (circled in red in Figure5 of KNN) in the misclassification cure spotted during presentation and then correction was made using R in Figure 6 of KNN. Also the label of X axis of the misclassification curve has been changed from 1/k to k to avoid confusion.

The Y axis on misclassification plot is Error rate. The X axis is k.

Test error is minimized at some point K≠1. In my case is K=6.

Interpretation

Pixels with near zero variance means it pixel value almost stays the same across different image. It is equivalent to adding a column of zero to the data. Adding of deleting this kind of column would not change the information in the data. Thus we can reduce the dimension of the problem without losing important information.

As Yuchun lee (1991) mentioned in her paper, with enough training examples, even a simple classifier such as a weighted kNN classifier was able to solve a seemly complex high-dimensional problem. This is proved by my result. Weighted KNN can achieve high classification accuracy 94.05% in test set when k=6.

MSE can be broken into two component bias^2 and variance. There is variance and bias trade off in selecting number of neighbours. Optimal balance between variance and bias should be found by adjusting number of neighbours to include.

Limitation

Also from the paper written by Yuchun li (1991), computer can be slow to find nearest neighbor in high dimensional space and need large amount of memory.

Good news is that various heuristic algorithms have been proposed to prune/ edit/ condense “irrelevant” points that are far from the decision boundaries (such as ‘cover tree’ algorithm)

Another problem for KNN is that it does not give probabilistic output.

A classification function like KNN returns a single best guess given an input.

A probabilistic classifier however returns a probability distribution over outputs given an input. If the probability is near 0.5, the system may choose to ask for human help. This can further enhance our classification accuracy.

**Decision tree**

Method

Algorithms

Classification tree is an unsupervised machine-learning method for constructing prediction model from data. The model are obtained by recursively partitioning feature space and fitting a simple prediction model within each partition. The parameters involving in decision tree are splitting variables and split points.

However, because these two variables will be decided automatically by algorithm, tree is a nonparametric classification rule. A splitting variable is the measurement or feature which can be used to distinguish or classify unknown outcomes. A Split point is a boundary value on that splitting variable. For example, an observation whose value of a splitting variable is larger than the boundary will be classified into one group and otherwise, the observation will be classified into another group. Accordingly, the tree outcome is a multiple-level binary tree. A binary tree consists of a set of sequential binary decisions applied to each case that leads to further binary decisions or to a final classification of that case (Vaugh and Wang 2008). A node on the tree represents each partition.

Optimal Model Selection

For each split, tree tries to find the splitting variable and split point which produce the largest improvement in goodness of fit (Vaugh and Wang 2008). To determine the improvement in goodness of fit, this report uses a measure called the Gini index, which is suggested by Vaugh and Wang (2008). In the R process, since the sample size is very large, this report chooses a small cp value at 0.00005. The cp in R is complexity parameter of which each split should at least increase the goodness of fit a factor. The cp can help save time and meaningless efforts. In order to find the optimal size tree, this report use R to grow a full size tree and then prune the tree to optimal size. The criterion for pruning is to minimise the cost complexity criterion.

Data Source

As mentioned in background, the data is from Kaggle. However, the test date provided by Kaggle does’t include the actual outcome (e.g. Label) for each observation. This report randomly chooses 80% of original training data to construct the training set, nearly 33600 observations. The rest is test set.

Findings

As the R output shows, the full size tree has 575 nodes. Its missclassification rate in the training set is 8.40% and that in the test set is 14.24%. The full size tree is overfitting. In the following picture, the y axis measures the complexity cost ( it is represented by ‘xerror’ in R) and the x axis is cp which is the proxy of tuning parameter. Although it is not obvious, there does exist a ‘U’ shape and from Table 1 it can be found that the optimal size is 503.

Figure 1

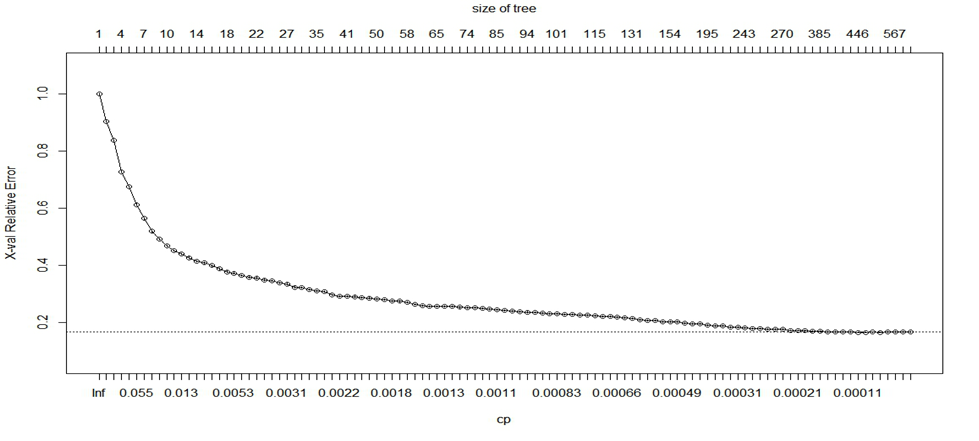
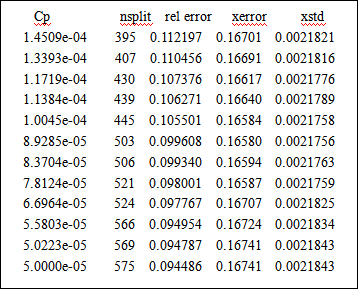
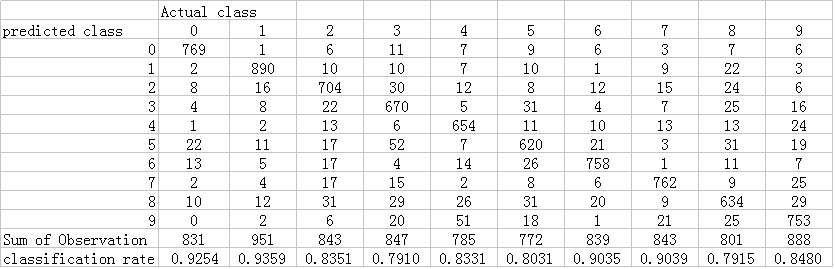


Table 1



Afterwards, this report prunes the tree based on minimising ‘xerror’. After pruning, the missclassification rate in training set increases to 8.85% while the missclassification in test set decreases to 14.12%. This shows pruned tree is an optimal result. Table 2 presents the confusion matrix of the pruned tree. One can find that for digit ‘3’ the classification rate is the lowest and it may indicate that ‘3’ is the hardest one for machine-learning.

Table 2



Interpretation

The R output shows that splitting variables in the pruned tree are about 245 pixels. It means there are 245 features taken into consideration by this tree model. For a new observation, the tree will assign the new observation into one of two binary results (next level nodes) based on the its value of the sequential pixels. Then the tree repeats this procedure until the new observation are assigned into a terminal nodes which is a predicted class. In our case, the predicted class is a label from 0 to 9. One can find that the tree method is very straight forward and intuitive. This is the key advantage of classification tree. Tree has many advantages (Vaugh and Wang 2008). It is a nonparametric technique not requiring distributional assumption. It is robust with respect to outliers and it is appropriate for data sets that are complex(Vaugh and Wang 2008). Another possible advantage of tree is modeling interactive predictors ( Loh 2002).

Limitation

There are three major problems with tree: instability, lack of smoothness, difficulty in capturing additive structure (Hastie, Tibshirani and Friedman 2009). Instability of trees is caused by the hierarchical nature of the process as an error in the top split affects all of the splits below it. In some cases, a small change in the data can result in a very different series of splits. The lack of smoothness stems from the fact that the prediction surface is discrete. Nevertheless, in our case, it may be not crucial since the outcome is discrete categorical value. Tree is difficult in modeling additive structure resulting from its binary structure. Additive structure include an additive zero-mean noise. It is possible for tree do split each terminal nodes again to distinguish the noise. However, to get the correct structure is fortuitous and cannot be investigated. In summary, tree has both advantages and drawbacks mainly because of its binary structure.

**Random forest [modeler: Hongliang Chi SID:430028388] :**

Method:

Algorithms:

Random forest is an ensemble supervised learner and operates by randomly constructing a multitude of decision trees without pruning. It uses majority principle to do classifications. Random forest was firstly introduced and developed by Breiman (2001). He came up with a method about how to randomly generate decision trees, which is the theoretical foundation of random forest. The randomness of decision trees is from randomly selecting a subset of variables to be used to split at each node. Common variables for random forests are number of trees and number of variables used to split at each node.

Optimal Model Selection:

The way to gain the optimal parameters of random forest is to use training set to train model and testing set to test its generalized capacity. Since the training set provided by Kaggle is too large to be trained. The original training set is split into a sub-training set (the first 1000 observations) and a testing set (the 1001st to 1600th observations). In terms of optimal number of trees, a comparison on classifying error has been run from the random forest with 50 trees to random forest with 1000 trees. Every new model between the two models will have more 50 trees than the previous one.

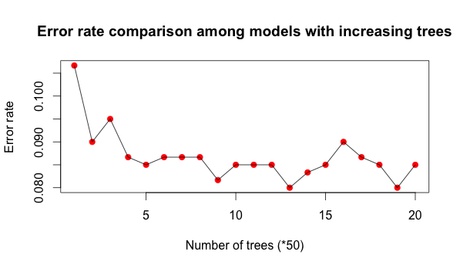


Figure 1 of Random Forest

In terms of the number of variables used to split each node, a similar comparison has also been run between the random forest with 50 variables used to split each node to random forest with 700 variables used to split each node. Likewise, every new model between the two models will have more 50 variables to split the node than the previous one.

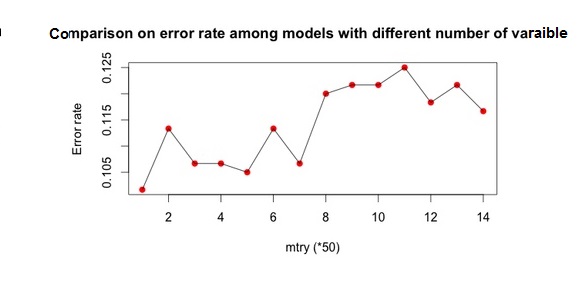


Figure 2 of random forest

According to the two comparisons between random-forest models with different parameters, it is concluded that 950 (see Figure 1 of random forest) is the optimal number of trees and 50 (see Figure 2 of random forest) is the optimal number of variables used to split each node. Therefore, our final random forest consists of 950 decisions tree, and each tree uses 50 randomly chosen variables to split each node.

Kaggle Testing Outcome:

Classification Accuracy (provided by Kaggle and see Figure 3 of random forest): 0.96829



Figure 3 of random forest

Findings:

From the previous plot about the number of decision trees (see Figure 1 of random forest), the error rate is not like “U” shape. Conversely, it seems that with the increase of number of trees, the error rate tends to converge to a constant. Surprisingly, there is no hint of over-fitting.

In fact, Breiman (2001) explains why random forest is unlikely to overfit in his papers. He wrote that in random forests,. For a large number of trees, it follows from the Strong Law of Large Numbers and the tree structure that:

This result explains why random forests do not overfit as more trees are added, but produce a limiting value of the generalization error.

Interpretation:

Therefore, if we want more accurate predictions, we need to plant more trees in random forest. However, accuracy is never a weakness of random forest. The real barrier which prevents random forest from being widely used is its scalability. In other words, the efficiency of random forest algorithm has no competitive advantage against other classifiers, especially its time-efficiency. Gerard (2012) have estimated random forest algorithm’s time complexity, which is O(M(mn log (n)) ) for n instances , m attributes , M trees. However, KNN’s straight forward implementation of finding k Nearest Neighbour takes O(dn) time, if we assume that the points are d-dimensional. It means that running time of random forest algorithm may be 10 times and even 50 times greater than KNN’s.

Since random forest cannot efficiently handle a growing amount of work in a capable manner, still, it’s industrial application is not as wide as other popular algorithms. It is believed that random forest would be more widely-accepted after the application of next-generation computing device like quantum computer.

Limitations:

Due to constraints of time and limited computational capacity, there is still some room for improvement in this research.

Firstly, the training set I chose is of 1000 observations and testing set is only 600 data points. Those relatively small training and testing set may affect the research accuracy on algorithm’s asymptotic property. The apparent evidence is with the increase of number of trees, the convergence of error rate is not stable as it should be.

Also, the lack of numerical optimisation is a limitation of this research. The image data we use, mathematically speaking, is a square matrix with a multitude of zero. This kind of matrix is referred as sparse matrix, which has good property in terms of optimisation. The efficiency of random forest algorithm would be improved if those optimalisation techniques are used.

**Support vector machine**

Method:

Algorithms:

Cortes, Corinna, and Vapnik (1995) introduced the method of constructing support vector machine:

Consider hyper-planes, where

. The set of decision functions

A has a VC dimension satisfying:

Where R is the radius of the smallest sphere around the origin containing.

By minimizing ||w||, we can get the maximum margin classifier. SVM separates data points with that maximum margin hyper-plane.

Also, SVM is known as efficient non-linear classifiers. It is able to separate non-linear cases with high accuracy. The technique that SVM used to solve non-linear problem is kernel trick, which is to map data into a richer feature space including nonlinear features, then construct hyper-planes in that space. One of important parameters is the selection of kernels.

Optimal Model Selection:

The way to gain the optimal parameters of SVM is to use training set to train model and testing set to test its generalized capacity. Since the training set provided by Kaggle is too large to be trained. The original training set is split into a sub-training set (the first 1000 observations) and a testing set (from the 1001st to 1600th observations). In terms of the kernel we use, a comparison on predicting accuracy had been run among 5 commonly-used kernels (Radial Basis Gaussian kernel, Polynomial kernel, Linear kernel, Laplacian kernel and Bessel kernel).

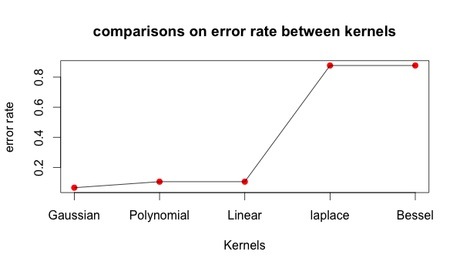


Figure 1 of SVM

It can be seen that Gaussian kernel is the best choice for us (see Figure 1 of SVM). Therefore, Gaussian Kernel is used on total classification of Kaggle’s testing set.

Kaggle Testing Outcome:

Classification Accuracy (provided by Kaggle): 0.97186 (See figure 2 of SVM).

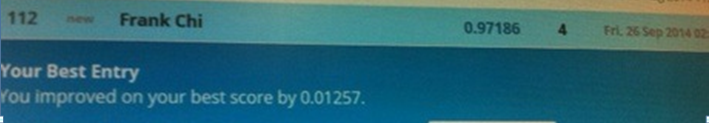


Figure 2 of SVM

Findings (answers to questions):

(1)Why the digit picture is non-linear?

Hinton, Geoffrey and Peter (1997) illustrated a highly nonlinear one-dimensional surface of digit images approximated by six locally linear models (see Figure 3 of SVM). It can be seen a support for that non-linear assumption.

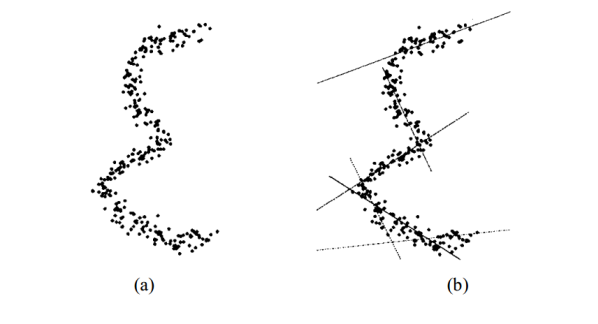


Figure 3 of SVM

(2)Why do you use SVM with kernel but not SVM without kernel?

Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts (Farid and Nahla 2012). In the previous comparison (see Figure 1 of SVM), it can be seen that some non-linear kernel like Gaussian Kernel can beat the linear kernel. This also proves that normally SVM with kernel performs better than the SVM without kernel trick.

Interpretation:

Philosophy of machine learning algorithms is likewise and interrelated. From the dimension-reducing technique of Fisher Discriminant Analysis (FDA) to the kernel trick of SVM, the core methodology is to project old feature space into new feature space. Also, the idea of maximizing within-group variance in FDA is similar to SVM’s maximum margin hyper-plane. The valuable yield from learning those data mining algorithm is to understand models’ philosophy, since all those algorithms are brought up by excellent statisticians or computer scientists. The benefit of mastering their way to think is far more meaningful and important than simple applications.

Limitations:

The main limitation about SVM is the lack of deeper research on how the different kernels work. Because that using different kernels means to project in different way. By deepening the research on inner mechanism of kernels, it is possible to improve the performance of classifiers by using appropriate kernels. This is known as Augmented Learning or Deep Learning, a extended research field of statistical learning.

Another limitation of this research is short of necessary dimension-reduction. By using PCA, PP or other techniques, we can remove redundant variables or information. In this way, the efficiency of algorithm will also be improved.

**Model comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **KNN** | **Decision tree** | **Random forest** | **SVM** |
| **Accuracy** | **0.9405** | **0.8588** | **0.96289** | **0.97186** |

From the comparison of those models, it can be concluded that support vector machine is the best model among four. It is easy to understand that Random forest is able to beat decision tree. This is because that RF is robust to over-fitting and its methodology of majority voting has better properties like flexibility and high-dimensional adaptability than single decision tree. This conclusion is not new for us. Breiman (2001) have proved random forest’s high performance in large-volume and high-dimensional data in his another valuable paper called Statistical Modeling: The Two Cultures. The following is the comparative table in that paper (see Figure 1 of Model comparison):

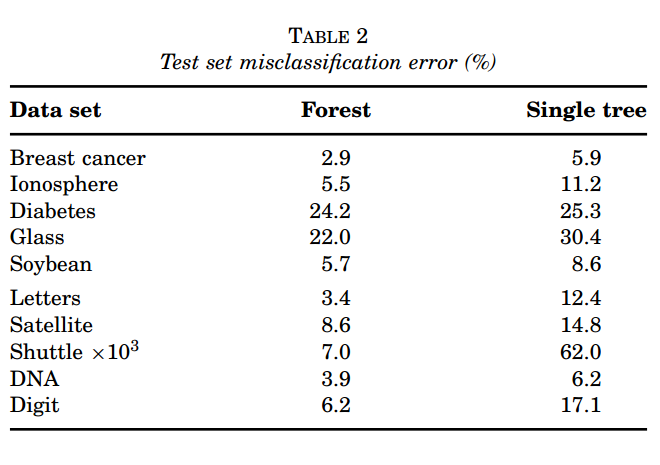


Figure 1 of Model comparison

Also, it is reasonable that Support Vector Machine performs better than KNN. This is because that both two models in based on distance to classify (one is Euclidean distance, one is one maximum margin). What makes SVM perform better than KNN is its kernel trick. The decision boundary of KNN is confined in limited dimensions. However, through projection, the decision boundary of SVM is able to exist in higher dimension. By dimension transformations, SVM’s decision boundaries or hyper-planes can conduct more oriented and flexible separation and classification.

In terms of the reason that SVM outperforms RF, it can be assumed that non-linear classifier have better property on classifying high-dimensional data. Actually, the difference on accuracy of two models is not large. It can only be proved that the SVM performs better than random forest in this data set. Also, from the perspective on the running-time and space-complexity, SVM is more practical for us to apply in real-world scenario.

**Referred Journal articles**

"[Analysis of a random forests model](http://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf)"

(Biau, Gérard.), Journal of Machine Learning Research, 2012, 1063-1095.

"[A Comparative Analysis for Support Vector Machines for Stroke Patients](http://www.wseas.us/e-library/conferences/2013/Dubrovnik/ECC/ECC-10.pdf)"

(Farid, Nahla, et al), Recent Advances in Information Science, 2012, 978-960.

"[Random forests](https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf)"

(Breiman, Leo.), Machine learning, 2001, 5-32.

"[Statistical modeling: The two cultures](http://projecteuclid.org/download/pdf_1/euclid.ss/1009213726)”

(Breiman, Leo**), Statistical Science** ,2001, 199-231.

"[Modeling the manifolds of images of handwritten digits](http://www.gatsby.ucl.ac.uk/~dayan/papers/hdr97.pdf)"

(Hinton, Geoffrey E., Peter Dayan, and Michael Revow), Neural Networks IEEE Transactions on 8.1, 1997, 65-74.

“[On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality](http://download.springer.com.ezproxy2.library.usyd.edu.au/static/pdf/766/art:10.1023/A:1009778005914.pdf?auth66=1414962617_764c27c2c7adcda1f328800921155e2d&ext=.pdf)”

(Friedman), **Data mining and knowledge discovery,** 1997, 55-77.

"[Support-vector networks](http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf)"

(Cortes, Corinna, and Vladimir Vapnik), Machine Learning, 1995, 273-297.

[“Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks”](http://www.mitpressjournals.org.ezproxy2.library.usyd.edu.au/doi/pdf/10.1162/neco.1991.3.3.440)

(Lee Yuchun), **Neural Computation**, 1991, 440-449.

“[Regression trees with unbiased variable selection and interaction detection](http://search.proquest.com.ezproxy1.library.usyd.edu.au/pqcentral/docview/217680801/F902EA9F3DE145D6PQ/1?accountid=14757)”

(Loh), **Statistica Sinica**, 2012, 12, 361-386.

["Classification Based on Tree-Structured Allocation Rules](http://search.proquest.com.ezproxy1.library.usyd.edu.au/pqcentral/docview/217680801/F902EA9F3DE145D6PQ/1?accountid=14757)”

(Vaughn and Wang), [Journal of Experimental Education](http://search.proquest.com.ezproxy1.library.usyd.edu.au/pqcentral/pubidlinkhandler/sng/pubtitle/The+Journal+of+Experimental+Education/$N/40928/DocView/217680801/fulltext/F902EA9F3DE145D6PQ/1?accountid=14757), 2008, 76(3), 315-325,328-329,334-340.

**Referred book Chapters**

“[Supervised learning](http://download.springer.com.ezproxy2.library.usyd.edu.au/static/pdf/677/bok:978-0-387-84858-7.pdf?auth66=1414899178_372180c292280b8d7f8fb2187ad0f240&ext=.pdf)”

(Hastie,Tibshirani and Friedman), in The Elements of Statistical Learning, 2009, Springer, ISBN 978-0-387-84857-0

“[Tree Based Methods](http://download.springer.com.ezproxy2.library.usyd.edu.au/static/pdf/677/bok:978-0-387-84858-7.pdf?auth66=1414899178_372180c292280b8d7f8fb2187ad0f240&ext=.pdf)”

(Hastie,Tibshirani and Friedman), in The Elements of Statistical Learning, 2009, Springer, ISBN 978-0-387-84857-0