University of Sydney

Machine Learning and Data Mining

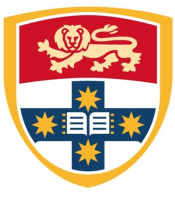
Handwritten Digit Recognition

Assignment 2

Authors:
Tinju Abraham - 450611748
Claudio Aracena - 450624988
Rafael Possas - 450645880
Tengfei Shan - 450267925

Lecturers:
Fabio Ramos
Roman Marchant

May 30, 2016



THE UNIVERSITY OF SYDNEY

Abstract

The use of algorithms in the recognition of human writing has become a standard after the popularization of computers with touch screens. Being able to understand digits and/or letters plays a major role in the Human-Computer interaction field. This work provides a novel approach on how the use of machine learning can leverage mathematics in order to accomplish this task. The main purpose of this work is to provide an example of how this task can be achieved with the use of some already popular algorithms in the field. A simple open source dataset comprised by digits (0-9) was used and their image representation and predictive model were achieved throughout this work. It has been proved that even though this problem looks complex, in reality it is not. The advancements in algorithms for classification and the development of data science frameworks helps to leverage the implementation of the solution for this problem. An experiment with 9 different algorithms and a brief study of the top 2 will be given in this paper along with some personal reflections on their performance.

1 Introduction

1.1 Problem description

Image recognition is a very popular topic in research, and also a broadly used approach in our daily life. As the machine learning and data mining theories developed, image recognition is now a mature technology. One of the very widely used application in this category is handwritten number recognition. It primarily remains a classification problem. Applications of this are many, such as automatic letter sort at post office and cheque processing in the bank.

In this paper, we focus on handwritten digits recognition by applying different algorithms. Algorithms like SVM, Random Forest, Logistic Regression, KNN, Linear Discriminant Analysis, Multinomial Naive Bayes, Gaussian Naive Bayes, AdaBoost, and Decission Trees are tried to train and test the data. One of the challenges in machine learning is that different algorithms provide different qualitative and quantitative results and this can have significant consequences in solutions of classifications problem. Comparison of results among these algorithms are also done in this work to come up with the best performing algorithm.

1.2 Dataset description

The dataset used in this experiment is Semeion Handwritten Digit Data Set,which is extracted from UCI machine learning repository [1]. About 80 individuals contributed to a total of 1593 handwritten digits. Every person wrote all 0-9 digits on a paper twice. They were asked to write the digits slowly and accurately for the first time, while fast and inaccurately for the second time. These digits were scanned, stretched in a rectangular box 16 x 16 in a gray scale of 256 values. Then a fixed threshold (127) was applied on each pixel of each image to scale into a boolean value (0/1). It means that the value would be 0 if the value is under 127 of the grey scale, or 1 if the value is above 127. In other words, every data in the 1593 records was scaled into a 16 x 16 square box (256 binary attributes).

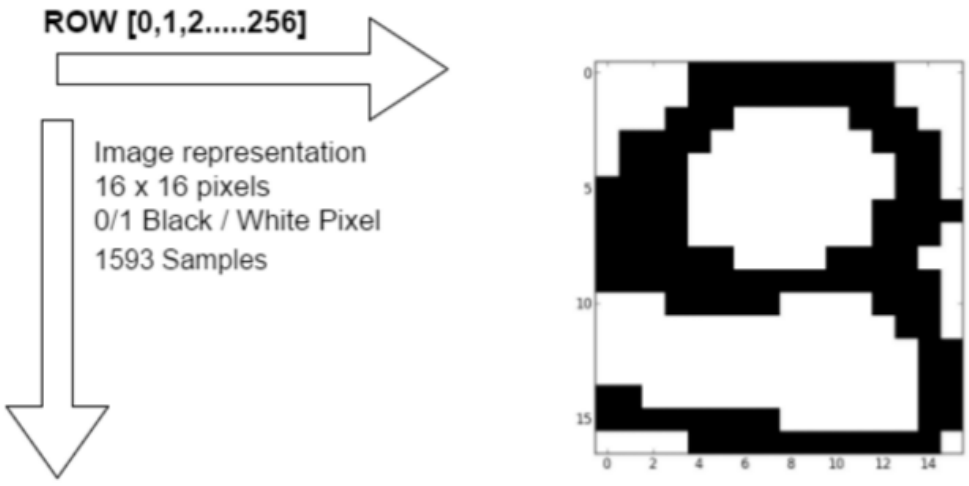


Figure 1: Data and Image Representation

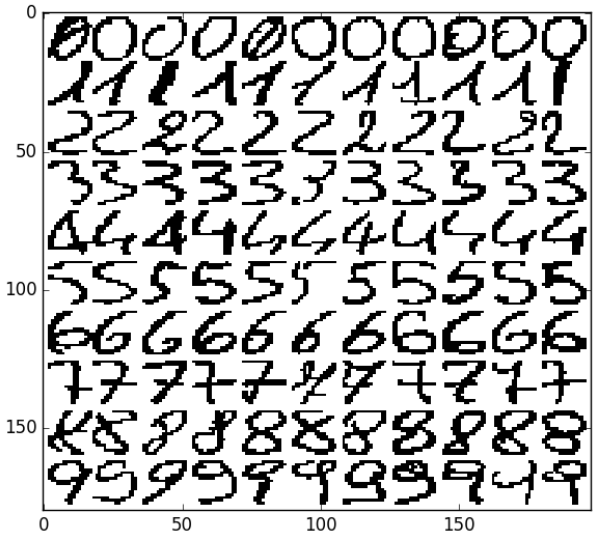


Figure 2: 100 digits sample from dataset

2 Methods and Experiments

2.1 Pre-processing

It is known that real world data is generally incomplete and full of inconsistent values. Techniques like pre-processing are useful in machine learning in order to make the data 'look' better before applying a specific predictive algorithm [2]. Generally, these methods can be classified in one of the following: data cleaning, data transformation, data reduction, discretization and normalization/regularization [3].

This work focuses in the data reduction technique as it helps to reduce the number of dimensions of the dataset. Dimensionality reduction was done through the use of PCA (Principal Component Analysis) which focuses in reducing the number of dimensions of the given matrix by finding the direction (principal components) in which the variance is the greatest [2].

Along with the dimensionality reduction, PCA was also used for compression. Each digit can be easily visualized by plotting a 16 x 16 matrix with the given pixels of the dataset. Therefore, the effects of compression can be easily seen after reconstructing the original matrix through the PCA inverse transformation.

2.2 Methods

In this section the algorithms which obtained more than 75% of accuracy are described. A detailed description is given for the algorithm with best performance, Support Vector Machine.

2.2.1 Support Vector Machine (SVM)

SVM performs classification by finding the hyperplane that maximizes the margin between two classes. The vectors that define the hyperplane are the support vectors. Given the labelled data (training set), the algorithm outputs an optimal hyperplane which categorizes the new examples.

We can draw a line on a graph of $x1$ vs $x2$ separating the two classes($y = -1\ or + 1$) when $D = 2$ and a hyperplane on graphs of $x1, x2...xD$ for when $D > 2$. This hyperplane can be described by:

$$w \cdot x + b = 0 \tag{1}$$

where w is normal to the hyperplane and $b/\|w\|$ is the perpendicular distance from the hyperplane to the origin.

Support vectors are the training examples that are closest to the hyperplane and the aim of SVM is to orientate this hyperplane to make it as far as possible from the closest members of both classes.

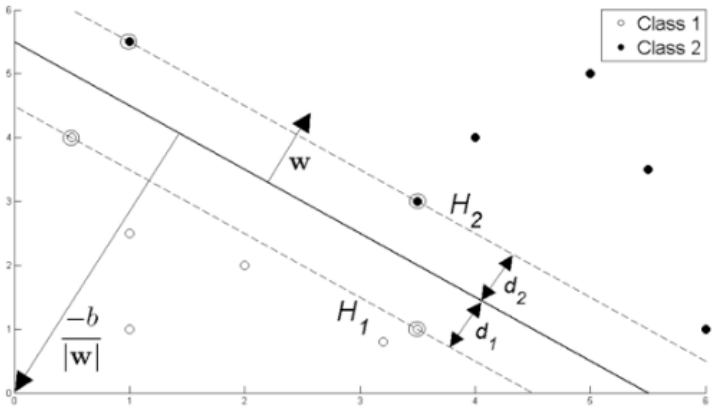


Figure 3: Hyperplane through two linearly separable classes

Referring to Figure 3, select w and b such that the training data can be described by:

$$x_i \cdot w + b \geq +1 \quad for\ y_i = +1 \tag{2}$$

$$x_i \cdot w + b \leq -1 \quad for\ y_i = -1 \tag{3}$$

These equations can be combined into:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i \tag{4}$$

For the points on the boundary (H1 and H2 in 3), it can be written as follows:

$$x_i \cdot w + b = +1\ or\ -1 \tag{5}$$

In order to orientate the hyperplane to be far from support vectors (maximize margin), the margin $1/\|w\|$ should be maximised. This is equivalent to minimising $\|w\|$ or $\frac{1}{2}\|w\|^2$:

$$min \frac{1}{2}\|w\|^2 \tag{6}$$

$$s.t\ y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i \tag{7}$$

In order to cater for the constraints in this minimization, we need to allocate them Lagrange multipliers α, where $\alpha_i \geq 0 \forall i$:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i y_i(x_i \cdot w + b) + \sum_{i=1}^{L} \alpha_i \tag{8}$$

This minimization can be done by finding w and b which minimizes, and the α which maximizes the above equation. This can be done by differentiating L_P with respect to w and b and setting the derivatives to 0. By doing that:

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \quad s.t.\ \alpha_i \geq 0 \quad \forall i \tag{9}$$

$$\sum_{i=1}^{L} \alpha_i y_i = 0 \tag{10}$$

Having moved from minimizing L_p to maximizing L_D, w and b can be calculated and each new data point can be classified. This new formulation L_D is referred to as the Dual form of the Primary L_P. The Dual form requires only the dot product of each input vector x_i to be calculated, this is important for the Kernel Trick [4].

Kernel Support Vector Machines The original SVM optimal hyperplane algorithm is a linear classifier. Kernel SVM is used to handle data which are not fully linearly separable. The basic idea is to gain linear separation by mapping the data to a higher dimensional space. $k(x_i, x_j)$ is an example of a family of functions called Kernel Functions. Kernel functions are basically based on the inner products of two vectors. This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $x \to \varphi(x)$, only inner products of the mapped inputs in the feature space need be determined without us needing to explicitly calculate φ.

Polynomial kernel

$$k(x, z) = (1 + x^T z)^2 \tag{11}$$

Gaussian Kernel

$$k(x, z) = exp(-\gamma \|x - z\|^2) \tag{12}$$

The library scikit-learn has an implementation of SVM using `sklearn.svm`. For the svm classifier, the important parameters used are 1) parameter `C`, which trades off the misclassification of training examples against simplicity of the decision surface. It is common for all kernel functions. A low `C` makes the decision surface smooth, while a high `C` aims at classifying all training examples correctly 2) parameter `gamma` defines how much influence a single training example has. The larger the `gamma` is, the closer other examples must be to be affected 3) parameter `Kernel` takes in the type of kernel function used for classification. The values can be linear, polynomial, sigmoid or rbf [5].

2.2.2 Random Forest

Random forest can be explained as a large collection of de-correlated trees. Trees can be used as a predictive model which maps observations about an item to conclusions about the item's target value. One drawback of this model are its propensity to overfit to the data. To avoid this issue random forest algorithm builds several random trees with a specific splitting process. Then, for a regression problem the resulting values of the trees are averaged to get the final predictions, and for a classification problem the most frequent predicted class by the trees is chosen as the final prediction [6].

As the problem under study is a classification task a random forest classifier was used. The scikit-learn library includes a class which implements random forest classifier, `sklearn.ensemble.RandomForestClassifier`. To create a `RandomForestClassifier` object the most important parameters to be defined are 1) number

of estimators or trees (`n_estimators`), 2) maximum depth of the trees from root to the leaves (`max_depth`), and 3) the number of features to consider when looking for the best split (`max_features`) [7].

2.2.3 Logistic Regression

Logistic regression sometimes called the logistic model or logit model, analyzes the relationship between multiple independent variables and a categorical dependent variable, and estimates the probability of occurrence of an event by fitting data to a logistic curve. This algorithm uses a sigmoid function, which maps the whole real line to [0,1], which is necessary for the output to be interpreted as a probability. To prevent overfitting, regularization can be used to avoid the parameters of the model reach high values and do not allow generalization [8].

The scikit-learn library includes an implementation of logistic regression in the class `sklearn.linear_model.LogisticRegression`. To create a `LogisticRegression` object the main parameter to be passed is 1) `C`, which is the inversed regularization strength (the higher `C` the less regularization is applied) [7].

2.2.4 K-Nearest Neighbour (KNN)

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point [9]. In k-nearest neighbor learning, the number of samples is specified by the user.

The scikit-learn library implements K nearest neighbor algorithm using `sklearn.neighbors.KNeighborsClassifier`. There are 3 important parameters used in this implementation 1) parameter `n_neighbors` represents the Number of neighbors 2) parameter `p`, an integer, represents the Power parameter for the Minkowski metric and 3) parameter `algorithm` takes the values auto, ball_tree, kd-tree, brute. It actually represents the choice of neighbors search algorithm. When the value is auto, the algorithm decides the best approach for training data [7].

2.2.5 Linear Discriminant Analysis (LDA)

Linear discriminant analysis is a classifier with a linear decision surface. It is an attractive classifier because it provides closed-form solutions that can be easily computed and its proven well in practice.It is inherently multiclass and has no hyperparameters to use. LDA can be used to perform supervised dimensionality reduction, by projecting the input data to a linear subspace consisting of the directions which maximize the separation between classes [10].

Linear Discriminant Analysis can be implemented by scikit-learn library using `sklearn.discriminant_analysis.LinearDiscriminantAnalysis`. The parameter `solver` decides if Singular value Decomposition(svd), Least squares solution(lsqr) or Eigenvalue decomposition(eigen) to be used by the algorithm in the classification task. Another parameter is `n_components` which represents the number of components for dimensionality reduction [7].

2.2.6 Multinomial Naive Bayes

Naive Bayes algorithms assume the independence of features and apply the Bayes' theorem for classification. Generally, NB classifiers require a small set of training data to estimate the parameters necessary for classification and they are quite fast in execution. Multinomial NB is a Naive Bayes algorithm for multinomially distributed data.

The scikit-learn library used for the implementation of Multinomial Naive Bayes classifier is `sklearn.naive_bayes.MultinomialNB`. The most important parameter used by this classifier is `alpha`, which is the smoothing parameter. The smoothing is a way of regularization and it accounts for the features not present in the learning samples and prevents zero probabilities in further computations [7].

2.2.7 Gaussian Naive Bayes

This is another Naive Bayes classifier and it implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian. The scikit-learn library used for the implementation of Gaussian Naive Bayes classifier is `sklearn.naive_bayes.GaussianNB`. No particular parameters have been tried in the configuration of this algorithm for this experiment [7].

2.3 Implementation Details

2.3.1 Parameter selection

As it is mentioned in the algorithm section, several algorithms were tried with several configurations. The Table 1 show all the combinations used and the best parameter configurations according its accuracy for each of them. Just algorithms which obtained more than 75% are shown.

Algorithm	Parameters	Combinations	Best parameter configuration
Support Vector Machine	C: 0.1, 1, 10, 100 gamma: auto, 0.03, 0.003 kernel: rbf, linear, poly, sigmoid	48	C: 10, gamma: 0.03 kernel: rbf
Random Forest	n_estimators: 10, 100, 300, 500 max_features: auto, sqrt, log2 max_depth: None, 5	24	n_estimators: 300 max_features: log2 max_depth: None
Logistic Regression	C: 0.1, 1, 10, 100	4	C:0.1
K-Nearest Neighbour	n_neighbors: 3, 5, 7 p: 1, 2, 3 algorithm: auto, ball_tree, kd_tree, brute	36	n_neighbors: 7 p: 3 algorithm: brute
Linear Discriminant Analysis	solver: svd, lsqr, eigen n_components:3, 5, 8	9	solver: eigen n_components:8
Multinomial Naive Bayes	alpha: 0.1, 0.3, 1	3	alpha: 1
Gaussian Naive Bayes	-	1	-

Table 1: Parameter configurations for algorithms

In addition to the previous parameters, some algorithms accept an optional parameter to make a parallel calculation, such algorithms are Random forest, Logistic regression, and K-nearest neighbors. This behavior can be used just if the parent task running the algorithms is not running in multiprocessing mode.

2.3.2 Cross validation

The main goal of cross-validation is to ensure that every sampling from the original set has the probability of appearing in the training and testing set. The k-fold cross validation strategy is that where the original sample is randomly partitioned into k subsamples and one is left out each iteration [11]. In the cross validation evaluation procedure experimented in the classifier, 10% of the given training dataset is taken as test data, the remaining 90% data (which is considered as the training data in that trial) has been used by the classifier to predict the classes of the test data and this procedure is repeated 10 (k=10) times.

For this assignment, in every experiment a 10-fold cross-validation procedure was applied. The library scikit-learn implements this process with its class `sklearn.cross_validation` which includes the function `cross_val_score` that calculates a score specified by the user using cross-validation [7].

2.3.3 Multiprocessing and Profiling

In order to speed up the task which run all the algorithms a multiprocessing procedure was implemented and a profile of the task with and without multiprocessing was done. To implement the multiprocessing procedure the Python library joblib was used and for profiling the algorithms the package cProfile was used.

The results of applying multiprocessing are considerable. According to the profile without applying multiprocessing, run all the algorithms it takes 470.9 seconds with 80,692,163 function calls. The same process

but implementing multiprocessing takes 151.2 seconds with 104,841 function calls.

2.3.4 Hardware and Software Specifications

The computer where the experiments were run has the next specifications: Intel i7-6700k 8 cores 4.2Ghz 16GB RAM 512 GB SSD.

The development of the project was done in the software tool PyCharm using Python 3.5.

2.4 Experiments

As machine learning has a broad spectrum of different algorithms, it is required to thoroughly test the highest number of possible combinations in order to achieve optimal results. This work was focused in testing 9 popular algorithms: SVM, Random Forest, Logistic Regression, KNN, Linear Discriminant Analysis, Multinomial Naive Bayes, Gaussian Naive Bayes, AdaBoost, and Decision Trees. In total, there were 169 different combinations of algorithms/configuration and from these we took the top 2 algorithms. Table 2 shows the best result of each algorithm. As all classes have a similar number of instances, accuracy was used as a metric of comparison.

Algorithm	Accuracy	Parameters
Support Vector Machine	95.92%	C=10, gamma=0.03, kernel= rbf
Random Forest	94.96%	n_estimators=300, max_features=log2, max_depth=None
Logistic Regression	91.58%	C=0.1, multi_class=ovr
K-Nearest Neighbors	90.82%	n_neighbors=7, p=3, algorithm=brute
Linear Discriminant Analysis	88.65%	solver=eigen, n_components=8
Multinomial Naïve Bayes	84.58%	alpha=1
Gaussian Naïve Bayes	79.21%	
AdaBoost	71.68%	n_estimators=10, learning_rate=0.3
Decision Trees	70.34%	max_depth=None, max_features=sqrt

Table 2: Best results for each algorithm

As it can be seen, Support Vector Machines and Random Forest were the best algorithms for this task. A thorough discussion will be given along with a personal reflection on why these two algorithms performed the best in the next sections.

2.4.1 PCA for Data Compression and Dimensionality Reduction

Matrix decomposition is an useful way of reducing the amount of processing and memory needed by a computer to process matrices where the number of features (columns) are very big [3]. However, when using images, decomposition for data compression can also be easily identified by the reduction in the overall 'quality' of the picture. Figure 4, shows the digits representations of 3 different explained variances of PCA. For both 90% and 50% explained variance, it can be seen that the digits are still recognizable, which can be supported by the 96% and 93% accuracy obtained through applying our predictive algorithms in the reduced dataset. Still supporting this conclusion, it can be noted that with only 10% of the explained variance, the numbers are barely recognizable, and, as a result, yielded a poor accuracy (49%).

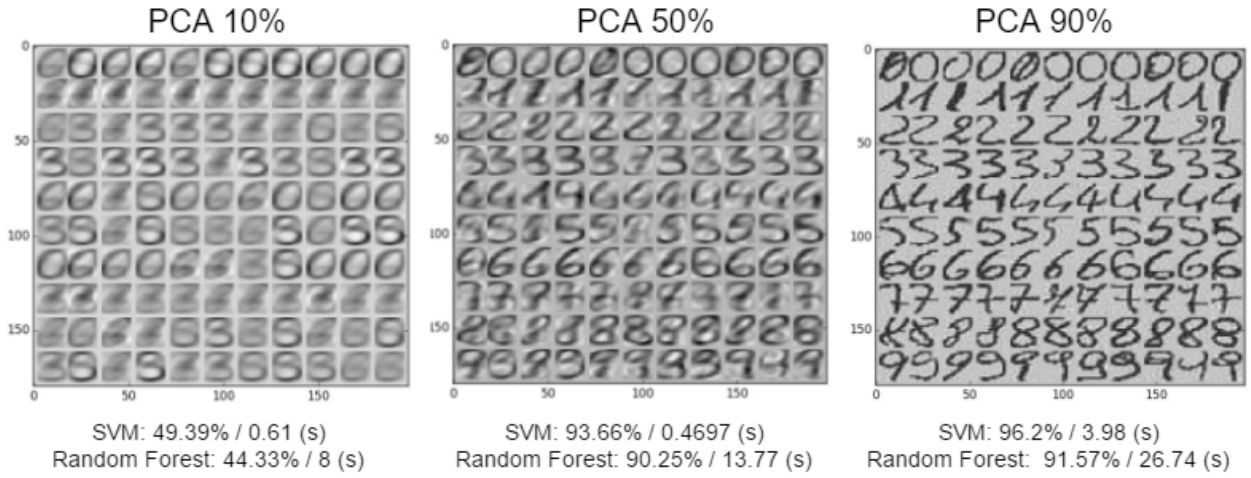


Figure 4: Resutls applying PCA and reconstruction of images using different explained variances

2.5 Comparison with MNIST

Over the last year many competitions in handwritten digit recognition have been held in the MNIST (Mixed National Institute of Standards and Technology database). This dataset contains 60,000 examples of training set and 10,000 examples of test set. In 2002, DeCoste and Scholkopf reached 99.44% accuracy rate when they applied virtual SVM algorithm (deg-9 poly, 2-pixel jittered) [12]. In their research, the handwritten images were positioned in a 28 x 28 features (pixels) for classification. Their features were gray scale pixels ranging from 0 to 255.

Comparing to their high resolution ratio, this work uses 16 x 16 features (pixels), and assigned only boolean value (0/1) to the field. In order to reach higher accuracy, there are few things that can be improved in this experiment. First, increase the size of features that contains the handwritten image, for instance, using 28 x 28 images. Second, apply gray scale range instead of boolean features which can give more information for each feature. Third, optimize the algorithm to fit the problem itself trying more options for tuning and testing. Finally, another factor that can influence the accuracy level is the size of the data set. The current dataset contains only 1,593 instances which is really small compared to MNIST. A bigger dataset should improve the accuracy of the classification task.

3 Discussion

SVM is a supervised learning model that analyses data and is a suitable option for recognizing patterns. Some of the best characteristics of SVM are 1) memory efficiency - as it uses only a subset of training points called support vectors in the decision function 2) Versatility - different Kernel functions can be specified for the decision function. Recent researches show that SVM demonstrated its accuracy, space efficiency and time efficiency in handwritten digit recognition [13]. Many improvements by combining SVM with other algorithms were conducted to increase its accuracy [14]. In 1998, B. Scholkopf etc. applied virtual SVM on handwritten digit recognization, it reached 99.20% accuracy rate [15]. In 2002, D. DeCoste and B. Scholkopf. used virtual SVM (jitter) to get 99.37% accuracy rate [12]. In our experiment also, SVM outperformed other algorithms in terms of accuracy for the chosen dataset. Therefore, SVM is selected as the the main topic of discussion in this report.

Although SVM has all the aforementioned advantages, it is worth to note its caveats as there is still place for improvement in the handwriting recognition field of study. In 2009, Marthias M proposed an alternative implementation of the popular SVM algorithm. In his studies, Marthias states that “SVM classifiers, like other kernel machines, gives a poor generalization when the hyperparameters are not tuned efficiently”. Thus, his work proposed a model selection technique for SVM using empirical error criterion through the LOO cross-validation procedure. LOO is the special case of cross-validation where the number K of folds is equal to the training set size. All the tests were also applied in the MNIST database, and the algorithm achieved higher performance when compared to the traditional SVM model [16].

On the other hand, Random Forest is still an extremely efficient and popular algorithm nowadays. The algorithm presents a straightforward implementation, a reasonable computing cost and a very good performance in terms of classification accuracy. However, in all the tests it was still outperformed by SVM. Studies like Bernand S, 2009 reveals good variants of the Random Forest algorithm in the MNIST database but they still reach poor accuracy when compared to the SVM results [17].

This study leads us to believe that SVM presents one of the best models for Handwritten recognition. Working on model selection and kernel parameters tuning is a good start for improving results even further.

4 Conclusions and future work

The development of this work has proved that machine learning can be a powerful tool in the handwriting recognition field. Even though the dataset used in this paper had limited features and a low amount of samples, it is believed that the results are already satisfactory for this problem. A good example of how a better dataset can help to improve the results is provided by the MNIST (Mixed National Institute of Standards and Technology database) work as their accuracy was close to 100%. Given more research time it could be possible to improve the accuracy by performing some feature engineering in the dataset along with more experiments with predictive algorithms.

Some possible future work could be to include recent techniques that have shown remarkable results in MNIST competitions. For example, deep learning techniques are the state of the art in this kind of problems reaching a test rate error of 0.21% (99.79% of accuracy) by Li Wan et al. using a Convolutional Neural Networks with DropConnect methodology in 2013 [18]. However, to reach these results the amount of data need to be bigger than what actually the chosen dataset has, but still it will be possible to improve the results shown in this report.

All in all, this work not only provided very good insights in how to use machine learning techniques but also helped the understanding of the concepts taught during the Knowledge Discovery subject.

References

[1] Semeion Research Center of Sciences of Communication. Semeion handwritten digit data set, 1994. [Online; accessed 30-May-2016].

[2] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn, 2007.

[3] Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012.

[4] Tristan Fletcher. Support vector machines explained. *Online]. http://sutikno. blog. undip. ac. id/files/2011/11/SVM-Explained. pdf.[Accessed 06 06 2013]*, 2009.

[5] Scikit-learn.org. Support vector machines — scikit-learn 0.17.1 documentation, 2016. [Online; accessed 30-May-2016].

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Hyeoun Park. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43(2):154–164, 2013.

[9] Scikit-learn.org. Nearest neighbors — scikit-learn 0.17.1 documentation, 2016. [Online; accessed 30-May-2016].

[10] Scikit-learn.org. Linear and quadratic discriminant analysis — scikit-learn 0.17.1 documentation, 2016. [Online; accessed 30-May-2016].

[11] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[12] Dennis Decoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine learning*, 46(1-3):161–190, 2002.

[13] Olga Kouropteva, Oleg Okun, and Matti Pietikäinen. Classification of handwritten digits using supervised locally linear embedding algorithm and support vector machine. In *ESANN*, pages 229–234. Citeseer, 2003.

[14] Daniel Keysers. Comparison and combination of state-of-the-art techniques for handwritten character recognition: topping the mnist benchmark. *arXiv preprint arXiv:0710.2231*, 2007.

[15] Bernhard Schölkopf, Patrice Simard, Alexander J Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. *Advances in neural information processing systems*, pages 640–646, 1998.

[16] Mathias M Adankon and Mohamed Cheriet. Model selection for the ls-svm. application to handwriting recognition. *Pattern Recognition*, 42(12):3264–3270, 2009.

[17] Simon Bernard, Sébastien Adam, and Laurent Heutte. Using random forests for handwritten digit recognition. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1043–1047. IEEE, 2007.

[18] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[19] Scikit-learn.org. Naive bayes — scikit-learn 0.17.1 documentation, 2016. [Online; accessed 30-May-2016].

5 Appendix

5.1 Instructions

The instructions of how to run the code are inside the README.md file.

5.2 Links to external open-source libraries used for analyses

- Semeion Repository - `https://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit`
- Python - `https://www.python.org/`
- scikit-learn - `http://scikit-learn.org/`
- Pandas - `http://pandas.pydata.org/`
- matplotlib - `http://matplotlib.org/`
- joblib - `https://pythonhosted.org/joblib/`
- cProfile - `https://docs.python.org/3.5/library/profile.html`

5.3 Contribution of each member

- Tinju Abraham - 450611748: Description of algorithms, report
- Claudio Aracena - 450624988: Code, algorithms, implementations details, latex files
- Rafael Possas - 450645880: Code, algorithms, discussion, pre-processing, presentation
- Tengfei Shan - 450267925: Comparison MNIST, discussion