A Primal Support Vector Machine for Handwritten Character Recognition using a Bag of Visual Words

(Bachelor's Project)

Anne Wanningen, s2219832, a.wanningen@gmail.com,
M.A. Wiering*

October 5, 2016

Abstract

Using a bag of visual words for unsupervised feature learning, a system of handwritten character recognition is developed using a support vector machine (SVM) for which the update rules are derived directly from the primary objective function. The system is tested on the MNIST dataset and outperforms a traditional SVM implementation in this particular set-up, not only in terms of accuracy but most notably in terms of computation time. Whether this approach should be labelled as part of the SVM family of algorithms remains a point of discussion.

1 Introduction

If one may assume as a necessary quality of intelligence the ability to make the correct choice given a situation and a set of possibilities, then for any intelligent system to perform well on non-trivial tasks the ability to generalize must be considered paramount. Generalization, by classifying sets of observations, reduces the necessary complexity of information-action mapping.

Handwritten Character Recognition

As classification in a high dimensional input space is a challenging field of research, the need for well-defined training data is very real. The fact that visual data are easily digitalized and available in a near infinite supply makes for a very appealing source of data for the topic.

*University of Groningen, Department of Artificial Intelligence

Within the domain of image classification, the topic of handwritten character recognition may be favoured above all other possible subcategories because of its systematic nature. All possible classes are well defined and each of those classes has prominent features that set it apart from other classes, a necessary consequence of the nature of this type of data; handwriting is after all meant to be classified.

Goal

The aim of the research of which this thesis is a part is to develop a system of image classification capable of state of the art performance, using proven techniques in a not often encountered set up. This thesis will focus on the classifier of this system, an SVM using an alternative mathematical approach, in which the update formulas are derived directly from the primary objective function.

Related Work

Other research shows that working from the primary objective function -instead of resorting to Lagrange multipliers as is usually done in traditional SVM theory- can be done and is indeed feasible in terms of computation time. One approach is to use smoothing to approximate the non-differentiable hinge-loss function that makes up a substantial part of the primary objective function. As was shown by Chapelle in (Chapelle, 2007), standard optimization methods may consequently be applied. Another approach would be using sub-gradient methods (Shor, 1985), as done by Zhang in

(Zhang, 2004). Another subgradient oriented solution is the PEGASOS algorithm (Shalev-Shwartz, 2007), though this last method is limited to linear SVMs.

Outline

In the second section the conceptual considerations for feature extraction will be presented, followed by the theoretical basis for SVM classification. Then the alternative approach will be presented. In the third section the experimental setup will be presented in detail. Section four will contain the results of this experiment and section five will list all the implications that follow from these results. The final section will present the reader with observations to consider when interpreting those conclusions.

2 The System

2.1 Unsupervised Feature Learning

In order for a fully automated classification system to function as expected, some function $F(\cdot)$ must be designed to operate on the data, describing them systematically whilst reducing dimensionality without reducing the amount of meaningful information within the data unnecessarily.
Systematic construction and repeated pattern usage are core principles necessary for any information describing system of human readable format. Taking this into consideration it makes sense to look for a form of $F(\cdot)$ that can describe the data in terms of these repeating patterns. Regardless of what these patterns might look like, observing such structural repetitions is a significant step towards describing the data in a meaningful way.
As it is unnecessary to have any inclination about the form of these structural building blocks, it suffices to find as many repeating patterns as possible. A sensible way to accomplish this step would be to systematically partition all individual data points of a sample of the training data, as long as the diversity in this sample is sufficient to represent the training data as a whole. Next, in order to find the most meaningful patches of data some function $d(\cdot, \cdot)$ to quantitatively describe the similarity between these patches is required.
Assuming $d(\cdot, \cdot)$ to provide a sufficiently accurate indication of similarity, clustering these patches in order to find meaningful representations of repeating patterns within the data may be considered a trivial task.
Given the distance function $d(\cdot, \cdot)$, as well as a set of repeating patterns V, one possible form for $F(\cdot)$ would be to describe a data point in terms of the similarity to each element in V, using $d(\cdot, \cdot)$. This would result in a feature vector in $\Re^n$ with n being the number of centroids as found in the clustering process.
An intuitive analogy with this method is a dictionary, as it lists the different structural segments that make up a sentence. The described method is known as the *bag of words* (BOW) model, as first formally introduced for application in computer vision in (Zhu, Rao, and Zhang, 2002) and further explored in (Csurka, Bray, Dance, and Fan, 2004). This method was also successfully applied to the CIFAR-10 dataset (Krizhevsky and Hinton, 2009) in (Coates, Lee, and Ng, 2011), which served to inspire this research.

2.2 Classification

Using the previously described method of data interpretation for feature vector construction, the next step would be classification. Given the amount of high-level information extracted in such a method, it should be possible to use a linear classification method, provided this method is capable of dealing with data that are presumably not perfectly linearly separable.
The *support vector machine* (SVM) (Cortes and Vapnik, 1995) is capable of performing just such a classification task, as also shown by Csurka et al. in (Csurka et al., 2004) using an approach very similar to the setup used in this work.

2.2.1 Previous Work and Theoretical Basis

The SVM classifier aims to find a vector $H \in \Re^n$ that divides the feature space with a maximum margin, such that taking the inner product with a data point $x \in \Re^n$ returns a value equal to, or bigger than 1 in case x belongs to the class that the SVM is trained to recognise. In order for this $\Re^n \to \Re^1$ mapping to appear around the zero-mark, a translation value b (bias) is added after the vector multiplication. This results in a classification

function $f(\cdot)$ as described in equation 2.1.

$$f(x) = x \cdot H + b \tag{2.1}$$

The output of such a classification function combined with a class value definition $y_i \in \{-1, 1\}$ for every x_i such that $y_i = 1$ if the class of x_i is the class that the SVM is trained to recognize and $y_i = -1$ otherwise, enables the construction of a product $y_i f(x_i)$ that indicates incorrect classification with a negative value, and margin violation that suggests hyperplane optimization necessity with positive values below 1.
The margin by which H is able to separate the data is equal to the smallest distance between the data on either side of H. This distance can be described as:

$$m = \min_{(x_i:y_i=1)} \frac{x_i \cdot H}{\parallel H \parallel} - \max_{(x_j:y_j=-1)} \frac{x_j \cdot H}{\parallel H \parallel} \tag{2.2}$$

If the data are linearly separable then the following inequality should hold for the optimal hyperplane H_o:

$$\forall x \in X : y_i f(x_i) \geq 1 \tag{2.3}$$

This, in combination with equation 2.2 means the margin for H_o may be expressed as:

$$m_{H_o} = \frac{2}{\parallel H_o \parallel} \tag{2.4}$$

Therefore, in order to maximize the margin by which H separates the training data X, $\parallel H \parallel$ should be minimized.
Cortes and Vapnik showed in their original paper (Cortes and Vapnik, 1995)(appendix A) that H can be expressed as a linear combination of training data. In order to find a solution for H that approaches maximum margin separation of the training data H can be expressed as:

$$H = \sum_i \alpha_i y_i x_i \tag{2.5}$$

With $\alpha_i > 0$ for those data points closest to H (and therefore referred to as the *support vectors*) and $\alpha = 0$ for all other training data.
So to approach H_o, one must find a weight vector $\alpha = \{\alpha_1, \alpha_2 ... \alpha_m\}$ that performs well on classifying the training data using $f(\cdot)$, with H constructed using equation 2.5. To solve this optimization problem a function is needed to measure the distance between H and H_o, which we want to minimize. This primary objective function as it is often referred to, consists of two terms. The first term represents the desired margin maximization. This can be achieved by minimizing H.
The conclusions concerning margin maximization as described above do not yet take into consideration the possibility that the data are in fact not linearly separable. In order to find H_o for such data an error term ξ_i is introduced for every $x_i \in X$. All error terms are assigned such that all errors in the dataset are accounted for, so the training data may be treated as linearly separable. ξ is therefore defined as $\xi_i \geq 1 - y_i f(x_i)$, resulting in:

$$\xi_i = \max(1 - y_i f(x_i), 0) \tag{2.6}$$

Such that we may now assume:

$$\forall x_i \in X : y_i f(x_i) + \xi_i \geq 1 \tag{2.7}$$

This second term, representing the total margin violation over the entire training set by summing over all ξ_i, must be minimized as well, in order to approach H_o and therefore maximum classification accuracy on X. Considering these two heuristics the primary objective function P then becomes

$$P = \tfrac{1}{2} \parallel H \parallel^2 + C \sum_{i=0}^{n} \xi_i \tag{2.8}$$

In which C acts as a meta-parameter controlling the balance between margin maximization and error tolerance.
To minimize this function, which is convex for any convex error function, the usual approach is gradient descent. As the derivative of the hinge loss function used to describe ξ (equation 2.6) is non-trivial as a consequence of the maximum-value function being non-continuous, the approach to find H as proposed originally by Cortes and Vapnik focusses on a reformulation of P using Lagrange multipliers known as the *dual objective form*. The possibility to solve the optimization problem in its primal form

has been pointed out originally by Cortes and Vapnik as well, but this option was not further explored in that paper.

2.2.2 Working from the Primal Form

As pointed out in the previous section, one difficulty with optimization in the primal form is the non-continuous nature of the loss function (see equation 2.6). But considering all the difficulties caused by the max function, one should take notice of the binary nature of this function. It covers only two possible situations; either (1) $1 - y_i f(x_i) > 0$ or (2) $1 - y_i f(x_i) \leq 0$. Both situations can individually be described with continuous, differentiable functions;

$$(1) \quad p_{x_i} = \tfrac{1}{2} \| H \|^2 + C(1 - y_i f(x_i)) \tag{2.9}$$

$$(2) \quad \hat{p}_{x_i} = \tfrac{1}{2} \| H \|^2 \tag{2.10}$$

It is therefore possible to construct two derivatives, provided they be correctly applied in every situation. This requirement however may be considered trivial implementation wise, given the outcome of $1 - y_x f(x)$.

Cortes and Vapnik described H in terms of the training data, by finding the best fitting vector of Lagrange multipliers α to define H as in equation 2.5.
Since H is a vector in $\Re^n$, and the primary objective function (equation 2.8) is aimed at optimizing H regardless of how H is constructed it should be possible to find H directly using the derivative functions 2.9 and 2.10. To avoid confusion this vector will be referred to as W, as W can also be interpreted as a weight vector describing how much each dimension of a data point contributes towards the final classification outcome.
Considering the first order derivatives to the conditional formulas 2.9 and 2.10 as well as equations 2.5 and 2.1 towards the members of the weight vector W:

$$\frac{\partial p_{x_i}}{\partial w_j} = \frac{\partial \frac{1}{2} \| W \|^2}{\partial w_j} - C y_x \frac{\partial f(x)}{\partial w_j} \tag{2.11}$$

with:

$$\frac{\partial f(x)}{\partial w_j} = x_{i,j} \tag{2.12}$$

$$\frac{\partial \frac{1}{2} \| W \|^2}{\partial w_j} = w_j \tag{2.13}$$

In which $x_{i,j}$ denotes dimension j of data point i. The final derivatives for the conditional primary objective functions described in equations 2.9 and 2.10 therefore become as follows:

$$(1) \quad \frac{\partial p_{x_i}}{\partial w_j} = w_j - C y_i x_{i,j} \tag{2.14}$$

$$(2) \quad \frac{\partial \hat{p}_{x_i}}{\partial w_j} = w_j \tag{2.15}$$

In order to apply the SVM to multi-class problems we take the one-against-all approach. Given a l-class problem, we train l SVMs, each distinguishing images from some category c from images from all the other $l - 1$ categories. Given a query image, we assign it to the class with the largest SVM output $f_c(x)$.

3 Experimental Setup

In this section the experimental setup used to generate the results presented in the next section is described in detail.
The performance of an SVM using the proposed partial derivatives as described in the previous section (hereafter L1) is compared to an SVM using the standard formulas (hereafter STD) as first described in (Cortes and Vapnik, 1995) in combination with gradient descent. An L2-regularized version (hereafter L2) of the proposed approach has been examined as well.

3.1 Data

Using bicubic interpolation (Keys, 1981) the 28x28 MNIST (Lecun, Bottou, Bengio, and Haffner, 1998) greyscale images are upscaled to 36x36 pixels.

3.2 Codebook Construction

First a codebook of visual words is constructed from the training data X. To ensure perfect reproducibility for each run, the system's random number sequence is set to a predetermined initialisation point (random seed $\in \{1, 500, 1269\}$). The pseudo-random number generator is used for data selection in codebook generation as well as in training set and test set construction. 500000 (pseudo-) random patches of 6x6 pixels are extracted from the training data. Using the pixel intensity values of each patch results in one vector in $\Re^{36}$ for every patch. These 500000 vectors are then clustered using K-means with 100 iterations and $x \mid x \in \{50, 100, 150, 200\}$ centroids, initialised at randomly selected samples. The resulting centroids are individually standardized, compensating for zero-vectors by adding a fixed value of 10 to the variance before calculating the standard deviation. These results are stored together as a codebook of visual words. A separate codebook is generated for each random seed and for each number of centroids, resulting in twelve distinct codebooks for this experiment.

3.3 Feature Vector Calculation

In order to construct a feature vector from a training data point x, a sliding window of 6x6 pixels samples each quadrant (18x18) of the image without padding with a stride of one pixel, resulting in four collections q of 169 patches p for every image, as illustrated in figure 1. For each of these collections an activation value a is calculated per visual word v as a sum over all patches in that collection, calculating per patch the euclidean distance d to that visual word and subtracting that from the mean distance $\bar{d}$ to that word for this collection, with a lower bound of zero, as described in equation 3.1.

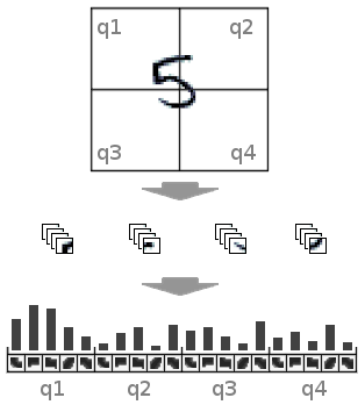


Figure 1: The process of feature vector calculation.

$$a_{q,v} = \sum_{p \in q} max(0, \bar{d}_{q,v} - d(v,p)) \tag{3.1}$$

with:

$$d(v,p) = \sqrt{\sum_{i \in v} v_i^2 + \sum_{i \in p} p_i^2 - 2 \sum_{i \in p} p_i \cdot v_i} \tag{3.2}$$

$$\bar{d}_{q,v} = \sum_{p \in q} \frac{d(p,v)}{|(p \mid p \in q)|} \tag{3.3}$$

The final feature vector F_x consists of the concatenated (left to right, top to bottom) results per quadrant. For STD a Gram matrix is constructed by calculating the inner product of all combinations of feature vectors.

3.4 Training for L1 and L2

For each class a separate SVM is trained as a one versus all classifier. In the case of L1 and L2 the training procedure for every SVM c is as follows:

1. Take the zero-vector as a starting point for the weight vector W.
2. Process data sample x using output function $f_c(\cdot)$ as described in equation 3.8.
3. If $f_c(x)y_x < 1$ update W_c for every $W_c \in W$ using equation 3.4 in case of L1 and equation 3.5 in case of L2.
4. If $f_c(x)y_x \geq 1$ update W_c for every $W_c \in W$ using equation 3.6

5. Update the bias b_c by adding the margin violation multiplied with the learning rate η as described in equation 3.7.

$$W_{c,v} \leftarrow W_{c,v} - \eta(\frac{W_{c,v}}{C} - y_x F_{x,v}) \quad (3.4)$$

$$W_{c,v} \leftarrow W_{c,v} - \eta(\frac{W_{c,v}}{C} - (1 - f_c(x) y_x) y_x F_{x,v}) \quad (3.5)$$

$$W_{c,v} \leftarrow W_{c,v} - \eta \frac{W_{c,v}}{C} \quad (3.6)$$

$$b_c \leftarrow b_c + \eta(y_x - f_c(x)) \quad (3.7)$$

with:

$$f_c(x) = \sum_{v \in W_c} (W_{c,v} F_{x,v}) + b_c \quad (3.8)$$

Both methods use a learning rate of $1e{-}6$, a C value of 1000 and 10000 iterations over the training data.

3.5 Training for STD

In the case of STD the training procedure for every SVM c is as follows:

1. Update the Lagrange multipliers α using equation 3.9.
2. Calculate the sum of the products of all α and their corresponding y_x values: $\sum_{x \in X} (\alpha_x y_x)$
3. If this sum exceeds the alpha constraint value $1e-5$, iteratively update all α values using equation 3.10 until this constraint is met.
4. After 500 iterations calculate the bias as the average margin violation per support vector as described in equation 3.11.

$$\alpha_i \leftarrow \alpha_i + \eta(1 - \sum_{j \in X} (\alpha_j y_i y_j K(x_i, x_j))) \quad (3.9)$$

$$\alpha_i \leftarrow \min(\max(\alpha_i - 2\eta\alpha_i \sum_{x \in X} y_x \alpha_x, 0), C) \quad (3.10)$$

$$b_c = \frac{\sum_{x \in X} (y_x - f_c(x)) s(\alpha_x)}{\sum_{x \in X} s(\alpha_x)} \quad (3.11)$$

with:

$$s(\alpha_x) = sgn(max(0, \alpha_x)\ max(0, C - \alpha_x)) \quad (3.12)$$

$$f_c(x) = \sum_{x_d \in X} (\alpha_{x_d} y_d K(x_d, x)) + b_c \quad (3.13)$$

$$K(x, x_d) = F_{x_d} \cdot F_x \quad (3.14)$$

Using a learning rate η of $1e-6$ and a C value of 1000.

3.6 Testing

After training each method using 10000 training images, classification accuracy was computed by testing on 1000 pseudo-randomly selected data samples that were not used for training.

4 Experimental Results

The results of the experiment are displayed in Figure 2.

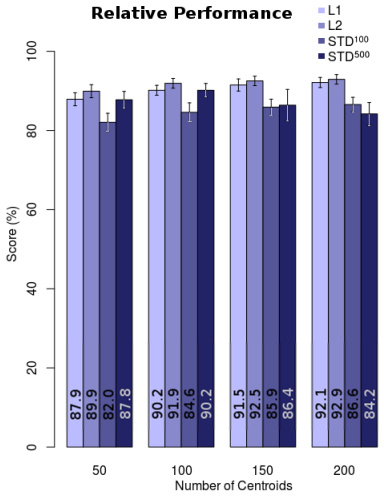
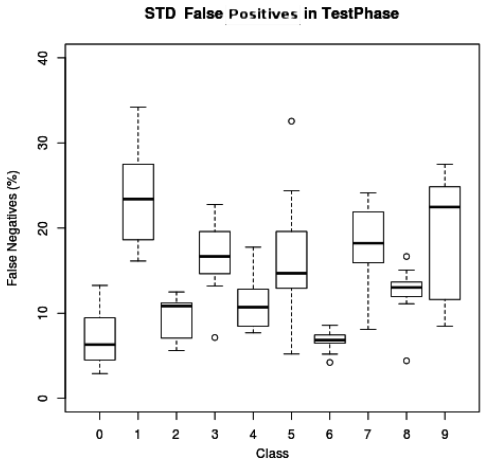
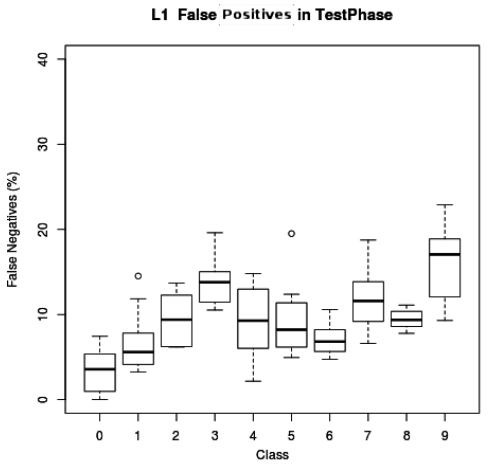


Figure 2: Classification accuracy of L1, L2 and STD. Note that STD was performed with 100 as well as 500 iterations.

5 Implications

The following are the conclusions drawn from the previously described results. In the next and final section of this thesis these conclusions will be





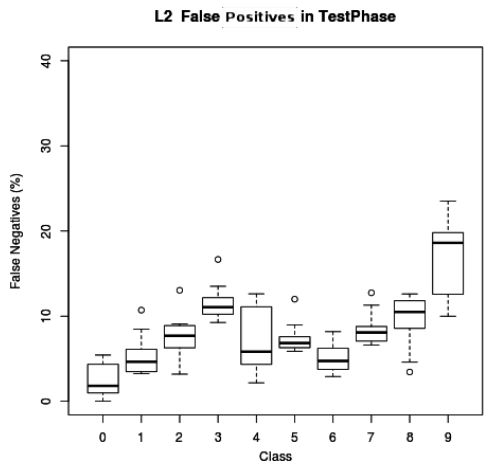
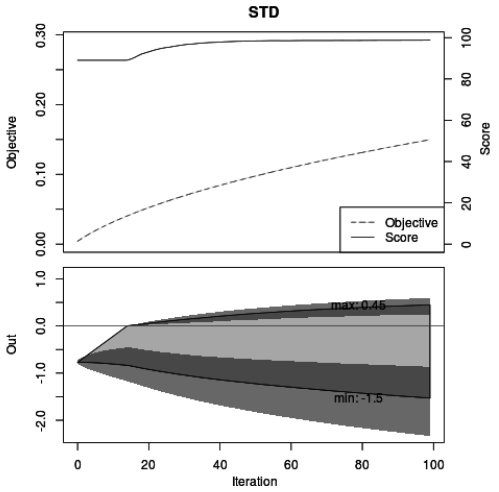
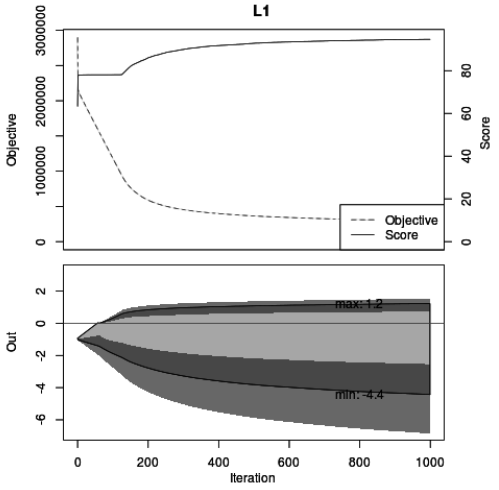


Figure 3: Spread of all false positives per class, per method.





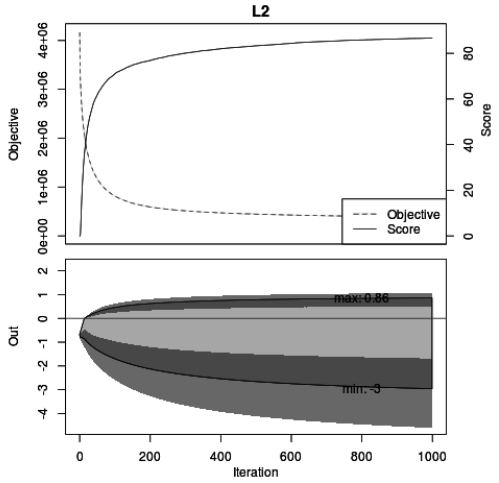


Figure 4: Training characteristics per method, with the upper graph depicting the different objective functions and classification accuracy on the training data as training progresses, and the lower graph depicting the mean numerical output of this SVM for this class (digit 1) as well as the standard deviation, as training progresses.

reviewed in a more critical sense.

5.1 Performance

The results of the experiment are suggestive of increased classification accuracy for L1 and L2, using this dataset, given these particular methods of information extraction and dimensionality reduction. L1 and L2 produce a typical pattern of false positive classification errors across the different classes (see figure 3). It is to be expected that some classes are more prone to this type of misclassification than others, as similarity between classes differs depending on the applied information extraction methods. This pattern of class-dependent misclassification susceptibility is therefore taken as further indication of successful training.
In the graphs of figure 4 a more in-depth comparison of the three methods during training is made in terms of objective function, performance and output values. Note that STD takes considerably less iterations. For this particular class (digit 1) the maximum training data classification accuracy is reached after approximately 100 iterations. Both other methods take about ten times as many. After these 100 iterations STD typically starts over fitting. The objective function for STD therefore does not always reflect its performance. The steepness in the objective function graph, as well as the output value decreases only marginally. The other methods, L2 especially, clearly approach a plateau, better reflecting their respective accuracy on the training data. Given more iterations these methods also prove less prone to over fitting than STD.

5.2 Time Complexity

Training for STD as described in section 3.5 is of time complexity $O(N^2W)$. The alternative methods are of $O(NW)$, with N being the number of training examples and W the number of visual words. The amount of visual words can be expected to be far less than the amount of training examples, as visual words are meant to represent generalizations over the data. The alternative methods are therefore significantly faster in training than STD.

6 Considerations

Though comparing classification accuracy of this setup to other, more established methods may not be the primary subject of this thesis, considering aspects of this system that may favour one classifier above the other most certainly is. Most of these aspects have been carefully monitored during the experiment described above. Following are some critical considerations meant to illustrate the extent to which the previously presented conclusions may be appreciated.

6.1 Number of Centroids

Considering figure 2 in section 4 one may conclude that the number of centroids makes for a positive contribution in classification accuracy for the alternative methods, but not for the traditional approach. This apparent effect shows for STD-500, but not for STD-100. This is probably due to overfitting, as described in section 5.1, and is therefore not as obvious for STD with 100 iterations, effectively as a result of early stopping.
Groefsema found that an increase in centroids does not entail a necessary increase in performance for the alternative methods either. He describes a decrease in performance in section 5 of (Groefsema, 2016) when increasing the number of centroids from 400 to 800. His observation however is made using a different dataset, CIFAR-10 (Krizhevsky and Hinton, 2009), aimed at object classification rather than handwritten character recognition.

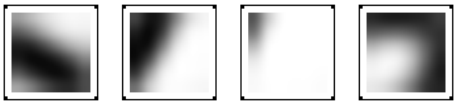


Figure 5: Most contributing centroids for each quadrant (TL, TR, BL, BR) for class 0, after training L2

As feature extraction in this system is done independent of classifier training, all methods have access to the same information (see fig. 5 for an impression), yet it is used quite differently. The traditional approach (STD) calculates a weighted sum of the inner product of two data points, while the alternative methods calculate a plain weighted sum of the input dimensions, with both methods expressing data points in terms of Euclidean distance to

the different centroids. This difference however is by design.

6.2 Amount of Data

While the MNIST dataset consists of 60.000 training images and an additional 10.000 images as a suggested test set, only 10.000 images were used for training in the previously described experiment. Here computation time was the primal consideration, in which STD was the main contributing factor in temporal sense.
Given the relative nature of the conclusions however, this is of no concern. All methods acted on the same amount of data and no indication of crucial relative performance increase given additional training data was observed.
For results for this set-up using all of MNIST, see (Groefsema, 2016).

6.3 Number of Training Rounds

As mentioned in sections 5.1 and 6.1, STD is prone to over-fit unless training is cut off after a certain amount of iterations. The alternative methods do not seem to suffer from this effect as strongly.

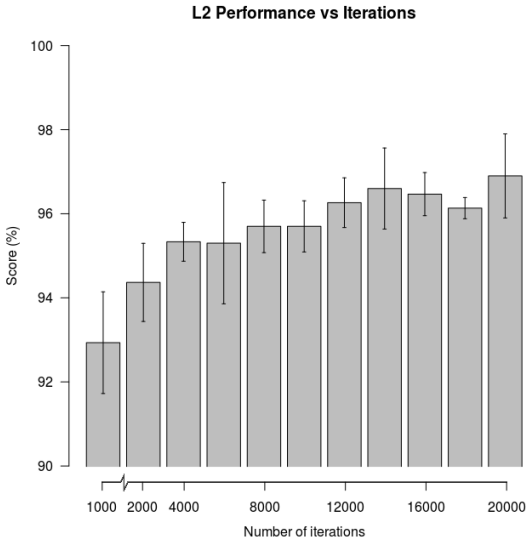


Figure 6: L2 performance given more iterations

As can be seen in Figure 6, the final score does vary with the number of training iterations, but if a general trend is to be observed from this graph, it would be the approach of an unstable limit, not a sudden collapse in precision as one would expect in case of strong over-fitting.

6.4 Feature Descriptor

At first sight this set-up uses image data without any further abstractions; concatenations of pixel intensity values were directly compared to centroids, which in turn are only averaged examples of often encountered subsets of these pixel intensity value sets. In (Maas, 2016) Maas describes his findings using this set up in combination with different feature descriptors, used instead of the raw pixel values used in this thesis.
In closer examination though, one may conclude that the role of the BOW model in this set-up is comparable to that of a feature descriptor. After all, in training and classification, images are expressed as a combination of hyper-distances to the centroids.

6.5 SVM Theory

STD is a traditional SVM only if we define the data upon which the classifier acts as the result of $F(x)$. $F(\cdot)$ is then considered part of the preprocessing step. The classification function $f(\cdot)$ for the alternative methods L1 and L2 can then be interpreted as a single layer network, as the output is only a weighted sum of all the input dimensions, combined with an offset. Cortes and Vapnik however, originally described the SVM as being comparable to a two layer network (Cortes and Vapnik, 1995). The first layer being the dot products between the input and the support vectors, and the second layer being the weighted sum of those dot products.
We may however also consider $F(\cdot)$ part of a complex kernel function, but, seeing as to how in this set up the final dimensionality of $F(x)$ is equal to four times the number of visual words resulting in $F(x) \in \Re^{200-800}$, while using only the raw pixel values would result in $F(x) \in \Re^{1296}$, this would seem to be a function that decreases dimensionality. While Cortes and Vapnik proposed mapping the input to a high dimensional feature space to increase linear separability. One must take into account however that the proposed method applies a patch-based approach. $F(x)$ is constructed as a

concatenated per-quadrant summary of patch-word distances, taking the mean patch-word distance for that patch into account as well (equation 3.1). The dimensionality may be reduced, but the amount of information that each dimension represents is far more than that of a single pixel value. So one might argue that the maximum dimensionality as used in this setup is still less than the original dimensionality, but the situation is slightly more complex than that.
The approach to SVM theory in this thesis may be controversial, but the core principles are still there. The input data are mapped to a high dimensional feature space, input is expressed in terms of similarity to other data points, the sign of the weighted sum of those components determines the outcome, and the original primary objective function indicates the direction and magnitude of change for the parameters.
But ultimately, it is up to the reader to conclude whether or not to classify this approach as a support vector machine.

References

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5): 11551178, 2007.

A. Coates, H. Lee, and A.Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011. URL `http://jmlr.csail.mit.edu/proceedings/papers/v15/coates11a.html`.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 1573-0565. doi: 10.1023/A:1022627411411. URL `http://dx.doi.org/10.1023/A:1022627411411`.

G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

M. Groefsema. Deep architectures using the bag of words model for object and handwritten character recognition. *Bachelor's thesis, Rijksuniversiteit Groningen*, 2016.

R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6): 1153–1160, Dec 1981. ISSN 0096-3518. doi: 10.1109/TASSP.1981.1163711.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. URL `https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

J.L. Maas. The dual codebook: Combining bags of visual words in image classification. *Bachelor's thesis, Rijksuniversiteit Groningen*, 2016.

Singer Y. Srebro N. Shalev-Shwartz, S. Pegasos: Primal estimated sub-gradient solver for SVM. *In Proceedings of the International Conference on Machine Learning 2007*, pages 807–814, 2007.

Naum Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, 1985. ISBN 0-387-12763-1.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *In Proceedings of the International Conference on Machine Learning 2004*, pages 919–926, 2004.

Lei Zhu, Al Bing Rao, and Aldong Zhang. Theory of keyblock-based image retrieval. *ACM Trans. Inf. Syst.*, 20(2):224–257, April 2002. ISSN 1046-8188. doi: 10.1145/506309.506313. URL `http://doi.acm.org/10.1145/506309.506313`.