Mathematical Aspects
of Machine Learning

Report

Digit Recognition with Support Vector Machines

Authors:
Lisa Gaedke-Merzhäuser
Paul Korsmeier
Lisa Mattrisch
Vanessa Schreck

Problem Statement

Our main goal is to correctly identify handwritten digits based on the MNIST ("Modified National Institute of Standards and Technology") data set. This data set consists of 42000 gray-scale images. Each image is 28 pixels in height and 28 pixels in width. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel (kag).

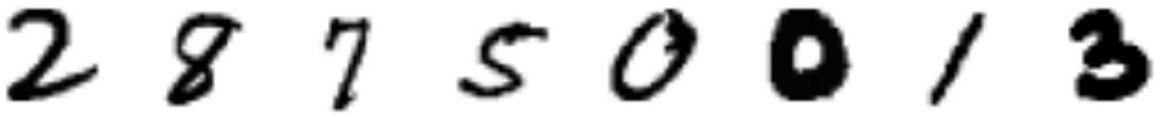

Figure 1: Visualization of eight of these images

Support Vector Machine

Given data points $\{x_i\}_{i=1}^l \subset \mathbb{R}^d$ with respective labels $\{y_i\}_{i=1}^l \subset \{\pm 1\}$ we want to seperate differently labelled points using a decision function of the form

$$f(x) = \operatorname{sgn}\left(w^T \Phi(x) + b\right)$$

where Φ is the feature map as introduced in the lecture.

The Support Vector Machine (SVM) is a binary classifier that aims to find the maximum margin hyperplane in the feature space, i.e. the goal is to maximize the margin while softly penalizing points that lie on the wrong side of the boundary or inside the margin (Bis06). Dualizing this optimization problem, we obtain the following equivalent quadratic program:

$$\underset{0 \le \alpha \le C}{\text{maximize}} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

with the dual variable α, kernel function k and penalty C. After finding the optimal solution α^* of this program, we can formulate the resulting decision function as

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i k(x_i, x_j) + b\right).$$

Note that only data points x_i with $\alpha_i \neq 0$ appear in the decision function. Those points are called *support vectors*. By analysing primal, dual and the corresponding KKT conditions, we find that any x_i with $\alpha_i = C$ was correctly classified and lies exactly on the margin boundary, and any x_i with $0 < \alpha_i < C$ lies either inside the margin or on the wrong side of the margin. Additionally, any data points x_i on the (correct) margin boundary must satisfy $y_i \cdot f(x_i) = 1$. We can use this property to find the parameter b (Bis06).

Solving the optimization problem with SMO

Multi Class Classification Problem

Support Vector Machines are binary classifiers, meaning that they only have two possible output labels. Often one chooses them to be 1 and -1. Our problem however was to classify unknown data into one out of 10 different categories. Since there is no direct way to change one support vector machine so that it can choose from more than two labels we had to find a way to combine several SVMs, who together would be able to do so. There are a number of well known strategies tackling this issue. Let us first look at a very intuitive but also primitive approach called One-vs-All classification. Using this method one trains as many SVMs as one has classes, so 10 in our case, and sets labels such that the i-th SVM has all training data with label i set to one and everything else to -1. When trying to predict a new label one simply looks for a classifier that gave this data point the label one. Unfortunately this approach has many issues. First of all new data points are generally not uniquely classified. Therefore it is necessary to add some sort of confidence score which label is considered to be most likely out of the possible ones. Finding good ways of assigning reliable confidence scores is often not an easy task. We decided to resolve this issue by computing the barycenters of each class and when ambguities arose chose the label of the closest barycenter. This simple strategy notably improved our results and we decided on not devoting any more time on a different strategy because the One-vs-All classification has another major drawback. The number of training points for each classifier being assigned positive and negative labels is very unbalanced, in our case on average 1 to 9. Therefore one is likely to get shifted margins to what one would normally perceive to be correct. The penalty term will lead each classifier to give less importance to the positives than the negatives simply because they are less in numbers and can be outweighed by the others.

Bild hier oder zu unwichtig?

Another common intuitive approach would be the One-vs-One classification. Here during training one only ever considers the data corresponding to two classes and sets one label to 1 and the other one to -1 for every possible pair. In our case this would mean training 45 different SVMs, although data sets for each classifier are much smaller (the number arises from the binomial coefficient of 10 choose 2... or $n * (n-1)/2$).

We only tried the first approach since as just mentioned the compute time for the second one was simply too high for us considering that we have a very large amount of training data. In the One-vs-All results were very very disappointing. In the case of a linear as well as a Gaussian kernel ... In the linear case this can be explained by the fact that our training data was simply not linearly seperable. Hence we added in an additional feature. We computed the barycenters of the data points of each class. When a data point could not be uniquely classified we would give it the label of the barycenter it was closest to. This method led to a major improvement of our results. Our algorithm now labeled

about 60% of our data correctly (... see appendix). Nevertheless the result was still not satisfactory.

So we decided to focus our attention on a different approach: Error Correcting Output Codes.
subheading?

They are easiest to explain considering a concrete example. We trained 15 different classifiers f_i.

Table 1: Error Correcting Output Codes

Class	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	-1	-1	-1	-1	1	-1	1	-1	-1	1	1	-1	1
1	-1	-1	1	1	1	1	-1	1	-1	1	1	-1	-1	1	-1
2	1	-1	-1	1	-1	-1	-1	1	1	1	1	-1	1	-1	1
3	-1	-1	1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	1
4	1	1	1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1
5	-1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1
6	1	-1	1	1	1	-1	-1	-1	-1	1	-1	1	-1	-1	1
7	-1	-1	-1	1	1	1	1	-1	1	-1	1	1	-1	-1	1
8	1	1	-1	1	-1	1	1	-1	-1	1	-1	-1	-1	1	1
9	-1	1	1	1	-1	-1	-1	-1	1	-1	1	-1	-1	1	1

The rows of the table show what class is assigned what label depending on each classifier. For example f_0 assigns 1's to all even numbers and -1's to all odd numbers. This way each class has a string of 1's and -1's it corresponds to which is also called its codeword. The codewords are chosen such that their Hamming distance (i.e. the number of entries where they differ) is maximized. In our case each codeword has a Hamming distance of at least six to any of the others strings. Now when one wants to label an unknown data point each of the classifiers assigns it a label and one gets an output string of 15 digits. We classify the data point according the codeword it has the least Hamming distance to. The name error correcting output codes is derived from the fact that we can for example three classifiers can misclassify a data point and we will still get the correct result. Depending on the difficulty of the problem one could choose more or less classifiers and hece in-or decrease the Hamming distance of the set of codewords. We took these codewords from ... and yielded much better results than with the approaches mentioned above. Our precise implementation can be found on page... in the appendix.

Results & Conclusions

When choosing ... many training points and .... many testing points our algorithm was correct in ... of the cases. Include things here...?

Linear and Gaussian.

In general ECOC?

Where do we bring in cross validation?

General Conclusions? Difficulties?

Bibliography

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning* -. 1st ed. 2006. Corr. 2nd printing 2011. Berlin, Heidelberg : Springer, 2006. – ISBN 978–0–387–31073–2

[kag] `https://www.kaggle.com/c/digit-recognizer/data`