Single-character OCR using Support Vector Machines

Olli Jarva & Jarno Rantanen

Aalto University School of Science

olli@jarva.fi & jarno@jrw.fi

Abstract

This paper describes a solution to an optical character recognition problem for bitmap characters using Support Vector Machines with an RBF kernel, including a description of RBF parameter search and bitmap normalization. Classification performance of 90.8% was achieved against a given training set of 40000 correctly labelled samples [**?**].

KEYWORDS: SVM, Support Vector Machine, RBF, OCR, Character Recognition

1 Data set description

The data set against which our solution was developed consisted of a provided set of 42152 black-and-white bitmaps, depicting hand-written instances of characters from the English alphabet (that is, the task was to classify the bitmaps into 26 distinct categories). Each bitmap was given as a 16-by-8 image, in the form of a binary vector of length 128 (meaning an array of 128 ones or zeroes). The vectors were delivered in a text file, with the correct label associated with each vector.

As an optional extra task, we were provided with the opportunity of participating in a competition amongst different solutions to the same classification problem. The competition was organized by first providing only a subset of the training data (10000 vectors), using that to train a classifier, and then calculating an error rate against the rest of the training data (which was not yet made available at that time). Our participation in the competition is discussed further in Section 6.

The same data set was used for both training our classifiers, and testing them afterwards. The data was split using k-fold cross-validation to minimize overfitting, while making the most of the available data. This testing technique is discussed in further detail in Section 5.

2 Method selection/Why SVM?

- What other options were there

- Why we chose SVM?
- A nod to why we think we chose correctly

3 What is SVM?

Support Vector Machines (SVM's) are a method for supervised machine learning, meaning they solve a classification problem by creating a classifier function from a set of existing, labeled data points. This function can then be used to classify (or *label*) subsequent data points *without* supervision. In contrast to *regression methods* which produce continuous output, the output of a classifier function is always exactly one class into which the input data point belongs.

In its most basic implementation, an SVM is trained with data labeled into exactly two separate groups. The resulting classifier is a *binary one*, meaning it will classify subsequent input into those same two categories. If the input data points belong to a two-dimensional space, this can be intuitively thought of separating the sets of points belonging to each category by as wide a band as possible (or simply a line, with as much space between the line and the nearest data points as possible).

SVM's generalize nicely into higher dimensional spaces. In an n-dimensional input space, the two classes are linearly separated by an (n-1)-dimensional hyperplane instead of a line. They also generalize into working with >2 classes by way of reducing the multi-class classification problem into a set of binary classification problems, for example by chaining the binary classifiers so that the first classifies the input data as either belonging to class "A" or "other", the second one to "B" or other, and so on.

TODO: Generalization into non-linear input spaces

4 Character preprocessing

- Minimize noise by moving characters to bottom left corner. 0.5% improvement

5 RBF kernel parameter search

γ and C

- Initial search space 2**x for x in range(-15, 15)
- Select best area for next round
- Validate by taking final arguments and calculating error rates for +- few percent for both variables.

6 Results and performance

- k-fold cross validation: k=20, error rate 11%
- k-fold cross validation: k=5, error rate 11.5%
- One iteration with training set n=40000 and validation set n=2152 about 17 min with 2.1GHz Xeon (single thread)
- about 300MB of memory for training set n=42152
- Predicting one character: about 2 milliseconds

7 Quick comparison to other algorithms

- kNN (+PCA/LDA)
- ...?

[1]

References

[1] D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman, and C. Furlanello. mlpy: Machine learning python, 2012.