Pattern Classification and Machine Learning

Miniproject

Taha Elgraini (SCIPER №...), Fabian Brix (SCIPER №236334)

December 14, 2013

1 Introduction

The imperative of this project was to apply two different machine learning techniques, the Multilayer Perceptron (henceforth referred to as "MLP") and the Support Vector Machine (henceforth "SVM") to the recognition of hand-written digits. More specifically error back-propagation gradient descent optimization had to be implemented for MLP and the Sequential Minimal Optimization (SMO) algorithm had to be implemented to solve the problem with an SVM. Focus lay on a good understanding and clean implementation of the aforementioned algorithms as well as a sound evaluation of the performance of the implementation on problems posed with excerpts? the MNIST dataset.

2 Multilayer Perceptron

2.1 Methods

This section covers how we went about implementing the MLP back-progagation gradient descent algorithm and which parameters we chose to make it run optimally.

2.1.1 Treatment of Data

First of all, we had to bring the data into a format where we could hand it over to our algorithm in a way that made training with early stopping possible.

Splitting The data was already available as separate training and test data. So, all what was left to do was to split the training data into a random $\frac{2}{3}$ training and $\frac{1}{3}$ validation set. This was easily achieved by computing a random permutation of the indices of the available data patterns and then dividing the dataset and the labelset at the $\frac{2}{3}$ mark according to the permutation.

Preprocessing Preprocessing of the created training and validation as well as the test set was further achieved by computing the min. and max. features of the untouched training data beforehand and then applying the normalization as given the project instructions to the training and the test data. So, the actual preprocessing was done before splitting the data in order to keep the operation concise and not unnecessarily complicate it.

2.1.2 MLP setup

The structure of the MLP we were advised to use through the project instructions is depicted in Fig. **??**. The differences to the two-layer MLP discussed in the lecture lie in the following properties: Firstly, the first hidden layer has double the outputs of a "normal" MLP perceptron. Secondly, the gating function $g(a_{2k+1}, a_{2k})$ takes pairs of values from the hidden layer accordingly and last but not least the last activation value is not passed through another gating function.

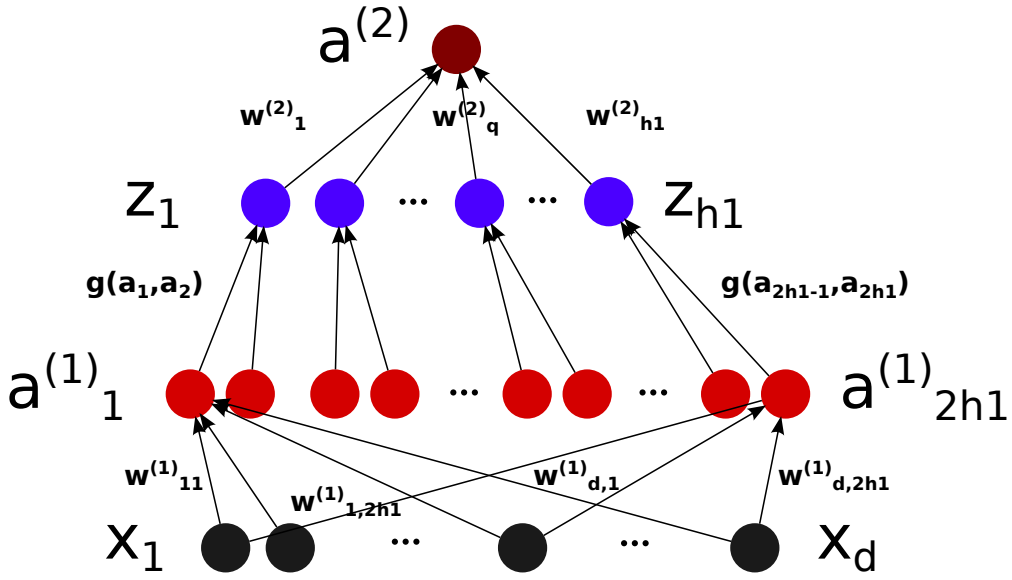

Figure 1: Schema of the MLP as described in the project instructions

Early stopping Evaluate empirical error $\hat{R}(\hat{f}; \mathcal{D}_V)$ for validation set $\mathcal{D}_V$ along-side the training-error $\hat{R}(\hat{f}; \mathcal{D}_T)$while training the MLP on $\mathcal{D}_T$. Stop at the training at the epoch where the validation error stops decreasing and starts increasing.
Justify design and parameter choices
Show comparative plots illustrating the effects of learning rate, number of hidden units, momentum, etc. on

Effects of parameters on convergence speed CLASSIFIERS NOT TESTED ON TEST SET, ONLY ON VALIDATION SET

- effects of the learning rate η
- Effects of number hidden units h_1
- Effects of the momentum term

2.1.3 Effects of parameter choice on overfitting

show for which parameters training error decreases while validation increases

include TYPICAL example of overfitting by letting the validation error increase while the training error decreases for a few, say 3, rounds!

2.1.4 Determining parameters for binary subtasks

COMMENT ON FINDINGS **qualitatively**

3 Results

plots annotated by the result of the final classifier on the test set

(a) example subcaption

(b) example subcaption

(c) example subcaption

(d) example subcaption

Figure 2: Example caption

(a) example subcaption

(b) example subcaption

Figure 3: Example caption

3.1 Discussion of performance on test set

3.2 Misclassified patterns

4 Support Vector Machine

4.1 Methods

4.1.1 Treatment of Data

Splitting For the implementation of the SMO algorithm the training data had not to be split into fixed training and validation sets. Instead the imperative was to implement 10-fold crossvalidation for parameter selection of tao and C. In order to achieve this task and save time we used the KFold() method of the python

(a) example subcaption

(b) example subcaption

Figure 4: Example caption

(a) example subcaption

(b) example subcaption

Figure 5: Example caption

scikit library. This method thus splits the dataset into 10 consecutive folds without shuffling We split the data into 10 bins and talk one of them alternatingly as a validation set. Split dataset into k consecutive folds (without shuffling).

Each fold is then used a validation set once while the k - 1 remaining fold form the training set.

Preprocessing Regarding preprocessing we used the same normalization as with the Multilayer Perceptron.

4.1.2 SVM setup

include all implementation details including choice of C and of *tao*
evaluate 10-fold CV scores for all combinations of the matrix

Plot the SVM criterion as function of SMO iterations (only every 20 SMO steps). Additionally, plot the value of the convergence criterion, based on KKT condition violations (see additional note). For the latter plot, the vertical axis should have a logarithmic scale.

4.2 Results

(a) example subcaption

(b) example subcaption

(c) example subcaption

(d) example subcaption

Figure 6: Example caption