sufficient, no bonus

Neural Networks
Practical Assignment 3: Learning by gradient descent

Rick van Veen (s1883933) Laura Baakman (s1869140)

January 13, 2015

I. Introduction

In this report we consider a multilayer feed forward neural network, which we will train using the stochastic gradient descent algorithm. Gradient descent (GD) is an optimization algorithm which tries to find the local minimum of a function, by taking steps proportional to the negative of the gradient at a certain point. In the case of a feed forward network gradient descent is applied to minimize the cost function (see (2) and section II for the explanation of this function).

The stochastic gradient descent (SGD) algorithm in contrast to the GD algorithm does not use the gradient of the cost function, but the gradient of the contribution (see (3)) of a randomly sampled data point. One advantage of SGD, as opposed to gradient descent is that it is faster on very large datasets. Gradient descent goes through the whole training set to perform an update whereas stochastic gradient descent updates the weight vectors iteratively. Each iteration it randomly selects one pattern to be used for updating the weights. One of the advantages to this it that performance of stochastic gradient descent increases each iteration [1].

II. Method

We consider a soft-committee machine with weight vectors, $\{\vec{w}^1, \vec{w}^2\} \in \mathbb{R}^N$, between the N input units and the two hidden units. The soft machine is a two-layer fully connected network consisting of non-linear hidden units and a linear output unit. The weight values between the hidden units and the output unit are fixed to $+1$. The real-valued output of the network given a pattern $\vec{\xi} \in \mathbb{R}^N$ is defined as:

$$\sigma(\vec{\xi}) = \left(\tanh\left[\vec{w}^1 \cdot \vec{\xi}\right] + \tanh\left[\vec{w}^2 \cdot \vec{\xi}\right]\right) \quad (1)$$

We train the network on a data set with P N-dimensional patterns with real valued labels: $\mathcal{D} = \left\{\vec{\xi}^\mu, \tau\left(\vec{\xi}^\mu\right)\right\}_{\mu=1}^{P}$. The goal of gradient descent training is to minimize the quadratic cost function E.

$$E = \frac{1}{P}\sum_{\mu=1}^{P} e^\mu \quad (2)$$

Where e^μ is the contribution of one of the P patterns ξ^μ to the cost function (2):

$$e^\mu = \frac{1}{2}\left(\sigma\left(\vec{\xi}^\mu\right) - \tau\left(\vec{\xi}^\mu\right)\right)^2 \quad (3)$$

This term is used to update the weight vectors according to:

$$\vec{w}^m \leftarrow \vec{w}^m - \eta\nabla\vec{w}^m e^\mu \qquad m \in \{1, 2\} \quad (4)$$

Where η is the learning rate and $\nabla\vec{w}^m e^\mu$ is the gradient of the contribution of one pattern to the cost function (3) with respect to the weight vector $\vec{w}^\mu$.

To determine $\nabla\vec{w}^m e^\mu$ we first determine the derivative of (3) with respect to w_n^m, the nth element of the mth weight vector:

$$\begin{aligned}\frac{\partial e}{\partial w_n^m} &= (\sigma(\vec{\xi}) - \tau(\vec{\xi}))\frac{\partial \sigma}{\partial w_n^m} \\ &= (\sigma(\vec{\xi}) - \tau(\vec{\xi}))(1 - \tanh^2(\vec{w^m} \cdot \vec{\xi}))\xi_n\end{aligned} \quad (5)$$

Using the derivation presented in (5) we

find for $m \in \{1, 2\}$:

$$\nabla \vec{w}^m = \left(\sigma\left(\vec{\xi}\right) - \tau\left(\vec{\xi}\right)\right)\left(1 - \tanh^2\left(\vec{w^m} \cdot \vec{\xi}\right)\right)\vec{\xi} \tag{6}$$

Note the absence of subscripted n in the last factor of (6) when compared with (5).

We have applied (4) in algorithm 1. Although the weight vectors are initialized randomly by `init()` they have to satisfy the following requirement: $\|\vec{w}\|^2 = 1$. This avoids that the weight vectors are initialized too large. Each time step i one pattern is randomly selected to be used by `get_random_pattern`, as is required for stochastic gradient descent. This algorithm executes $t_{max} \cdot P$ updates on the weight vectors.

Algorithm 1: $SGD(\mathcal{D}, t_{max}, \eta)$

input : $\mathcal{D} = \left\{\vec{\xi}^\mu, \tau\left(\vec{\xi}^\mu\right)\right\}_{\mu=1}^{P}$
t_{max} maximum number of epochs
η the learning rate
output: $\vec{w}^1$, $\vec{w}^2$ the weights

```
[w¹, w²] := init(D)        /* ||w||² = 1 */
for t ← 1 to t_max do
    for i ← 1 to P do
        μ := get_random_pattern()
        [w¹, w²] := update(μ)    /* (4) */
```

III. Experiment

To measure the performance of the SGD algorithm, we split the available data into two data sets. The first set we use to train the algorithm and the other to test its performance.

To measure the performance while learning, the learning rate of the SGD algorithm can be calculated using the error (the cost function from (2)) on the training data set. The performance of the algorithm on unseen data, the generalization error, can be measured using the same error function on the test data set. Both errors are computed after every epoch t and thus after every P training steps. The two curves are shown in fig. 1.

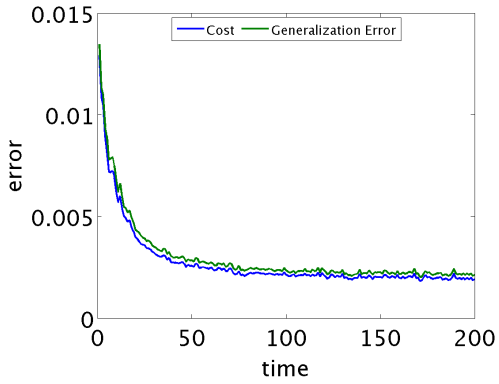

Figure 1: *The cost, shown in blue, and the generalization error, shown in green, of a training set and test set with each 2000, 50-dimensional points for each epoch, with $t_{max} = 200$, $\eta = 0.05$.*

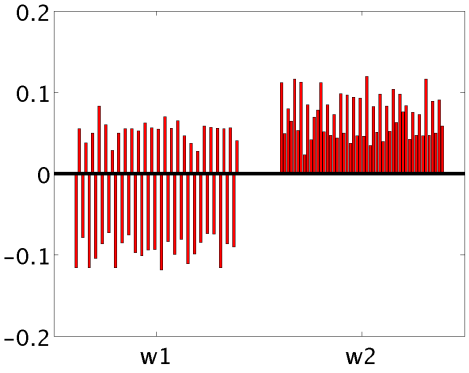

Figure 2: *The values of the elements of the two weights vectors at $t = 200$.*

One would expect that the error on the training data would be lower than on the test data, because the network is trained specifically on the data from that training data set. One can observe this difference between the two errors in fig. 1.

The resulting weights, at $t = 200$, are plotted in fig. 2. It seems as tough $|\vec{w}^1| = \vec{2}$, however comparing the actual values of the weight vectors we found that the differences between the elements of the two vectors lie in the range $[-0.05, 0.05]$. It is strange that $\vec{w}^2$ contains only positive weights, however executing the algorithm multiple times on test and training samples from the same data set showed that this is not always the case.

References

[1] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.