Assignment III

Andrei Secuiu, Roberto Schinina'

Introduction

This assignment aims to give an overview of gradient descent-based training of layered networks. In particular, we concentrate our attention on a shallow feedforward neural network with one hidden layer. The hidden layer, whose weights are denoted by w, is trained by stochastic gradient descent. The hidden-to-output weights v are set to 1 in the interest of simplicity. This type of structure is also called a soft committee machine. This assignment aims at discussing the following points:

1. providing an implementation of a stochastic gradient descent learning rule for a network with $K = 2$ nodes in the hidden layer and a sigmoidal activation function
2. investigating the evolution of the training and generalization errors as a function of the training iterations t
3. analyzing the evolution of the training and generalization errors as a function of the number of training data points used in learning the network

Method

This section introduces the relevant theory and algorithms necessary for implementing stochastic gradient descent learning for the soft committee machine and the experimental setup.

Cost function and training algorithm

Assume we are given a data set $\mathbb{D} = \{\xi^\mu, \tau(\xi^\mu)\}_{\mu=1}^P$ where $\xi \in \mathbb{R}^N$ and $\tau(\xi^\mu) \in \mathbb{R}$ are continuous training labels. Let $g : \mathbf{R} \to \mathbf{R}$ be the activation function of the hidden layer, which we assume to be differentiable with derivative g'. The output of the network is given by:

$$\sigma(\xi) = \sum_{i=1}^{K} v_i \; g(\xi \cdot w_i)$$

where w_i's are the N-dimensional weight vectors that connect the input layer with the hidden layer while $v_i's$ are the second layer wights. A quote from the lectures states that *there is nothing objective about the objective function*, implying that the choice of the function which penalizes deviations from the data labels is up to the researcher. That being said, it is common to rely on the quadratic loss cost function because of its interpretability and mathematical convenience in optimization.

$$E = \frac{1}{P} \sum_{\mu=1}^{P} \frac{1}{2} \left(\sigma(\xi^\mu) - \tau(\xi^\mu)\right)^2 = \frac{1}{P} \sum_{\mu}^{P} e^\mu \tag{1}$$

This function is to be optimized in the weights w_i using an algorithm called Stochastic Gradient Descent (SGD). In particular, the special case of SGD used in this assignment is called incremental SGD, where, at each learning step, we select one of the P examples ν at random with probability $\frac{1}{P}$ and then use *only* the gradient with respect to the contribution e^ν. The gradients at ξ^ν are computed in the following manner:

$$\nabla_{w_i} e^\nu = \frac{\partial}{\partial w_i} \frac{1}{2} \left(\sigma(\xi^\nu) - \tau(\xi^\nu)\right)^2 = \left(\sigma(\xi^\nu) - \tau(\xi^\nu)\right) \cdot \nabla_{w_i} \sigma(\xi^\nu)$$
$$\nabla_{w_i} \sigma(\xi^\nu) = \sum_{j=1}^{K} v_j \nabla_{w_i} g(\xi^\nu \cdot w_j) = \sum_{j=1}^{K} v_j g'(\xi^\nu \cdot w_j) \cdot \xi^\nu \delta_{ij} = v_i g'(\xi^\nu \cdot w_i) \xi^\nu$$

In the interest of computational efficiency, it is useful to rewrite the above gradients as vectorized operations. If $W = [w_1 \ w_2 \ \dots \ w_K] \in \mathbf{R}^{N \times K}$ is the weight matrix (i.e. the matrix whose columns are the vectors of weights w_i) and if we include the assumption $v_i = 1$, we obtain:

$$\begin{aligned}\nabla_W \sigma(\xi^\nu) &= [\nabla_{w_1}\sigma(\xi^\nu) \ \nabla_{w_2}\sigma(\xi^\nu) \ \dots\ ] = [\ \dots \ g'(\xi^\nu \cdot w_i)\xi^\nu \ \dots\ ] = \xi^\nu \cdot [\ \dots \ g'(\xi^{\nu,T} w_i) \ \dots\ ] \\ &= \xi^\nu g'(\xi^{\nu,T}[\ \dots w_i \ \dots\ ]) = \xi^\nu g'(\xi^{\nu,T} W) \end{aligned} \tag{2}$$

where in the last equality the operations are matrix multiplications, $\xi^{\nu,T}$ is the transpose of ξ^ν and g' is applied element-wise to the elements of the matrix $\xi^{\nu,T} W$. Therefore, the update rule of the weight vectors for a SGD iteration with data point ξ^ν is:

$$W \leftarrow W - \eta \ (\sigma(\xi^\nu) - \tau(\xi^\nu)) \ \xi^\nu g'(\xi^{\nu,T} W) \tag{3}$$

In particular, for this assignment, we restrict our investigations to the activation $g(x) = \tanh(x)$, $g'(x) = 1 - \tanh^2(x)$ and $K = 2$.

Algorithm. The SGD routine used in this assignment is shown in Algorithm 1. Its inputs have the following meaning:

- P is the dimension of the training set.
- Q is the dimension of the test set.
- M is the dimension of the whole dataset (again, $P + Q \leq M$).
- $t_{\max}$ is the maximum training time in units of P.
- η is the learning rate.

Generalization error. When investigating the adequacy of a model, it is important to estimate its performance on novel data. The cost function evaluated on data points used in optimizing the network (a.k.a. the training error) is not a good metric because the model is designed to reproduce the training data labels. This is the reason why a model's quality is judged on novel data, not used in its training at all. The cost function evaluated on it is called the generalization error.

To that end, let us consider a data (sub)set with $P+Q \leq M$ data points, $\mathbb{D}\{\xi^\mu, \tau(\xi^\mu)\}_{\mu=1}^{P+Q}$. We partition it into a training set with P points and a test set with Q points. The weights W are learned on the training set, while the generalization error is computed on the test set:

$$E_{test} = \frac{1}{Q} \sum_{\rho=P+1}^{P+Q} \frac{1}{2} \left(\sigma(\xi^\rho) - \tau(\xi^\rho)\right)^2 \tag{4}$$

Algorithm 1 Incremental SGD.

```
Require: $\{\xi^\mu\}_{\mu=1}^M$, $\{\tau(\xi^\mu)\}_{\mu=1}^M$ , $P$, $Q$, $\eta$, $t_{max}$
  $E \leftarrow 0^{t_{max}+1}$
  $E_{test} \leftarrow 0^{t_{max}+1}$
  Generate $w_1, w_2 \sim \mathcal{N}(0, I)$ with $||w_1||^2 = ||w_2||^2 = 1$. Initialize $W = [w_1\ w_2]$
  Calculate $E[0]$ and $E_{test}[0]$ according to equations 1, 4.
  for $i \in \{0, \ldots, t_{max} \cdot P\}$ do
      Select uniformly at random one of the training data points $\nu$
      Compute the gradients $\nabla_W e^\nu$
      Update the weights according to equation 3
      if $(i+1)\mathrm{mod}P = 0$ then                    ▷ Save the cost function every $P$ iterations
          Set $E[(i+1)//P]$ equal to the results obtained using equation 1 for the updated weights.
          Set $E_{test}[(i+1)//P]$ equal to the results obtained using equation 4 for the updated weights.
      end if
  end for
  return $W, E, E_{test}$
```

Experimental setup

Implementation. Our code has been written in a Python Notebook, which was run on the cloud platform Google Colab. The following libraries have been imported: `pandas`, `numpy` and `matplotlib.pyplot`. The code is designed to rely on vectorized operations that are efficiently computed by `numpy`; any "implementational tricks" are those common to `numpy` array manipulations. The code further allows the user the flexibility to change K, the activation g and the weights v, though the flexibility is not exploited in this assignment.

Analysis strategy. Our analysis separates the core problem from the bonus.

- *Core problem*
  - Train the network using the first P data points, and obtain the generalization (test) error using the next Q data points
  - Display the final weights W and the cost function on both the training and the test sets
- *Robustness analysis.* Visualize the variability of the cost function curves $E(t)$ that arises from two sources of stochasticity: (1) SGD and (2) the choice of data points in the larger data set
  1. Repeat the core problem analysis for $R = 10$ times. Display the average and the 95% confidence interval of the R curves $E(t)$
  2. Select randomly $R = 10$ data sets with P (training) and Q (testing) data points. Learn each data set *once* via SGD. Display the average and the 95% confidence interval of the R curves $E(t)$
- *Bonus*
  - For each different P, select the first P points for the training set and the next Q points for the test set
  - Train the network via SGD $R = 10$ times. Obtain the curves $E(t)$
  - Visualize $E(t)$ on the training and on the test set in a manner similar to the robustness analysis. Overlap the curves originating from different P

Parameters. In our study, we have set the following:

- The dataset of size 50×5000 given in the assignment. That implies $N = 50$ and $M = 5000$.
- $P = 200$ for the core assignment and $P \in \{20, 100, 500, 2500\}$ for the bonus
- $Q = 100$, $t_{max} = 500$, $\eta = 0.05$

Results

Figure 1 shows the result for the core problem. Figures 3, 4 correspond to the robustness analysis, while Figure 5 answers the bonus.

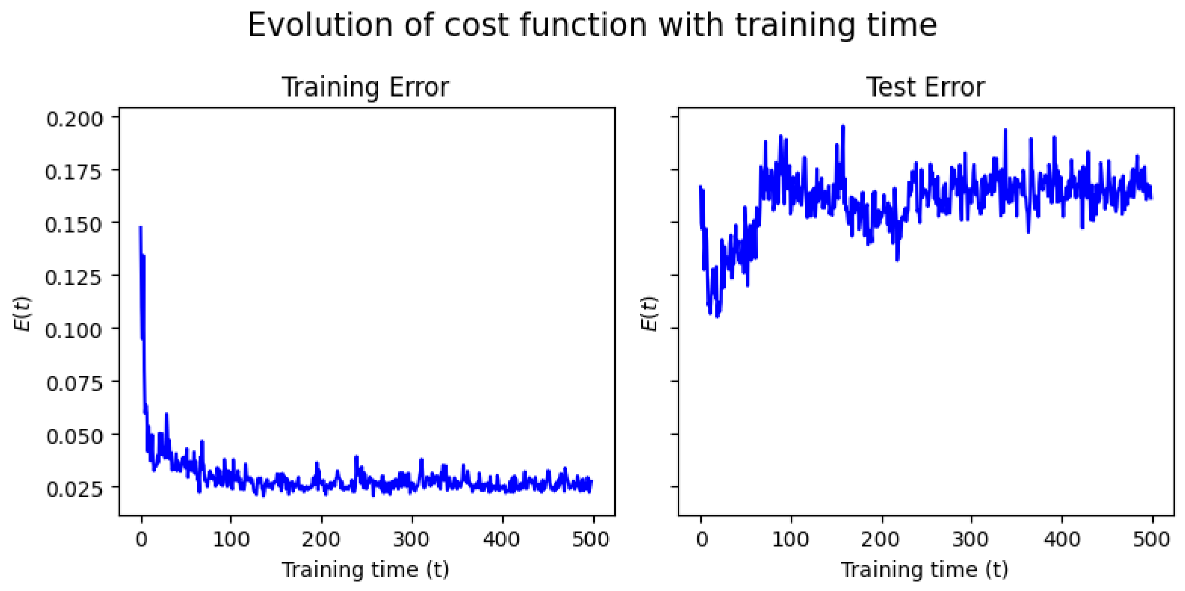

Figure 1: The cost function as a function of the training iteration t (units of P)

Discussion

Final weight vectors. Using the first $P = 200$ data points, the output weight vectors w_i after one SGD routine are shown in Figure 2. Qualitatively, we remark that w_1, w_2 seem to be (1) anti-correlated and (2) have a vaguely sinusoidal shape. These may hint at the generation mechanism of the artificial data set.

Robustness analysis. The training curves (i.e. the cost function $E(t)$ on the training set) always show a small variance compared to the test curves (i.e. the cost function $E(t)$ on the test set). In the main case studied ($P = 200$), there is a higher variance associated to SGD stochasticity than from the selection of P random points from the initial data set. Qualitatively, curves from repeated runs behave similarly.

Training error. With the exception of small fluctuations, the training error always decreases with the number of iterations t, reaching a plateau at a certain point. Furthermore, across data sets and repeated SGD runs, there is little variation in the training error curves (as compared to the same curve on the test data). The fluctuations are an expected part of SGD, being caused by the randomization in selecting a data point with respect to which the gradient is evaluated.

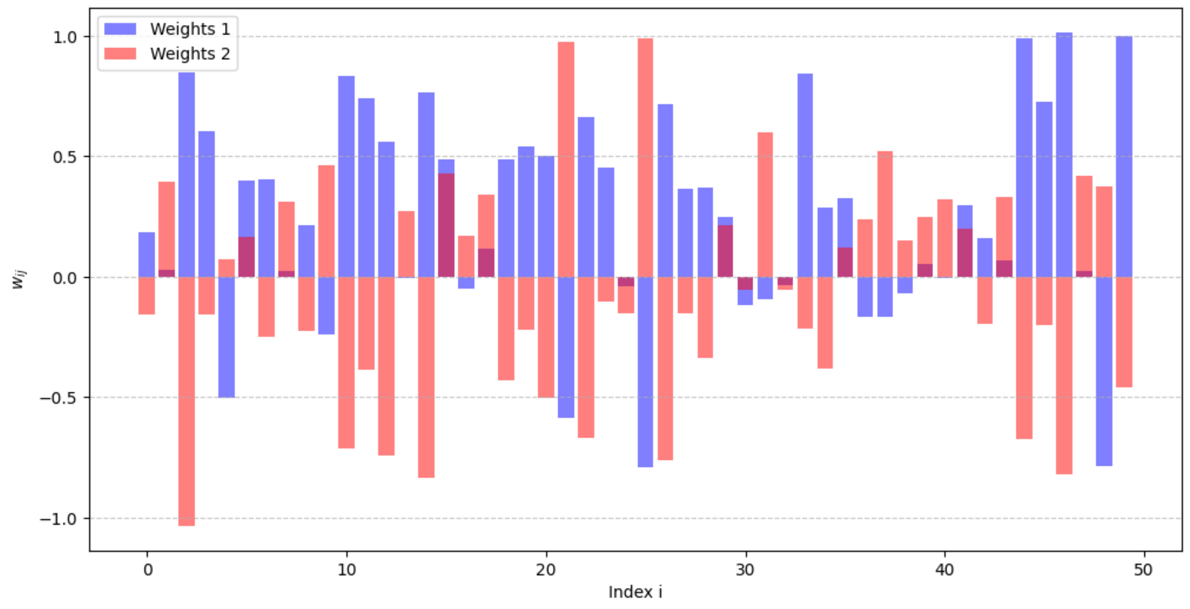

Figure 2: The weight vectors that correspond to each hidden unit, differentiated by color, at the end of the SGD routine. In the figure, the j-th bar from left to right corresponds to the j-th component of the vector w_i

Bonus. The plateaus of the training error are ordered by P: the lower P is, the smaller the training error. It could be that for low P, the current model is too complex and the data is stored ("learned by heart") instead of approaching the underlying rule (a "teacher perceptron" equivalent to the problem). A larger P means that a model of the same complexity has more data to learn by heart, which leads to higher training errors than for when P is low.

Test error. The behaviour of the test error curves is different from that of the training error curves. On top of the typical SGD fluctuations, we remark the following:

- In the main problem, as seen from the single run of Figure 1, the test curve seems to stabilize at a certain plateau after evolving through a transitory period. The transitory period is associated to the part of the training curve in which the cost transitions towards its plateau
- For repeated SGD runs at a single data set of $P = 200$ (Figure 3), averaging over the SGD fluctuations, a curious trend emerges: the test curve drops abruptly initially, after which it increases slowly. We have explored with (t_{max}, P) and found that the curve eventually stabilizes at a new plateau; the behaviour is consistent in a range of $P \in [80, 200]$. Furthermore, for many iterations, there seem to be multiple plateaus between which the test curve may transition.

  We speculate that in the cost function landscape built from the training data, there are multiple points $\{W^*\}$ in which the same optimal value is attained. While training the network, the weights can transition between these points; the transition is not reflected in the training curve because the optimal values are the same in these points. However, the cost function landscape built from the *test* data is sufficiently different from the former, such that evaluated in $\{W^*\}$, it has different values (hence the difference in plateaus). If this phenomenon is truly caused by this mechanism, then including more data in both sets can diminish this effect; adding more data makes the two landscapes more similar.

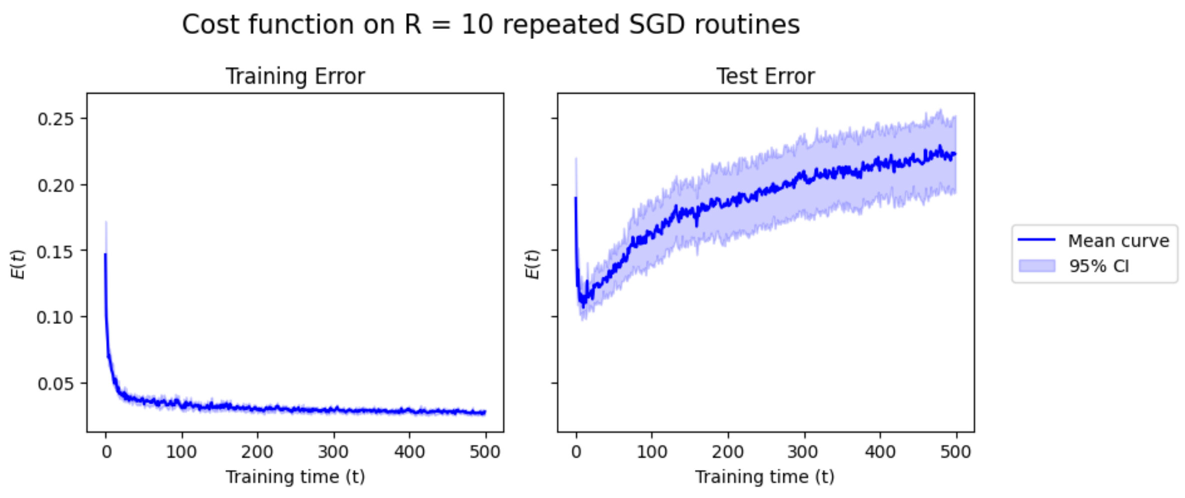

Figure 3: A repetition of the analysis in Figure 1, for multiple SGD runs. The mean curve, as well as the 95% confidence region (a measure of the variability of the $E(t)$ across SGD runs) is highlighted

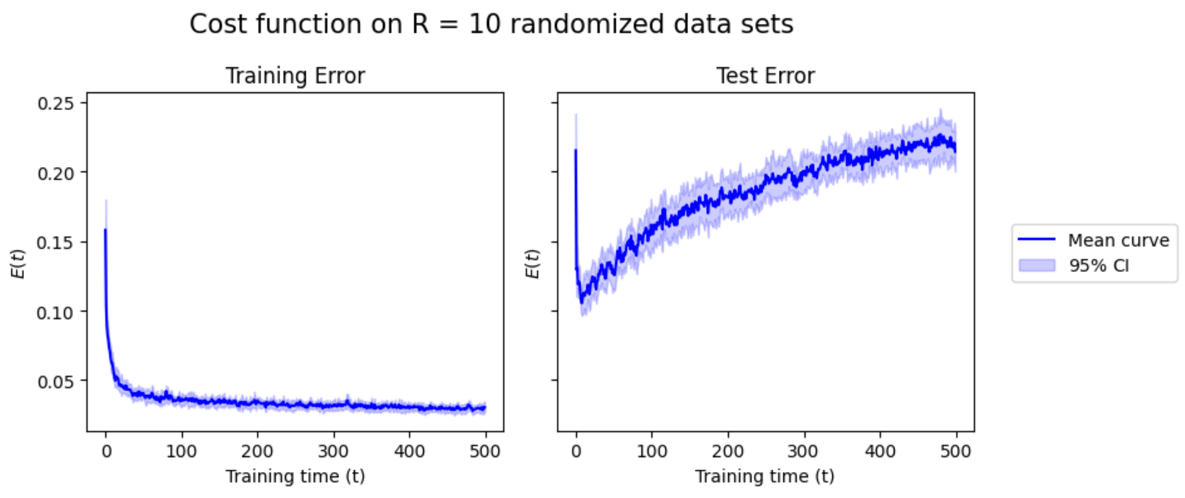

Figure 4: The cost functions for single SGD runs, but for 10 randomly selected data sets of fixed size P and Q. The data is displayed in a similar format to that of Figure 3

Shapes as in Figure 3 justify the notion of early stopping

- *Bonus.* The test curves have different behaviours qualitatively across P. As seen in Figure 5, when P is either very small or sufficiently large, the test curve transitions towards a stable plateau. It seems that the transition between plateaus is an issue which appears only for a certain range of P. We could not identify a possible cause

- *Bonus.* A larger P results in a lower test error, which quantitatively justifies the maxima *learning begins where storage ends.* With high P, the version space of W is restricted enough such that we approach the true learning rule, not only storing the training data

- *Bonus.* The variability of the test curves decreases with P. One could justify it by thinking in terms of the version space. Because the version space is not very constrained for low P, W is more likely to be affected by the randomness in the choice of data points

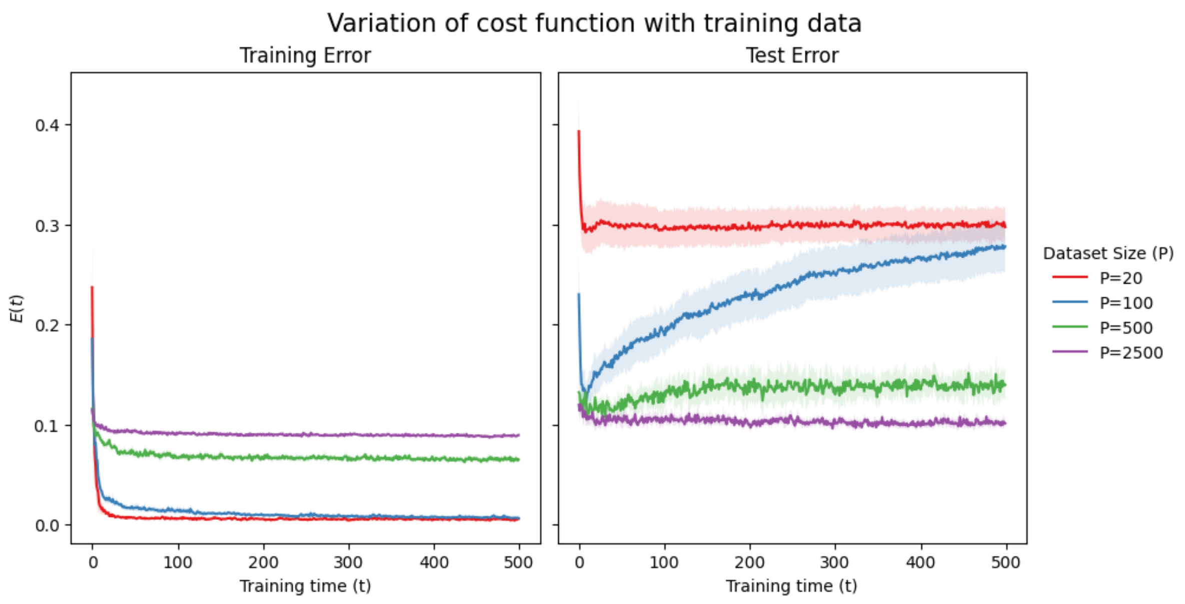

Figure 5: The figure has a similar interpretation to that in Figure 3, but for varying P

Conclusion

In this assignment, we have derived the gradient of a shallow feedforward neural network with $K = 2$ hidden nodes and a sigmoidal activation function, and we have provided a learning method via Stochastic Gradient Descent. It was found that the training errors and the test errors behave very differently as a function of the number of training iterations. The training error always decreases or stabilizes, whereas the generalization error is more complicated: it may transition between different plateaus (for better or for worse), or essentially not change. It is apparent why early stopping is a sensible strategy, especially for few data points relative to the number of dimensions. By the evolution of the test error, we can see how a model that is too complex can overfit and "store rather than learn". Unsurprisingly, including more data in the training set always has a benefit. We have identified a lowering of the generalization error, and stability against the stochasticity of SGD routines.

Individual contributions to the assignment

The division of tasks was fluid, as we have helped each other with ideas and verifications everywhere. Main contributions:

- Andrei Secuiu: Discussion, Conclusion.
- Roberto Schinina: Introduction, Method
- Both: The coding was done in parallel and the results were compared