Machine Learning 2016/2017, Assignment 5

Yevgeny Seldin, Christian Igel
Department of Computer Science, University of Copenhagen

The deadline for this assignment is **12:00 pm (noon, not midnight) 2/01/2017**. You must submit your individual solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your source code in this PDF file.
- Your solution source code (Matlab / R / Python scripts or C / C++ / Java code) with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- Your code should be structured such that there is one main file that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Your code should also include a README text file describing how to compile and run your program, as well as list of all relevant libraries needed for compiling or using your code.

1 Neural Networks

In this part of the assignment, we consider standard feedforward neural networks (also called multi-layer perceptrons) with a single hidden layer.

In the experiments, we use artificial toy data stored in the files `sincTrain25.dt` and `sincValidate10.dt` . The data has been generated from a sinc : $\mathbb{R} \to \mathbb{R}$ function

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x} \tag{1}$$

with additive normally distributed noise. This is a frequently used benchmark function for regression tasks. Here we use this artificial example because it can be easily visualized and therefore helps you to find possible mistakes in your implementation.

1.1 Neural network implementation

Implement a multi-layer neural network with a linear output neuron and a single hidden layer with non-linear neurons. All neurons should have bias (offset) parameters.

For the hidden neurons, use the non-linearity (transfer function, activation function)

$$h(a) = \frac{a}{1+|a|} \tag{2}$$

with derivative

$$h'(a) = \frac{1}{(1+|a|)^2} \ . \tag{3}$$

(Not for submission: Check that the derivative is correct. To this end, consider the cases $a < 0$ and $a > 0$ separately and then discuss what happens at $a = 0$.)

Consider the mean-squared error as loss/error function E. Implement backpropagation to compute the gradient of the error with respect to the network parameters.

Compute gradients of the network using some arbitrary sample data. For instance, you could use parts of the sinc data. To verify your implementation, calculate the numerically estimated partial derivatives of each network parameter $[\boldsymbol{w}]_i$ by computing

$$\frac{\partial E(\boldsymbol{w})}{\partial [\boldsymbol{w}]_i} \approx \frac{E(\boldsymbol{w} + \epsilon \boldsymbol{e}_i) - E(\boldsymbol{w})}{\epsilon} \tag{4}$$

for small positive $\epsilon \ll 1$. Here, the vector $\boldsymbol{w}$ is composed of all neural network parameters (weights w_{ij} and bias parameters w_{i0} for all i and j), the ith component of $\boldsymbol{w}$ is denoted by $[\boldsymbol{w}]_i$, and $\boldsymbol{e}_i$ denotes a vector of all zeros except for the ith component that is 1. Compare the numerically estimated gradients with the analytical gradients computed using backpropagation. These should be very close (i.e., differ less than, say, 10^{-8}) given a careful adjustment of ϵ.

Deliverables: source code of neural network with a single hidden layer including backpropagation to compute partial derivatives; verification of gradient computation using numerically estimated gradients; derivation of derivative of the transfer function

1.2 Neural network training

The goal of this exercise is to gather experience with gradient-based optimization of models, to understand the influence of the number of hidden units in neural networks, and to think about early-stopping and overfitting.

For all experiments in this part of the assignment, use the sample data in `sincTrain25.dt`.

Do not produce a single plot for every function you are supposed to visualize. Combine results in the plots in a reasonable, instructive way.

Apply gradient-based (batch) training to your neural network model. For the exercise, it is sufficient to consider standard steepest descent. In practice, more advanced gradient based optimization algorithms are advisable for batch training (e.g., RProp).

Train neural networks with 2 and 20 hidden neurons using all the data in `sincTrain25.dt`. Use batch learning until the error on the training data stops decreasing significantly (make sure that you watch it long enough). The validation data is used to monitor the training process, but not for gradient-based optimization of the network weights. Feel free to play around with the number of hidden neurons and different learning rates. It is not part of the assignment, but you are encouraged to consider other datasets and to explore the benefits of shortcut connections.

What happens for very small learning rates? What happens for very large learning rates? Plot the mean-squared error on the training set as well as on the validation data set `sincValidate10.dt` over the course of learning. That is, generate a plot with the learning epoch on the x-axis and the error on the y-axis. Use a logarithmic scale on the y-axis. Briefly discuss the results.

Plot both the function (1) and the output of your trained neural networks over the interval $[-10, 10]$ (e.g., by sampling the functions at the points -10, -9.95, -9.9, -9.85,. . . ,9.95, 10).

Comment on overfitting and how early-stopping can be used to prevent overfitting.

Deliverables: Plots of error trajectories and final solutions of neural networks with 2 and 20 neurons using steepest descent with different learning rates, respectively; brief discussion of the plots; discussion of overfitting and early-stopping in the context of the experiments

2 The growth function

1. Let $\mathcal{H}$ be a finite hypothesis set with $|\mathcal{H}| = M$ hypotheses. Prove that $m_{\mathcal{H}}(n) \leq \min\{M, 2^n\}$. What is the VC-dimension of $\mathcal{H}$?
2. Prove that $m_{\mathcal{H}}(2n) \leq m_{\mathcal{H}}(n)^2$.
3. Prove by induction that
$$\sum_{i=0}^{d} \binom{n}{i} \leq n^d + 1.$$
4. Use the above result to derive a bound on $m_{\mathcal{H}}(n)$.
5. Substitute the result into the VC generalization bound (note that bounding $m_{\mathcal{H}}(2n)$ directly is tighter than going via the result in Point 2). What should be the relation between d and n in order for the bound to be meaningful?

3 VC-dimension

You do not have to be very formal in this question, some "hand-waving" is allowed. A formal (mathematical) proof will earn extra honor and credit.

1. Let $\mathcal{H}_+$ be the class of "positive" circles in $\mathbb{R}^2$ (each $h \in \mathcal{H}_+$ is defined by the center of the circle $c \in \mathbb{R}^2$ and its radius $r \in \mathbb{R}$; all points inside the circle are labeled positively and outside negatively). What is the VC-dimension $d_{VC}(\mathcal{H}_+)$?
2. Let $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_-$ be the class of "positive" and "negative" circles in $\mathbb{R}^2$ (the "negative" circles are negative inside and positive outside). What is the VC-dimension of $\mathcal{H}$?