Sapienza University of Rome

Optimization Methods for Machine Learning

Homework 1

Ivan Bergonzani, Michele Cipriano

November 28, 2017

1 Introduction

The aim of the homework is to train and compare different neural networks on a regression problem. In particular, the regression task is performed on the Franke's function, building a dataset from it by sampling 100 random points (x^i, y^i) with noise, i.e. with $y^i = F(x^i) + \varepsilon^i$ where ε^i is a random number in $[-10^{-1}, 10^{-1}]$ and F is the Franke's function. The dataset has been split into a training set (70% of the dataset) and a test set (the remaining 30%).

Two architectures have been compared using different training methods and different hyperparameters. In particular, the multi-layer perceptron (MLP) and the radial basis function network (RBFN) have been trained with full minimization, two blocks method and decomposition method. In the full minimization, the training error is minimized w.r.t all the weights using the gradient descent algorithm. In the two blocks method, for the MLP the error is minimized via an extreme learning procedure and for the RBFN the error is minized by first selecting the centers through a clustering algorithm and then by solving a linear least squares problem. In the decomposition method the error is minimized by alternating a convex minimization w.r.t. the output weights and a non-convex minimization w.r.t. all the other weights.

The best result has been obtained with TODO, which has an error of **TODO** on the test set. The project has been developed in Python with TensorFlow and Numpy for the learning algorithms and the computation of the tensors.

2 Full Minimization

As mentioned before, the full minimization is done by using the gradient descent algorithm on the training error function, defined as:

$$E(\omega;\pi) = \frac{1}{2P}\sum_{p=1}^{P} \|f\left(x^p\right) - y^p\|^2 + \rho\|\omega\|^2$$

where $\omega = (v,\ w,\ b)$ are the weights, $\pi = (N,\ \rho,\ \sigma)$ are the hyperparameters, P is the dimension of the training set and f is the function computed by the neural network.

Let's consider a shallow MLP with linear output:

$$f(x) = \sum_{j=1}^{N} v_j g\left(\sum_{i=1}^{n} w_{ji}x_i - b_j\right)$$

with n dimension of the input, N dimension of hidden layer, $v \in \mathbb{R}^N$ weights from the hidden to the output layer and g activation function, defined as:

$$g(t) = tanh\left(\frac{t}{2}\right) = \frac{1-e^{-\sigma t}}{1+e^{-\sigma t}}$$

The training has been performed over 15000 iterations on the training set with a learning rate $\eta = 0.001$. The hyperparameters have been tuned with a grid search over all the possible combinations of the hyperparameters in:

$$(N,\rho,\sigma) \in \left\{25, 50, 75\right\} \times \left\{10^{-3}, 10^{-4}, 10^{-5}\right\} \times \left\{1, 2, 3, 4\right\}$$

The best result obtained an error of TODO on the training set and an error of TODO on the test set as it is also possible to see in table 5.1. The hyperparameters found are $N = 0$, $\rho = 0$ and $\sigma = 0$. In figure 5.1 the plot of the function computed by the MLP found.

TODO: explain overfitting/underfitting

An analogous experiment has been done with the radial basis function network, defined as:

$$f(x) = \sum_{j=1}^{N} v_j \phi(\|x^i - c_j\|)$$

where $c_j \in \mathbb{R}^2$ is the center of the j-th hidden neuron, $v \in \mathbb{R}^N$ weights from the hidden to the output layer and ϕ is the activation function, defined as:

$$\phi(\|x - c_j\|) = e^{-(\|x-c_j\|/\sigma)^2}$$

with $\sigma > 0$. The training has been performed over 15000 iterations on the training set with a learning rate $\eta = 0.001$. The hyperparameters have been tuned with a grid search over all the possible combinations of the hyperparameters in:

$$(N,\rho,\sigma) \in \left\{25, 50, 70\right\} \times \left\{10^{-3}, 10^{-4}, 10^{-5}\right\} \times \left\{0.25, 0.5, 0.75, 1\right\}$$

The best result obtained an error of TODO on the training set and an error of TODO on the test set (see table 5.1). The hyperparameters found are $N = 0$, $\rho = 0$ and $\sigma = 0$. In figure 5.1 the plot of the function computed by the RBFN found.

As already said before, both training procedures for MLP and RBFN have been performed using the gradient descent algorithm for 15000 iterations with a learning rate $\eta = 0.001$. This step has been implemented with TensorFlow's gradient descent optimizer, defined in `tf.train.GradientDescentOptimizer`. For each hyperparameter setting the training took 0.00s for the MLP and 0.00s for the RBFN on average.

TODO: explain overfitting/underfitting

TODO: MLP vs RBFN

3 Two Blocks Method

The idea, here, is to train the network by first setting up a subset of weights and then by solving efficiently the least squares problem obtained. This speeds up the training of the network obtaining (similar?) results w.r.t the previous section.

For what regards the MLP all the weights w_{ji} and the biases b_j, with $j \in \{1, ..., N\}$ and $i \in \{1, ..., n\}$, have been set up randomly. This reduces the minimization problem to the solution of:

$$\nabla_v E(\omega; \pi) = 0 \tag{3.1}$$

where the hyperparameters π are the values found in the previous section. Hence, the problem becomes a simple linear system:

$$\left(\frac{1}{2P} G^T G + \rho I \right) v = \frac{1}{2P} G^T y$$

where $G \in \mathbb{R}^{P \times N}$ is defined as:

$$G_{rc} = g \left(\sum_{i=1}^{n} w_{ci} x_i^r - b_c \right)$$

This method has been performed 10000 times obtaining an error of TODO on the training set and an error of TODO on the test set (table 5.1). In figure 5.1 the plot of the function computed by the MLP found.

TODO: COMPARISON WITH Q1E1

An analogous procedure has been applied to RBFN. The centers c_j, with $j \in \{1, ..., N\}$ have been selected by using the K-means clustering algorithm. This allowed, as before, to reduce the problem to the solution of equation 3.1. The hyperparameters are the ones found in the previous section. Here G is defined as:

$$G_{ij} = \phi \left(\|x^i - c_j\| \right)$$

This method has been performed 10000 times obtaining an error of TODO on the training set and an error of TODO on the test set (table 5.1). In figure 5.1 the plot of the function computed by the RBFN found.

TODO: COMPARISON WITH Q1E2

TODO: implementation of both MLP and RBFN

Neural Network	N	σ	ρ	Training Error	Test Error	Time
Full MLP	TBD	TBD	TBD	TBD	TBD	TBD
Full RBFN	TBD	TBD	TBD	TBD	TBD	TBD
Extreme MLP	TBD	TBD	TBD	TBD	TBD	TBD
Unsupervised c. RBFN	TBD	TBD	TBD	TBD	TBD	TBD
TBD	TBD	TBD	TBD	TBD	**TBD**	TBD

Table 5.1: TODO

4 Decomposition Method

TODO: E3

5 Conclusion

Conclusion. (Remember to say that all experiments have been performed on a kind of computer)

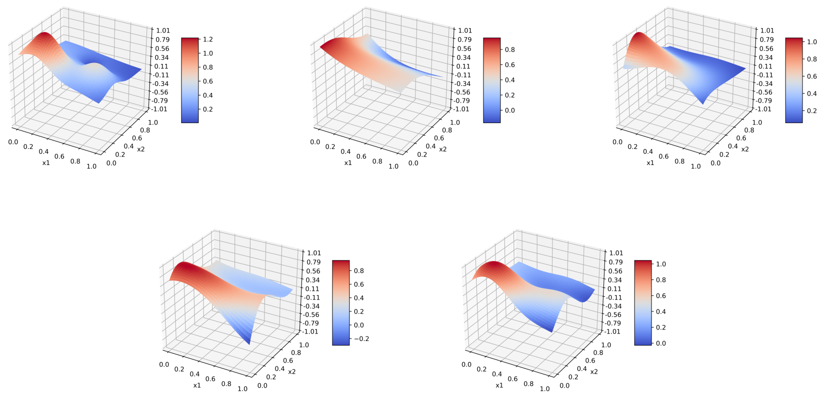

Figure 5.1: In order: the Franke's function and the functions obtained by Full MLP, Full RBFN, Extreme MLP, Unisupervised c. RBFN and TBD. The hyperparameters used for each network are specified in table 5.1. ANYTHING ELSE TO ADD?