Università degli studi di Padova

Homework #1 Supervised Deep Learning

Monaco Saverio - 2012264

Neural Networks and Deep Learning - Professor: A. Testolin

***Abstract*—As for the first homework, the main models of Supervised Deep Learning are investigated. For the regression task, it is needed to effectively approximate a noisy unknown 1-dimensional function. The classification task instead consists in building a Convolutional Neural Network for the FashionMNIST Dataset.**
For both tasks, more advanced techniques are further explored and compared.

I. Regression task

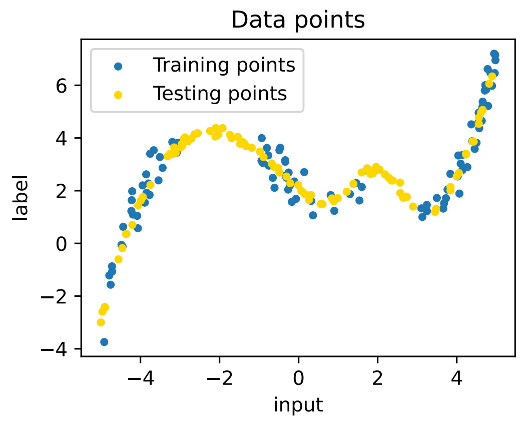
The task of Regression in a Neural Network framework consist in approximating a scalar function:

$$f : \mathbb{R} \to \mathbb{R}$$

through the use of a Network. Usually, the deeper the network is, the more complex patterns and behaviours of the target function it can grasp, however it might also encounter overfitting.
For the current exercise, training and testing points are generated according to a theoretical and unknown function, plus some noise:

$$\hat{y} = f(x) + noise$$

The data for the task is the following:



Data seems to be generated from a grade-5 polynomial function. It is worth noticing that the training dataset is not optimal, compared to the testing dataset: it is noisier and it has missing values, namely for $x \in [-3, -1]$ and $x \in [2, 3]$.. The latter obstacle will be the most hard problem to overcome, since it requires the model to properly generalize on those two areas it has never been trained on.

A. *Methods*

For the Regression task, it was implemented a Python class: `RegFC` to generate a Fully Connected Neural Network (FCNN) model with the following parameters:

- `Ni`: dimension of input vector (int);
- `No`: dimension of output vector (int);
- `Nhs`: number of neurons in each hidden layer (list of int);
- `activation`: torch activation function;
- `dropout`: dropout rate after each hidden layer (float);
- `lr0`: initial Learning Rate for the optimizer (float);
- `reg_term` : Regularization term for the optimzier (float);

Both `Ni` and `No` must be 1, being the model a 1-D Regressor, while the depth of the hidden layers was kept general: for example `Nhs = [10,20,10]` will create a 1-10-20-10-1 model.

Loss function and optimizer were instead kept fixed:

- Loss function: `nn.MSELoss()` (L2 norm);
- Optimizer: `optim.Adam()`;

The models were evaluated by the Validation Error implementing *k-fold cross validation* using the skleanr function

`sklearn.model_selection.KFold`.
To obtain optimal hyperparameters a Grid Search was manually implemented with nested for loops and keeping the model with the lowest validation error.

Parameters of Grid-Search

Lr	1e-2, 1e-3
Reg. terms	1e-2, 1e-1
H. layers	[1000,1000], [1000,1000,1000]
Dropout	1e-2, 1e-1
Act. fun.	nn.Tanh(), nn.Sigmoid()

B. Results

1) Fully Connected Neural Network:

Best parameters

Lr	1e-3
Reg. terms	1e-2
H. layers	[1000,1000,1000]
Dropout	1e-2
Act. fun.	nn.Tanh()

MSE on Test set: 0.27585697

2) Polynomial model: Lastly, the FCNN was compared to a much simpler model: a *Polynomial model* fitted to the training dataset fitted using the method of least squares.
In this particular case, in which data is clearly generated by adding noise to a polynomial function, a Polynomial model can be considered as a viable alternative, however it restricts our hypothesis space to a much smaller space, meaning that it is not a good option in a general scenario.

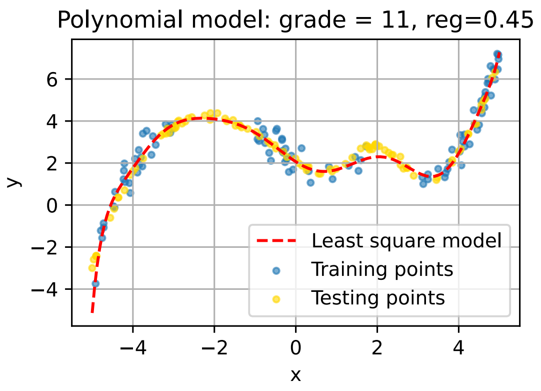
The best model was found by the use of a grid search of the parameters:

- `grade`: maximum grade of the fitting polynomial function;
- `reg`: regularization term;

applying the least square method on the training dataset and choosing the model with the lowest error on the test dataset. However, as one can see, the grade of the best model may be too high and it will not generalize well for points outside the range [-4,4].

II. Classification Task

The objective of classification is to obtain a *rule* that outputs the most probable label (belonging to discrete space $\mathcal{L}$) starting from a set of parameters



$\mathcal{X}$.
Solely for Classification, it is possible to define a particular metric that intuitively tells how well the model is performing:

$$Accuracy : \frac{\#\text{samples} - \text{misclassified\_samples}}{\#\text{samples}}$$

A. Methods

B. Results

III. Conclusions

a

IV. **Appendix**