LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework 3
Deadline 18/10/2024 21:00
Submit on Fenix as pdf

I) Polynomial Regression (3 pts)

Consider a training set with 5 observations (sample) with dimension $D = 1$

$$x_1=-1,\ x_2=1,\ x_3=-1.2,\ x_4=1.4,\ x_5=1.9$$

With targets

$$t_1=-2,\ t_2=3,\ t_3=-3,\ t_4,=0,\ t_5=-3$$

$$\phi_j(x) = x^j$$

which can lead to a polynomial regression of the third degree

$$y(x, \mathbf{w}) = \sum_{j=0}^{3} w_j \cdot \phi_j(x) = w_0 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3.$$

You can (should?) use your computer with Python, NumPy, MATLAB, Octave, Mathematica, etc. (whatever tool/language you like). Please indicate your results step by step. In the exam the examples will be much simpler, so you can get a solution using a calculator right on time.

(a) (1 pts)

Compute the design matrix Φ.

(b) (1 pts)

Compute the polynomial regression weights.

(c) (1 pts)

Bayesian regression (l2 regularization) a closed form solution exists. Why does a closed form solution not exist for LASSO regression (l1 regularization)?

LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework 3
Deadline 18/10/2024 21:00
Submit on Fenix as pdf

II) Neural Network NN (4 pts)

Given the weights.:

$$W^{[1]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$b^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$W^{[2]} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix}$$

$$b^{[2]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

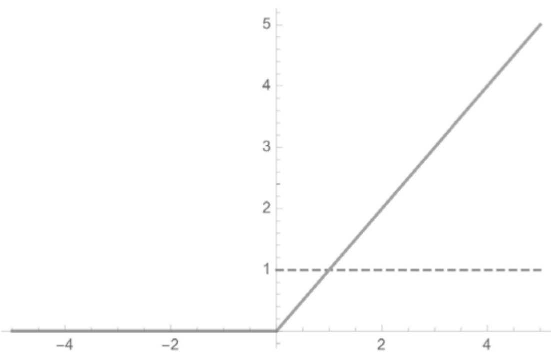
and the activation function ReLU

Rectifier also known as a ramp function

$$f(x) = \max(0, x). \tag{12.3}$$

is defined as the positive part of its argument [Jarrett *et al.* (2009)], [Nair and Hinton (2009)], [Goodfellow *et al.* (2016)]. The function is non-differentiable at zero; however, it is differentiable anywhere else and we can use the subderivative with sgn_0 function

$$f'(x) = sgn_0(x) = \begin{cases} 1 \text{ if } & x \geq 0 \\ 0 \text{ if } & x < 0 \end{cases}. \tag{12.4}$$



of the hidden layer and SoftMax of the output layer using the cross-entropy error loss do a stochastic gradient descent update (with learning rate η = 0.1) for the training example:

LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework 3
Deadline 18/10/2024 21:00
Submit on Fenix as pdf
$\mathbf{x}=(1,1,1,1,1)^T$ and the target $\mathbf{t}=(1,0)^T$,

III Software Experiments (3pts)

Download the jupyter notebook HM3__24_NN.ipynb.

Split the data using the command (in the notebook)

```
digits = datasets.load_digits()

X, y = digits.data, digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.5,
stratify=y, random_state=your_group number)
```

Compare the accuracy on the test set of Logistic Regression with NN.

```
predictor = MLPClassifier(hidden_layer_sizes=(10,4),random_state=42,activation
='logistic',solver='sgd',max_iter=2000)
```

Layer size 10, 4 means two hidden layers, first layer 10 neurons and second hidden layer 4 neurons.
Can you improve accuracy on the test set by changing the parameters of hidden_layer_size?
(a) (2pts)
Indicate your best parameters of the hidden layer and output size? Indicate the loss curve. What is your conclusion (one sentence)?

(b) (1 pts)
Using the best architecture change for the activation function 'identity'. Is the result still better?
Indicate why not to use the identity function in the hidden layer

Pls, do not spend too much time on the experiments! You should get a feeling what the search for the right parameters for NN models is, not the search for the best model!