MEIC 2020/2021
Aprendizagem - Machine Learning
Homework II
Deadline 23/04/2021
Submit on Fenix as pdf

I) Polynomial Regression (6 pts)

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

$$x_1=0,\ x_2=1.5,\ x_3=3,\ x_4=4.5,\ x_5=6$$

With targets

$$t_1=1,\ t_2=0,\ t_3=-1,\ t_{4,}=0,\ t_5=1$$

Consider as well the basis function

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

which can lead to a polynomial regression of the third degree

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

(a) (1 pts)

Compute the design matrix Φ.

(b) (1 pts)

Compute the polynomial regression weights.

(c) (1 pts)

Determine the gradient of

$$E(\mathbf{w}) = \frac{1}{2} \cdot \sum_{k=1}^{M} (t_k - \mathbf{w}^T \cdot \Phi_k)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

(d) (1 pts)

Perform the ridge regression (l2 regularization) using the closed solution (use the design matrix which you computed in (a)) with $\log(\lambda/2) = 0$.

(e) (1 pts)

Determine the gradient of

$$E(\mathbf{w}) = \frac{1}{2} \cdot \sum_{k=1}^{M} (t_k - \mathbf{w}^T \cdot \Phi_k)^2 + \lambda \cdot \|\mathbf{w}\|_1$$

(f) (1 pts)

LASSO regression (l1 regularization) lacks a closed form solution, why?

II) Neural Network NN (14 pts)

(a) (7 pts)

Given the weights

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

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

$$W^{[2]} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

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

$$W^{[3]} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$b^{[3]} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

and the activation function: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = tanh(x)$

using the squared error loss do a stochastic gradient descent update (with learning rate $\eta = 0.1$) for the training example:

$\mathbf{x}=(1,1,1,1,1)^T$ and the target $\mathbf{t}=(1,-1)^T$,

(b) (1pts)

We cannot use the cross entropy error. Why?

(c) (6 pts)

With the output units having a softmax activation function (the hidden units having the hyperbolic tangent activation function tanh as before) and the error function being cross-entropy do a stochastic gradient descent update (with learning rate $\eta = 0.1$) for the training example:

$$\mathbf{x}=(1,1,1,1,1)^T \text{ and the target } \mathbf{t}=(1,0)^T,$$