TÉCNICO LISBOA

LEIC-T 2023/2024
Aprendizagem - Machine Learning
Homework 3
Deadline 20/10/2024 20:00
Submit on Fenix as pdf

I) Polynomial Regression (3 pts)

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

x1=-0.8, x2=1, x3=-1.2, x4=1.4, x5=1.9

With targets

$t_1 = -20,\ t_2 = 20,\ t_3 = -10,\ t_4 = 13,\ t_5 = 12$

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.

$$\Phi = \begin{pmatrix} 1 & x^1 & x^{1^2} & x^{1^3} \\ 1 & x^2 & x^{2^2} & x^{2^3} \\ 1 & x^3 & x^{3^2} & x^{3^3} \\ 1 & x^4 & x^{4^2} & x^{4^3} \\ 1 & x^5 & x^{5^2} & x^{5^3} \end{pmatrix} = \begin{pmatrix} 1 & -0.8 & 0.64 & -0.512 \\ 1 & 1 & 1 & 1 \\ 1 & -1.2 & 1.44 & -1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.9 & 3.61 & 6.589 \end{pmatrix}$$

(b) (1 pts)

Compute the polynomial regression weights.

$\Phi^{\dagger} = (\Phi^T\Phi)^{-1}\Phi^T \mathrm{t} =$

$$= \left(\begin{pmatrix} 1 & -0.8 & 0.64 & -0.512 \\ 1 & 1 & 1 & 1 \\ 1 & -1.2 & 1.44 & -1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.9 & 3.61 & 6.589 \end{pmatrix}^T \begin{pmatrix} 1 & -0.8 & 0.64 & -0.512 \\ 1 & 1 & 1 & 1 \\ 1 & -1.2 & 1.44 & -1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.9 & 3.61 & 6.589 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & -0.8 & 0.64 & -0.512 \\ 1 & 1 & 1 & 1 \\ 1 & -1.2 & 1.44 & -1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.9 & 3.61 & 6.589 \end{pmatrix}^T \begin{pmatrix} -20 \\ 20 \\ -10 \\ 13 \\ 12 \end{pmatrix}$$

$$= \begin{pmatrix} -8.998594 \\ 24.2188612 \\ 8.685827 \\ -8.471881 \end{pmatrix}$$

$w_0 = -8.998594$

$w_1 = 24.2188612$

$w_2 = 8.685827$

$w_3 = -8.471881$

(c) (1 pts)

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

A regressão LASSO (com regularização L1) não possui uma solução em forma fechada devido à natureza não suave da penalização L1. Isso impede a aplicação direta da derivação e exige, em vez disso, o uso de métodos iterativos de otimização para encontrar os coeficientes ótimos.

II) Neural Network NN (4 pts)

Given the weights.:

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

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

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

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

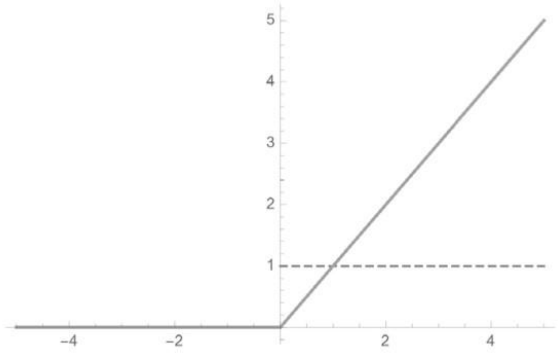
and the activation function ReLU

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:

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

$$Z^{[1]} = W^{[1]} X^{[0]} + b^{[1]} = \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}$$

$$X^{[1]} = \text{ReLU}(Z^{[1]}) = \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}$$

$$Z^{[2]} = W^{[2]} X^{[1]} + b^{[2]} = \begin{pmatrix} 15 \\ 15 \end{pmatrix}$$

$$X^{[2]} = \text{SoftMax}(Z^{[2]}) = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$\delta^{[2]} = X^{[2]} - t = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}$$

$$\delta^{[1]} = \left(\frac{\partial Z^{[2]}}{\partial X^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial X^{[1]}}{\partial Z^{[1]}} = (W^{[2]})^T \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix} (\text{ReLU}(Z^{[2]}))' = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\frac{\partial E}{\partial W^{[1]}} = \delta^{[1]} \cdot \left(\frac{\partial Z^{[1]}}{\partial X^{[1]}}\right)^T = \delta^{[1]} \cdot (X^{[0]})^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$W^{[1]} = W^{[1]} - \eta \frac{\partial E}{\partial W^{[1]}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\frac{\partial E}{\partial b^{[1]}} = \delta^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$b^{[1]} = b^{[1]} - \eta \frac{\partial E}{\partial b^{[1]}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\frac{\partial E}{\partial W^{[2]}} = \delta^{[2]} \cdot \left(\frac{\partial Z^{[2]}}{\partial W^{[2]}}\right)^T = \delta^{[2]} \cdot \left(X^{[1]}\right)^T = \begin{pmatrix} -2.5 & -2.5 & -2.5 \\ 2.5 & 2.5 & 2.5 \end{pmatrix}$$

$$W^{[2]} = W^{[2]} - \eta \frac{\partial E}{\partial W^{[2]}} = \begin{pmatrix} 1.25 & 1.25 & 1.25 \\ -6{,}5 & -6.5 & 0.75 \end{pmatrix}$$

$$\frac{\partial E}{\partial b^{[2]}} = \delta^{[2]} = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}$$

$$b^{[2]} = b^{[2]} - \eta \frac{\partial E}{\partial b^{[1]}} = \begin{pmatrix} 0.05 \\ -0.05 \end{pmatrix}$$

III Software Experiments (3pts)

Download the jupyter notebook HM3_NN.ipynb.

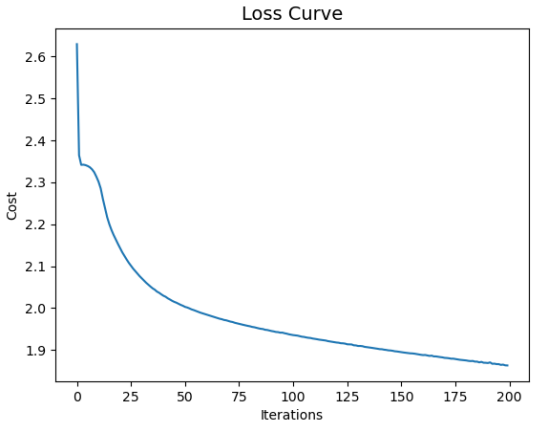
Split the data using the command (in the notebook)

```
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=**0.7**,stratify=y,random_state=**your_group number**)

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

Output:
train size: 1257
test size: 540
Logistic Regression → accuracy on testing set: 0.97
NN → accuracy on testing set: 0.2167



A precisão do classificador de regressão logística é significativamente maior (aproximadamente 97%) em comparação com a precisão do classificador de rede neural (aproximadamente 21,67%) no mesmo conjunto de teste. Isto sugere que, para este conjunto de dados e configuração específicos, a regressão logística tem um desempenho muito superior em termos de precisão. A rede neural pode necessitar de ajustes adicionais nos seus parâmetros para melhorar o seu desempenho.

```
MLPClassifier(hidden_layer_sizes=(10,4),random_state= your_group number,activation
='relu',solver='sgd')
```

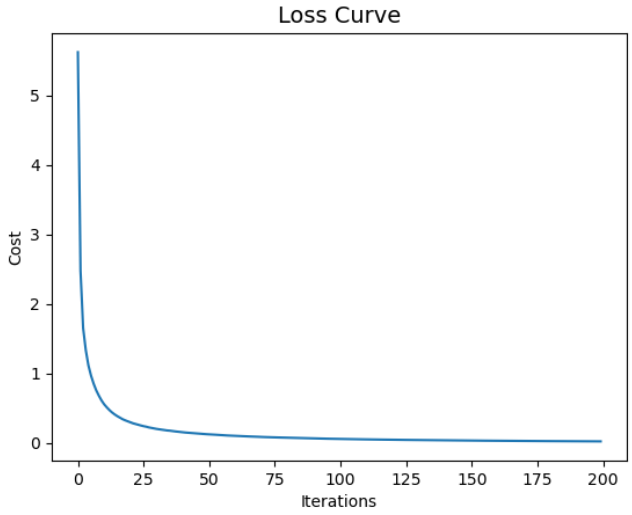
Layer size 10, 4 means two hidden layers, first layer 10 neurons and second hidden layer 4 neurons.

Can you improve accuracy on the test set by changing the parameters of hidden_layer_size? Indicate your best parameters of the hidden layer size? Indicate the loss curve. What is your conclusion concer?

Pls do not spend too much time on the experiments!!

Utilizando os parâmetros (40,30) é possível obter o seguinte output:

train size: 1257
test size: 540
Logistic Regression → accuracy on testing set: 0.97
NN → accuracy on testing set: 0.9722



Com estes parâmetros é possível perceber uma melhoria significativa na precisão, sendo esta agora de 97,22%, o que comprova que é possível melhorar a precisão de NN utilizando valores diferentes para a hidden_layer_size.
Também é possível perceber que com uma hidden_layer_size de (40,30), conseguimos uma precisão muito parecida com a precisão de Logistic Regression.