# Deep Learning (IST, 2022-23)

# Practical 3: Linear and Logistic Regression

André Martins, Andreas Wichert, Taisiya Glushkova, Luis Sá Couto, Margarida Campos

## Pen-and-Paper Exercises

The following questions should be solved by hand. You can use, of course, tools for auxiliary numerical computations.

### Question 1

Consider the following training data:

$$\boldsymbol{x}^{(1)} = [-2.0], \boldsymbol{x}^{(2)} = [-1.0], \boldsymbol{x}^{(3)} = [0.0], \boldsymbol{x}^{(4)} = [2.0]$$

$$y^{(1)} = 2.0, y^{(2)} = 3.0, y^{(3)} = 1.0, y^{(4)} = -1.0.$$

1. Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..
2. Predict the target value for $\boldsymbol{x}_{\text{query}} = [1]$.
3. Sketch the predicted hyperplane along which the linear regression predicts points will fall.
4. Compute the mean squared error produced by the linear regression.

### Question 2

Consider the following training data:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}^{(2)} = \begin{bmatrix} 0 \\ 0.25 \end{bmatrix}, \quad \boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}^{(4)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$y^{(1)} = 0, \quad y^{(2)} = 1, \quad y^{(3)} = 1, \quad y^{(4)} = 0$$

In this exercise, we will consider binary logistic regression:

$$p_{\boldsymbol{w}}(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})}$$

And we will use the cross-entropy loss function:

$$L(\boldsymbol{w}) = -\sum_{i=1}^{N} \log\left(p_{\boldsymbol{w}}\left(y^{(i)} \mid \boldsymbol{x}^{(i)}\right)\right) = -\sum_{i=1}^{N}\left(y^{(i)} \log \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right)\right)$$

1. Determine the gradient descent learning rule for this unit.
2. Compute the first stochastic gradient descent update assuming an initialization of all zeros. Assume a learning rate of 1.0.

## Programming Exercises

The following exercises should be solved using Python, you can use the corresponding practical's notebook for guidance.

1. Consider the following training data:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \boldsymbol{x}^{(2)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \boldsymbol{x}^{(4)} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$y^{(1)} = 1.4, y^{(2)} = 0.5, y^{(3)} = 2, y^{(4)} = 2.5$$

   (a) Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data.

   (b) Predict the target value for $\boldsymbol{x}_{\text{query}} = \begin{bmatrix} 2 & 3 \end{bmatrix}^\top$.

   (c) Sketch the predicted hyperplane along which the linear regression predicts points will fall.

   (d) Compute the mean squared error produced by the linear regression.

2. Consider the following training data:

$$\boldsymbol{x}^{(1)} = [3], \quad \boldsymbol{x}^{(2)} = [4], \quad \boldsymbol{x}^{(3)} = [6], \quad \boldsymbol{x}^{(4)} = [10], \quad \boldsymbol{x}^{(5)} = [12]$$

$$y^{(1)} = 1.5, \quad y^{(2)} = 11.3, \quad y^{(3)} = 20.4, \quad y^{(4)} = 35.8, \quad y^{(5)} = 70.1$$

   (a) Adopt a logarithmic feature transformation $\phi(x_1) = \log(x_1)$ and find the closed form solution for this non-linear regression that minimizes the sum of squared errors on the training data.

   (b) Repeat the exercise above for a quadratic feature transformation $\phi(x_1) = x_1^2$.

   (c) Plot both regressions.

   (d) Which is a better fit, a) or b)?

3. Consider training set and problem setting of **Question 2** from the Pen-and-Paper exercises.

   (a) Compute three epochs of gradient descent update assuming an initialization of all zeros. Assume a learning rate of 1.0.

   (b) Compute three epochs of stochastic gradient descent update assuming an initialization of all zeros. Assume a learning rate of 1.0.

   (c) Plot final predicted separation hyperplanes.

4. Now it's time to try multi-class logistic regression on real data and see what happens.
   Load the UCI handwritten digits dataset using `scikit-learn`.
   This is a dataset containing 1797 8x8 input images of digits, each corresponding to one out of 10 output classes.

   (a) Randomly split this data into training (80%) and test (20%) partitions.

   (b) Implement a function that performs one epoch of SGD for multi-class logistic regression

   (c) Run 100 epochs of your algorithm on the training data, initializing all weights to zero and a learning rate of 0.001

   (d) Compute the accuracies on both train and test sets

   (e) Use `scikit-learn`'s implementation of multi-class logistic regression and compare the resulting accuracies.