Machine Learning - 2020-2021
Universitat Pompeu Fabra

Problems 7: Logistic Regression and SoftMax

Goal

The goal of this practice is to understand two different classifiers, one for binary classification and the other for multiclass. This two classifiers, that does not allow closed form solutions take they main role in Neural Network. We begin with a general linear classifier. We assume $\mathbf{w} = (w_0, w_1 \ldots, w_D)^T$ and $\mathbf{x}^{(n)} = (1, x_1^{(n)}, \ldots, x_D^{(n)})^T$ when necessary.

- *LS Linear Classifier*:

Model	$h_{\mathbf{w}}(\mathbf{x})$	$= \text{sign}(\mathbf{w}^T\mathbf{x})$
Least Square Error	$\mathbb{E}(\mathbf{w})$	$= \frac{1}{2}\sum_{n=1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{w})}{\partial w_j}$	$= \sum_{n=1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)x_j^{(n)}$

- *Logistic Regression*:

Model	$h_{\mathbf{w}}(\mathbf{x})$	$=P(\mathcal{C}_1\|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ where $\sigma(a) = \frac{1}{1+e^{-a}}$ is the *Sigmoid* or *Logistic* function
Cross-entropy error	$\mathbb{E}(\mathbf{w})$	$= -\sum_{n=1}^{N}\left(y^{(n)}\ln h_{\mathbf{w}}(\mathbf{x}^{(n)}) + (1-y^{(n)})\ln(1-h_{\mathbf{w}}(\mathbf{x}^{(n)}))\right)$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{w})}{\partial w_j}$	$=\sum_{1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)x_j^{(n)}$

- *Softmax function*:

Model	$h_k(\mathbf{x})$	$=P(\mathcal{C}_k\|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}}}$
Cross-entropy error	$\mathbb{E}(\mathbf{w}_1, \ldots, \mathbf{w}_K)$	$= -\sum_{n=1}^{N}\left(y_1^{(n)}\ln h_1^{(n)} + \ldots, y_K^{(n)}\ln h_K^{(n)}\right) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)}\ln h_k^{(n)}$ where $h_k^{(n)} = h_k(\mathbf{x}^{(n)})$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{w}_1,\ldots,\mathbf{w}_K)}{\partial \mathbf{w}_j}$	$=\sum_{n=1}^{N}\left(h_j^{(n)} - y_j^{(n)}\right)x_j^{(n)}$

Exercises

Linear and Logistic classification

1. Consider two classes: the class $C_1 = \{\mathbf{x}^{(1)} = \binom{1}{2}\}$ with a label 1 and the class $C_2 = \{\mathbf{x}^{(2)} = \binom{-1}{-2}, \mathbf{x}^{(3)} = \binom{-2}{-1}\}$ with label -1.

   Consider a **linear classifier** $h_{\mathbf{w}}(\mathbf{x})$.

   (a) Plot the points, write the parametric form of $h_{\mathbf{w}}(\mathbf{x})$ and guess -and draw- what could be a good solution.

   (b) Write the error function (least square cost) that is minimised when estimating the linear classifier that separates the given data.

   (c) Find the closed-form solution for the linear classifier that separates both classes. To do so, derive the expression of the error from the previous exercise with respect to the weight vector and set it to 0.

   (d) Estimate the linear classifier using the closed-form solution found in the previous exercise. Draw the data points and the resulting linear classifier in the same figure.

2. Consider two classes: the class $C_1 = \{\mathbf{x}^{(1)} = \binom{1}{2}\}$ with a label 1 and the class $C_2 = \{\mathbf{x}^{(2)} = \binom{-1}{-2}, \mathbf{x}^{(3)} = \binom{-2}{-1}\}$ with label -1.

   (a) Plot the points and write the parametric form of a **logistic classifier** $h_{\mathbf{w}}(\mathbf{x})$. What is the difference with a linear classifier?

   (b) Labels 1 and -1 are not adequate when solving a two-class problem with logistic regression. Why? Which labels should we choose?

(c) Write the error function that is minimised when estimating the logistic classifier that separates the given data and develop it.

(d) (Optional) Derive the expression of the error from the previous exercise with respect to the weight vector.

(e) (Jupyter) Generate a gradient descend algorithm with ridge regularisation. To do so, you only have to add to the gradient an extra term λw, where λ is the regularisation parameter. Plot the data points and the resulting linear classifier in the same figure.

(f) (Jupyter, optional) Estimate the logistic classifier using the function sklearn.linear_model.LogisticRegression. Draw the data points and the resulting linear classifier in the same figure.

3. Consider three classes:
Class 0: $\{\mathbf{x}^{(1)} = \binom{0.5}{0.4}\mathbf{x}^{(2)} = \binom{0.8}{0.3}, \mathbf{x}^{(3)} = \binom{0.3}{0.8}\}$
Class 1: $\{\mathbf{x}^{(4)} = \binom{-0.4}{0.3}, \mathbf{x}^{(5)} = \binom{-0.3}{0.7}, \mathbf{x}^{(6)} = \binom{0.7}{0.2}\}$
Class 2: $\{\mathbf{x}^{(7)} = \binom{0.7}{-0.4}, \mathbf{x}^{(8)} = \binom{0.5}{-0.6}, \mathbf{x}^{(9)} = \binom{-0.4}{-0.5}\}$

And assume that the values obtained for the three discriminants are: $\mathbf{w}_0 = \begin{pmatrix} -0.3 \\ 0.87 \\ 1.47 \end{pmatrix}$, $\mathbf{w}_1 = \begin{pmatrix} -0.01 \\ 0.58 \\ 1.02 \end{pmatrix}$ and $\mathbf{w}_3 = \begin{pmatrix} 0.43 \\ -1.90 \\ 0.33 \end{pmatrix}$

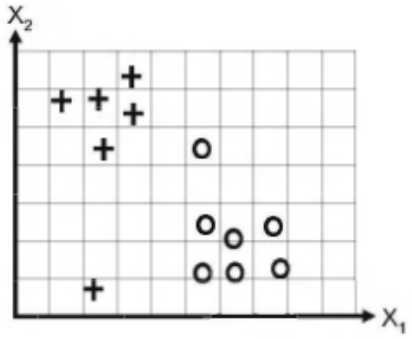
(a) Draw the points and codify the class of the vectors according to 1 of K coding.

(b) Determine at which class the calculated softmax will classify the points $x^{(1)}$, $x^{(4)}$ and $x^{(7)}$.

(c) Calculate and plot the discriminating surfaces.

(d) Calculate the error for the given solutions.

4. We will work with the data of the figure:



(a) (Jupyter) We will fit the model $p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(w_0 + w_1x_1 + w_2x_2)$ by minimizing the Cross-Entropy error $\mathbb{E}(\mathbf{w})$. For the solution you have obtained, How many points are wrong classified?

(b) (Jupyter) Now suppose we regularize only the w_0 parameter, i.e, we minimize

$$\mathbb{E}_T(\mathbf{w}) = \mathbb{E}(\mathbf{w}) + \lambda w_0^2$$

Use the Gradient Descent to estimate the classifier. Comment the results.