TP: Perceptron, Logistic and Softmax regression

Wednesday 27th September, 2023
joao.candido@hesge.ch
Deadline: Wednesday 18th October, 2023, 23:59

This TP is divided in to two parts:

1. In the first part, the goal is to understand and implement all the aspects of the perceptron, which means understand and implement the forward step, the loss, compute the gradients and update the weights.
2. In the second part, the goal is to understand and implement the Logistic regression and the Softmax classifier with an $L2$ norm regularizer.

1 Perceptron

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector [1].

The Perceptron is a two class algorithm, with class labels $c_1 = -1$, $c_2 = 1$. This algorithm will find a set of weights only if the problem is linearly separable. When the problem is not linear separable the perceptron is the non-convergence. We will consider a set of data X with length n and d attributs, where $\mathbf{x_i}$ for $i := 1...n$ and $\mathbf{x_i} \in R^d$ is a instance of X, that comes with a label $c \in \{c_1, c_2\}$ in the form of a vector :

$$\mathbf{x_i} = [x_{i1}, x_{i2}, ..., x_{id}]$$

For notation convinience we will consider that $\mathbf{x_i}$ is the augmented $d+1$ dimensional vector, where the $d+1$ dimension corresponds to the bias term and is set to 1, thus:

$$\mathbf{x_i} = [x_{i1}, x_{i2}, ..., x_{id}, 1]$$

[1] More information about perceptron here.

Remember that perceptron learns a set of weights

$$\mathbf{w_t} = (w1, ..., wd, w_{d+1})$$

and using these weights produces the thresholded output

$$sign(\mathbf{x_i w_t})$$

The error function of perceptron is:

$$E(\mathbf{w}) = -\sum_{\mathbf{x_i} \in M} (\mathbf{x}_i \mathbf{w}) y_i$$

where M is the set of missclassified instances and y_i is the class associated to the sample i.

The gradient of $E(\mathbf{w})$ is:

$$\nabla E(\mathbf{w}) = \sum_{\mathbf{x_i} \in M} \mathbf{x_i} y_i = -\mathbf{x}^T (y - sign(\mathbf{xw}))$$

The gradient descent direction is $-\nabla E(\mathbf{w}) = \mathbf{x}^T (y - sign(\mathbf{xw}))$ and we update the weight matrix $\mathbf{w}$ in the gradient descent direction by $\Delta \mathbf{w} = -\alpha \nabla E(\mathbf{w})$, thus:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w} = \mathbf{w} + \alpha \mathbf{x}^T (y - sign(\mathbf{xw}))$$

where α is the learning rate.

2 Logistic regression

1. The forward step, i.e. the computation of the scores.
2. The cost function of the logistic regression.
   (a) Write down the cost function of the logistic regression on your report.
   (b) Derive the gradient of the logistic regression cost with respect to the weights (learning parameters).
   (c) Write down the cost function of the logistic regression when a L_2 regularizer is added.
   (d) Derive the the gradient of the logistic regression cost when a L_2 regularizer is added.
3. Based on your derivations, implement the loss function in the logistic regression class and its derivative with an L_2 regularizer. **Reminder :** All the formulas and derivations (i.e. how do you get a given formula) must be in the report ! Otherwise the code it's not taken into account.
   (a) Fill the *scores* part in loss function inside the logistic_regression.py
   (b) Fill the *loss* part in loss function inside the logistic_regression.py

(c) Fill the *grads* part in loss function inside the logistic_regression.py

4. Implement Stochastic Gradient Descent, SGD

   (a) Fill the missing part of the train() method inside the classifier.py script

5. Fill the missing part in TP7_logistic_softmax_classifiers.ipynb notebook and train your classifier for different learning rates and regularization strengths.

   (a) Comment **in details** how the different learning rates and regularization strengths influence the performance of the classifier. Which is the effect of very large/small regularizers? How the learning rate change the prediction? Which (and why) is the optimal why to update the weights? etc..

2.1 Softmax classifier

Follow exactly the same thing as before (logistic regression) but with softmax classifier. All the steps, formulas derivations have to be in report to make your code count !

3 Reminders

- The Softmax classifier is the generalization of the binary Logistic Regression classifier to multiple classes. The Softmax classifier gives as output normalized class probabilities
- When we minimize the cost function using Gradient Descent (GD) the weights are updated after seeing all the training instances
- When we minimize the cost function using Stachastic Gradient Descent (SGD) the weights are updated after seeing a mini batch the training instances.
- The L2 regularizer is also called Ridge Regression. It adds "squared magnitude" of weights as penalty term to the cost function. If the regularizer parameter is zero then you can imagine we get back to the cost function.

General instructions

You have to put your work in *cyberlearn* saved in a zip file using as name this format: TP_PLS_LASTNAME_Firstname. This work should be accompanied by a pdf report. Explain the problem and discuss the results, it should be a summary of the main findings, factual, clear, and concise. It should **also** have the answers to the questions asked in the notobook.