Practical Work 02 – 4/3/2021
Gradient Descent

Objectives

The main objectives of this Practical Work for Week 2 are the following :

a) Get more experienced in python, particularly with numpy.

b) Refresh or deepen your maths skills (calculus)

c) Implement gradient descent for the perceptron model and then apply it to MNIST. Study some of the main features.

Submission

— **Deadline** : Wednesday 17 March, 10am

— **Format** :

  — Exercise 1 (Numpy) : Optional

    — Jupyter notebook `numpy-tutorial-stud.ipynb` completed with your solutions.

  — Exercise 2 (Sigmoid Function) :

    — Jupyter Notebook with the sigmoid function and the plot (incl. derivative).

    — Maths calculations either in a pdf with your handwritten solutions or all calculations all in the Jupyter Notebook (by using latex). Don't just state the final results but expand on how you have obtained them.

  — Exercise 3 (Gradient Descent Implementation) :

    — Jupyter Notebook `MNIST_Binary_batchgd_stud.ipynb` completed with your solutions.

    — The answers to the questions either in a small pdf-report or in the Jupyter Notebook.

Exercise 1 Optional : Numpy in a Nutshell

This exercise is to become more familiar with `numpy`. Read the content of the jupyter notebook `numpy-tutorial-stud.ipynb` that you will find on Moodle. Pay special attention to the *broadcasting* section that allows to gain significant speedup when processing large numpy arrays. Regarding this, it is usually more efficient to use *broadcasting* instead of for loops in your code.

At the end of the tutorial, you have to complete some manipulations of images stored by arrays.

Exercise 2 Sigmoid Function

(a) Compute the derivative of the sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

(b) Show that the derivative fullfills the equation

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$$

(c) Compute the first and second derivative of

$$\zeta(z) = -\log(\sigma(-z)) \qquad \log\ : \text{Natural Logarithm}$$

Compute the asymptotes for $z \to \pm\infty$. Create a plot of ζ.
(Hint : For $z \to +\infty$ you may consider the limit of the function $\zeta(z)/z$.)
Remark : This function is also referred to as softplus, a smooth alternative of $x^+ = \max(0, x)$, the *rectifier*.

(d) Implement the sigmoid function in a Jupyter Notebook. Make it work such that you can pass numpy arrays of arbitrary shape and the function is applied element-wise. Plot the sigmoid function and its derivative by using matplotlib.

(f) Show that the function

$$c_1(x) = (\sigma(x) - 1)^2$$

is non-convex.
Explain in which situations (initial settings) optimising $c_1(x)$ with gradient descent may become difficult. For the explanation create a plot. Note that this exercise should give an intuition on why mean-square error loss is less suited for classification problems.

(g) Compute the first and second derivative of the function

$$c_2(x) = -\left(y \log(\sigma(w \cdot x)) + (1 - y) \log(1 - \sigma(w \cdot x))\right)$$

with respect to $w \in \mathbb{R}$ and for given $y \in \{0, 1\}$. Show that c_2 is convex for $x \neq 0$.

Exercise 3 Gradient Descent for Perceptron

Implement gradient descent learning for the single perceptron and analyse the results. Do this on the basis of the Jupyter Notebook `MNIST_Binary_batchgd_stud.ipynb`. Only use numpy functionality (scikit learn only for loading the data and splitting into train and test sets). The sections of code that you need to implement are marked (as usual) with

START YOUR CODE

END YOUR CODE

Proceed as follows :

(a) Work your way through the preprocessing steps in the notebook consisting of
  - loading the data
  - filtering the data for the digits of interest (e.g. '1' and '7') so that a binary classification problem is obtained
  - splitting the data into a train and a test set
  - standardise the data (z-normalisation)

(b) Implement the training with the the model function (sigmoid), the cost function (CE and MSE), associated update rules (`step_CE` and `step_MSE`) and the `optimise` function. Make sure that your implementation works for images of arbitrary shapes.

(c) Run the training by optimising on cross-entropy cost and plot the learning curves : Cost, Error Rate, Learning Speed and convince yourself that you obtain curves similar as seen in the class.

(d) Analyse the dependency of the final error rate on the learning rate, the number of epochs, the type of cost.

(g) Optional : Learn binary classification for all the pairs of digits (using reasonable values for the learning rate and the number of epochs). Compare the finally obtained error rates. For which pairs of digits works the classification specifically well - for which rather bad ? Interpret these findings. Are some of the problems linearly separable ? How can you recognise that ?

See the comments and instructions in the notebook for further details.

Hints :

- Keep an eye on the shapes of the arrays (as passed into the functions and as used within the functions (and declared in the function descriptions).
- Run the tests (kind of unit tests) marked with `## TEST ##` which should not raise an Assertion Error.
- Possibly, add `assert()` statements in the code.

Exercise 4 Optional : Review Questions

a) Explain why normalisation is beneficial.

b) In what sense are optimisation techniques important for machine learning problems ?

c) Describe what problems gradient descent can be applied to. In what problems will gradient descent lead to a unique solution ? Describe what can go wrong in more general problems and why.

d) Why is learning with MSE cost considered less suitable for classification problems ?

e) What may happen if the learning rate is chosen too large ?

f) Why is it possible that the test error starts increasing after some epochs of training ?

g) Is it becoming more or less difficult to reach small values of the cost function if we have more training data ?

Exercise 5 Optional : Reading Assignment

Read the start of the Section 4.3 on *Gradient-Based Optimization* in the *Deep Learning* book by Ian Goodfellow et al (see https ://www.deeplearningbook.org/contents/numerical.html, p.80-84).