Teachers: J. Hennebert, M. Melchior
C. Gisler, P. Matvulj, O. Zayene

Practical Work 02 – 29/02/2024
Gradient Descent

Objectives

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

a) Get more experienced in python, numpy and pytorch.

b) Learn the gradient descent technique.

c) Use gradient descent for learning linear regression based on the example of PW01 - all using pytorch tensors.

d) Use gradient descent for learning multinomial logistic regression for MNIST classification, by using pytorch but with manually calculated gradients.

e) Use gradient descent for learning multinomial logistic regression for MNIST classification, with full-fledge pytorch (using autograd).

f) Study the behavior of gradient descent under different learning rates and batch sizes.

g) (Optional) Refresh or deepen some of your maths skills (calculus).

Submission

— **Deadline** : Wednesday 13 March, noon

— **Format** :

For Exercises 1,2,3 : jupyter notebooks completed with your solutions and answers.

For Exercise 4,6 (optional) : pdf with your maths calculations.

In the different notebooks, you will find sections to complete marked with
# YOUR CODE (START) #

# YOUR CODE (END) #

Exercise 1 Gradient Descent (GD) for Linear Regression

Implement gradient descent learning for the linear regression example of The Jupyter Notebook `pw02_lin_regression_GD_stud.ipynb` will guide you through the steps to accomplish this task, including

— loading and normalising the data

— reproducing the closed-form solution from PW01.

— implementing (full batch) GD for linear regression without autograd

— playing with different learning rates

— implementing (full batch) GD for linear regression with autograd

Exercise 2 Binary Classification with Logistic Regression

Implement (binary) logistic regression to classify the spine dataset (downloadable from kaggle) into 'normal' or 'abnormal' and apply gradient descent for training the model by completing `pw02_binary_classification_GD_stud.ipynb`.

This includes the steps :

— loading and normalising the data

— implementing the logistic regression model

— implementing (full batch) GD for linear regression without and with autograd

— playing with different learning rates

Remark : For two class classification you don't need the softmax function, but just the sigmoid function that outputs the probability of observing the class with label 1́.́ In principle, you could also use *softmax* with two classes. Nevertheless, we think that it is instructive to go through and implement the formulas for standard binary logistic regression with the sigmoid function.

Exercise 3 Gradient Descent for MNIST Classification

Implement gradient descent for multinomial logistic regression for MNIST. Here, the Jupyter Notebook `pw02_softmax_classification_stud.ipynb` will guide you through the steps to achieve this.

Proceed as follows :

— Load the MNIST dataset (train and test) and depict some samples. Make sure that only the training set will be used for training and the test dataset just for evaluation (accuracy).

— Implement the multinomial logistic regression model by completing the functions `linear_trsf`, `softmax`, `predict`, `loss_ce`, `cost_ce`.

— Implement the mini-batch gradient descent training loop configured by number of epochs, batch size and learning rate. Implement the gradient (without autograd). Train the model with `nepochs=10`, `nbatch=64` and `lr=0.01`.

— Tune the learning rate and batch size and qualitatively characterise the behaviour. What is your favorite combination (learning rate, batch size, number of epochs) ?

— Build and train the model with full-fledge pytorch incl. autograd. Implement a model class inheriting from *torch.nn.Module* and by using the layer functionality in the package *torch.nn* (see lecture). Prove that you obtain the same accuracy.

Hints :

— Keep an eye on the shapes of the tensors (as passed into the functions and as used within the functions (and declared in the function descriptions).

— For each implemented function, run a small test with dummy input tensors (of the required shape) and check whether it outputs a tensor with expected shape and no 'nan'.

— Possibly, add `assert()` statements in the code.

— For the training loop perform a training with only a single batch of size 64. Here, you should be able to obtain 100% training accuracy.

Exercise 4 Optional : Review Questions

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

b) 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.

c) Explain why is learning with MSE cost considered less suitable for classification problems ?

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

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

f) 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).

Exercise 6 Optional : Calculus Training

(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 derivatives of the function

$$-\log(\sigma(xW^{\mathrm{T}} + b))$$

with respect to W and b where σ is the sigmoid function, W and x are $(1, n)$-arrays and b a scalar.

(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 $x \neq 0$.