Teachers: J. Hennebert, M. Melchior
Assistants: C. Gisler, M. Willi, Y.-I. Beffa

Practical Work 03 – 11/3/2021
Shallow Networks

Objectives

The main objectives of this Practical Work for Week 3 are the following :

a) Implement MBGD and Softmax and learn what it means to choose hyper-parameters such as learning rate, batch size or number of epochs. Also play with different initialisers and evaluate their impact on the learning curves.

b) Analyse the error rates and the trained weights.

c) Deepen your understanding of the Universal Representation Theorem.

Submission

— **Deadline** : Wednesday 24 March, 10pm

— **Format** :

  — Exercise 1 : MNIST Classification with Softmax

    Jupyter notebook `PW_classifier_softmax_stud.ipynb` completed with your solution and comments.

  — Exercise 2 : Universal Approximation Theorem (Optional)

    Jupyter notebook `function_approximation_sigmoid_stud.ipynb` completed with your solutions - including the formulas for the gradient of the MSE cost.

Exercise 1 MNIST Classification with Shallow Network

Implement Mini-Batch Gradient Descent for Softmax. Do this on the basis of the Jupyter notebook `PW_classifier_softmax_stud.ipynb`. As in PW 02, do this by only using numpy functionality (scikit learn used only for loading the data and splitting it into train and test sets). Look out for the suitably marked sections that you need to implement.

Note that you will train softmax with the whole MNIST dataset. This will take more CPU and RAM, so that you need to be more careful in efficiently implementing the code. Make sure to properly use numpy array arithmetics! All the training runs should complete in at most a couple of minutes.

Follow the instructions in the notebook to do the following tasks :

(a) Implement the functions to prepare the data (very similar to PW02), softmax, the entropy cost for multiclass and its gradient, the optimisation loop (with given function signature) by using the classes *Metrics* and *MiniBatches*.

(b) Run several trainings with different hyper-parameter settings and determine your favorite setting (learning rate, batch size, number of epochs).

(c) Compute the Error Rates for the individual Digits.

(d) Analyze misclassified images that have worst score.

(e) Plot the weights as images. Answer the additional questions regarding the bias.

(f) Analyze different weights initialisation strategies (all weights set to 0, standard normally distributed weights, scaled normally distributed weights). Which strategy works best? How do you judge this? Explain why the your selection works best.

Exercise 2 Function Approximation [Optional]

In this exercise, you train a single (hidden) layer neural net to represent a given function $f : [0, 1] \to \mathbb{R}$. Since it is a regression problem, we will use the MSE cost function.

The MSE cost for a neural net with a 1d input x, a single hidden layer with n units and a linear output layer is given by

$$J_{\text{MSE}}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - (\sum_{k=1}^{n} w_{2,k} \sigma(w_{1,k} \cdot x^{(i)} + b_{1,k}) + b_2) \right)^2 \tag{1}$$

The dataset is given by suitable x-values (in the interval $[0, 1]$) and associated function values $f(x)$, i.e. $\{(x^{(i)}, y^{(i)} = f(x^{(i)})) \,|\, i = 1, \ldots, m\}$.

a) Compute the formulas for gradient descent for this problem, i.e. compute the derivatives w.r.t. parameters $w_{1,k}, w_{2,k}, b_{1,k}, b_2$ and formulate the according update rules.

b) Implement MBGD for this model in the notebook `function_approximation_stud.ipynb`. Apply input and output normalisation. With the settings provided in the notebook (learning rate, batchsize, etc.) and the given dataset generated for the Beta-function (with

$\alpha = \beta = 2.0$ and $m = 1000$ samples) the learning should work quite well - see the learning curve (cost vs epochs) and the final MSE cost ($J_{\text{MSE}}(\theta_{\text{trained}}) \approx 4 \times 10^{-4}$).

You can now also try with different functions (such as the sine function with different frequencies). Start with a frequency $\omega = 1$ (ie. one cycle in the interval $[0, 1]$). Then proceed and investigate how large the (integer) frequency can be chosen to still obtain reasonable approximations. Possibly, also consider generating a larger dataset. Give an interpretation for why the learning breaks down at larger frequencies.

Exercise 3 Review Questions [Optional]

a) What is the purpose of the softmax layer ? Where is it typically used ? When is it the method of choice preferred over solving m binary classification problems (one for each of the classes) ?

b) Mention indicators for the situation where the training has not yet converged.

c) What are the factors that drive the performance of MNIST classification with a single (hidden) layer perceptron.

d) What kind of mappings can be represented by neural networks with just linear activation functions ?

e) Describe what statement is made by the Universal Approximation Theorem.

f) Describe an approximation scheme for a given $1d$ function.

g) Describe why the Universal Approximation Theorem is only of limited value in practice.