Practical Work 08 – 14/03/2022
Batchnorm and Parameter Initialisers

Objectives

The main objective of this PW is to explore the impact of parameter initialisation and batch normalisation on the speed of learning NNs. You can carry through this practical work with your framework of choice (PyTorch or Tensorflow/Keras).

Submission

— **Deadline** : Wednesdays 27th April, noon

— **Format** : ipynb

Exercise 1 Parameter Initialisation

Create a Jupyter Notebook (named `eval_param_init_<group name>.ipynb`) in which you conduct the following steps :

a) Implement a model for classifying Fashion-MNIST. Use at least (in total) 5 hidden conv or fully connected layers with tanh non-linearity. Use the defaults for parameter initialisation (i.e. don't specify anything for initialising weights and biases).

b) Use this as a baseline and train the model over (at least) 10 epochs (by using Adam with default settings). Remember the training/validation curves (cost, accuracy). Use input data normalisation.

c) Learn how you can change the weights and bias initialisers for MLP and CNN layers and for your model in your framework of choice.

d) Now play with other initialisers. Train the model for each setting over (at least) 10 epochs and remember the resulting training /validation curves.

— zero weight and zero bias

— standard normal weights and zero bias

— uniform weights ($\sim \mathcal{U}(-1/2, 1/2)$) and zero bias.

— Xavier Glorot initialization (for tanh) with zero bias.

— Kaiming He initialization (for ReLU) with zero bias.

e) Create a comparison plot and discuss the results. Are they as expected ?

Exercise 2 Benefit of BatchNorm with CIFAR10 Classification

Create a Jupyter Notebook (named `eval_batchnorm_<group name>.ipynb`) in which you conduct the following steps :

a) Implement a mixed CNN/ML architecture (at least 3 conv and at least 2 fully connected hidden layers, ReLU) for classifying CIFAR10 images - without any batchnorm layers nor regularisation. To give an example, this may look as follows : (Conv2d, ReLU, MaxPool), (Conv2d, ReLU, MaxPool), (Linear, ReLU), (Linear,ReLU), Linear.

b) Train this over a suitable number of epochs until you see stable test performance or before you observe overfitting (by using Adam with default settings). Use this as baseline and remember train and test curves (cost, accuracy) for later reference.

c) Do the same (with the same number of epochs as used above), but now using tanh instead of ReLU.

d) Now add batchnorm after each Conv2d and Linear layer (before the non-linearity). Again perform the training (over the same number of epochs). Do this twice : with ReLU and Tanh.

e) Now study the impact when adding dropout regularisation before each fully connected layer (not CNN). Do this for the architecture without and with batchnorm. Perform according trainings. How far can you bring up the test accuracy by continuing the training possibly over more epochs ?

f) Create one or several comparisons plots with the learning curves with/without batchnorm also under the different other settings : with/without regularisation and with ReLU or Tanh. Estimate a factor of speedup when using batchnorm (with/without reg, with ReLU or Tanh). Discuss your findings and make a statement about whether the results are as you expect.