**Wednesday Morning Exercises**

The Exercises below can be done on a personal computer, each training run should take couple of seconds to a minute.

**Exercise 1.**

Perceptron (linear NN) for the Iris dataset, using Mean Squared Error as risk. Analysis of the behavior of training risk and accuracy for different learning rates.

Detailed steps:

a) Use pandas to load the iris dataset. Create dummy variables for the classes

b) Define pytorch tensors for the dataset using:

torch.tensor

c) Define pytorch tensors (with gradient) for weights and biases (W & b). W should be n\_features x n\_classes, b should be 1 x n\_classes. Initialize b to zeros (torch.zeros), and W to random values sampled from a normal distribution with null mean – try different values for the standard deviation and observe changes in the training behavior.

d) Define pytorch optimizer over variables W & b

torch.optim.SGD or torch.optim.Adam

e) Create the main loop that goes over the dataset in multiple epochs. In each epoch

e1) clear gradients (using optimizer.zero\_grad)

e2) calculate linear predictions: pred=X W + b using

torch.matmul

e3) pass the linear predictions through the unipolar sigmoid: sigmoid(pred)=1/(1+exp(-pred)). Use these functions:

torch.log, torch.exp

e4) calculate the squared difference between the predictions (after sigmoid) and the true classes, for all three output neurons. Use:

torch.pow

e5) calculate risk = average the squared difference over the training samples. Use:

torch.mean

e6) calculate gradients of risk with respect to W & b (call risk.backwards)

e7) make optimizer step (using optimizer.step)

e8) calculate accuracy

Experiment with different learning rates for the two optimizers and observe the behavior of the training loss and accuracy.

**Exercise 2.**

Perceptron (linear NN) for the Iris dataset, using CrossEntropy as risk. Analysis of the behavior of training risk and accuracy for different learning rates.

Detailed steps - follow Exercise 1, but replace MSE with CrossEntropy:

e3) pass the linear predictions through softmax (i.e., normalize the unipolar sigmoids for classes i=1,...,3 to sum up to 1 for each sample)

e4) calculate the cross entropy after softmax (sum\_{i=1}^3 y\_i ln(softmax\_i)).

torch.multiply, torch.log, torch.sum

e5) calculate risk = average the cross entropy over the training samples

Experiment as in Exercise 1.

**Exercise 3.**

Starting from Exercise 2, add a split of the Iris dataset into a training set and a test set. Also, in the training loop, go over small batches of samples (e.g. 20 samples) instead of always over the whole training set. Experiment with batch size and learning rate.

**Exercise 4:**

Perceptron (linear NN) for MNIST Digits dataset. Explore the behavior of the code from Exercise 3 on a larger, more complicated dataset.

The number of training samples is 50,000 - analyze training behavior if a random subset of 100, 500, 1000, 2000 samples is used instead. Also, experiment with the learning rate and the batch size.

For loading the dataset, use:

full\_train\_dataset = datasets.MNIST(root='./data', train=True, download=True, transform=None)

full\_test\_dataset = datasets.MNIST(root='./data', train=False, download=True, transform=None)

x\_train = full\_train\_dataset.data.numpy().reshape(-1,n\_features).astype(dtype=np.float)/255.0;

x\_test = full\_test\_dataset.data.numpy().reshape(-1,n\_features).astype(dtype=np.float)/255.0;

y\_train\_cat = full\_train\_dataset.targets.numpy()

y\_test\_cat = full\_test\_dataset.targets.numpy()

Note that the download of the dataset may take long, start the download early. As with Iris, convert categorical variables for classes into dummy variables (there are 10 classes).

**Exercise 5.**

Using the same MNIST dataset as in Exercise 4, add a hidden layer to the neural network. Using the same dataset as in Exercise 4, add a hidden layer to the neural network.

You will need weights W1 & b1 for the first layer, and W2 & b2 for the second layer.

Use bipolar sigmoid activation after the first layer: bipoloar(x)=2/(1+exp(-x))-1

Alternatively, use ReLU activation function: relu(x)=max(0,x)

Explore training and test set accuracy for different learning rates, different std.dev. during initialization of W1 and W2, different batch sizes, different optimizers, and different activation functions.