CSC 2515 FAQ for A2

Shenlong Wang
slwang@cs.toronto.edu

Please post a thread on piazza.com/utoronto.ca/winter2015/csc2515 **if you have more questions. I will try to answer them on a daily basis.**

1. **Q: What exactly is** wts, b, wts_grad, wts_inc, b_grad, b_inc, **and** input_grad **supposed to be?**

   A: Please see the following:

   - wts: weights for each layer.
   - b: bias for each layer.
   - wts_grad: gradient for weights you calculated from back_prop for each layer.
   - wts_inc: actual update you will do for wts in a SGD step for each layer.
   - b_grad: gradient for bias you calculated from back_prop for each layer.
   - b_inc: actual update you will do for b in a SGD step for each layer.
   - act_grad in @layer/back_prop: gradient wrt activation function of this layer.
   - input_grad in @layer/back_prop : gradients wrt the input of this layer.

2. **Q: What do you expect us to implement?**

   A: In general your task is to

   1. implement the following functions:
      - @nn/fwd_prop: perform a feedforward pass over all layers, and return a list of (cell in matlab) activations for each layer. You may want to call @layer/fwd_prop for each layer.
      - @nn/back_prop: perform a backpropagation, return self with updated gradient of weights and biases for all layers. You may want to call @layer/back_prop for each layer.
      - @nn/apply_gradients: perform stochastic gradient descent step. You may want to call @layer/apply_gradients for each layer.

   - `@layer/fwd_prop`:perform a forward pass.
   - `@layer/back_prop`: back propagate activation gradients and compute gradients for one layer. The output is a struct consisting of three parts, `wts_grad`, `b_grad`, `input_grad`. Please refer to Q1 for their meaning.
   - `@layer/compute_act_gradients_from_targets`: compute the gradients wrt activations of *sigmoid* layer, the input are the current activations of this layer and the gradients wrt outputs of the sigmoid. This funcion is needed for *sigmoid* layer.
   - `@layer/compute_act_grad_from_output_grad`: compute the gradients wrt activations of the *softmax* layer, given the targets and the outputs of the *softmax*, the inputs are the current activations of this layer and the target.
   - `@layer/apply_gradients`: update `wts_inc`(`b_inc`) and use `wts_inc`(`b_inc`) to update the weight(bias). You may want to use the gradient `wts_grad`(`b_grad`) as well as momentum and learning rate.

2. choose proper hyper-parameters in `train_nnet`:
   - `eps`, learning rate for SGD
   - `l2`, coefficient for ℓ_2 regularizer
   - `momentum`, momentum for SGD
   - `batch_size` , batch size for SGD
3. implement stopping criterion in `train_nnet`. Consider how to control overfit.
4. apply pre-processing in `creat_pred` and `train_nnet`. You may want to use functions implemented in `speech_data`.

3. **Q: Do we have to use both *svd* and *eig* in order to calculate PCA?**

   A: No. You may use either one of them. Try to think about the difference between two approaches.