Princeton COS 495: Introduction to Deep Learning
Homework 2

Lecturer: Yingyu Liang — Due date: Mar 30th, 2016

TA: Bochao Wang — Office: Electrical Engineering Department, C319B

---

1. (Math) Recall that when ℓ_1 regularization is used, and assuming that the Hessian matrix is diagonal with positive entries, the objective function can be approximated by

$$\hat{L}_R(\theta) := \sum_{i=1}^{d} \left[\frac{1}{2} H_{ii}(\theta_i - \theta_i^*)^2 + \alpha|\theta_i|\right]$$

Solve this in the close form expression: show that the optimal solution θ_R^* for the objective $\hat{L}_R(\theta)$ is that as shown on slide 26 of "Deep learning basics lecture 3: Regularization I". Hint: note that $\alpha/H_{ii} > 0$.

2. (Math) Consider a three layer network:

$$h^1 = \sigma(W^1 x), h^2 = \sigma(W^2 h^1), f(x) = \langle w^3, h^2 \rangle.$$

See Figure 1 for an illustration. Compute $\frac{\partial f}{\partial W^1_{i,j}}$.

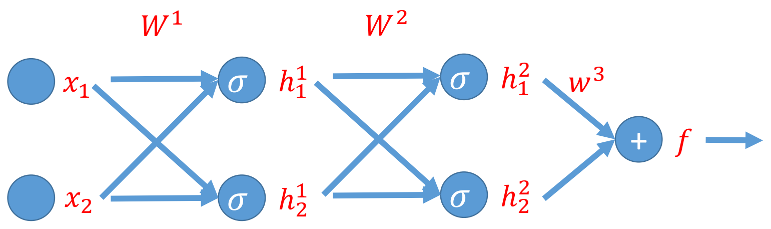

Figure 1: An illustration of the three layer network

3. (Coding) Choose a software framework. A few candidates:

- Marvin `http://marvin.is/`
- Tensorflow `https://www.tensorflow.org/`
- Caffe `http://caffe.berkeleyvision.org/`
- Pylearn2 `http://deeplearning.net/software/pylearn2/`

Read the tutorial, try it on MNIST, so you can reuse the data downloaded for the last homework. (Note that the first examples in the tutorials are typically on MNIST, so you can follow the steps.) More precisely, build a three layer feedforward network:

$$x \to h^1 \to h^2 \to p(y|h^2).$$

The hidden layers h^1 and h^2 have dimension 500. Train the network for 250 epochs[1] and test the classification error. Do not use regularizations. Plot the cross entropy loss on the batches and also plot the classification error on the validation data.
Comments: you can also use another data set, like CIFAR10 or CIFAR100. Or you can pick your own data set.

4. (Coding) Repeat the above experiment, but train the network with the following regularizations:

- ℓ_2 regularization
- Dropout
- Early stopping

Compare with the results in the previous experiment.
Comments: no need to implement them by your own; the software framework typically provides implementations for ℓ_2 regularization and dropout. Early stopping is done in training, so you only need to tune your training code slightly.

[1]Each epoch is a pass over the training data. Suppose you use batches of size b, and the training data set has n points, then an epoch consists of n/b batches. Note that you can divide the data set into batches, and then round robin over the batches. You can also randomly sample say 64 points for each batch. Either way is OK, and typically there is no performance difference between them. When these batches are randomly sampled, it is possible that some point are not in any of them, but we still call these batches a pass over the data.