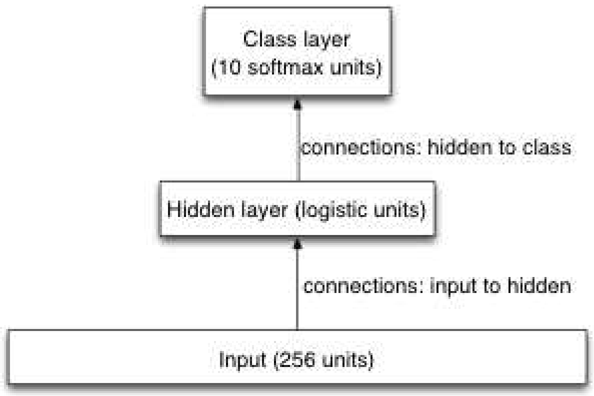
Assignment 1

In this assignment, you will train a simple multilayer neural net and investigate stochastic gradient algorithm and regularization.

Description of the dataset: The dataset for this assignment is the USPS collection of handwritten digits. It consists of scans (images) of digits that people wrote. The input is a 16 by 16 image of greyscale pixels, showing an image of a handwritten digit. The output is simply which of the 10 different digits it is, so we're using a 10-way softmax as the output layer of our Neural Network. The input layer is simply 256 units, i.e. one for each pixel. We use one hidden layer of logistic units. One of the issues we'll be investigating is what number of hidden units works best for generalization. To keep things as simple as possible, we're not including biases in our model.


Class layer (10 softmax units)
connections: hidden to class
Hidden layer (logistic units)
connections: input to hidden
Input (256 units)

To investigate generalization, we need a training set, a validation set, and a test set, so the dataset has been split in 3 groups. We train our networks on the training set; we use the validation set to find out what's generalizing well and what isn't; and after we've made our choice of regularization strategy, we'll see how well our model performs on the test set. Those three subsets have already been made for you, and you're not expected to change them (you're not even allowed to change them until Part 3). The full dataset has 11,000 images. We're using 1,000 of them as training data, another 1,000 as validation data, and the remaining 9,000 as test data. Normally, one would use most of the data as training data, but for this assignment we'll use less, so that our programs run more quickly.

In the lectures, we discussed about the regularized learning:

$$\min_{\theta} \tilde{J}(\theta) = J(\theta; X, y) + \alpha\Omega(\theta)$$

Here we're mostly interested in the cross-entropy error, as opposed to the classification error rate. so $J(\theta; X, y)$ is the *classification loss* (cross entropy), and $\Omega(\theta)$ is the *weight decay loss* $\Omega(\theta) = \frac{1}{2}\|W_{\text{hid}}\|_F^2 + \frac{1}{2}\|W_{\text{out}}\|_F^2$, where W_{hid} and W_{out} are the weights of hidden and output layer respectively; $\tilde{J}(\theta)$ is the (total) *loss*. In the code, we provide the training algorithm and auxiliary methods for you to begin with. In particular, we have provided a gradient check procedure so that you should make sure the computation of loss $\tilde{J}(\theta)$ and gradient $\nabla\tilde{J}(\theta)$ is correct.

Part 1

In the assignment1.py file, the "todo" part is explicitly left for you to fill in. Specifically, you will write forward-propagation and backward propagation which are in the eval_obj and eval_obj_grad functions respectively. In addition, you should take care of the cross entropy and softmax function, avoiding numerical issues such as underflow/overflow. Make sure that your code can pass gradient check procedure.

Part 2

The optimizer that we're using is gradient descent (with momentum), but we'll need to find good values for the learning rate and the momentum multiplier. Throughout this part, we set weight_decay=0 and early stopping=False.

Question1 (Learning rate)

Run train(0, 10, 70, 0.005, 0, False, 4) and what is the total loss?
Now let learning rate be chosen from $\{0.002, 0.01, 0.05, 0.2, 1.0, 5.0\}$. Find out the best learning rate on validation performance.

Question2 (Classical momentum)

Let hidden layer size =10, iteration number = 100. Try a variety of learning rates and momentum, learning rate is chosen from {0.01, 0.05, 0.2, 1.0, 5.0} and momentum is {0, 0.5, 0.9}. Hence, there are $6 \times 3 = 18$ different combinations and you need to run 18 different experiments. Run the code and find out the best combination on validation dataset.

Question3 (Nesterov accelerated method)

Implement Nesterov's algorithm and compare its performance with classical momentum based on the above parameter settings. Summarize your observations.

Part 3

In addition to optimization efficiency, we care about the generalization performance of the neural nets. We can improve the performance by tuning: a) weight decay 2) early stopping 3) number of hidden layers.

Question 1 (weight decay)

First disable the early stopping and set number of hidden units = 200, iteration number =1000, learning rate=0.2, try weight decay of value [0, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10], compare the validation loss and then find out the best weight decay.

Question 2 (hidden units, early stopping)

First set weight decay=0, iteration number =1000, and learning rate=0.2. Then, try the hidden unit size from $\{10,50,100,200,300\}$ when early stopping is on or off. Now we have 5×2 pairs of parameters. Find out which performs best.

Question 3 (Testing)

We have examined several regularization methods for improving the model performance. Now we need to decide the one with the best generalization performance. By saying generalization performance, we mean classification error on the testing data. To achieve that goal, choose a combination of parameters (e.g. weight decay, number of hidden units, early stopping) by model selection using the training/validation part. You can combine the training/validation sets and apply cross-validation if necessary. However, testing data is untouched until the last moment, which means you should only use it once when reporting the performance of the selected model.