Peer-graded Assignment: my1stNN

Submit by April 22, 11:59 PM PDT

Important Information
It is especially important to submit this assignment before the deadline, April 22, 11:59 PM PDT, because it must be graded by others. If you submit late, there may not be enough classmates around to review your work. This makes it difficult - and in some cases, impossible - to produce a grade. Submit on time to avoid these risks.

Instructions

My submission

Discussions

Goals:

1. Code a deep (with at least 1 hidden layer) neural network in tensorflow
2. Fit it on the train dataset, estimate quality on the test dataset
3. Plot the train loss and test loss as a function of the training iteration number

Long description

Your ultimate task for this part is to build your first neural network [almost] from scratch and pure tensorflow. This time you will have the same digit recognition problem as for the logistic regression assignment, but at a larger scale:

- images are now 28x28
- 10 different digits
- 50k samples

Note that you are not required to build 152-layer monsters here. A 2-layer (one hidden, one output) NN should already have an edge over logistic regression.

[bonus score] If you've already beaten logistic regression with a two-layer net, but enthusiasm still ain't gone, you can try improving the test accuracy even further! The milestones would be 95%/97.5%/98.5% accuracy on the test set.

Please use the preprocessed_mnist.py in week2 folder to load the data:

```
from preprocessed_mnist import load_dataset
X_train, y_train, X_val, y_val, X_test, y_test = load_dataset()
print(X_train.shape, y_train.shape)
import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow(X_train[0], cmap="Greys");
```

Help Center

There is a notebook with this boilerplate code for your convenience.

SPOILERS!

Recommended pipeline:

- Begin with logistic regression from the previous assignment to classify some number against others (e.g. zero vs nonzero)
- Generalize it to multiclass logistic regression. Either try to remember the week 1 lectures or google it.
- Instead of a weights vector you'll have to use a matrix with `shape=(features, classes)`
- softmax (exp over sum of exps) can implemented manually or as `tf.nn.softmax`
- probably better to use STOCHASTIC gradient descent (minibatch)
- in which case sample should probably be shuffled (or use random subsamples on each iteration)

Add a hidden layer. Now your logistic regression uses hidden neurons instead of inputs.

- Hidden layer uses the same math as output layer (ex-logistic regression), but uses some nonlinearity (sigmoid) instead of softmax
- You need to train both layers, not just output layer :)
- Do not initialize layers with zeros (due to symmetry effects). A gaussian noize with small sigma will do.
- 50 hidden neurons and a sigmoid nonlinearity will do for a start. Many ways to improve here.
- In an ideal case this totals to 2 .dot's, 1 softmax and 1 sigmoid

Review criteria **less**

1. The solution runs in the course environment. This is a must, for sake of the fellow learner, make you work not require additional effort to assess.
2. The solution runs in less than 5 minutes on the Coursera server (or a single CPU). There is no need to overengineer.
3. You correctly implement an MLP in tensorflow. Usage of external high-level libraries such as keras is forbidden.
4. The solution has a clear training part and an evaluation part where the accuracy on the test dataset is assessed.
5. You don't use the test dataset for training in any way. Do not use any datasets other than the one provided via week2/mnist.py
6. You have a plot with train loss and test losses as a function of the iteration number
7. To pass, the accuracy must be at least 95%, to get the full points, 98%

Help Center