Machine Learning Milestone

Location Independent Weather Forecasting

**Overview**:

**Data**:

**Algorithm**:

We have implemented a basic feed forward backpropagation neural network with the option of zero, one or two hidden layers. The number of neurons per layer is currently fixed to the number of features in the data set, but will later be set by a hyper-parameter. Using stochastic steepest descent, the error from the final output is backpropagated to adjust the weights into each neuron. The neurons on each of the hidden layers perform a non-linearity soft-threshold function (*tanh*). The output layer contains neurons equal to the number of output features. These neurons do not perform the threshold function since we want continuous output values.

The forward pass through the network functions by taking the input to a layer and then scaling by its respective weights to generate a signal. This signal is then passed through the non-linearity function to generate the input for the next layer (or the output). [[1]](#footnote-1)

where

Once the forward pass through all layers of the network is completed, the error is calculated using the predicted and actual values. This error is used to calculate a delta value which indicates how the weights for each layer should be adjusted. This final error is a function of the weight matrix that adjusts values into the neuron. Thus in order to compute the gradient, we want to derive in respect the weights of the current layer. This derivative can be solved using the chain rule for the two functions involved in the layer – the scaling by weights and the neuron function (which is just the linearity transfer function).

For the hidden layers, we cannot directly calculate the output error for that layer, but we can infer the error using the error gradient of the layer after it. Thus we can take the gradient value for the final layer and recursively calculate the gradient values for all the prior layers.

))

Our algorithm starts by loading the randomized data set and allocating subsets for various purposes. Half of the data (Train set) is used for stochastically training the network. A quarter of the data (validation set) is used to evaluate the current network error to determine convergence and overtraining. Currently we do not have convergence checks in the algorithm and are just running on a set number of iterations. The final quarter of data (test set) is used to evaluate the network error after the training is complete. The error in the network is calculated as mean square error per sample per feature.

Results:

**Future Steps**:

We met part of our proposal milestone by having a working neural network, but did not completely implement the dynamic portion of it. We’ve read through a lot of literature about dynamic neural networks but are still working on implementing it. Based on our results, our model isn’t accurately modeling the variance and complexity of our data. A dynamic neural network, which will take into account the time dependent nature of our data, will hopefully allow us to treat the three days of input data as a pattern over time rather than a single linear combination.

Another improvement we need to make is to normalize the error. Currently we just compute the mean square error across all of the features in all the samples. But the ranges of each feature vary from values between 0 and 10 and values between 0 and 360. So certain features are weighted more heavily in our error calculations.

After making the neural network dynamic and adding convergence checking conditions, we still need to tweak the number of hidden layers, number of neurons per layer and learning rate to minimize network error. We may use an adaptive learning rate to improve the speed of convergence if necessary. We will choose these hyperparameters by running a cross validation over a set of values. Then we can train the model on the entire training set and test on U.S. cities not in the training set. Also, we will test on weather data from cities outside of the U.S. to see how generalizable the model is.

**References**

1. “Lecture 10 – Neural Networks (Professor Yaser Abu-Mostafa).” *YouTube*. YouTube, 6 May 2012. 26 Oct 2014. [↑](#footnote-ref-1)