## Accomplishments

* Completed initial debugging of the RbmStack.
* Converted my MnistAutoencoder experiment from last semester so that I can run these experiments in python.
* Add cudamat support to RbmStack to speed up run time.
* Review some of the SVM, LSTM and bidirectional LSTM papers more thoroughly.
* Check out other possible paper sources (ICML, KDD)
* Dataset (traceable to source), Problem (poses a learning problem based on a Dataset), Algorithm (standard API), Experiment(applies an Algorithm to solve a Problem)

## To Do

Research possibility of using GitHub account.

What is the most convenient way to install all the right packages?

Is there a convenient way to switch versions?

Install python 2.7.4 on PC instead of 3.3.5. (Make notes of installation procedure including pandas, numpy, scipy, etc.)

## Setup procedure for machine learning computer

Download Ubuntu 14.04 ISO and burn image to DVD

Reboot the computer with the DVD in the drive and select boot from CD

Allow installation to erase all other installations

Install useful packages for python3:

* sudo apt-get install python3-numpy
* sudo apt-get install python3-scipy
* sudo apt-get install pandas

Install ssh-server to allow remote login:

* sudo apt-get install openssh-server
* sudo /etc/init.d/ssh restart

Download the CUDA 6.5 .deb file:

* cuda-repo-ubuntu1404\_6.5-14\_amd64.deb

Install CUDA:

* sudo dpkg -i cuda-repo-<distro>\_<version>\_<architecture>.deb
* sudo apt-get update
* sudo apt-get install cuda

Edit .bashrc to update environment variables (note that .bash\_profile might actually be the correct place to put these):

* export PATH=/usr/local/cuda-6.5/bin:$PATH
* export LD\_LIBRARY\_PATH=/usr/local/cuda-6.5/lib64:$LD\_LIBRARY\_PATH

Make ~/ML directory

Get and make cudamat:

* cd ~/ML
* git clone <https://github.com/cudamat/cudamat.git>
* cd cudamat
* make
* python3 test\_cudamat.py
* python3 test\_learn.py

Edit .bashrc to update environment variables:

* export PYTHONPATH=$PYTHONPATH:/home/mark/ML/cudamat

Wrote a python script to download MNIST files from their online source and assemble them into a single large pandas.DataFrame. The resulting DataFrame includes the complete raw dataset: pixel data, class labels, and an attribute signifying membership in the test or the training set. This is part of a general effort to develop a library of python benchmark datasets with a standard API.

Converted my RbmStack from matlab to a python class. I haven’t finished debugging this yet.

Reviewed all of the paper titles from NIPS 2013 to see if there might be papers dealing with time series prediction or time series classification. I didn’t find anything very relevant using this approach.

Searched Google scholar for recent papers on *time series prediction*. There are lots of these, heavily focused on financial market prediction. I found at least one paper claiming that SVMs outperform other methods.

Searched Google scholar for recent papers on *multiresolution neural networks* and *decimation neural networks*. There were not very many, but most of them also made reference to wavelets.

Searched Google scholar for recent papers referencing to Schmidhuber’s long short term memory (LSTM) paper. There are lots of these, and many of them are of potential interest. They revealed additional applications for *time series prediction* or *time series classification* could be applicable (speech recognition, phoneme recognition, text generation, song identification, voice activity detection, onset music onset detection…):

* A maze learning comparison of Elman, long short-term memory, and Mona neural networks.pdf
* Advanced Methods for Time Series Prediction Using Recurrent Neural Networks.pdf
* Advances in optimizing recurrent networks.pdf
* Draft Deep Learning in Neural Networks An Overview.pdf
* Enhanced beat tracking with context-aware neural networks.pdf
* First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs.pdf
* Generating Text with Recurrent Neural Networks.pdf
* High-dimensional sequence transduction.pdf
* How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation.pdf
* ICML2011Martens\_532.pdf
* Identifying Cover Songs Using Information-Theoretic Measures of Similarity.pdf
* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.pdf
* Multi-resolution Linear Prediction Based Features for Audio Onset Detection with Bidirectional LSTM Neural Networks.pdf
* On the difficulty of training recurrent neural networks.pdf
* Philosophers & futurists, catch up- Response to The Singularity.pdf
* Polyphonic piano note transcription with recurrent neural networks.pdf
* Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies.pdf
* Recurrent Models of Visual Attention.pdf
* Speech Recognition with Deep Recurrent Neural Networks.pdf
* Training deep and recurrent networks with hessian-free optimization.pdf
* Training recurrent neural networks.pdf
* Understanding Deep Architectures using a Recursive Convolutional Network.pdf
* Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks.pdf

## Next Steps

* Complete debugging of the RbmStack and conversion.
* Convert my MnistAutoencoder experiment from last semester so that I can run these experiments in python.
* Add cudamat support to RbmStack to speed up run time.
* Review some of the SVM, LSTM and bidirectional LSTM papers more thoroughly.
* Check out other possible paper sources (ICML, KDD)
* Dataset (traceable to source), Problem (poses a learning problem based on a Dataset), Algorithm (standard API), Experiment(applies an Algorithm to solve a Problem)

­