CS 51. Final Specification.

Angela Fan, Andre Nguyen, Vincent Nguyen, George Zeng.

Due Friday, April 17, 2015.

1 Signatures/Interfaces

See attached `Python` *code and comments for more detailed descriptions. //*

Python is not strongly typed and we will not necessarily need to create too many objects outside of the arrays, lists, and other data structures provided in the `numpy` package for most of the interactions and implementations.

Python does not provide a strict method of defining public and private data types. However, within the overarching RBM class, we will have the following private methods:

- Hopfield Energy Function
- Logistic Sigmoid Function
- Partition Function
- Probability of Visible/Hidden Pair
- Probability of Visible

We will also have the following public methods:

- Learner training method
- Method for predicting hidden to visible unit
- Method for predicting visible to hidden unit

Input formats will be pre-processed so that we can input a wide variety of types such as varying image sizes.

2 Modules/Actual Code

See attached `Python` *code and comments for more detailed descriptions.*

3 Timeline

3.1 Week of Monday, April 13th

1. Finish final draft specification
2. Start creating the pseudocode for each function/class in the program
3. Within actual code, start skeleton code/signature for each function/class
4. Delegate concrete responsibilities to team members
5. Create fixed timeline for internal deadlines
6. Meet together on Sunday, April 19th for a day-long group coding session

3.2 Week of Monday, April 20th

1. Implement basic version with binary hidden and visible units
2. Fill in all the implementations of the function definitions
3.
4.
5.

3.3 Week of Monday, April 27th

1. *Overarching goals for this week:* finish implementation, start adjusting meta-parameters in application/implementation
2. Start testing the RBM on a simple data set such as the MNIST handwritten digit database
3. Analyze performance, make adjustments on RBM, choose optimal input data set (such that the computation is feasible given our time constraints)
4. Tidy up the code, add in any extra comments
5. Finish writing up the README
6. *Extension:* Change the implementation such that hidden and visible units are not necessarily binary
7. *Extension:* Add regularization to the algorithm
8. *Extension:* Implement mini-batches

9. *Potential Extension:* Add graphical displays to check learning progress, overfitting, adjust learning rate

10. *Potential Extension:* Implement momentum method

4 Progress Report

The attached `RBM_code.py` Python code contains the skeleton code for the RBM along with comments addressing particular parts of the final specification.

5 Version Control

We have already created a GitHub repository for our final project. The clone URL is `https://github.com/huihuifan/CS51-FinalProject.git`.