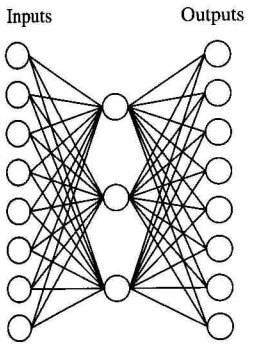
**CS 6140: Machine Learning** **Spring 2015**
**College of Computer and Information Science**
**Northeastern University**
**Assignment 5** **April, 01**
**Instructor: Bilal Ahmed** **Due Date: April, 13**

# 1 Autoencoder Neural Network

Consider the following neural network (left graph), with 8 input units (for data with 8 features), 3 hidden units and 8 output units, and assume the nonlinear functions are all sigmoid.




| Input | | Hidden Values | | | | Output |
|---|---|---|---|---|---|---|
| 10000000 | → | .89 | .04 | .08 | → | 10000000 |
| 01000000 | → | .15 | .99 | .99 | → | 01000000 |
| 00100000 | → | .01 | .97 | .27 | → | 00100000 |
| 00010000 | → | .99 | .97 | .71 | → | 00010000 |
| 00001000 | → | .03 | .05 | .02 | → | 00001000 |
| 00000100 | → | .01 | .11 | .88 | → | 00000100 |
| 00000010 | → | .80 | .01 | .98 | → | 00000010 |
| 00000001 | → | .60 | .94 | .01 | → | 00000001 |

Figure 1: An auto encoder neural network, all the hidden and output units use the sigmoid activation function. (*Taken from: TM*)

1. The 8 training data inputs are identical with the outputs, as shown in the right side table. Implement this network and the backpropagation algorithm to compute all the network weights; you should initialize the weights with nontrivial values (i.e not values that already minimize the erorr as shown in Figure-1). *HINT: on the trained network, you should obtain values for the hidden units somehow similar with the ones shown in the table (up to symmetry). Feel free to make changes to the algorithm that suit your implementation, but briefly document them.*

2. Since the outputs and inputs are identical for each datapoint, one can view this network as an encoder-decoder mechanism. (this is one of the uses of neural networks). In this context, explain the purpose of the training algorithm. (I expect a rather nontechnical –but well-explained– answer).

3. Train the same network with different number of hidden units $\in \{1, 2, 4\}$, and explain the effects of changing the size of the hidden layer on the training time, convergence of backpropagation, etc. With these changes can we still think of the network as an encoder/decoder?

4. Consider that the output units for the neural network shown in Figure-1 are linear units. What changes should be made to the backpropagation algorithm to train the network. Provide the

mathematical details needed to change the algorithm and generalize it to training an arbitrary network whose hidden units are sigmoid while the output units are linear (i.e., their output is $\sum_{k \in output} w_{jk} x_{jk}$).

# 2 Perceptron and Dual Perceptron

## 2.1 Data and Pre-processing

- PerceptronData: This is a binary classification dataset consisting of four features and the classes are linearly separable.
- Two Spirals: This dataset has two features and it is non-linearly separable (Figure-2).

## 2.2 Implementation

1. Implement the perceptron algorithm and test it on the PerceptronData dataset using ten fold cross validation.
2. Implement the dual perceptron algorithm and test it on the PerceptronData dataset using ten fold cross validation.
3. Compare the performance of the two algorithms on the PerceptronData dataset and make sure that they have (almost) identical performance.

## 2.3 Kernelizing Dual Perceptron

1. Run the dual perceptron with the linear kernel on the Two Spiral dataset and show that the data is not separable using ten-fold cross validation.

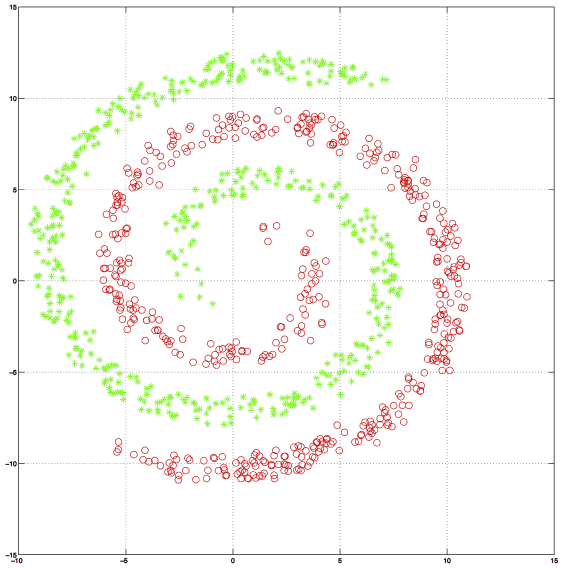



Figure 2: The two spirals dataset, the colors represent class labels.

2. Run the dual perceptron with the Gaussian (RBF) kernel on the Two Spiral dataset and show that the data is separable using ten-fold cross validation.

**Note:** You would need to select an appropriate value for the scale $\gamma$ of the RBF kernel. Design a search strategy to set the value of $\gamma$ that searches for possible values of $\gamma \in [0, 0.1]$.) For both the perceptron and dual perceptron be sure to set an upper limit on the number of iterations.

## 3 Regularized Logistic Regression

In this question you will develop a regularized version of Logistic regression. Let $(x_i, y_i)$, $i \in \{1, 2, \ldots, N\}$ represent the training data where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Under the logistic regression model, the probability of an instance belonging to either class is given as:

$$P(y_i|w, x_i) = \frac{1}{1 + e^{-y_i w^T x_i}} \tag{1}$$

where, $w$ are the model parameters. The maximum likelihood solution can be obtained by minimizing the negative log-likelihood. For the regularized case, we add an L2 penalty on the norm of $w$ which results in the objective function for regularized Logistic regression:

$$\min_w \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{N} \ln(1 + e^{-y_i w^T x_i}) \tag{2}$$

1. Do you think that $w_0$ should be included in the regularization?
2. Calculate the gradient of the objective with respect to the model parameters.
3. Develop a gradient descent algorithm for learning the parameters from given training data.
4. Contrast the performance of Logistic Regression with Regularized Logistic Regression for the Spambase, Diabetes and Breast Cancer datasets using ten fold cross validation.
5. Is it possible to kernelize Equation-2?

**Note:** If you google Regularized Logistic Regression, Andrew Ng has a video lecture that details the development of Regularized Logistic Regression from simple Logistic Regression. I would recommend you to watch the video after you have spent some time trying to develop the solution yourself.