605.649 — Introduction to Machine Learning

Programming Project #3

1 Project Description

The purpose of this assignment is to give you experience developing one of the principle algorithms for training feedforward neural networks—backpropagation—and to give you experience working within an area of machine learning known as deep learning. In this project, you will begin by considering logistic regression and linear regression as foundational algorithms. You will then evaluate the performance of a feedforward network trained with backprop on classification and regression problems. In addition, you will compare the basic feedforward networks to a method involving the pre-training of an autoencoder to do feature extraction, forming the basis for modern deep neural networks.

Autoencoders will be discussed as part of the module on deep learning, but you don't need the details of that discussion for this assignment. To get you going on this project, we will present the fundamentals of the autoencoder here. For a simple autoencoder, the idea is to train a network in such a way that the output pattern matches the input pattern. In other words, if we have some pattern $\mathbf{x}$, we wish to learn a pattern $\hat{\mathbf{x}}$ such that

$$L(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 = 0.$$

Note that this is basically the same as squared error, which we often use to train a regression neural network. To learn the reconstruction of a pattern, we interpret two functions of the network. The first introduces an "encoding" layer, which we can define as $\mathbf{y} = f(\mathbf{x})$. We can think of this as corresponding to the transformation that happens at the hidden layer of the network. The second then introduces a "decoding" layer, where the idea is to transform the encoded pattern back to match the input pattern. We define the decoding layer as $\hat{\mathbf{x}} = g(\mathbf{y})$. Then the task is to minimize $L(\mathbf{x}, g(f(\mathbf{x})))$. Note that the number of input nodes equals the number of output nodes.

For the autoencoder, we will use a feedforward neural network with one hidden layer. That hidden layer becomes the encoding layer, and the output layer becomes the decoding layer. When taking this approach, if the hidden layer has fewer nodes than the input or output layers, we have effectively found a way to compress the patterns. In other words, an autoencoder provides a means to reduce the dimensions in the data space, providing a form of feature extraction.

For this project, once you have trained the autoencoder, you want to remove the output layer of the network and then append a traditional feedforward network to the output of the autoencoder's hidden layer. Specifically, we take the hidden layer, and rather than connect it back to the output layer of the autoencoder, we tie it directly to a classification or regression layer(s). Then we train the network using backpropagation. Note that there are two ways to do this. For the first, we simply train the new network while locking the weights of the encoding layer. The second is to train the entire network as a complete feedforward network. This has the effect of "fine tuning" the weights feeding the encoding layer from the autoencoder as well as learning the weights in the prediction layer(s). It is your choice which approach you take.

2 Data Sets

You will also use the same six datasets from the UCI Machine Learning Repository, namely:

1. Breast Cancer [Classification]

   `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29`

   This breast cancer data set was obtained from the University of Wisconsin

2. Car Evaluation [Classification]

   `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`

   The data is on evaluations of car acceptability based on price, comfort, and technical specifications.

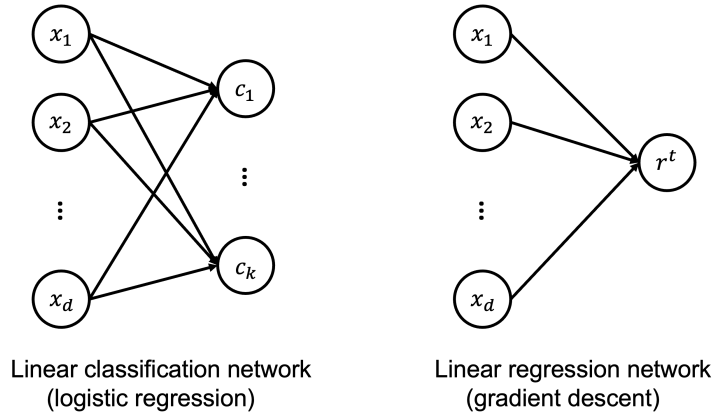

Figure 1: Linear Network Topologies

3. Congressional Vote [Classification]

   `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`

   This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac.

4. Abalone [Regression]

   `https://archive.ics.uci.edu/ml/datasets/Abalone`

   Predicting the age of abalone from physical measurements.

5. Computer Hardware [Regression]

   `https://archive.ics.uci.edu/ml/datasets/Computer+Hardware`

   The estimated relative performance values were estimated by the authors using a linear regression method. The gives you a chance to see how well you can replicate the results with these two models.

6. Forest Fires [Regression]

   `https://archive.ics.uci.edu/ml/datasets/Forest+Fires`

   This is a difficult regression task, where the aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data .

3 Project Requirements

For this project, the following steps are required:

- Download the six (6) data sets from the UCI Machine Learning repository. You can find this repository at `http://archive.ics.uci.edu/ml/`. All of the specific URLs are also provided above.
- Pre-process each data set as necessary to handle missing data and non-Boolean data (both classes and attributes).
- Implement logistic regression for the three classification problems and a simple linear regression network for the three regression problems. Note that we can think of logistic and linear regression as simple linear networks with the inputs feeding directly into the output (softmax or linear) layer. For regression, the linear network where the inputs feed directly into a single linear output node is equivalent to ordinary least squares regression trained via gradient descent. The topologies of these linear networks are shown in Figure 1.

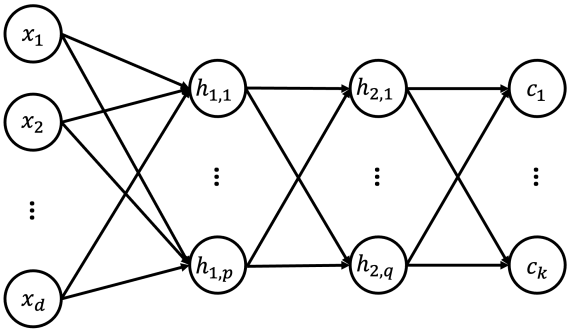

Figure 2: Feedforward Network Topology

- Implement backpropagation for training feedforward neural networks. You may choose whether or not you wish to use the logistic activation function or the hyperbolic tangent activation function for the hidden layers. The output should be softmax for classification and linear for regression. It is your choice whether or not you use momentum.
- The classification problems should use cross-entropy loss, and the regression problems should use mean squared error.
- Train a traditional feedforward network (without using the autoencoder) with *two* hidden layers. Remember to tune the number of hidden nodes in each layer. The topology for this network is shown in Figure 2.
- Train an autoencoder based network where the autoencoder has one hidden layer and the prediction layer has one hidden layer. After you have built this network, notice that you will have yet another feedforward network with two hidden layers (one from the autoencoder and one from the prediction part). The topology for this networks is shown in Figure 3. Note that the top network is trained first using backpropagation. The gray nodes are then removed and the bottom nodes are added in to complete training, also via backpropagation.
- Run your algorithms on each of the data sets. These runs should be done with 5×2 cross-validation so you can compare your results statistically. To be clear, note that you will be training three different networks on each data set, ten times each:

  1. A linear network for each of the classification and regression data sets,
  2. A simple feedforward network with two hidden layers (Input $\Rightarrow$ Hidden 1 $\Rightarrow$ Hidden 2 $\Rightarrow$ Prediction) for each of the classification and regression data sets, and
  3. A feedforward network where the first hidden layer is trained from an autoencoder and the second hidden layer is trained from the prediction part of the network (Input $\Rightarrow$ Encoding $\Rightarrow$ Hidden 2 $\Rightarrow$ Prediction) for each of the classification and regression data sets.

  Remember that "Prediction" in the above should use a softmax output layer for classification and a linear output for regression.

  It is up to you to tune the number of hidden nodes in each layer, and be sure to explain how you did the tuning. Remember that, for the encoding layer of the autoencoder, the number of hidden nodes should be strictly less than the number of input nodes.
- Write a very brief paper that incorporates the following elements, summarizing the results of your experiments. Your paper is required to be at least 5 pages and no more than 10 pages using the JMLR format You can find templates for this format at `http://www.jmlr.org/format/format.html`. The format is also available within Overleaf.

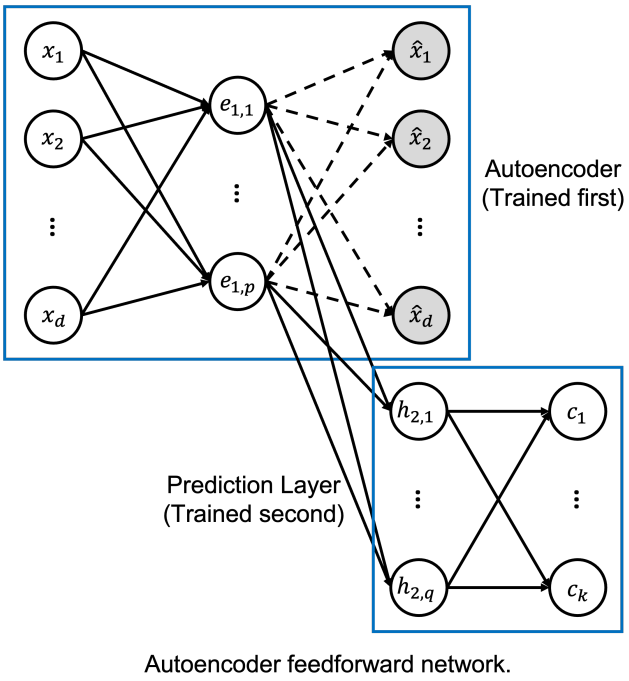

Figure 3: Autoencoder Network Topology

1. Title and author name
2. A brief, one paragraph abstract summarizing the results of the experiments
3. Problem statement, including hypothesis, projecting how you expect each algorithm to perform
4. Brief description of your experimental approach, including any assumptions made with your algorithms
5. Presentation of the results of your experiments
6. A discussion of the behavior of your algorithms, combined with any conclusions you can draw
7. Conclusion
8. References (Only required if you use a resource other than the course content.)

- Submit your fully documented code, the video demonstrating the running of your programs, and your paper.
- For the video, the following constitute minimal requirements that must be satisfied:
  - The video is to be no longer than 5 minutes long.
  - The video should be provided in mp4 format. Alternatively, it can be uploaded to a streaming service such as YouTube with a link provided.
  - Fast forwarding is permitted through long computational cycles. Fast forwarding is *not permitted* whenever there is a voice-over or when results are being presented.
  - Be sure to provide verbal commentary or explanation on all of the elements you are demonstrating.
  - Provide sample outputs from one test fold showing performance on each of your three networks.
  - Demonstrate and explain how an example is propagated through each network. Be sure to show the activations at each layer being calculated correctly.
  - Demonstrate the weight updates for the linear networks (one each).

  - Demonstrate the weight updates occurring on a two-layer network for each of the layers.
  - Demonstrate the weight updates occurring during the training of the autoencoder.
  - Demonstrate the autoencoding functionality (i.e., recovery of an input pattern on the output).

4 Project Evaluation

Your grade will be broken down as follows:

- Code structure – 10%
- Code documentation/commenting – 10%
- Proper functioning of your code, as illustrated by a 5 minute video – 30%
- Summary paper – 50%