BMI203 Winter 2021
Homework Assignment #3
35 Total Points

Due Friday 03/19/2021 by 11:59 PST

In this assignment, you will try out your machine learning skills by developing a classifier for transcription factor binding sites. Transcription factors are proteins that bind DNA at promoters to drive gene expression. Most preferentially bind specific patterns of sequence, while ignoring others. Traditional methods to determine this pattern (called a motif) have assumed that binding sites in the genome are all independent. However, there are cases where people have identified motifs where positional interdependencies exist. Interestingly, some of these cases are isomorphic to the XOR problem that killed perceptrons a few decades ago.

To address this, you will implement a multilayer neural network capable of accurately predicting whether a short DNA sequence is a binding site for the yeast transcription factor Rap1.

The data provided include both positive examples of TF binding sites for the yeast transcription factor Rap1 along with negative examples consisting of a few thousand yeast promoters to which Rap1 is known not to bind. The specific input data files are the following:

- 137 true positive binding sites for the yeast Rap1 transcription factor, each 17 base pairs long (rap1-lieb-positives.txt)
- Yeast genomic sequence as the negative training data. (yeast- upstream-1k-negative.fa) Each of the approximately 3000 fasta sequences contains 1000 bases upstream of a yeast gene.

Your tasks for the assignment follow. Please answer the indented questions in your writeup.

Project skeleton: https://github.com/ucsf-bmi-203-2021/Project3

- **You must fork / clone this repo for this assignment**

Part 1: Autoencoder implementation (10 points)

1) Implement a feed-forward, neural network with standard sigmoidal units. Your implementation should accept a vector as input, be able to adjust network weights by backpropagation, and report the average loss per epoch. It should also allow for variation in the size of input layer, hidden layers, and output layer. We expect that you will be able to produce fast enough code to be of use in the learning task at hand.

a. To confirm that your implementation functions correctly, demonstrate its ability to correctly solve the 8 x 3 x 8 autoencoder task. Specifically, set up an autoencoder network consisting of an input layer (8 nodes), a hidden layer (3 nodes), and an output layer (8 nodes), all with sigmoidal units.

NOTE: If you are unable to successfully implement the neural network code directly yourself, you should not stop working. While it will be reflected in your project grade, you should continue with the Rap1 learning task using code for the core learning method obtained elsewhere. In this case, you should still document your effort toward making a NN that can learn the 8 x 3 x 8 encoder.

Part 2: Adapt for classification, and develop training regime (10 points)

2) Set up a procedure to encode DNA sequences (Rap1 binding sites) as input for your neural network. Consider how your encoding strategy may influence your network predictions.

a. Describe your process of encoding your training DNA sequences into input vectors in detail. Include a description of how you think the representation might affect your network's predictions.

3) Design a training regime that will use both positive and negative training data to train your predictive model. Note that if you use a naive scheme here, which overweights the negative data, your system will probably not converge (it will just call everything a non-Rap1 site).

Your system should be able to quickly learn aspects of the Rap1 positives that allow elimination of much of the negative training data. Therefore, you may want to be clever about caching negative examples that form "interesting" cases and avoid running through all of the negative data all of the time.

In addition, there are a few factors to consider:

- Your network input is a fixed size - consider how best to modify your negative examples such that they can be input into your network.
- You don't have to use the full 17 bases of each binding site.
- You can use a subset of the data if you think performance can be improved.
- By chance, there may be some examples of subsequences from the negative set that correspond exactly to one of the positive example sequences. If this occurs, you probably should not include the example in your training!

a. Describe your training regime. How was your training regime designed so as to prevent the negative training data from overwhelming the positive training data?

4) Modify your implementation to take as input positive and negative examples of Rap1 binding sites (using your encoding from Q2) and produce an output probability between [0 - 1.0] indicating classification as a binding site (1.0) or not (0.0) . Select a network architecture, and train your network using the training regime you described in Q3 on all the data.

a. Provide an example of the input and output for one true positive sequence and one true negative sequence.
b. Describe your network architecture, and the results of your training. How did your network perform in terms of minimizing error?
c. What was your stop criterion for convergence in your learned parameters? How did you decide this?

Part 3: Cross-validation (5 points)

5) Evaluate your model’s classification performance via k-fold cross validation.

a. How can you use k-fold cross validation to determine your model’s performance?
b. Given the size of your dataset, positive and negative examples, how would you select a value for *k*?
c. Using the selected value of k, determine a relevant metric of performance for each fold. Describe how your model performed under cross validation.

Part 4: Extension (required but graded generously) (5 points)

Try something fun to improve your model performance! This should include implementation of alternative optimization methods (particle swarm, genetic algorithms, etc), you can also optionally add changes in the network architecture such as modifying the activation function, changing the architecture, adding regularization etc. For this section, we want to see a description of what you want to try and why. As long as we have this, and some effort towards implementation, you will get full points.

- What set of learning parameters works the best? Please provide sample output from your system.
- What are the effects of altering your system (e.g. number of hidden units or choice of kernel function)? Why do you think you observe these effects?
- What other parameters, if any, affect performance?

Part 5: Evaluate your network on the final set.

6) To provide an objective measure of your neural network's ability to classify binding sites, a test dataset has been provided (rap1-lieb-test.txt). There are no class labels on these sequences. Your goal is to maximize the separation in scores for the true Rap1 binding sites as compared with the non-sites.

a. Select a final model (encoding, architecture, training regime). This can be the same as your model in Part 3, Part 4, or something completely different.
b. Train your final model on the entire training dataset. Run the trained system on the test dataset. For each sequence, output the sequence and its output value from the network, separated by a tab, as follows:

```
ACATCCGTGCACCATTT 0.927
AAAAAAACGCAACTAAT 0.123
```

If you do not use the full 17 bases, please still output the original sequence in column 1 and the binding score in column 2, as in the above example.

Part 6: Automated Testing with Github Workflows (1 point)

The project skeleton is currently set up with a “workflow” that triggers when a commit is pushed to the main branch. This runs pytest with your python testing files located in the `/test/` folder.

For this reason, you must implement your project code in a **forked / cloned version of the project skeleton repo**. Code not built on this skeleton will not be accepted.

If this is your first time using github workflows, you must enable them under the “**Actions**” tab in your forked repo. If you do not enable them, commits pushed to main will not automatically activate testing.

To verify that your unit tests have passed, your testing badge, located in the README.md, must point to the correct URL. You can alter the URL for this badge as follows:

https://github.com/{your-username}/{your-repo-name}/workflows/HW3/badge.svg?event=push

To submit this assignment,

- Use Google Classroom to submit a link to your **public** Github repository that:
    - Contains a single pdf or ipython notebook labeled **Firstname_Lastname_BMI203_HW3.(pdf/ipynb)**, including your answers to the above questions in prose and associated plots.
    - Contains a README.Md that explains the layout of your repo (what code is where, how to use your implementation)
    - Contains all code relevant for this assignment
    - Is commented explaining the role of each functional code block
    - Has passed the implemented unit tests (as indicated by the github badge)
    - ***Note!*** we will only consider commits before the submission deadline

Other graded requirements:

1. API/Docs - 2 point
2. Thoughtful code structure - 2 point