CS689: Machine Learning - Fall 2020

Homework 3

Assigned: Monday, Oct 12. Due: Monday, Oct 26 at 11:59pm

Getting Started: You should complete the assignment using your own installation of Python 3.6. Download the assignment archive from Moodle and unzip the file. The data files for this assignment are in the `data` Directory. Code templates are in the `code` directory. You can use any Numpy and PyTorch modules for this assignment.

Deliverables: This assignment has three types of deliverables: a report, code files, and prediction files.

- **Report:** The solution report will give your answers to the homework questions (listed below). The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You can use any software to create your report, but your report must be submitted in PDF format. You will upload the PDF of your report to Gradescope under `HW03-Report` for grading. Access to Gradescope will be enabled one week before the assignment is due. For full credit, all figures must have proper axes, labels, legends, axis ticks, and titles. Any tables must have proper row and/or column headings and titles or captions. You do **not** need to include the text of questions in your report.
- **Code:** The second deliverable is your code. Your code must be Python 3.6 compatible (no iPython notebooks, other formats, or code from other versions of Python). You will upload a zip file (not rar, bz2 or other compressed format) containing all of your code to Gradescope under `HW03-Programming` for autograding. Access to the autograder will be enabled one week before the assignment is due. When unzipped, your zip file should produce a directory called `code`. If your zip file has the wrong structure, the autograder may fail to run. To receive credit for implementation questions, your code must run in Gradescope.
- **Prediction Files:** For Question 3, the test outputs are not included in the data set. Instead, you will output a file containing predictions for test instances and upload it to Gradescope. Your predictions will be graded at the end of the assignment development period.

Academic Honesty Reminder: Homework assignments are individual work. Being in possession of another student's solutions for any reason is considered cheating. Sharing your solutions with other students for any reason is considered cheating. Copying solutions from external sources (books, web pages, etc.) is considered cheating. Posting your code to public repositories like GitHub (during or after the course) is also considered cheating. Collaboration indistinguishable from copying is considered cheating. Manual and algorithmic cheating detection is used in this class. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Questions:

1. (*20 points*) **Dual for Logistic Regression:** While logistic regression does not have a useful Lagrange dual, an alternative dual can be derived using the fact that $\log \sigma(z) = \min_{\alpha \in [0,1]} \alpha \cdot z - H(\alpha)$ where $\sigma(z)$

is the logistic function and $H(\alpha)$ is the entropy of the Bernoulli distribution $H(\alpha) = -\alpha \log(\alpha) - (1 - \alpha) \log(1 - \alpha)$. Using this fact, we can express the MAP objective for logistic regression via a saddle point problem as shown below:

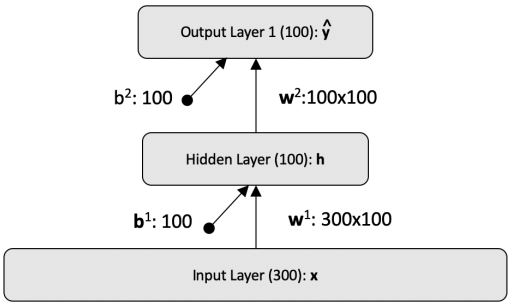
$$\min_{\mathbf{w}} \max_{\alpha_{1:N}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} (\alpha_n y_n \mathbf{x}_n \mathbf{w} - H(\alpha_n)) \quad \text{s.t.} \quad \forall n \;\; \alpha_i \in [0, 1]$$

a. (*14 pts*)Derive a dual objective for logistic regression that only has the α's as parameters by minimizing $\mathbf{w}$ out of the above saddle point problem. Simplify the resulting dual objective as much as possible. Show your work.

b. (*6 pts*) Give two reasons why one one might want to optimize this dual objective for logistic regression instead of the primal objective.

2. (*20 points*) **Neural Networks for Sequence-to-Sequence Time Series Forecasting:** In this question, you will implement a neural network model for time series forecasting. The inputs for each data case n are an array of values $\mathbf{x}_n = [x_{n1}, ..., x_{nT}]$ where x_{nt} corresponds to the value of the t^{th} time point for time series n. T is the same for all instances in this problem. The desired output $\mathbf{y}_n$ corresponds to the next T' values of time series n. Specifically, $\mathbf{y}_n = [y_{n1}, ..., y_{nT'}]$ where y_{nt} corresponds to the value of time series n at time $T + t$. T' is the same for all instances in this problem. This problem is thus a special case of the general sequence-to-sequence problem where the input and output sequences have the same length for all data cases.

The architecture diagram for the basic model you will implement is shown below. We will use the following model specification: $\hat{y}_{nt} = \mathbf{h}_n \mathbf{w}_t^2 + b_t^2$ and $h_{nk} = \text{relu}(\mathbf{x}_n \mathbf{w}_k^1 + b_k^1)$ (Note: superscripts refer to layer index, not powers). The loss function we will use to learn the model is the mean squared error over the data cases and output time series values. Let $\hat{\mathbf{y}}_n = [\hat{y}_{n1}, ..., \hat{y}_{nT'}]$ be the predicted outputs for time series n. The loss function is thus: $\frac{1}{NT'} \sum_{n=1}^{N} \sum_{t=1}^{T'} (y_{nt} - \hat{y}_{nt})^2$.



The data set we are using for this problem consists of one-lead echocardiogram (ECG) traces[1]. These data have significant structure, but will be very challenging to predict accurately. This problem has not been investigated on this data set previously, so optimal architectures are not known, nor is the floor for achievable loss values.

a. (*10 pts*) Starting from the provided template (`nn.py`), implement a Scikit-Learn compatible class for the model shown above. You will develop your implementation using NumPy and PyTorch. You are strongly

[1] See `https://en.wikipedia.org/wiki/Electrocardiography`

encouraged to use automatic differentiation methods to develop your implementation. In the implementation of `fit` for this question, use the RMSprop optimizer included with PyTorch. Use a learning rate of $1e-3$ and the RMSprop optimizer's built-in weight decay set to $1e-4$ and momentum set to 0.9. You should learn the model using a batch size equal to the full training set size.

b. (*5 pts*) Using the provided training data set, learn the model for 400 epochs (in this case epochs and iterations are the same as we are learning the model on full batches). Report the loss on both the training and validation sets.

c. (*5 pts*) Provide line plots of the first five validation set outputs and their corresponding predictions. There should be one plot per validation case showing the true and predicted time series values. Be sure to provide a legend and axis labels.

3. (*60 points*) **More Neural Networks:** In this problem, your task is to attempt to improve on the performance of the model and learning approach presented in Question 2. You can make any modifications that you like to the model or the learning algorithm, but the best model you select is limited to one minute of compute time per epoch as measured in the Gradescope autograding environment.

a. (*20 pts*) Begin by exploring different models and/or algorithms for this problem. You must try at least one CNN and one RNN architecture. Describe at a high level the different architectures and/or algorithm choices that you tried. Summarize your results using tables or figures. Upload the code for one of your RNN and one of your CNN architectures in `rnn.py` and `cnn.py`.

b. (*10 pts*)Next, you will need to select a single best model/algorithm combination among the choices you explored in part (a). As your answer to this question, describe what experimental procedures you used to select the single best model. Also describe why you selected this approach.

c. (*5 pts*) For the single best model that you selected, provide an architecture diagram (similar to the one shown above) or specify your architecture in detail using equations (Note: this question is not asking you to include a listing of the PyTorch code defining your model in your report). Upload code for your best model in `best_nn.py`.

d. (*10 pts*)Describe all details of the learning algorithm used for your best model. How did you select final values for any hyperparameters?

e. (*5 pts*) Provide line plots of the first five validation set outputs and their corresponding predictions using your best model. There should be one plot per validation case showing the true and predicted time series values. Be sure to provide a legend and axis labels.

f. (*10 pts*) Using the provided test data, compute predictions for all test data cases. Save a Numpy array of predictions to your code folder using `np.save("predictions.npy", ...)`. Upload the prediction file to Gradescope for scoring. (Note: point allocations for this question will be based on the performance of your best model relative to an improved baseline model developed by the course staff, as well as to the best solutions found by other students in the class).