CS689: Machine Learning - Fall 2019

Homework 3

Assigned: Wednesday, Oct 23. Due: Wednesday, Nov 6 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. The only modules allowable during autograding are those already imported in the code templates.

Deliverables: This assignment has two types of deliverables: a report and code 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 `HW02-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 `HW02-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.

Academic Honesty Reminder: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Homework assignments are individual work. Sharing your solutions with other students is considered cheating. Posting your code to public repositories like GitHub is also considered cheating. Collaboration indistinguishable from copying is considered cheating. 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*) **Lagrangian Dual for Absolute Loss Regression:** Consider the problem of finding optimal linear regression model weights in the RRM framework using the absolute loss and a squared 2-norm regularizer:

$$\theta_* = \arg\min_{\theta} C \sum_{n=1}^{N} |y_n - f(\mathbf{x}_n, \theta)| + \|\mathbf{w}\|_2^2$$

where $\theta = [\mathbf{w}, b]$ and $f(\mathbf{x}, \theta) = \mathbf{w}\mathbf{x}^T + b$. This problem is convex, but not differentiable. However, the problem can be converted into an alternative linearly constrained form where the objective function is quadratic in an expanded set of variables $[\theta, \epsilon]$ where $\epsilon = [\epsilon_1^+, \epsilon_1^-, ..., \epsilon_N^+, \epsilon_N^-]$:

$$\begin{aligned}
\theta_*, \epsilon_* = \arg\min_{\theta,\epsilon} & C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \|\mathbf{w}\|_2^2 \\
\text{s.t.} \quad & \forall n \;\; y_n \leq f(\mathbf{x}_n, \theta) + \epsilon_n^+ \\
& \forall n \;\; y_n \geq f(\mathbf{x}_n, \theta) - \epsilon_n^- \\
& \forall n \;\; \epsilon_n^+ \geq 0 \\
& \forall n \;\; \epsilon_n^- \geq 0
\end{aligned}$$

a. (*5 pts*) Derive the Lagrangian function for the constrained formulation shown above.

b. (*10 pts*) Derive the Lagrangian dual for the constrained formulation shown above.

c. (*5 pts*) Explain the advantages of solving the Lagrangian dual problem instead of the constrained version of the primal problem.

2. (*60 points*) **Multi-Output Neural Networks:** In this question, you will implement a custom neural network model that simultaneously solves a regression problem (detecting the angle of an object in an image), and a multi-class classification problem (determining the class of the object present in an image) using a composite loss. The prediction model $f(\mathbf{x}, \theta)$ will simultaneously produce a vector of class probabilities $\mathbf{p} = f^c(\mathbf{x}, \theta)$ where $\mathbf{p}_y$ gives the probability for class y, and an angle output $f^a(\mathbf{x}, \theta) \in \mathbb{R}$. A data set for this problem is a set of triples $\mathcal{D} = \{(\mathbf{x}_n, y_n, a_n) | 1 \leq n \leq N\}$ where $\mathbf{x}_n$ is the feature vector, $y_n \in \{0, ..., C-1\}$ is the class label, and a_n is the angle in degrees. The objective function for this learning problem is shown below where α is a parameter that trades off between the classification loss (cross entropy, L_{ce}) and a loss defined on angles ($L_{\cos}$).[1]

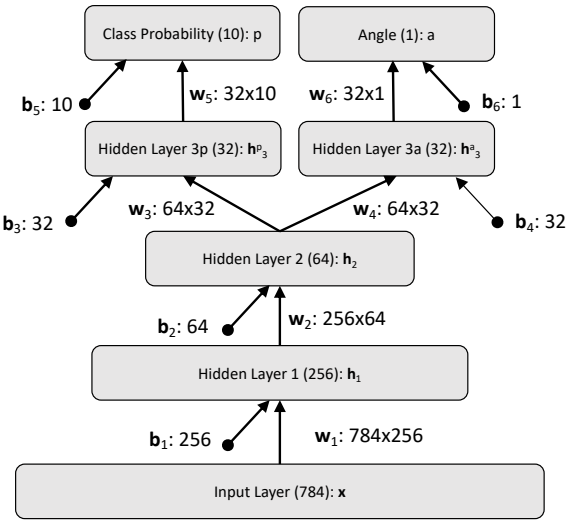
$$\theta_* = \arg\min_{\theta} \sum_{n=1}^{N} \alpha L_{ce}(y_n^c, f^c(\mathbf{x}_n, \theta)) + (1-\alpha) L_{cos}(a_n, f^a(\mathbf{x}_n, \theta))$$

$$L_{ce}(y, \mathbf{p}) = -\sum_{i=0}^{C} [y = i] \log p_i$$

$$L_{\cos}(a, a') = 0.5(1 - \cos(0.01745 \cdot (a - a')))$$

In this question, you will begin by implementing an architecture for this problem consisting of an input layer, three hidden layers, and two parallel output layers (one each for localization and classification). The inputs are 28x28 images, resulting in 784-long input vectors. All hidden layers will be use RELU non-linearities. The class probability output layer will use a softmax non-linearity. The angle output layer will consist of one linear unit. The diagram of the network architecture you will implement is shown below. All layers joined by arrows in the figure are fully connected.

[1] This form of the loss assumes that the cos function expects inputs in radians, while the data set specifies that angles are provided in degrees. 0.01745 is then the approximate conversion factor from degrees to radians.



a. (*40 pts*) Starting from the provided template (`nn.py`), implement a Scikit-Learn compatible class for the model shown above. You can develop your implementation using NumPy and Torch (version 1.2). You are strongly encouraged to use automatic differentiation methods to develop your implementation. In the implementation of `fit` for this question, use the Adam optimizer included with Torch. Use a learning rate of $1e-4$ and the Adam optimizer's built-in weight decay with a regularization constant of $1e-4$. Your fit function should use mini-batches of 64 examples. To ensure that learning is numerically stable, it is recommended that you do not explicitly compute the softmax function during learning and prediction to avoid issues with underflow/overflow. See `nn.functional.cross_entropy` for methods to help with this.

b. (*5 pts*) Using the provided training data set, learn the model using $\alpha = 0.5$ for 20 epochs.[2] Report the classification error rate and the mean absolute error (MAE) of the angle predictions obtained on both the training and validation sets.[3] Training should take less than 5 minutes if properly implemented.

c. (*15 pts*) Use your implementation to perform experiments to assess how changing the value of α affects the classification and angle prediction results on the validation set. In particular, vary the value of α from 0 to 1 in steps of 0.1 and plot the classification error rate and the mean absolute error of the angle prediction task on the validation data. Use two plots, one for each performance measure. Note that setting $\alpha = 0$ is equivalent to solving the angle prediction task on its own, while setting $\alpha = 1$ is equivalent to solving the classification problem on its own. Is there a setting of α where jointly learning these two tasks using this model provides a benefit compared to learning the angle prediction task on its own?

3. (*20 points*) **More Neural Networks:** In this problem, your task is to attempt to improve on the predictive performance of the model presented in Question 2. You must still use a single multi-output neural network model, but you are free to experiment with any other architecture and algorithm choices. Performance should still be assessed in terms of the classification error rate and the mean absolute error of the angle predictions. Answer the following questions.

[2] One epoch of training consists of one full sweep through the data set such that every data case is seen one time (the last batch may have size less than 64.

[3] Predict the class with the highest probability in all questions

a. (*5 pts*) Begin by exploring different models and/or algorithms. Describe at a high level some of the different architectures and/or algorithm choices that you tried.

b. (*5 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 procedure you used to selected the single best model.

c. (*5 pts*) For the single best model that you selected, provide a detailed architecture diagram (similar to the one shown above) and describe all details of the learning algorithms used.

d. (*5 pts*) Using the provided test data, compute predictions for the class labels and angles. Save the array of class predictions to your code folder using `np.save("class_predictions.npy", ...)`. Save the array of angle predictions to your code folder using `np.save("angle_predictions.npy", ...)`. Upload both files to Gradescope along with your code for scoring. Your model should be implemented in `best_nn.py`.