CS689: Machine Learning - Fall 2018

Homework 2

Assigned: Wednesday, Oct 3. Due: Wednesday, Oct 17 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.

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.
- **Code:** The second deliverable is your code. Your code must be Python 3.6 (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.

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. 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 $\epsilon = [\epsilon_1^+, \epsilon_1^-, ..., \epsilon_N^+, \epsilon_N^-]$:

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

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. (*40 points*) **Subgradient Descent for SVC:** Consider the hinge loss formulation of the support vector classifier learning problem shown below:

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} C \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}\mathbf{x}_n^T + b)) + ||\mathbf{w}||_2^2$$

a. (*5 pts*) Derive an expression for a subgradient of the objective function with respect to $\mathbf{w}$ and b.

b. (*20 pts*)Starting from the provided template (svm.py), implement a Scikit-Learn compatible class for this model including `objective`, `subgradient`, `fit`, `predict`, `set_model`, and `get_model` functions. Your fit method must use a subgradient descent procedure. As your answer to this question, describe your approach to learning in detail (stepsize and convergence rules used, any acceleration techniques, etc.) and submit your commented code for auto grading as described above.

c. (*5 pts*) Use your implementation to learn optimal model parameters $\mathbf{w}_{SVC}$ and b_{SVC} on the provided training data using $C = 1$. Report the value of the SVC objective function at the optimal model parameters, as well as the classification error rate on the training data.

d. (*5 pts*) Apply Scikit-learn's linear logistic regression implementation `sklearn.linear_model.LogisticRegression` to the provided training data using $C = 1$ to learn optimal model parameters $\mathbf{w}_{LR}$ and b_{LR}. Report the value of the logistic regression objective function at $\mathbf{w}_{LR}$ and b_{LR}, the value of the SVC objective function at $\mathbf{w}_{LR}$ and b_{LR}, and the classification error rate on the training data.

e. (*5 pts*) How do SVC and logistic regression compare on the provided data? Under what circumstances would you expect SVC to provide lower generalization error on test data (drawn from the same distribution as the training data) than a logistic regression model?

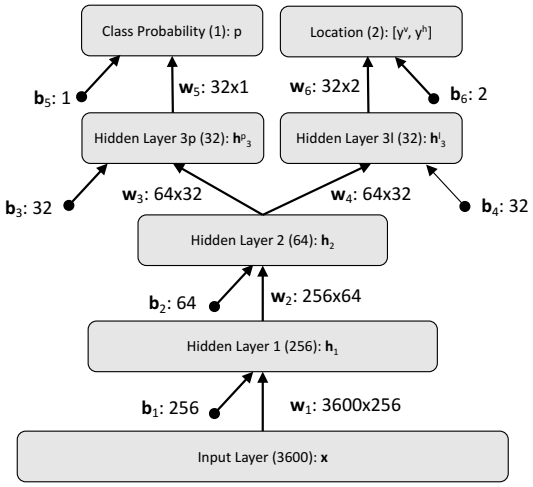
3. (*40 points*) **Multi-Output Neural Networks:** Neural networks are flexible models that can be optimized with respect to many different losses. In this question, you will implement a custom neural network model that simultaneously solves a regression problem (localizing an object in an image), and a binary 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 class probability $f^c(\mathbf{x}, \theta) \in [0, 1]$ and a localization output $f^l(\mathbf{x}, \theta) \in \mathbb{R} \times \mathbb{R}$. The outputs are formally defined as $\mathbf{y} = [y^c, \mathbf{y}^l]$ where $y^c \in \{0, 1\}$ is the class label for data case n and $\mathbf{y}^l = [y^v, y^h]$ are the vertical and horizontal locations of the object. 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 the localization loss (squared error):

$$\theta_* = \arg\min_{\theta} \sum_{n=1}^{N} \alpha L_{ce}(y_n^c, f^c(\mathbf{x}_n, \theta)) + (1 - \alpha)\|\mathbf{y}_n^l - f^l(\mathbf{x}_n, \theta)\|_2^2$$

$$L_{ce}(y, p) = -y \log(p) - (1 - y) \log(1 - p)$$

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 60x60 images, resulting in 3600-long input vectors. All hidden layers will be use RELU non-linearities. The class probability output layer will use a logistic non-linearity. The localization output layer will consist of two linear units. The diagram of the network architecture you will implement is shown below. All layers joined by arrows in the figure are fully connected.



a. (*25 pts*)Starting from the provided template (`nn.py`), implement a Scikit-Learn compatible class for the model shown above including `objective`, `fit`, `predict`, `set_model`, and `get_model` functions. You can develop your implementation using NumPy and PyTorch (version 0.4.1). You may use automatic differentiation methods to develop your implementation. As your answer to this question, describe your approach to learning in detail (parameter initialization, optimization algorithm, stepsize selection and convergence rules used, acceleration techniques, etc.), and submit your commented code for auto grading as described above. (**Hint:** to ensure that learning is numerically stable, make sure to use the PyTorch function `BCEWithLogitsLoss` or `binary_cross_entropy_with_logits` to implement the binary cross entropy portion of the objective function.)

b. (*5 pts*)Using the provided training data set, learn the model using $\alpha = 0.5$. Report the classification error rate and the mean squared error (MSE) of the location predictions obtained on both the training and test sets.

c. (*10 pts*) Use your implementation to perform experiments to assess how changing the value of α affects the classification and location prediction results on the test set. In particular, vary the value of α from 0 to 1 in steps of 0.1 and report both the classification accuracy and the mean squared error of the localization task on the test data. Is there a setting of α where jointly learning these two tasks provides a benefit?