Homework 7

Due May 28th, 2021 by 11:59pm

Instructions: This homework consists of a reading assignment and two coding exercises. Please submit your solutions via Gradescope. Solutions should consist of three files: a PDF containing your solutions to the *non-coding questions*; a Jupyter notebook (`.ipynb` file) and an html print-out (`.html` file) with your solution to the *coding exercise*. **All coding exercises must be completed in Python.** Please be sure to comment the code appropriately. Students are encouraged to discuss homework problems, particularly on Canvas and in the TA hours, but must submit their own solutions.

Reading Assignment

- Review Lecture 7.
- Review Lab 7 (available from the Course Materials page on Canvas).

Exercise 1

In this exercise you will implement your own version of a kernel support vector machine with the squared hinge loss. The kernel support vector machine with the squared hinge loss writes as

$$\min_{\alpha \in \mathbb{R}^n} F(\alpha) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, (K\alpha)_i) + \lambda \alpha^T K \alpha \,, \tag{1}$$

where $(K\alpha)_i$ is the ith entry in the vector $K\alpha$, and

$$\ell(y, t) := (\max\{0, 1 - yt\})^2 \,. \tag{2}$$

Note: Despite the piecewise nature of this loss function, we claim that the quantity

$$\frac{1}{n} \sum_{i=1}^{n} \ell(y_i, (K\alpha)_i)$$

is differentiable at every $\alpha \in \mathbb{R}^n$. For reference on computing the gradient of a piecewise function (and confirming that this gradient exists everywhere), please refer to the math review notes on the course website.

(a) Compute the gradient $\nabla F(\alpha)$ of F.

(b) The polynomial kernel of order p is given by

$$k(x, y) = (x^T y + b)^p.$$

Its parameters are the offset b and order p. Implement this kernel function.

(c) Write a function *computegram* that computes, for any kernel k and set of datapoints $x_1, \ldots, x_n$, the kernel matrix K with $(i, j)^{th}$ entry $k(x_i, x_j)$.

(d) Write a function *kerneleval* that computes, for any kernel k, set of datapoints $x_1, \ldots, x_n$ and a new datapoint $x^\star$, the vector of kernel evaluations $[k(x_1, x^\star), \ldots, k(x_n, x^\star)]^T$.

(e) Write a function *mysvm* that implements the fast gradient algorithm to train the kernel support vector machine with the squared hinge loss. The function takes as input the initial step-size value for the backtracking rule and a stopping criterion based on the norm of the gradient.

(f) Consider the Digits dataset (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html). Download the data. Re-shape the images as vectors. Normalize each image vector such that it has norm 1; note that this is *not* the same as standardizing the data. You may use the function `normalize` from `sklearn.preprocessing`. Perform an 80-20 train-test split of the data.

(g) For each digit $d = 0, ..., 9$ in the Digits dataset, train a kernel SVM classifier using the data X and binary label $y^{(d)}$, defined as

$$y_i^{(d)} = \begin{cases} 1 & y_i = d \\ -1 & y_i \neq d \end{cases}.$$

In this way, we obtain 10 "one vs rest" classifiers, each of which predicts whether a given image comes from class d or not. Use the polynomial kernel with $p = 7$ and $b = 1$. Set $\lambda = 10.0$. For each classifier trained, plot the objective function versus the iteration number.

(h) Now use 5-fold cross-validation to select the best choice of λ for each classifier trained above. Report the value of λ selected for each classifier. Re-train each classifier with its corresponding best choice of λ.

(i) Recall that the SVM prediction is given by

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i).$$

In the multi-class setting, define the class prediction to be given by classifier that maximizes this score, out of the d "one vs rest" classifiers that you have trained. Generate test set predictions and report your final misclassification error.

Exercise 2

Expand on the above analysis in an Azure virtual machine by comparing the performance of the following kernel functions:

- Linear:
$$k(x, y) = x^T y.$$
- Polynomial:
$$k(x, y) = (x^T y + b)^p$$
Try a few settings for the polynomial order p. Set either $b = 1$ or $b = 0$.
- Gaussian / radial basis function (RBF):
$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$
Try a few settings of the scale parameter γ.

  For each kernel investigated, report your final misclassification error on the test set.

  As in the previous submission, please include the following: (1) a screenshot of your running VM in the Azure dashboard; (2) a screenshot of your command line after you `ssh` to the VM; (3) a screenshot of a running Python notebook on your VM, where you have run the following code in a cell block:

```
import os
os.getcwd()
```

  The screenshots may either be embedded in your notebook or submitted as additional files.