TTIC 31020: Introduction to Machine Learning
Autumn 2019

Problem Set #3 (80 Points)

Out: Tuesday, October 29, 2019

Due: Thursday, November 7, 11:59pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want, e.g.,LaTeX (recommended), Word + export as PDF, scan handwritten solutions (note: must be legible!), etc. Please name this document ⟨`firstname-lastname`⟩`-sol3.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. Rename the notebook ⟨`firstname-lastname`⟩`-sol3.ipynb`.

Late submissions: there will be a penalty of 20 points for any solution submitted within 24 hours past the deadline. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include in the `ipynb` file the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers) and references in a legend or in a caption. When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. If there is a mathematical answer, provide it precisely (and accompany by succinct wording, if appropriate).

When submitting code (in Jupyter notebook), please make sure it's reasonably well-documented, runs, and produces all the requested results. If discussion is required/warranted, you can include it either directly in the notebook (you may want to use markdown style for that) or in the PDF writeup.

Collaboration policy: collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, and (2) specify names of student(s) you collaborated with in your writeup.

1 Support Vector Machines

We will consider some details of the dual formulation of SVM, the one in which we optimize the Lagrange multipliers α_i. In class we saw how to derive a constrained quadratic program, which can then be "fed" to an off-the-shelf quadratic program solver. These solvers are usually constructed to handle certain standard formulations of the objective and constraints.

The canonical form of a quadratic program with linear constraints is:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}^\top \boldsymbol{\alpha} \tag{1}$$

$$\text{such that: } \mathbf{A}\boldsymbol{\alpha} \le \mathbf{a} \tag{2}$$

$$\mathbf{B}\boldsymbol{\alpha} = \mathbf{b} \tag{3}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ contains the unknown variables to be solved for. The matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ and vector $\mathbf{f} \in \mathbb{R}^n$ specify the quadratic objective. The matrix $\mathbf{A} \in \mathbb{R}^{k_{ineq}\times n}$ and vector $\mathbf{a} \in \mathbb{R}^{k_{ineq}}$ specify k_{ineq} inequality constraints. Similarly, $\mathbf{B} \in \mathbb{R}^{k_{eq}\times n}$ and vector $\mathbf{b} \in \mathbb{R}^{k_{eq}}$ specify k_{eq} equality constraints. Note that you can express a variety of inequality constraints by adding rows to $\mathbf{A}$ and elements to $\mathbf{a}$; think how you would do so to express, e.g., a "greater than or equal to" constraint.

Problem 1 [20 points]
The primal optimization problem for an SVM is shown below:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) - w_0\right)\right) \right\}$$

Describe in detail how you would define $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{a}$, $\mathbf{B}$, and $\mathbf{b}$ to create a quadratic program for the **dual optimization problem** (see the Lecture 10 slides) for the kernel SVM given a kernel function $K(\cdot,\cdot)$ corresponding to the dot product in $\boldsymbol{\phi}$ space and n training examples $(\mathbf{x}_i, y_i)$. Hint: We intentionally used the same variable n to describe the dimensions of the notation in the canonical form of a QP because there will be one unknown variable α_i in the QP corresponding to each training example $(\mathbf{x}_i, y_i)$.

End of problem 1

Next, we will consider finding the value of b, the bias weight for a kernel SVM

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i:\alpha_i>0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

where α_i are the coefficients for the support vectors in the training data.

Problem 2 [10 points]
Suppose you have solved for α_i in the SVM classifier. Explain how you can exactly calculate b.

End of problem 2

2 Twitter sentiment analysis with SVMs

In this section we will consider the problem of predicting *sentiment* from tweets regarding US airline service quality.[1] We will formulate this as a binary classification problem where one class consists of positive or neutral tweets and the other class consists of negative tweets. The provided dataset of a few thousand tweets has been manually labeled to reflect these binary sentiment labels.

We will represent each tweet by a feature vector based on word occurrences. This representation for documents is sometimes called "bag of words" since it discards the ordering of words and treats them as interchangeable "tokens" in a bag (document). To skip non-trivial engineering issues, we will rely on existing packages to do the low-level processing and feature extraction for us. The details are addressed in the notebook.

Problem 3 [10 points]
Fill in missing pieces of the linear SVM code and run the test code (`test_linear_SVM`) for various hyperparameter values; include the generated figures.

End of problem 3

Problem 4 [20 points]
Fill in missing pieces of the kernel SVM code and run the test code (`test_rbf_SVM`) for various hyperparameter values; include the generated figures.

End of problem 4

Problem 5 [20 points]
Using your code, train your tweet classifier on the training set (`train.csv`) and tune on the validation set (`val.csv`). Include at least the basic linear SVM classifier; feel free to experiment with hyperparameters (training duration, regularization) and with the representation (i.e., choice of kernel). Evaluate your single best model on the development test set (`devtest.csv`) and report your final results. What hyperparameter values lead to the best performance? Do you see a large gap between the validation and devtest accuracies? They are drawn from the same distribution, so if you see sizable differences, that might be due to overfitting to the validation set in your hyperparameter tuning.

End of problem 5

[1]As may be expected for this topic, the data contains some unsavory expressions, left unfiltered.