TTIC 31020: Introduction to Machine Learning
Autumn 2020

Problem Set #3

Out: Friday October 30, 2018

Due: Tuesday November 10, 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 *link to a Colab notebook*. Rename the notebook ⟨`firstname-lastname`⟩`-sol3.ipynb`.

Late submissions: there will be a penalty of 25 points for any solution submitted within 24 hours past the deadlne. 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 is precisely (and accompany by only succinct words, if appropriate).

When submitting code (in Jupyter notebook), please make sure it's reasonably 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 the 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, (2) specify names of student(s) you collaborated with in your writeup.

1 Support Vector Machines

Now 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.

Specifically the canonical form of a quadratic program with linear constraints is, mathematically:

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

$$\text{such that: } \mathbf{A}\cdot\boldsymbol{\alpha} \leq \mathbf{a}, \tag{2}$$

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

$$\tag{4}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^N$, where N is the number of training examples, 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 $\mathbf{B}$ and elements to $\mathbf{b}$; think how you would do it to express, e.g., a "greater or equal" constraint.

Problem 1 **[20 points]**
Describe in detail how you would compute $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{a}$, $\mathbf{B}$, and $\mathbf{b}$, to set up the dual optimization problem for the kernel SVM

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

given a kernel function $K(\cdot,\cdot)$ corresponding to the dot product in $\boldsymbol{\varphi}$ space, and N training examples $(\mathbf{x}_i, y_i)$

End of problem 1

Next, we will consider finding the value of b, the threshold (bias) term 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 exactly you can calculate b.

End of problem 2

2 Perceptron

Problem 3 **[25 points]**
In this problem, we will flesh out the analysis of the convergence rate of the Perceptron algorithm.

To do so, let's make the following assumptions. Suppose that our data lies in the unit ball and is linearly separable by a *unit-norm, origin-containing* hyperplane $\mathbf{w}^*$ with margin γ. This means that for every training data point $\mathbf{x}_i$ and corresponding label y_i, we have:

$$\|\mathbf{x}_i\| \leq 1$$
$$y_i \mathbf{w}^* \cdot \mathbf{x}_i \geq \gamma$$

Note that we can always ensure (by manipulating the data) that $\|\mathbf{x}_i\| \leq 1$, and $b = 0$ (think why this is true).

Let M denote the number of mistakes Perceptron makes. Recall from lecture that we want to show that $M \leq 1/\gamma^2$.

1. Without loss of generality, we'll assume that all our datapoints have label $+1$. Explain why this is a valid assumption.
2. In showing the convergence of Perceptron, we'll need two lemmas. The first roughly says, "we make decent progress towards a good solution after each mistake." The second roughly says, "the norm of our solution vector isn't too big." Think about why these might be good things to show (no need to write anything here).
3. OK, let's formalize the statement of the first lemma. Let w_i denote our weight vector after we've made i mistakes. Show that:

   $$\mathbf{w_{i+1}} \cdot \mathbf{w}^* - \mathbf{w_i} \cdot \mathbf{w}^* \geq \gamma$$

   Then, think about how this mathematical statement fits with the English description above.
4. Before we move on to the next lemma, let's try a naive bound to capture the idea of the second lemma. Using the triangle inequality, bound $\|\mathbf{w_M}\|$.

   Hint: The triangle inequality states that for any two vectors $\mathbf{a}$ and $\mathbf{b}$:

   $$\|\mathbf{a} + \mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$$

5. Let's now strengthen this. In general, the following "squared triangle inequality" doesn't hold (if you're unconvinced, find some counterexample):

$$\|\mathbf{a}+\mathbf{b}\|^2 \leq \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2$$

However, this is true for the iterates of our Perceptron algorithm! To see this, show that if $\mathbf{a}\cdot\mathbf{b} \leq 0$, then the above "squared triangle inequality" is actually true. Use this to show that:

$$\|\mathbf{w_M}\| \leq \sqrt{M}$$

Contrast this with the bound we saw in the previous part. the w* is automatically norm =1?

6. Let's combine everything. Get an upper and lower bound on the quantity $\mathbf{w}^* \cdot \mathbf{w_M}$, simplify appropriately, and conclude that $M \leq 1/\gamma^2$. Here, it should become clear why the bound from part (4) is not sufficient to tell us anything interesting.

Hint: You might find it useful to know the *Cauchy-Schwarz inequality*, which states that for any two vectors $\mathbf{a}$ and $\mathbf{b}$:

$$\mathbf{a}\cdot\mathbf{b} \leq \|\mathbf{a}\| \cdot \|\mathbf{b}\|$$

The proof of this is straightforward:

$$\begin{aligned}\mathbf{a}\cdot\mathbf{b} &= \|\mathbf{a}\| \cdot \|\mathbf{b}\| \cdot \cos\left(\theta_{\mathbf{a},\mathbf{b}}\right) \\ &\leq \|\mathbf{a}\| \cdot \|\mathbf{b}\|\end{aligned}$$

End of problem 3

3 Text sentiment analysis with SVMs

In this section we will consider the problem of predicting *sentiment* (positive/neutral or negative) from tweets regarding US arline service quality.[1] The data set of a few thousand tweets has been manually labeled to reflect the binary sentiment labels.

We will represent each tweet by a feature vector based on word occurences. This representaton for documents is sometimes called "bag of words" in natural language processing, 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 4 [25 points]

Fill in missing pieces of the general SVM code, and run the tests with and without kernels; include figures produced by `test_linear_SVM` and `test_rbf_SVM`.

End of problem 4

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

Problem 5 **[20 points]**

Using your code, train and evaluate (on validation) your classifier for the tweets. Include at least the basic linear SVM classifier; feel free to experiment with learning hyper-parameters (training duration, regularization) and with the representation (i.e., choice of kernel). Submit your results (again, including at least the linear SVM predictions) on the test set to the Kaggle competition (**whose link is on Canvas**).

You must make at least one valid submission, but feel free to try to improve your predictions and submit multiple times.

End of problem 5