Matthew Leifer
matthewleifer@college.harvard.edu
CS181-S17

Assignment #3
Due: 5:00pm March 24, 2016

Collaborators: John Doe, Fred Doe

Homework 3: Max-Margin and SVM

Introduction

This homework assignment will have you work with max-margin methods and SVM classification. The aim of the assignment is (1) to further develop your geometrical intuition behind margin-based classification and decision boundaries, (2) to explore the properties of kernels and how they provide a different form of feature development from basis functions, and finally (3) to implement a basic Kernel based classifier.

There is a mathematical component and a programming component to this homework. Please submit your PDF and Python files to Canvas, and push all of your work to your GitHub repository. If a question requires you to make any plots, like Problem 3, please include those in the writeup.

Problem 1 (Fitting an SVM by hand, 7pts)

For this problem you will solve an SVM without the help of a computer, relying instead on principled rules and properties of these classifiers.

Consider a dataset with the following 7 data points each with $x \in \mathbb{R}$:

$$\{(x_i, y_i)\}_i = \{(-3, +1), (-2, +1), (-1, -1), (0, -1), (1, -1), (2, +1), (3, +1)\}$$

Consider mapping these points to 2 dimensions using the feature vector $\boldsymbol{\phi}(x) = (x, x^2)$. The hard margin classifier training problem is:

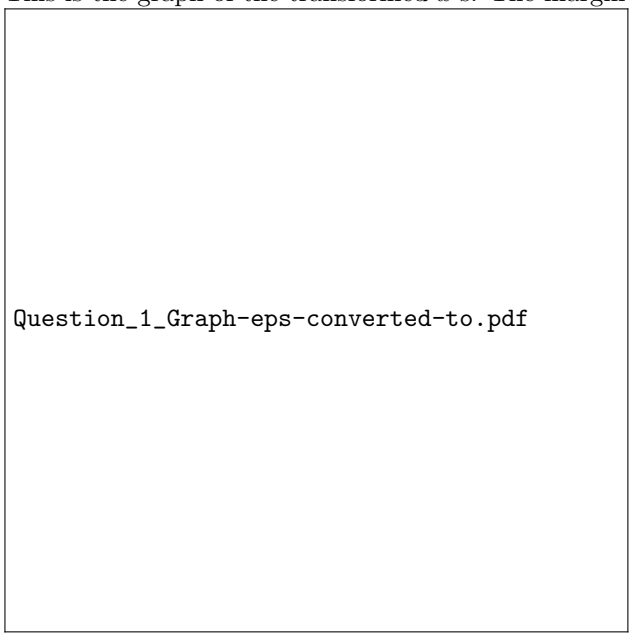
$$\begin{aligned} &\min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad &y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0) \geq 1, \ \forall i \in \{1, \ldots, n\} \end{aligned} \tag{1}$$

The exercise has been broken down into a series of questions, each providing a part of the solution. Make sure to follow the logical structure of the exercise when composing your answer and to justify each step.

1. Plot the training data in $\mathbb{R}^2$ and draw the decision boundary of the max margin classifer.
2. What is the value of the margin achieved by the optimal decision boundary?
3. What is a vector that is orthogonal to the decision boundary?
4. Considering discriminant $h(\boldsymbol{\phi}(x); \mathbf{w}, w_0) = \mathbf{w}^\top \boldsymbol{\phi}(x) + w_0$, give an expression for *all possible* $(\mathbf{w}, w_0)$ that define the optimal decision boundary. Justify your answer.
5. Consider now the training problem (1). Using your answers so far, what particular solution to $\mathbf{w}$ will be optimal for this optimization problem?
6. Now solve for the corresponding value of w_0, using your general expression from part (4.) for the optimal decision boundary. Write down the discriminant function $h(\boldsymbol{\phi}(x); \mathbf{w}, w_0)$.
7. What are the support vectors of the classifier? Confirm that the solution in part (6.) makes the constraints in (1) binding for support vectors.

Solution

1. This is the graph of the transformed x's. The margin boundary corresponds to $y = 2.5$.



2. The margin is 1.5.

3. Because the decision boundary corresponds to $y = 2.5$, an orthogonal vector is $[0, 1]$

4. By definition, the value of the discriminant function for a point on the decision boundary must be 0.
   So, $h(\phi(x); \mathbf{w}, w_0) = \mathbf{w}^T \phi(x) + w_0 = [0, C] \cdot [\sqrt{2.5}, 2.5]^T + w_0 = 0$
   $2.5c + w_0 = 0$
   $w_0 = -2.5c$
   So, $(\mathbf{w}, w_0) = ([0, C], -2.5C)$

5. The optimal form of $\mathbf{w}$ is the one such that the constraint in equation (1) is binding. Thereofre, the C that optimized it is the one that makes that constraint equal to 1 for the x's on the margin and that the value of the discriminant is as small as possible without making any of the (correctly classified) data points less than 1 away from the margin. Those are (-1,1), (1,1), (-2, 4), and (2, 4).
   So, for $(\pm 2, 4)$, $1(4C - 2.5C) = 1$, which means $C = \frac{2}{3}$ and for $(\pm 1, 1)$, $-1(C - 2.5C) = 1$, which means $C = \frac{2}{3}$

   $\mathbf{w} = [0, \frac{2}{3}]$

6. Because $w_0 = -2.5C$ and $C = \frac{2}{3}$, $w_0 = -\frac{5}{3}$
   $h(x; \mathbf{w}, w_0) = \frac{2}{3}x^2 - \frac{5}{3}$

7. The support vectors are $x = \pm 1, \pm 2$, which as show in (4) make sthe constraint binding.

Problem 2 (Composing Kernel Functions, 10pts)

A key benefit of SVM training is the ability to use kernel functions $K(\mathbf{x}, \mathbf{x}')$ as opposed to explicit basis functions $\boldsymbol{\phi}(\mathbf{x})$. Kernels make it possible to implicitly express large or even infinite dimensional basis features. We do this by computing $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ directly, without ever computing $\boldsymbol{\phi}(\mathbf{x})$.

When training SVMs, we begin by computing the kernel matrix $\mathbf{K}$, over our training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The kernel matrix, defined as $K_{i,i'} = K(\mathbf{x}_i, \mathbf{x}_{i'})$, expresses the kernel function applied between all pairs of training points.

In class, we saw Mercer's theorem, which tells us that any function K that yields a positive semi-definite kernel matrix forms a valid kernel, i.e. corresponds to a matrix of dot-products under *some* basis $\boldsymbol{\phi}$. Therefore instead of using an explicit basis, we can build kernel functions directly that fulfill this property.

A particularly nice benefit of this theorem is that it allows us to build more expressive kernels by composition. In this problem, you are tasked with using Mercer's theorem and the definition of a kernel matrix to prove that the following compositions are valid kernels, assuming $K^{(1)}$ and $K^{(2)}$ are valid kernels. Recall that a positive semi-definite matrix $\mathbf{K}$ requires $\mathbf{z}^\top \mathbf{K} \mathbf{z} \geq 0, \ \forall \ \mathbf{z} \in \mathbb{R}^n$.

1. $K(\mathbf{x}, \mathbf{x}') = c\, K^{(1)}(\mathbf{x}, \mathbf{x}') \quad \text{for } c > 0$
2. $K(\mathbf{x}, \mathbf{x}') = K^{(1)}(\mathbf{x}, \mathbf{x}') + K^{(2)}(\mathbf{x}, \mathbf{x}')$
3. $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\, K^{(1)}(\mathbf{x}, \mathbf{x}')\, f(\mathbf{x}') \quad$ where f is any function from $\mathbb{R}^m$ to $\mathbb{R}$
4. $K(\mathbf{x}, \mathbf{x}') = K^{(1)}(\mathbf{x}, \mathbf{x}')\, K^{(2)}(\mathbf{x}, \mathbf{x}')$

   [Hint: Use the property that for any $\boldsymbol{\phi}(\mathbf{x})$, $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ forms a positive semi-definite kernel matrix.]
5. (a) The exp function can be written as,

$$\exp(x) = \lim_{i \to \infty} \left(1 + x + \cdots + \frac{x^i}{i!}\right).$$

   Use this to show that $\exp(xx')$ (here $x, x' \in \mathbb{R}$)) can be written as $\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x')$ for some basis function $\boldsymbol{\phi}(x)$. Derive this basis function, and explain why this would be hard to use as a basis in standard logistic regression.

   (b) Using the previous identities, show that $K(\mathbf{x}, \mathbf{x}') = \exp(K^{(1)}(\mathbf{x}, \mathbf{x}'))$ is a valid kernel.
6. Finally use this analysis and previous identities to prove the validity of the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma^2}\right)$$

Solution

1. $K(\mathbf{x}, \mathbf{x}') = cK^{(1)}(\mathbf{x}, \mathbf{x}')$ for $c > 0$
   $\mathbf{z}^T K \mathbf{z} = \mathbf{z}^T cK\mathbf{z} = c(\mathbf{z}^T K(1)\mathbf{z})$ and because c is greater than zero and $K^{(1)}$ is a valid kernel matrix $c(\mathbf{z}^T K^{(1)}\mathbf{z})$ is always greater than 0 and so this is a valid kernel.

2. $K(x, x') = K^{(1)}(\mathbf{x}, \mathbf{x}') + K^{(2)}(\mathbf{x}, \mathbf{x}')$
   $\mathbf{z}^T K z = \mathbf{z}^T (K^{(1)} + K^{(2)})\mathbf{z} = \mathbf{z}^T K^{(1)}\mathbf{z} + \mathbf{z}^T K^{(2)}\mathbf{z}$ and because $K^{(1)}$ and $K^{(2)}$ are both valid kernel functions then $\mathbf{z}^T K^{(1)}\mathbf{z} + \mathbf{z}^T K^{(2)}\mathbf{z}$ is always greater than zero and so K is also a valid kernel.

3. $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})K^{(1)}f(\mathbf{x}')$, which can be rewritten as $f(\mathbf{x})\phi(\mathbf{x})^T\phi(\mathbf{x}')f(\mathbf{x}')$ per Mercer's theorem. Then, using the hint in part 4, this can be rewritten as $g(\mathbf{x})^T g(\mathbf{x}')$ where $g(\mathbf{x}) = f(\mathbf{x})\phi(\mathbf{x})$ which is a valid kernel.

4. $K(\mathbf{x}, \mathbf{x}') = K^{(1)}(\mathbf{x}, \mathbf{x}')K^{(2)}(\mathbf{x}, \mathbf{x}')$

   $= \boldsymbol{\phi}_1(\mathbf{x})^T\boldsymbol{\phi}_1(\mathbf{x}')\boldsymbol{\phi}_2(\mathbf{x})^T\boldsymbol{\phi}_2(\mathbf{x}')$

   $= \sum_{i=1}^{N} \phi_{1i}(x)\phi_{1i}(x') \cdot \sum_{j=1}^{M} \phi_{2j}(x)\phi_{2j}(x')$

   $= \sum_{i=1}^{N}\sum_{j=1}^{M} \phi_{1i}(x)\phi_{2j}(x)\phi_{1i}(x')\phi_{2j}(x')$

   and so we can construct a new function ϕ_3 such that this sum equals
   $\sum_{k=1}^{MN} \phi_{3k}(x)\phi_{3k}(x')$
   $= \boldsymbol{\phi}_3(x)^T\boldsymbol{\phi}_3(x')$, which is indeed a valid kernel.

5. (a) The Taylor series for $\exp(x, x') = \sum_{i=0}^{\infty} \frac{(xx')^i}{i!}$. Therefore, if we define $\phi : \mathbb{R} \to \mathbb{R}^\infty$ such that $\phi(x) = [1x\frac{x^2}{\sqrt{2!}}...\frac{x^n}{\sqrt{n!}}]^T$ and so when we take $\phi(x)^T \cdot \phi(x')$ we recover the Taylor series for $\exp(x, x')$. Because this is an infinite dimensional basis function, it would be hard to use in logistic regression.

   (b) $K(\mathbf{x}, \mathbf{x}')$

   $= \exp(K^1(\mathbf{x}, \mathbf{x}'))$

   $= \sum_{i=0}^{\infty} \frac{K^{(1)}(\mathbf{x}, \mathbf{x}')^i}{i!}$

   $= \sum_{i=0}^{\infty} \frac{(\phi(x)^T\phi(x'))^i}{i!}$

   By part (4), we know that $K^{(1)}(\mathbf{x}, \mathbf{x}')^i$ is a valid kernel function because $K^{(1)}$ is.

   By part (1), because $i!$ is always greater than 0 for all whole numbers, and so $\frac{K^{(1)}(\mathbf{x}, \mathbf{x}')^i}{i!}$ is a valid kernel function.

   By part(2), the entire sum is a valid kernel function because each individual term of the sum is a valid kernel function. Therefore we can conclude that $\exp(K^1(\mathbf{x}, \mathbf{x}'))$ is a valud kernel.

6. $K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-||\mathbf{x}-\mathbf{x}'||_2^2}{2\sigma^2}\right)$

   $= \exp(\frac{-||\mathbf{x}||_2^2}{2\sigma^2})\exp(\frac{-||\mathbf{x}'||_2^2}{2\sigma^2})\exp(\frac{\mathbf{x}^T\mathbf{x}'}{\sigma^2})$

   The third term in this product is a valid kernel per part 5(b) because $\mathbf{x}^T\mathbf{x}'$ is a valid kernel (the trivial kernel), and the first two terms keep this a valid kernel because they are both always greater than 0 and by part (1) this keeps the product a valid kernel.

Problem 3 (Scaling up your SVM solver, 10pts (+opportunity for extra credit))

For this problem you will build a simple SVM classifier for a binary classification problem. We have provided you two files for experimentation: training *data.csv* and validation *val.csv*.

- First read the paper at http://www.jmlr.org/papers/volume6/bordes05a/bordes05a.pdf and implement the Kernel Perceptron algorithm and the Budget Kernel Perceptron algorithm. Aim to make the optimization as fast as possible. Implement this algorithm in *problem3.py*.

  [Hint: For this problem, efficiency will be an issue. Instead of directly implementing this algorithm using numpy matrices, you should utilize Python dictionaries to represent sparse matrices. This will be necessary to have the algorithm run in a reasonable amount of time.]

- Next experiment with the hyperparameters for each of these models. Try seeing if you can identify some patterns by changing β, N (the maximum number of support vectors), or the number of random training samples taken during the Randomized Search procedure (Section 4.3). Note the training time, training and validation accuracy, and number of support vectors for various setups.

- Lastly, compare the classification to the naive SVM imported from scikit-learn by reporting accuracy on the provided validation data. *For extra credit, implement the SMO algorithm and implement the LASVM process and do the same as above.*[a]

We are intentionally leaving this problem open-ended to allow for experimentation, and so we will be looking for your thought process and not a particular graph. Visualizations should be generated using the provided code. You can use the trivial $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ kernel for this problem, though you are welcome to experiment with more interesting kernels too.

In addition, provide answers the following reading questions **in one or two sentences for each**.

1. In one short sentence, state the main purpose of the paper.
2. Describe each of the parameters in Eq. 1 in the paper
3. State, informally, one guarantee about the Kernel perceptron algorithm described in the paper.
4. What is the main way the budget kernel perceptron algorithm tries to improve on the perceptron algorithm?
5. (*if you did the extra credit*) In simple words, what is the theoretical guarantee of LASVM algorithm? How does it compare to its practical performance?

[a] Extra credit only makes a difference to your grade at the end of the semester if you are on a grade boundary.

Solution

1. The goal of the paper is to create a machine learning classifier that can reach the accuracy of SVMs at a fraction of the cost with respect to both time and space.
2. w' is some set of learned weights. ϕ is a basis function. b is the bias of the model, which we usually represent as w_0.
3. The perceptron will converge to a boundary after a finite number of mistakes (i.e. inserting a finite number of support vectors).
4. The budget perceptron keeps the number of support vector below a certain, predetermined cap N so that the algorithm does not use up too much memory (see step 4b in the algorithm description in the paper), which could happen in the regular Kernel Perceptron algorithm.
5. In the paper, the authors show that the LASVM algorithm will converge to SVM within a finite number

of steps. In practice, it performs very well. As discussed on page 1603 of the paper, LASVM reaches the performance of an SVM after only a single epoch of training.

First off, just so the graphs were a little less jagged, I changed the increement from 0.05 to 0.005 to make them smoother and to give a better sense of what the actual decision boundary is. Because of the randomized nature of the Kernel Percpetron and the Budget Kernel Perceptron the results of running these algorithms often varied significantly between trials.
For the Kernel Perceptron algorithm on 20000 training Samples, it took my computer anywhere between 5 and 10 sseconds to train, and the training and validation accuracy were both always incredibly high and in my trials were neve below 97.98% and often were closer to 99.99% accurate and had anywhere between 65 and 256 support vectors. By and large, the large N and β were, the more accurate the Budget Kernel Perceptron was but the more time it took to train that algorithm. Here's a table with some of the results.

β	N	Train_acc	Valid_acc
0.01	20	.8530	.8488
0.01	50	.9964	.9971
0.01	100	.996	.996
0.01	200	.9999	.99989
0.05	20	.9679	.9664
0.05	100	.8476	.84689
0.10	20	.7687	.8812
0.10	50	.9056	.9051
0.10	100	.8786	.8786
0.10	200	.9955	.9953
0.20	20	.6868	.6842
0.20	50	.8077	.8048
0.20	100	.9133	.9113
0.20	200	.9606	.9584

Calibration [1pt]

Approximately how long did this homework take you to complete? Somewhere between 10 and 12 hours.