Matheus C. Fernandes
fernandes@g.harvard.edu
CS181-S17

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

Collaborators:
Bolei Deng
Callin Switzer
Joe Sedlak

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 $\phi(x) = (x, x^2)$. 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

Problem 1

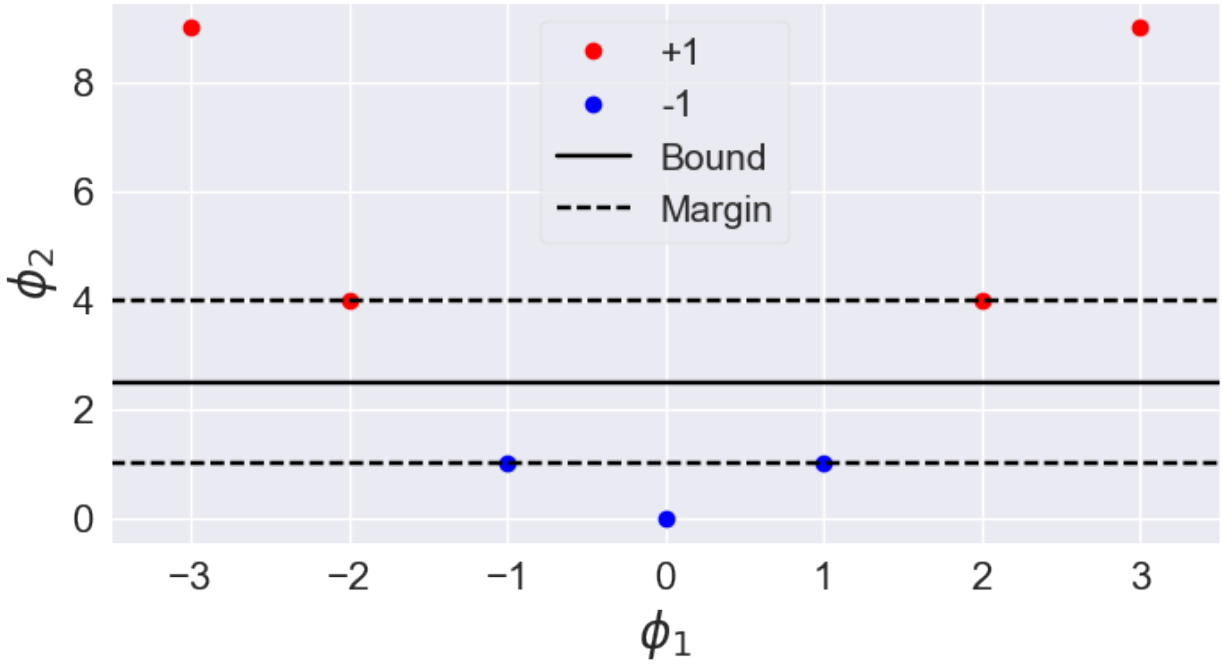

Figure 1: Training data in $\mathbb{R}^2$ with decision boundary and margin.

Problem 2

The value of the margin achieved by the optimal decision boundary is

$$\frac{y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0)}{||\mathbf{w}||} = \boxed{1.5}$$

for an $\mathbf{x}_i$ laying on the margin. Furthermore, for the above result we clearly see that the margin width is 3.

Problem 3

The vector orthogonal to the decision boundary is given by $\mathbf{w}$ and in this case is of arbitrary magnitude such that:

$$\boxed{\mathbf{w} = \lambda[0, 1], \forall \lambda \in \mathbb{R}}$$

Problem 4

We know that there are 4 support vectors that can help us define all of the parameters we need. We also know that the margin does not depend on $\phi_1 = x$, meaning $w_1 = 0$ is always satisfied from $\mathbf{w} = [w_1, w_2]$.

Now, we must find a relationship between w_1 and w_0. To do that, we resort to the margin equation:

$$\frac{y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0)}{||\mathbf{w}||} = 1.5.$$

Furthermore, we know that for a point a on the margin, the numerator becomes 1, thus yielding the equation:

$$\frac{1}{||\mathbf{w}||} = \frac{1}{\sqrt{0^2 + w_2^2}} = 1.5 \Rightarrow w_2 = \frac{2}{3}.$$

Now, we also know that for a support vector, the discriminant $h(\boldsymbol{\phi}(x); \mathbf{w}, w_0) = \pm 1$, depending on whether the support vector is located above or below the margin. Therefore, assuming the SV along the right top margin, we obtain the equation constraint

$$-2w_1 + 4w_2 + w_0 = 1$$

where, from graphical inspection, we know that $w_1 = 0$. Therefore, we have a relationship in that $w_0 = 1 - 4w_2$. So for a $w_2 = \frac{2}{3}$ we have that $w_0 = -\frac{5}{3}$

Problem 5

Problem 6

Problem 7

The support vectors of the classifier are the following $\{(x_i, y_i)\}_i$:

$$\{(x, y)\}_{SV} = \{(-2, +1), (-1, -1), (1, -1), (2, +1)\}$$

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 $\phi(\mathbf{x})$. Kernels make it possible to implicitly express large or even infinite dimensional basis features. We do this by computing $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$ directly, without ever computing $\phi(\mathbf{x})$.

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 ϕ. 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. $K(\mathbf{x}, \mathbf{x}') = c\, K^{(1)}(\mathbf{x}, \mathbf{x}') \quad$ 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 $\phi(\mathbf{x})$, $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$ 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 $\phi(x)^\top \phi(x')$ for some basis function $\phi(x)$. 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

Problem 1

For this problem there are two ways to show, 1 way is by multiplying it out by a vectors on both sides such that:

$$\mathbf{z}^\top c\mathbf{K}(\mathbf{x}, \mathbf{x}')^{(1)}\mathbf{z} = c\mathbf{z}^\top \mathbf{K}(\mathbf{x}, \mathbf{x}')^{(1)}\mathbf{z} \geq 0.$$

Another way to show this is by looking at the eigenvalues such that if we multiply the kernel by a vector, we should get the same value of the vector multiplied by a stretch know as the eigenvalue, where if the eigenvalue is greater than or equal to 0, than the matrix is semi-positive definite, such that:

$$c\mathbf{K}(\mathbf{x}, \mathbf{x}')^{(1)}\mathbf{z} = c\alpha_i\mathbf{z} \quad \text{where} \quad \alpha_i \geq 0$$

and if $c \geq 0$, then $c\alpha_i \geq 0$.

■

Problem 2

If each individual $\mathbf{K}$ is a

Problem 3

Problem 4

Problem 5

Part (a)

For this problem we know the series expansion for the exponential function above. However, if we need to separate the series expansion into a multiplication of expansions, we obtain that:

$$\exp(\mathbf{x}\mathbf{x}') = \begin{bmatrix} 1 & x & \frac{x^2}{\sqrt{2!}} & \dots & \frac{x^i}{\sqrt{i!}} \end{bmatrix} \begin{bmatrix} 1 \\ x' \\ \frac{x'^2}{\sqrt{2!}} \\ \dots \\ \frac{x'^i}{\sqrt{i!}} \end{bmatrix}$$

such that

$$\phi(x) = \sum_{i=0}^{\infty} \frac{x^i}{\sqrt{i!}}$$

and

$$\exp(\mathbf{x}\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}').$$

Where from hint of part 4, we know that this yields a valid positive semi-definite kernel matrix.

■

Part (b)

We know that if

$$\mathbf{K}(\mathbf{x}, \mathbf{x}')^{(1)} = \boldsymbol{\phi}(\mathbf{x})^{(1)\top} \boldsymbol{\phi}(\mathbf{x}')^{(1)}$$

then,

$$\exp(\boldsymbol{\phi}(\mathbf{x})^{(1)\top}\boldsymbol{\phi}(\mathbf{x}')^{(1)}) = \begin{bmatrix} 1 & \boldsymbol{\phi}(\mathbf{x})^{(1)\top} & \frac{\boldsymbol{\phi}(\mathbf{x})^{(1)\top 2}}{\sqrt{2!}} & \dots & \frac{\boldsymbol{\phi}(\mathbf{x})^{(1)\top i}}{\sqrt{i!}} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{\phi}(\mathbf{x}')^{(1)\top} \\ \frac{\boldsymbol{\phi}(\mathbf{x}')^{(1)\top 2}}{\sqrt{2!}} \\ \dots \\ \frac{\boldsymbol{\phi}(\mathbf{x}')^{(1)\top i}}{\sqrt{i!}} \end{bmatrix}$$

which we can then use such that

$$\boldsymbol{\phi}(x)^{(2)} = \sum_{i=0}^{\infty} \frac{x^i}{\sqrt{i!}}.$$

Thus,

$$\exp(\boldsymbol{\phi}(\mathbf{x})^{(1)\top}\boldsymbol{\phi}(\mathbf{x}')^{(1)}) = \boldsymbol{\phi}(\boldsymbol{\phi}(\mathbf{x})^{(1)})^{(2)\top}\boldsymbol{\phi}(\boldsymbol{\phi}(\mathbf{x}')^{(1)})^{(2)}$$

Where from hint of part 4, we know that this yields a valid positive semi-definite kernel matrix.

■

Problem 6

From the identity:

$$\begin{aligned} ||x-y||^2 &= \langle x-y, x-y\rangle \\ &= \|x\|^2 + \langle x, y\rangle + \langle y, x\rangle + \|y\|^2 \\ &\leq \|x\|^2 + 2|\langle x, y\rangle| + \|y\|^2 \\ &\leq \|x\|^2 + 2\|x\|\|y\| + \|y\|^2 \\ &= ||x||^2 + ||y||^2 + 2xy \end{aligned}$$

Then we can take:

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

and decompose it into

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

which we know from previous identities that

$$\exp(\mathbf{x}\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\phi}(\mathbf{x}')$$

and that $\exp$ is a valid kernel. Thus also from problem 4 we know that multiplying valid kernels yields a total valid kernel for the above expression.

■

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 purpose of this paper is to survey different ways to improve fast kernel classifiers and introduce various ways to allocate attention to the training data without compromising the fact that we are taking into consideration all of the examples.

(2)

The w' is a weight vector parameter consisting of weights that tells us how important each feature of the vector $\phi(x)$ is. The b parameter is known as the bias parameter and allows the data to be corrected in space by a constant denoted by this parameter.

(3)

When a solution exists, the perceptron algorithm is guaranteed to converge after a finite number of iterations, or equivalently after inserting a finite number of support vectors. Meaning, the there is a mathematical proof that shows we can achieve the exact convergence of a solution without requiring infinite iterations of the code, but instead only a finite number.

(4)

The budget kernel perceptron algorithm tries to improve on the perceptron algorithm by improving it's performance on noisy datasets – this is due to the fact that the number of support vectors increases very rapidly and potentially causes over fitting and poor convergence. This avoids such problem by educatedly removing support vectors from the list that have a large bound given by the argmax term in 4b.

Hyperparameters Experimentation

By playing with the hyperparameters, I noticed that changes in the number of support vectors, training time, and accuracy occur. When tweaking the β parameter I noticed that as it is increased, we see that the accuracy of the budget kernel algorithm is decreased. Furthermore, the time it takes for it to converge is also increased as now we have to check more parameters through the for loop. This analysis was conducted holding N=100 and number of samples =1000 fixed. Another thing noticed was that the number of support vectors is also increased as β is increased. However, because N=100, the maximum number of support vectors is always lower or equal to 100 for any value of β.

Another parameter that was tweaked was N, the number of maximum support vector for any given iteration. Here, as expected, the lower N, the lower the accuracy as well as the time it takes to train. As N is increased, the accuracy is also increased up to 99.71% for N=100. Here we also assume remain constant at β=20. Another thing noticed, is that when N is small, the variance in the accuracy varies largely between different runs. This is expected as we are sampling random selected points and each run we are sampling different random points generated trough the indexes.

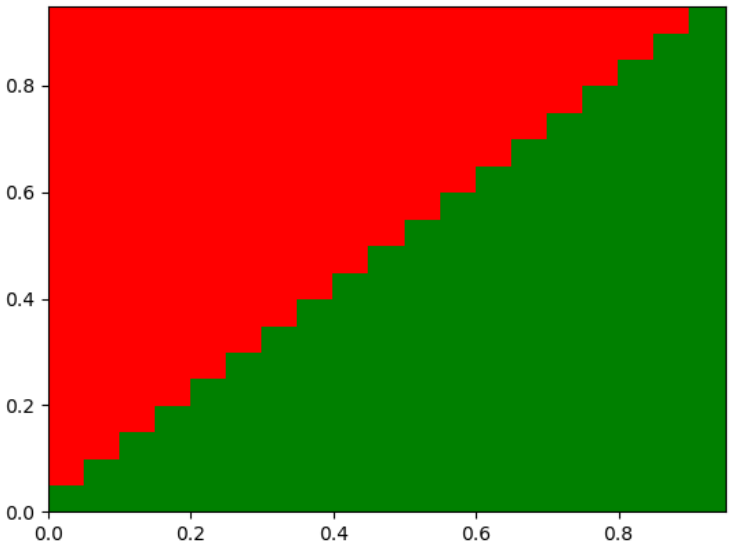

Figure 2: Kernel Optimal result with 100 % accuracy

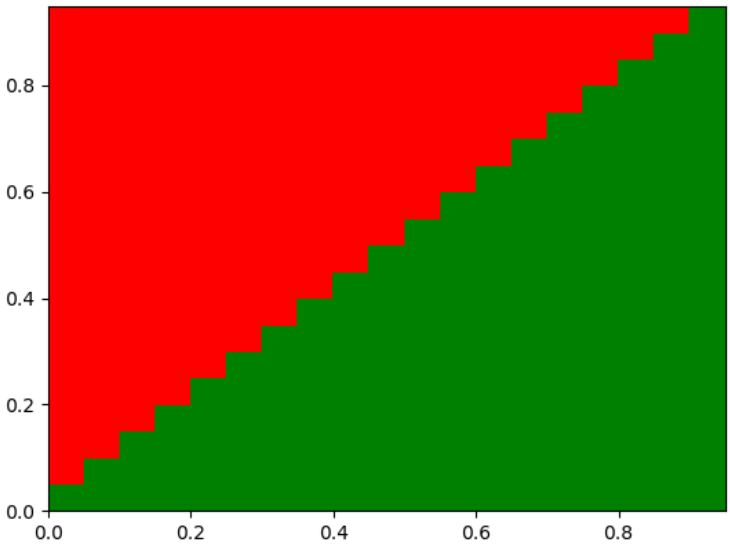

Figure 3: Budget Kernel Optimal result with 100 % accuracy

Calibration [1pt]

Approximately how long did this homework take you to complete?