Russell Kunes
russellkunes@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:

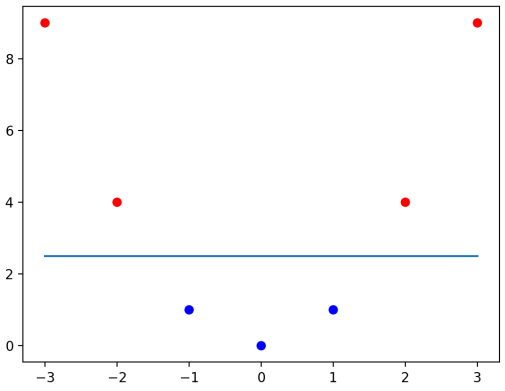
$$\begin{aligned} &\min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad &y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0) \geq 1, \ \forall i \in \{1, \ldots, n\} \end{aligned} \tag{1}$$

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

(a)



(b) The optimal decision boundary, is the line $\phi(x)_2 = 2.5$ so the margin achieved is **1.5**

(c) Any vector with first dimension 0 will be orthogonal to the decision boundary so $\mathbf{v} = [0, 1]^\top$ works.

(d) First note that we must have $\mathbf{w}^\top \mathbf{v} = 0$ for any v in the decision boundary. So $\mathbf{w} = [0, k]^\top$ for some k. Now suppose we have $\mathbf{w}^\top \boldsymbol{\phi}(x) + w_0 = 0$. Then $x^2 = 2.5$, since this is the optimal decision boundary and this is $kx^2 + w_0 = 0$ and so $-\frac{w_0}{k} = 2.5$. So $\mathbf{w} = [0, k]^\top$ with $w_0 = -2.5k$.

(e) First compute the constraint $y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0) \geq 1$ for all i and then we will choose the smallest magnitude $\mathbf{w}$ that satisfies these constraints. The constraint simplifies to $y_i(k(x_i^2 - 2.5) \geq 1$. Plugging in $x_i = 3$, $y_i = 1$, we get $k \geq \frac{1}{6.5}$.Plugging in $x_i = 2$, $y_i = 1$ we get $k \geq \frac{2}{3}$. Plugging in $x_i = 1$, $y_i = -1$, we get $k \geq \frac{2}{3}$. Plugging in $x_i = 0$, $y_i = -1$, we get $k \geq \frac{1}{2.5}$. So we see that $k \geq 2/3$. Minimizing the magnitude of $\mathbf{w}$ means making k as small as possible so the optimal value of $\mathbf{w}$ is $[0, \frac{2}{3}]^\top$.

(f) From part 4 we have: $w_0 = -2.5k = -\frac{5}{3}$. The discriminant function is $h = [0, \frac{2}{3}]^\top \boldsymbol{\phi}(x) - \frac{5}{3}$

(g) The support vectors are the points with $x = -1, x = 1, x = -2, x = 2$. Plugging these into $y_i(\frac{2}{3}x_i^2 - \frac{5}{3})$ gives 1 for all of these points; so the constraints are 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

(a) Consider the kernel matrix of $K(\mathbf{x}, \mathbf{x}')$. Each element is $K_{i,i'} = cK^{(1)}(\mathbf{x}_i, \mathbf{x}_i')$, so $\mathbf{K} = c\mathbf{K}^{(1)}$. Then $\mathbf{z}^\top \mathbf{K} \mathbf{z} = c\mathbf{z}^\top \mathbf{K}^{(1)} \mathbf{z} \geq 0$ by assumption.

(b) Again, consider the matrix $\mathbf{K}$, with element $K_{i,i'} = K^{(1)}(\mathbf{x}_i, \mathbf{x}_i') + K^{(2)}(\mathbf{x}_i, \mathbf{x}_i')$. So we have $\mathbf{K} = \mathbf{K}^{(1)} + \mathbf{K}^{(2)}$. Then $\mathbf{z}^\top \mathbf{K} \mathbf{z} = \mathbf{z}^\top (\mathbf{K}^{(1)} + \mathbf{K}^{(2)}) \mathbf{z} = \mathbf{z}^\top \mathbf{K}^{(1)} \mathbf{z} + \mathbf{z}^\top \mathbf{K}^{(2)} \mathbf{z} \geq 0$.

(c) Consider the matrix $\mathbf{F} = \mathbf{diag}(f(x_1), f(x_2), \ldots, f(x_n))$; denoting a n x n diagonal matrix with non-

zeros entries specified. Then $\mathbf{K} = \mathbf{F}\mathbf{K}^{(1)}\mathbf{F}$; you can check that $\mathbf{F}\mathbf{K}^{(1)}\mathbf{F}_{i,i'}$ is $f(\mathbf{x}_i)K^{(1)}(\mathbf{x}_i, \mathbf{x}'_i)f(\mathbf{x}'_i)$. Then $\mathbf{z}^\top\mathbf{K}\mathbf{z} = \mathbf{z}^\top\mathbf{F}\mathbf{K}^{(1)}\mathbf{F}\mathbf{z} = (\mathbf{F}\mathbf{z})^\top\mathbf{K}^{(1)}(\mathbf{F}\mathbf{z}) \geq 0$; where in the last step we use the fact that $\mathbf{F}$ is symmetric.

(d) We can write $\mathbf{K}^{(1)}(\mathbf{x}, \mathbf{x}')\mathbf{K}^{(2)}(\mathbf{x}, \mathbf{x}')$ as $(\sum_{i=1}^d \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{x}'))(\sum_{j=1}^d \phi_j^{(2)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x}'))$. Distributing this sum gives:

$$\mathbf{K}^{(1)}(\mathbf{x}, \mathbf{x}')\mathbf{K}^{(2)}(\mathbf{x}, \mathbf{x}') = \sum_i \sum_j \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{x}')\phi_j^{(2)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x}')$$

Since we want to write this in the form of an inner product; this suggests a basis of $\mathbf{x}$ given by the product of $\phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})$ over all pairs of indices (i, j); in $\mathbb{R}^{d^2}$. Specifically,

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_1^{(1)}(\mathbf{x})\phi_1^{(2)}(\mathbf{x}), \phi_1^{(1)}(\mathbf{x})\phi_2^{(2)}(\mathbf{x}), \ldots, \phi_d^{(1)}(\mathbf{x})\phi_d^{(2)}(\mathbf{x})]^\top$$

In this basis, we have $\boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\phi}(\mathbf{x}') = \mathbf{K}^{(1)}(\mathbf{x}, \mathbf{x}')\mathbf{K}^{(2)}(\mathbf{x}, \mathbf{x}')$; so the kernel is valid.

(e)
(i) Let $\boldsymbol{\phi}(x) = [1, x, \frac{x^2}{\sqrt{2!}}, \ldots, \frac{x^i}{\sqrt{i!}}, \ldots]$. Then $\boldsymbol{\phi}(x)^\top\boldsymbol{\phi}(x') = \lim_{i\to\infty} 1 + xx' + \frac{(xx')^2}{2!} + \cdots + \frac{(xx')^i}{i!} = \exp(xx')$
This would be hard to use as a basis function in logistic regression because it is an infinite basis and requires transforming your features into an infinite dimensional space. It would use a lot of memory and have to be approximated. The kernel trick bypasses this by computing the kernel directly and not storing the basis functions in memory.
(ii) $K(\mathbf{x}, \mathbf{x}') = \exp(K^{(1)}(\mathbf{x}, \mathbf{x}'))$ can be rewritten as $K(\mathbf{x}, \mathbf{x}') = \exp(\boldsymbol{\phi}(x)^\top\boldsymbol{\phi}(x'))$. Expanding the inner product:

$$K(\mathbf{x}, \mathbf{x}') = \exp(\sum_i \phi_i(x)\phi_i(x')) = \prod_i \exp(\phi_i(x)\phi_i(x'))$$

This is valid by the rule shown in part a for exponentials; and the product rule applied inductively.

(f)First, take the valid kernel $\mathbf{x}^\top\mathbf{x}'$. The kernel $\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2}$ is valid from rule (1). The kernel $\exp(\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2})$ is valid by the result from part (e). Take $f(\mathbf{x}) = (\exp(\frac{\mathbf{x}^\top\mathbf{x}}{2\sigma^2}))^{-1}$. Then from rule (3), $(\exp(\frac{\mathbf{x}^\top\mathbf{x}}{2\sigma^2}))^{-1}\exp(\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2})\exp(\frac{\mathbf{x}'^\top\mathbf{x}'}{2\sigma^2}))^{-1}$ is valid. Simplifying this, we have $\exp(\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2} - \frac{\mathbf{x}^\top\mathbf{x}}{2\sigma^2} - \frac{\mathbf{x}'^\top\mathbf{x}'}{2\sigma^2}) = \exp(\frac{-||\mathbf{x}-\mathbf{x}'||_2^2}{2\sigma^2})$, showing the kernel is valid.

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.

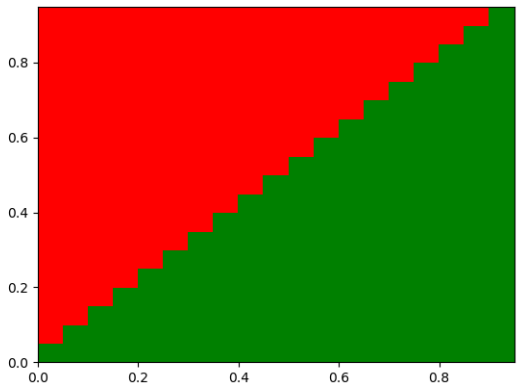
In addition, provide answers the following reading questions **in one or two sentences for each**.

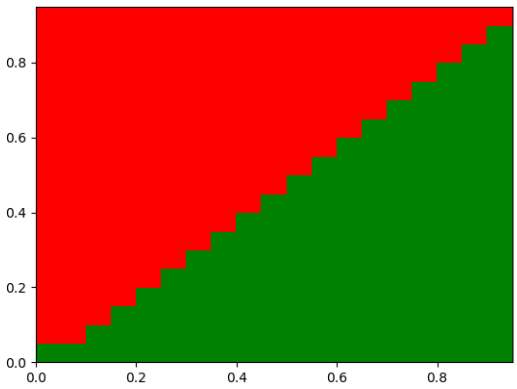
1. In one short sentence, state the main purpose of the paper.
2. Describe each of the parameters in Eq. 1 in the paper
3. State, informally, one guarantee about the Kernel perceptron algorithm described in the paper.
4. What is the main way the budget kernel perceptron algorithm tries to improve on 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

The algorithm is implemented in the attached python file. Here are the decision boundary plots:





The left graphic is the decision boundary for the Kernel Perceptron algorithm; and the right graphic is the decision boundary for the Budget Kernel Perceptron algorithm. For the default settings($\beta = 0$ and $N = 100$), we get the following:

```
Beta:  0
N:  100
Number of Random Samples:  20000
Budget Kernel Perceptron Training Time:  2.66074705124
Kernel Perceptron Training Time:  3.47409915924
Kernel Perceptron training accuracy:  1.0
Budget Kernel perceptron training acc:  1.0
Kernel Perceptron validation accuracy:  1.0
BK validation accuracy:  1.0
Standard SVM validation accuracy:  1.0
BK, Number of SV:  100
K, Number of SV:  209
```

First I experimented with adjusting the N parameter, the maximum number of support vectors for the Budget Kernel Perceptron Algorithm. At low values of N, this sacrifices accuracy for faster training time; holding the number of samples and β constant. For example, for $N = 25$, the following results were typical:

```
Beta:  0
N:  25
Number of Random Samples:  20000
Budget Kernel Perceptron Training Time:  1.94084095955
Kernel Perceptron Training Time:  4.1329741478
Kernel Perceptron training accuracy:  1.0
Budget Kernel perceptron training acc:  0.969444104934
Kernel Perceptron validation accuracy:  1.0
BK validation accuracy:  0.968928863451
Standard SVM validation accuracy:  1.0
BK, Number of SV:  25
K, Number of SV:  228
```

For values of N greater than 50 there was little decrease in accuracy, but training time was still faster. Experimenting with β, larger values of β tend to make the performance worse. For very small values of β, increasing β seems to improve the accuracy but it is hard to tell because the accuracy was already high.

```
Beta:  0.6
N:  30
Number of Random Samples:  20000
Budget Kernel Perceptron Training Time:  18.2466111183
Kernel Perceptron Training Time:  4.69864797592
Kernel Perceptron training accuracy:  0.997466638518
Budget Kernel perceptron training acc:  0.531439238214
Kernel Perceptron validation accuracy:  0.997649223222
BK validation accuracy:  0.527800490597
Standard SVM validation accuracy:  1.0
Standard SVM, time 0.514189004898
BK, Number of SV:  30
K, Number of SV:  332
```

Increasing the number of random samples tends to increase the computation time and also increase the accuracy.

The results of sklearn's implementation of standard SVM without regularization term are included in the above outputs. The standard SVM usually classifies all the validation examples correctly and fits in < 1 second.

1. The paper discusses the problem of high computational cost in high dimensional learning systems introduces a new online SVM algorithm called LASVM that achieves competitive accuracy and runs faster than contemporary SVM solvers; as well as discussing methods to reduce training time by prioritizing more informative examples.
2. w is the vector of weights that are associated with the features, representing their importance in deciding between classes. b is the bias parameter; it is the offset that determines the orthogonal distance from the decision boundary to the origin.
3. When a solution exists (i.e. the data is linearly separable in feature space); then the algorithm converges after a finite number of mistakes; so S is finite.
4. The Budget Kernel Perceptron first modifies the decision boundary even for correctly classified points, that have insufficient margin. This increases the number of mistakes, and thus the number of support vectors; which the Budget Kernel Perceptron corrects for by capping the number of support vectors at N.

Calibration [1pt]

Approximately how long did this homework take you to complete? 10 hours