Homework 4

Due: April 13, 2018 at 7:00PM

Written Questions

Recall the definition for a halfspace $h_w(x)$ given in lecture:

$$h_w(x) = \begin{cases} 1 & \text{if } \langle w, x \rangle \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that halfspaces may employ the use of bias terms. That is, each input has an additional constant '1' attribute, and a corresponding weight for that attribute. Given the bias weight has value w_{bias}, this has an effect equivalent to the following (where x does not have the additional constant):

$$h_w(x) = \begin{cases} 1 & \text{if } \langle w, x \rangle \geq -w_{bias}, \\ 0 & \text{otherwise.} \end{cases}$$

Problem 1

(15 points)
Define a halfspace for the following functions. Provide a description for how the weights are to be defined so that the classifier has the desired behavior. Weights may be expressed in terms of d.

a. Conjunction: Output is 1 if and only if all d attributes are 1.

b. Majority: Output is 1 if and only if more than half of the d attributes are 1.

Note: These attributes are binary.

Problem 2

(8 points)
Consider the function $h_{\text{equiv}}(x_1, x_2)$ (where $x_1, x_2 \in \{0, 1\}$) defined as

$$h_{\text{equiv}}(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 = x_2, \\ 0 & \text{otherwise.} \end{cases}$$

Show that h_{equiv} cannot be represented as a halfspace. *Hint:* In class, we showed that XOR cannot be represented with a halfspace. You can use a similar argument here.

Problem 3

(15 points)
Consider the set of lines in the plane that pass through the origin. All such lines may be expressed in the form $w_1 \cdot x_1 + w_2 \cdot x_2 = 0$ with $||w|| = 1$. Prove that the distance from a point x to a line defined by w is $|\langle w, x \rangle|$. You should define a distance function and set the derivative to zero to find the minimum distance.

Problem 4

(12 points)
Describe an efficient algorithm for a string kernel comparison. Your algorithm should output the similarity score between two given strings. Your algorithm should use the similarity score defined in lecture. For example, $\langle v_{\text{fame}}, v_{\text{family}} \rangle = 6$ and $\langle v_{\text{fame}}, v_{\text{farm}} \rangle = 4$.

Compute the runtime of your algorithm, and justify that your algorithm produces the desired comparison. *Hint:* Consider the vectorization of a string as described in lecture. While it is infinite-dimensional, the number of attributes with non-zero values is finite.

Programming Assignment

Introduction

In this assignment, you will be implementing an SVM to solve binary classification problems. While some modern solvers use gradient-based methods to train the SVM, you will be using a Python quadratic programming library, `quadprog` as an optimizer.

Stencil Code & Data

You can find the stencil code for this assignment on the course website. We have provided the following three files:

- `main.py` is the entry point of your program, which will read in the data, run the classifiers and print the results.
- `svm.py` contains the `SVM` model that you will be implementing.
- `qp.py` contains a `quadprog` wrapper function, `solve_QP`, which can efficiently solve quadratic programs.

You should *not* modify any code in `main.py` or `qp.py`. All the functions you need to fill in reside in `svm.py`, marked by `TODO`s. To run the program, run `python main.py <path_to_dataset>` in a terminal.

Fake Datasets

We've provided two fake datasets, `fake-data1.csv` and `fake-data2.csv`, for you to train your SVM classifier with. Both datasets contains only two dimensional data so that you can easily plot the data if you like. The first dataset is linearly separable while the second is not. You should use the fake datasets for debugging purposes.

Spambase Dataset

You will also be testing your SVM classifier on a real world dataset, the Spambase dataset. You can find more details on the dataset **here**. We will only be using a subset of the full dataset. Our version is available along with the other two datasets on department machine at `/course/cs1420/data/hw4/spambase.csv`.

We suggest that you copy over both datasets into a `data` folder in the same directory as your source code. On a department machine, you can copy the files to your working directory by running the command

```
cp /course/cs1420/data/hw4/* <DEST DIRECTORY>
```

where `<DEST DIRECTORY>` is the directory where you would like to copy the two data files. If you are working locally, you will need to use the `scp` command to copy the files to your computer over `ssh`:

```
scp <login>@ssh.cs.brown.edu:/course/cs1420/data/hw4/* <DEST DIRECTORY>
```

The Assignment

SVMs as Quadratic Programs

Recall that the hard margin SVMs can be trained by solving a quadratic program. In lecture, we showed that we can formulate the problem in the following way.

$$\max_{\alpha \in R^n} \Big(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (2y_i - 1)(2y_j - 1)\alpha_i \alpha_j \langle x_i, x_j \rangle \Big).$$

$$\text{subject to } \alpha \geq 0$$

However, in the assignment, you will be dealing with data that isn't necessarily linearly separable, so we must solve a similar, yet different, quadratic program. It is the dual of the soft-margin SVM classifier. The corresponding quadratic program has the same objective function but also has the following additional constraints.

$$\text{subject to } \sum_{i=1}^{n} \alpha_i (2y_i - 1) = 0 \text{ and } 0 \leq \alpha_i \leq \frac{1}{2n\lambda}, \forall i.$$

Kernels

We can use the kernel trick to implicitly transform the data to higher dimensions. Two popular kernels for SVMs are the polynomial kernel and the radial basis function kernel. The kernel function replaces the inner product, so you can replace $\langle x_i, x_j \rangle$ with $K(x_j, x_k)$, which takes a different form depending on the kernel used:

- **Linear Kernel** (the same as the dot product):

$$K(x_j, x_k) = x_j^T x_k.$$

- **Polynomial Kernel**:

$$K(x_j, x_k) = (x_j^T x_k + c)^d.$$

- **Radial Basis Function Kernel**:

$$K(x_j, x_k) = \exp\Big(- \gamma ||x_j - x_k||^2 \Big).$$

Note that c, d and γ are all *hyperparameters* here. That is, the kernels will behave differently depending on the values you use for these parameters. For example, as d increases, the polynomial kernel can represent more flexible decision boundaries. We provide you with hyperparameter values that perform well, and do not expect you to change them; however, feel free to experiment yourselves.

Training and Classification

After solving for the various values of α via the quadratic program detailed above, you can calculate the bias b by finding an index i such that $0 < \alpha_i < \frac{1}{2n\lambda}$ (notice that these are strict inequalities, unlike the constraints) and then computing

$$b = w \cdot x_i - (2y_i - 1) = \Big(\sum_{j=1}^{n} \alpha_j (2y_j - 1) K(x_i, x_j) \Big) - (2y_i - 1)$$

Important Note: When finding an α_i that satisfies this strict inequality, we recommend using `np.isclose` with a absolute tolerance of `1e-3` as opposed to $<$ or $>$ due to floating point imprecision.

Then, to classify a data point z, calculate the following expression:

$$c = w \cdot z - b = \Big(\sum_{i=1}^{n} \alpha_i (2y_i - 1) K(x_i, z) \Big) - b$$

If $c > 0$, the example should be classified as 1 and 0 otherwise.

Quadprog

We will be using the Python library `quadprog` to solve quadratic programs. You do not need to call any of the methods provided by `quadprog` since we have written a wrapper function `solve_QP`. However, you will still need to make sure that `quadprog` is installed on the machine where you are running your code.

If you are on a department machine, we have installed `quadprog` in our virtualenv. If you are running locally, you should run `pip3 install quadprog`. You can test that `quadprog` is successfully installed by running the example quadratic program located in `qp.py`.

Recall that quadratic programs follow the formulation

$$\text{minimize } \frac{1}{2} x^T Q x + c^T x$$

$$\text{subject to } Ax \leq b \text{ and } Ex = d.$$

Note: The b we use here is different from the b that denotes the bias of the SVM above.

In this assignment, we will be using the Python library `quadprog` to solve quadratic programs. We've written a wrapper around the function, called `solve_QP`, located in the `qp.py` file. Given Q, c, A, b, E, and d, `solve_QP` will return the values of x that obtain the optimal solution. The file `qp.py` contains an example quadratic program solved with `solve_QP`.

It is important to note that all quadratic programs can be converted to the form above. To do so, you might need to make use of the following small tricks:

- To convert a maximization problem to a minimization problem, you can simply multiply the objective function by -1.
- You can write greater-than inequality constraints like $Ax \geq b$ as less-than inequality constraints by multiplying the A matrix by -1, $(-A)x \leq b$.

Project Report

As introduced in HW1, each programming assignment in this course will be accompanied by a short report in which you will answer a few guiding questions about the results of your algorithm. To answer these questions, you may need to write new code that generates some output (potentially a value or graph) that you will want to include in your writeup. For example, you may be asked to contrast the results of running the same algorithm on different datasets or to explore the effect of changing a certain hyperparameter in your algorithm. By the end of this course, you will not only be able to implement common machine-learning algorithms but also develop intuition as to how the results of a given algorithm should be interpreted.

The next section outlines some guiding questions that you should answer in your report. Please leave any code that you use in your final handin but make sure that it is **not** run by default when your program is run. We ask that your final report be a PDF file named `report.pdf`, which must be handed in along with your code. You may use any program to create the PDF file, but we highly recommend using LaTeX. We have provided an example report available on our course website to get you started.

Guiding Questions

- Comment on the testing and training accuracy of your SVM classifier on the Spam dataset. Discuss how kernels (and their hyperparameters) affect the classifier's accuracy.
- (Extra Credit) Plot the differences in training and testing error as you change the hyperparameters of the polynomial and RBF kernel.
- (Extra Credit) Produce a visualization of the decision boundaries on the both of the 2D datasets (`fake-data1.csv`, `fake-data2.csv`). Your plot should contain the training data and decision boundaries. In addition, the label of each training point should be included. This should all be contained within a single figure for each dataset.

Grading Breakdown

We will not be providing accuracy targets for this assignment. You should use the fake datasets that we have provided to ensure that your algorithm is working correctly.

Written Questions	50%
SVM Classifier	45%
Report	5%
Total	100%

Handing in

Written Questions

Answers to the written questions should be printed, stapled and labeled with the date, homework number and your unique course ID. You should place your handin in the CS1420 handin box located on the second floor of the CIT.

Programming Assignment

To hand in the programming component of this assignment, first ensure that your code runs on *Python 3* using our course `virtualenv`. You can activate the `virtualenv` on a department machine by running the following command in a Terminal:

```
source /course/cs1420/cs142_env/bin/activate
```

Once the `virtualenv` is activated, run your program and ensure that there are no errors. We will be using this `virtualenv` to grade all programming assignments in this course so we recommend testing your code on a department machine each time before you hand in. Note that handing in code that does not run may result in a significant loss of credit.

To hand in the coding portion of the assignment, run `cs142_handin hw4` from the directory containing all of your source code and your report in a file named `report.pdf`.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin. Instead, you should use the course ID that you generated when filling out the collaboration policy form. If you do not have a course ID, you should email the HTAs as soon as possible.

Obligatory Note on Academic Integrity

Plagiarism—don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and, if you have any questions, please contact a member of the course staff.

Feedback

Many aspects of this course have been radically redesigned compared to previous iterations of the course. To continually improve this class and future iterations of the class, we are offering an online **anonymous feedback form**. If you have comments, we would love to hear your constructive feedback on this assignment and on any other aspects of the course.