Homework #4

ECE 661: Introduction to Machine Learning
Prof. Carlee Joe-Wong and Prof. Virginia Smith
Due: Tuesday October 23rd, 2018 at 8:30AM PT / 11:30AM ET

Please remember to show your work for all problems and to write down the names of any students that you collaborate with. The full collaboration and grading policies are available on the course website: `https://18661.github.io/`.

Your solutions should be uploaded to Gradescope (`https://www.gradescope.com/`) in PDF format by the deadline. We will not accept hardcopies. If you choose to hand-write your solutions, please make sure the uploaded copies are legible. Gradescope will ask you to identify which page(s) contain your solutions to which problems, so make sure you leave enough time to finish this before the deadline. We will give you a 30-minute grace period to upload your solutions in case of technical problems.

1 Dual of Squared Hinge Loss SVM *[15 points]*

Consider the following variant of SVM, where we use a **squared** hinge loss instead of hinge loss

$$\min_{\boldsymbol{w}\in\mathbb{R}^d,\xi_1,\dots\xi_n} \|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i^2,$$

$$\text{s.t. } \forall i, y_i\langle \boldsymbol{w}, X_i\rangle \geq 1-\xi_i, \quad \xi_i \geq 0.$$

a. Write the Lagrangian of the problem.

b. Derive the dual formulation of the problem.

c. How does the dual formulation of the squared hinge loss SVM compare to the dual of the standard hinge loss SVM discussed in lecture? Which optimization problem is more complex?

2 Kernel functions and linear separability *[15 points]*

The power of kernel methods stems from the observation that data which are not linearly separable in their input feature space may in fact be separable in a higher-dimensional space, which is a function of the input features. For example, the data shown in Figure 1 are not linearly separable in the feature space $(\boldsymbol{x}[1], \boldsymbol{x}[2])$, but they are separable in the transformed space where we transform each point $\boldsymbol{x} = (\boldsymbol{x}[1], \boldsymbol{x}[2])$ according to $\Phi(\boldsymbol{x}) = (\boldsymbol{x}[1]^2, \boldsymbol{x}[2]^2, \sqrt{2}\,\boldsymbol{x}[1]\boldsymbol{x}[2])$, as shown in Figure 2. If we map the decision boundary plane back to the two-dimensional space, the decision boundary is shown as in Figure 3, which achieves 100% accuracy.

Further, we can observe that for two data points $\boldsymbol{x}$ and $\boldsymbol{y}$, we have that

$$\Phi(\boldsymbol{x})\cdot\Phi(\boldsymbol{y}) = \boldsymbol{x}[1]^2\boldsymbol{y}[1]^2 + \boldsymbol{x}[2]^2\boldsymbol{y}[2]^2 + 2\,\boldsymbol{x}[1]\boldsymbol{y}[1]\boldsymbol{x}[2]\boldsymbol{y}[2] = (\boldsymbol{x}\cdot\boldsymbol{y})^2 \triangleq \mathcal{K}(\boldsymbol{x},\boldsymbol{y}), \quad (1)$$

where $\mathcal{K}$ is a *kernel function*[1].

[1] In case the notation $\boldsymbol{x}\cdot\boldsymbol{y}$ is new to you, it's fine to think of it as the same as $\boldsymbol{x}^T\boldsymbol{y}$. We use the "dot" notation here

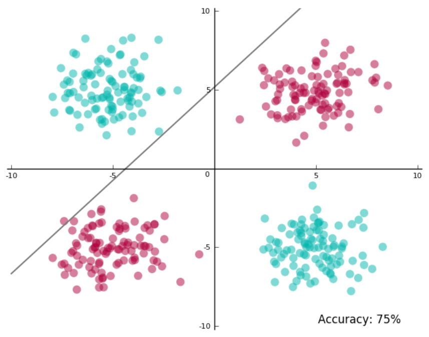

Figure 1: A two-dimensional XOR dataset. It is not linearly separable. A linear SVM can only achieve 75% accuracy.

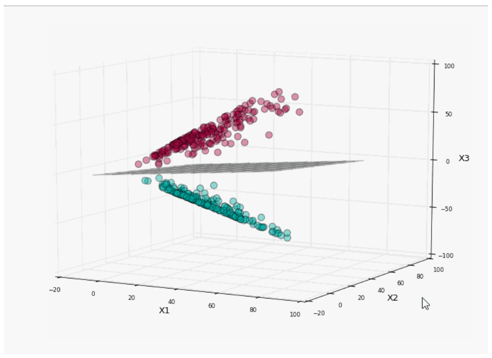

Figure 2: The data have been mapped into the three-dimensional input space $(\boldsymbol{x}[1]^2, \boldsymbol{x}[2]^2, \sqrt{2}, \boldsymbol{x}[1]\boldsymbol{x}[2])$. In the higher dimensional space, the data can be linearly separable by a linear decision boundary.

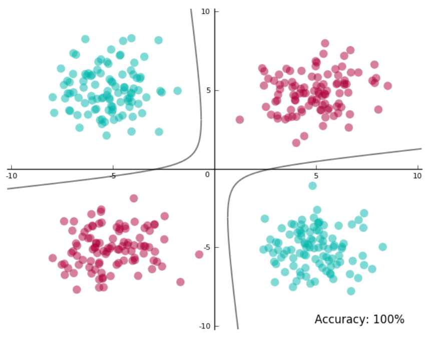

Figure 3: The linear decision boundary in three dimensions (a plane) found in Figure 2 can be mapped back to the two-dimensional space as a non-linear boundary, which separates the two classes with 100% accuracy.

because it turns out that inner products can be defined on spaces more general than $\mathbb{R}^n$; in particular, they can be defined over infinite-dimensional spaces such as Hilbert space. In such a space, the notation $\boldsymbol{x}^T\boldsymbol{y}$ doesn't have an obvious semantics — what does it mean to transpose an infinite-dimensional object? However, the inner product is still well-defined. Kernel methods thus allow you to manipulate *infinite-dimensional objects* with *finite memory*, by working only with similarities between object pairs and not representing the objects explicitly. Did that just totally blow your mind? If this sort of things interests you, search online or in the textbook "(KM): Machine Learning: A Probabilistic Perspective" for the phrase "reproducing kernel Hilbert space". On the other hand, if this discussion sounds like total nonsense, forget you ever read this.

In this problem, we will generalize the situation shown in Figures 1, 2 and 3.[2]

a. Data separated by an ellipse of the form: $a_1(\boldsymbol{x}[1]-a_4)^2+a_2(\boldsymbol{x}[1]-a_4)(\boldsymbol{x}[2]-a_5)+a_3(\boldsymbol{x}[2]-a_5)^2-1=0$ can be separated by a linear decision boundary, $c_1\boldsymbol{x}[1] + c_2\boldsymbol{x}[2] + c_3\boldsymbol{x}[1]^2 + c_4\boldsymbol{x}[2]^2 + c_5\boldsymbol{x}[1]\boldsymbol{x}[2] - 1 = 0$, when the original features $(\boldsymbol{x}[1], \boldsymbol{x}[2])$ are projected into the five-dimensional feature space $(\boldsymbol{x}[1], \boldsymbol{x}[2], \boldsymbol{x}[1]^2, \boldsymbol{x}[2]^2, \boldsymbol{x}[1]\boldsymbol{x}[2])$. Express the coefficients $\{c_1, c_2, c_3, c_4, c_5\}$ in terms of $\{a_1, a_2, a_3, a_4, a_5\}$.

b. Suppose a linear decision boundary $6\boldsymbol{x}[1] + 10\boldsymbol{x}[2] + \boldsymbol{x}[1]^2 + \boldsymbol{x}[2]^2 + 18 = 0$ can separate two classes of data in the feature space $(\boldsymbol{x}[1], \boldsymbol{x}[2], \boldsymbol{x}[1]^2, \boldsymbol{x}[2]^2)$, which was projected from the two-dimensional features $(\boldsymbol{x}[1], \boldsymbol{x}[2])$. What is the equation of this decision boundary in the two-dimensional space $(\boldsymbol{x}[1], \boldsymbol{x}[2])$? What shape is it?

3 k-Nearest Neighbors Decision Boundary *[15 points]*

For some applications, it may make sense to perform k-nearest neighbors with respect to a distance other than the usual Euclidean distance. In this problem, we will look at the k-nearest neighbor problem when the distance between the points is the following modified form of the Euclidean distance. Given two vectors $v_1 = (x_1, y_1), v_2 = (x_2, y_2)$, the modified distance measure $d_M(v_1, v_2)$ is defined as:

$$d_M(v_1, v_2) = \sqrt{\frac{1}{2}(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Consider the following labelled training datasets $\mathcal{D}_1$:

$$((0,0),1), ((2,2),2), ((4,0),3)$$

and $\mathcal{D}_2$:

$$((0,0),1), ((1,1),1), ((-1,1),2)$$

a. First consider the 1-nearest neighbor classifier on the data points in $\mathcal{D}_1$ with respect to the usual Euclidean distance, and draw the decision boundary for the classifier. Write down the equations of the decision boundary. Clearly mark each region in your drawing with the label assigned by the classifier to a test example in this region.

b. Now, consider the 1-nearest neighbor classifier on the data points in $\mathcal{D}_1$ with respect to the **modified** Euclidean distance d_M. In a separate figure, draw the decision boundary for this classifier. Again write down the equations for the different segments of the decision boundary, and clearly mark each region in your drawing with the label assigned by the classifier to a test example in this region.

c. Repeat part (a), i.e. draw the decision boundary with respect to Euclidean distance, for data points in $\mathcal{D}_2$.

d. Repeat part (b), i.e. draw the decision boundary with respect to the modified Euclidean distance, for data points in $\mathcal{D}_2$.

4 Robustness of k-Nearest Neighbors *[15 points]*

Suppose that we have two labels 0 and 1. Suppose we are given any k, and any test point x. Let $z_1, \cdots, z_k$ be the k closest neighbors of x in the training data. For the rest of the question, we make the assumption that for all $i = 1, \cdots, k$, the probability that the label of z_i is not equal to the label of x is $p = 0.2$. Moreover, for $i \neq j$, the events that the label of z_i is not equal to the label of x and that the label of z_j is not equal to the label of x are independent. In reality of course, this assumption will not hold for very large k, but for small k, this is a fairly reasonable assumption to make.

[2]Image source of Figures 1, 2 and 3: https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

a. What is the probability that the 1-nearest neighbor classifier makes a mistake on x?

b. Now calculate the probability that 3-nearest neighbor classifier and the 5-nearest neighbor classifier make a mistake on x. What can you conclude from these calculations about the robustness of these classifiers?

5 Implementing SVMs *[40 points]*

In this problem, you will experiment with SVMs on a real-world dataset. You will implement a linear SVM (i.e., an SVM using the original features). **Please submit your source code as part of the homework PDF that you submit.**

Dataset: We have provided the *Splice Dataset* from UCI's machine learning data repository.[3] The provided binary classification dataset has 60 input features, and the training and test sets contain 1,000 and 2,175 samples, respectively. The files containing features are called `train_data.txt` and `test_data.txt`, and the files containing labels are called `train_label.txt` and `test_label.txt`.

5.1 Data preprocessing

Preprocess the training and test data by

a. Computing the mean of each dimension and subtracting it from each dimension

b. Dividing each dimension by its standard deviation

Please use the **sample** standard deviation formula:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})}$$

where $x_1, \cdots, x_N$ are sample points, and $\overline{x}$ is the sample mean.

This type of preprocessing is useful for SVMs, as SVMs attempt to maximize the distance between the separating hyperplane and the support vectors. If one feature (i.e., one dimension in this space) has very large values, it will dominate the other features when calculating this distance. Rescaling the features (e.g. to [0, 1]), will ensure that they all have the same influence on the distance metric.

Note that the mean and standard deviation should be estimated from the *training data* and then applied to both datasets. Explain why this is the case. Also, report the mean and the standard deviation of the 3rd and 10th features on the training data.

5.2 Implement linear SVM

Please fill in the functions `train_svm` and `test_svm` in `svm.py`.

The input of `train_svm` contain training feature vectors and labels, as well as the tradeoff parameter C. The output of `train_svm` contain the SVM parameters (weight vector and bias). In your implementation, solve the SVM in its primal form

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \forall_{1\leq i\leq N} \\ &\qquad\ \ \xi_i \geq 0, \forall_{1\leq i\leq N} \end{aligned}$$

[3] https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+%28Splice-junction+Gene+Sequences%29.

To solve the above quadratic problem, we encourage you to use the `cvxopt.solvers.qp` function in the CVXOPT [4] Python package. You can also use any package of your choice. If you are not familiar with CVXOPT, please refer to this tutorial at https://courses.csail.mit.edu/6.867/wiki/images/a/a7/Qp-cvxopt.pdf.

For `test_svm`, the input contains testing feature vectors and labels, as well as SVM parameters. The output contains the test accuracy.

Please make sure to include in the report the code of functions `train_svm` and `test_svm`, as well as other functions/scripts that generate your results.

5.3 Cross validation for linear SVM

Use 5-fold cross validation (**without shuffling** the order of training data, and samples 1 to 200 in the first fold, 201 to 400 in the second fold, etc) to select the optimal C for your implementation of linear SVM.

a. Report the 5-fold cross-validation accuracy (averaged accuracy over each validation set) and average training time (averaged over each training subset) on different C taken from $\{4^{-6}, 4^{-5}, \cdots, 4^{5}, 4^{6}\}$. How does the value of C affect the cross-validation accuracy and average training time? Explain your observation.

b. Which C do you choose based on the averaged cross-validation accuracy?

c. For the selected C, report the test accuracy.

[4] https://cvxopt.org