Homework 4
K-Means, KNN, Kernel Methods

CMU 10-701: Introduction to Machine Learning (Spring 2018)

OUT: March 26, 2018
DUE: **April 9, 2018, 10:30 AM**

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution independently: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Submitting your work:** Assignments should be submitted as PDFs using Gradescope unless explicitly stated otherwise. Each derivation/proof should be completed on a separate page. Submissions can be handwritten, but should be labeled and clearly legible. Else, submissions can be written in LaTeX. Upon submission, label each question using the template provided by Gradescope. Please refer to Piazza for detailed instruction for joining Gradescope and submitting your homework.
- **Programming**: All programming portions of the assignments should be submitted to Gradescope as well. For this homework, we will also use Autolab to automatically evaluate your implementations. Please see details in in Problem 5.

1 K-Means (20 pts) [Satya]

In this problem we will look at the K-means clustering algorithm. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the given set of n data points, and let γ be an indicator matrix such that $\gamma_{ij} = 1$ if x_i belongs to the j^{th} cluster, and $\gamma_{ij} = 0$ otherwise. Let $\mu_1, \ldots, \mu_k$ be the means of the clusters.

We can then define the distortion J as:

$$J(\gamma, \mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \mu_j||^2.$$

Finally, we let $C = \{1, \ldots, k\}$ be the set of clusters.

The most common form of the K-means algorithm proceeds as follows:

- Initialize $\mu_1, \ldots, \mu_k$.
- While J is decreasing, repeat the following:
  - Determine γ breaking ties arbitrarily:

$$\gamma_{ij} = \begin{cases} 1, & ||x_i - \mu_j||^2 \leq ||x_i - \mu_{j'}||^2, \forall j' \\ 0, & \text{otherwise.} \end{cases}$$

  - Recompute μ_j using the updates γ Remove j from C if $\sum_{i=1}^{n} \gamma_{ij} = 0$. Otherwise,

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_{ij} x_i}{\sum_{i=1}^{n} \gamma_{ij}}.$$

(a) **[5 pts]** Show that this algorithm will always terminate in a finite number of steps.
Hint 1: Note that the algorithm stops when J stops decreasing.
Hint 2: How many different values can γ take?

(b) **[7 pts]** Show that the minimum of J is a non increasing function of the number of clusters k. Argue that this means we cannot just choose the number of clusters by minimizing J.
Hint: Use induction.

(c) **[8 pts]** Suppose that we now use the ℓ_1 norm in the distortion J as opposed to the squared Euclidean distance:

$$J'(\gamma, \mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \mu_j||_1$$

Derive the K-means algorithm steps corresponding to this new distortion.

2 k Nearest Neighbors for Regression (20 pts) [Shaojie & Sreena]

In this question, we will explore and analyze how to use the k nearest neighbor (kNN) algorithm on **regression** tasks. Assume that we have n data points $(X_i, Y_i)_{i=1,\ldots,n}$ from some model $Y = f(X) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Note that f does not need to be a linear function. The goal is to train a predictor $\widehat{m}(x)$ using the kNN algorithm that predicts a value for the input x.

With respect to the dimensions of various quantities, assume that $X_i \in \mathbb{R}^d$, $Y_i \in \mathbb{R}$, and $f : \mathbb{R}^d \to \mathbb{R}$. **Also, for simplicity, assume that input points X_i, $i = 1, \ldots, n$ are equally spaced over** $[0, 1]^d$. For example, when $d = 1$, we would have $X_i = \frac{i}{n}$ (where $i \in [n]$), without a point at 0.

2.1 Warm up [8 pts]

Recall that for classification using kNN, the responses $Y_i \in \{1, 2, \ldots, C\}$ where C is the total number of classes. The kNN predictor for class c is then given by:

$$\widehat{m}_c(x) = \mathbb{P}[Y = c | X = x] = \frac{1}{k} \sum_{i: X_i \in \mathcal{N}_k(x)} 1(Y_i = c), \tag{1}$$

where $\mathcal{N}_k(x)$ gives k nearest neighbors of x. The overall kNN prediction is then given by $\arg\max_{c \in \{1,\ldots,C\}} \widehat{m}_c(X)$.

(a) **[2 pts]** For kNN regression, the basic idea is that we can take the average of the outputs of the k nearest neighbors given an input x. Write down the formula for $\widehat{m}(x)$ for kNN **regression**, by extending the classification based expression in Equation (1) to the regression case.

(b) **[3 pts]** Fix x. Derive the formula for $\mathrm{Var}(\widehat{m}(x))$.

(c) **[3 pts]** If all the data points $X_i \in \mathbb{R}^d$ are uniformly spaced over $[0, 1]^d$ (as stated above), for some point X_j, what is the distance between X_j and its closest neighbor?

2.2 kNN Regression Bounds [12 pts]

Assume we fix some $x \in \mathbb{R}^d$ and take $k = k_n$ such that $k_n \to \infty$ but $\frac{k_n}{n} \to 0$.

You can use the following fact without proving it: for any $x \in [0, 1]^d$, and $X_i \in \mathcal{N}_k(x)$, we must have $\|X_i - x\|_2 \leq C \cdot \left(\frac{k}{n}\right)^{\frac{1}{d}}$, where C is a positive constant[1].

(a) **[6 pts]** Assume that the groundtruth function f is L-Lipschitz. Recall from HW1 that this means

$$|f(y) - f(z)| \leq L\|y - z\|_2$$

Show that the risk of your predictor $\hat{m}$ satisfies

$$\mathbb{E}[(\hat{m}(x) - f(x))^2] \leq \left(\frac{L}{k} \sum_{i: X_i \in \mathcal{N}_k(x)} \|X_i - x\|_2\right)^2 + \frac{\sigma^2}{k} \tag{2}$$

$$\leq (CL)^2 \left(\frac{k}{n}\right)^{\frac{2}{d}} + \frac{\sigma^2}{k} \tag{3}$$

Hint: You should use Section 2.1 and the bias-variance decomposition:

$$\mathbb{E}[(\hat{m}(x) - f(x))^2] = \underbrace{(\mathbb{E}[\hat{m}(x)] - f(x))^2}_{\mathrm{Bias}(\hat{m}, f)^2} + \underbrace{\mathbb{E}[(\hat{m}(x) - \mathbb{E}[\hat{m}(x)])^2]}_{\mathrm{Var}(\hat{m}(x))}.$$

[1] It's actually easy to see why this is true intuitively. You can imagine a small cube within $[0, 1]^d$ that contains the k nearest neighbors of x, which has volume $\frac{k}{n}$.

(b) **[6 pts]** Because $k = k_n$ depends on n, show that the optimal k (which minimizes the upper bound (3)) is $k^* \in O(n^{2/(2+d)})$, and that for k^* the upper bound on the risk is

$$\mathbb{E}[(\hat{m}(x) - f(x))^2] \leq C' \cdot n^{-2/(2+d)}$$

for constant C'. Note that this implies as $n \to \infty$ (and d fixed) the estimator's risk approaches 0. This also implies the *curse of dimensionality*: for larger d we need exponentially more data.

3 Kernel SVM (20 pts) [Dimitris & Wenhao]

In this question, we are considering the kernel version of SVMs:

$$\min_{\omega\in\mathbb{R}^d, b, \xi_i\in\mathbb{R}, i=1,2,...,n} \frac{1}{2}||\omega||_2^2 + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad y^{(i)}(\omega^T\phi(x)^{(i)} + b) \geq 1 - \xi_i, \forall i = 1, ..., n$$

$$\xi_i \geq 0, \forall i = 1, ..., n$$

where $x^{(i)} \in \mathbb{R}^p, i = 1, 2, ..., n$ is the original training data coming along with the label $y^{(i)} \in \{-1, 1\}, C > 0$, is a constant and $\phi : \mathbb{R}^p \leftarrow \mathbb{R}^d$ is a mapping function that maps the original data to a new space. Generally speaking, $d > p$ (In fact, d can be $+\infty$). Please answer the following questions. Note that the questions with complexity could be answered with big-O notation.

1. **[8 pts]** First write down the Lagrangian of the above problem and then derive the dual problem step by step.
2. **[12 pts]** Suppose we have obtained the solution of the dual problem, denoted as $\alpha_i^\star$ for $i = 1, 2, ..., n$. Following the course slides, write down the corresponding primal solution $\omega^\star$ and $b^\star$. In the sub-questions below, suppose we now have to use the kernel SVM to make a classification decision at some test data point $z \in \mathbb{R}^p$. (12 points, 4 for each subproblem)
   (a) **[4 pts]** What is the time complexity of making the classification decision if the mapping is given by

   $$\phi(x) = (\underbrace{p^2, x_{p-1}^2, \cdots, x_1^2}_{\text{p}}, \underbrace{\sqrt{2}x_p x_{p-1}, \cdots, \sqrt{2}x_p x_1}_{\text{p-1}}, \underbrace{\sqrt{2}x_{p-1}x_{p-2}, \cdots, \sqrt{2}x_{p-1}x_1}_{\text{p-2}}, \cdots, \sqrt{2}x_2x_1, \sqrt{2c}x_p, \cdots, \sqrt{2c}x_1, c)^T,$$

   with a constant $c > 0$, and we need to compute it from scratch?

   (b) **[4 pts]** Let $K(u, v) = \phi(u)^T\phi(v)$ where $\phi(\cdot)$ has the same definition as above. Please give a compact form of $K(u, v)$. What is the time complexity of making the classification decision if we directly compute $K(\cdot, \cdot)$?

   (c) **[4 pts]** Now please go back to the dual formulation you derived previously. To avoid repetitive computation, one can precompute all the inner products $K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T\phi(x^{(j)})$ before solving the dual problem. What is the space complexity of this approach and what might be a problem if n is huge?

4 Multiple Choice Questions (10 pts) [Adarsh]

Please provide explanations. Without any explanation, no points will be awarded.

1. **(True or False). [2 pts]** A mixture of k-Gaussians always has exactly k modes (where a mode of a distribution is any point, possibly more than one, where the density attains its maximum value).

2. **Multiple Choice [5 pts]** An estimator f is called a linear smoother if the vector of fitted values $\hat{\mu} = (f(x_1), f(x_2), \ldots, f(x_n)) \in \mathbb{R}^n$ can be written as $\hat{\mu} = Sy$ where S is the smoother matrix and $y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ is the vector of input values.[2] Then, which of the following statements is/are **True**?

   A. k-nearest-neighbors regression, with a fixed k is a linear smoother.

   B. The (Nadaraya-Watson) kernel regression estimator with box-car kernel and fixed bandwidth is a linear smoother.

   C. The (Nadaraya-Watson) kernel regression estimator with box-car kernel and bandwidth chosen by leave-one-out cross-validation is not a linear smoother.

   D. Local linear regression estimator with box-car kernel and fixed bandwidth is a linear smoother.

3. **Multiple Choice [3 pts]** Which of the following statements is/are true about the k-Nearest Neighbor classifier? Select all that apply.

   A. As the value of k increases, the variance of the model increases

   B. As the value of k increases, the bias of the model increases

   C. As the value of k increases, the model complexity increases

   D. As the value of k increases, the number of parameters in the model increases

   E. As the number of training data points increases, the memory requirements of the model increase

[2] To be clear, this means that for fixed inputs $x_1, \ldots, x_n$ the vector of fitted values $\widehat{\mu}$ is a linear function of y; it does not mean that $f(x)$ need behave linearly as a function of x.

5 Programming Exercise (20 pts) [Otilia & George]

Note: Your code for the programming exercise should also be submitted to Autolab . In particular, there is a separate 'programming assignment' on Autolab to which you should upload your code, while visualizations and written answers should still be submitted within the primary Gradescope assignment.

Use Python 2.7 for your solution as the Autograder is set up for that programming language. A tutorial can be found at `https://docs.python.org/2.7/tutorial/`

You may **NOT** use any packages aside from the ones already used in the starter code. Doing so counts as an academic integrity violation and will be processed accordingly.

This assignment is challenging, make sure to start early!

In this problem you will implement our own **kernel SVM classifier**. For this, we will provide you with a starter code, and you will need to fill in the missing steps.

Consider a binary classification problem for which we have n training points, $x_1, ..., x_n$, each $x_i \in \mathbf{R}^2$, and corresponding labels $y_1, ..., y_n$, where $y_i \in \{-1, 1\}$. When using SVMs, we want to solve the following primal problem:

$$\begin{aligned} \min_{w,b,\{\epsilon_j\}} \quad & \frac{1}{2} w^T w + C \sum_j \epsilon_j \\ \text{subject to} \quad & (w^T x_j + b) y_j \geq 1 - \epsilon_j, \forall j \\ & \epsilon_j \geq 0, \forall j \end{aligned} \tag{4}$$

which has the following dual formulation:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \end{aligned}$$

Data. For this problem, we provide three different datasets, called `cluster_easy`, `cluster_harder` and `camel`. For each dataset, the data is already split into a train and a test dataset. The starter code already contains a function that loads each of these datasets into the following variables: x_train, y_train, x_test, y_test. Here, x_train and x_test are arrays of shape $n \times 2$ containing the train and test inputs, respectively, and y_train and y_test are arrays of length n containing the train and test labels, respectively. The data can be found under the `data/` folder in the starter code. A visualization of the datasets is shown in Figures 1, 2, 3, 4, 5, 6.

Code. We provide you with skeleton code for each of the functions you are responsible for implementing. You can find the code on Github at `https://github.com/otiliastr/svm_starter_code`, and under Resources in Piazza.

Evaluation Your code will be automatically evaluated and graded using **Autolab**. We will test your code on 3 datasets similar to the ones you have for training (the data follows the same distributions, but contains different points than the datasets you received). To ensure that partially correct implementations will receive partial points, we will evaluate separately each of the functions that you have to fill in. For this reason,

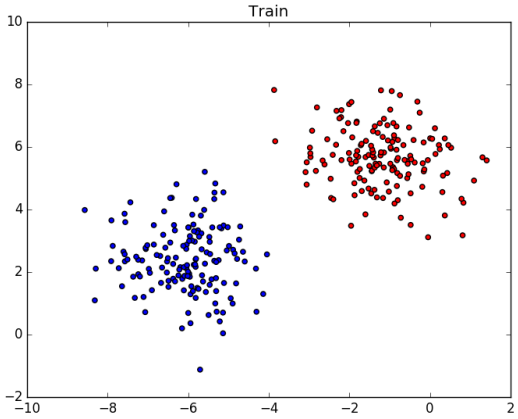

Figure 1: Cluster easy - Training set

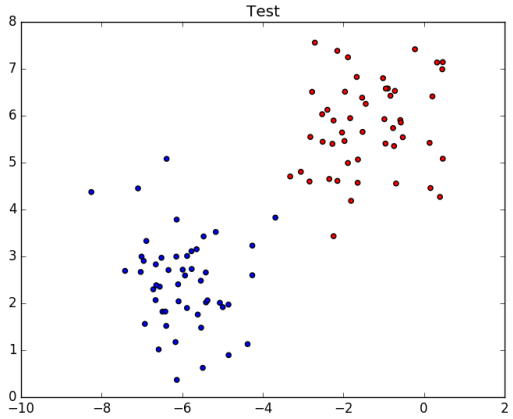

Figure 2: Cluster easy - Test set

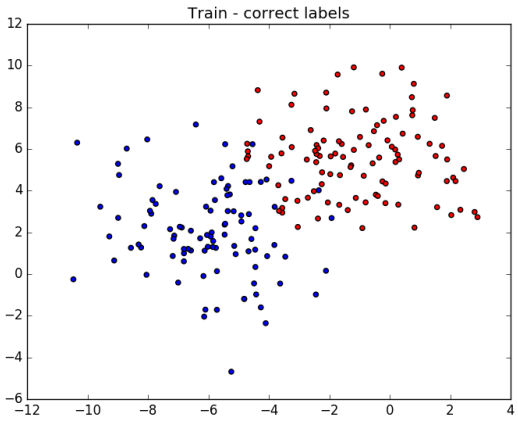

Figure 3: Cluster harder - Training set

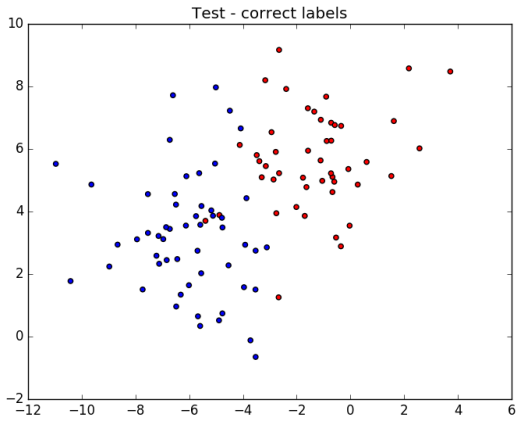

Figure 4: Cluster harder - Test set

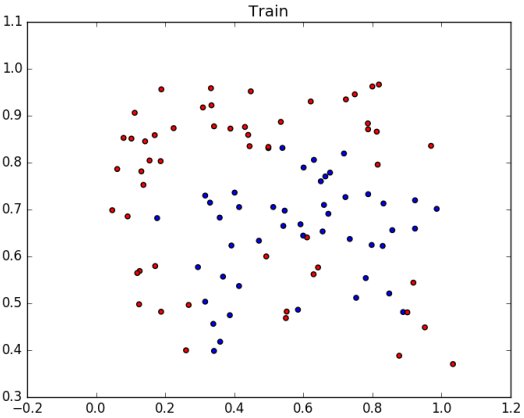

Figure 5: Camel - Training set

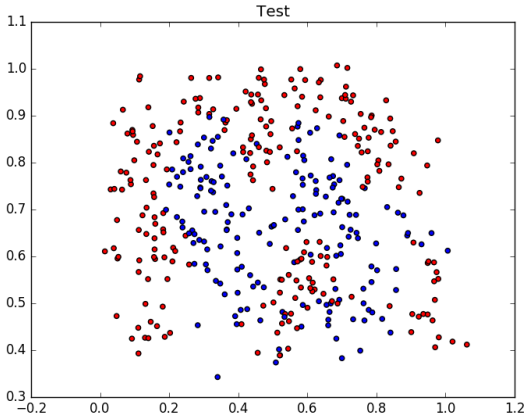

Figure 6: Camel - Test set

please **do not change the signature of any of the provided functions**, and only fill in the code where indicated. To prevent cheating, Autolab also checks your submission against all other student submissions, so make sure to write your own code!

Implementation [13 pts]

- Fill in the missing code in the provided starter code.
- Start your implementation in the function `run_svm.py`. This is the main function that loads the data, creates the SVM learner, and calls the train and predict functions.
- Fill in the SVM implementation in `classification.py`. The comments will guide you through the missing steps. Do not change any of the function signatures, and make sure that all your functions return the data types indicated in the comments.
- The function `solve_dual` solves the dual problem and computes the α_i's. In order to solve the quadratic program (QP) in the dual, we will use an existing QP solver that exists in the package `scipy.optimize`. Other optimization packages may not be installed in Autolab. To solve the QP, you are required to provide the objective function and the constraints, in the format specified in the documentation of the solver. See the `https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html` for details.
- The function `find_support_vectors`, uses the learnt dual variables in order to find the support vectors. Be careful of numerical instability here, which might cause unexpected results.
- The function `compute_bias`, uses the inputs and the found support vectors to compute the bias term b. A reference of how to correctly compute the bias in the presence of slack variables can be found at pages 7-8 from `http://fouryears.eu/wp-content/uploads/svm_solutions.pdf`.
- The function `predict`, uses the support vectors in order to predict the labels for new test points.
- The function `decision_function` evaluates the decision boundary equation on new points $f(x) = w^T x + b$. You will need to implement this in order to draw the plots required in the questions below.
- For question b) below, provide your own implementation of the RBF kernel in `kernels.py`. The use of external libraries is not allowed.
- Do not use any other Python packages that are not already imported in the starter code. If you import other packages, your code will most likely fail in Autolab.

Report Questions [7 pts]

(a) **[2 pts]** Train your SVM classifier on the datasets `cluster_easy` and `cluster_harder`, using a linear kernel function, with values of C in the following set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3, 10^4\}$. Use the function `plot_svm_decision_boundary` to plot decision boundary and the support vectors. Use these plots to answer the following questions:

(1) As we increase C, the number of support vectors tends to _____ (increase / decrease / stay the same).

(2) As we increase C, the width of the margin tends to _____ (increase / decrease / stay the same).

(b) **[5 pts]** Train your SVM classifier on the `camel` dataset, with $C = 1$ and using an RBF kernel with the following values for γ: $\{10^{-3}, 1, 10, 10^2, 10^3, 10^4\}$. For each γ, print the train and test accuracies, and use the function `plot_svm_decision_boundary` to plot decision boundary and the support vectors. Use these results to answer the following questions:

(1) As we increase γ, the number of support vectors tends to _____ (increase / decrease / increase at first and decrease later / decrease at first and increase later / stay the same).

(2) As we increase γ, the decision boundary tends to _____ (become smoother / become wigglier / stay the same).

(3) As we increase γ, the training accuracy tends to _____ (increase / decrease / increase at first and decrease later / decrease at first and increase later / stay the same).

(4) As we increase γ, the test accuracy tends to _____ (increase / decrease / increase at first and decrease later / decrease at first and increase later / stay the same).

(5) As we increase γ, the model _____ (overfits / underfits / is robust and will not be affected).

Autolab submission instructions: Please follow these instructions to make sure your code can run on Autolab:

- When you are ready to submit your solutions, you will create a new zip archive of the top-level directory, named `your-andrew-id hw4.zip`, and upload that through the Autolab website. Please include in this archive **only the `src` folder** containing your implementation, and remove the data and any extra files from the directory before you upload your submission.
- To check this was compressed correctly, make sure that after unzipping `your-andrew-id hw4.zip` it contains a single folder called `src`, with your implementation.
- Before submitting to Autolab, you can check if your code executes on the `linux.andrew.cmu.edu` machines, which have the same version of Python as Autolab.
- Go to Autolab at `https://autolab.andrew.cmu.edu`. There you should see 10-701 in your list of courses. Submit your `your-andrew-id hw4.zip`.
- You are allowed **only 25 submissions**. Use them wisely!