Georgia Institute of Technology

ECE 6254 - Final Take-Home

Fall 2020 - v1.0

- There are 2 problems over 12 pages (including the cover page).
- The problems are not necessarily in order of difficulty.
- Every question in a problem is worth 2 points, so problem with many questions are worth more than problems with few questions.
- Each question is graded as follows: no credit without meaningful work, half credit for partial work, full credit if essentially correct.
- Unless otherwise specified, you should concisely indicate your reasoning and show all relevant work.
- The grade on each question is based on our judgment of your level of understanding as reflected by what you have written. If we cannot read it, we cannot grade it.
- Please use a pen and not a pencil if you handwrite your solution.
- **You must submit your assignment on Gradescope**.

Problem 1: Least-Square Support Vector Regression

As seen in class and through homework problems, Support Vectors Machines (SVMs) allow us to solve machine learning problems by solving *quadratic* optimization problems. This is true in particular when using kernels that correspond to operating in a large or even infinite dimensional space. Another intriguing property of SVMs is that there is often a certain *sparsity* since the solution of the quadratic problem only relies on the support vectors. However, there are problems for which a quadratic optimization problem might still prove too complex. The goal of this take-home exam is to explore how much one could simplify SVMs without losing its benefits.

This take-home exam involves both theoretical and programming parts. The theoretical parts rely heavily on applying concepts of convex optimization seen in class, while the programming part illustrates the power of the technique. You are *allowed* and *encouraged* to use built-in python functions provided by `scikit-learn` in the programming part.

In the entire problem, we consider a dataset $\mathcal{D} \triangleq \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that consists of N pairs of feature vectors in $\mathbb{R}^d$ and associated targets in $\mathbb{R}$. We will study both classification and regression problems.

Until mentioned otherwise, we first consider a classification problem in which the targets take values $y_i \in \{\pm 1\}$. We consider the following optimization problem

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\frac{1}{2}\mathbf{w}^\intercal \mathbf{w} + \frac{C}{2}\sum_{i=1}^N \xi_i^2 \right) \text{ such that } \forall i \in [\![1, N]\!] \quad y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) = 1 - \xi_i, \tag{1}$$

where we have defined $\mathbf{w} \triangleq \begin{bmatrix} w_1 & \cdots & w_d \end{bmatrix}^\intercal$ and $\boldsymbol{\xi} \triangleq \begin{bmatrix} \xi_1 & \cdots & \xi_N \end{bmatrix}^\intercal$. Upon solving the problem, classification is performed as

$$y(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^\intercal \mathbf{x} + b\right). \tag{2}$$

[Q1] By careful inspection of (1), identify how this problem *differs* from the canonical SVM problem studied in class. Provide a concise and precise answer in words.

[Q2] Form the Lagrangian associated to the optimization problem in (1). Make sure to call α_i the Lagrange multipliers associated to the constraints and clearly indicate all the constraints, if any, and use a minus sign in front (justify why it's ok!).

[Q3] Write the KKT conditions for the Lagrangian

[Q4] Using the stationary conditions and the primal feasibility conditions obtained above, show that $\mathbf{w}$ and $\boldsymbol{\xi}$ can be eliminated from the equations to obtain a system in matrix form for solving b and $\boldsymbol{\alpha}$

$$\left[\begin{array}{c|c} 0 & \mathbf{y}^\mathsf{T} \\ \hline \mathbf{y} & \mathbf{M} \end{array}\right] \left[\begin{array}{c} b \\ \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \mathbf{1} \end{array}\right] \tag{3}$$

where $\mathbf{M} = \Omega + \frac{1}{C}\mathbf{I}$, $\mathbf{I}$ is an identity matrix, and the components of Ω are $\Omega_{ij} = y_i y_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j$.

[Q5] Show that the classification can then be performed as

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^\mathsf{T}\mathbf{x} + b\right). \tag{4}$$

[Q6] Explain in your own words the major difference between the system in (3) and the standard optimization problem for SVMs. Is there any potential advantage to solving the new system?

[Q7] Explain how to kernelize the optimization problem (3) and the classification function $\mathbf{y}(x)$ in (4).

In the rest of the problem we consider a regression problem in which the targets take values $y_i \in \mathbb{R}$. We consider the following optimization problem

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \right) \text{ such that } \forall i \in [\![1, N]\!] \quad y_i = \mathbf{w}^\intercal\mathbf{x}_i + b + \xi_i, \tag{5}$$

where we have defined $\mathbf{w} \triangleq \begin{bmatrix} w_1 & \cdots & w_d \end{bmatrix}^\intercal$ and $\boldsymbol{\xi} \triangleq \begin{bmatrix} \xi_1 & \cdots & \xi_N \end{bmatrix}^\intercal$. Upon solving the problem (5), regression is performed as

$$y(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x} + b. \tag{6}$$

[Q8] How does the problem (5) differ from support vector regression?

[Q9] Form the Lagrangian corresponding to (5) using α_i as your Lagrange multiplier (with a minus sign in front).

[Q10] Write the corresponding KKT conditions.

[Q11] Using the KKT conditions, show that $\mathbf{w}$ and $\boldsymbol{\xi}$ can be eliminated from the equations to obtain a system in matrix form for solving b and $\boldsymbol{\alpha}$

$$\left[\begin{array}{c|c} 0 & \mathbf{1}^\intercal \\ \hline \mathbf{1} & \mathbf{M} \end{array}\right] \left[\begin{array}{c} b \\ \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \mathbf{y} \end{array}\right] \tag{7}$$

where $\mathbf{M} = \Omega + \frac{1}{C}\mathbf{I}$, $\mathbf{I}$ is an identity matrix, and the components of Ω are $\Omega_{ij} = \mathbf{x}_i^\intercal \mathbf{x}_j$.

[Q12] Show that the regression then consists in computing

$$y(x) = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^\intercal \mathbf{x} + b. \tag{8}$$

[Q13] Explain how to kernelize the above algorithm.

We will now explore the above results using a numerical example. We consider a dataset consisting of 250 points $\{x_i\}$ uniformly sampled on the interval $[-7, 7]$. The function to estimate is $f(x) = \sin(\pi x)/(\pi x)$, but we only get to observe $y_i = x_i + n_i$, where n_i is the realization of a zero-mean Gaussian noise with standard deviation $\sigma = 0.1$. To ensure consistency during grading, please use the following code to generate the data.

```
Nsamples = 250
X = np.random.uniform(-7,7,size=Nsamples)
sigma = 0.1
Y = np.sin(np.pi*X)/(np.pi*X) + sigma * np.random.randn(Nsamples)
```

We will perform two different kernelized regressions using the radial basis function defined as

$$K(x, y) \triangleq \exp\left(-\frac{\|x - y\|^2}{2\gamma^2}\right) \tag{9}$$

[Q14] Provide a graph of the true function $f(x)$ in solid red (–), showing the data samples superposed as blue dots (•). No need to provide your code.

[Q15] Run a standard support vector regression using the `scikit-learn` toolbox (or equivalent). To ensure consistency, use the hyperparameters `epsilon`=0.1, `C`=1e3, `kernel='rbf'` and `gamma=0.25`. Plot the resulting regression by evaluating the SVR on 100 uniformly spaced points between -7 and 7, superposing the plot in solid green (–) on top of the previous plots. Show the code to run the support vector regression but don't show the code for creating the plot.

[Q16] Create a plot showing the absolute value of the 250 Lagrange multipliers in order of increasing magnitude. Show the code you use to retrieve the multipliers but not the code used for creating the graph. Comment on the result.

[Q17] Implement a solver for the kernelized version of (7) using radial basis functions to implement (8). Attach you full code as an appendix. Superpose the plot of the resulting regression in solid magenta (–) on top of the previous plots. To ensure grading consistency use hyperparameters $C = 100$ and $gamma = 0.5$

[Q18] Create a plot showing the absolute value of the 250 Lagrange multipliers in order of increasing magnitude. Show the code you use to retrieve the multipliers but not the code used for creating the graph. Comment on the result, especially how this differs from the support vector regression.

Problem 2: Multiple choices

In the following you do not need to provide a justification. Simply circle your answer(s). Each question carries the same weight.

[Q1] Consider the following dataset in $\mathbb{R}^2$:

$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 4 \\ 0 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Suppose we run the k-means clustering algorithm on this dataset, with initial centers at $\mathbf{m}_1 = \mathbf{x}_2$ and $\mathbf{m}_2 = \mathbf{x}_4$. Upon convergence, the centers will be at

(a) $\mathbf{x}_2$ and $\mathbf{x}_4$

(b) $\mathbf{x}_2$ and the midpoint between $\mathbf{x}_3$ and $\mathbf{x}_4$

(c) $\mathbf{x}_3$ and the midpoint between $\mathbf{x}_3$ and $\mathbf{x}_4$

(d) $\mathbf{x}_4$ and the midpoint between $\mathbf{x}_3$ and $\mathbf{x}_4$

(e) $\mathbf{x}_3$ and $\mathbf{x}_4$

[Q2] Which of the following cannot be interpreted as a plug-in method for the Bayes classifier? (Circle all that apply)

(a) k-nearest neighbor classifier

(b) linear discriminant analysis

(c) logistic regression

(d) naïve Bayes classifier

(e) support vector machines

[Q3] Suppose that $\mathcal{H}$ shatters a dataset of k vectors in $\mathbb{R}^d$. What can we say about the VC dimension of $\mathcal{H}$?

(a) $d_{\text{VC}} \leq k$

(b) $d_{\text{VC}} = k$

(c) $d_{\text{VC}} = k + 1$

(d) $d_{\text{VC}} \geq k$

(e) $d_{\text{VC}} \geq k + 1$

(f) none of the above

[Q4] Which of the following are true statements about the Lagrange multipliers λ_i in the dual formulation of the support vector machine optimization problem? (Circle all that apply.)

(a) $\lambda_i \neq 0$ indicates that $\mathbf{x}_i$ is a support vector

(b) $\lambda_i = 0$ indicates that $\mathbf{x}_i$ is a support vector

(c) if $\lambda_i \neq 0$, then $\mathbf{x}_i$ must be classified correctly

(d) if $\lambda_i = 0$, then $\mathbf{x}_i$ must be classified correctly

(e) if $\lambda_i \neq 0$, then we must have $\lambda_i = C$

(f) if $\lambda_i < C$, then we must have $\lambda_i = 0$

[Q5] In terms of the bias-variance decomposition, a 1-nearest neighbor classifier has _____ than a 3-nearest neighbor classifier. (Circle all that apply.)

(a) lower variance

(b) higher variance

(c) the same variance

(d) lower bias

(e) higher bias

(f) the same bias

[Q6] For a fixed choice of learning model, as the size of the training set goes to ∞, the model trained on the data will have (circle all that apply):

(a) lower variance

(b) higher variance

(c) the same variance

(d) lower bias

(e) higher bias

(f) the same bias

[Q7] Given $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, you perform PCA and choose k principal components. Can you reconstruct the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ perfectly from their projections onto these k principal components? (Circle all that apply.)

(a) yes, of course you can

(b) yes, when $k < n$

(c) yes, when $k \geqslant n$

(d) yes, when $k = d$

(e) no, you can never do it

[Q8] Let h^* be a classifier chosen from a set $\mathcal{H}$ of finite size using empirical risk minimization on n independent data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$, $y \in \{\pm 1\}$. Let $R(h^*) - \hat{R}_n(h^*)$ be the *excess risk*, that is, the difference between the true risk of h^* and the empirical risk. Which of the following is most likely to decrease the excess risk the most?

(a) halving the size of $\mathcal{H}$

(b) doubling the size of $\mathcal{H}$

(c) halving the number of data points n

(d) doubling the number of data points n

(e) doubling the number of features d