Boston University
Department of Electrical and Computer Engineering

ENG EC 414 Introduction to Machine Learning

HW 6 Solution

Issued: Fri 16 Oct 2020 **Due:** 10:00am Fri 23 Oct 2020 in Gradescope (non-code) + Blackboard

Important: Before you proceed, please read the documents pertaining to *Homework formatting and submission guidelines* in the Homeworks section of Blackboard. **In particular, for computer assignments you are prohibited from using any online code or built-in MATLAB functions except as indicated in the problem or skeleton code (when provided).**

Important: To obtain full grade, please clearly motivate all your answers.

Note: Problem difficulty = number of coffee cups ☕

Problem 6.1 [6pts] *(Verify that a function is a Kernel)*
In class we said that in general it is a difficult task to understand if a function is a kernel or not. However, in some case it is easy: it is enough to find a transformation ϕ such that $\mathcal{K}(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})^\top \phi(\mathbf{v})$ for all $\mathbf{u}$ and $\mathbf{v}$.

(a) [2pts] Consider the kernel $\mathcal{K}_{\text{linear}}(\mathbf{u},\mathbf{v}) := \mathbf{u}^\top\mathbf{v}$, where $\mathbf{u},\mathbf{v} \in \mathbb{R}^d$. Show that it is a kernel by finding the corresponding function ϕ.

Solution: In this question, there are an infinite of functions ϕ that will work. The simplest one is the identity function: $\phi : \mathbb{R}^d \to \mathbb{R}^d$ defined as $\phi(\boldsymbol{x}) = \boldsymbol{x}$.

(b) [2pts] Consider the kernel $\mathcal{K}_{\text{product}}(\mathbf{u},\mathbf{v}) := u_1u_2v_1v_2$, where $\mathbf{u},\mathbf{v} \in \mathbb{R}^2$. Show that it is a kernel by finding the corresponding function ϕ.

Solution: Again, there might be multiple solutions. The simplest one is $\phi : \mathbb{R}^2 \to \mathbb{R}$ defined as $\phi(\boldsymbol{x}) = x_1x_2$.

(c) [2pts] Consider the kernel $\mathcal{K}_{\text{one}}(\mathbf{u},\mathbf{v}) := 1$, where $\mathbf{u},\mathbf{v} \in \mathbb{R}^d$. Show that it is a kernel by finding the corresponding function ϕ.

Solution: Infinite solutions even here, the simplest one maps everything to 1, that is $\phi : \mathbb{R}^d \to \mathbb{R}$ defined as $\phi(\boldsymbol{x}) = 1$.

Problem 6.2 [22pts] *(Soft-Margin Binary Kernel SVM)*
In this problem, we will implement and use the Subgradient Descent Algorithm to learn the parameters of the soft-margin Kernel SVM classifier. Let's start summarizing the derivation we did in class. The following notes are a useful companion to the lecture slides as well.

Let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be a training set of m examples consisting of feature vector $\mathbf{x}_j \in \mathbb{R}^d$ and label $y_j \in \{-1, +1\}$ for each j. The cost function (in the unconstrained form) of the binary soft-margin SVM optimization problem is given by:

$$(\mathbf{w}_{\text{SVM}}, b_{\text{SVM}}) := \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{j=1}^{m} \text{hinge}\left(y_j\left(\mathbf{w}^\top\mathbf{x}_j + b\right)\right)$$

where $\text{hinge}(t) := \max(0, 1-t)$. Since $\mathbf{w}_{\text{SVM}}$ is a linear combination of the feature vectors in the training set, we can set

$$\mathbf{w} = \sum_{j=1}^{m} \alpha_j \mathbf{x}_j = X^\top \boldsymbol{\alpha}, \quad \text{where } \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}_{m\times 1} \text{ and } X = \begin{bmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{bmatrix}_{m\times d},$$

and solve for $\mathbf{w}$ in terms of $\boldsymbol{\alpha}$. Then, $\mathbf{w}_{\text{SVM}} = X\boldsymbol{\alpha}_{\text{SVM}}$ where,

$$(\boldsymbol{\alpha}_{\text{SVM}}, b_{\text{SVM}}) := \arg\min_{\boldsymbol{\alpha}, b} \ \frac{1}{2}\boldsymbol{\alpha}^\top XX^\top \boldsymbol{\alpha} + c\sum_{j=1}^{m} \text{hinge}\left(y_j\left(\boldsymbol{\alpha}^\top X\mathbf{x}_j + b\right)\right).$$

For convenience, let us define the following quantities:

$$K := XX^\top, \quad \text{and for } j = 1, \ldots, m,\ K_j := X\mathbf{x}_j,\ K_j^{\text{ext}} := \begin{bmatrix} 1 \\ K_j \end{bmatrix}_{(m+1)\times 1}.$$

K is an $m \times m$ matrix consisting of inner products between all pairs of training feature vectors, i.e., $\forall i, j = 1, \ldots, m$, the ij-th entry of K is given by $K_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$. Moreover, K_j is the j-th column of K and K_j^{ext} is the column vector K_j extended by one row by appending the value 1 at the beginning. We introduce K_j^{ext} to easily deal with the bias term b, similarly to what we did in LS. If we further define $\tilde{\boldsymbol{\alpha}} := \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}$, then the soft-margin SVM is the solution to the following optimization problem:

$$\tilde{\boldsymbol{\alpha}}_{\text{SVM}} := \arg\min_{\tilde{\boldsymbol{\alpha}}} F(\tilde{\boldsymbol{\alpha}}), \quad \text{where } F(\tilde{\boldsymbol{\alpha}}) := \frac{1}{2}\tilde{\boldsymbol{\alpha}}^\top \begin{bmatrix} 0 & 0 \\ 0 & K \end{bmatrix} \tilde{\boldsymbol{\alpha}} + c\sum_{j=1}^{m} \text{hinge}\left(y_j \tilde{\boldsymbol{\alpha}}^\top K_j^{\text{ext}}\right) \tag{1}$$

Given a test feature vector $\mathbf{x}_{\text{test}}$, the label predicted by the soft-margin SVM is given by:

$$f_{\text{SVM}}(\mathbf{x}_{\text{test}}) = \text{sign}\left(\mathbf{w}_{\text{SVM}}^\top \mathbf{x}_{\text{test}} + b_{\text{SVM}}\right) = \text{sign}\left(\boldsymbol{\alpha}_{\text{SVM}}^\top X\mathbf{x}_{\text{test}} + b_{\text{SVM}}\right) = \text{sign}\left(\tilde{\boldsymbol{\alpha}}_{\text{SVM}}^\top K_{\mathbf{x}_{\text{test}}}^{\text{ext}}\right)$$

where $K_{\mathbf{x}_{\text{test}}} := X\mathbf{x}_{\text{test}}$ is an $m \times 1$ column vector whose i-th element is equal to $\mathbf{x}_i^\top \mathbf{x}_{\text{test}}$. Thus,

$$\boxed{f_{\text{SVM}}(\mathbf{x}_{\text{test}}) = \text{sign}\left(\tilde{\boldsymbol{\alpha}}_{\text{SVM}}^\top K_{\mathbf{x}_{\text{test}}}^{\text{ext}}\right)} \tag{2}$$

Observe that the training of the soft-margin SVM depends on the feature vectors in the training set only through their pairwise inner products which is captured by the matrix K (see Eq. (1)). Similarly, predicting the label of a test sample also depends on the feature vectors of the training set and the test feature vector only through the inner products between the training and test feature vectors captured by the vector $K_{\mathbf{x}_{\text{test}}}^{\text{ext}}$ (see Eq. (2)). The *Kernel SVM* generalizes this idea by replacing the inner products between any two feature vectors $\mathbf{u}$ and $\mathbf{v}$ by $\mathcal{K}(\mathbf{u}, \mathbf{v})$, where $\mathcal{K}(\cdot, \cdot)$ is a real-valued function of two vector-variables, referred to as a *kernel*.

To summarize, the Kernel SVM is learned by solving Eq. (1) using K equal to an $m \times m$ matrix with $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ for all $i, j = 1, \ldots, m$ and K_j equal to the j-th column of K. The prediction for a test point $\mathbf{x}_{\text{test}}$ is obtained by evaluating Eq. (2) using an $m \times 1$ vector $K_{\mathbf{x}_{\text{test}}}^{\text{ext}}$ whose i-th row is equal to $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_{\text{test}})$.

The optimization problem described by Eq. (1) is convex and unconstrained and can therefore be solved by the Subgradient Descent algorithm. A subgradient of the hinge function $\text{hinge}(t)$ is the following:

$$\begin{cases} -1, & \text{when } t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 1 Subgradient Descent for Kernel SVM

input: Training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, Maximum number of iterations T, Kernel function, Trade-off hyperparameter c
initialize: $\tilde{\boldsymbol{\alpha}}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  $obj_t = F(\tilde{\boldsymbol{\alpha}}_t)$ *//Evaluate the value of the objective function in* $\tilde{\boldsymbol{\alpha}}_t$
  $zeroOneAverageLoss_t = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}[y_i \neq f_{\tilde{\boldsymbol{\alpha}}_t}(\boldsymbol{x}_i)]$ *//Average 0/1 loss on the training set in* $\tilde{\boldsymbol{\alpha}}_t$
  $\mathbf{g}_t = \begin{bmatrix} 0 & 0 \\ 0 & K \end{bmatrix} \tilde{\boldsymbol{\alpha}}_t$ *// Compute subgradient first term*
  //Add subgradient second term
  for $j = 1, 2, \ldots, m$ **do**
    if $(y_j \tilde{\boldsymbol{\alpha}}_t^\top K_j^{\text{ext}} < 1)$ **then**
      $\mathbf{g}_t = \mathbf{g}_t - c y_j \, K_j^{\text{ext}}$
    end if
  end for
  Update parameters: $\tilde{\boldsymbol{\alpha}}_{t+1} = \tilde{\boldsymbol{\alpha}}_t - \eta_t \mathbf{g}_t$
end for
Extract $\boldsymbol{\alpha}$ and b from $\tilde{\boldsymbol{\alpha}}_{T+1}$
return $\boldsymbol{\alpha}, b, obj, zeroOneAverageLoss$

We can use the above subgradient to have the pseudocode for implementing subgradient descent for soft-margin Kernel SVM loss as follows:

We will use the *kernel-svm-2rings* dataset. We will use the entire dataset for training and visualization. There will be no testing done. We will use the Gaussian kernel which is defined as $\mathcal{K}(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$, where γ is called the bandwidth parameter. Use the following choices of hyperparameters:

$$T = 10000, \quad \eta_t = \frac{1}{t}, \quad c = 100, \quad \gamma = 2 \, .$$

Note that for non-differentiable functions we said that we should take the average of the solutions and use a learning rate of $\frac{1}{\sqrt{t}}$. However, this is an easier problem to solve (because it is 1-*strongly convex*), so we can use a learning rate that goes to zero faster and we can avoid the averaging.

(a) [10pts] Implement the algorithm in Algorithm 1 in a Matlab function with prototype

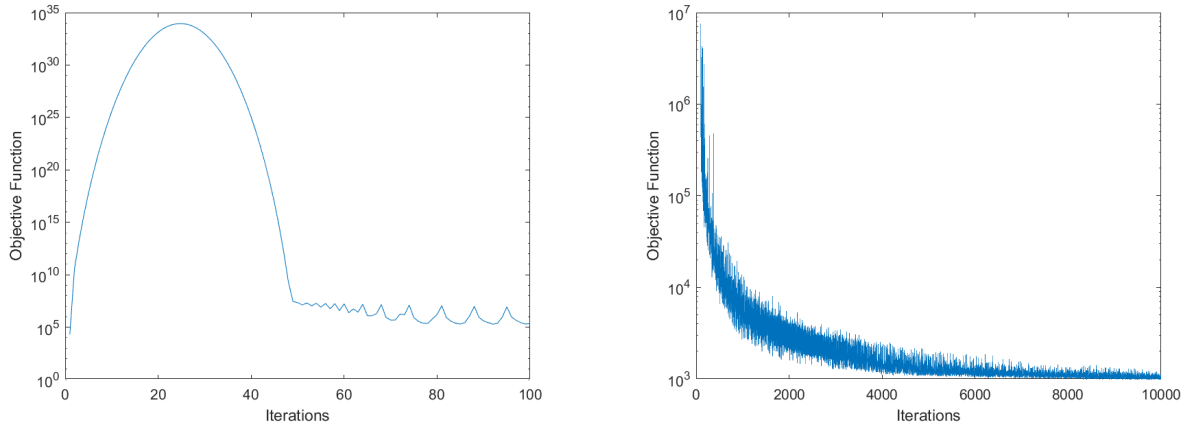
```
[alpha, b, obj, zeroOneAverageLoss] = train_ksvm_sd(X, y, T, c, gamma)
```

where `alpha` $\in \mathbb{R}^m$ and `b` $\in \mathbb{R}$ are the solution found by subgradient descent, `obj` $\in \mathbb{R}^T$ is the vector of objective function values, `zeroOneAverageLoss` $\in \mathbb{R}^T$ is the vector of the average 0/1 loss on the training samples, `X` is the matrix of the training samples $\in \mathbb{R}^{m \times d}$, `y` is the vector of labels $\in \mathbb{R}^m$, `T` is the maximum number of iterations to run, `c` is the trade-off hyperparameter in Eq. (1), `gamma` is the bandwidth parameter of the Gaussian kernel. There is no skeleton code this time.

Solution: See code.

(b) [2pts] Create a plot which shows how the objective function F evolves with iteration number t. Is the objective function decreasing? Should it be decreasing? Discuss its behavior.

Solution:



Iterations 1 to 100 Iterations 101 to $t_{\max}$

Figure 1: Evolution of objective function $F(\tilde{\alpha})$ with iteration number when training a soft-margin kernel-SVM classifier.

The cost versus iteration curve is plotted in Figure 1. The cost starts at the value 20000 corresponding to zero initialization and then "sky-rockets" to an extremely large value (about 9×10^{33}) over the course of the first 25 iterations or so. Thereafter, the cost quickly plummets to a value that is much smaller than the initial value of 20000. Overall, with enough iterations subgradient descent does minimize the function, but, as said in class, there is no reason to expect that it will monotonically decrease the objective function.

(c) [2pts] Create a plot which shows how the average 0/1 loss on the training set evolves with iteration number. Is it decreasing? Should it be decreasing? Discuss its behavior.

Solution:

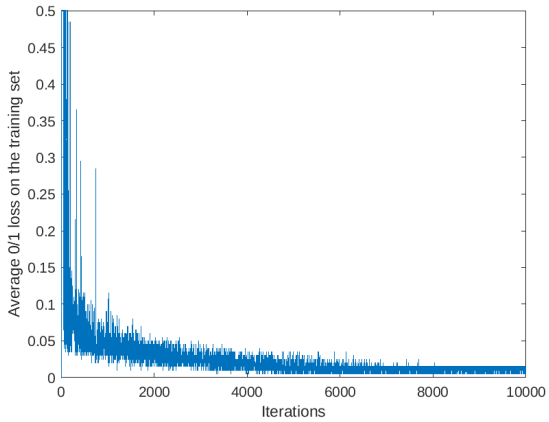


Figure 2: Evolution of training average 0/1 loss with iteration number when training a soft-margin binary kernel-SVM classifier.

The training average 0/1 loss versus iteration curve is plotted in Figure 2. It starts out at a value which equivalent to random guessing and then gradually decreases, with moderate-magnitude short-term

fluctuations. The magnitude of the short-term fluctuations appear to gradually diminish with more iterations. With enough iterations, the 0/1 loss is expected to decrease because we are minimizing a convex upper bound of the 0/1 loss.

(d) [2pts] After the subgradient descent algorithm terminates, report the final training error measured with the 0/1 loss. Discuss your observations.

Solution:

Only 2 training examples out of 200 are misclassified and consequently the resulting average 0/1 training loss is very low (0.01). This shows that the kernel SVM has succeeded, to a great extent, in separating the examples from the two classes even though the ideal decision boundary between the two classes is curved.

(e) [6pts] *(Visualization of decision boundary and margins)* Plot on the same graph the following: (1) the training set (use different colors for the two classes) and (2) the decision boundary and the margins of the trained Kernel SVM. *Hint:* To plot the decision boundary and the margins, you can use the function `contour`, look for its Matlab help.

Solution:

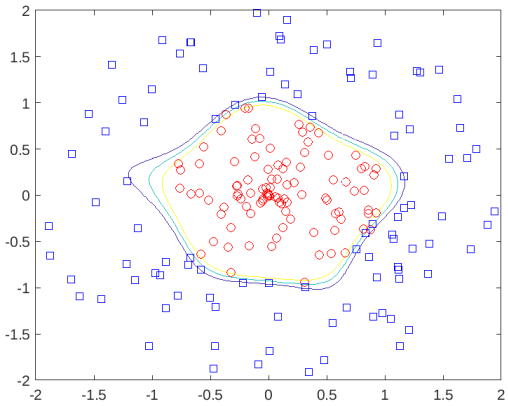


Figure 3: Scatter plot of dataset with decision boundary.

As can be seen from the scatter plot of the dataset in Figure 3, the examples from the two classes are distributed within concentric ring-like regions (one for each class). The regions are non-overlapping except for a small "transition ring" where a small subset of examples from both classes are "mixed together". The Kernel SVM decision boundary based on the Gaussian kernel nicely separates the two classes. The boundary forms a smooth closed curve, but it is not a circle.

Problem 6.3 [13pts] *(Questions)*
Clearly explain your answers.

(a) [2pts] Consider the soft-margin SVM formulation. Does the variance increases or decreases with c?

Solution: Increasing c, we give more importance to the training errors and less to the margin. This makes the resulting classifier more complex, hence it increases the variance.

(b) [2pts] Consider a linearly separable training set of m samples. Let's now gather other n different training samples for the same classification problem, to have a total of $m + n$ samples. Is the new dataset still linearly separable?

Solution: Linear separability depends on the training data. A linearly separable training set is not guaranteed to remain linearly separable after adding more training data.

(c) [2pts] We have a linearly separable dataset. We run the Perceptron till convergence and measure the margin of the obtained solution, that is we measure the minimum distance between the hyperplane and the training points. Will this margin be smaller, equal, or bigger than the margin of the hard-margin SVM solution on the same dataset?

Solution: By definition, the hard-margin SVM solution has the biggest possible margin on a linearly separable dataset. Hence, the Perceptron's solution can have equal or smaller than the hard-margin SVM's one.

(d) [2pts] You have a binary classification dataset with roughly equal number of positive and negative examples. You use a soft-margin SVM with some setting of c and the 0/1 loss on the training and validation sets are close to 50%. You want to try a different value of c to try to improve the performance: Would you decrease or increase c? Why?

Solution: Given that the training error is high, we deduce that we might be in an underfitting regime, that is the classifier is too simple for our task. Also, we are not overfitting because training and validation error are similar. So, we might try to make the classifier more complex, reducing the margin or equivalently giving more importance to the training error. This is achieved increasing c.

(e) [2pts] You have a binary classification dataset with roughly equal number of positive and negative examples. You use a soft-margin SVM with some setting of c and the 0/1 loss on the training set is close to 0 and validation error is close to 50%. You want to try a different value of c to try to improve the performance: Would you decrease or increase c? Why?

Solution: In this situation, the classifier is complex enough for our task because the training error is low. However, we are in an overfitting situation because the validation error is high. So, it is likely that there is too much variance and we might try to reduce it making the classifier simpler (and increasing the bias). In a soft-margin SVM, this is achieved reducing c.

(f) [3pts] ☕ Using the definition of subgradient we gave in class, find a subgradient of the function $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$ in $\boldsymbol{x} = \mathbf{0} \in \mathbb{R}^d$. This would be useful if we want to change the regularizer from $\|\boldsymbol{x}\|_2^2$ to $\|\boldsymbol{x}\|_2$ and use subgradient descent. Hint: there are infinite subgradients in that point, any of them will do.

Solution: By definition, $\boldsymbol{g}$ is a subgradient of a function f in a point $\boldsymbol{x}$ if

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^\top(\boldsymbol{y} - \boldsymbol{x}), \ \forall \boldsymbol{y} \in dom f \ .$$

In our case, we have $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$ in $\boldsymbol{x} = \mathbf{0} \in \mathbb{R}^d$. Hence, we need $\forall \boldsymbol{y} \in \mathbb{R}^d$ a $\boldsymbol{g}$ such that,

$$\|\boldsymbol{y}\|_2 \geq \|\mathbf{0}\|_2 + \boldsymbol{g}^\top(\boldsymbol{y} - \mathbf{0}) = \boldsymbol{g}^\top \boldsymbol{y}$$

For any vector $\boldsymbol{g}$ with L2 norm less or equal than 1, using Cauchy–Schwarz inequality we have that

$$\boldsymbol{g}^\top \boldsymbol{y} \leq \|\boldsymbol{g}\|_2 \|\boldsymbol{y}\|_2 \leq \|\boldsymbol{y}\|_2 \ .$$

Hence, any vector with L2 norm less or equal than 1 is a subgradient of $\|\mathbf{0}\|_2$ in $\mathbf{0}$. In particular, the null vector is also a valid subgradient.

Code-submission via Blackboard: You must prepare 1 file: `train_ksvm_sd.m` for Problem 6.2(a). Place it in a **single** directory which should be zipped and uploaded into Blackboard. Your directory must be named as follows: `<yourBUemailID>_hwX` where `X` is the homework number. For example, if your BU email address is `charles500@bu.edu` then for homework number 6 you would submit a single directory named: `charles500_hw6.zip` which contains all the MATLAB code (and only the code).

Three corresponding skeleton code files are provided for your reference. Reach-out to the TAs via Piazza and their office/discussion hours for questions related to coding.