CS 475 Machine Learning: Homework 2
Supervised Classifiers 2
Analytical Problems

Due: Sunday, March 8, 2020, 11:59 pm

50 Points Total Version 1.1

YOUR_NAME (YOUR_JHED)

Instructions

We have provided this LaTeX document for turning in this homework. We give you one or more boxes to answer each question. The question to answer for each box will be noted in the title of the box.

Other than your name, do not type anything outside the boxes. Leave the rest of the document unchanged.

Do not change any formatting in this document, or we may be unable to grade your work. This includes, but is not limited to, the height of textboxes, font sizes, and the spacing of text and tables. Additionally, do not add text outside of the answer boxes. Entering your answers are the only changes allowed.

We strongly recommend you review your answers in the generated PDF to ensure they appear correct. We will grade what appears in the answer boxes in the submitted PDF, NOT the original latex file.

Notation

$\mathbf{x_i}$ One input data vector. $\mathbf{x_i}$ is M dimensional. $\mathbf{x_i} \in \mathbb{R}^{1 \times M}$.
We assume $\mathbf{x_i}$ is augmented with a 1 to include a bias term.

$\mathbf{X}$ A matrix of concatenated $\mathbf{x_i}$'s. There are N input vectors, so $\mathbf{X} \in \mathbb{R}^{N \times M}$

y_i The true label for input vector $\mathbf{x_i}$. In regression problems, y_i is a continuous value.
In general y_i can be a vector, but for now we assume y_i is a scalar. $y_i \in \mathbb{R}^1$.

$\mathbf{y}$ A vector of concatenated y_i's. There are N input vectors, so $\mathbf{y} \in \mathbb{R}^{N \times 1}$

$\mathbf{w}$ A weight vector. We are trying to learn the elements of $\mathbf{w}$.
$\mathbf{w}$ is the same number of elements as $\mathbf{x_i}$ because we will end up computing the dot product $\mathbf{x_i} \cdot \mathbf{w}$.
$\mathbf{w} \in \mathbb{R}^{M \times 1}$. We assume the bias term is included in $\mathbf{w}$.

Notes: In general, a lowercase letter (not boldface), a, indicates a scalar.
A boldface lowercase letter, $\mathbf{a}$, indicates a vector.
A boldface uppercase letter, $\mathbf{A}$, indicates a matrix.

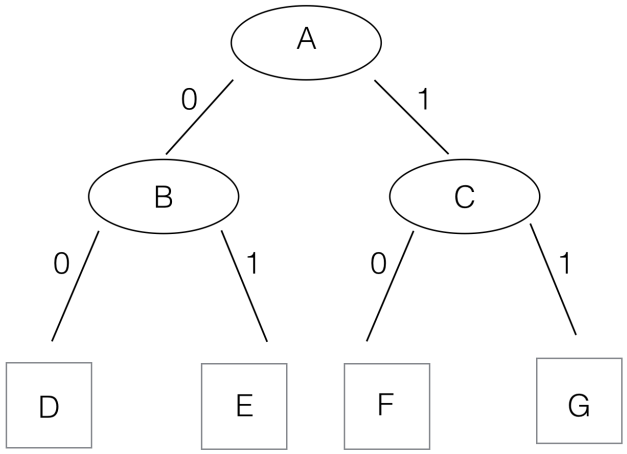
1) Decision Trees (10 points)

Consider the classification task where you want to predict y given $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$.

x_1	x_2	x_3	x_4	x_5	y
1	0	0	0	1	1
1	0	1	0	1	1
0	1	0	1	1	1
0	0	0	1	1	0
1	1	1	0	1	0
1	0	1	1	1	0
1	0	0	1	1	0
0	1	0	0	1	0

(1) (4 points) Construct a decision tree based on the above training examples following the algorithm we specified in class using the information gain criterion and a maximum depth of 2. As an additional base case, stop expanding if all training examples have the same label. You may use each feature at most once along any path from the root to a leaf.

Using the decision tree schematic below, specify the correct feature number for internal nodes A, B, and C. For each nodes D, E, F, and G, specify the correct label. Put answers in the pre-formatted table by the "?" with the correct feature or label.



Node	Id or label
A=	?
B=	?
C=	?
D=	?
E=	?
F=	?
G=	?

(2) (2 points) Apply the decision tree learned in part 1 of this question to classify the following new data points. Replace the "?" in the table below with the classifier's prediction.

x_1	x_2	x_3	x_4	x_5	y
0	1	1	1	1	?
1	1	0	1	1	?
1	1	0	0	1	?

(3) (4 points) For the training dataset in part 1, can any of the methods listed in the table below obtain a *training accuracy* of 100% using only the features given? Answer below by replacing the "?" with "Y" or "N".

Method	Y/N
Decision tree of depth 1	?
Decision tree of unlimited depth	?
Linear Kernel SVM	?
Quadratic Kernel SVM	?
AdaBoost with decision stumps	?

2) Hinge Loss (10 points)

Linear SVMs using a squared hinge loss can also be formulated as an unconstrained optimization problem:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{N} H(y_i \mathbf{w}^T \mathbf{x}_i) \right], \tag{1}$$

where λ is the regularization parameter and $H(a) = \max(0, 1-a)^2$ is the squared hinge loss function. The hinge loss function can be viewed as a convex surrogate of the 0/1 loss function $\mathbb{1}(a \leq 0)$.

(a) (3 points) Compared with the standard hinge loss function, what do you think are the advantages and disadvantages of the square hinge loss function?

(b) (3 points) The function $G(a) = \max(-a, 0)^2$ can also approximate the 0/1 loss function. What is the disadvantage of using this function instead?

(c) (4 points) We can choose a different loss function $H'(a) = \max(0.5 - a, 0)^2$. Specifically, the new objective becomes:

$$\mathbf{w}'^{\star} = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\lambda' \|\mathbf{w}\|^2 + \sum_{i=1}^{N} H'(y_i \mathbf{w}^T \mathbf{x}_i)) \right]. \tag{2}$$

In the situation that the classification doesn't change, consider how switching to H' from H will effect the solution of the objective function. Explain your answer in terms of the relationship between λ and λ'.

3) Kernel Trick (10 points)

The kernel trick extends SVMs to learn nonlinear functions. However, an improper use of a kernel function can cause serious over-fitting. Consider the following kernels.

(a) (3 points) Inverse Polynomial kernel: given $\|x\|_2 \leq 1$ and $\|x'\|_2 \leq 1$, we define $K(x, x') = 1/(d - x^\top x')$, where $d \geq 2$. Does increasing d make over-fitting more or less likely?

(b) (4 points) Chi squared kernel: Let x_j denote the j-th entry of x. Given $x_j > 0$ and $x'_j > 0$ for all j, we define $K(x, x') = \exp\left(-\sigma \sum_j \frac{(x_j - x'_j)^2}{x_j + x'_j}\right)$, where $\sigma > 0$. Does increasing σ make over-fitting more or less likely?

We say K is a kernel function, if there exists some transformation $\phi : \mathbb{R}^m \to \mathbb{R}^{m'}$ such that $K(x_i, x_{i'}) = \langle \phi(x_i), \phi(x_{i'}) \rangle$.

(c) (3 points) Let K_1 and K_2 be two kernel functions. Prove that $K(x_i, x_{i'}) = K_1(x_i, x_{i'}) + K_2(x_i, x_{i'})$ is also a kernel function.

4) Dual Perceptron (10 points)

(a) (3 points) You train a Perceptron classifier in the primal form on an infinite stream of data. This stream of data is not-linearly separable. Will the Perceptron have a bounded number of prediction errors?

(b) (4 points) Switch the primal Perceptron in the previous step to a dual Perceptron with a linear kernel. After observing T examples in the stream, will the two Perceptrons have learned the same prediction function?

(c) (3 points) What computational issue will you encounter if you continue to run the dual Perceptron and allow T to approach ∞? Will this problem happen with the primal Percepton? Why or why not?

5) Linear SVM (10 points)

Suppose we are given the four data points in $\mathbb{R}^2$: $[(3,1),(3,-1),(4,2),(4,-2)]$ labeled as positive and four data points in $\mathbb{R}^2$: $[(1,0),(0,2),(2,0),-1,2)]$ labeled as negative. The goal is to build an SVM classifier.

(a) (3 points) What are the support vectors for this classifier.

(b) (4 points) Write out the optimization problem in dual formulation (include only the support vectors). And solve the problem.

(c) (3 points) Derive the hyperplane for classification. Hint: You can solve for b using the following equation:

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}$$