CS178 Midterm Exam

Machine Learning & Data Mining: Winter 2014

Wednesday February 12th, 2014

Your name:

Your UCINetID (e.g., myname@uci.edu):

Your seat (row and number):

- Total time is 50 minutes. READ THE EXAM FIRST and organize your time; don't spend too long on any one problem.
- Please **write clearly** and **show all your work**.
- If you need clarification on a problem, please raise your hand and wait for the instructor to come over.
- Turn in any scratch paper with your exam.

Problem 1: (9 points) Bayes Classifiers and Naïve Bayes

In this problem you will use Bayes Rule: $p(y|x) = p(x|y)p(y)/p(x)$ to perform classification. Suppose we observe some training data with two binary features x_1, x_2 and a binary class y. After learning the model, you are also given some validation data.

Table 1: Training Data

x_1	x_2	y
1	1	0
1	0	0
1	0	1
0	0	0
0	1	1
1	1	0
0	0	1
1	0	1

Table 2: Validation Data

x_1	x_2	y
1	1	0
1	0	1
0	1	0
0	0	1

In the case of any ties, we will prefer to predict class 0.

(a) What is the classification validation error rate of the naïve Bayes classifier on these data?

(b) What is the classification validation error rate of the joint Bayes classifier on these data?

(c) Suppose you are given the option to use only one feature to train the data – in other words, you may train a model that uses either x_1 or x_2, but not both. Based on validation set performance, which model would you select? Report the validation error of the model you chose.

Problem 2: (8 points) Gradient Descent

Suppose that we have a linear classifier on two features,

$$\hat{y} = T[\, a + bx_1 + cx_2 \,]$$

with three parameters $\theta = [\, a\,,\, b\,,\, c\,]$ and $T[z]$ being the sign function. We decide to train our classifier using gradient descent on the "exponential loss",

$$J(\theta) = \frac{1}{m} \sum_i \exp\left[\, -y^{(i)}\,(a + bx_1^{(i)} + cx_2^{(i)})\,\right]$$

where $y^{(i)} \in \{-1, +1\}$.

(a) Write down the gradient of the loss function.

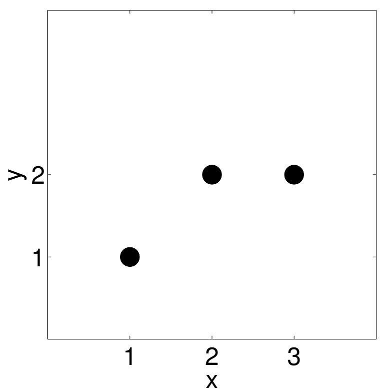
(b) Give pseudocode for a gradient descent function `theta = train(X,Y)`, including all necessary elements for it to work.

Problem 3: (8 points) Cross-validation and Linear Regression

Consider the following data points, copied in each part. We wish to perform linear regression to minimize the mean squared error of our predictions.

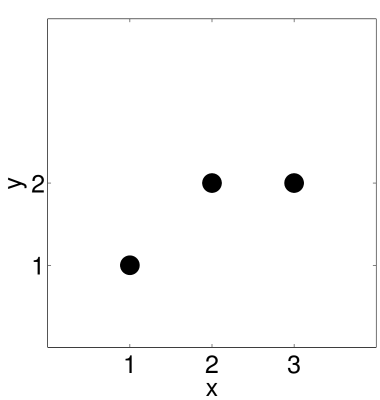
(a) Compute the leave-one-out cross-validation error of a zero-order (constant) predictor,

$$\hat{y}(x) = \theta_0$$



(b) Compute the leave-one-out cross-validation error of a first-order (linear) predictor,

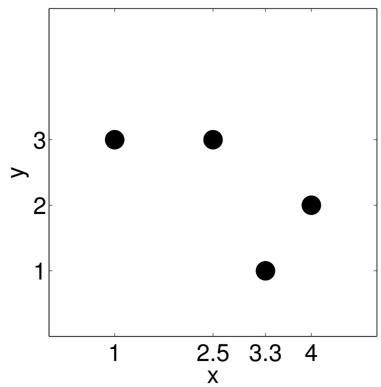
$$\hat{y}(x) = \theta_0 + \theta_1\, x$$



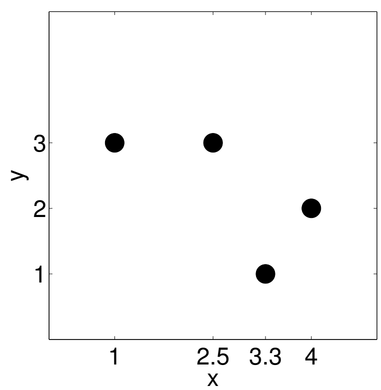
Problem 4: (9 points) K-Nearest Neighbor Regression

Consider a **regression** problem for predicting the real-valued target y (vertical axis) given the data points shown on the left using a k-nearest neighbor regression algorithm. Under each of the following scenarios, **(a) sketch** the regression function when trained on all the data; **(b) compute** its resulting training error (MSE). (If you would like you may leave an arithmetic expression, e.g., leave MSE as "$(.2)^2 + (.6)^2$")

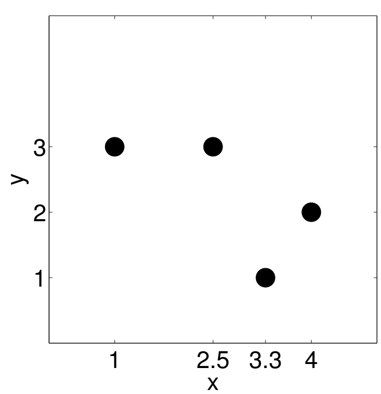
(a) $k = 1$.



(b) $k = 2$



(c) $k = 3$



Problem 5: (8 points) Multiple Choice

For the following questions, assume that we have m data points $y^{(i)}$, $x^{(i)}$, $i = 1 \ldots m$, each with n features, $x^{(i)} = [x_1^{(i)} \ \ldots \ x_n^{(i)}]$. **Circle one answer for each:**

Suppose we are using a Gaussian Bayes classifier to discriminate between two classes $y \in \{-1, +1\}$. If we force the classifier to use equal covariances for the models $p(x|y)$, this will typically make it **more** **equally** **less** likely to overfit the data.

Suppose that we are training a linear classifier (perceptron). Discarding features from our data before training will typically make it **more** **equally** **less** likely to overfit the data.

Suppose we are using gradient descent to train a linear classifier. Increasing the maximum number of iterations performed by the algorithm from 10 to 100 will most likely make it **more** **equally** **less** likely to overfit the data.

Suppose that, when training a k-nearest neighbor model, we discard the second half of our training data to reduce memory overhead (i.e., we use the first $m/2$ data points). This will most likely make our model **more** **equally** **less** likely to overfit the data.

After training a k-nearest neighbor model, we increase the value of k. This will most likely make our model **more** **equally** **less** likely to overfit the data.

Suppose that before training a linear regression model, we add m additional, artificial "all zero" points to our training set, with $y^{(i)} = 0$ and $x^{(i)} = \underline{0}$ for $i = m + 1 \ldots 2m$. This will most likely make our model **more** **equally** **less** likely to overfit the data. (Hint: is our model more or less sensitive to the training data points?)

True or **false**: if two models have the same VC dimension, they are equally likely to overfit the data.

True or **false**: if the VC dimension of a model is H, then the model can shatter any set of H training points.