CS273A Midterm Exam

Introduction to Machine Learning: Winter 2020

Tuesday February 11th, 2020

Your name:

Solutions

Row/Seat Number:

Your ID #(e.g., 123456789)

31415926S

UCINetID (e.g.ucinetid@uci.edu)

panteater@uci

- Please put your name and ID on every page.
- Total time is 80 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 or TA to come over.
- You may use one sheet containing handwritten notes for reference, and a (basic) calculator.
- Turn in your notes and any scratch paper with your exam.

Problems

Total, *(64 points.)*

Name: ID#:

True/False, *(12 points.)*

Here, assume that we have m data points $y^{(i)}, x^{(i)}$, $i = 1 \ldots m$, each with n features, $x^{(i)} = [x_1^{(i)} \ \ldots \ x_n^{(i)}]$. For each of the scenarios below, circle one of "true" or "false" to indicate whether you agree with the statement.

True or **false:** In a soft-margin SVM (i.e., loss $\sum_j w_j^2 + R \sum_i \epsilon^{(i)}$), increasing the value of R will make the model more likely to overfit.

True or **false:** A soft-margin SVM model is harder to optimize than a hard-margin SVM, since it is not a quadratic program.

True or **false:** A kernel SVM will be more efficient than a linear SVM when the number of training data, m, is large.

True or **false:** Applying "early stopping" by increasing the convergence tolerance in SGD increases the bias of the learner to reduce overfitting.

True or **false:** When training a perceptron using the logistic negative log-likelihood loss, gradient descent can never become stuck in a local optimum.

convex loss fn

True or **false:** Given sufficently many data m, the 1-nearest neighbor classifier error rate approaches the Bayes optimal error rate.

2x Bayes optimal

True or **false:** Stochastic gradient descent is often preferred over batch when the number of data points m is very large.

True or **false:** For a perceptron, increasing the regularization penalty of a linear regression model will decrease the resulting model's variance.

True or **false:** For a perceptron, doubling the number of training data available will decrease the resulting model's bias.

True or **false:** For a perceptron, using $2 \times n$ features per data point by adding n random values to each will increase the resulting model's variance.

True or **false:** With enough hidden nodes, a neural network can approximate any function.

True or **false:** Using backpropagation to train a neural network will avoid getting stuck in local optima.

Name: ID#:

Bayes Classifiers, *(10 points.)*

Consider the table of measured data given at right. We will use the two observed features x_1, x_2 to predict the class y. Each feature can take on one of three values, $x_i \in \{a, b, c\}$. In the case of a tie, we will prefer to predict class $y = 0$.

x_1	x_2	y
c	b	0
b	b	0
b	c	0
a	c	1
a	c	1
a	b	1
a	a	1
b	b	1
c	a	1

(1) Write down the probabilities learned by a naïve Bayes classifier: *(4 points.)*

$p(y=0)$: 1/3 $p(y=1)$: 2/3

$p(x_1=a \mid y=0)$: ∅ $p(x_1=a \mid y=1)$: 2/3

$p(x_1=b \mid y=0)$: 2/3 $p(x_1=b \mid y=1)$: 1/6

$p(x_1=c \mid y=0)$: 1/3 $p(x_1=c \mid y=1)$: 1/6

$p(x_2=a \mid y=0)$: ∅ $p(x_2=a \mid y=1)$: 1/3

$p(x_2=b \mid y=0)$: 2/3 $p(x_2=b \mid y=1)$: 1/3

$p(x_2=c \mid y=0)$: 1/3 $p(x_2=c \mid y=1)$: 1/3

(2) Using your naïve Bayes model, what value of y would you predict given $(x_1 = a, x_2 = b)$?: *(3 points.)*

$\hat{y} = 1$

$p(x = ab \mid y=0) = \emptyset$.

(3) Using your naïve Bayes model, compute the probabilities: *(3 points.)*

$p(y = 0 \mid x_1 = b, x_2 = c)$: $p(y = 1 \mid x_1 = b, x_2 = c)$: 1/3

$$= \frac{1/3 \cdot 2/3 \cdot 1/3}{1/3 \cdot 2/3 \cdot 1/3 + 2/3 \cdot 1/6 \cdot 1/3}$$

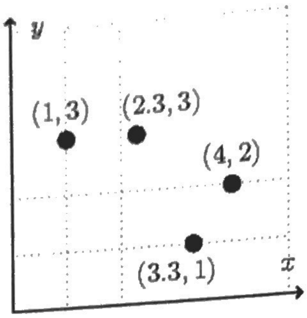
$$= \frac{2/9}{2/9 + 1/9} = \frac{2}{3}$$

Name: ID#:

Nearest Neighbor Regression, *(12 points.)*

For a regression problem to predict y given a scalar feature x, we observe training data (pictured at right):

	x	y
a	1	3
b	2.3	3
c	3.3	1
d	4	2



(1) Compute **training** MSE of a 1-nearest neighbor predictor. *(3 points.)*

ϕ

(2) Compute the **leave-one-out** cross-validation error (MSE) of a 1-nearest neighbor predictor. *(3 points.)*

	predict	error
a:	3	ϕ
b:	1	2
c:	2	1
d:	1	1

$MSE = \frac{1}{4} \cdot (2^2 + 1^2 + 1^2) = \frac{6}{4} = 1\frac{1}{2}$.

(3) Compute the **training** MSE of a 2-nearest neighbor predictor. *(3 points.)*

	predict	error
a:	3	ϕ
b:	2	1
c:	2.5	1.5
d:	1.5	.5

$MSE = \frac{1}{4}\left(1^2 + (\frac{3}{2})^2 + (\frac{1}{2})^2\right) = \frac{14}{16} = \frac{7}{8}$.

(4) Compute the **leave-one-out** cross-validation error (MSE) of a 2-nearest neighbor predictor. *(3 points.)*

	predict	error
a:	2	1
b:	2	1
c:	$2\frac{1}{2}$	$1\frac{1}{2}$
d:	2	ϕ

$MSE = \frac{1}{4}\left(1^2 + 1^2 + (\frac{3}{2})^2\right) = \frac{17}{16} = 1\frac{1}{16}$

Name: ID#:

Gradient Descent, *(10 points.)*

Suppose that we have training data $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$, where $x^{(i)}$ is a scalar feature with which we wish to predict the real-valued target $y^{(i)}$. We decide to use a nonlinear regression model with two parameters, $\theta = [a, b]$:

$$\hat{y}(x) = f(x\,;\theta) = a + \exp(x_1 + b)$$

and train our model using gradient descent on the mean squared error (MSE) loss.

(1) Write down the gradient of our loss function. $MSE = \frac{1}{m}\sum(y-\hat{y})^2 = J(a,b)$

$$\nabla J = \left[\frac{\partial J}{\partial a} \quad \frac{\partial J}{\partial b}\right]$$

$$\frac{\partial J}{\partial a} = \frac{1}{m}\sum\left(y - (a + \exp(x_1+b))\right)\cdot 1$$

$$\frac{\partial J}{\partial b} = \frac{1}{m}\sum\left(y - (a + \exp(x_1+b))\right)\cdot(-1)\exp(x_1+b)$$

(2) Give one advantage of batch gradient descent over stochastic gradient

Many: Easier to set step size; monotonic; easier to gauge convergence.

(3) Give one advantage of stochastic gradient descent over batch gradient

Mostly: faster for large m (more steps per epoch)

May avoid small local optima

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

Eg.:

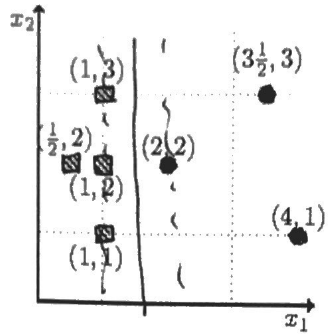
Init $\Theta = [a,b]$
Set convergence tolerance ϵ.
do {
  Compute ∇J
  Select step size α
  $\Theta \leftarrow \Theta - \alpha \nabla J$
} while ($\|\nabla J\| > \epsilon$)

Name: ____________ ID#: ____________

Support Vector Machines, *(10 points.)*

Suppose we are learning a linear support vector machine with two real-valued features x_1, x_2 and binary target $y \in \{-1, +1\}$. We observe training data (pictured at right):

x_1	x_2	y
0.5	2	+1
1	1	+1
1	2	+1
1	3	+1
2	2	-1
3.5	3	-1
4	1	-1



Our linear classifier takes the form

$$f(x; w_1, w_2, b) = \text{sign}(w_1 x_1 + w_2 x_2 + b).$$

(1) Consider the optimal linear SVM classifier for the data, i.e., the one that separates the data and has the largest margin. **Sketch** its decision boundary in the above figure, and **list** the support vectors here. *(2 points.)* SVs are : (1,3) (1,2) (1,1) and (2,2)

(2) Derive the parameter values w_1, w_2, b of this $f(x)$ using these support vectors. What is the length of the margin? *(3 points.)*

Have $b + 1w_1 + 2w_2 = -1$ and $b + 2w_1 + 2w_2 = +1$

$b + 1w_1 + \cancel{2}w_2 = -1$

$\Rightarrow w_1 = 2$

$\Rightarrow w_2 = \emptyset.$

$\Rightarrow b = -3$

(3) What is the *training error* of a linear SVM on these data? *(2 points.)*

$\emptyset$. (seperable)

(4) What is the the *leave-one-out cross validation error* for a linear SVM trained on these data? *(3 points.)*

1/7 - Decision boundary is the same if any point but (2,2) is removed.

Remove (2,2) ⇒ boundary shifts to (at least) 2.25 ⇒ error.

Name: [ ] ID#: [ ]

VC-Dimensionality, *(10 points.)*

Consider the VC dimension of two classifiers defined using two features x_1, x_2.

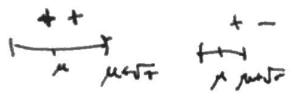
(1) First, consider a simple classifier f_A that uses only x_1 and predicts class +1 "close to" a point μ:

$$f_A(x) = \begin{cases} +1 & \text{if } (x_1 - \mu_1)^2 < r \\ -1 & \text{otherwise} \end{cases}$$

What is the VC dimension of f_A? Justify your answer. *(5 points.)*

2 - Decision function is a continuous interval from $\mu_1 - \sqrt{r}$ to $\mu_1 + \sqrt{r}$

Patterns



etc

But, cannot shatter 3: Pattern

No contiguous interval w/ both +1's inside.

(2) Now, suppose that we use both x_1 and x_2, but keep the number of parameters of the model the same by forcing $r = 1$, giving the classifier:

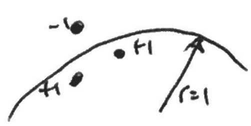
$$f_B(x) = \begin{cases} +1 & \text{if } ((x_1 - \mu_1)^2 + (x_2 - \mu_2)^2) < 1 \\ -1 & \text{otherwise} \end{cases}$$

What is the VC dimension of f_B? Justify your answer. *(5 points.)*

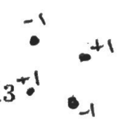
3 - Place points close together ($\|x^{(1)} - x^{(2)}\| \ll 1$).

Then

any one point is easy to separate from the others



But, 4 cannot:



similar to linear classifiers.