Midterm Recitation

10-601: Introduction to Machine Learning

09/27/2019

1 Quick Review

1. What is the difference between supervised learning and unsupervised learning?
2. What are the two major categories of supervised learning? What're their differences?
3. Which category of supervised learning does each of the following machine learning algorithms fall into?
   (a) Perceptron
   (b) linear regression
   (c) logistic regression
   (d) kNN
   (e) Decision tree
4. What kind of decision boundary can each of the classification algorithms generate?
5. What is regularization?
6. What are the main components of a supervised learning pipeline? (This is not covered in exams, but really good to know: data preprocessing, feature engineering, model selection/training/evaluation, error analysis) Think about the theory/models/tricks/methods you've learned so far in this course and which component(s) they apply to.

2 kNN Recitation Examples

1. **Select all that apply:** Please select all that apply about kNN in the following options:

   Assume a point can be its own neighbor.

   ◯ k-NN works great with a small amount of data, but struggles when the amount of data becomes large.

   ◯ k-NN is sensitive to outliers; therefore, in general we decrease k to avoid overfitting.

   ◯ k-NN can only be applied to classification problems, but it cannot be used to solve regression problems.

   ◯ We can always achieve zero training error (perfect classification) with k-NN, but it may not generalize well in testing.

3 Decision Tree

1. Given three binary input features, can a decision tree make 0 training and test error on a 9-class classification problem?
2. ID3 algorithm is a greedy algorithm for growing Decision Tree and it suffers the same problem as any other greedy algorithm that finds only locally optimal trees. Which of the following method(s) can make ID3 "less greedy"? **Select all that apply:**

   ☐ Use a subset of attributes to grow the decision tree

   ☐ Use different subsets of attributes to grow many decision trees

   ☐ Change the criterion for selecting attributes from information gain (mutual information) to information gain ratio (mutual information divided by entropy of splitting attributes) to avoid selecting attributes with high degree of randomness

   ☐ Keep using mutual information, but select 2 attributes instead of one at each step, and grow two separate subtrees. If there are more than 2 subtrees in total, keep only the top 2 with the best performance (e.g., top 2 with lowest training errors at the current step)

4 Perceptron

1. Why does the "the intercept term" (the bias term) not correspond to the actual intercept where the linear boundary crosses through in Perceptron? Why is it different from the intercept defined in linear regression?
2. What is the mistake bound? If you have a larger mistake bound, will you make more mistakes when running perceptron online? (Hint: HW3 Recitation has a proof for the mistake bound and it sure will make it more understandable)

3. ***Perceptron Trees:** To exploit the desirable properties of decision tree classifiers and perceptrons, Adam came up with a new algorithm called "perceptron trees", which combines features from both. Perceptron trees are similar to decision trees, however each leaf node is a perceptron, instead of a majority vote.

   To create a perceptron tree, the first step is to follow a regular decision tree learning algorithm (such as ID3) and perform splitting on attributes until the specified maximum depth is reached. Once maximum depth has been reached, at each leaf node, a perceptron is trained on the remaining attributes which have not been used up in that branch. Classification of a new example is done via a similar procedure. The example is first passed through the decision tree based on its attribute values. When it reaches a leaf node, the final prediction is made by running the corresponding perceptron at that node.

   Assume that you have a dataset with 6 binary attributes **(A, B, C, D, E, F)** and two output labels **(-1 and 1)**. A perceptron tree of depth 2 on this dataset is given below. Weights of the perceptron are given in the leaf nodes. Assume bias=1 for each perceptron:

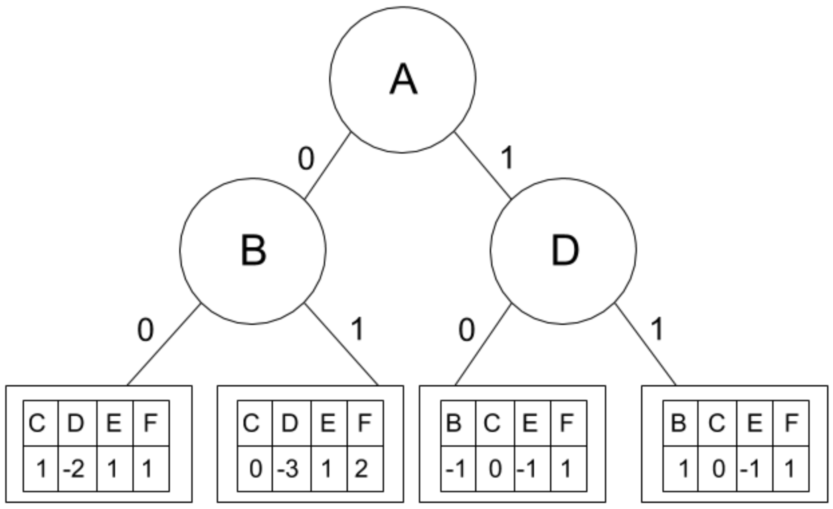

Figure 1: Perceptron Tree of max depth=2

(a) **Numerical answer:** Given a sample $\mathbf{x} = [1, 1, 0, 1, 0, 1]$, predict the output label for this sample

(b) **True or False:** The decision boundary of a perceptron tree will *always* be linear.

○ True

○ False

(c) **True or False:** For small values of max depth, decision trees are *more* likely to underfit the data than perceptron trees

○ True

○ False

5 Linear Regression

1. (1 point) **Select one:** The closed form solution for linear regression is $\theta = (X^T X)^{-1} X^T y$. Suppose you have n = 35 training examples and m = 5 features (excluding the intercept term). Once the intercept term is now included, what are the dimensions of X, y, θ in the closed form equation?

   ○ X is 35×6, y is 35×1, θ is 6×1

   ○ X is 35×6, y is 35×6, θ is 6×6

   ○ X is 35×5, y is 35×1, θ is 5×1

   ○ X is 35×5, y is 35×5, θ is 5×5

2. (2 points) **Select all that apply:** You are given a variable z to predict with 2 co-variates x_1 and x_2. Which of the following equations relating x_1 and x_2 **could** lead to a closed-form solution $\boldsymbol{\theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ if you run a linear-regression of x_1 and x_2 on z : $z = \beta_1 x_1 + \beta_2 x_2$?

   Note: The i^{th} row of $\mathbf{X}$ contains the i^{th} data point $(x_{i,1}, x_{i,2})$ while the i^{th} row of $\mathbf{y}$ contains the i^{th} data point y_i. Ignore the intercept term in $\mathbf{X}$.

   □ $x_2 = 2x_1 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$

   □ $x_2 = 3x_1$

   □ $x_2 = 2x_1^2 + x_1$

   □ $x_2 = x_1^3$

   □ $x_2 = \sin x_1$

   □ None of the Above

3. *Consider linear regression on 1-dimensional data $\mathbf{x} \in \mathbb{R}^n$ with label $\mathbf{y} \in \mathbb{R}^n$. We apply linear regression in both directions on this data, i.e., we first fit y with x and get $y = \beta_1 x$ as the fitted line, then we fit x with y and get $x = \beta_2 y$ as the fitted line. Discuss the relations between β_1 and β_2:

   (i) **True or False:** The two fitted lines are always the same, i.e. we always have $\beta_2 = \frac{1}{\beta_1}$.

   □ True

☐ False

(ii) **Numerical answer:** We further assume that $\mathbf{x}^T\mathbf{y} > 0$. What is the minimum value of $\frac{1}{\beta_1} + \frac{1}{\beta_2}$?

6 Regularization & Optimization

1. (3 points) **Derivation.** We introduce a modified method of linear regression where we now put different importance on each feature.

   We introduce a new $p \times p$ diagonal matrix: Q:

$$\begin{pmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & q_p \end{pmatrix}$$

   Each diagonal entry q_i is a quantified "importance" that we give each of the p features.

   The new objective function is given as:

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \boldsymbol{\theta}^T Q x^{(i)})^2$$

   ,where $\boldsymbol{\theta} \in \mathbb{R}^k$

   Derive the update rule for this new modified regression.

2. **Time complexity:** Given N samples of M features in a linear regression problem, what is the time complexity of one parameter update using GD? What about using SGD? What about closed form solution?

3. (1 point) **Select one:** Which of the following is true about the regularization parameter λ (the parameter that controls the extent of regularization):

   ◯ Larger values of λ can overfit the data.

   ◯ Larger λ does not affect the performance of your hypothesis

◯ Adding a regularization term to a classifier, ($\lambda \neq 0$), may cause some training examples to be classified incorrectly.

4. **Select all that apply:** Which of the following are correct regarding Gradient Descent (GD) and stochastic gradient descent (SGD)

    - ☐ Each update step in SGD pushes the parameter vector closer to the parameter vector that minimizes the objective function.
    - ☐ The gradient computed in SGD is, in expectation, equal to the gradient computed in GD.
    - ☐ The gradient computed in GD has a higher variance than that computed in SGD, which is why in practice SGD converges faster in time than GD.

7 Summary

KNN:
pros:
1. simple and intuitive
2. no training step
3. apply to multi-class problems and can use different metrics
cons:
1. become slow as dataset grows
2. require homogeneous features
3. selection of K
4. not good at handling imbalanced data
5. sensitive to outliers
when to use:
small dataset, small dimensionality, data is clean (missing data), classification
inductive bias:
Similar (i.e. nearby) points should have similar labels. All label dimensions are created equal.

Perceptron:
pros:
1. one of the simplest ML model
2. perform to correct mistakes one by one
cons:
1. only converge on linearly separable data
2. decision boundary is linear
3. decision boundary is not guaranteed to be optimal
when to use:
linearly separable dataset with clear boundaries not commonly used in real life but expandable to multi-layer perceptron.
inductive bias:
(perceptron) decision boundary should be linear 2. (online) prefer to correct most recent mistakes

Linear Regression:
pros:
1. easy to understand and train
2. works for most cases
cons:
1. assume that the relations of the dependent and independent variables are linear
2. sensitive to noises
when to use:
most regression cases
inductive bias:
The relationship between the inputs x and output y is linear. i.e. Hypothesis space is Linear Functions

Decision Tree:
pros:
1. easy to understand and interpret
2. non-parametric model
3. very fast for inference
cons:
1. tree may grow very large and tends to overfit.
when to use:
Most cases. (Need to prepocess the continuous data first.)
inductive bias:
In ID3, the Inductive bias is that ID3 Searches for the smallest tree consistent w/ the training data"(i.e. 0 error rate) (which is an appilcation of occam's razer).

Logistic Regression [NOT COVERED IN MIDTERM 1]:
pros:
1. nice, intuitive probabilistic interpretation
2. can be adapted to deal with multi-class classification
3. SGD trainable
cons:
1. assume a linear model of dependency of probabilities on features
2. sensitive to feature scaling
when to use:
Most classification cases.
inductive bias:
For binary LR, the inductive bias is that the log odds between two classes is a linear function of input features.