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.

   True: A, Curse of dimensionality; D, by setting k = 1

   False: B, we increase k to avoid overfitting; C, KNN regression

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?

   No, $2^3 = 8 < 9$

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)

2nd and 4th choices; 1st choice should be wrong as the best performance will be determined by the deepest tree. Any shallower tree will make more mistakes, so ensemble learning can only make performance worse and it won't change the local optimality of the forest.

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?

   In perceptron, suppose we have 2d features. When we draw $w^T x + b$ on a 2d grid with two axes, x_1 and x_2, there is no y axis there. The intercept we see on the vertical axis is the intercept with respect to x_2. That means we need to write a function with x_2 alone on the left hand side and everything else on the right hand side. Only then will the constant term on the right hand side represent the intercept on the graph, and it's different from b. In this case, the linear boundary is

   $$w^T x + b = 0$$
   $$w_1 x_1 + w_2 x_2 + b = 0$$
   $$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

   $[0, -\frac{b}{w_2}]$ is the point you see on the graph that crosses through the x_2 axis.

   In linear regression, the two axes you have are x and y and we are fitting $y = w^T x + b$, where y is already on the left hand side alone, which is why b becomes the actual intercept on the graph.

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)

   The mistake bound is saying the number of mistakes is $\leq (R/\gamma)^2$. R is the distance from the farthest point to the ORIGIN! The origin is where the zero vector lies in the coordinates. γ is the distance between the closest point to the separating hyperplane that gives the largest margin.

   If the dataset have a larger mistake bound, it does not mean that we will make more mistakes. The mistake bound is just what we use to generally understand how hard the problem is using perceptron on a certain dataset.

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:

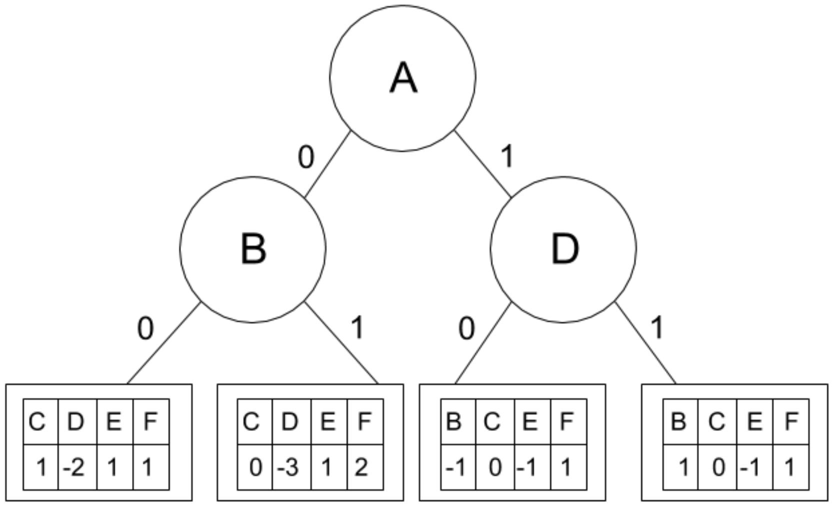

Figure 1: Perceptron Tree of max depth=2

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

1, Explanation: A=1 and D=1 so the point is sent to the right-most leaf node, where the perceptron output is (1*1)+(0*0)+((-1)*0)+(1*1)+1 = 3. Prediction = sign(3) = 1.

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

◯ True

◯ False

False, since decision tree boundaries need not be linear.

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

◯ True

◯ False

True. For smaller values of max depth, decision trees essentially degenerate into majority-vote classifiers at the leaves. On the other hand, perceptron trees have the capacity to make use of "unused" attributes at the leaves to predict the correct class. Decision trees: Non-linear decision boundaries
Perceptron: Ability to gracefully handle unseen attribute values in training data/ Better generalization at leaf nodes

5 Linear Regression

1. (1 point) **Select one:** The closed form solution for linear regression is $\theta = (X^TX)^{-1}X^Ty$. 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

   A.

2. (2 points) **Select all that apply:** You are given a variable z to predict with 2 covariates 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

   (a), (c), (d), (e). Only in (b) are we guaranteed that the columns corresponding to x_1 and x_2 in the X matrix are linearly dependent. In (c), (d) and (e), the columns corresponding to x_1 and x_2 in the X matrix are not linearly independent. Additionally, in (a), due to the ϵ, x_1 and x_2 may not be linearly dependent.

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

False.

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

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 $\boldsymbol{\theta} \in \mathbb{R}^p$

Derive the update rule for this new modified regression.

We start by deriving partial derivative formula for one component of θ, namely θ_1 and then generalize it to all components (vector form).

$$\frac{dJ(\theta)}{d\theta_1} = \frac{\partial}{\partial\theta_1}\frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \sum_{j=1}^{p}\theta_j q_j x_j^{(i)})^2$$

where $x_j^{(i)}$ is the jth feature value of ith sample, or the (i,j) element in the design matrix X $(m \times n)$. The derivation above carries out like,

$$\begin{aligned}\frac{dJ(\theta)}{d\theta_1} &= \sum_{i=1}^{N}(y^{(i)} - \sum_{j=1}^{p}\theta_j q_j x_j^{(i)})\frac{\partial}{\partial\theta_1}(-\sum_{j=1}^{p}\theta_j q_j x_j^{(i)}) \\ &= \sum_{i=1}^{N}(y^{(i)} - \sum_{j=1}^{p}\theta_j q_j x_j^{(i)})(-q_1 x_1^{(i)}) \\ &= -\sum_{i=1}^{N} q_1 x_1^{(i)} y^{(i)} + \sum_{i=1}^{N}\sum_{j=1}^{p} q_1 x_1^{(i)} x_j^{(i)} q_j \theta_j\end{aligned}$$

The key observation here is to identify the first term as the 1st element of the vector $-QX^TY$, and the second term is the first element of vector $QX^TXQ\theta$. We'll show the first term correspondence here, second term correspondence can be derived similarly, just needs more math.

$$\begin{aligned}(-QX^TY)_{(1,1)} &= -\sum_{j=1}^{p}\sum_{i=1}^{N} Q_{(1,j)}(X^T)_{(j,i)}Y_{(i,1)} \\ &= -\sum_{i=1}^{N} q_1 (X^T)_{(1,i)} y^{(i)} \\ &= -\sum_{i=1}^{N} q_1 x_1^{(i)} y^{(i)}\end{aligned}$$

In the derivation above, the subscript (x, y) notation is specifying the position of an element in a matrix, so $X^T_{(j,i)}$ means the jth row, ith column element in matrix X^T, or ith row, jth column element of matrix X, which is just $x_j^{(i)}$. Y is the $N \times 1$ vector formed by stacking all sample $y^{(i)}$s. And the equality on the second line comes from the diagonal shape of Q.

Given the general update rule (α is the learning rate):

$$\theta^{(k+1)} \leftarrow \theta^{(k)} + \alpha\frac{dJ(\theta)}{d\theta}$$

, the new update rule should be:

$$\theta^{(k+1)} \leftarrow \theta^{(k)} + \alpha Q X^T (Y - XQ\theta)$$

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?

   For GD, time complexity involves calculating $\mathbf{Xw} - \mathbf{y}$, $X \in \mathbb{R}^{N \times M}$, $w, y \in \mathbb{R}^{m \times 1}$, so it's $O(NM)$; For SGD, single update requires only $O(M)$; For closed form solution $w = (X^T X)^{-1} X^T y$, the time complexity will be $O(M^3 + M^2 N)$, first term from matrix inverse (best we can do for now is $O(M^{2.373})$), second from matrix multiplication $X^T X$.

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.

   C

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.

   B.

   A is incorrect, SGD updates are high in variance and may not go in the direction of the true gradient. C is incorrect, for the same reason. D is incorrect since they can converge if the function is convex, not just strongly convex.

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 (no 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.