Recitation 3
Decision Tree, KNN, Perceptron, Linear Regression

10-601: Introduction to Machine Learning

02/08/2019

1 kNN Recitation Examples

1.1 A Classification Example – Warm Up

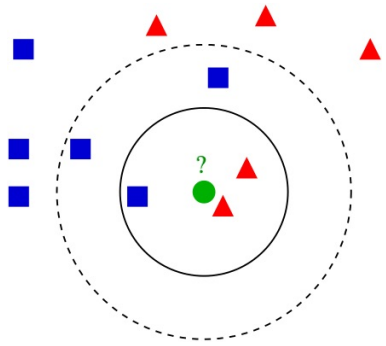

Figure 1: From wiki

1.2 kNN for Regression

Name	Quiz 1	Quiz 2	Final
Alice	12	15	80
Bob	10	6	65
Cathy	5	12	52
David	3	4	28
Emily	1	15	48
Ford	20	25	90
George	25	25	100
Zack	2	4	**???**

Question: Can k-NN be used for regression? E.g. can it predict the final score of Zack? How?

DISTANCE METRICS

sub-question 1.2.1 If k=2, and we use Euclidean Distance as the distance metric, what are the k-nearest-neighbors?

sub-question 1.2.2 If k=2, and we use Manhattan Distance as the distance metric, what are the k-nearest-neighbors? (Manhattan Distance is: the sum of the horizontal and vertical distances between points on a grid.)

1.3 Take home:

1.3.1 kNN for Classification – Simplified Recipe

1. If we have collected the training data, what is the training process?

2. Do we need to design anything before the training process(any metric, or hyper-parameters)? What is the hyper-parameters of kNN?

3. How do we choose hyper-parameters? Can we use test set to find the BEST one?

4. For the testing/predicting part, we shall we do? Say we have now a new data point, what should we do?

1.3.2 REGRESSION METHODS

Let's pick: k=2, Manhattan Distance (so we know who the kNN's are). Now, could you please find some regression methods?

1. Could Majority Vote works here? If so, please conduct regression and write down the answer.

2. Could Taking Average works here? If so, please conduct regression and write down the answer.

3. Could Taking Weighted Average works here? If so, please conduct regression and write down the answer.

4. What kinds of weighting function could we have?

5. Do you think these averaging methods good methods? See our prediction result, the final score of Zach.

6. Could Linear Regression works here? If so, please conduct regression and write down the answer. Let's assume k=5, and use Manhattan Distance, here. (BTW, do you think it is strange to combine these 2 topics? Because it seems to be two separate topics (kNN vs. Linear Regression) when we are learning, right?)

7. Could you please find the position of Zach in the following figures?

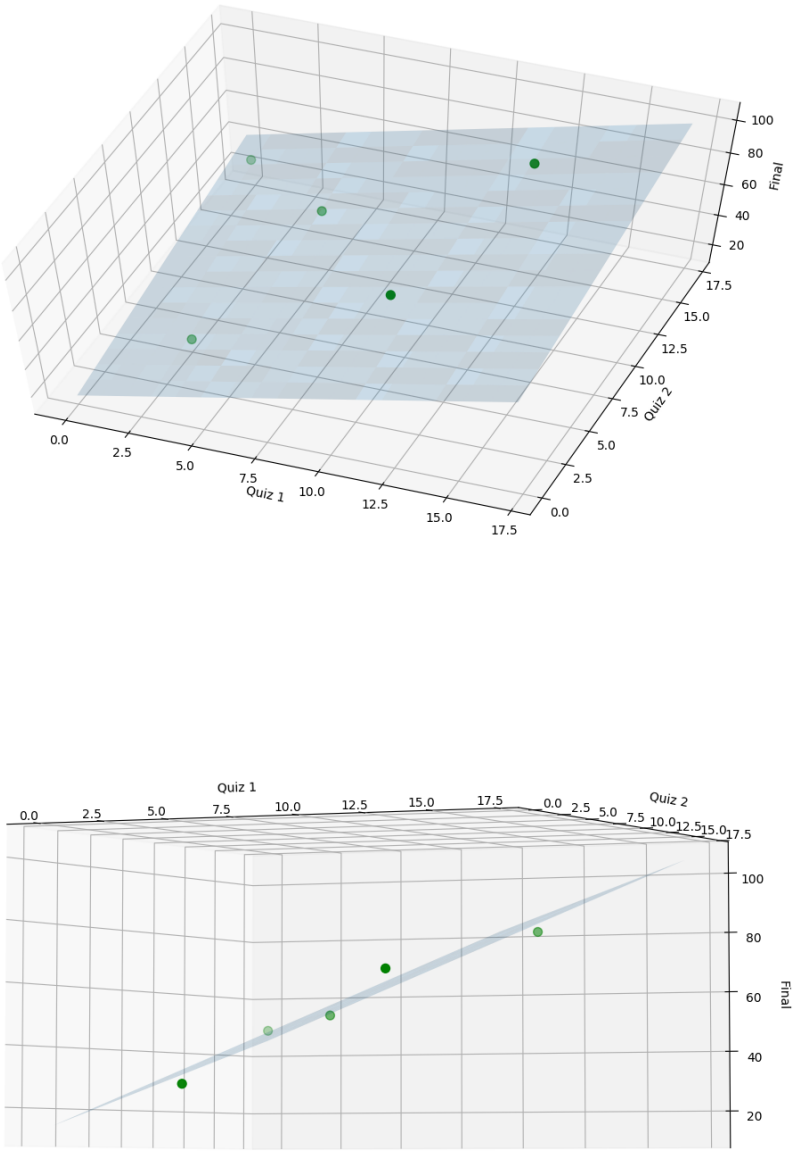

Figure 2: Figures to show kNN Regression using Linear Regression.

1.3.3 Remarks.

1. Will kNN suffers the Curse of Dimension? Could you please give an example to illustrate this? If so, how could we deal with it?
2. When k=1, what will the decision boundary of kNN (1-NN, here) look like? Could you please search "Voronoi Diagram" online?
3. Do you know "Locally Weighted Regression"? What is the relation between this with kNN Regression?

2 Perceptron

Algorithm 1 Perceptron

Initialize $\vec{w} = \vec{0}, b = 0$
for $i = 1, 2, 3...$:
    Receive instance $\vec{x}^{(i)} \in \mathbb{R}^m$
    Predict $\vec{y} = sign(\vec{w}^T\vec{x} + b)$ where

$$sign(a) = \begin{cases} 1, & if\ a \geq 0, \\ -1, & otherwise. \end{cases} \tag{1}$$

    Receive label $y^{(i)}$
    if positive mistake ($y^{(i)} = 1$ and $y^{(i)} \neq \vec{y}$):
        $\vec{w} \leftarrow \vec{w} + \vec{x}^{(i)}$
        $b \leftarrow b + 1$
    else if negative mistake ($y^{(i)} = -1$ and $y^{(i)} \neq \vec{y}$):
        $\vec{w} \leftarrow \vec{w} - \vec{x}^{(i)}$
        $b \leftarrow b - 1$

2.1 Recitation example:

1. For the following sequence of points presented to the perceptron learning algorithm, complete the table below.

b	w_1	w_2	x_1	x_2	y	$\hat{y}$
0	0	0	1	0	-1	?
?	?	?	0	0	-1	?
?	?	?	1	1	$+1$	?
?	?	?	0	1	$+1$	?

2. At the end of the fourth row of the table, does the perceptron correctly classify all of the points shown? Will the perceptron algorithm eventually converge?

2.2 Take home:

1. Given dataset: $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$, suppose
    (a) Finite-size input, i.e., $\|\mathbf{x}^{(i)}\| \leq R$, and
    (b) Linearly separable data, i.e., $\exists \boldsymbol{\theta}^*$ such that $\|\boldsymbol{\theta}^*\| = 1$ and for every $1 \leq i \leq N$, $y^{(i)}(\boldsymbol{\theta}^*\mathbf{x}^{(i)}) \geq \gamma$.

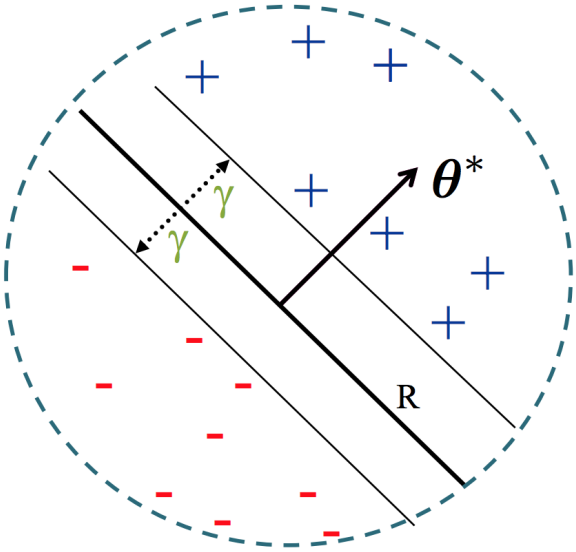

Figure 3: mistake bound for perceptron

Then, show that the number of mistakes k made by the Perceptron algorithm on this dataset is bounded by $k \leq \frac{R^2}{\gamma^2}$.

Solution:

Define initial weight vector $\theta^{(0)} = \mathbf{0}$. Let $\theta^{(k)}$ be the weight vector after making k mistakes.

First, we show that $\|\theta^{(k)}\| \geq k\gamma$.

$$\theta^{(k)} \cdot \theta^* = (\theta^{(k-1)} + y^{(i)}\mathbf{x}^{(i)}) \cdot \theta^* \tag{2}$$
$$= \theta^{(k-1)} \cdot \theta^* + y^{(i)}\mathbf{x}^{(i)} \cdot \theta^* \tag{3}$$
$$\geq \theta^{(k-1)} \cdot \theta^* + \gamma \tag{4}$$

By repeatedly applying the above rule for $\theta^{(j)}$ for $j \in [0, k-1]$, this implies

$$\theta^{(k)} \cdot \theta^* \geq \theta^{(0)} \cdot \theta^* + k\gamma \tag{5}$$
$$\theta^{(k)} \cdot \theta^* \geq k\gamma \qquad \theta^{(0)} = \mathbf{0} \tag{6}$$
$$\|\theta^{(k)}\|\|\theta^*\| \geq k\gamma \qquad \text{Cauchy Schwarz Inequality} \tag{7}$$
$$\|\theta^{(k)}\| \geq k\gamma \qquad \|\theta^*\| = 1 \tag{8}$$

Next, we show that $\|\theta^{(k)}\| \leq R\sqrt{k}$.

$$\|\theta^{(k)}\|^2 = \|\theta^{(k-1)} + y^{(i)}\mathbf{x}^{(i)}\|^2 \tag{9}$$
$$= (\theta^{(k-1)} + y^{(i)}\mathbf{x}^{(i)})^T(\theta^{(k-1)} + y^{(i)}\mathbf{x}^{(i)}) \tag{10}$$
$$= \|\theta^{(k-1)}\|^2 + (y^{(i)})^2\|\mathbf{x}^{(i)}\|^2 + 2y^{(i)}\theta^{(k-1)} \cdot \mathbf{x}^{(i)} \tag{11}$$
$$\leq \|\theta^{(k-1)}\|^2 + (y^{(i)})^2\|\mathbf{x}^{(i)}\|^2 \qquad y^{(i)}\theta^{(k-1)} \cdot \mathbf{x}^{(i)} < 0\text{; mistake was made} \tag{12}$$
$$\leq \|\theta^{(k-1)}\|^2 + R^2 \qquad \text{definition of } R \text{ and } (y^{(i)})^2 = 1 \tag{13}$$

Again, combining these for $\theta^{(j)}$ for $j \in [0, k-1]$, this implies

$$\|\theta^{(k)}\|^2 \leq kR^2 \tag{14}$$
$$\|\theta^{(k)}\| \leq R\sqrt{k} \tag{15}$$

Combining the above two results, we get

$R\sqrt{k} \geq \|\theta^{(k)}\| \geq k\gamma \implies k \leq (\frac{R}{\gamma})^2$

2. Lets suppose we have M weather models $\{f_i\}_{i=1}^M$, each of which predicts if it is going to rain the next day. Each model outputs either -1 or $+1$, where -1 corresponds to no rain and $+1$ corresponds to rain. We do not know anything about these models; we only know that there are K models whose majority vote always gives the correct prediction (K is an odd number). However, we dont know the identity of these K models.

   Our goal is to learn the best possible way to combine the outputs of these M models to reliably predict rain. To be more formal, we would like to learn a function $f : \{-1, +1\}^M \to \{-1, +1\}$, which takes the predictions of the M models as inputs and reliably predicts whether it is going to rain the next day.

   Suppose we use the Perceptron algorithm to learn a linear decision boundary f. Each day, we make a prediction based on the current f and update f the next day if our prediction was wrong. How many mistakes will you make at most using this approach?

3 Linear Regression

Main Idea: The 2-Step Process:

1. Define Objective Function: $J(\theta)$
   - What does this function do ?

- What are some properties of this function?
- What are some examples?

2. Work Towards the Min/Max: 2 Methods

(i) **Closed-form**

Example: We are given the following data where x is the input and y is the output:

x	1.0	2.0	3.0	4.0	5.0
y	2.0	4.0	7.0	8.0	11.0

Based on our inductive bias, we think that the linear hypothesis with no intercept should be used here. We also want to use the Mean Squared Error as our objective function: $\frac{1}{5}\sum_{i=1}^{5}(y^{(i)}-wx^{(i)})^2$, where $y^{(i)}$ is our i^{th} data point and w is our weight. Using the closed-form method, find w.

- What is the closed-form formula for w?
- What is the value of w?

We now extend the data set to include more features (x_1, x_2, x_3):

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$
x_1	1.0	2.0	3.0	4.0	5.0
x_2	-2.0	-5.0	-6.0	-8.0	-11.0
x_3	3.0	8.0	9.0	12.0	14.0
y	2.0	4.0	7.0	8.0	11.0

We again think that the linear hypothesis with no bias should be used here. We also want to use the Mean Squared Error as our objective function:

$$\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w}^T x^{(i)})^2$$

, where $\mathbf{w} = [w_1, w_2, w_3]^T$, $x^{(i)}$ is the i^{th} datapoint and $y^{(i)}$ is the i^{th} $y-$value.

- What is the closed-form formula for w_1?
- What is the closed-form solution for $\mathbf{w}$?

(ii) **Gradient Descent**

We use the same data set from above. However, we want to implement the gradient descent method.

Assuming that $\lambda = 0.1$ and w has been initialized to $[0, 0, 0]^T$, perform one iteration of gradient descent:

i. What is the gradient of the objective function , $J(\mathbf{w})$, w.r.t $\mathbf{w}$: $\nabla_{\mathbf{w}} J(\mathbf{w})$

ii. How do we carry out the update rule?

Take Home:

4 Decision Tree and beyond

1. **Feature selection - More data doesn't equal easier life.** See following table: we have 4 input features (x_1, x_2, x_3, x_4) and one target variable (y), all of which are discrete. If we are to train a decision tree model on this dataset, using ID3 algorithm with mutual information as node-splitting criterion, which feature would you select first?

x_1	x_2	x_3	x_4	y
0	0	0	1	-
0	1	1	2	+
0	1	0	3	+
0	0	1	4	-
1	0	0	5	+
1	0	1	6	-

OK...if you are to grow tree using the feature you just selected, what would the tree be like?
Is this a good decision tree? What do you think the problem is?
Can we use the tricks taught in class to prevent overfitting (pruning, setting maximum depth, setting lower bounds on criterion, etc.) to fix this?

2. **Feature engineering - Don't be greedy.** Continue our exploration with the decision tree algorithm. Now, suppose we have a data table like

x_1	x_2	x_3	x_4'	y
0	0	0	1	-
0	0	1	1	+
0	1	0	1	+
0	1	1	0	-
1	0	0	1	+
1	0	1	0	-
1	1	0	0	-
1	1	1	1	+

Using mutual information as criterion, and following ID3 algorithm, which feature would you choose as the first node to split?
What is the best error rate you can achieve on this dataset with a decision tree of depth 3 (break ties by choosing feature with smaller index)?
Is there a problem with this approach? Can you do better?

3. **Feature engineering - A good model isn't everything.** Given a two-dimensional dataset shown in Fig.4, where red circles are positive examples and blue circles are negative examples. Which of the following model would you choose to classify the samples using the two coordinates as features?

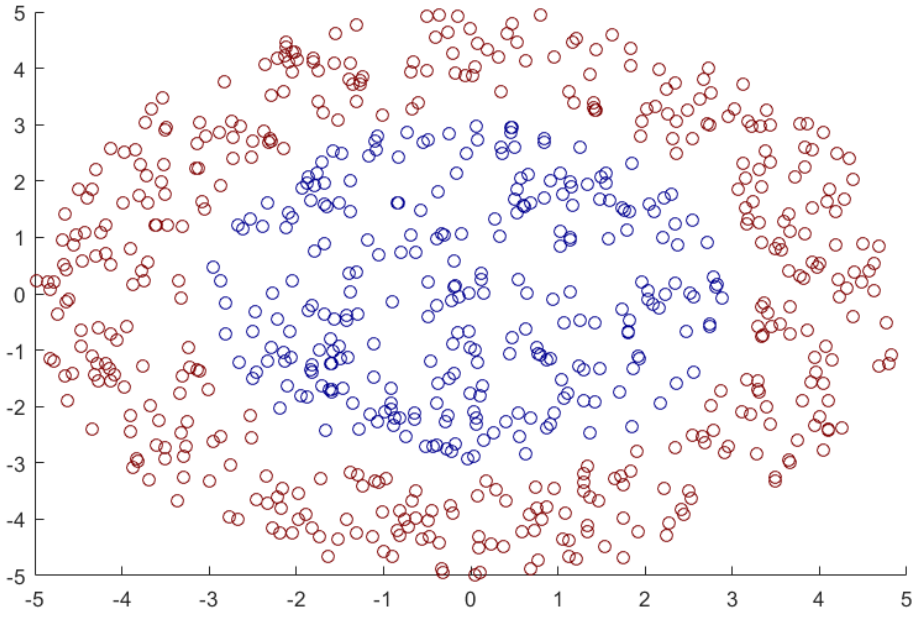

Figure 4

Which model would you choose for this classification task? (There is no best answer here)

☐ Linear regression on x and y

☐ Decision tree using attributes x and y trained with ID3 algorithm

☐ kNN with $k = 5$ (any distance metric)

☐ A single Perceptron

☐ Ensemble of 1000 Perceptrons

☐ Deep neural network

4. **Entropy and mutual information.** Given a discrete random variable X that can take n values, prove that the maximum entropy the variable can attain is $\log_2(n)$, i.e., $H(X) \leq \log_2(n)$. When does the equality hold?

   What discrete random variables have 0 entropy?

   Given the probabilistic definition of mutual information (capital letters X, Y refer to the random variables, small letters x, y refer to the actual values those variables take. $\mathcal{X}, \mathcal{Y}$ denote the set of values, or domains, of those variables)

$$MI(X;Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log(\frac{p(x)p(y)}{p(x,y)})$$

   Fill in the boxes below with the appropriate relational symbols ($>, <, =, \geq, \leq$ or ? for no definitive relationship). $H(X)$ denotes the entropy of a random variable X.

$$MI(X;Y) \square\ MI(Y;X)$$
$$MI(X;Y) \square\ 0$$
$$H(X) \square\ H(X|Y)$$
$$H(X) \square\ H(X|Y=y)$$

5 Summary

KNN:
pros:
1. simple and no assumption
2. no training step
3. apply to multi-class problems and use different metric

cons:
1. become slow as dataset grows
2. require homogeneous features
3. selection of K
4. 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 optimal
when to use:
linearly separable dataset with clear boundaries not commonly used in real life but consider 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 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 occams razer).