

# RECITATION 3

## CLASSIFICATION AND REGRESSION

10-301/10-601: INTRODUCTION TO MACHINE LEARNING

02/09/2024

### 1 Decision Trees and Beyond

#### 1. Decision Tree Classification with Continuous Attributes

Given the dataset  $\mathcal{D}_1 = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$  where  $\mathbf{x}^{(i)} \in \mathbb{R}^2, y^{(i)} \in \{\text{Yellow, Purple, Green}\}$  as shown in Fig. 1, we wish to learn a decision tree for classifying such points. Provided with a possible tree structure in Fig. 1, what values of  $\alpha, \beta$  and leaf node predictions could we use to perfectly classify the points? Now, draw the associated decision boundaries on the scatter plot.

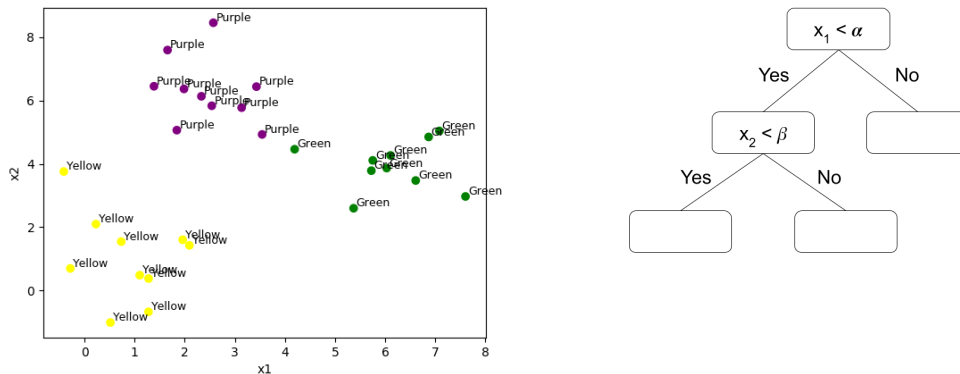


Figure 1: Classification of 2D points, with Decision Tree to fill in

## 2. Decision Tree Regression with Continuous Attributes

Now instead if we had dataset  $\mathcal{D}_2 = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$  where  $\mathbf{x}^{(i)} \in \mathbb{R}^2, y^{(i)} \in \mathbb{R}$  as shown in Fig. 2, we wish to learn a decision tree for regression on such points. Using the same tree structure and values of  $\alpha, \beta$  as before, what values should each leaf node predict to minimize the training Mean Squared Error (MSE) of our regression? Assume each leaf node just predicts a constant.

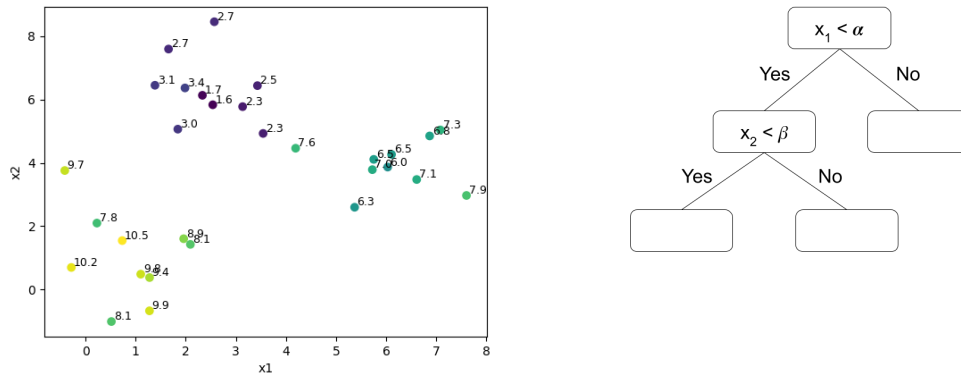
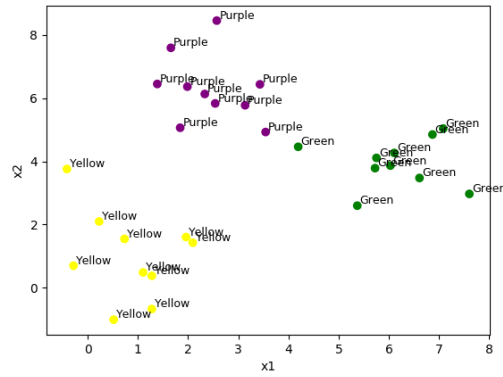


Figure 2: Regression on 2D points, with Decision Tree to fill in

3. **Choosing a Tree:** What might happen if we increased the max-depth of the tree? When predicting on unseen data, would we prefer the depth-2 tree above or a very deep tree?



## 2 Perceptron

### 2.1 Perceptron Mistake Bound Guarantee

If a dataset has margin  $\gamma$  and all points inside a ball of radius  $R$ , then the perceptron makes less than or equal to  $(R/\gamma)^2$  mistakes.

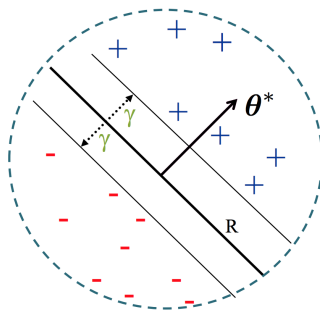


Figure 3: Perceptron Mistake Bound Setup

### 2.2 Definitions

**Margin:**

- The margin of example  $x$  wrt a linear separator  $w$  is the (absolute) distance from  $x$  to the plane  $w \cdot x = 0$ .
- The margin  $\gamma_w$  of a set of examples  $S$  wrt a linear separator  $w$  is the smallest margin over points  $x \in S$ .
- The margin  $\gamma$  of a set of examples  $S$  is the maximum  $\gamma_w$  over all linear separators  $w$ .

**Linear Separability:** For a binary classification problem, a set of examples  $S$  is linearly separable if there exists a linear decision boundary that can separate the points.

**Update Rule:** When the  $k$ -th mistake is made on data point  $\mathbf{x}^{(i)}$ , the parameter update is

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{y}^{(i)} \mathbf{x}^{(i)}$$

We say the (batch) perceptron algorithm has *converged* when it stops making mistakes on the training data.

### 2.3 Perceptron Mistake Bound: Example

Given dataset  $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ , suppose:

1. Finite size inputs:  $\|x^{(i)}\| \leq R$
2. Linearly separable data:  $\exists \theta^*$  and  $\gamma > 0$  s.t.  $\|\theta^*\| = 1$  and  $y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i$

Then, the number of mistakes  $k$  made by the perceptron algorithm on  $\mathcal{D}$  is bounded by  $(R/\gamma)^2$ .

The following table shows a dataset of linearly separable datapoints.

x1	x2	y
1	-1	1
0	2	-1
4	0	1

Assuming that the linear separator with the largest margin is given by:

$$\theta^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \text{ where } \theta = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Calculate the theoretical mistake bound for the perceptron.

## 2.4 Theorem: Block, Novikoff

Given dataset  $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ , suppose:

1. Finite size inputs:  $\|x^{(i)}\| \leq R$
2. Linearly separable data:  $\exists \theta^*$  and  $\gamma > 0$  s.t.  $\|\theta^*\| = 1$  and  $y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i$

Then, the number of mistakes  $k$  made by the perceptron algorithm on  $\mathcal{D}$  is bounded by  $(R/\gamma)^2$ .

### Proof:

Part 1: For some  $A$ ,  $Ak \leq \|\theta^{(k+1)}\|$

$$\begin{aligned} \theta^{(k+1)} \cdot \theta^* &= (\theta^{(k)} + y^{(i)}x^{(i)}) \cdot \theta^*, \text{ Perceptron algorithm update} \\ &= \theta^{(k)} \cdot \theta^* + y^{(i)}(\theta^* \cdot x^{(i)}) \\ &\geq \theta^{(k)} \cdot \theta^* + \gamma, \text{ by assumption} \\ \implies \theta^{(k+1)} \cdot \theta^* &\geq k\gamma, \text{ by induction on } k \text{ since } \theta^{(1)} = 0 \\ \implies \|\theta^{(k+1)}\| &\geq k\gamma, \text{ since } \|w\| \times \|u\| \geq w \cdot u \text{ and } \|\theta^*\| = 1 \end{aligned}$$

Part 2: For some  $B$ ,  $\|\theta^{(k+1)}\| \leq B\sqrt{k}$

$$\begin{aligned} \|\theta^{(k+1)}\|^2 &= \|\theta^{(k)} + y^{(i)}x^{(i)}\|^2, \text{ Perceptron algorithm update} \\ &= \|\theta^{(k)}\|^2 + (y^{(i)})^2\|x^{(i)}\|^2 + 2y^{(i)}(\theta^{(k)} \cdot x^{(i)}) \\ &\leq \|\theta^{(k)}\|^2 + (y^{(i)})^2\|x^{(i)}\|^2, \text{ since } k^{\text{th}} \text{ mistake} \implies y^{(i)}(\theta^{(k)} \cdot x^{(i)}) \leq 0 \\ &= \|\theta^{(k)}\|^2 + R^2, \text{ since } (y^{(i)})^2\|x^{(i)}\|^2 = \|x^{(i)}\|^2 \leq R^2, \text{ by assumption and } (y^{(i)})^2 = 1 \\ \implies \|\theta^{(k+1)}\|^2 &\leq kR^2, \text{ by induction on } k \text{ since } (\theta^{(1)})^2 = 0 \\ \implies \|\theta^{(k+1)}\| &\leq \sqrt{k}R \end{aligned}$$

Part 3: Combine the bounds

$$\begin{aligned} k\gamma &\leq \|\theta^{(k+1)}\| \leq \sqrt{k}R \\ \implies k &\leq (R/\gamma)^2 \end{aligned}$$

- Perceptron will not converge.
- However, we can achieve a similar bound on the number of mistakes made in one pass (Freund, Schapire)

Main Takeaway:

### 3 $k$ -NN

#### 3.1 A Classification Example

Using the figure below, what would you categorize the green circle as with  $k = 3$ ?  $k = 5$ ?  $k = 4$ ?

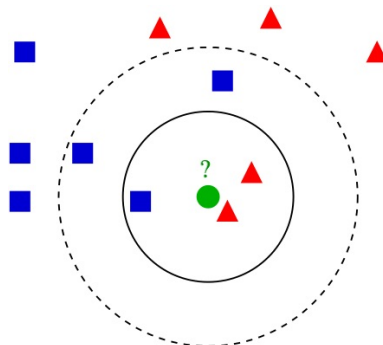
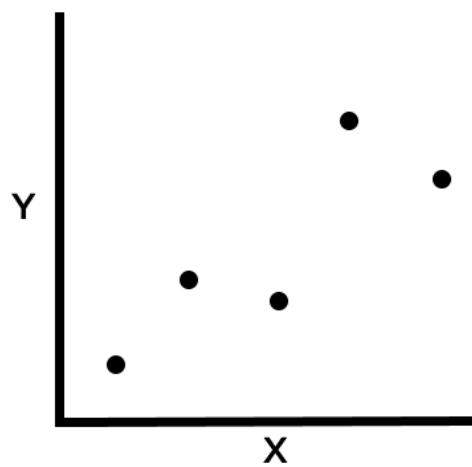
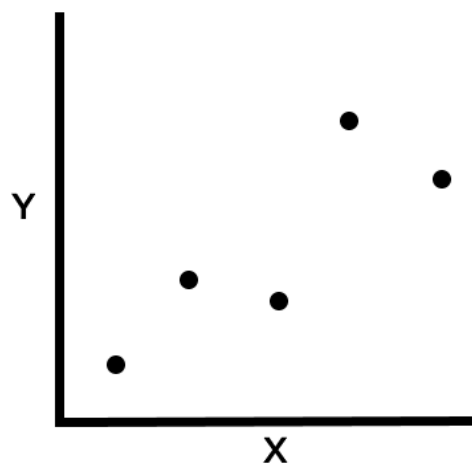
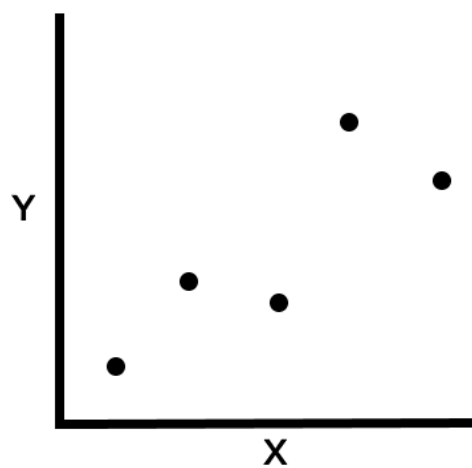


Figure 4: An example of  $k$ -NN on a small dataset; image source from [Wikipedia](#)

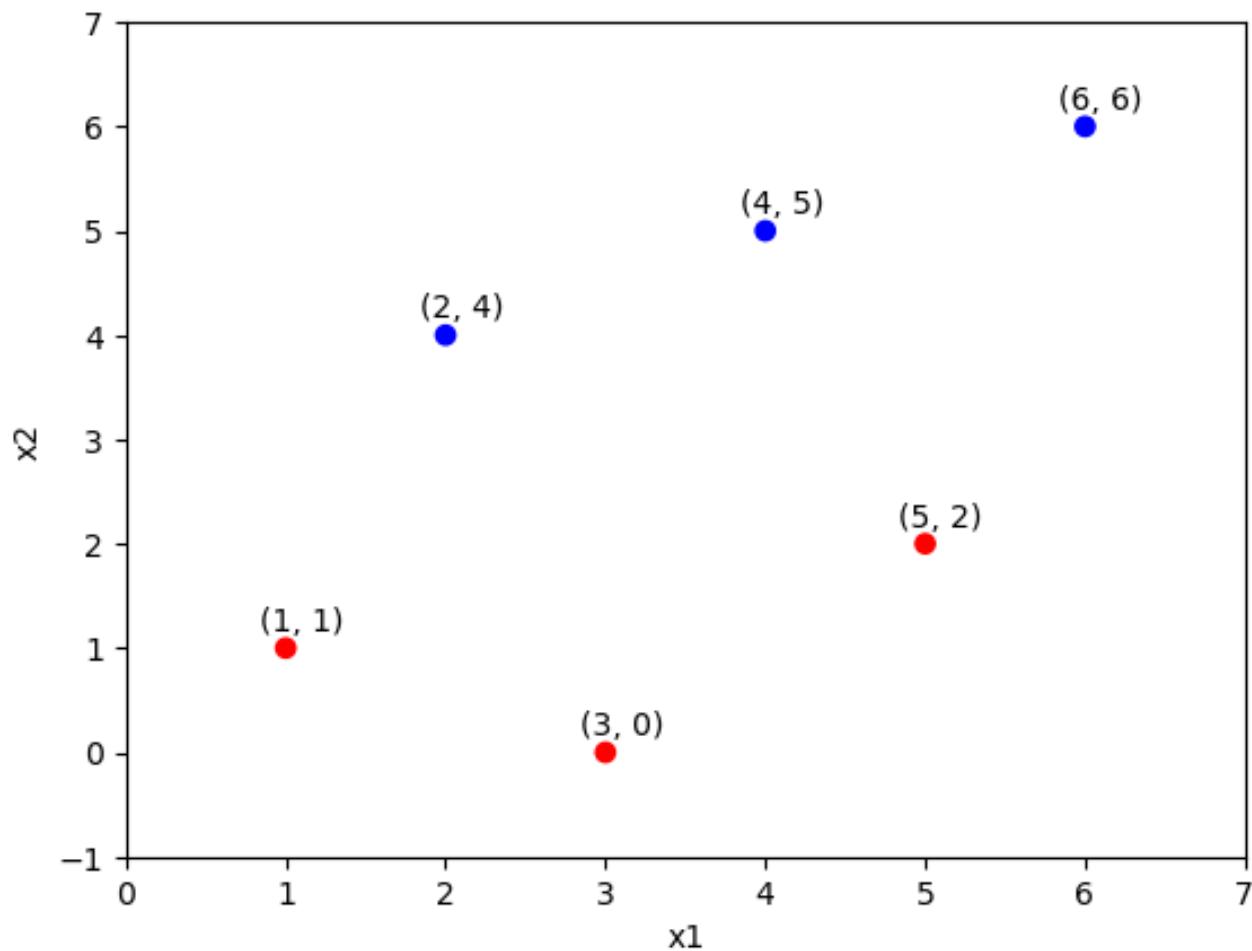
#### 3.2 $k$ -NN for Regression

You want to predict a continuous variable  $Y$  with a continuous variable  $X$ . Having just learned  $k$ -NN, you are super eager to try it out for regression. Given the data below, draw the regression lines (what  $k$ -NN would predict  $Y$  to be for every  $X$  value if it was trained for the given data) for  $k$ -NN regression with  $k = 1$ , weighted  $k = 2$ , and unweighted  $k = 2$ . For weighted  $k = 2$ , take the weighted average of the two nearest points. For unweighted  $k = 2$ , take the unweighted average of the two nearest points. (Note: the points are equidistant along the  $x$ -axis)

(a)  $k = 1$ (b) weighted  $k = 2$ (c) unweighted  $k = 2$



### 3.3 $k$ -NN Decision Boundary and Cross Validation



Draw the decision boundaries for the above training dataset given using  $k$ NN algorithm considering  $k=1$ .

Suppose we use 3-fold Cross Validation for this  $k$ NN, with  $k=1$ . The folds are  $[(1,1),(3,0)]$ ,  $[(2,4),(6,6)]$ , and  $[(4,5),(5,2)]$  What is the cross-validation error?

## 4 Linear Regression

### 4.1 Defining the Objective Function

1. What does an objective function  $J(\theta)$  do?
2. What are some examples?
3. What are some desirable properties of this function?

## 4.2 Solving Linear Regression using Gradient Descent

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

Now, we want to implement the gradient descent method.

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

1. What is the gradient of the objective function  $J(\theta)$  with respect to  $\theta$ :  $\nabla_{\theta} J(\theta)$ ?
2. How do we carry out the update rule?
3. How could we pick which value of  $\gamma$  to use if we weren't given the step size?

## 5 Summary

### 5.1 Decision Tree

Pros	Cons	Inductive bias	When to use
<ul style="list-style-type: none"> <li>• Easy to understand and interpret</li> <li>• Very fast for inference</li> </ul>	<ul style="list-style-type: none"> <li>• Tree may grow very large and tend to overfit.</li> <li>• Greedy behaviour may be sub-optimal</li> </ul>	<ul style="list-style-type: none"> <li>• Prefer the smallest tree consistent w/ the training data (i.e. 0 error rate)</li> </ul>	<ul style="list-style-type: none"> <li>• Most cases. Random forests are widely used in industry.</li> </ul>

### 5.2 $k$ -NN

Pros	Cons	Inductive bias	When to use
<ul style="list-style-type: none"> <li>• No training of parameters</li> <li>• Can apply to multi-class problems and use different metrics</li> </ul>	<ul style="list-style-type: none"> <li>• Slow for large datasets</li> <li>• Must select good <math>k</math></li> <li>• Imbalanced data and outliers can lead to misleading results</li> </ul>	<ul style="list-style-type: none"> <li>• Similar (i.e. nearby) points should have similar labels</li> <li>• All label dimensions are created equal</li> </ul>	<ul style="list-style-type: none"> <li>• Small dataset</li> <li>• Small dimensionality</li> <li>• Data is clean (no missing data)</li> <li>• Inductive bias is strong for dataset</li> </ul>

### 5.3 Linear regression

Pros	Cons	Inductive bias	When to use
<ul style="list-style-type: none"> <li>• Easy to understand and train</li> <li>• Closed form solution</li> </ul>	<ul style="list-style-type: none"> <li>• Sensitive to noise (other than zero-mean Gaussian noise)</li> </ul>	<ul style="list-style-type: none"> <li>• The true relationship between the inputs and output is linear.</li> </ul>	<ul style="list-style-type: none"> <li>• Most cases (can be extended by adding non-linear feature transformations)</li> </ul>

### 5.4 Perceptron

Pros	Cons	Inductive bias	When to use
<ul style="list-style-type: none"> <li>• Easy to understand and works for online learning.</li> <li>• Provable guarantees on mistakes made for linearly separable data.</li> </ul>	<ul style="list-style-type: none"> <li>• No guarantees on finding best (maximum-margin) hyperplane.</li> <li>• Output is sensitive to noise in the training data.</li> </ul>	<ul style="list-style-type: none"> <li>• The binary classes are separable in the feature space by a line.</li> </ul>	<ul style="list-style-type: none"> <li>• Not used much anymore, but variants (kernel perceptron, structured perceptron) may have more success.</li> </ul>