Recitation 3
Classification and Regression

10-301/10-601: Introduction to Machine Learning

09/19/2025

1 Slido

Figure 1: Join the Slido with this QR code or go to Slido.com and enter code 2613 742

2 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 $\mathbf{x}^{(i)} \in \mathbb{R}^2, y^{(i)} \in \{\text{Yellow}, \text{Purple}, \text{Green}\}$ as shown in Fig. 2, we wish to learn a decision tree to classify such points. Using the tree structure in Fig. 2, what values of α, β and leaf node predictions could we use to perfectly classify the points? Draw the associated decision boundaries on the plot.

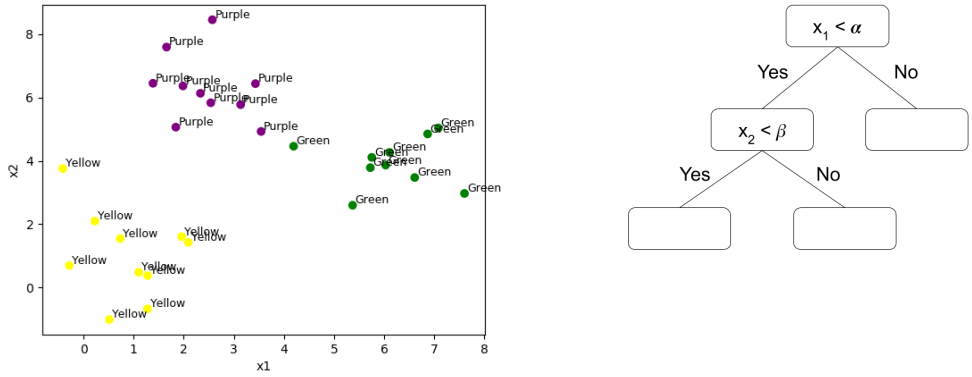
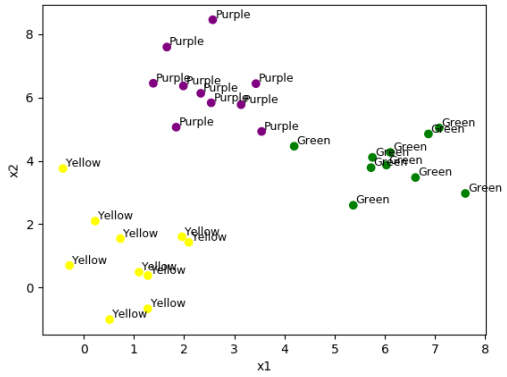
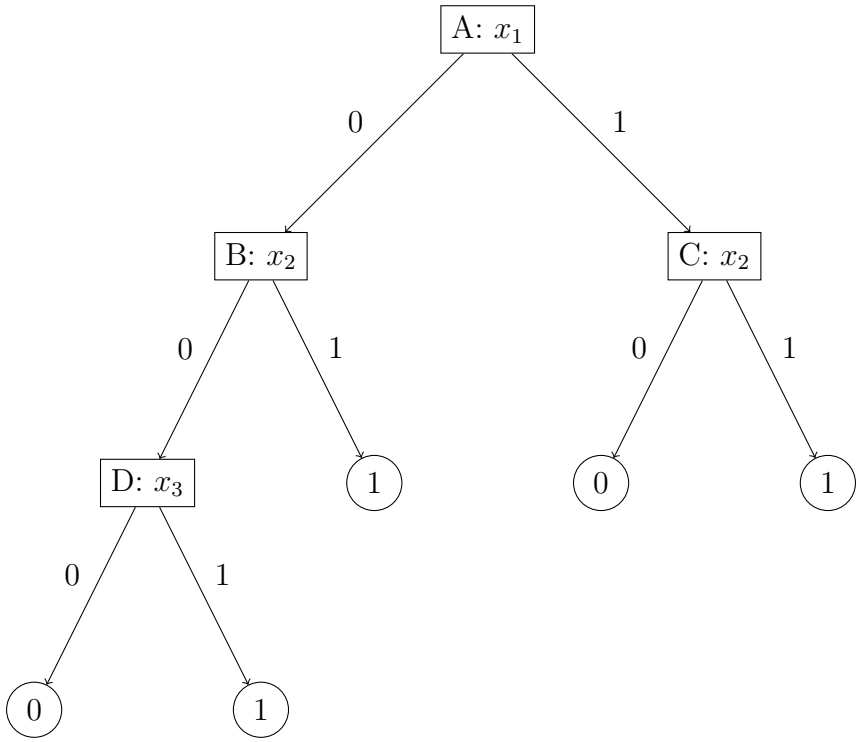


Figure 2: Classification of 2D points, with Decision Tree to fill in

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



3. **Pruning a Tree:** Which node would be the first to be pruned in the following decision tree? In the case of a tie, break the tie in favor of the alphabetically earlier node (eg. prune node B before D)



The following is the validation set, along with additional columns for us to use while solving. For simplicity, suppose that a pruned node is replaced by a prediction of 1.

x_1	x_2	x_3	Label	No prune	Prune A	Prune B	Prune C	Prune D
0	0	0	1					
0	0	1	0					
0	1	0	1					
1	0	1	0					
1	1	0	1					

Follow-up question: How do we know when we are done pruning a decision tree?

3 k-NN

3.1 A Classification Example

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

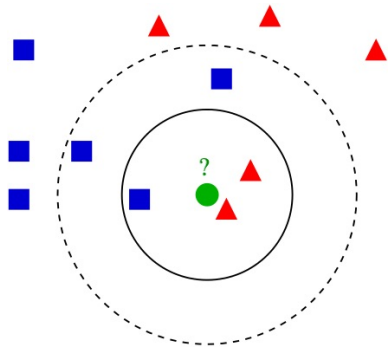
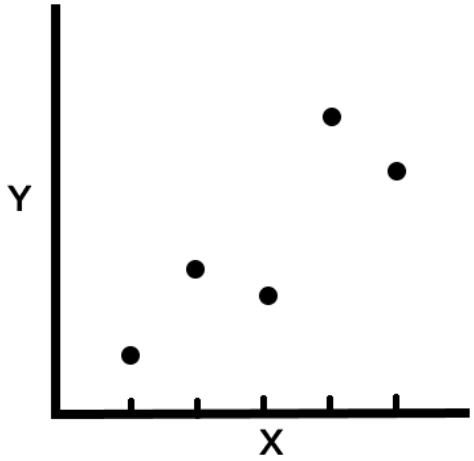


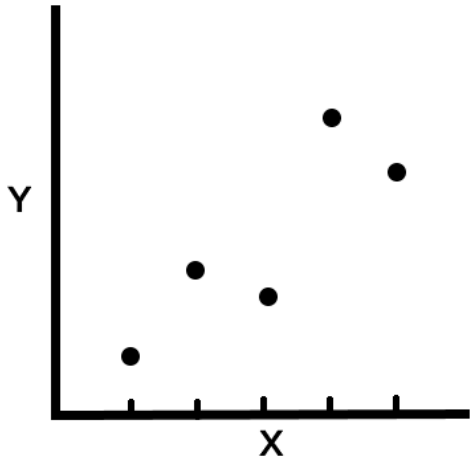
Figure 3: An example of k-NN on a small dataset; image source from Wikipedia

3.2 k-NN for Regression

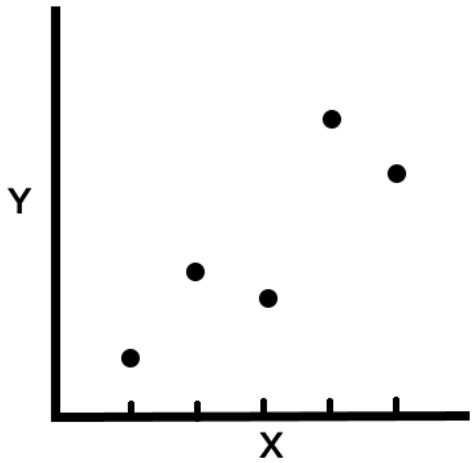
You want to predict a continuous variable Y with a continuous variable X. Having just learned k-NN, you are super eager to try it out for regression. Given the data below, draw the regression lines (what k-NN would predict Y to be for every X value if it was trained for the given data) for k-NN regression 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) $k = 1$

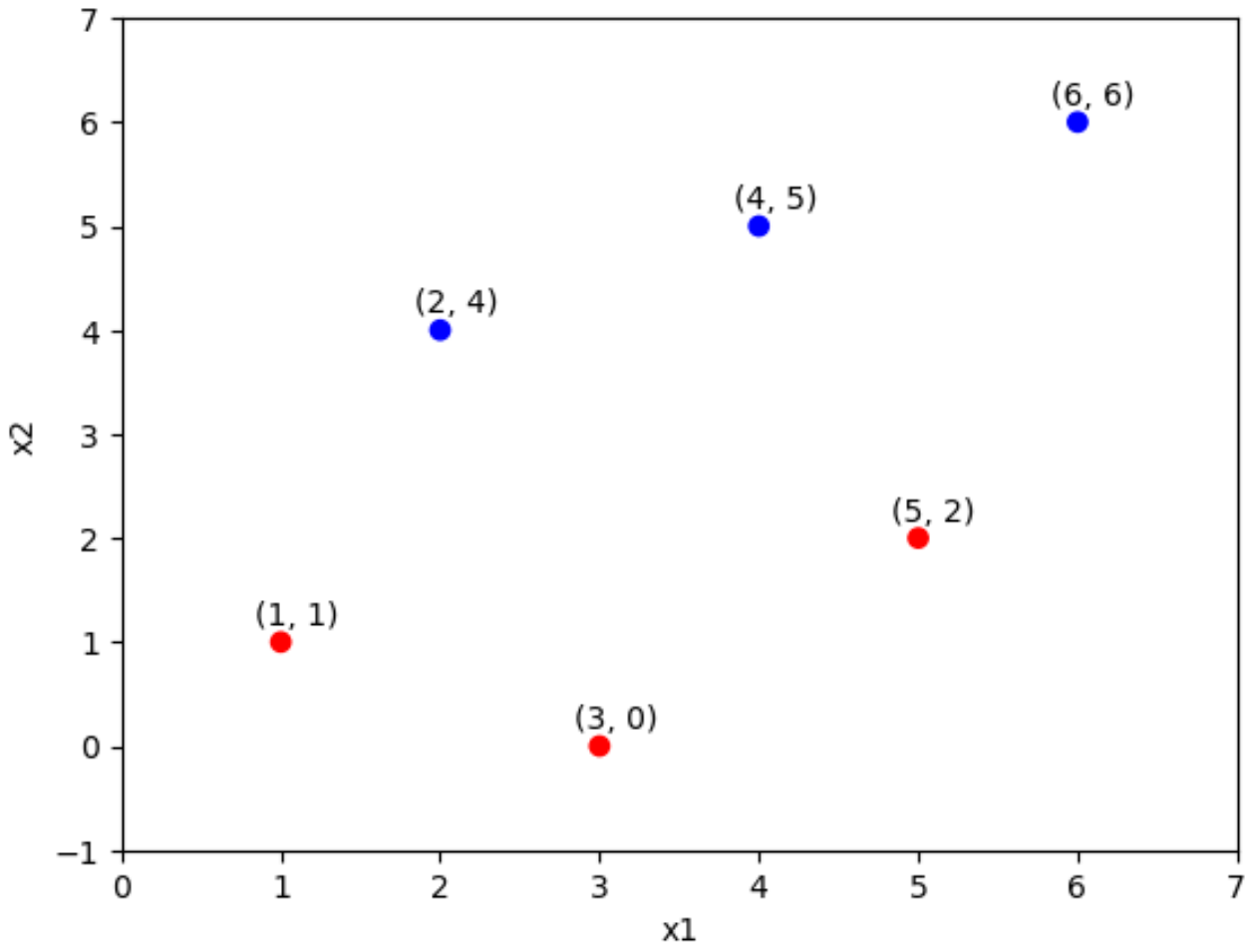


(b) weighted $k = 2$



(c) unweighted $k = 2$

3.3 *k*-NN Decision Boundary and Cross Validation



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

Suppose we use 3-fold Cross Validation for this kNN, 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 Perceptron

4.1 Perceptron Mistake Bound Guarantee

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

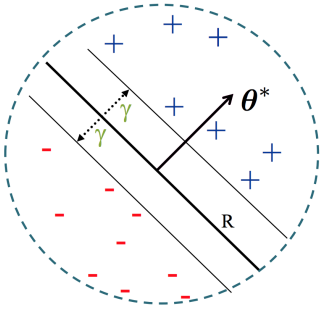


Figure 5: Perceptron Mistake Bound Setup

4.2 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 γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.
- The margin γ of a set of examples S is the maximum γ_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.

4.3 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: $\exists \boldsymbol{\theta}^*$ and $\boldsymbol{\gamma} > 0$ s.t. $||\boldsymbol{\theta}^*|| = 1$ and $y^{(i)}(\boldsymbol{\theta}^* \cdot x^{(i)}) \geq \boldsymbol{\gamma}, \forall i$

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

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.

4.4 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: $\exists \boldsymbol{\theta}^*$ and $\boldsymbol{\gamma} > 0$ s.t. $||\boldsymbol{\theta}^*|| = 1$ and $y^{(i)}(\boldsymbol{\theta}^* \cdot x^{(i)}) \geq \boldsymbol{\gamma}, \forall i$

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

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

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

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

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

Part 3: Combine the bounds

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

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

Main Takeaway:

5 Linear Regression

5.1 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?

5.2 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 $\boldsymbol{\theta}$ 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 θ: $\nabla_\theta J(\theta)$?

2. How do we carry out the update rule?

3. How could we pick which value of γ to use if we weren't given the step size?

6 Summary

6.1 Decision Tree

Pros	Cons	Inductive bias	When to use
• Easy to understand and interpret • Very fast for inference	• Tree may grow very large and tend to overfit. • Greedy behaviour may be sub-optimal	• Prefer the smallest tree consistent w/ the training data (i.e. 0 error rate)	• Most cases. Random forests are widely used in industry.

6.2 k-NN

Pros	Cons	Inductive bias	When to use
• No training of parameters • Can apply to multi-class problems and use different metrics	• Slow for large datasets • Must select good k • Imbalanced data and outliers can lead to misleading results	• Similar (i.e. nearby) points should have similar labels • All label dimensions are created equal	• Small dataset • Small dimensionality • Data is clean (no missing data) • Inductive bias is strong for dataset

6.3 Perceptron

Pros	Cons	Inductive bias	When to use
• Easy to understand and works for online learning. • Provable guarantees on mistakes made for linearly separable data.	• No guarantees on finding best (maximum-margin) hyperplane. • Output is sensitive to noise in the training data.	• The binary classes are separable in the feature space by a line.	• Not used much anymore, but variants (kernel perceptron, structured perceptron) may have more success.

6.4 Linear regression

Pros	Cons	Inductive bias	When to use
• Easy to understand and train • Closed form solution	• Sensitive to noise (other than zero-mean Gaussian noise)	• The true relationship between the inputs and output is linear.	• Most cases (can be extended by adding non-linear feature transformations)