Recitation 3
Classification and Regression

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

06/02/2022

1 Decision Trees and Beyond

1. **Decision Tree Classification with Continuous Attributes**
   Given the dataset $\mathcal{D}_1 = \{\mathbf{x}^{(i)}, y\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^2, y \in \{\text{Yellow}, \text{Purple}, \text{Green}\}$ 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 α, β and leaf node predictions could we use to perfectly classify the points? Now, draw the associated decision boundaries on the scatter plot.

$x \leq c$

$\alpha = 4$

$\beta = 4$

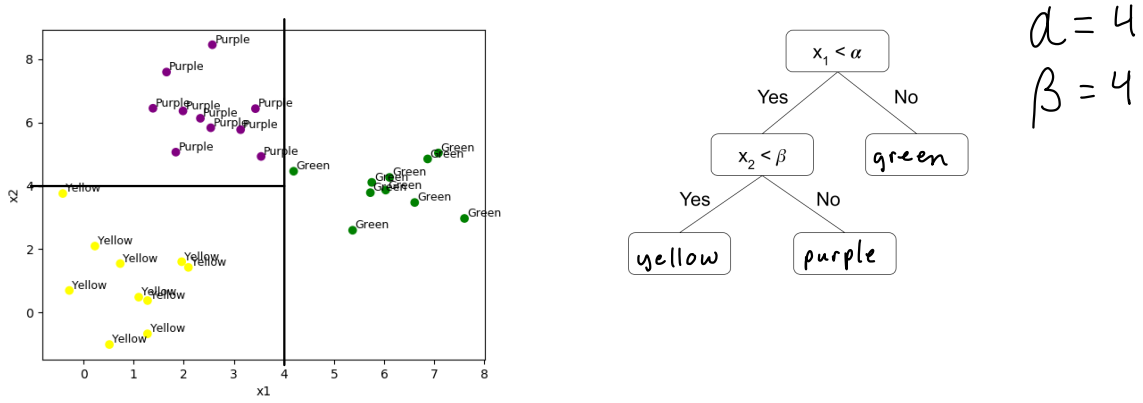
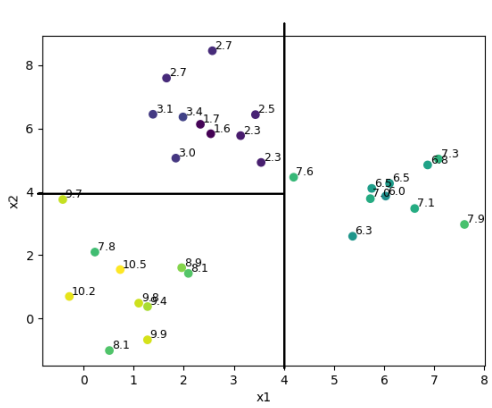


Figure 1: Classification of 2D points, with Decision Tree to fill in

Decision Tree Regression with Continuous Attributes

Now instead if we had dataset $\mathcal{D}_2 = \{\mathbf{x}^{(i)}, y\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^2, y \in \mathbb{R}$ 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 α, β 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.



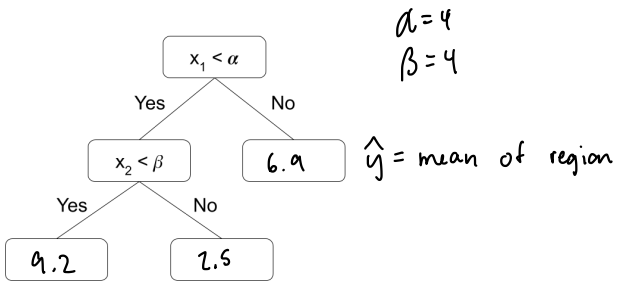


Figure 2: Regression on 2D points, with Decision Tree to fill in

$$\frac{1}{N} \sum_{i \in N} (y_i - \hat{y})^2 \qquad \text{diff} = 0$$

$$\hat{y} = \frac{1}{N} \sum y_i$$

2 k-NN

2.1 A Classification Example

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

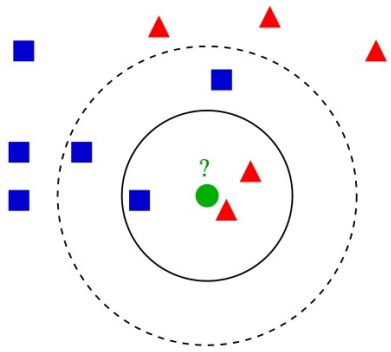


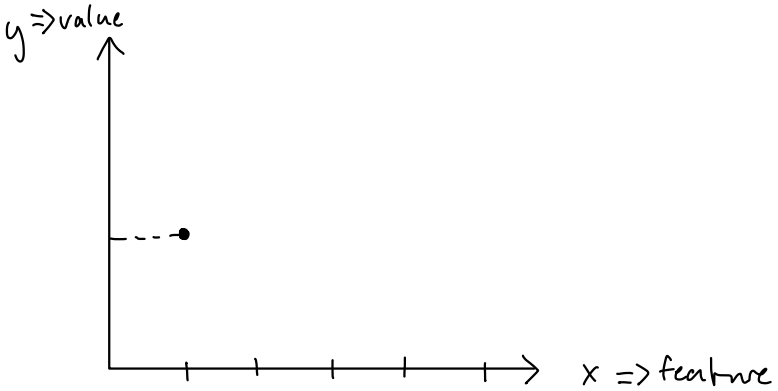
Figure 3: From wiki

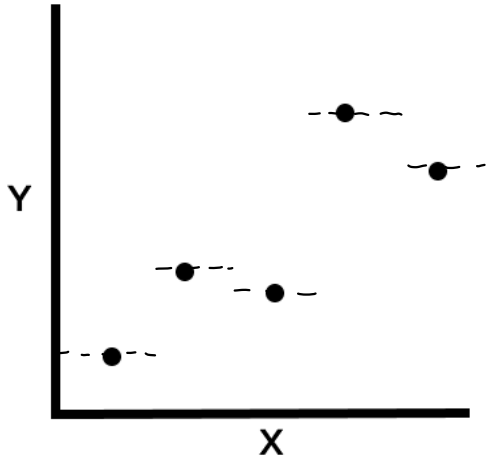
inductive bias:
points that are close have sim properties & sim properties have sim labels

k=3 => red triangle
k=5 => blue square

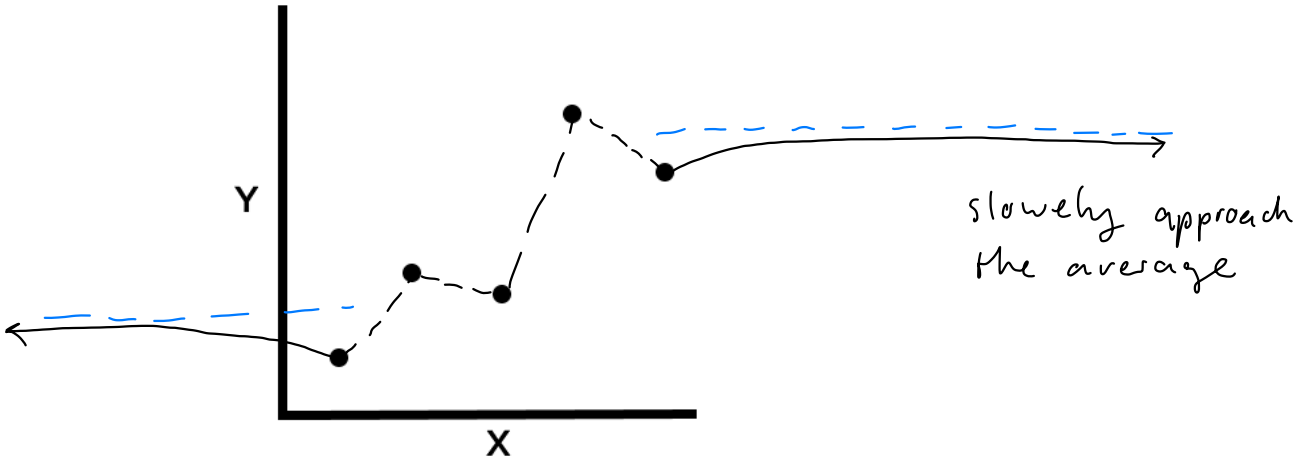
2.2 k-NN for Regression

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)

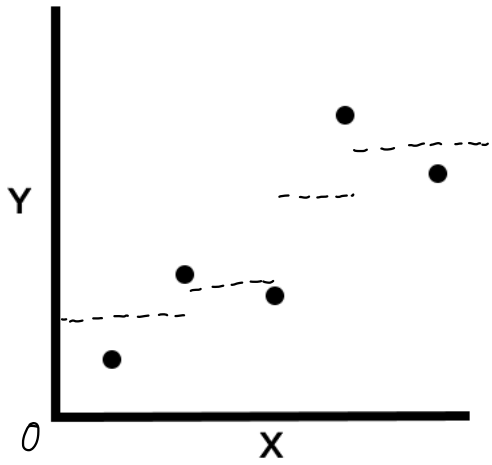




(a) $k = 1$



(b) weighted $k = 2$



(c) unweighted $k = 2$

3 Linear Regression

3.1 Defining the Objective Function

1. What does an objective function $J(\theta)$ do ? "cost function" → how good are our parameters (θ)?
2. What are some properties of this function?
   differentiability, convex
3. What are some examples?
   MSE, LQR, MAE

y = predicted
$\hat{y}$ = true

$$\frac{1}{N}\sum (y-\hat{y})^2$$

3.2 Solving Linear Regression using Gradient Descent

		points				
		$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	$\mathbf{x}^{(3)}$	$\mathbf{x}^{(4)}$	$\mathbf{x}^{(5)}$
features	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
label	y	2.0	4.0	7.0	8.0	11.0

Now, we want to implement the gradient descent method.

Assuming that $\alpha = 0.1$ and w 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?

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$$

gradient; α: hyperparameter (learning rate) → update to better weights

$$w = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

adding bias term

1) $$J(\theta) = \frac{1}{5}\sum_{i=1}^{5}(y-\hat{y})^2 = \frac{1}{5}\sum_{i=1}^{5}\left(y-\sum_{j=0}^{3}\theta_j x_j^{(i)}\right)^2$$

$\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \dots \theta_3 x_3^{(i)}$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{5}\sum_{i=1}^{5} 2\left(y-\sum_{j=0}^{3}\theta_j x_j^{(i)}\right)\cdot -x_0^{(i)}$$

different; a

$$\nabla_\theta J_\theta = \begin{bmatrix} a \cdot -x_0^{(i)} \\ a \cdot -x_1^{(i)} \\ \vdots \\ a \cdot -x_3^{(i)} \end{bmatrix}$$

2) $$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{5}\sum_{i=1}^{5} 2\left(y-\sum_{j=0}^{3}\theta_j x_j^{(i)}\right)\cdot -x_0^{(i)}$$

$$= \frac{1}{5}\sum_{i=1}^{5} 2(y)\cdot -x_0^{(i)}$$

$$= \frac{2}{5}(2\cdot -1 + 4\cdot -1 + 7\cdot -1 + 8\cdot -1 + 11\cdot -1)$$

$$= -12.8$$

$$\nabla_\theta J(\theta) = \begin{pmatrix} -12.8 \\ -47.2 \\ 100.4 \\ -140.4 \end{pmatrix}$$

$$\theta' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 0.1 \cdot \begin{bmatrix} -12.8 \\ -47.2 \\ 100.4 \\ -140.4 \end{bmatrix} = \begin{bmatrix} 1.28 \\ 4.72 \\ -10.04 \\ 14.04 \end{bmatrix}$$

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.

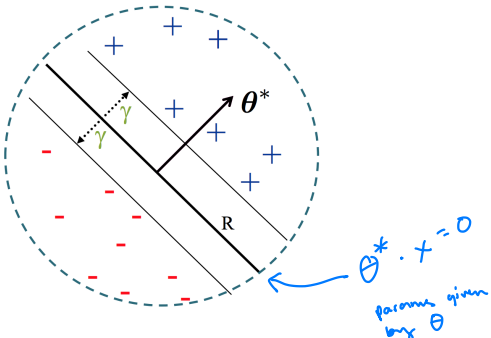


Figure 5: Perceptron Mistake Bound Setup

Bound on mistakes we can make

4.2 Definitions

distance between line & closest point to line

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.

point to plane formula

$$\frac{|ax+by+c|}{\sqrt{a^2+b^2}}$$

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

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, $Ak \leq ||\boldsymbol{\theta}^{(k+1)}||$

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

Part 3: Combine the bounds

Main Takeaway:

the perceptron does converge for linearly seperable data in finitely many steps.

proof both lower & upper bound

$\gamma k \leq ||\theta^{(k+1)}||$

$||\theta^{(k+1)}|| \leq R\sqrt{k}$

$\gamma k \leq ||\theta^{(k+1)}|| \leq R\sqrt{k}$

$\gamma k \leq R\sqrt{k}$

$\sqrt{k} \leq \frac{R}{\gamma}$

$k \leq \left(\frac{R}{\gamma}\right)^2$

5 Summary

5.1 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

5.2 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)

5.3 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.

5.4 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.