Recitation 4
Classification and Regression

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

02/25/2021

1 k-NN

1.1 Warm Up - A Classification Example

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

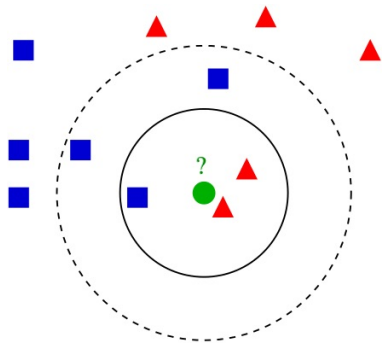
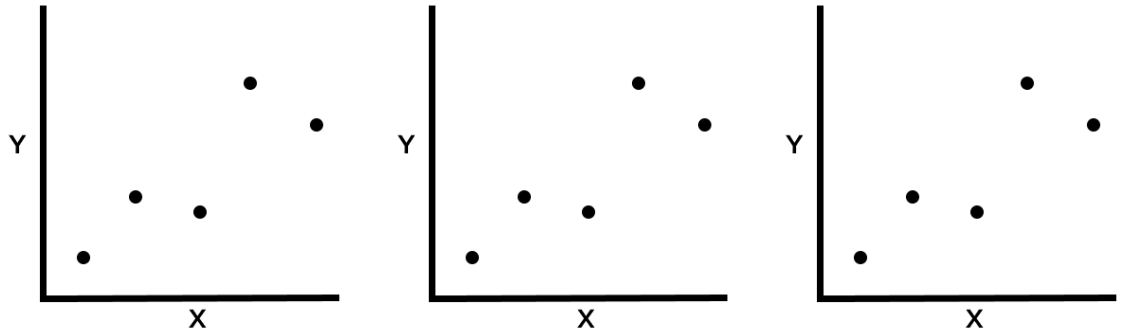


Figure 1: From wiki

1.2 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.



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

2 Linear Regression

2.1 Defining the Objective Function

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

2.2 Deriving the Closed-form Solution

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.

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

We now extend the data set to include more features, $\mathbf{x} \in \mathbb{R}$:

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

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

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

1. What is the closed-form formula for w_1?

2. What is the closed-form matrix solution for $\mathbf{w}$?

3 Gradient Descent

3.1 Solving Linear Regression using Gradient Descent

We use the same data set from last section. However, 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 , $J(\mathbf{w})$, w.r.t $\mathbf{w}$: $\nabla_{\mathbf{w}} J(\mathbf{w})$

2. How do we carry out the update rule?

4 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. 3, we wish to learn a decision tree for classifying such points. Provided with a possible tree structure in Fig. 3, 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.

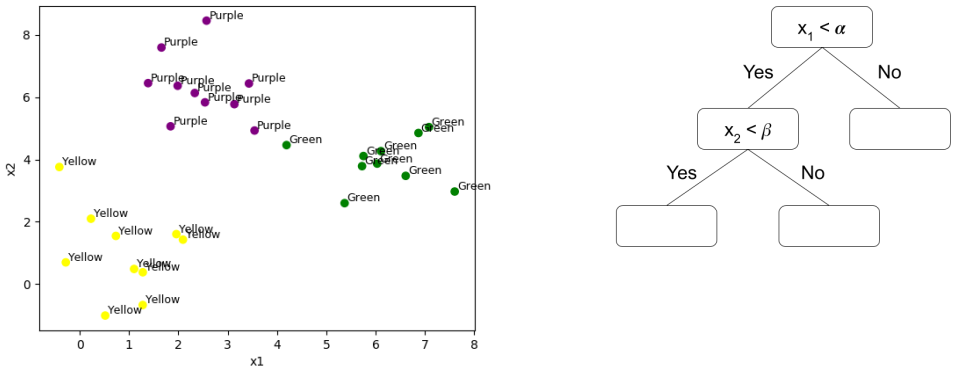


Figure 3: Classification of 2D points, with Decision Tree to fill in

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. 4, 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.

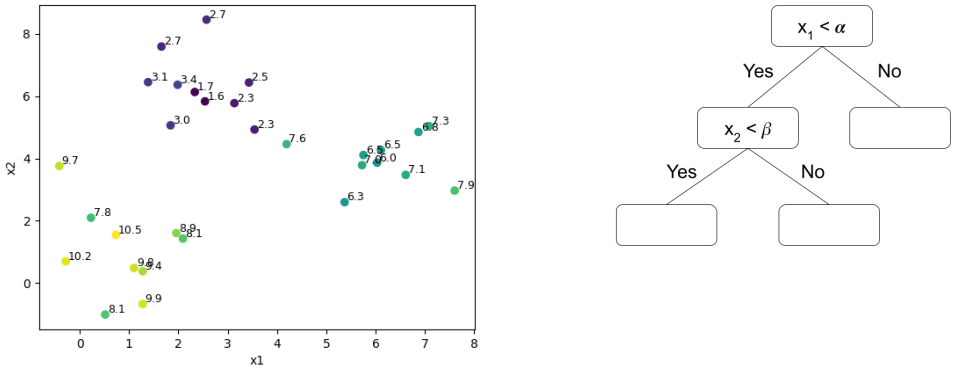


Figure 4: Regression on 2D points, with Decision Tree to fill in

2. **Feature engineering - A good model isn't everything.** Given a two-dimensional dataset shown in Fig.5, 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?

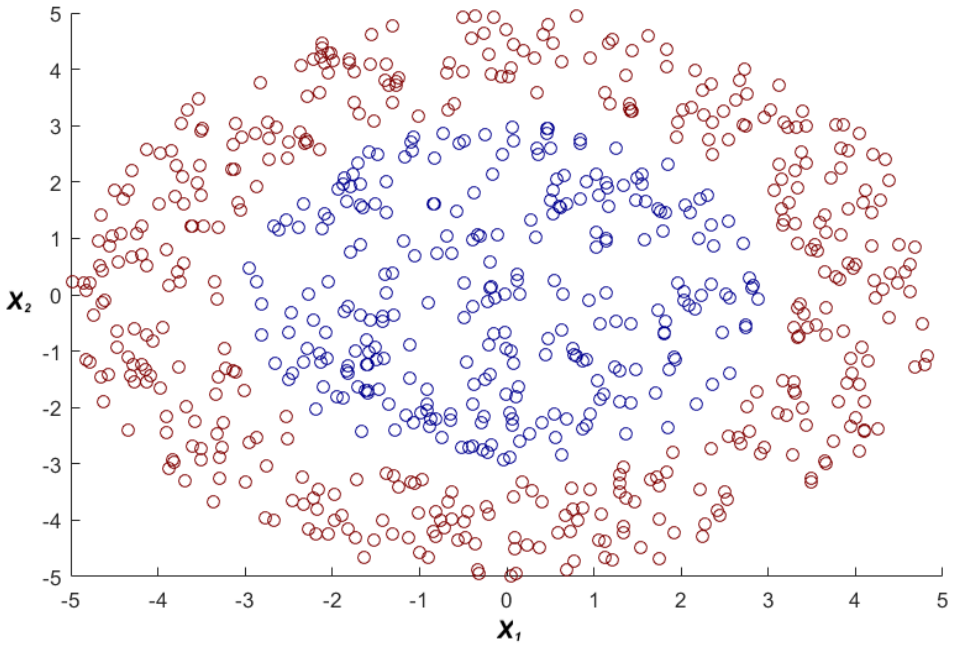


Figure 5

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

☐ Linear regression on $\mathbf{x}$ and y

☐ Decision tree using attribute $\mathbf{x}$ and label y trained with ID3 algorithm

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

5 Summary

5.1 k-NN

Pros	Cons	Inductive bias	When to use
• Simple, minimal assumptions made about data distribution • No training of parameters • Can apply to multi-class problems and use different metrics	• Becomes slow as dataset grows • Requires homogeneous features • Selection of k is tricky • Imbalanced data can lead to misleading results • Sensitive to outliers	• 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 relationship between the inputs x and output y is linear. i.e. hypothesis space is Linear Functions	• 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 in an online learning setting. • Provable guarantees on mistakes made if the data is known to be linearly separable (perceptron mistake-bound).	• No guarantees on finding maximum-margin hyperplane (like in SVM), only that you will find **a** separating hyperplane. • Output is sensitive to noise in the training data.	• The binary classes are separable in the feature space by a line.	• The basic perceptron algorithm is not used much anymore, but other variants mentioned in class such as kernel perceptron or structured perceptron may have more success.