K Nearest Neighbor

Anahita Zarei, Ph.D.

Outline

- K Nearest Neighbor Classification and Regression
  - Distance Measures for Continuous and Symbolic Features
  - Curse of Dimensionality
  - Normalization and Weighting of the Features
- Reading:
  - Mitchell: 8.1-8.2
  - Abu-Mustafa: 6.2
  - James: 3.5

Overview of KNN

One of the simplest and oldest Machine Learning algorithms.

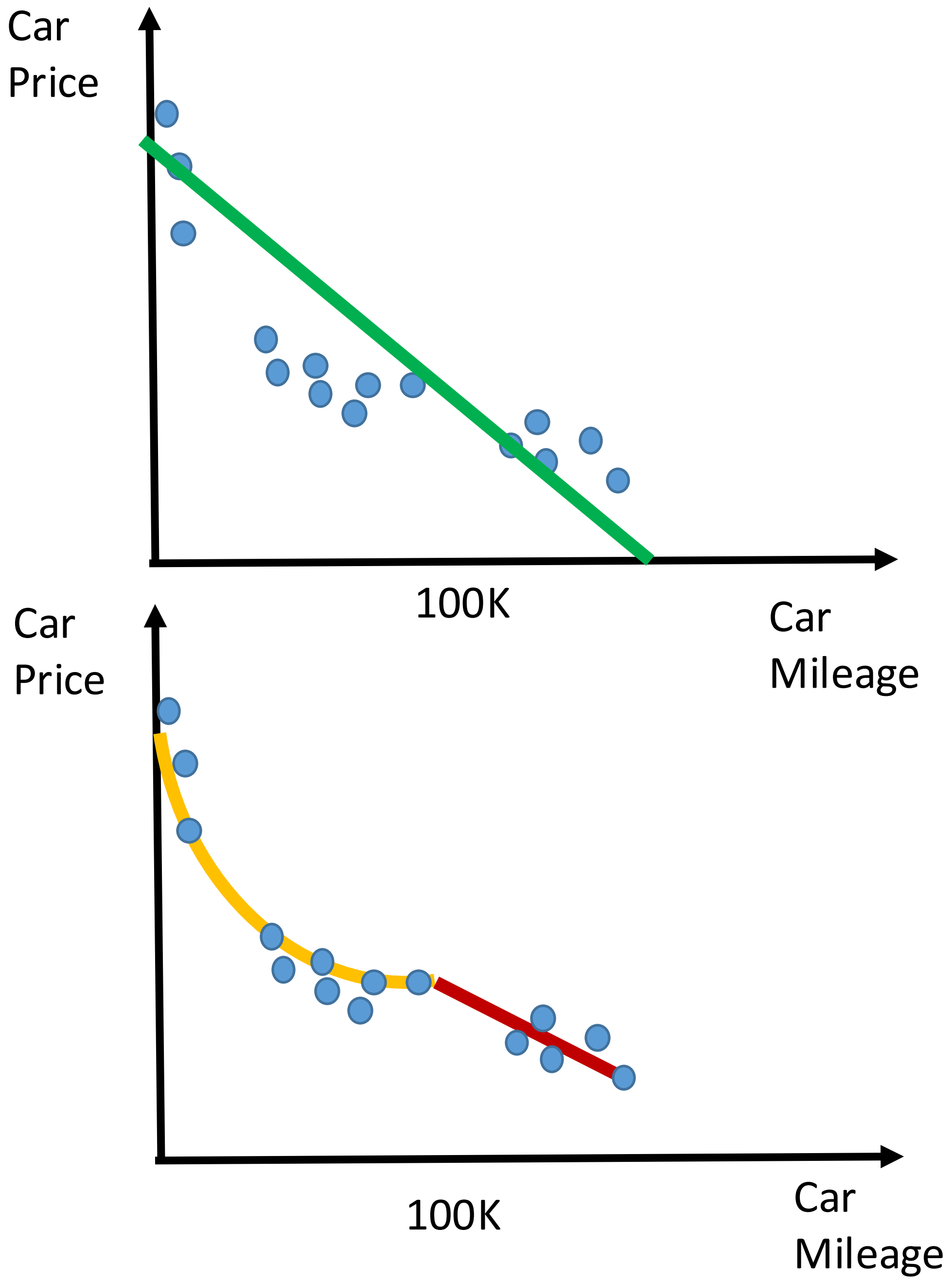
An example of:

- Instance-based Learning
- Memory-based Learning
- Lazy Learning
- Nonparametric
- Local Fit

Instead of explicitly training a model, we do nothing until seeing the test samples. We then compare the new test instances with instances seen in training, which have been stored in memory to determine their output.

Local vs. Global Fits

- Global fits, fit a global function across the whole input space
- Local fits have the flexibility to have a more local description for different regions of the input space.



Rational Behind Nearest Neighbor Method

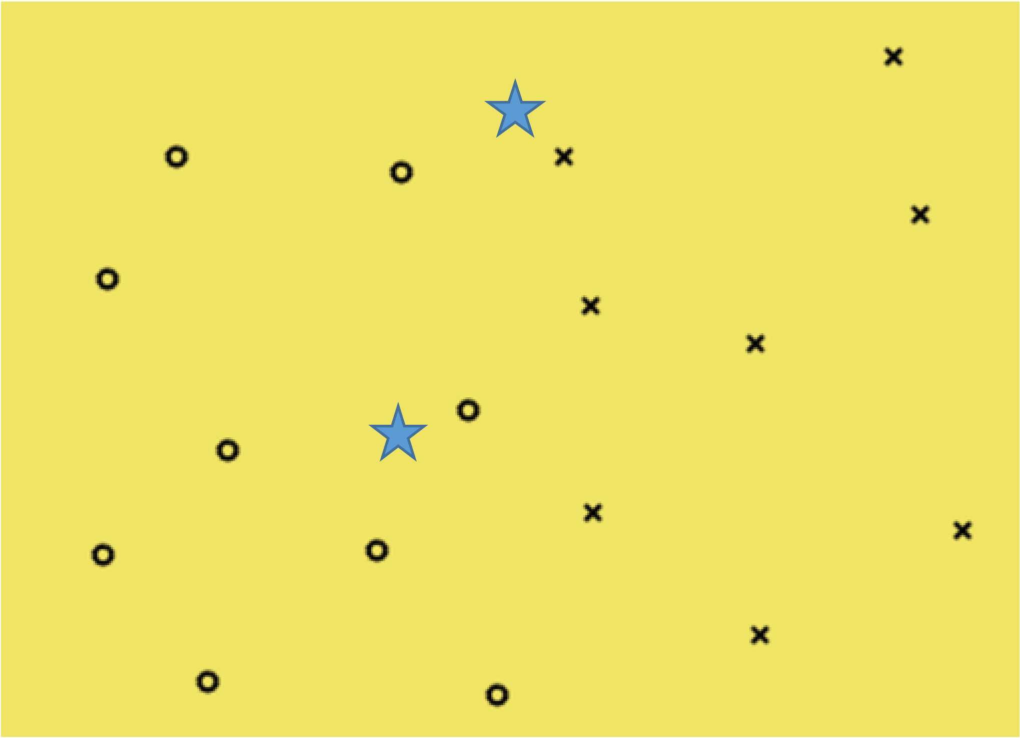
How much do you sell your car on Craig's List?!

Your car $ = ?

The most similar car on Craig's List = $8,000

Rational Behind Nearest Neighbor Method

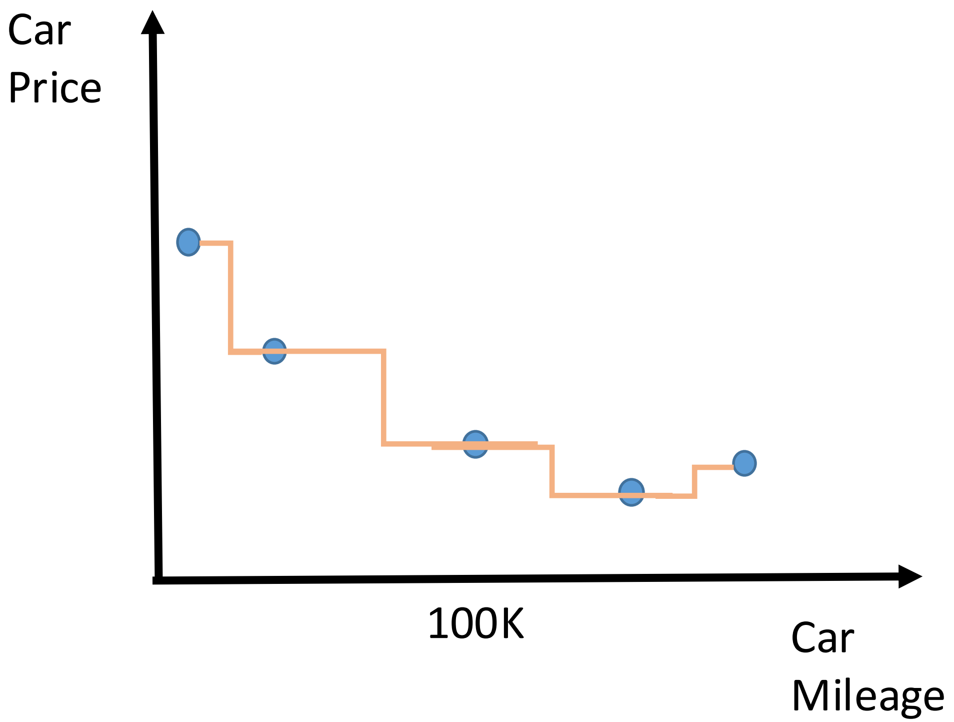
- Objects that are closer to each other are more *similar* than those who are far apart.
- Nearest Neighbor is a method for finding the output of an object based on *closest* training examples in the feature space.
- **Classification:** The output is the label of the closest neighbor as in the figure.
- **Regression:** The output is the continuous value of the closest neighbor.
- A new point will inherit the label or value of the closest point in the training set.



O: Class 1
X: Class 2

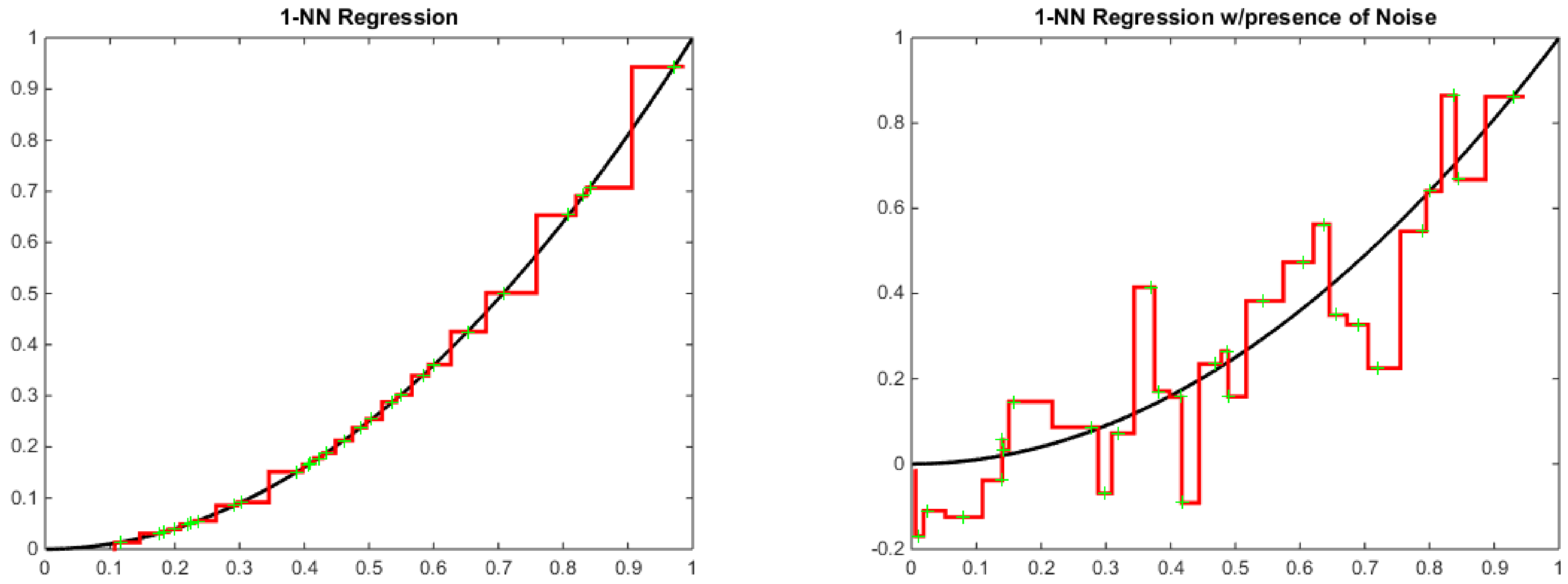
What's the class of stars?

Regression Example in 1-D



1-NN Performance

- 1-NN performs well when there's a lot of data and a low level of noise.
- Doesn't perform that well when data is sparse.
- It struggles to interpolate across regions of the input space where there are not any observations
- Is highly sensitive to noise and as such prone to overfitting.



Improving Nearest Neighbor Method

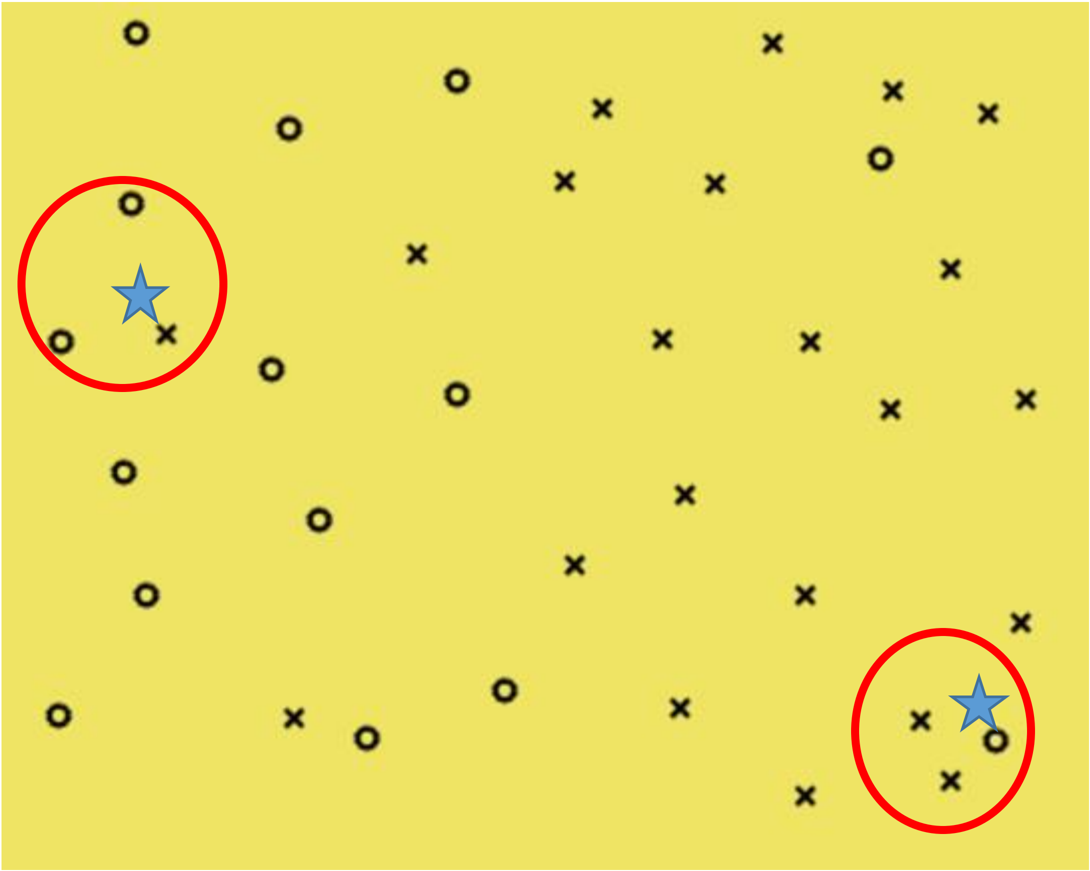
How much do you sell your car on Craig's List?!

Your car $ = ?

The most 3 similar cars on Craig's List

Improving the One Nearest Neighbor

- A non-parametric prediction algorithm increases the complexity of the model as the data increases and it can achieve zero bias. 1-Nearest Neighbor has **a small bias**, but can have **a large variance** for any finite sample size.
- What if there are outliers or unusual patterns?
- One way to reduce the variance is to instead of looking at the nearest neighbor to look at the K-nearest neighbors.
- **Classification:**
  – pick the class label which is most common in this set (take vote among neighbors).
  – classify test point as belonging to this class
  – Note: for two-class problems, if k is odd (k=1, 3, 5, …) there will never be any “ties”.
  – Example: see figure for the case of k=3.
- **Regression:**
  – Usually just average the y-values of the k closest training examples



O: Class 1
X: Class 2

What’s the class of stars?

KNN Output

Given x_q , when target is real values, take the mean of f values of k nearest neighbors.

$$f(x_q) = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$$

Distance-Weighted KNN

- One obvious refinement to the KNN is to weight the contribution of each of the k neighbors according to their distance to the query point.
- This can be accomplished by modifying the expression from previous slide as follows:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Distance Measures for Continuous Features

- What does close mean?
- We can use an arbitrary distance function to define a nearest neighbor.
- For a lot of applications simple measurements work.
- We can use any of the L norms for continuous features:

$$\left|\left|L(\overrightarrow{X_1}, \overrightarrow{X_2})\right|\right|_n = \sqrt[n]{|x_{1,1} - x_{2,1}|^{\mathrm{n}} + \cdots |x_{1,d} - x_{2,d}|^{\mathrm{n}}}$$

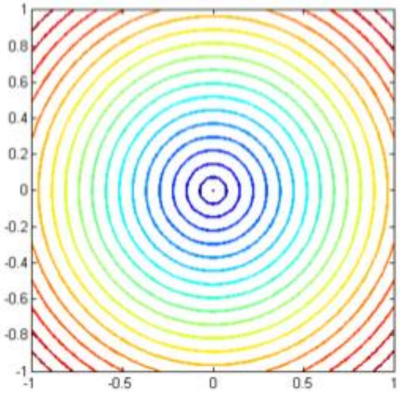
where

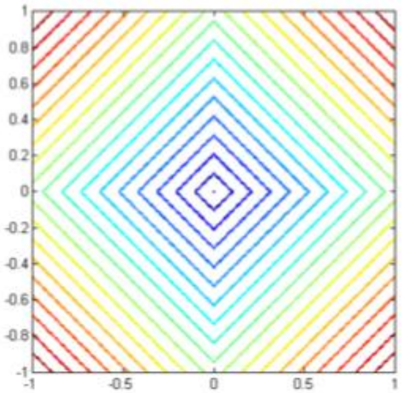
$$\overrightarrow{X_1} = (x_{11}, x_{12}, \ldots, x_{1d})$$
$$\overrightarrow{X_2} = (x_{21}, x_{22}, \ldots, x_{2d})$$

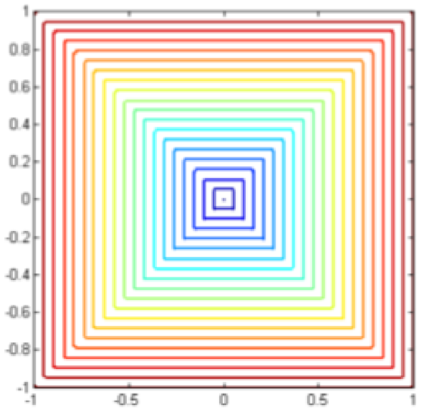
Distance Measures for Continuous Features

- Euclidean Distance: $||L||_2 = \sqrt[2]{|x_{1,1} - x_{2,1}|^2 + \cdots |x_{1,d} - x_{2,d}|^2}$
- Manhattan distance: $||L||_1 = |x_{1,1} - x_{2,1}| + \cdots + |x_{1,d} - x_{2,d}|$
- $||L||_\infty = \max_d |x_{1,d} - x_{2,d}|$

Contours of $||L(\vec{X}, \vec{X_i})||_n$ for n=2, 1, and ∞
where x is placed at the center:

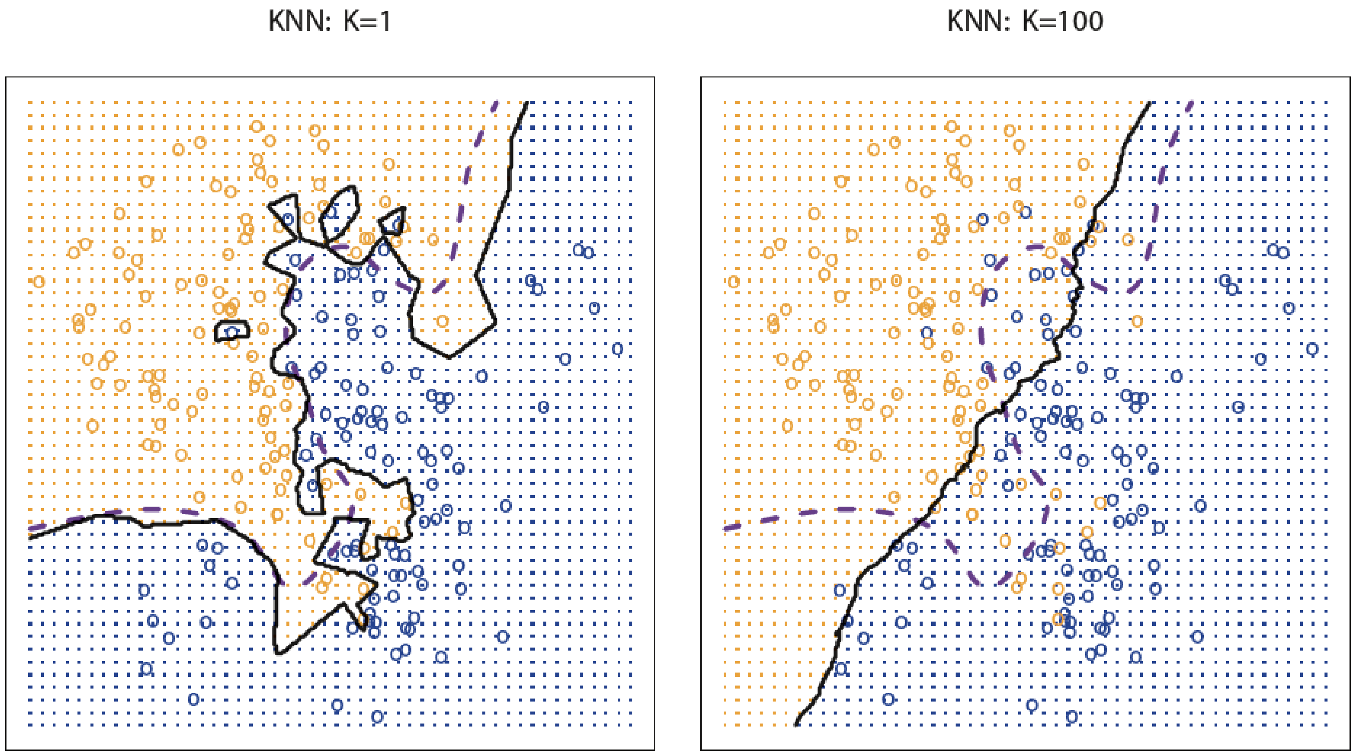


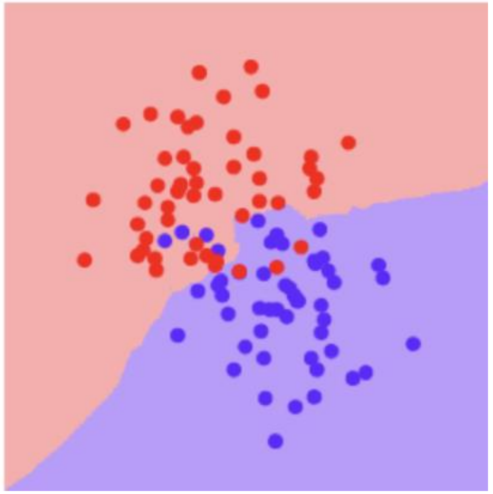


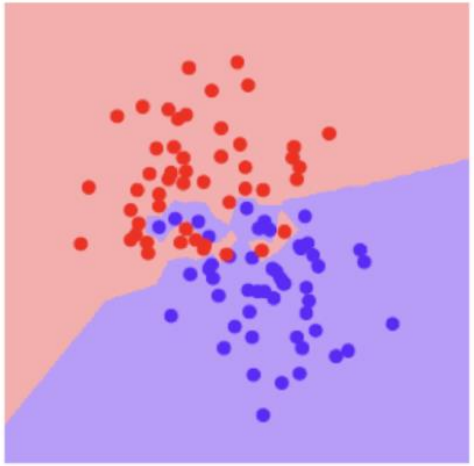


Effect of K on Decision Boundary

- The choice of K has a drastic effect on the KNN classifier obtained.
- Figure on the right displays two KNN fits using K = 1 and K = 100. When K = 1, the decision boundary (solid black) is overly flexible and finds patterns in the data that don't correspond to the Bayes decision boundary (dashed purple).
- As K grows, the method becomes less flexible and produces a decision boundary that is close to linear.
- On this simulated data set, neither K = 1 nor K = 100 give good predictions.







Effect of K on Decision Boundary

- Increasing K simplifies the decision boundary.
- Majority voting maps to less emphasis on individual points.
- Which one of the these pictures correspond to a decision boundary with a larger k?
- The decision boundary on top is smoother/simpler, hence corresponding to a larger K.

Distance Measure for Symbolic (Categorical) Features

- A variable representing color is an example of symbolic feature.
- It might have values such as *red, black* and *white*, which could be represented by the integers 1 through 3, respectively. Using a linear distance measurement on such values makes little sense.
- The **Hamming Distance** is a number used to denote the difference between two binary strings.
- For each feature if they're the same, the distance is zero, otherwise the distance is one.
- For example, denote the colors, red, black and white by the binary string c1,c2,c3. Using hamming distance red and black have the same distance as red and white.

	C1	C2	c3
Red	1	0	0
Black	0	1	0
White	0	0	1

Value Difference Metric (VDM)

- VDM was introduced in 1986 to provide an appropriate distance function for symbolic attributes.
- It's based on the idea that the goal of finding the distance is to find the right class. One way to measure this is to look at the following conditional probabilities.
- $\delta(val_i, val_j) = \sum_{h=1}^{\#\ classes} |P(c_h|val_i) - P(c_h|val_j)|^2$
- You then plug in delta in the Euclidean Distance:
- $||L||_2 = \sqrt[2]{|x_{1,1} - x_{2,1}|^2 + \cdots |x_{1,d} - x_{2,d}|^2}$

Example

The following table summarizes data on cars sold at a dealer.

A) Find the VDM among all colors.

B) Assume your friend tells you that she bought a red car with MPG of 28. Predict her car type using 3-NN algorithm.

Color	HWY MPG	Car Type
White	23	Van
Red	28	Sport
Black	32	Sport
Red	42	Sedan
Red	40	Sedan
White	20	Van

Solution

- $\delta(val_i, val_j) = \sum_{h=1}^{\# classes} |P(c_h|val_i) - P(c_h|val_j)|^2$

Color	HWY MPG	Car Type
White	23	Van
Red	28	Sport
Black	32	Sport
Red	42	Sedan
Red	40	Sedan
White	20	Van

Class 1:Van	Class 2: Sport	Class 3: Sedan
P(Van|White)	P(Sport|White)	P(Sedan|White)
P(Van|Red)	P(Sport|Red) 1/3	P(Sedan|Red)
P(Van|Black)	P(Sport|Black)	P(Sedan|Black)

Solution – cont.

- $\delta(val_i, val_j) = \sum_{h=1}^{\#\, classes} |P(c_h|val_i) - P(c_h|val_j)|^2$

Color	Car Type
White	Van
Red	Sport
Black	Sport
Red	Sedan
Red	Sedan
White	Van

Class 1:Van	Class 2: Sport	Class 3: Sedan
P(Van|White) 1	P(Sport|White) 0	P(Sedan|White) 0
P(Van|Red) 0	P(Sport|Red) 1/3	P(Sedan|Red) 2/3
P(Van|Black) 0	P(Sport|Black) 1	P(Sedan|Black) 0

Solution – cont.

Class 1:Van	Class 2: Sport	Class 3: Sedan
P(Van|White) 1	P(Sport|White) 0	P(Sedan|White) 0
P(Van|Red) 0	P(Sport|Red) 1/3	P(Sedan|Red) 2/3
P(Van|Black) 0	P(Sport|Black) 1	P(Sedan|Black) 0

$$\delta(val_i, val_j) = \sum_{h=1}^{\#\,classes} |P(c_h|val_i) - P(c_h|val_j)|^2$$

$\delta(Red, White)$= $|P(Van|Red) - P(Van|White)|^2$+ $|P(Sport|Red) - P(Sport|White)|^2$+$|P(Sedan|Red) - (Sedan|White)|^2$

$\delta(Red, White)$ = $(0-1)^2$ + $(1/3 - 0)^2$ + $(2/3-0)^2$ = 1.556

$\delta(Red, Black) = |P(Van|Red) - P(Van|Black)|2 + |P(Sport|Red) - P(Sport|Black)|^2 + |P(Sedan|Red) - (Sedan|Black)|^2$

$\delta(Red, Black)$ = $(0-0)^2$ + $(1/3 - 1)^2$ + $(2/3-0)^2$ =0.889

$\delta(Red, Red)$=0

Solution – cont.

$$||L||_2 = \sqrt[2]{\delta_{ij} + |x_{1,2} - x_{2,2}|^2}$$

Color	HWY MPG	Car Type	Distance
White	23	Van	$\sqrt{1.55 + (28-23)^2}$=5.153
Red	28	Sport	$\sqrt{0 + (28-28)^2}$=0
Black	32	Sport	$\sqrt{0.889 + (28-32)^2}$=4.110
Red	42	Sedan	$\sqrt{0 + (28-42)^2}$=14
Red	40	Sedan	$\sqrt{0 + (28-40)^2}$=12
White	20	Van	$\sqrt{1.55 + (28-20)^2}$=8.096
Red	28	?	

Using 3-NN, the prediction will be that her car is a sport car.

Question: What would have been the prediction if you only used the color?

Answer: The minimum distance would have corresponded to row 2, 4, and 5 and therefore using 3-NN the type would have been Sedan.

- If all features are equally important we must normalize the features to balance the impact of different features in prediction.
- For example in previous example we don't normalize the MPG, it will have a dominant effect on distance measure.
- Typical Normalization Methods:
  - Linear: (x – minX)/(maxX-minX)
  - Gaussian: $(x-\mu)/\sigma$

Normalization

- KNN can be greatly impacted by curse of dimensionality and is easily misled in high dimensions.
- The K observations that are nearest to a given test observation in low dimension may be very far away high-dimensional space, leading to a very poor KNN fit.
- This results from the fact that in higher dimensions there is effectively a reduction in sample size.
- For example a few hundred training observations provide enough information to accurately estimate f(x) in 2 dimension.
- However, spreading the same number of observations over 20 dimensions results in ***curse of dimensionality*** in which a given observation has no nearby neighbors.

Curse of dimensionality

- One issue with KNN algorithms is that all attributes are weighted equally in the distance formula:

$$\left|\left|L(\overrightarrow{X_1},\overrightarrow{X_2})\right|\right|_n = \sqrt[n]{|x_{1,1}-x_{2,1}|^{\mathrm{n}} + \cdots |x_{1,d}-x_{2,d}|^{\mathrm{n}}}$$

- For example, maybe 2 out of 20 attributes are relevant in determining the class.
    - Instances that have identical values for the 2 relevant attributes may be distant from one another in the 20-dimensional instance space. As such, the similarity metric used by KNN depending on all 20 attributes-will be misleading
- For example, maybe the mileage and the year of the car are more important attributes in determining the price than color of the car.
- This difficulty, which arises when many irrelevant attributes are present, is referred to as the **curse of dimensionality**. KNN is especially **sensitive** to this problem.

Weighting the Features

- If we'd like to increase or decrease the impact of some features we can apply weighting schemes.
- $||L||_2 = \sqrt[2]{z_1|x_{1,1} - x_{2,1}|^2 + \cdots z_d|x_{1,d} - x_{2,d}|^2}$
- If some features are more important, we can increase z. If some features are irrelevant, we can decrease z.
- This corresponds to stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes.
- In a drastic case one can eliminate the least relevant attributes by setting their weights to zero.
- The amount by which each axis should be stretched can be determined automatically using gradient decent and cross-validation.

Weighting the Features

KNN in Python

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import make_column_selector as selector
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

from sklearn import set_config #To display pipeline
set_config(display="diagram")
```

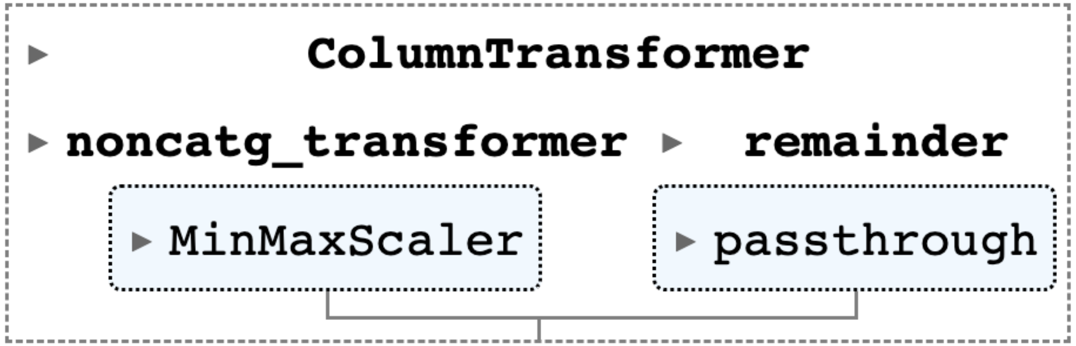
```
noncatg_pipeline = Pipeline(
    steps = [
        ("normalize", MinMaxScaler())
    ]
)
noncatg_pipeline
```

▸ **Pipeline**

▸ MinMaxScaler

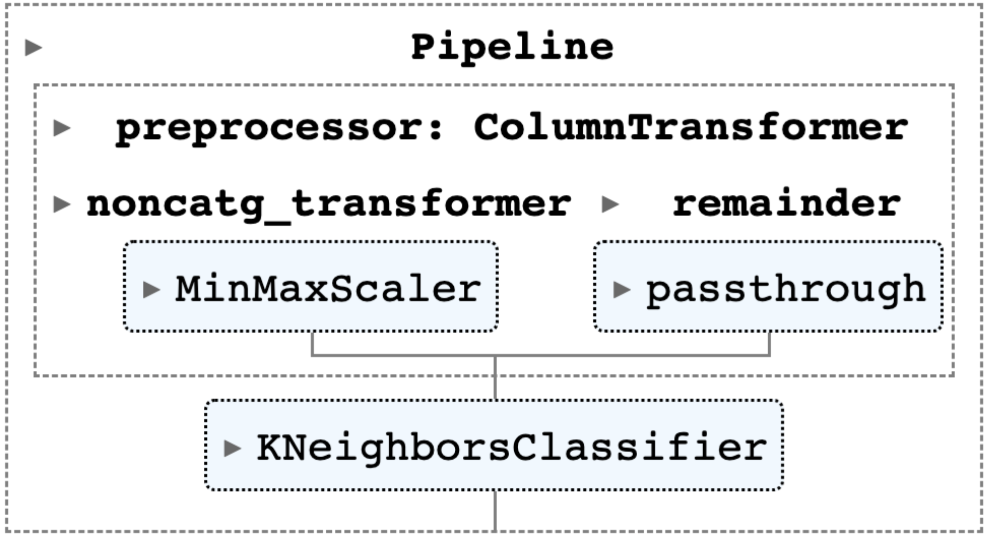
KNN in Python

```
columns_preprocessor = ColumnTransformer(
    transformers = [
        ('noncatg_transformer', noncatg_pipeline,\
         selector(dtype_exclude = "object")),
    ],
    remainder='passthrough'
    )
columns_preprocessor
```



KNN in Python

```
knn_model = Pipeline(
    steps = [
        ('preprocessor', columns_preprocessor),
        ( 'KNN', KNeighborsClassifier()),
    ]
)
knn_model
```




```
knn_model.fit(X_train, y_train)
```

Summary: Advantages and Disadvantages of KNN

Advantages

- There is no training!
- Can implicitly learn complex target functions easily.
- Has the ability to adapt its model to previously unseen data: instance-based learners may simply store a new instance or throw an old instance away.

Disadvantages

- Very slow at query time.
- Requires a lot of storage.
- Easily fooled by irrelevant features.

References

- Machine Learning by Tom Mitchell
- Learning from Data by Abu-Mustafa, Ismail, and Lin
- An Introduction to Statistical Learning with Applications in R by James, Witten, Hastie, and Tibshirani