



Machine Learning and Computational Intelligence Lecture 8

Sanjeeb Prasad Panday, PhD

Associate Professor

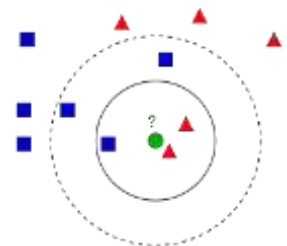
Dept. of Electronics and Computer Engineering

Director (ICTC)

IOE, TU

K-Nearest Neighbour

Tushar B. Kute,
<http://tusharkute.com>



What sort of Machine Learning?

- An idea that can be used for machine learning—as does another maxim involving poultry: "birds of a feather flock together."
- In other words, things that are alike are likely to have properties that are alike.
- We can use this principle to classify data by placing it in the category with the most similar, or "nearest" neighbors.

Nearest Neighbor Classification

- In a single sentence, nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of the most similar labeled examples. Despite the simplicity of this idea, nearest neighbor methods are extremely powerful. They have been used successfully for:
 - Computer vision applications, including optical character recognition and facial recognition in both still images and video
 - Predicting whether a person enjoys a movie which he/she has been recommended (as in the Netflix challenge)
 - Identifying patterns in genetic data, for use in detecting specific protein or diseases

The kNN Algorithm

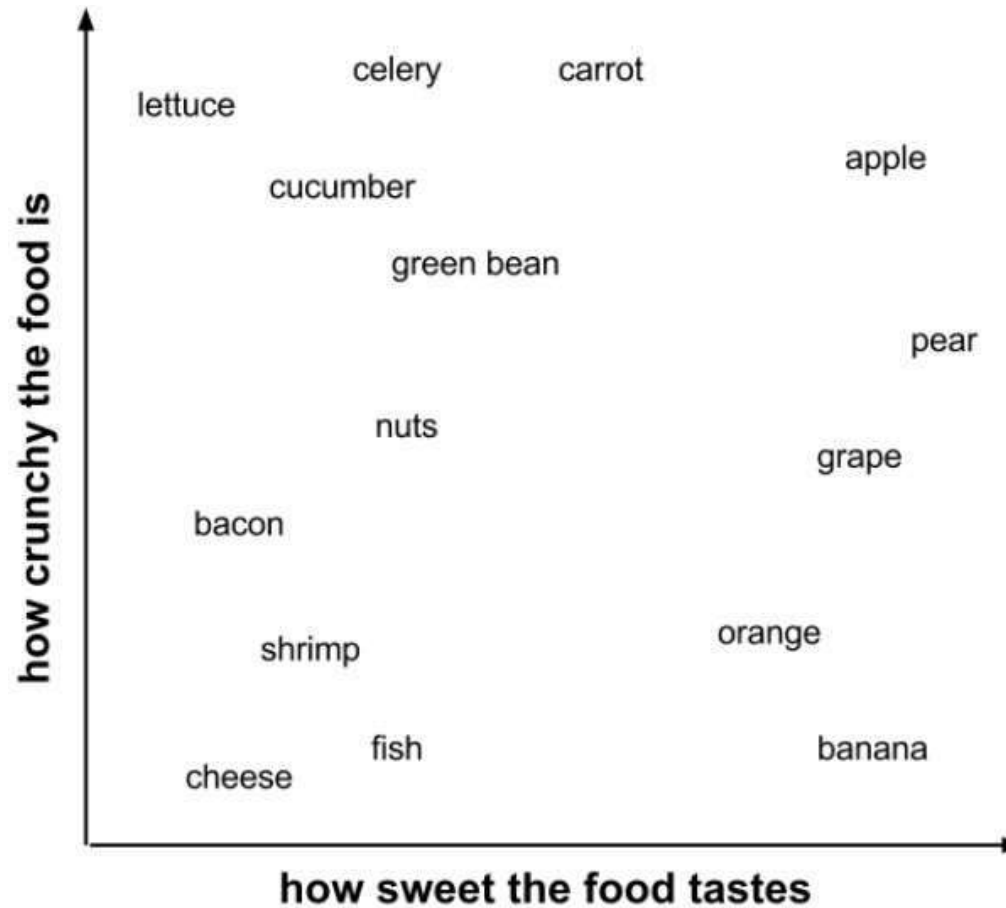
- The kNN algorithm begins with a training dataset made up of examples that are classified into several categories, as labeled by a nominal variable.
- Assume that we have a test dataset containing unlabeled examples that otherwise have the same features as the training data.
- For each record in the test dataset, kNN identifies k records in the training data that are the "nearest" in similarity, where k is an integer specified in advance.
- The unlabeled test instance is assigned the class of the majority of the k nearest neighbors

Example:

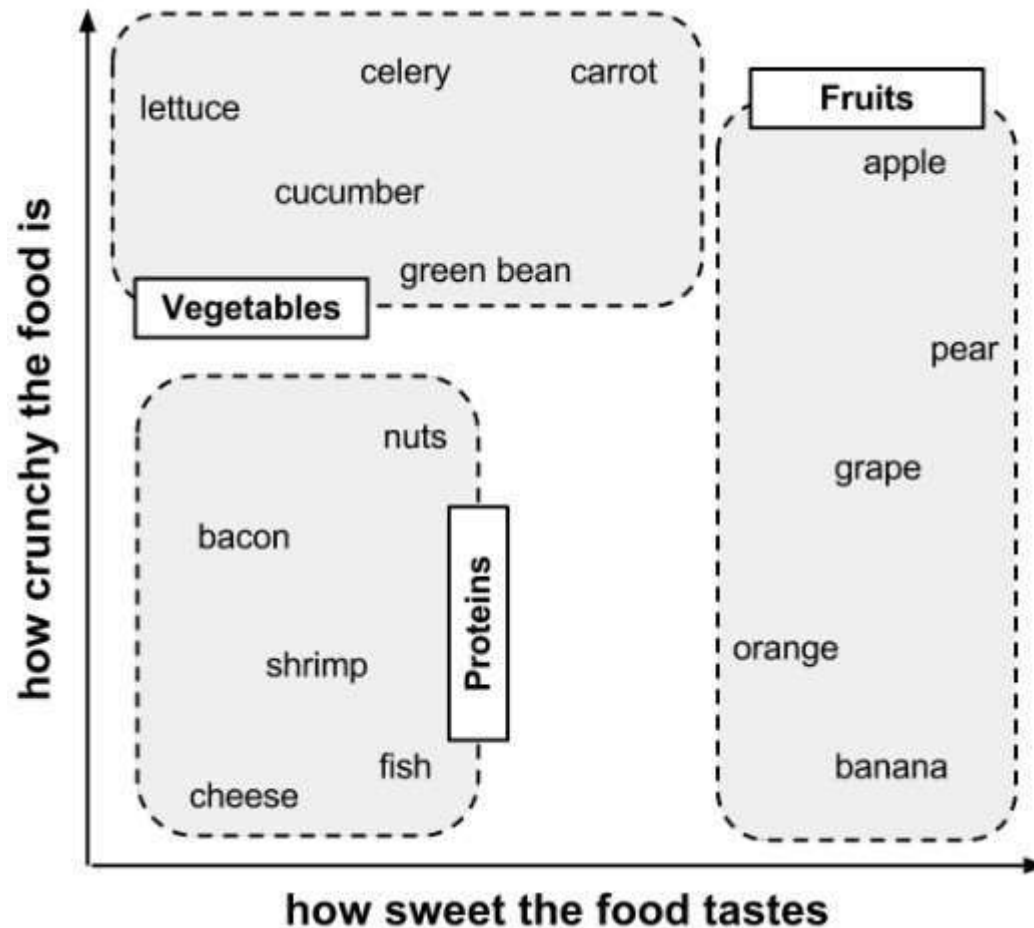
ingredient	sweetness	crunchiness	food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

Reference: Machine Learning with R, Brett Lantz, Packt Publishing

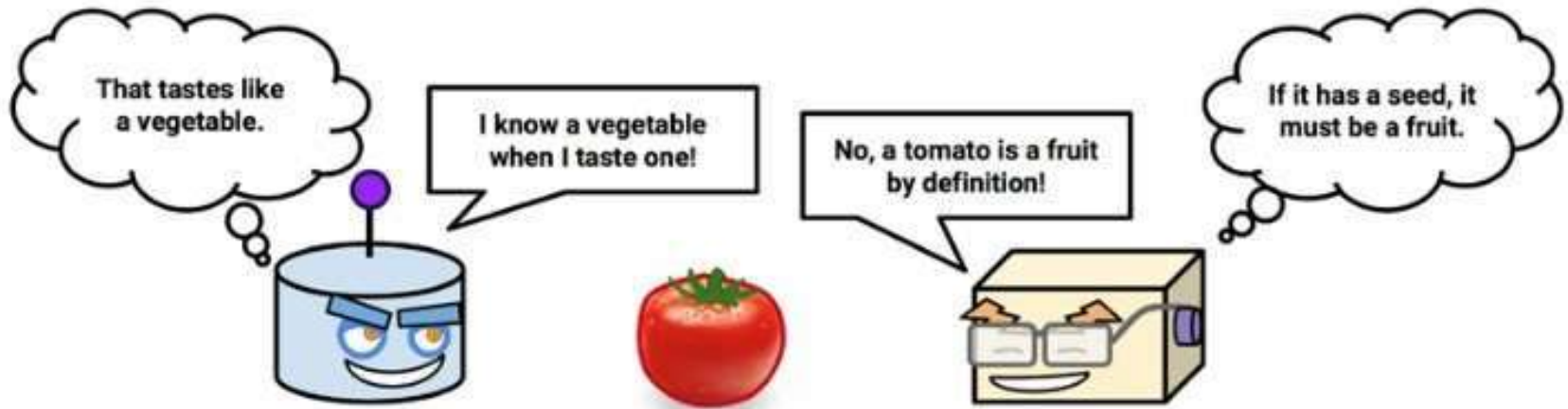
Example:



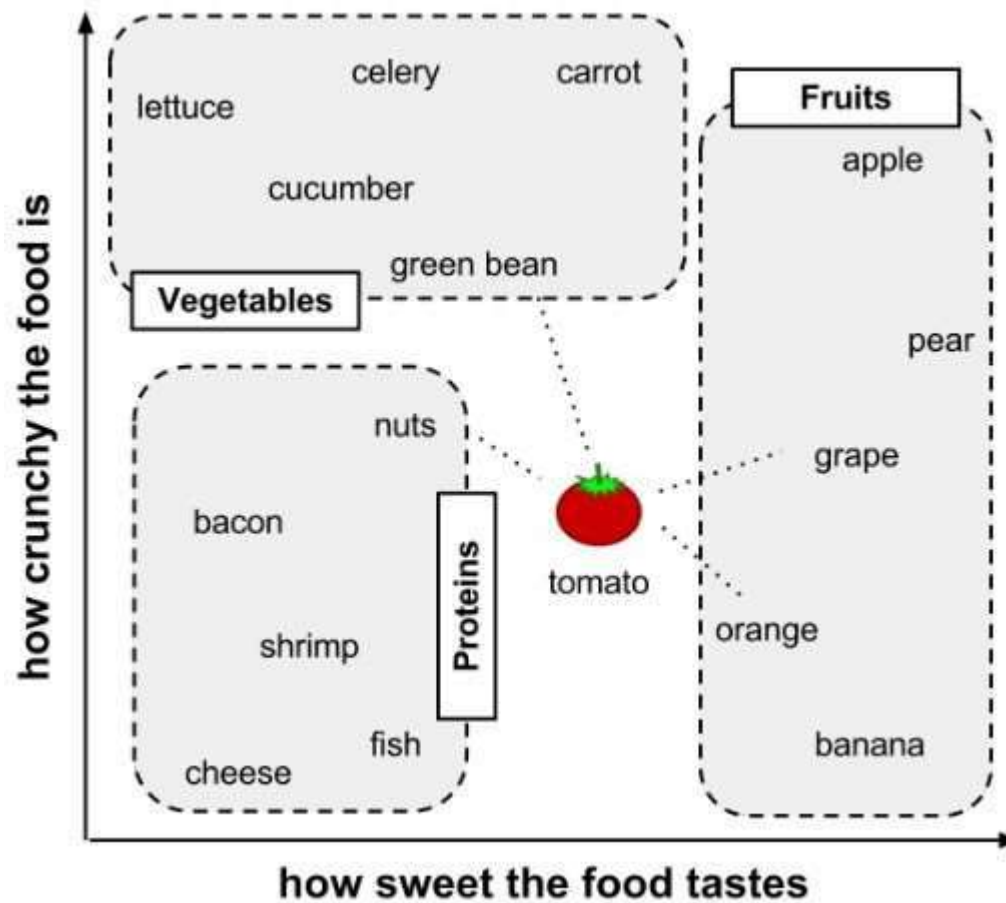
Example:



Classify me now!



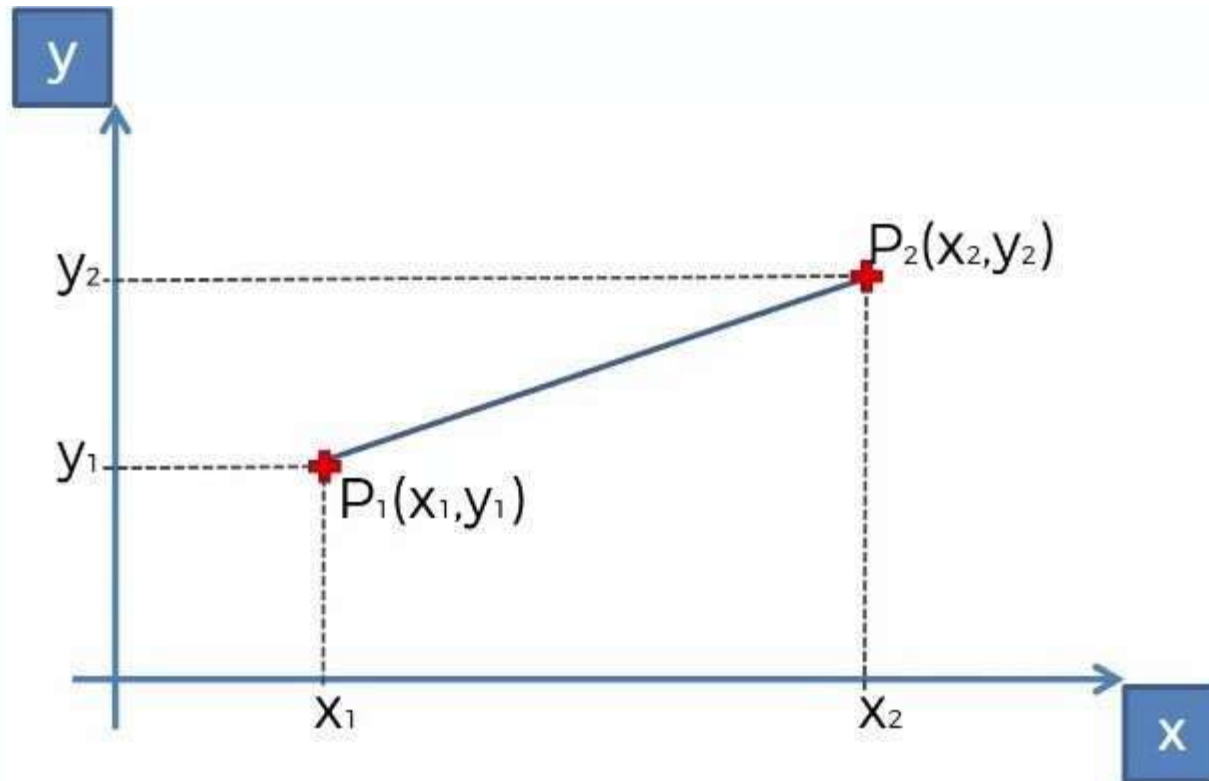
Example:



Calculating Distance

- Locating the tomato's nearest neighbors requires a distance function, or a formula that measures the similarity between two instances.
- There are many different ways to calculate distance.
- Traditionally, the kNN algorithm uses Euclidean distance, which is the distance one would measure if you could use a ruler to connect two points, illustrated in the previous figure by the dotted lines connecting the tomato to its neighbors.

Calculating Distance



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Reference: Super Data Science

Distance

- Euclidean distance is specified by the following formula, where p and q are the examples to be compared, each having n features. The term p_1 refers to the value of the first feature of example p , while q_1 refers to the value of the first feature of example q :

$$\text{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

- The distance formula involves comparing the values of each feature. For example, to calculate the distance between the tomato (sweetness = 6, crunchiness = 4), and the green bean (sweetness = 3, crunchiness = 7), we can use the formula as follows:

$$\text{dist}(\text{tomato}, \text{green bean}) = \sqrt{(6 - 3)^2 + (4 - 7)^2} = 4.2$$

Distance

Distance functions

Euclidean

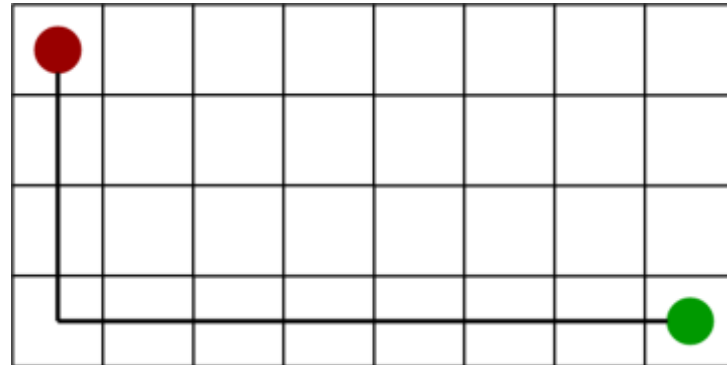
$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

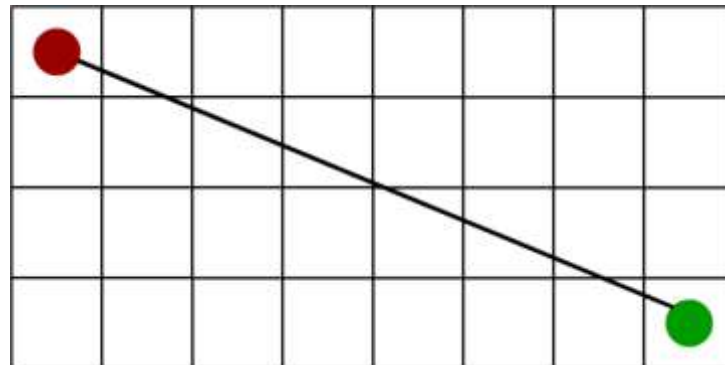
$$\sum_{i=1}^k |x_i - y_i|$$

Distance

Manhattan Distance



Euclidean Distance



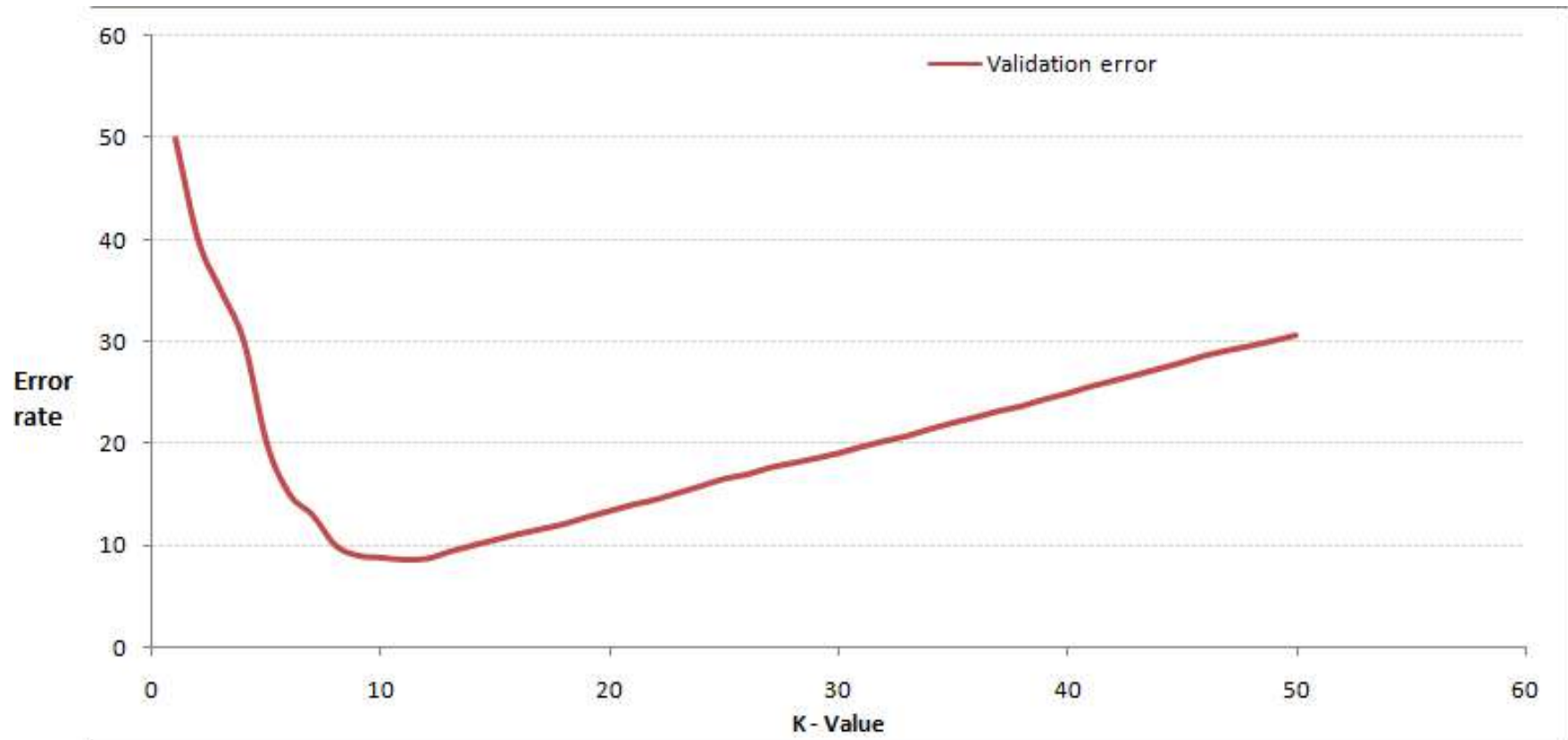
Closest Neighbors

ingredient	sweetness	crunchiness	food type	distance to the tomato
grape	8	5	fruit	$\text{sqrt}((6 - 8)^2 + (4 - 5)^2) = 2.2$
green bean	3	7	vegetable	$\text{sqrt}((6 - 3)^2 + (4 - 7)^2) = 4.2$
nuts	3	6	protein	$\text{sqrt}((6 - 3)^2 + (4 - 6)^2) = 3.6$
orange	7	3	fruit	$\text{sqrt}((6 - 7)^2 + (4 - 3)^2) = 1.4$

Choosing appropriate k

- Deciding how many neighbors to use for kNN determines how well the model will generalize to future data.
- The balance between overfitting and underfitting the training data is a problem known as the bias-variance tradeoff.
- Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner such that it runs the risk of ignoring small, but important patterns.

Choosing appropriate k



Choosing appropriate k

- In practice, choosing k depends on the difficulty of the concept to be learned and the number of records in the training data.
- Typically, k is set somewhere between 3 and 10. One common practice is to set k equal to the square root of the number of training examples.
- In the classifier, we might set $k = 4$, because there were 15 example ingredients in the training data and the square root of 15 is 3.87.

Min-Max normalization

- The traditional method of rescaling features for kNN is min-max normalization.
- This process transforms a feature such that all of its values fall in a range between 0 and 1. The formula for normalizing a feature is as follows. Essentially, the formula subtracts the minimum of feature X from each value and divides by the range of X :

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

- Normalized feature values can be interpreted as indicating how far, from 0 percent to 100 percent, the original value fell along the range between the original minimum and maximum.

The Lazy Learning

- Using the strict definition of learning, a lazy learner is **not** really learning anything.
- Instead, it merely stores the training data in it. This allows the training phase to occur very rapidly, with a potential downside being that the process of making predictions tends to be relatively slow.
- Due to the heavy reliance on the training instances, lazy learning is also known as **instance-based learning** or **rote learning**.

Few Lazy Learning Algorithms

- K Nearest Neighbors
- Local Regression
- Lazy Naive Bayes

The kNN Algorithm

Strengths	Weaknesses
<ul style="list-style-type: none">• Simple and effective• Makes no assumptions about the underlying data distribution• Fast training phase	<ul style="list-style-type: none">• Does not produce a model, which limits the ability to find novel insights in relationships among features• Slow classification phase• Requires a large amount of memory• Nominal features and missing data require additional processing

Python Packages needed

- pandas
 - Data Analytics
- numpy
 - Numerical Computing
- matplotlib.pyplot
 - Plotting graphs
- sklearn
 - Classification and Regression Classes

Sample Application

```
X = [[10,9],[1,4],[10,1],[7,10],[3,10],[1,1]]
# sweetness and crunchiness
# apple, bacon, banana, carrot, celery, cheese
y = ['fruit','protein','fruit','vegetable',
     'vegetable','protein']

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)

classifier.fit(X, y)

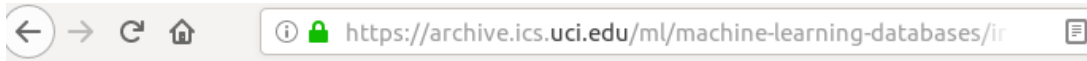
tomato = [[6,4]]
print classifier.predict(tomato)

carrot = [[4,9]]
print classifier.predict(carrot)
```

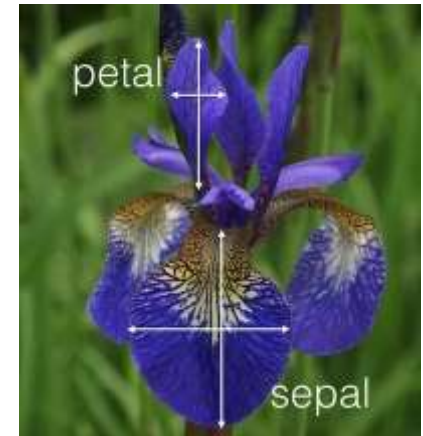
KNN – Classification : Dataset

- The best small project to start with on a new tool is the classification of iris flowers (e.g. the iris dataset).
- This is a good project because it is so well understood.
 - Attributes are numeric so you have to figure out how to load and handle data.
 - It is a classification problem, allowing you to practice with perhaps an easier type of supervised learning algorithm.
 - It is a multi-class classification problem (multi-nominal) that may require some specialized handling.
 - It only has 4 attributes and 150 rows, meaning it is small and easily fits into memory (and a screen or A4 page).
 - All of the numeric attributes are in the same units and the same scale, not requiring any special scaling or transforms to get started.

Dataset



```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
```



Iris Versicolor

Iris Setosa

Iris Virginica

KNN - Classification : Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# URL to read the dataset
url = "https://archive.ics.uci.edu/ml/
machine-learning-databases/iris/iris.data"

# Assign column names to the dataset
names = ['sepal-length', 'sepal-width',
'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names)

print dataset.head()
```

Pre-processing

```
# Input variables
```

```
X = dataset.iloc[:, :-1].values
```

```
# Output variable
```

```
y = dataset.iloc[:, 4].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test =  
train_test_split(X, y, test_size=0.20)
```

```
# Feature scaling
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

```
X_train = scaler.transform(X_train)
```

```
X_test = scaler.transform(X_test)
```


Characterize

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

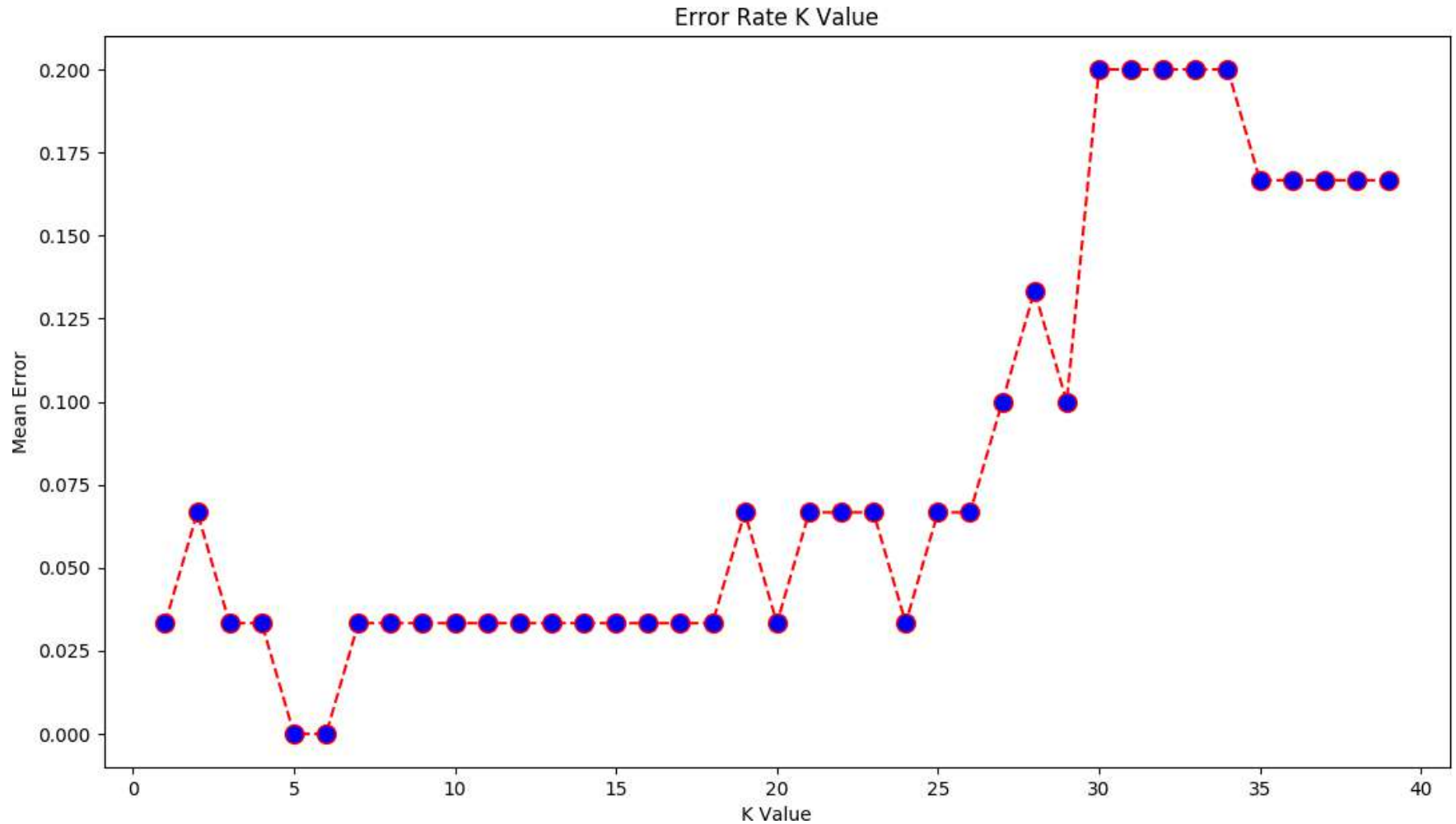
```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
print confusion_matrix(y_test, y_pred)
print classification_report(y_test, y_pred)
print accuracy_score(y_test, y_pred) * 100
```

Finding error with changed k

```
error = []  
# Calculating error for K values between 1 and 40  
for i in range(1, 40):  
    knn = KNeighborsClassifier(n_neighbors=i)  
    knn.fit(X_train, y_train)  
    pred_i = knn.predict(X_test)  
    error.append(np.mean(pred_i != y_test))  
  
plt.figure(figsize=(12, 6))  
plt.plot(range(1, 40), error, color='red', linestyle='dashed',  
marker='o', markerfacecolor='blue', markersize=10)  
plt.title('Error Rate K Value')  
plt.xlabel('K Value')  
plt.ylabel('Mean Error')  
plt.show()
```

Error rate



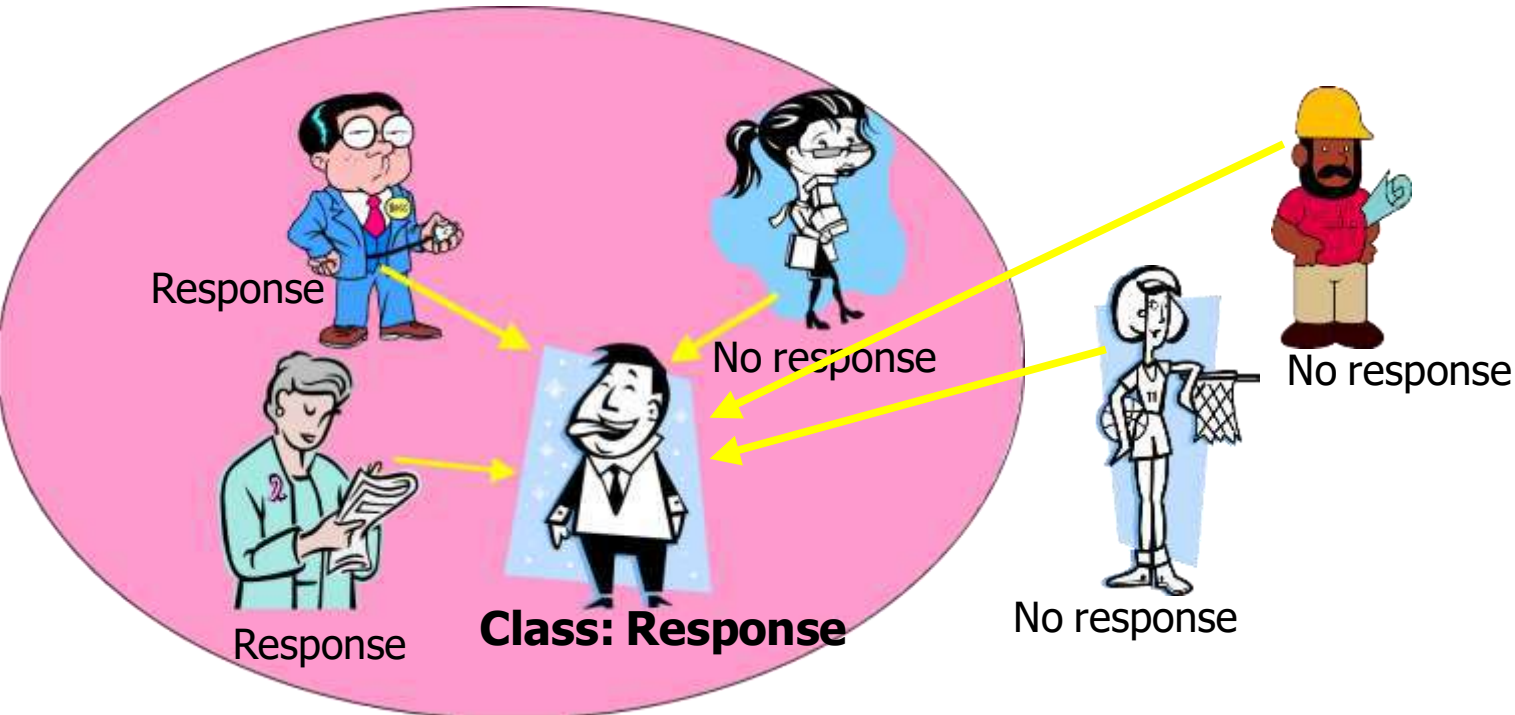
K-Nearest Neighbor Classifiers

- Learning by analogy:
- Tell me who your friends are and I'll tell you who you are
- A new example is assigned to the most common class among the (K) examples that are most similar to it.



K-Nearest Neighbor Algorithm

- To determine the class of a new example E:
 - Calculate the distance between E and all examples in the training set
 - Select K-nearest examples to E in the training set
 - Assign E to the most common class among its K-nearest neighbors



Distance Between Neighbors

- Each example is represented with a set of numerical attributes



John:
Age=35
Income=95K
No. of credit cards=3



Rachel:
Age=41
Income=215K
No. of credit cards=2

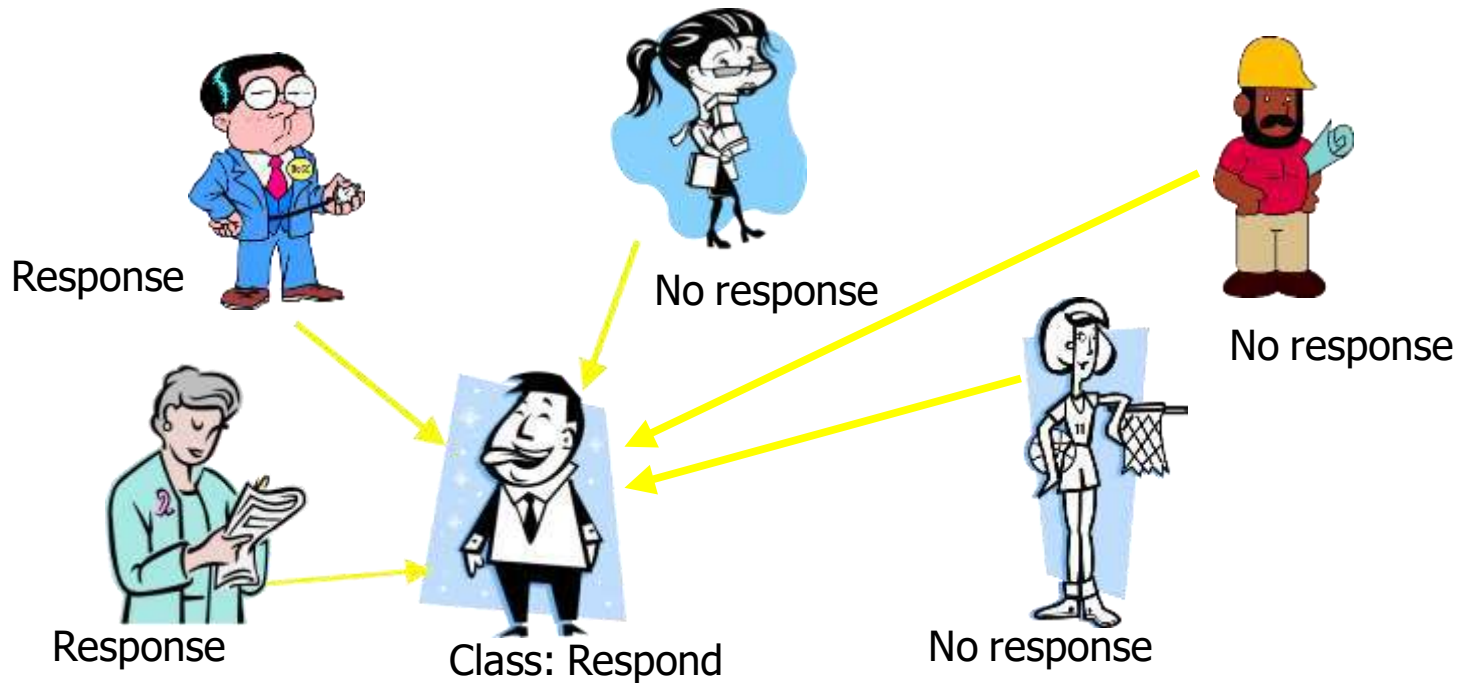
- “Closeness” is defined in terms of the Euclidean distance between two examples.
 - The Euclidean distance between $X=(x_1, x_2, x_3, \dots, x_n)$ and $Y=(y_1, y_2, y_3, \dots, y_n)$ is defined as:

$$D(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$







- Distance (John, Rachel)=sqrt $[(35-41)^2 + (95K-215K)^2 + (3-2)^2]$

K-Nearest Neighbor: Instance Based Learning




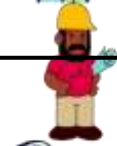


- No model is built: Store all training examples
- Any processing is delayed until a new instance must be classified.



Example : 3-Nearest Neighbors

Customer	Age	Income	No. credit cards	Response
John 	35	35K	3	No
Rachel 	22	50K	2	Yes
Hannah 	63	200K	1	No
Tom 	59	170K	1	No
Nellie 	25	40K	4	Yes
David 	37	50K	2	?

Example: Distance from David

Customer	Age	Income (K)	No. cards	Response	Distance from David
John 	35	35	3	No	$\text{sqrt} [(35-37)^2 + (35-50)^2 + (3-2)^2] = 15.16$
Rachel 	22	50	2	Yes	$\text{sqrt} [(22-37)^2 + (50-50)^2 + (2-2)^2] = 15$
Hannah 	63	200	1	No	$\text{sqrt} [(63-37)^2 + (200-50)^2 + (1-2)^2] = 152.23$
Tom 	59	170	1	No	$\text{sqrt} [(59-37)^2 + (170-50)^2 + (1-2)^2] = 122$
Nellie 	25	40	4	Yes	$\text{sqrt} [(25-37)^2 + (40-50)^2 + (4-2)^2] = 15.74$
David 	37	50	2	Yes	

K-Nearest Neighbor Classifier

- Strengths

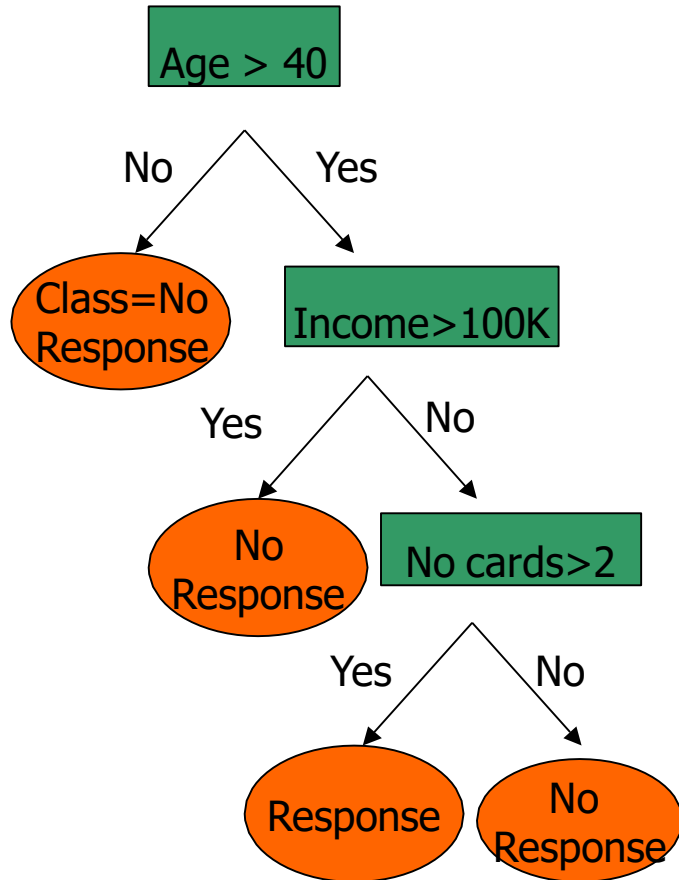
- Simple to implement and use
- Comprehensible – easy to explain prediction
- Robust to noisy data by averaging k-nearest neighbors.
- Some appealing applications

- Weaknesses

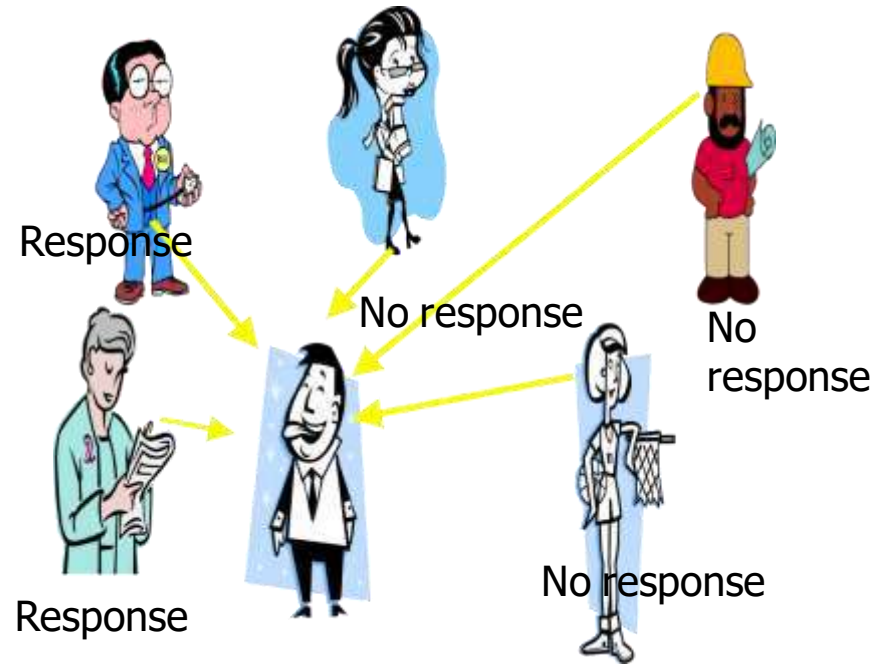
- Need a lot of space to store all examples.
- Takes more time to classify a new example than with a model (need to calculate and compare distance from new example to all other examples).

K-Nearest Neighbor Classifier

Classification Tree Modes



K-Nearest Neighbors



Class: Response

Strengths and Weaknesses K-Nearest Neighbor Classifier



John:

Age=35

Income=95K

No. of credit cards=3



Rachel:

Age=41

Income=215K

No. of credit cards=2

Distance (John, Rachel) = $\sqrt{(35-41)^2 + (95,000-215,000)^2 + (3-2)^2}$

- Distance between neighbors could be dominated by some attributes with relatively large numbers (e.g., income in our example). Important to normalize some features (e.g., map numbers to numbers between 0-1)







Example: Income

Highest income = 500K

Davis's income is normalized to $95/500$, Rachel income is normalized to $215/500$, etc.)

Strengths and Weaknesses K-Nearest Neighbor Classifier

Normalization of Variables

Customer	Age	Income (K)	No. cards	Response
John 	$55/63 = 0.55$	$35/200 = 0.175$	$\frac{3}{4} = 0.75$	No
Rachel 	$22/63 = 0.34$	$50/200 = 0.25$	$2/4 = 0.5$	Yes
Hannah 	$63/63 = 1$	$200/200 = 1$	$\frac{1}{4} = 0.25$	No
Tom 	$59/63 = 0.93$	$170/200 = 0.85$	$\frac{1}{4} = 0.25$	No
Nellie 	$25/63 = 0.39$	$40/200 = 0.2$	$4/4 = 1$	Yes
David 	$37/63 = 0.58$	$50/200 = 0.25$	$2/4 = 0.5$	Yes

Strengths and Weaknesses K-Nearest Neighbor Classifier

- Distance works naturally with numerical attributes

$$D(\text{Rachel\&Johm}) = \text{sqrt} [(35-37)^2 + (35-50)^2 + (3-2)^2] = 15.16$$

What if we have nominal attributes?

Example: married

Customer	Married	Income (K)	No. cards	Response
John	Yes	35	3	No
Rachel	No	50	2	Yes
Hannah	No	200	1	No
Tom	Yes	170	1	No
Nellie	No	40	4	Yes
David	Yes	50	2	

Issues regarding classification (1): Data Preparation

- **Data cleaning**
 - **Preprocess data in order to reduce noise and handle missing values**
- **Relevance analysis (feature selection)**
 - **Remove the irrelevant or redundant attributes**
- **Data transformation**
 - **Generalize and/or normalize data**

Issues regarding classification (2): Evaluating Classification Methods

- Predictive accuracy
- Speed and scalability
 - time to construct the model
 - time to use the model
- Robustness
 - handling noise and missing values
- Scalability
 - efficiency in disk-resident databases
- Interpretability:
 - understanding and insight provided by the model
- Goodness of rules
 - decision tree size
 - compactness of classification rules

Conclusions

- K-Nearest Neighbor Classifier is Learning by analogy
- K-Nearest Neighbor algorithm:
 - To determine the class of a new example E:
 - Calculate the distance between E and all examples in the training set
 - Select K-nearest examples to E in the training set
 - Assign E to the most common class among its K-nearest neighbors
- **Strengths**
 - Simple to implement and use
 - Comprehensible – easy to explain prediction
 - Robust to noisy data by averaging k-nearest neighbors.
 - Some appealing applications
- **Weaknesses**
 - Need a lot of space to store all examples.
 - Takes more time to classify a new example than with a model (need to calculate and compare distance from new example to all other examples).

KNN Classifier Solved Example - 1

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Versicolor
7.3	2.9	Virginica
6.0	2.7	Versicolor
5.8	2.8	Virginica
6.3	2.3	Versicolor
5.1	2.5	Versicolor
6.3	2.5	Versicolor
5.5	2.4	Versicolor

Sepal Length	Sepal Width	Species
5.2	3.1	?

Step 1: Find Distance

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(x - a)^2 + (y - b)^2}$$

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(5.2 - 5.3)^2 + (3.1 - 3.7)^2}$$

$$\text{Distance (Sepal Length, Sepal Width)} = 0.608$$

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

KNN Classifier Solved Example - 1

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Versicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Versicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Versicolor	1.36	12
5.1	2.5	Versicolor	0.60	4
6.3	2.5	Versicolor	1.25	11
5.5	2.4	Versicolor	0.75	7

Step 2: Find Rank

KNN Classifier Solved Example - 1

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Versicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Versicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Versicolor	1.36	12
5.1	2.5	Versicolor	0.60	4
6.3	2.5	Versicolor	1.25	11
5.5	2.4	Versicolor	0.75	7

Step 3: Find the Nearest Neighbor

If $k = 1$ – Setosa

If $k = 2$ – Setosa

If $k = 5$ – Setosa

K-Nearest Neighbors Algorithm Solved Example

- The "Restaurant A" sells burger with optional flavors: Pepper, Ginger and Chilly.
- Every day this week you have tried a burger (A to E) and kept a record of which you liked.
- Using Hamming distance, show how the 3NN classifier with majority voting would classify

{ pepper: false, ginger: true, chilly : true}

K-Nearest Neighbors Algorithm Solved Example

	Pepper	Ginger	Chilly	Liked
A	True	True	True	False
B	True	False	False	True
C	False	True	True	False
D	False	True	False	True
E	True	False	False	True

New Example - Q: pepper: false, ginger: true, chilly : true

K-Nearest Neighbors Algorithm Solved Example - 3

- But How to calculate the distance for attributes with nominal or categorical values.
- Here we can use Hamming distance to find the distance between the categorical values
- Let x_1 and x_2 are the attribute values of two instances.
- Then, in hamming distance, if the categorical values are same or matching that is x_1 is same as x_2 then distance is 0, otherwise 1.
- For example,
- If value of **x_1 is blue** and **x_2 is also blue** then the distance between x_1 and x_2 is **0**.
- If value of **x_1 is blue** and **x_2 is red** then the distance between x_1 and x_2 is **1**.

K-Nearest Neighbors Algorithm Solved Example

	Pepper	Ginger	Chilly	Liked	Distance
A	True	True	True	False	$1 + 0 + 0 = 1$
B	True	False	False	True	$1 + 1 + 1 = 3$
C	False	True	True	False	$0 + 0 + 0 = 0$
D	False	True	False	True	$0 + 0 + 1 = 1$
E	True	False	False	True	$1 + 1 + 1 = 3$

New Example - Q: pepper: false, ginger: true, chilly : true

Use Hamming Distance and

find the distance from Query Example (Q) to training examples (A-E)

K-Nearest Neighbors Algorithm Solved Example -

	Pepper	Ginger	Chilly	Liked	Distance	3NN
A	True	True	True	False	$1 + 0 + 0 = 1$	2
B	True	False	False	True	$1 + 1 + 1 = 3$	
C	False	True	True	False	$0 + 0 + 0 = 0$	1
D	False	True	False	True	$0 + 0 + 1 = 1$	2
E	True	False	False	True	$1 + 1 + 1 = 3$	

New Example - Q: pepper: false, ginger: true, chilly : true

New Example - Q is classified as **FALSE** (as two nearest neighbors are FALSE and only one is TRUE)



**Thank you for your
attention**