Lecture 3: K-Nearest Neighbors

COMP90049
Introduction to Machine Learning

Semester 2, 2022

Lida Rashidi, CIS

Acknowledgement: Lea Frermann

Roadmap

Last time... Machine Learning concepts

- data, features, classes
- models, training
- practical considerations

Today... Our first machine learning algorithm

- K-nearest neighbors
- Application to classification
- Application to regression

Also: the topic of your **first assignment**!

- Released on Canvas on Wed 3rd August at 7pm!
- Questions: assignment1 discussion board (don't share solutions!)

Introduction

K-Nearest Neighbors: Example

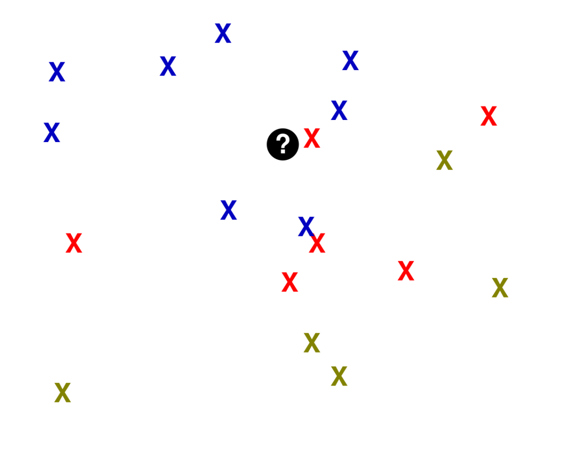
Your ’photographic memory’ of all handwritten digits you’ve ever seen:

K-Nearest Neighbors: Example

Your 'photographic memory' of all handwritten digits you've ever seen:

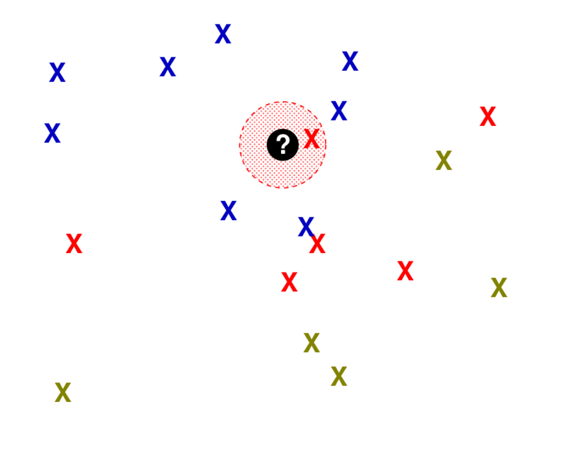
Given a new drawing, determine the digit by comparing it to all digits in your 'memory'.

K-Nearest Neighbors: Visualization



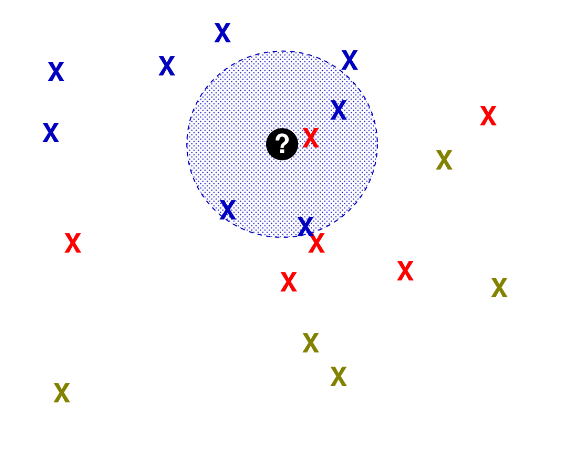
K nearest neighbors = K closest stored data points

K-Nearest Neighbors: Visualization



1 nearest neighbor = single closest stored data point

K-Nearest Neighbors: Visualization



4 nearest neighbors = 4 closest stored data points

K-Nearest Neighbors: Algorithm

Training

- Store all training examples

Testing

- Compute **distance** of test instance to all training data points
- Find the K closest training data points (*nearest neighbors*)
- Compute **target concept** of the test instance based on labels of the training instances

K-Nearest Neighbors: Target Concepts

KNN Classification

- Return the most common class label among neighbors
- Example: cat vs dog images; text classification; ...

KNN Regression

- Return the average value of among K nearest neighbors
- Example: housing price prediction;

Four problems

1. **How to represent each data point?**
2. How to measure the distance between data points?
3. What if the neighbors disagree?
4. How to select K?

Feature Vectors

A data set of 6 instances (a...f) with 4 features and a label

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no

THE UNIVERSITY OF MELBOURNE

Feature Vectors

A data set of 6 instances (a...f) with 4 features and a label

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no

We can represent each instance as a feature vector

$$\text{feature vector} = \begin{bmatrix} \text{Outlook} \\ \text{Temperature} \\ \text{Humidity} \\ \text{Windy} \end{bmatrix}$$

THE UNIVERSITY OF MELBOURNE

Feature (or attribute) Types

Recall, from last lecture?

1. Nominal
   - set of values with no intrinsic ordering
   - possibly *boolean*
2. Ordinal
   - explicitly ordered
3. Numerical
   - real-valued, often no upper bound, easily mathematical manipulatable
   - vector valued

Outline

1. How to represent each data point?
2. **How to measure the distance between data points?**
3. What if the neighbors disagree?
4. How to select K?

THE UNIVERSITY OF MELBOURNE

Comparing Nominal Feature Vectors

First, we convert the nominal features into numeric features.

instance	features			
	color	shape	taste	size
apple	red	round	sweet	–
banana	yellow	curved	sweet	–
cherry	red	round	sweet	small

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

Comparing Nominal Features: Hamming Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of differing elements in two ‘strings’ of equal length.

Comparing Nominal Features: Hamming Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of differing elements in two ‘strings’ of equal length.

$d(apple, banana) = 4$

Comparing Nominal Features: Simple Matching Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of matching features divided by the number of all features in the sample

$$d = 1 - \frac{k}{m}$$

- d: distance
- k: number of matching features
- m: total number of features

THE UNIVERSITY OF MELBOURNE

Comparing Nominal Features: Simple Matching Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of matching features divided by the number of all features in the sample

$$d = 1 - \frac{k}{m}$$

- d: distance
- k: number of matching features
- m: total number of features

$$d(apple, banana) = 1 - \frac{2}{6} = \frac{4}{6}$$

Comparing Nominal Feature Vectors: Jaccard Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

Jaccard *similarity* J: intersection of two **sets** divided by their union.

$$
\begin{aligned}
d &= 1 - J \\
&= 1 - \frac{|A \cap B|}{|A \cup B|} \\
&= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}
\end{aligned}
$$

Comparing Nominal Feature Vectors: Jaccard Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

Jaccard *similarity* J: intersection of two **sets** divided by their union.

$$
\begin{aligned}
d &= 1 - J \\
&= 1 - \frac{|A \cap B|}{|A \cup B|} \\
&= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}
\end{aligned}
$$

$$d(apple, banana) = 1 - \frac{1}{5} = \frac{4}{5}$$

Comparing Numerical Feature Vectors: Manhattan Distance

Manhattan Distance (or: L1 distance)

- Given two instances a and b, each with a set of numerical features, e.g.,
  $a = [2.0,\ 1.4,\ 4.6,\ 5.5]$
  $b = [1.0,\ 2.4,\ 6.6,\ 2.5]$
- Their distance d is the sum of absolute differences of each feature

$$d(a, b) = \sum_{i=1}^{m} |a_i - b_i| \tag{1}$$

Example

$$\begin{aligned} d(a, b) &= |2.0 - 1.0| + |1.4 - 2.4| + |4.6 - 6.6| + |5.5 - 2.5| \\ &= 1 + 1 + 2 + 3 \\ &= 7 \end{aligned}$$

Comparing Numerical Feature Vectors: Euclidean Distance

Euclidean Distance (or: L2 distance)

- Given two instances a and b, each with a set of numerical features, e.g.,
  $a = [2.0,\ 1.4,\ 4.6,\ 5.5]$
  $b = [1.0,\ 2.4,\ 6.6,\ 2.5]$
- Their distance d is the distance in Euclidean space (2-dimensional space). Defined as the squared root of the sum of squared differences of each feature

$$d(a,b) = \sqrt{\sum_{i=1}^{m}(a_i - b_i)^2} \tag{2}$$

Example

$$\begin{aligned} d(a,b) &= \sqrt{(2.0-1.0)^2 + (1.4-2.4)^2 + (4.6-6.6)^2 + (5.5-2.5)^2} \\ &= \sqrt{1+1+4+9} = \sqrt{15} \\ &= 3.87 \end{aligned}$$

Comparing Numerical Feature Vectors: Cosine distance

Cosine Distance

- Cosine similarity = cosine of angle between two vectors (= inner product of the *normalized* vectors)
- Cosine distance d: one minus cosine similarity

$$cos(a, b) = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

$$d(a, b) = 1 - cos(a, b)$$

- Cosine distance is **normalized by the magnitude** of both feature vectors, i.e., we can compare instances of different magnitude

$\rightarrow$ word counts: compare long vs short documents
$\rightarrow$ pixels: compare high vs low resolution images

Comparing Numerical Feature Vectors: Cosine distance

Example

$$cos(a,b) = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

$$d(a,b) = 1 - cos(a,b)$$

feature	doc1	doc2	doc3
word1	200	300	50
word2	300	200	40
word3	200	100	25

$$cos(doc1, doc2) = \frac{200 \times 300 + 300 \times 200 + 200 \times 100}{\sqrt{200^2 + 300^2 + 200^2}\sqrt{300^2 + 200^2 + 100^2}} = 0.93$$

$$d(doc1, doc2) = 0.07$$

$$cos(doc2, doc3) = \frac{300 \times 50 + 200 \times 40 + 100 \times 25}{\sqrt{300^2 + 200^2 + 100^2}\sqrt{50^2 + 40^2 + 25^2}} = 0.99$$

$$d(doc2, doc3) = 0.01$$

Comparing Ordinal Feature Vectors

Normalized Ranks

- sort values, and return a rank $r \in \{0 \ldots m\}$
- map ranks to evenly spaced values between 0 and 1

$$z = \frac{r}{m}$$

- compute a distance function for numeric features (e.g., Euclidean distance)

Example: Customer ratings

1. Sorted ratings: { -2: 😭, -1: ☹️, 0: 😐, 1: 🙂, 2: 😀 }
2. Ranks: { 0, 1, 2, 3, 4 }

feature	A	B
safety	0	2
comfortable	-2	1
convenient	-1	2

Comparing Ordinal Feature Vectors

Normalized Ranks

- sort values, and return a rank $r \in \{0...m\}$
- map ranks to evenly spaced values between 0 and 1

$$z = \frac{r}{m}$$

- compute a distance function for numeric features (e.g., Euclidean distance)

Example: Customer ratings

1. Sorted ratings: { -2: 😭, -1: 😢, 0: 😐, 1: 🙂, 2: 😃 }
2. Ranks: { 0, 1, 2, 3, 4 }

feature	A	B
safety	0	2
comfortable	-2	1
convenient	-1	2

→

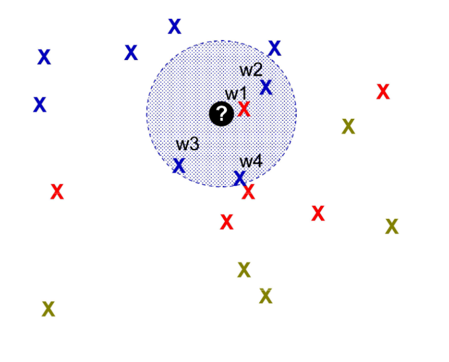
feature	A	B
safety	2/4	4/4
comfortable	0	3/4
convenient	1/4	4/4

Four problems

1. How to represent each data point?
2. How to measure the distance between data points?
3. **What if the neighbors disagree?**
4. How to select K?

Majority Voting

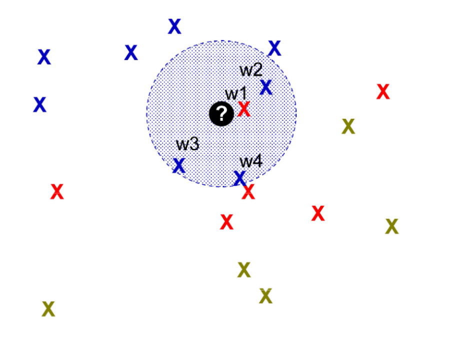
Equal weights (=majority vote)



Voting Example (k=4)

- $w_1 = w_2 = w_3 = w_4 = 1$

Equal weights (=majority vote)



Voting Example (k=4)

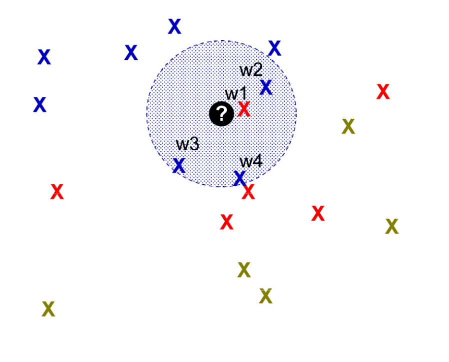
- $w_1 = w_2 = w_3 = w_4 = 1$
- red: 1
  blue: 1+1+1=3

Weighted KNN: Inverse Distance

Inverse Distance

$$w_j = \frac{1}{d_j + \epsilon}$$

with $\epsilon \approx 0$, e.g., $1e-10$



Voting Example (k=4)

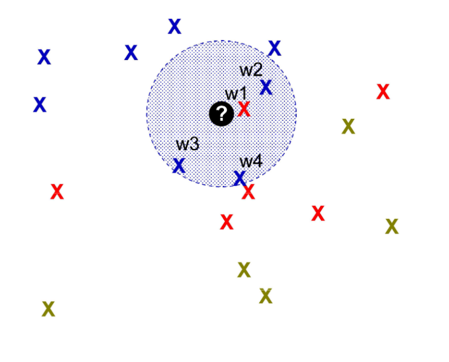
- $d_1 = 0$; $d_2 = 1$; $d_3 = d_4 = 1.5$
- $\epsilon = 1e-5$

Weighted KNN: Inverse Distance

Inverse Distance

$$w_j = \frac{1}{d_j + \epsilon}$$

with $\epsilon \approx 0$, e.g., $1e - 10$



Voting Example (k=4)

- $d_1 = 0$; $d_2 = 1$; $d_3 = d_4 = 1.5$
- $\epsilon = 1e - 5$

red: $\frac{1}{0+\epsilon} = 100000$

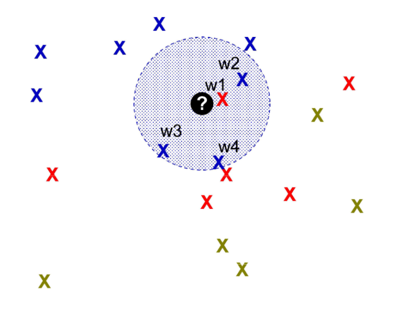
blue: $\frac{1}{1+\epsilon} + \frac{1}{1.5+\epsilon} + \frac{1}{1.5+\epsilon} = 1.0 + 0.67 + 0.67 = 2.34$

Weighted K-NN: Inverse Linear Distance

Inverse Linear distance

$$w_j = \frac{d_k - d_j}{d_k - d_1}$$

d_1 = min d among neighbors
d_k = max d among neighbors
d_j = distance of jth neighbor



Voting Example (k=4)

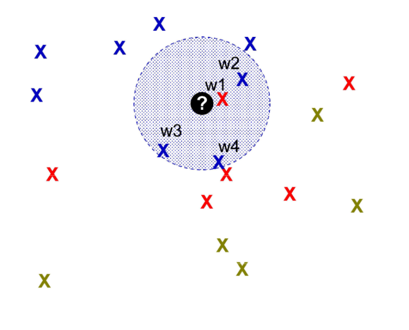
- $d_1 = 0$; $d_2 = 1$; $d_3 = d_4 = 1.5$

Weighted K-NN: Inverse Linear Distance

Inverse Linear distance

$$w_j = \frac{d_k - d_j}{d_k - d_1}$$

d_1 = min d among neighbors
d_k = max d among neighbors
d_j = distance of jth neighbor



Voting Example (k=4)

- $d_1 = 0$; $d_2 = 1$; $d_3 = d_4 = 1.5$

red: $\frac{1.5-0}{1.5-0} = 1$

blue: $\frac{1.5-1}{1.5-0} + \frac{1.5-1.5}{1.5-0} + \frac{1.5-1.5}{1.5-0} = 0.3 + 0 + 0 = 0.3$

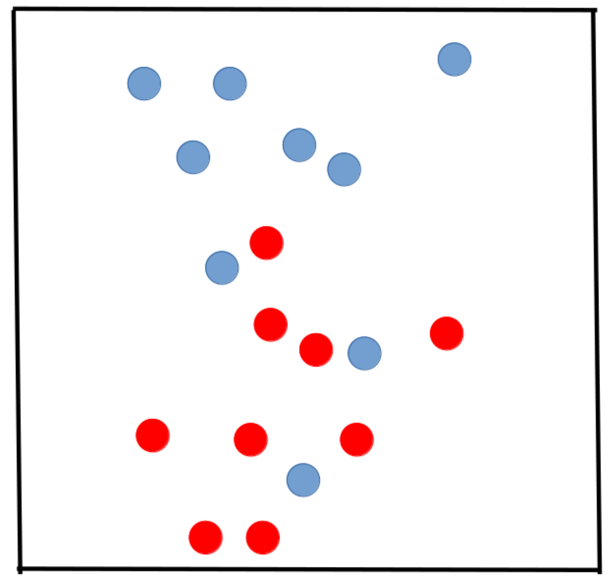
THE UNIVERSITY OF MELBOURNE

Four problems

1. How to represent each data point?
2. How to measure the distance between data points?
3. What if the neighbors disagree?
4. **How to select K?**

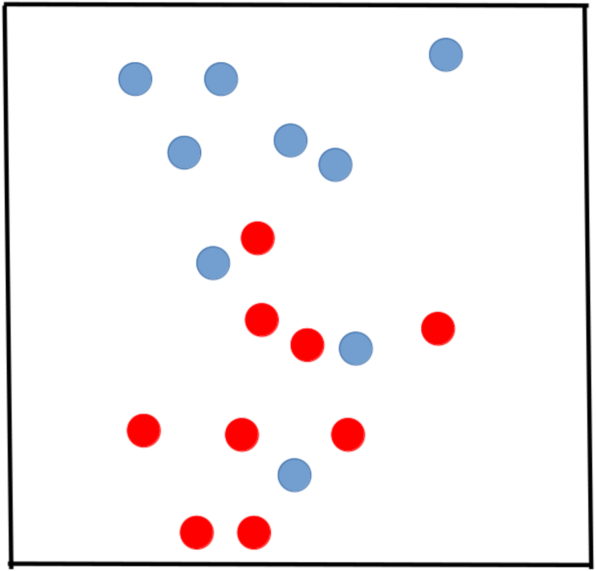
Selecting the value of K

K=1



Selecting the value of K

K=3



Selecting the value of K

Small K

- jagged decision boundary
- we capture noise
- lower classifier performance

Large K

- smooth decision boundary
- danger of grouping together unrelated classes
- also: lower classifier performance!
- what if $K == N$? (N=number of training instances)

Draw validation error:

Breaking Ties

What if more than K neighbors have the same (smallest) distance?

- select at random
- change the distance metric

What if two classes are equally likely given the current neighborhood?

- avoid an even K
- random tie breaking
- include $K + 1$th neighbor
- use class with highest prior probability

THE UNIVERSITY OF MELBOURNE

Quiz!

`pollev.com/comp90049`

Why K-NN?

Pros

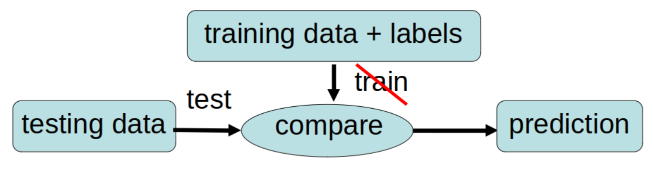
- Intuitive and simple
- No assumptions
- Supports classification and regression
- **No training**: new data $\rightarrow$ evolve and adapt immediately

Cons

- How to decide on best distance functions?
- How to combine multiple neighbors?
- How to select K?
- Expensive with large (or growing) data sets

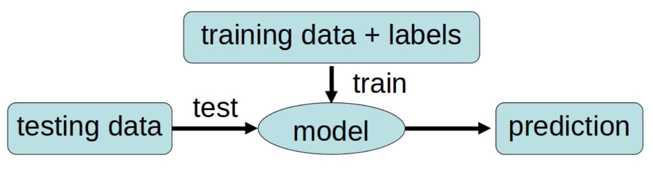
Lazy vs Eager Learning

Lazy Learning (also known as **Instance-based Learning**)



- **store** the training data
- **fixed** distance function
- **fixed** prediction rule (majority, weighting, ...)
- **compare** test instances with stored instances
- **no learning**

Eager Learning



- **train** a **model** using labelled training instances
- the model will **generalize** from seen data to unseen data
- use the model to **predict** labels for test instances
- we will look at a variety of **eager** models and their learning algorithms over the next couple of weeks

Summary

Today... Our first machine learning algorithm

- K-nearest neighbors
- Application to classification
- Application to regression

Also: the topic of your **first assignment**!

Next: Probabilities (recap) and probabilistic modeling

Further Reading

- *Data Mining: Concepts and Techniques*, 2nd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006. Chapter 2, Chapter 9.5.
- *The elements of statistical learning*, 2nd ed., Trevor Hastie, Jerome Friedman and Robert Tibshirani. New York: Springer series in statistics, 2001. Chapter 2.3.2

THE UNIVERSITY OF MELBOURNE