Ciencia de Datos

Lecture 9:
Nearest Neighbour Methods

Dimosthenis Karatzas (dimos@cvc.uab.es)

K-NEAREST NEIGHBOUR ALGORITHM

A challenging problem

Where was this photo taken?

Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

A challenging problem

Where was this photo taken?

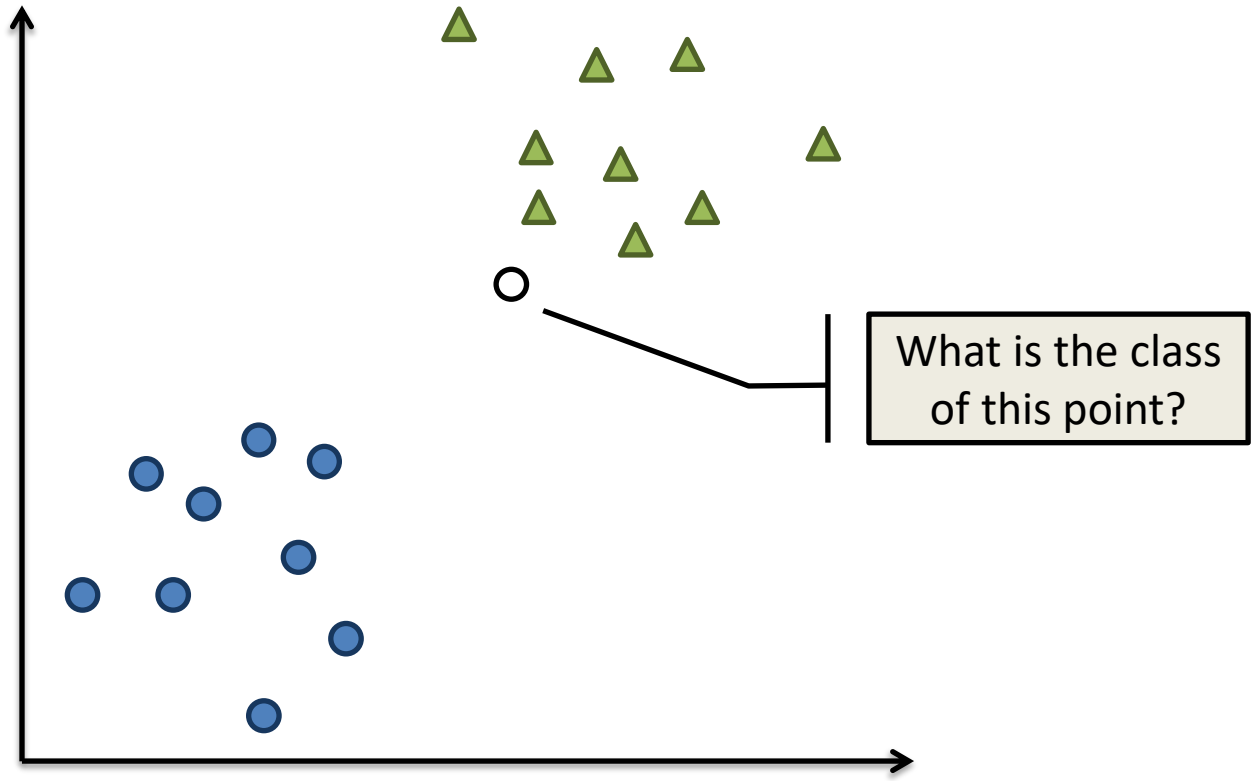
Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

A challenging problem

Where was this photo taken?

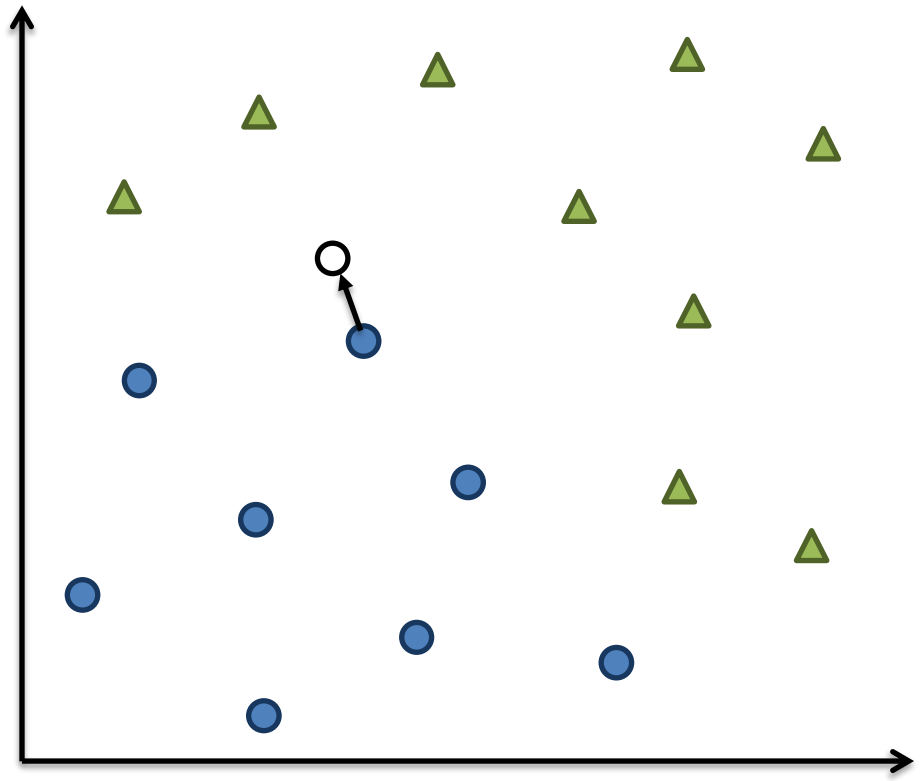
Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

Nearest Neighbour Algorithm



Intuition: nearby points are green, hence this seems to fall closer to the “green” territory than the “blue” territory. Nearby things should have the same class

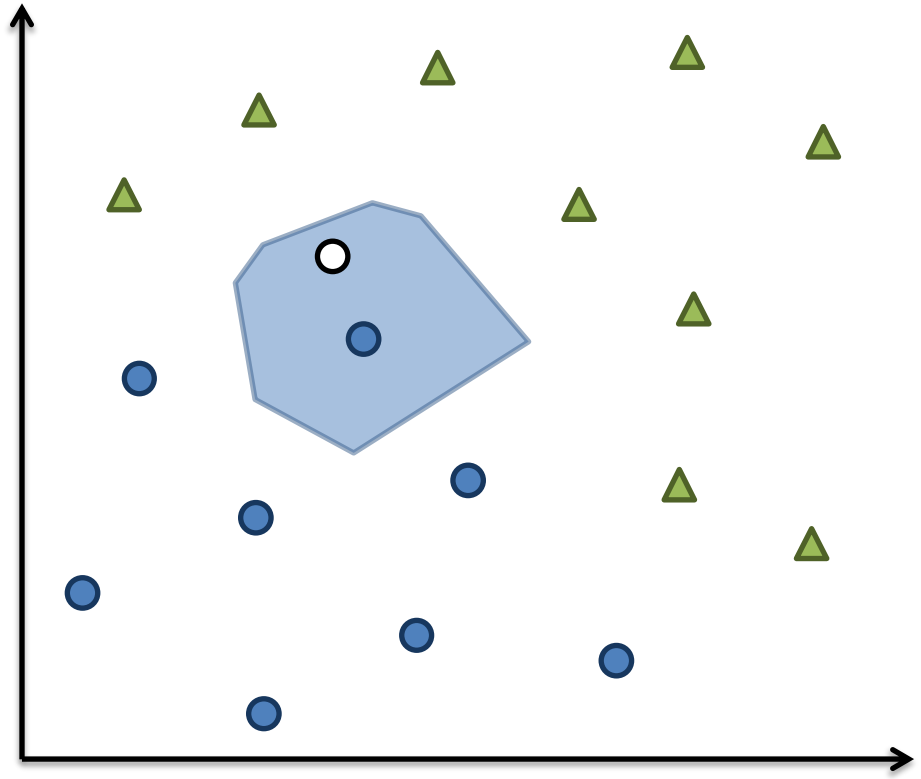
Nearest Neighbour Classification



Nearest Neighbour Classification algorithm:

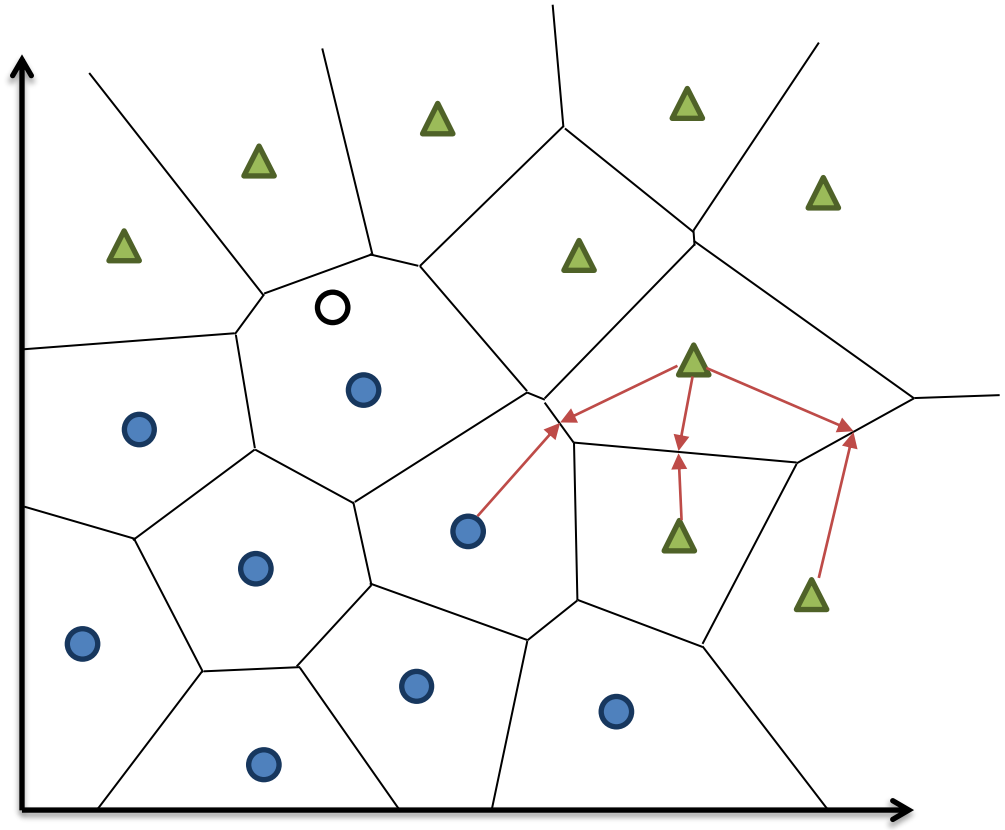
1. Find the most similar (closest) training example
2. Use its class as the prediction

Nearest Neighbour Algorithm



Each training sample defines a region around it where it is dominant. All the points within this region, are closer to this particular sample than they are to any other sample

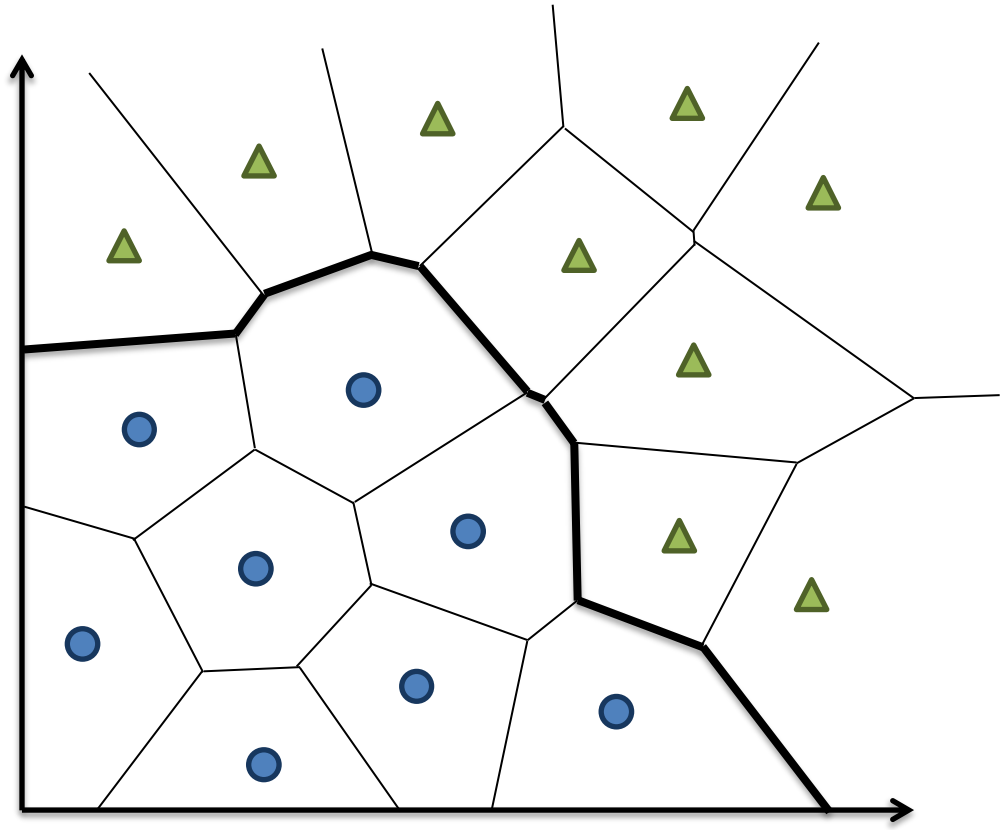
Nearest Neighbour Algorithm



This partition of the feature space into cells is called **Voronoi Tesselation**

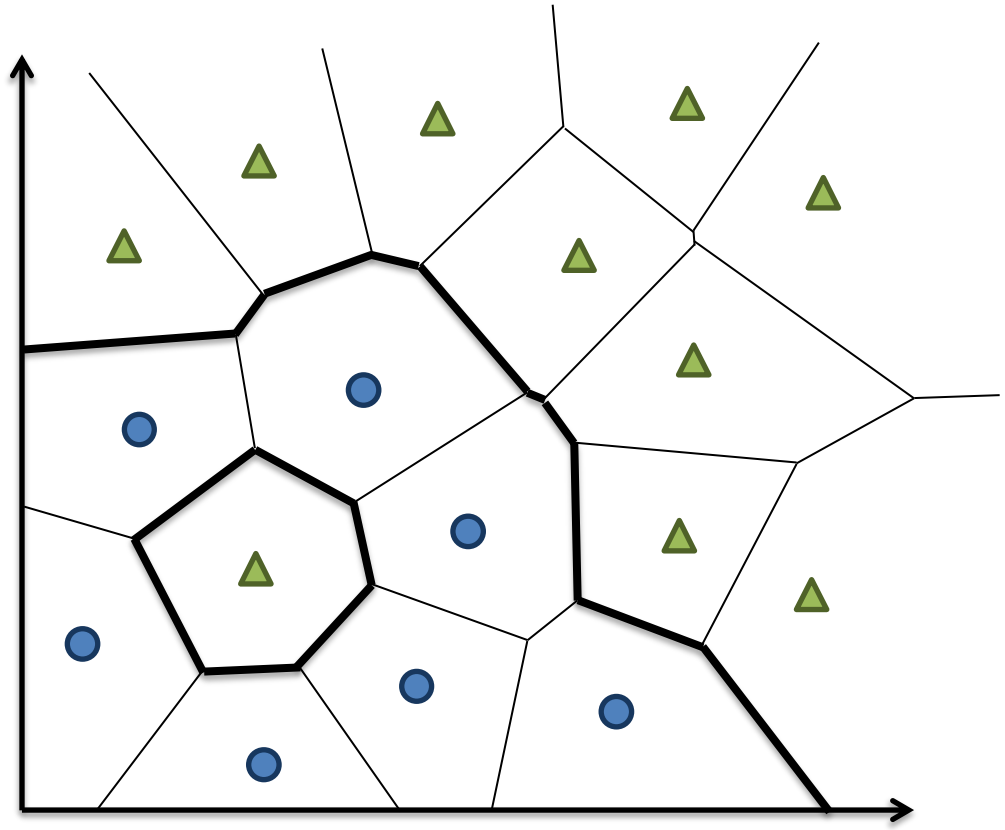
The **boundary** between two training examples is defined by the set of points that have the same distance to the two training examples

Nearest Neighbour Algorithm



The **decision boundary** of a nearest neighbour classifier comprises the set of boundary segments that separate neighbouring points that belong to different classes

Dealing with Outliers

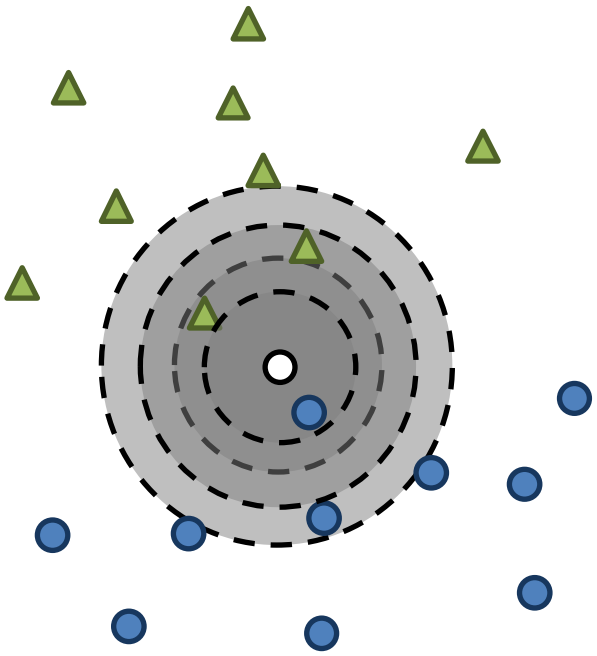


Outliers have significant effects on the decision boundary

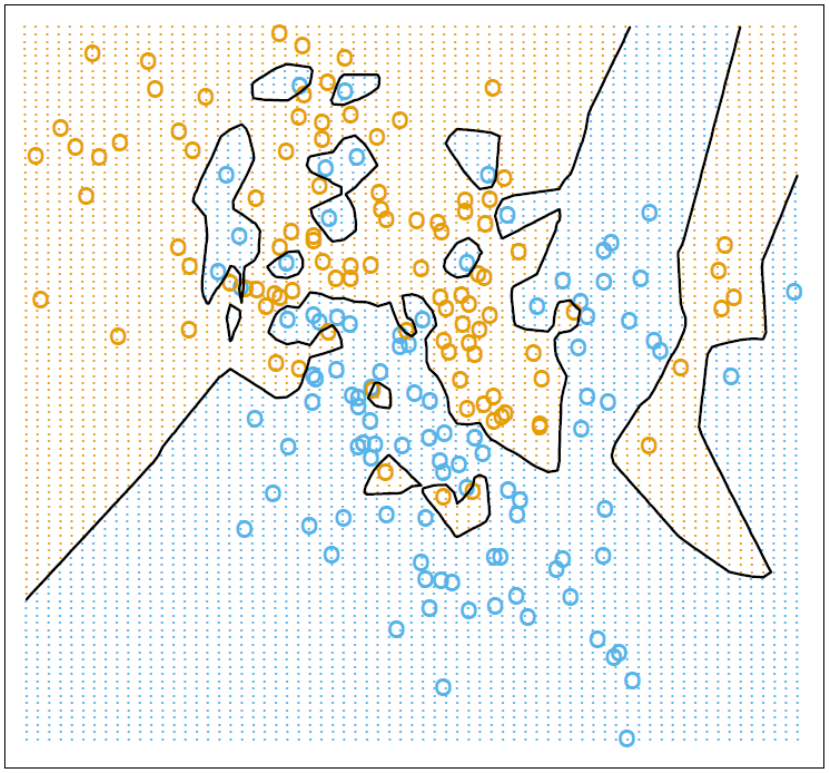
Intuition: we could use more than one nearest neighbour to make a decision. Vote, based on the class of k nearest neighbours

How many Neighbours

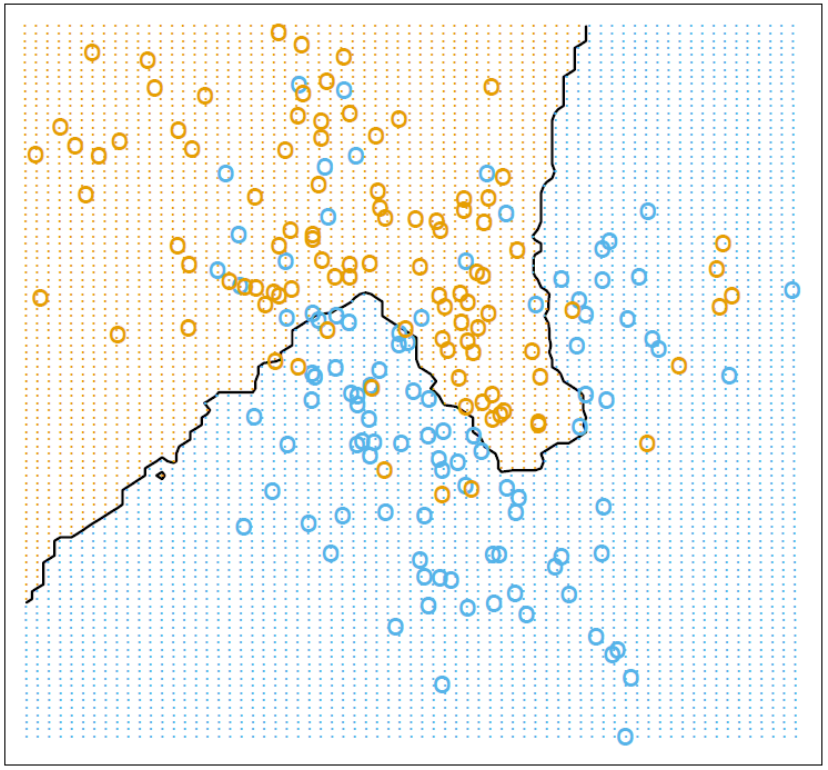
- The value of k has strong effect on k-NN performance
  - Large value: everything is classified as the most probable class: $P(y)$
  - Small value: highly variable, unstable decision boundaries
- One way to choose the right k: through validation
  - Leave aside part of the training set
  - Measure performance of different k values on this subset



The effect of k



k=1



k=15

From T. Hastie, R. Tibshirani, J. Friedman, "*The Elements of Statistical Learning*", 2nd edition, Springer, 2008

Resolving Ties

- In binary classification – use odd k
- Random decision: flip a coin to decide
- Use the prior: select the class with the highest prior (number of samples in the training set)
- Nearest: use a 1-NN classifier to decide

kNN Classification Algorithm

Training Phase: **N/A**

- We are given a dataset of m samples and their labels:

$$\{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots \ (x^{(m)}, y^{(m)})\}$$

Testing Phase:

- Given a new sample point x that we need to classify
- Compute distance $D(x, x^{(i)})$ to every training sample $x^{(i)}$
- Select k closest (most similar) instances $x^{(i1)}, \ldots \ x^{(ik)}$
- Output the class $\hat{y}$ which is most frequent in $y^{(i1)}, \ldots \ y^{(ik)}$

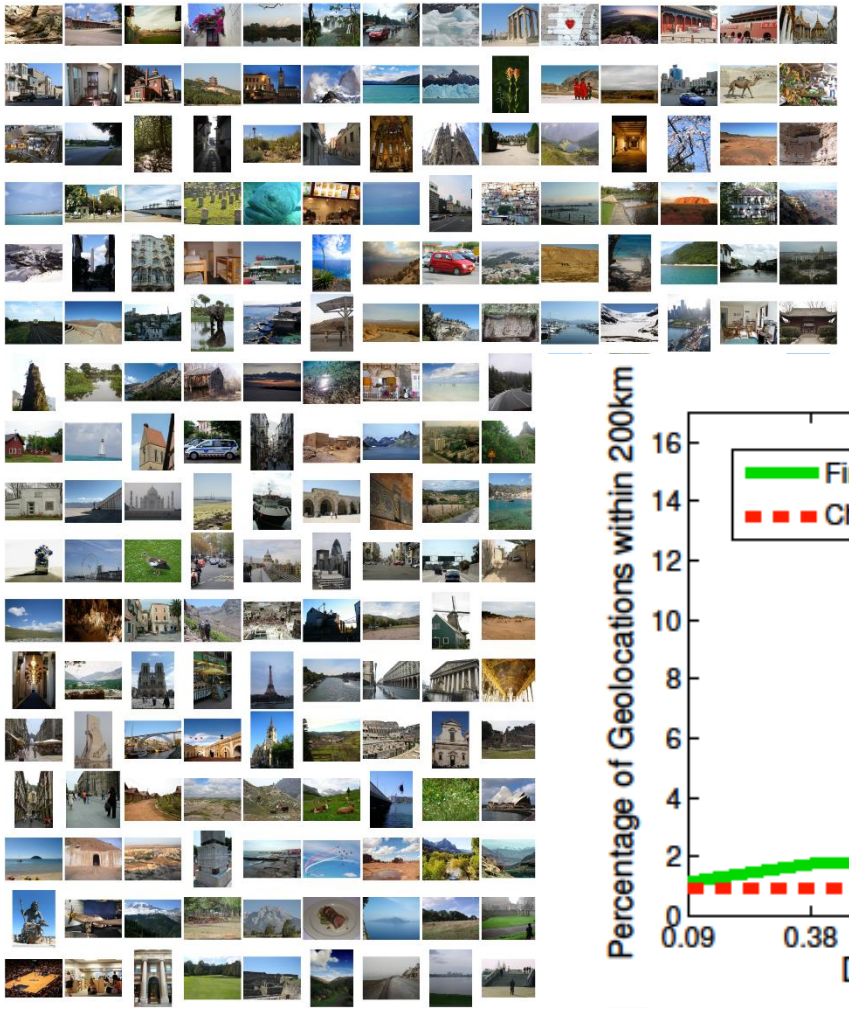
Pros and Cons of kNN

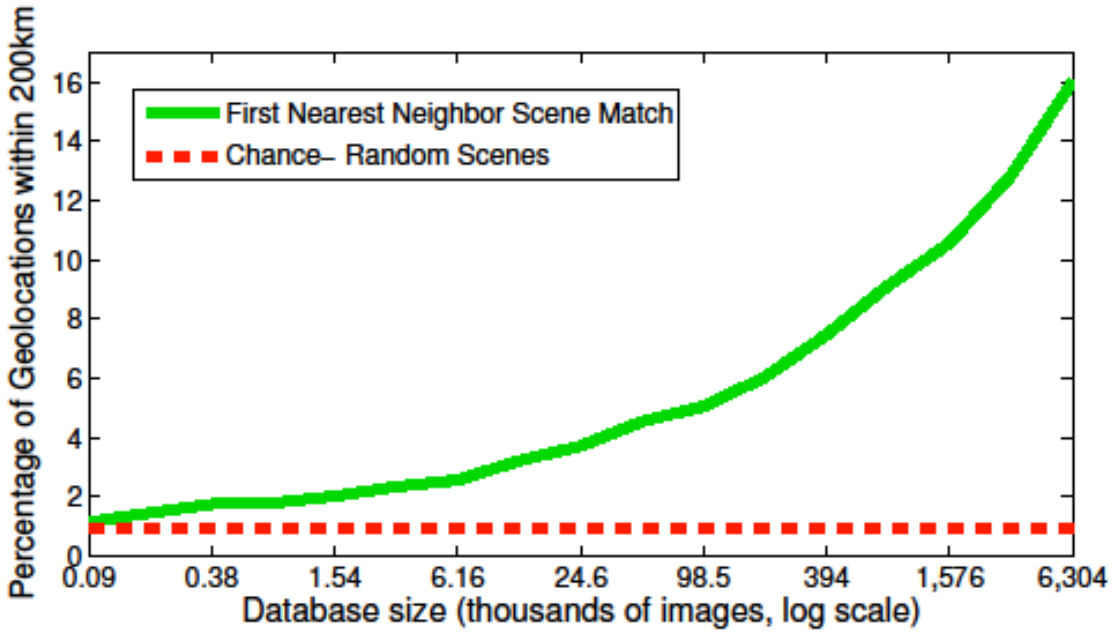
- Pros:
  - Almost no assumptions about the data
  - Smoothness: nearby regions in space means same class
  - Assumptions implied by distance function (locally)
  - Non-parametric approach: nothing to infer from the data except k and possibly the distance function $D(\cdot)$
  - Easy to update online
- Cons:
  - Need to handle missing data
  - Sensitive to class-outliers (mislabelled training instances)
  - Sensitive to lots of irrelevant attributes (affect distance)
  - Computationally expensive: need to store all training examples, and calculate distance to all examples

Finding the Location

Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

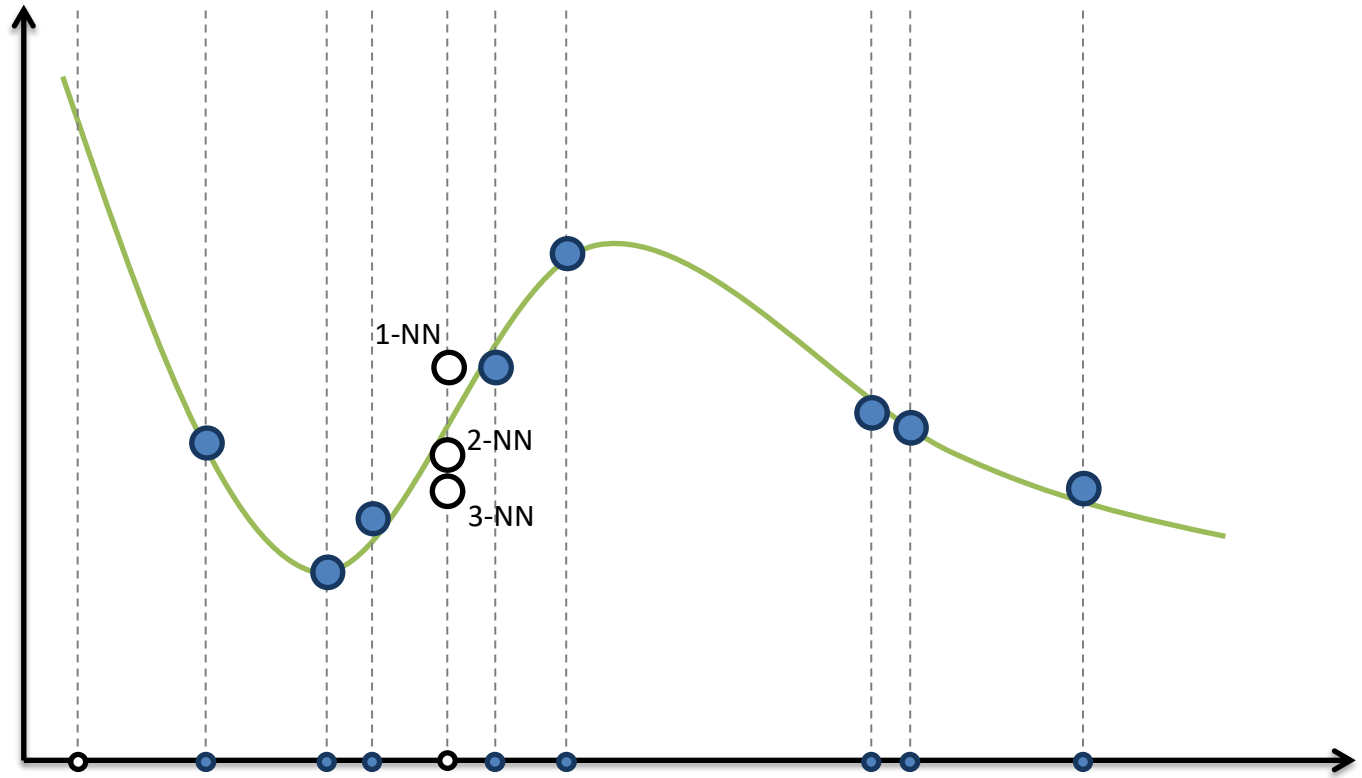
Finding the Location





Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

Nearest Neighbour Regression



Nearest Neighbour Regression

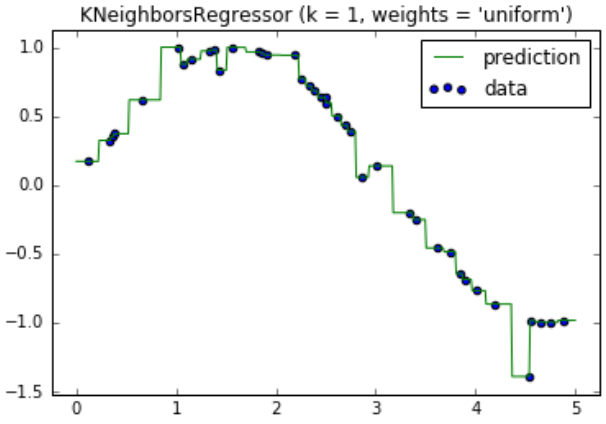
- Given:
  - A set of training examples $\{x^{(i)}, y^{(i)}\}$
  - A testing point x for which we want to predict the output value
- K-NN regression algorithm
  - Compute distance $D(x, x^{(i)})$ to every training example x_i
  - Select k closest examples

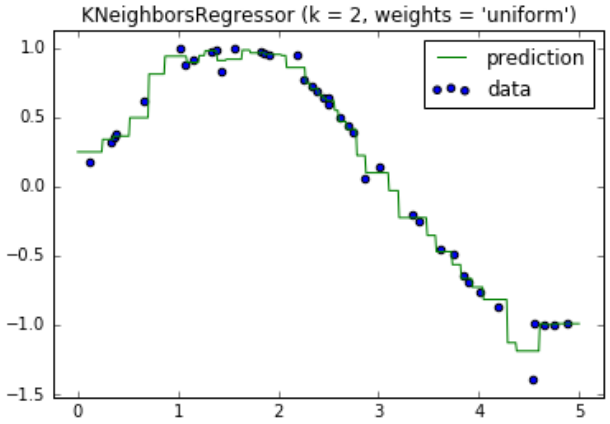
$$\{(x^{(i1)}, y^{(i1)}) \ldots (x^{(ik)}, y^{(ik)})\}$$

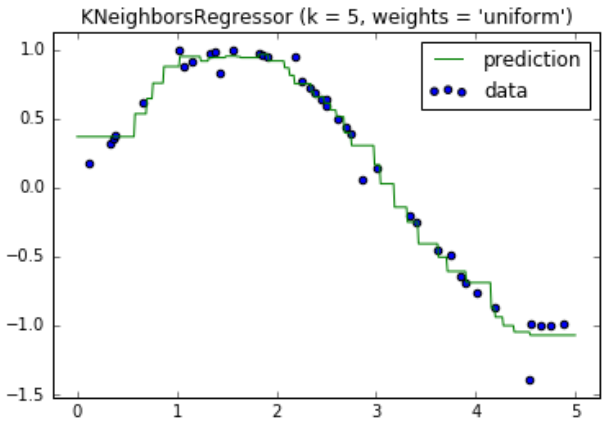
  - Calculate the output as the mean of $y^{(i1)} \ldots y^{(ik)}$:

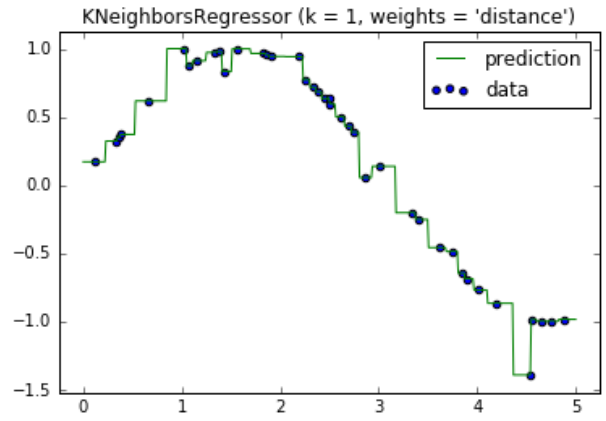
$$\hat{y} = \frac{1}{k} \sum_{j=1}^{k} y^{(ij)}$$

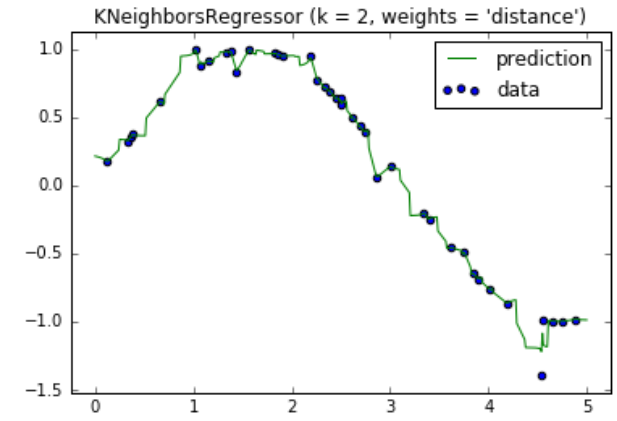
Nearest Neighbour Regression

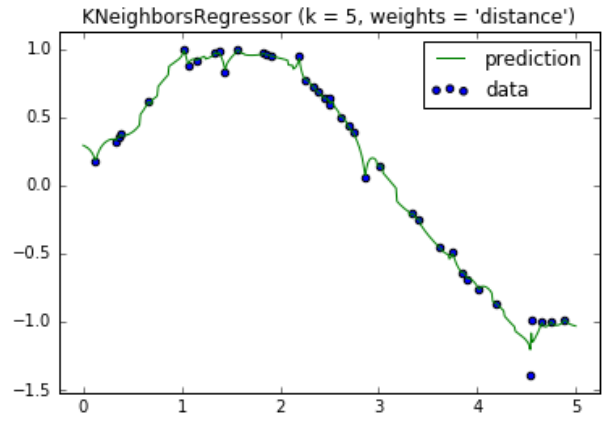












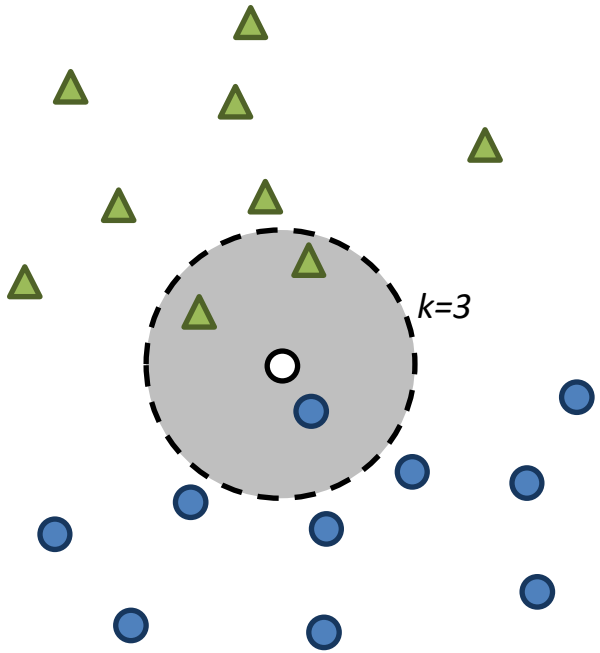
Requirements for kNN

- kNN is a memory-lazy algoritm
  - Data is not modelled and then discarded, but instead used at test time and need to be stored
  - All distances between query and training data examples need to be calculated
- This imposes three requirements
  - **Data** availability
  - **Efficiency** (fast queries)
  - Quality of the **distance function**

KNN – EFFICIENCY

Computational complexity of finding nearest neighbours

What you see



What the algorithm "sees"

Training set:

$$\{(8.4, 5.0), (4.2, 2.0), (8.7, 6.9), (6.0, 6.1), (8.3, 3.0), (6.1, 9.2), \ldots\}$$

Testing instance:

$$(1.5, 7.2)$$

To find nearest neighbours we need to compare one-by-one the testing instance to each training instance

Computational complexity of finding nearest neighbours

This has a complexity of $O(nd)$, where n is the number of training samples and d is the dimensionality of our feature space

Option 1: Reduce d (dimensionality reduction)

Simple feature selection, other methods

Option 2: Reduce n (compare to a selection of training samples)

Quickly identify $m \ll n$ potential near neighbours $O(md)$

Computational complexity of finding nearest neighbours

To reduce n we need methodologies to quickly identify a subset of m training samples ($m \ll n$) which are near neighbours of the testing sample

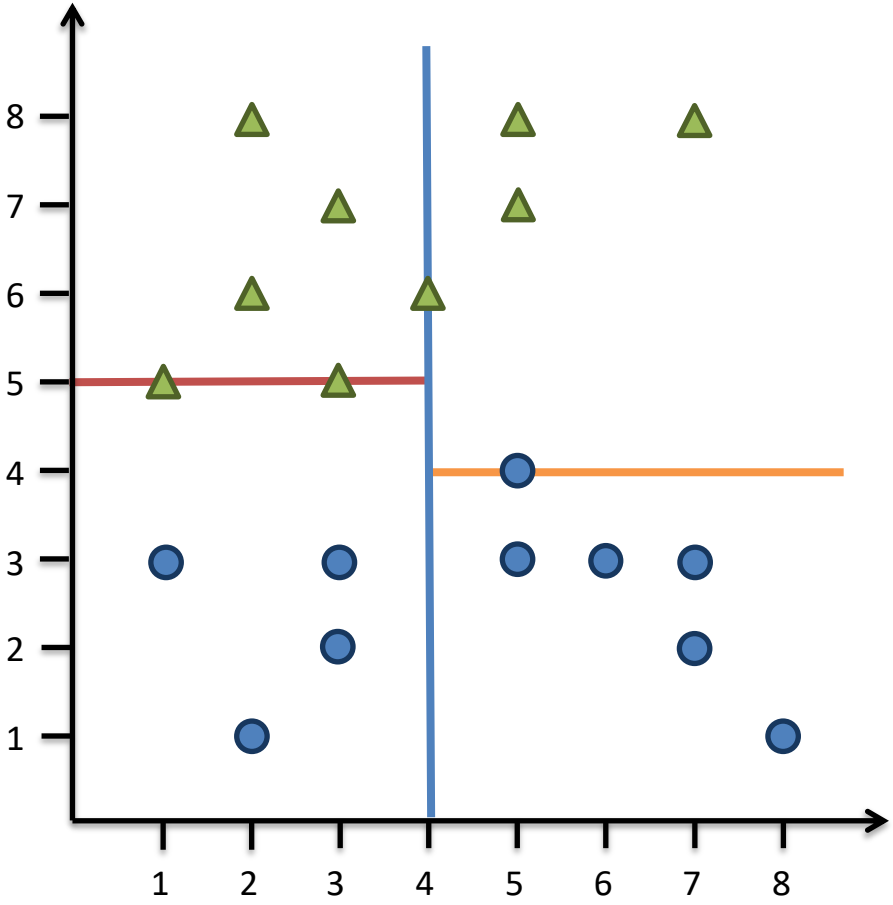
This basically implies clever ways to structure our data in memory, so that such a quick search for near neighbours is feasible

Typical approaches include:

- **K-D Trees**
- **Locality-sensitive hashing**
- **Inverted index**

K-D Tree Algorithm

To build a K-D tree from training data: (1) pick random dimension, (2) find median, (3) split data, (4) repeat



(1,3), (1,5), (2,8), (2,6), (2,1), (3,2), (3,3), (3,5), (3,7), (4,6), (5,3), (5,4), (5,7), (5,8), (6,3), (7,2), (7,3), (7,8), (8,1)

$x_1 \geq 4$

(1,3), (1,5), (2,8), (2,6), (2,1), (3,2), (3,3), (3,5), (3,7),

(4,6), (5,3), (5,4), (5,7), (5,8), (6,3), (7,2), (7,3), (7,8), (8,1)

$x_2 \geq 5$

(1,3), (2,1), (3,2), (3,3)

(1,5), (2,8), (2,6), (3,5), (3,7),

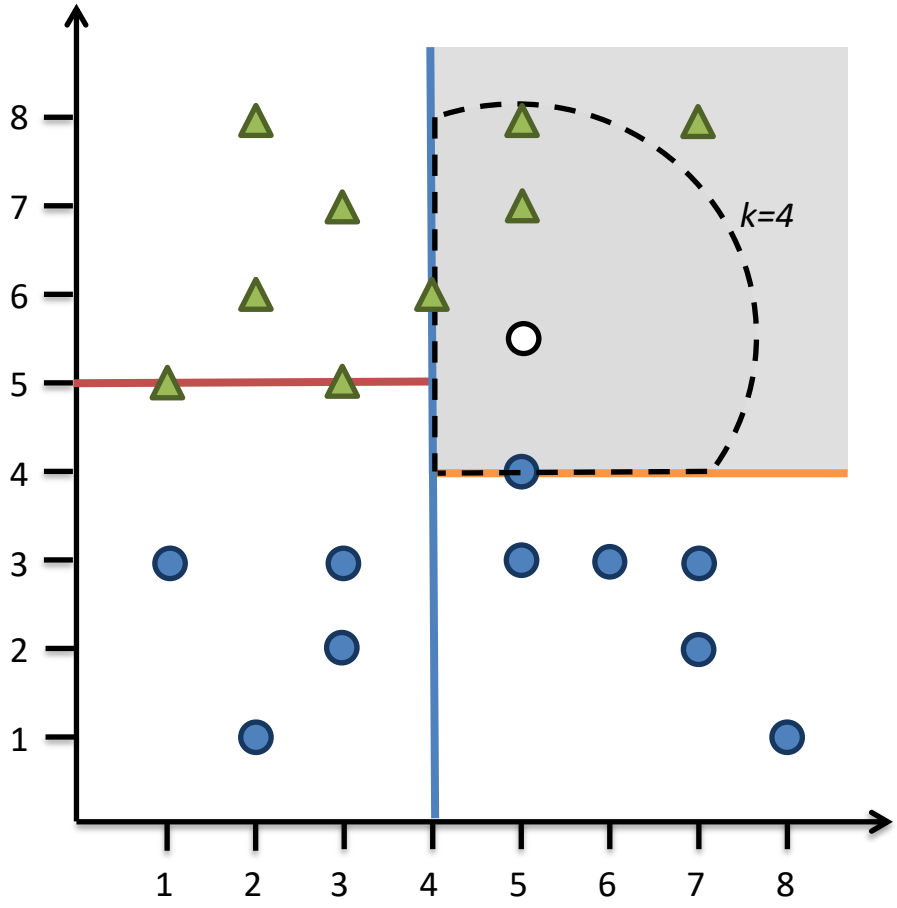
$x_2 \geq 4$

(5,3), (6,3), (7,2), (7,3), (8,1)

(4,6), (5,4), (5,7), (5,8), (7,8)

K-D Tree Algorithm

To find the nearest neighbour of a training point, e.g. (5.0, 5.5): (1) find the region containing the point, (2) compare to all points within that region



(1,3), (1,5), (2,8), (2,6), (2,1), (3,2), (3,3), (3,5), (3,7), (4,6), (5,3), (5,4), (5,7), (5,8), (6,3), (7,2), (7,3), (7,8), (8,1)

$x_1 \geq 4$

(1,3), (1,5), (2,8), (2,6), (2,1), (3,2), (3,3), (3,5), (3,7),

(4,6), (5,3), (5,4), (5,7), (5,8), (6,3), (7,2), (7,3), (7,8), (8,1)

$x_2 \geq 5$

(1,3), (2,1), (3,2), (3,3)

(1,5), (2,8), (2,6), (3,5), (3,7),

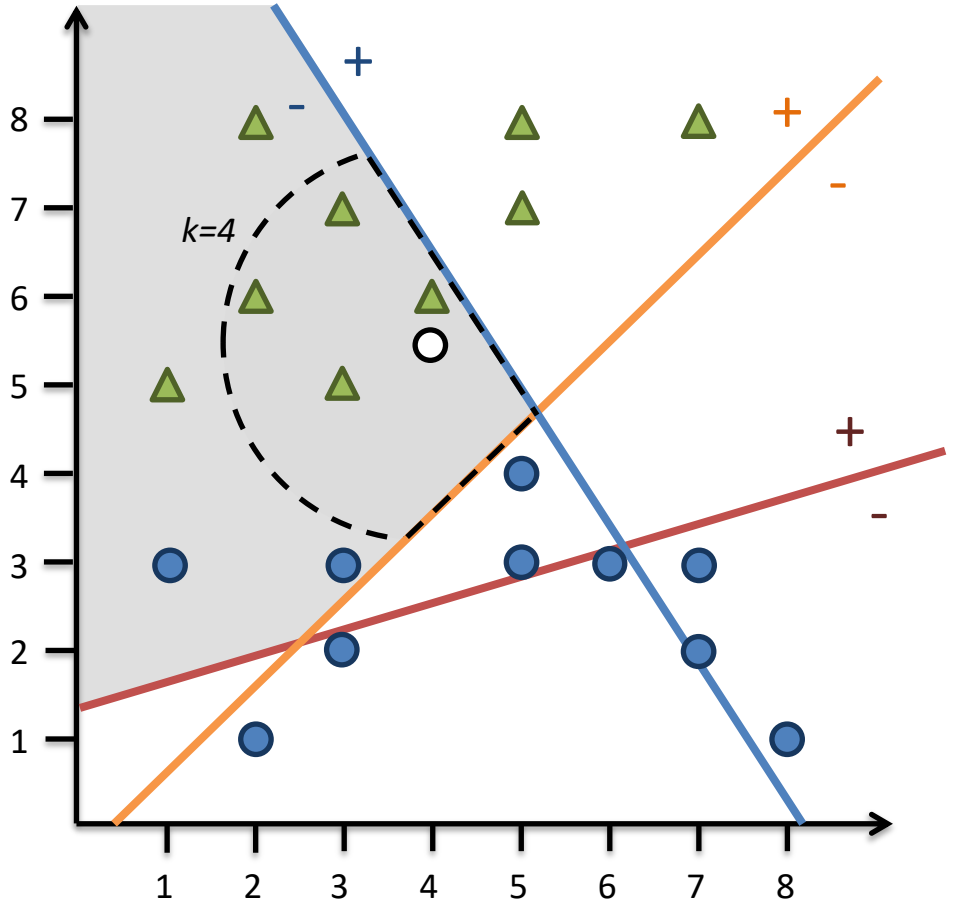
$x_2 \geq 4$

(5,3), (6,3), (7,2), (7,3), (8,1)

(4,6), (5,4), (5,7), (5,8), (7,8)

Locality Sensitive Hashing

To index training points, split space based on m random hyper-planes $h_1, \ldots\ h_m$. This defines 2^m regions (polytopes).



Given a testing point:

- Find the region R where it lies in (requires k dot products with $h_1, \ldots, h_m$)
- Compare to $n/2^m$ points that lie within R

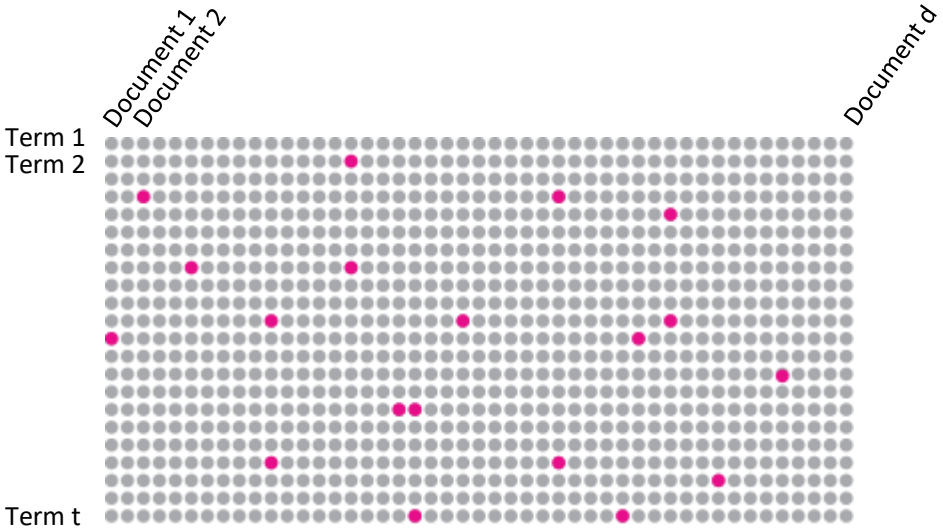
To deal with missing neighbours, the process can be repeated with different $h_1, \ldots, h_m$

Inverted Index

Makes sense when data is sparse (e.g. text in Web pages or emails)

Instead of listing all attributes (terms) for every instance (document), list every instance (document) that contain a particular attribute (term)

For a new testing example, merge inverted lists for attributes present



Dataset

1. “Bank account details”	**Ham**
2. “Personal details”	**Spam**
3. “Transaction details”	**Ham**
4. “Submit password”	**Spam**
5. “Confirm account”	**Ham**
6. “Confirm transaction”	**Ham**
Q: “Confirm password”	?

Inverted Index

Bank	→	1
Account	→	1, 5
Details	→	1, 2, 3
Personal	→	2
Transaction	→	3
Submit	→	4
Password	→	4
Confirm	→	5, 6

Computational complexity of finding nearest neighbours

- **K-D Trees**
  - Use when: low-dimensional, real-valued data. Only works when $d \ll n$
  - $O(d \log_2 n)$
  - inexact: can miss neighbours
- **Locality-sensitive hashing**
  - Use when: high-dimensional, real-valued or discrete data
  - $O(kd + {}^{nd}/_{2^k}) \approx O(d \log n)$ where $k \ll n$ bits in fingerprint
  - Inexact: can miss neighbours
- **Inverted index**
  - Use when: high-dimensional, sparse (discrete) data, (e.g. text)
  - $O(n'd')$ where $d' \ll d, n' \ll n$
  - Exact

KNN – DISTANCE FUNCTION

Which Distance Function

- Key component of the kNN algorithm
  - Defines which examples are similar and which are not
  - Has a strong effect on performance
- Euclidian distance
  - Symmetric, spherical, treats all dimensions equally
  - Sensitive to extreme differences in a single attribute

$$D(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_d \left| x_d - x'_d \right|^2}$$

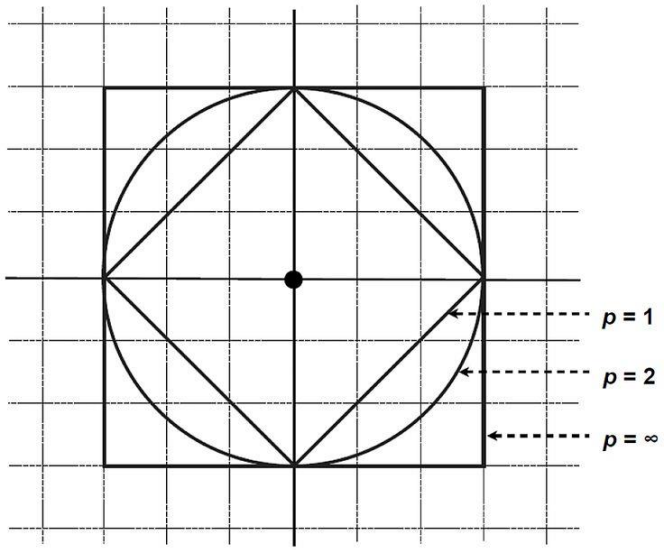
Definition of Distance/Similarity

Given a set S, a function $d: S \times S \to \mathbb{R}$ is a **metric** or **distance function** if:

- $d(X, Y) \geq 0$ *Non-negativity*
- $d(X, Y) = d(Y, X)$ *Symmetry*
- $d(X, Y) = 0 \Leftrightarrow X = Y$ *Identity of indiscernibles*
- $d(X, Z) \leq d(X, Y) + d(Y, Z)$ *Triangle inequality for* $X, Y, Z \in S$

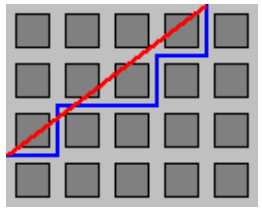
Minkowski Distance (p-norm)

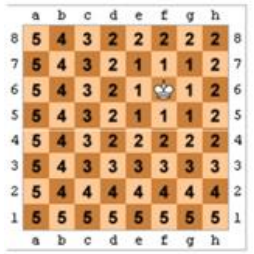
$$D(\mathbf{x}, \mathbf{x}') = \sqrt[p]{\sum_d |x_d - x'_d|^p}$$



- p=1: Manhattan (L1-norm, City-block)
- p=2: Euclidian (L2-norm)
- $p \to \infty$: Chebyshev (Lmax-norm), logical OR

$$\max_d |x_d - x'_d|$$



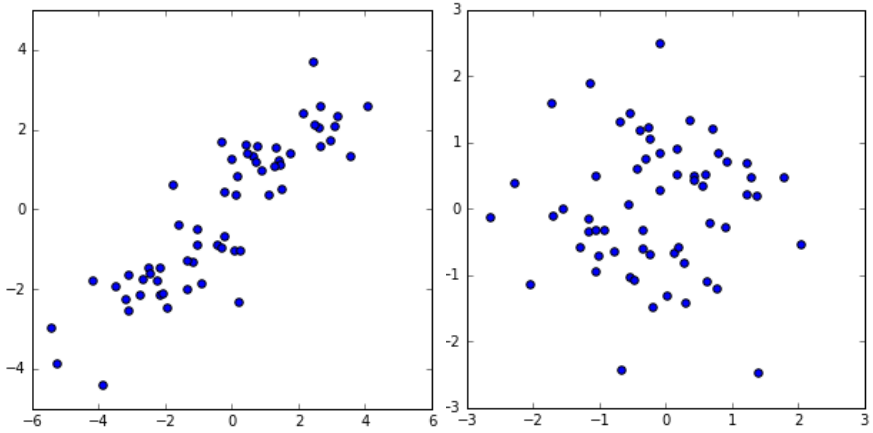


Mahalanobis distance

- Mahalanobis distance
  - Measures how many standard deviations away a point is from the mean
  - Equivalent to applying a *whitening transform* (if axes are rescaled to unit variance, it corresponds to Euclidean distance)

$$D(x, x') = \sqrt{(x - x')^T S^{-1} (x - x')}$$

$$S = \sum_{i,j} (x_i - \mu)\left(x_j - \mu\right)^T$$



Hamming Distance

- Use when you deal with categorical attributes
- Counts in how many of the attributes x and x’ differ

$$D(\mathbf{x}, \mathbf{x}') = \sum_d \mathbf{1}_{x_d \neq x'_d}$$

GAGCCTACTAACGGGAT
CATCGTAATGACGGCCT

Kullback-Leibler (KL) divergence

- Good for histograms and probability density functions ($x(i) > 0, \sum_i x(i) = 1$)

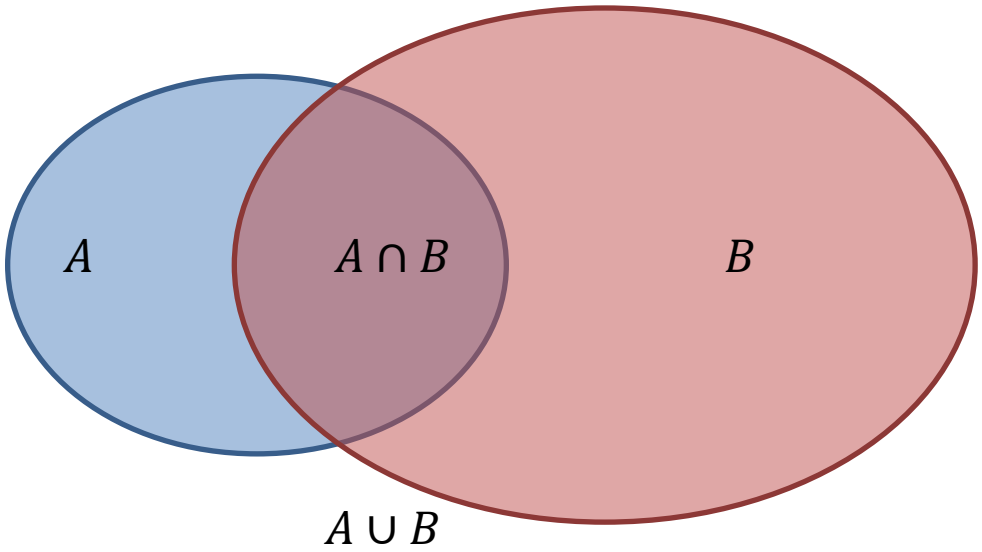
$$D(P \parallel Q) = -\sum_i P(i) \log \frac{P(i)}{Q(i)}$$

- Asymmetric function
- A measure (but not a metric) of information gain
  - In Bayesian terms: a measure of the information gained when one revises one's beliefs from the prior probability distribution Q to the posterior probability distribution P.

Jaccard similarity (index)

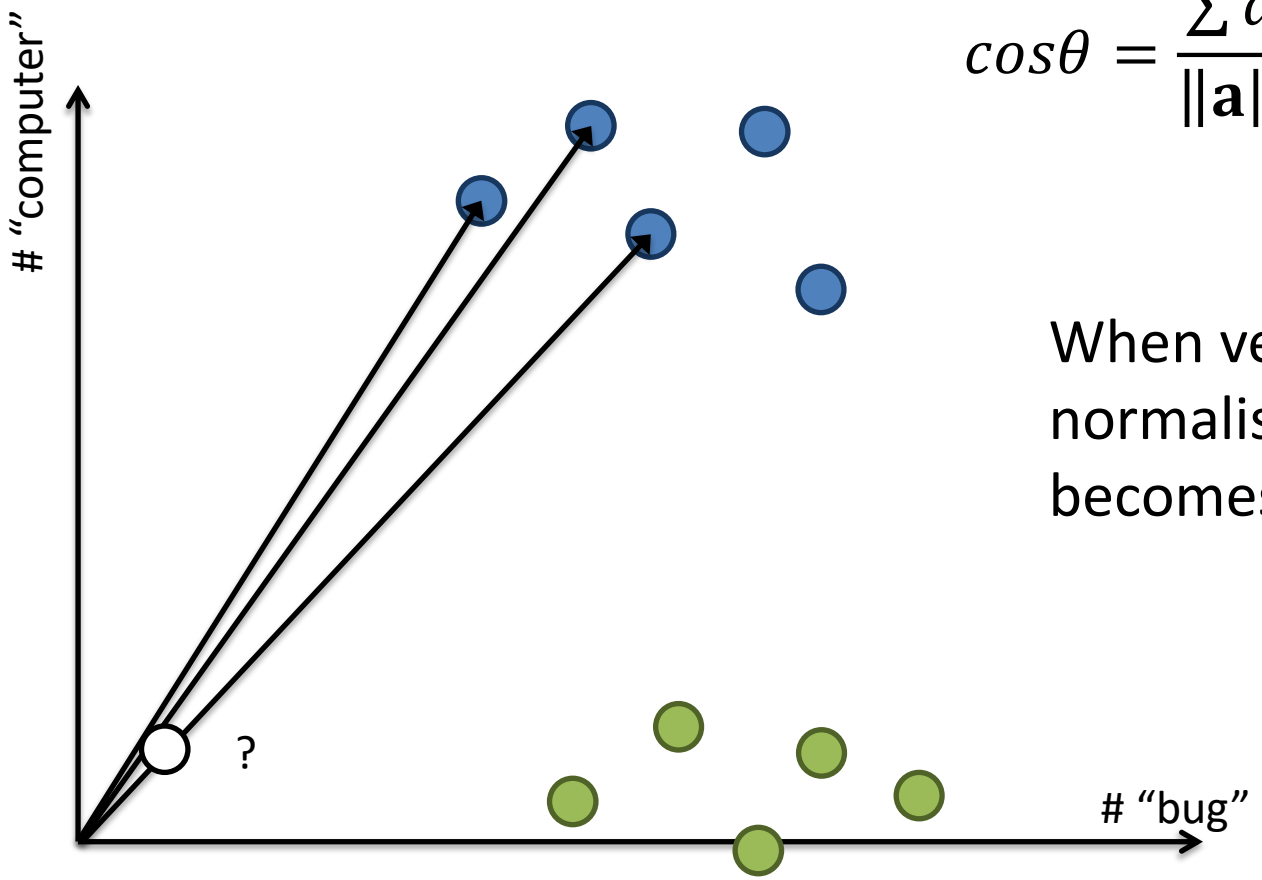
- Also known as “Intersection over Union” - IoU
- Similarity between sets of things

$$J(A,B) = \frac{A \cap B}{A \cup B}$$



Cosine Similarity (Distance)

Cosine Similarity (Distance)



$$cos\theta = \frac{\sum a_d b_d}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

When vectors are normalised to one, it becomes the dot product

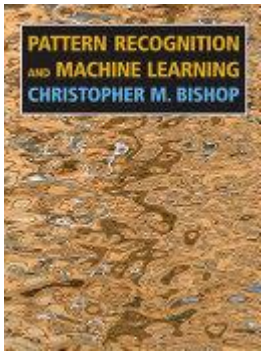
Which Distance Function

- Other custom distance measures such as
  - BM 25 ranking function (e.g. for retrieving documents according to their relevance to a given search query)
  - Levenshtein (string edit) distance
  - Graph edit distance
  - Dynamic time warping
  - …

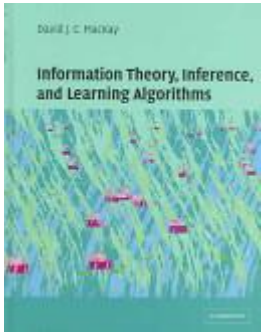
Resources

See the following sources for further information:

C. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006
Some related material available:
http://research.microsoft.com/en-us/um/people/cmbishop/prml/index.htm



D. MacKay, "Information Theory, Inference and Learning Algorithms", Cambridge University Press, 2003.
Book available online:
http://www.inference.phy.cam.ac.uk/mackay/



R.O. Duda, P.E. Hart, D.G. Stork, "Pattern Classification", Wiley & Sons, 2000
Have a look inside at selected chapters:
http://books.google.es/books/about/Pattern_Classification.html?id=Br33IRC3PkQC&redir_esc=y