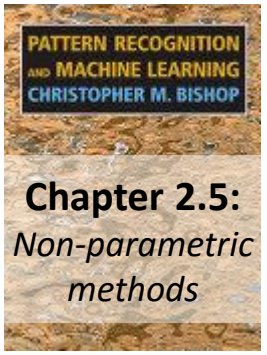
Pattern Analysis and Recognition

Lecture 8: k-Nearest Neighbour

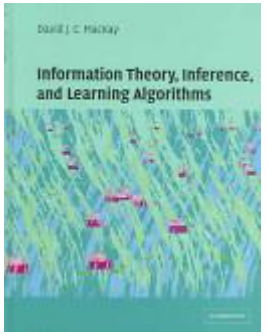
Resources

Some of the material in this slides was sourced from:

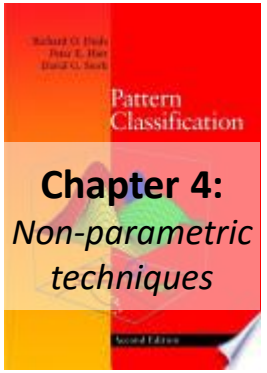
C. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006
Some related material available:
http://research.microsoft.com/en-us/um/people/cmbishop/prml/index.htm



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



Last time on Pattern Analysis and Recognition

RECAP

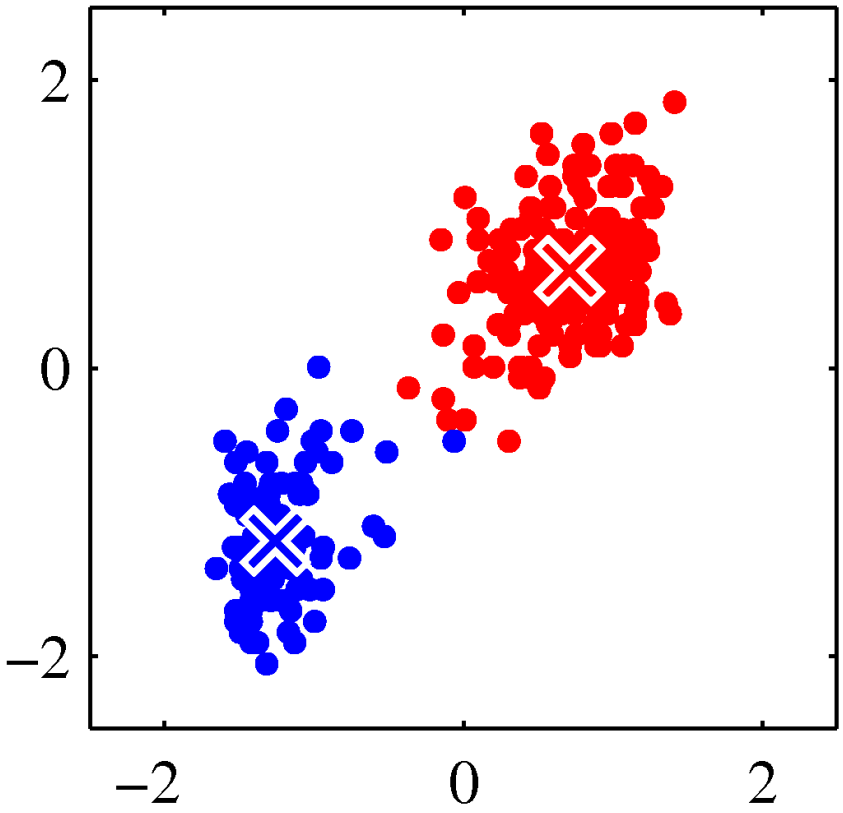
K-Means Algorithm

Step 1 Initialize K "means" $\boldsymbol{\mu}_k$, one for each class

Step 2 Assign each point to the closest mean $\boldsymbol{\mu}_k$

Step 3 Update the means $\boldsymbol{\mu}_k$ of the new clusters

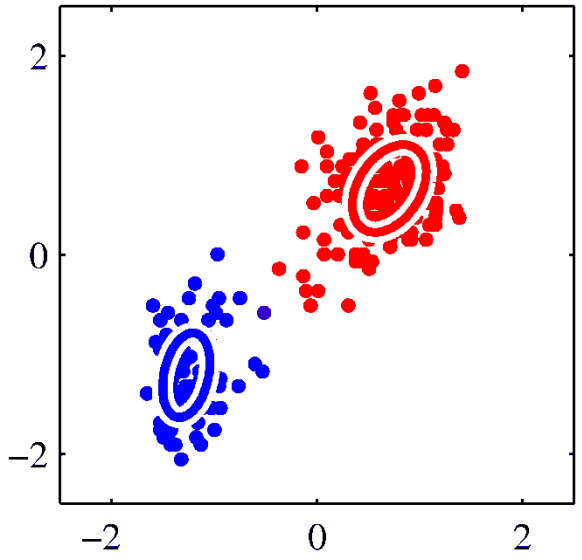
Step 4 If means do not change terminate, else back to step 2



K=2

Expectation Maximization (EM) for Gaussian Mixtures

Initialization	Initialize the parameters of K Gaussians ($\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k)
Expectation	Give each point $x^{(n)}$ an assignment score $\gamma(z_{nk})$ for each cluster k
Maximisation	Given the scores, adjust $\{\boldsymbol{\mu}_k, \pi_k, \boldsymbol{\Sigma}_k\}$ for each cluster k
Terminate	Evaluate log likelihood, stop when converged , else back to E-step



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?

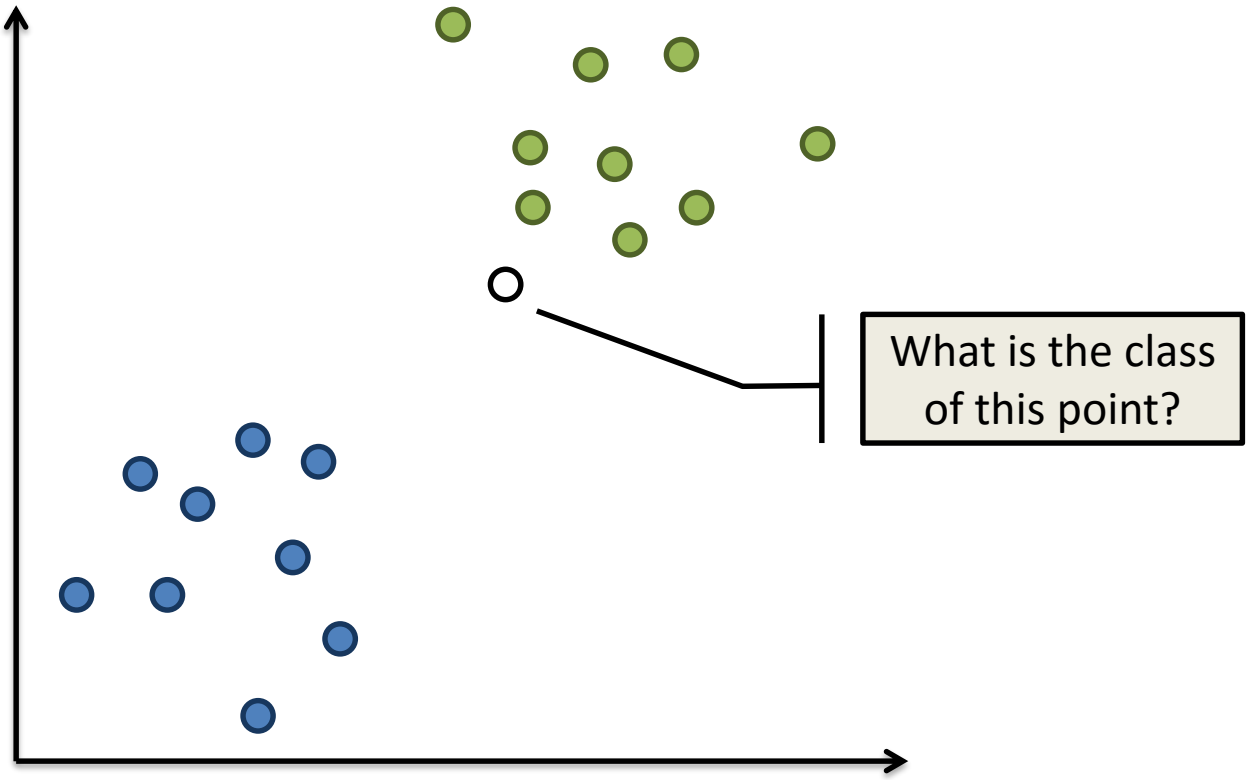
Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

A challenging problem

Where was this photo taken?

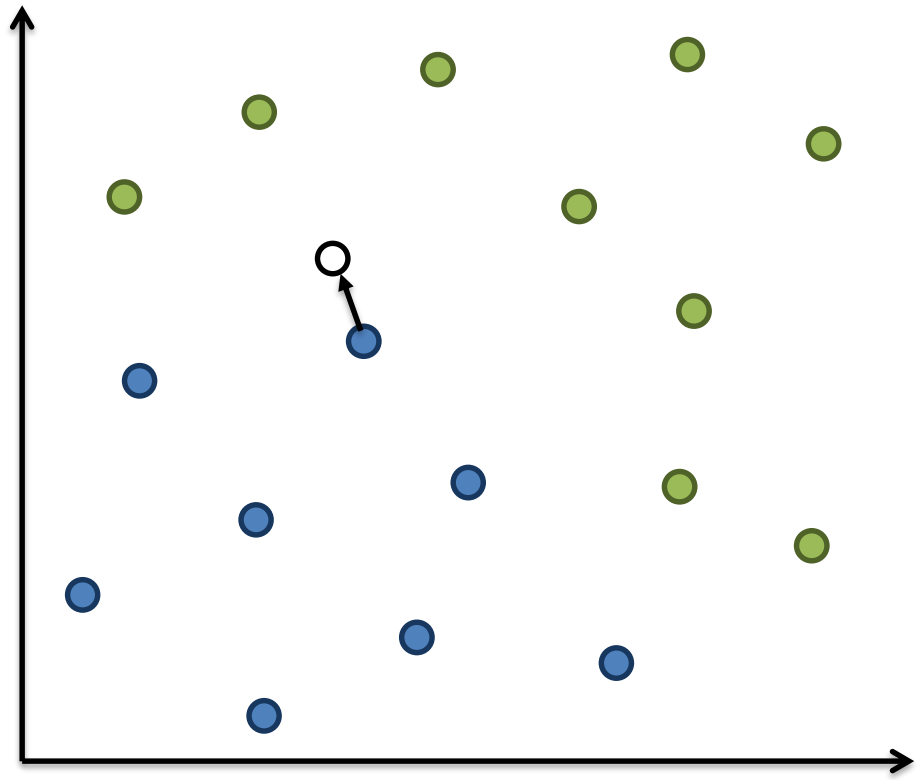
Hays, James, and Alexei A. Efros. "IM2GPS: estimating geographic information from a single image." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

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

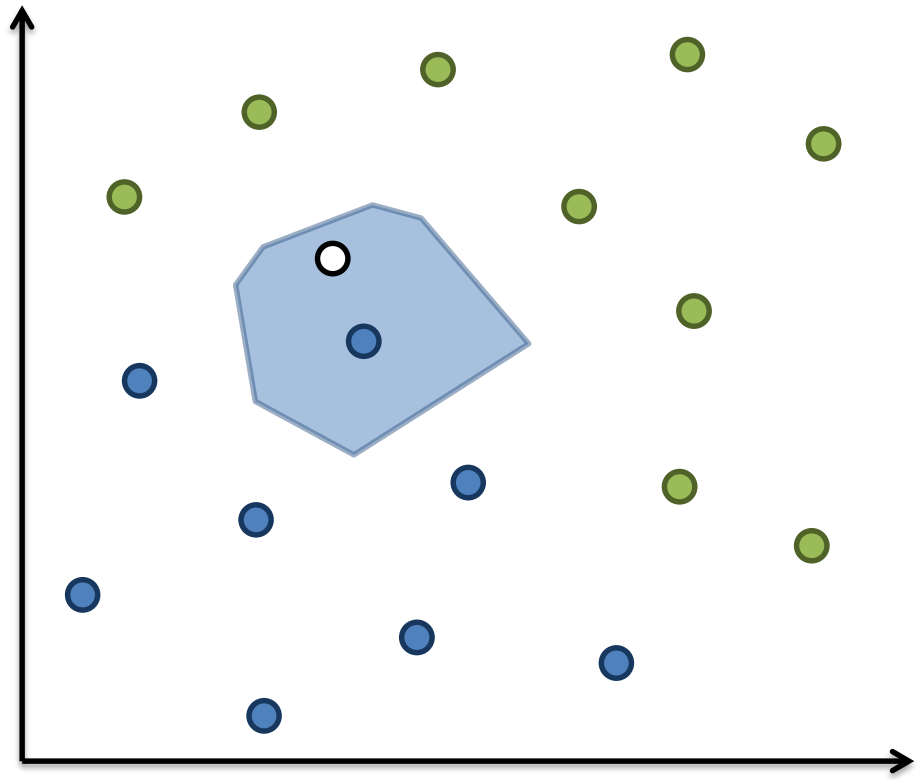
Nearest Neighbour Classification



Nearest Neighbour Classification algorithm:

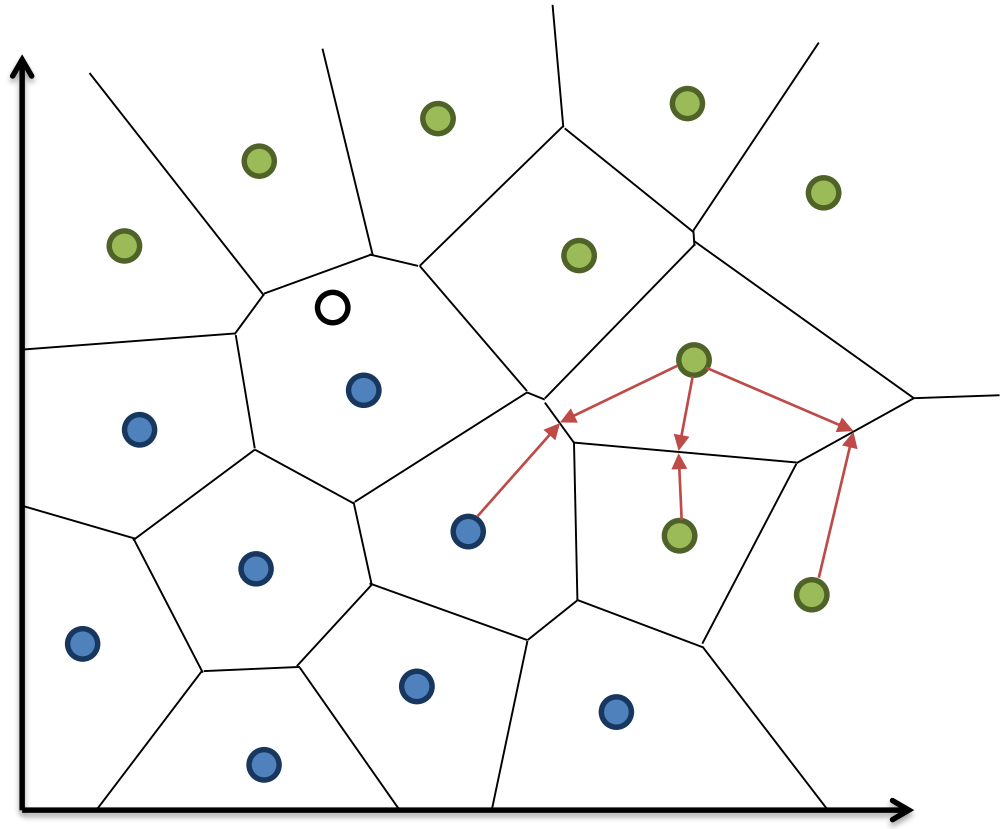
1. Find the most similar (closest) training example
2. Use its class as the prediction

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

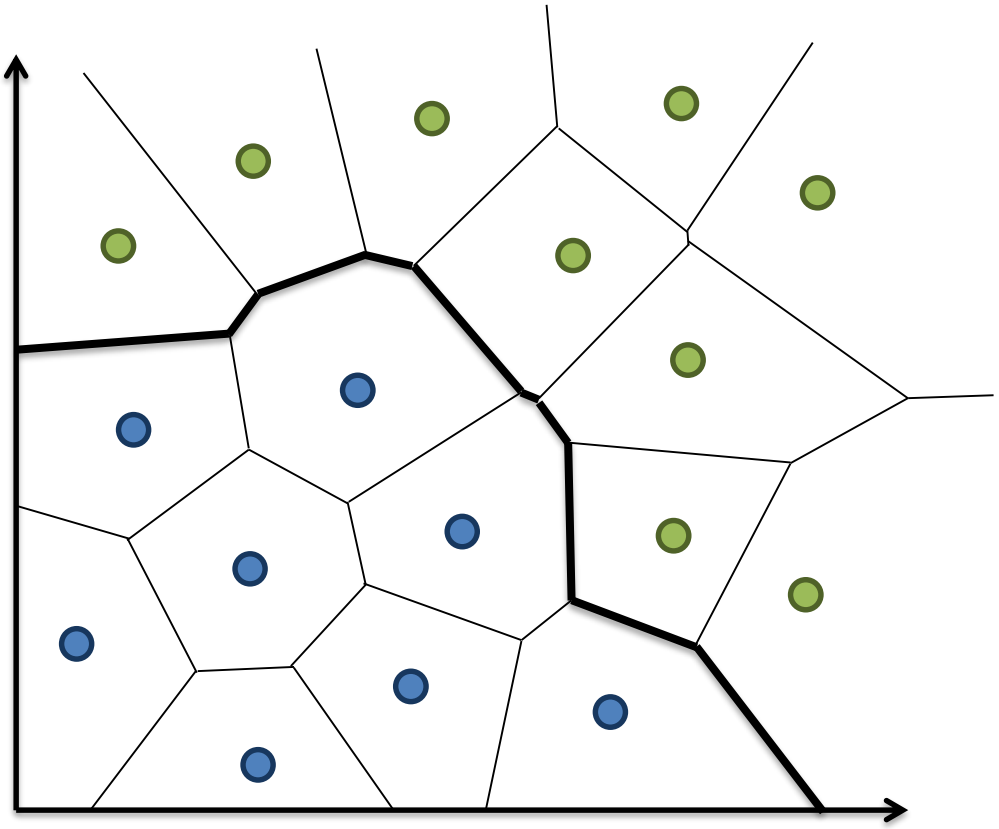
Nearest Neighbour Algorithm



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

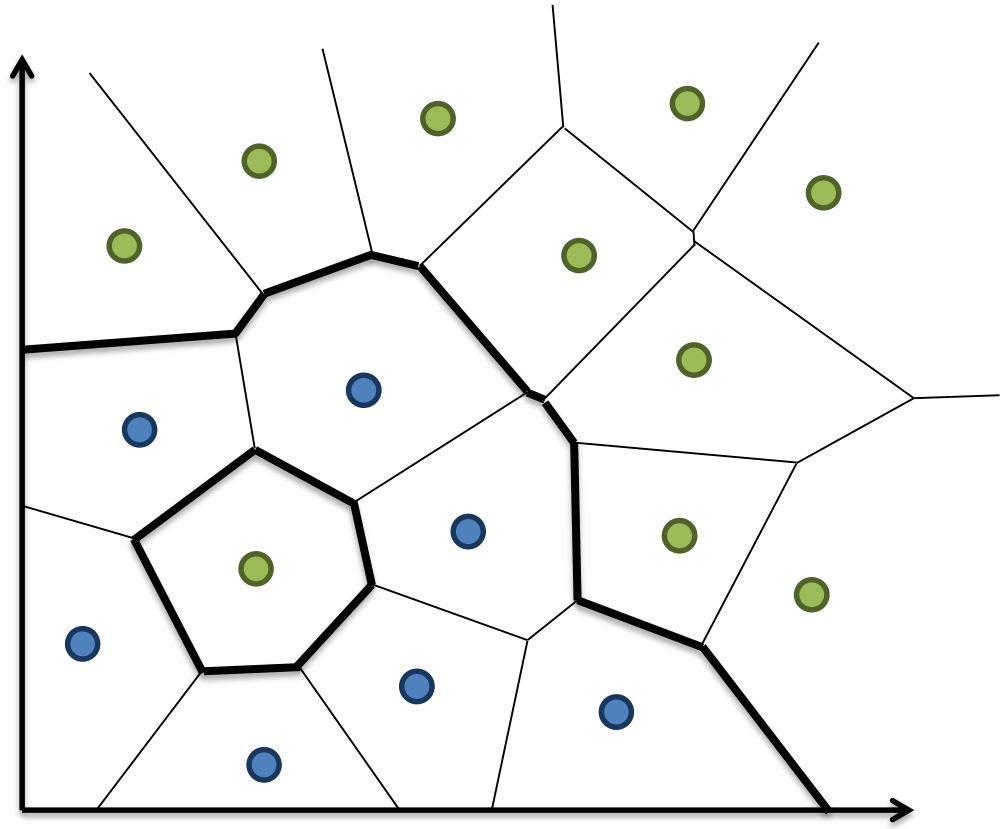
The **boundary** between two training examples is defined by the set of points that have the same distance to the two training examples

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

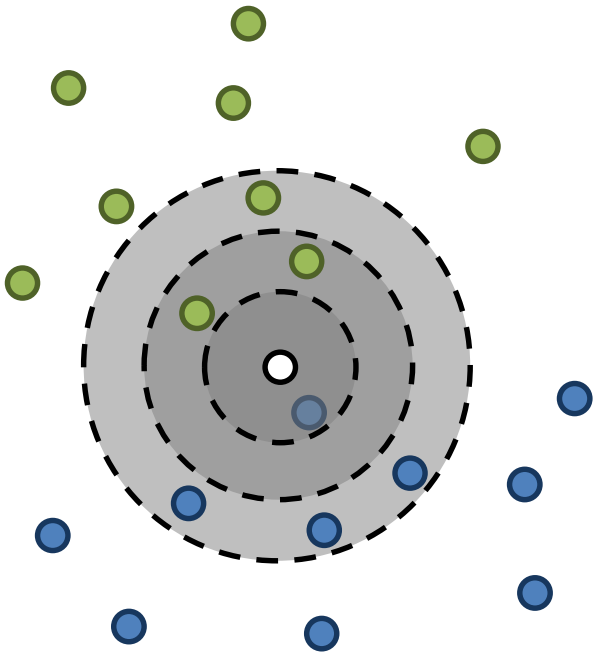


Outliers have significant effects on the decision boundary

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

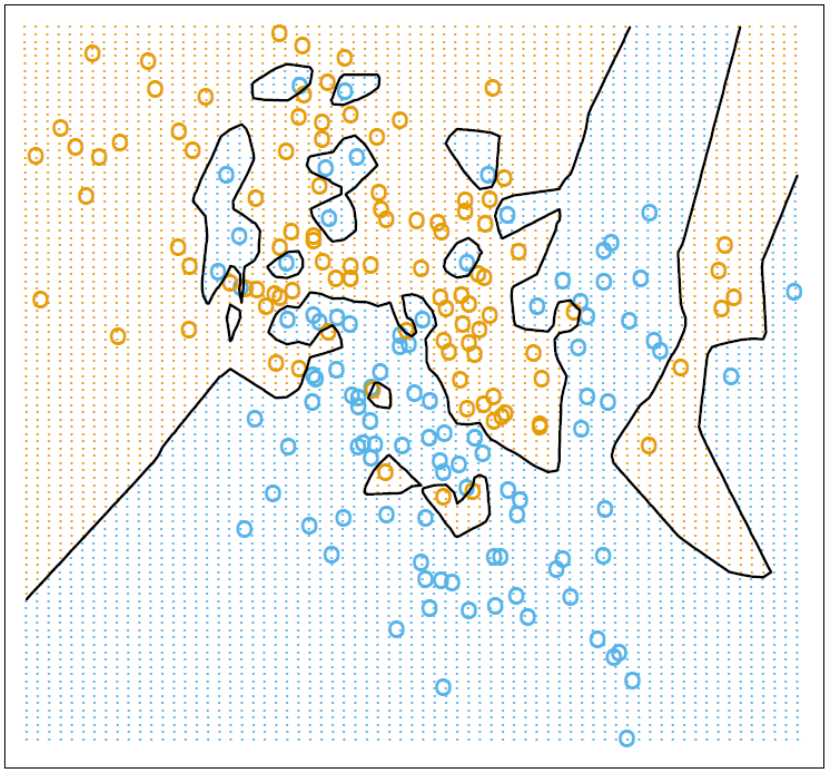
How many Neighbours

- 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

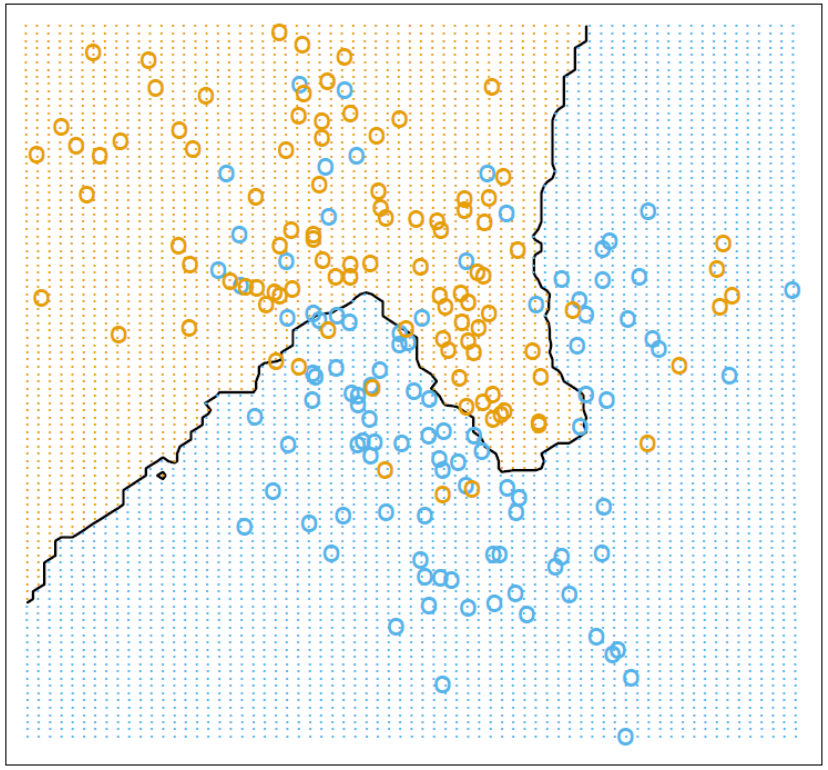


- What if you use the whole training set for validation?
- If your smallest class has n samples, what is the maximum k that permits classifying as this class?

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:

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

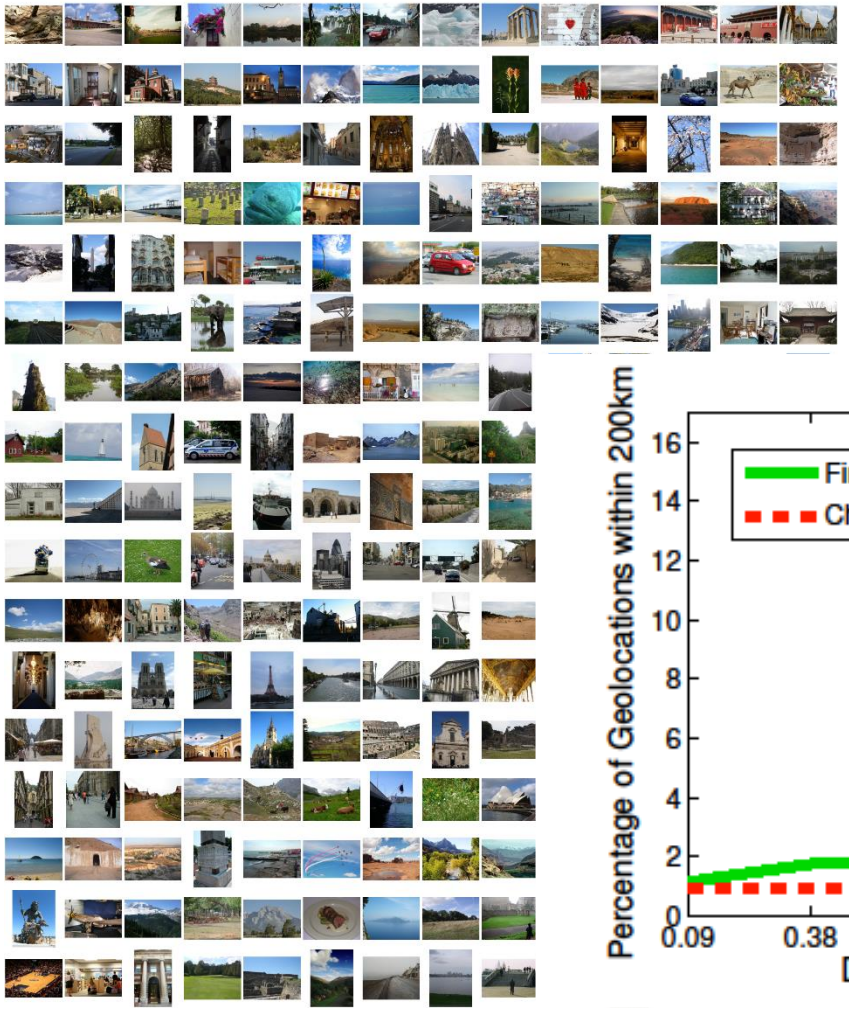
Testing Phase:

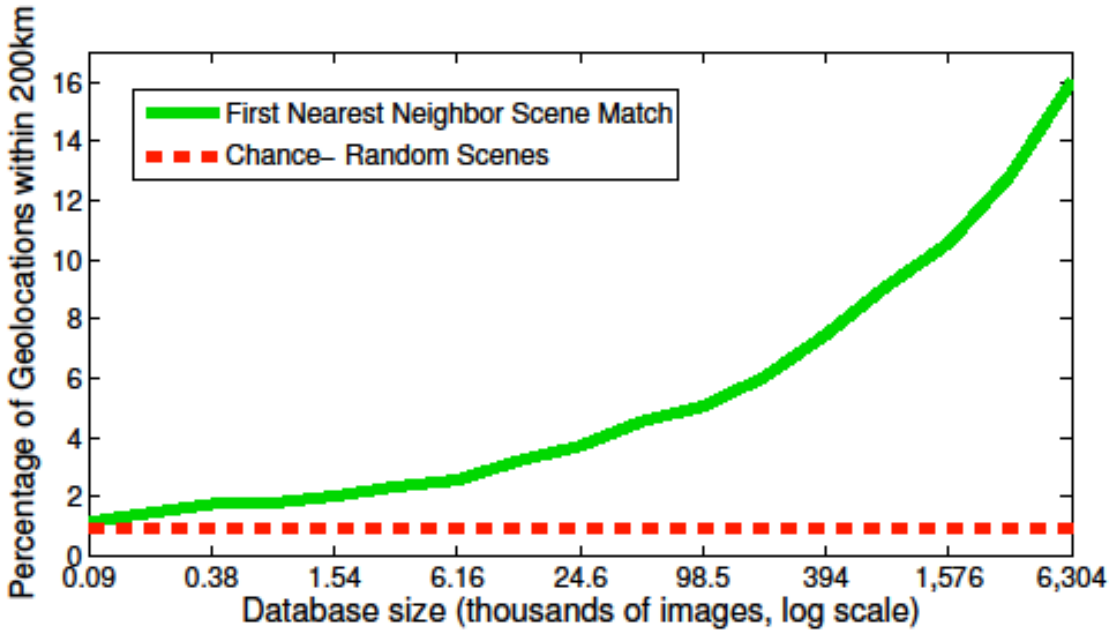
- Given a new sample point x that we need to classify
- Compute distance $D(x, x^{(i)})$ to every training sample $x^{(i)}$
- Select k closest (most similar) instances $x^{(i1)}, \ldots \ x^{(ik)}$
- Output the class $\hat{y}$ which is most frequent in $y^{(i1)}, \ldots \ y^{(ik)}$

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.

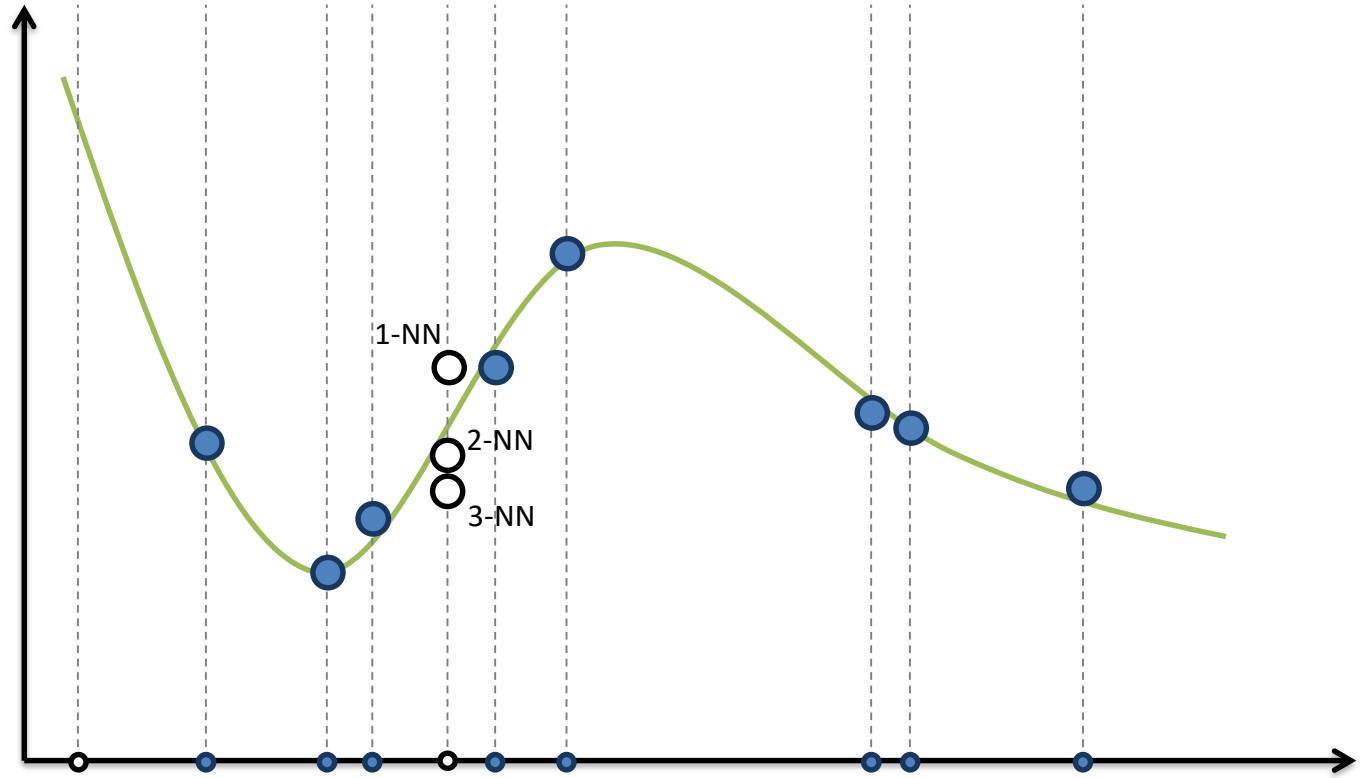
Finding the Location





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

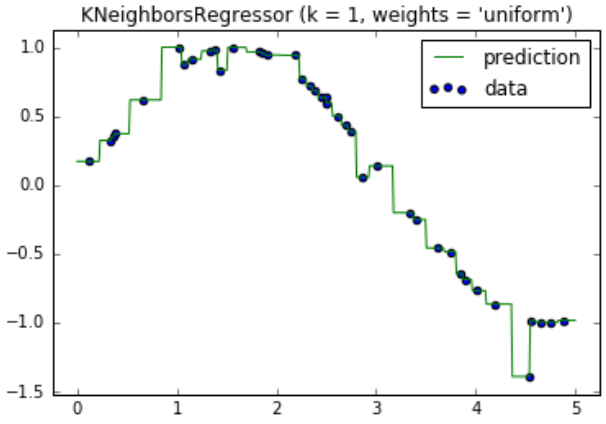
- Given:
  - A set of training examples $\{x^{(i)}, y^{(i)}\}$
  - A testing point x for which we want to predict the output value
- K-NN regression algorithm
  - Compute distance $D(x, x^{(i)})$ to every training example x_i
  - Select k closest examples

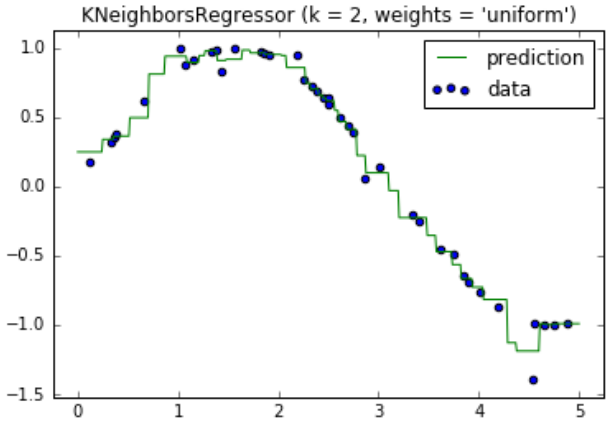
$$\{(x^{(i1)}, y^{(i1)}) \ldots (x^{(ik)}, y^{(ik)})\}$$

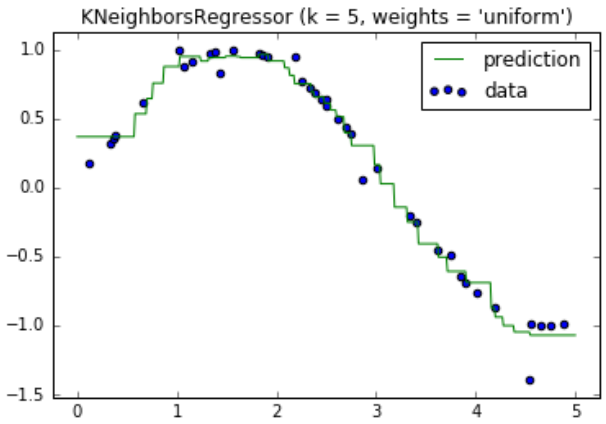
  - Calculate the output as the mean of $y^{(i1)} \ldots y^{(ik)}$:

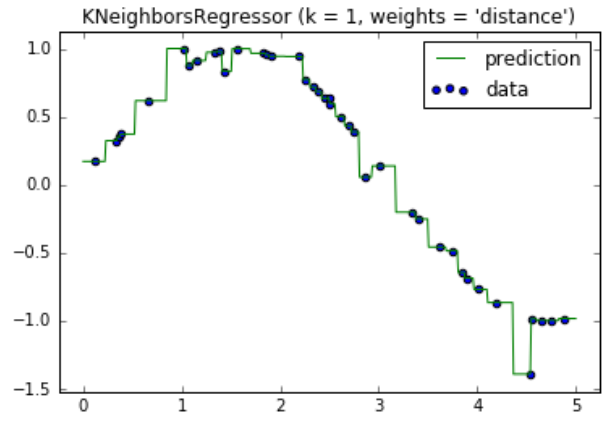
$$\hat{y} = \frac{1}{k} \sum_{j=1}^{k} y^{(ij)}$$

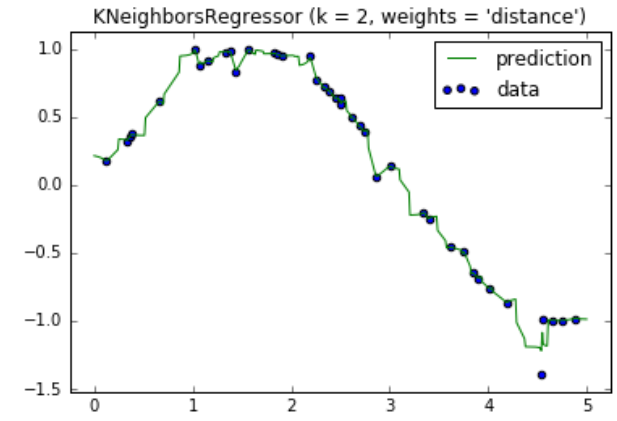
Nearest Neighbour Regression

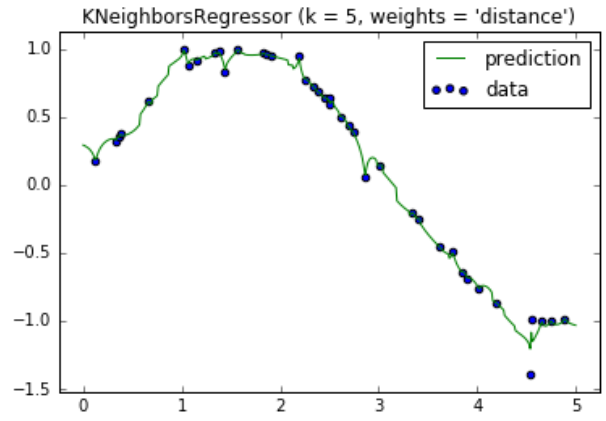












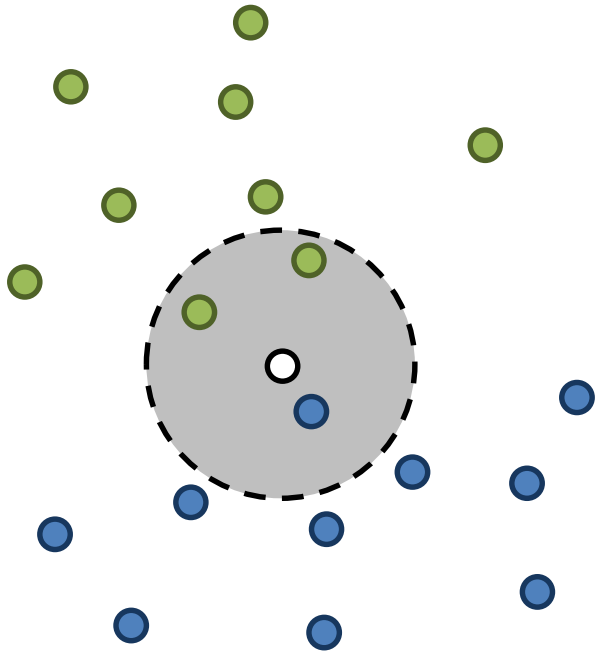
Requirements for kNN

- 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 $O(md)$

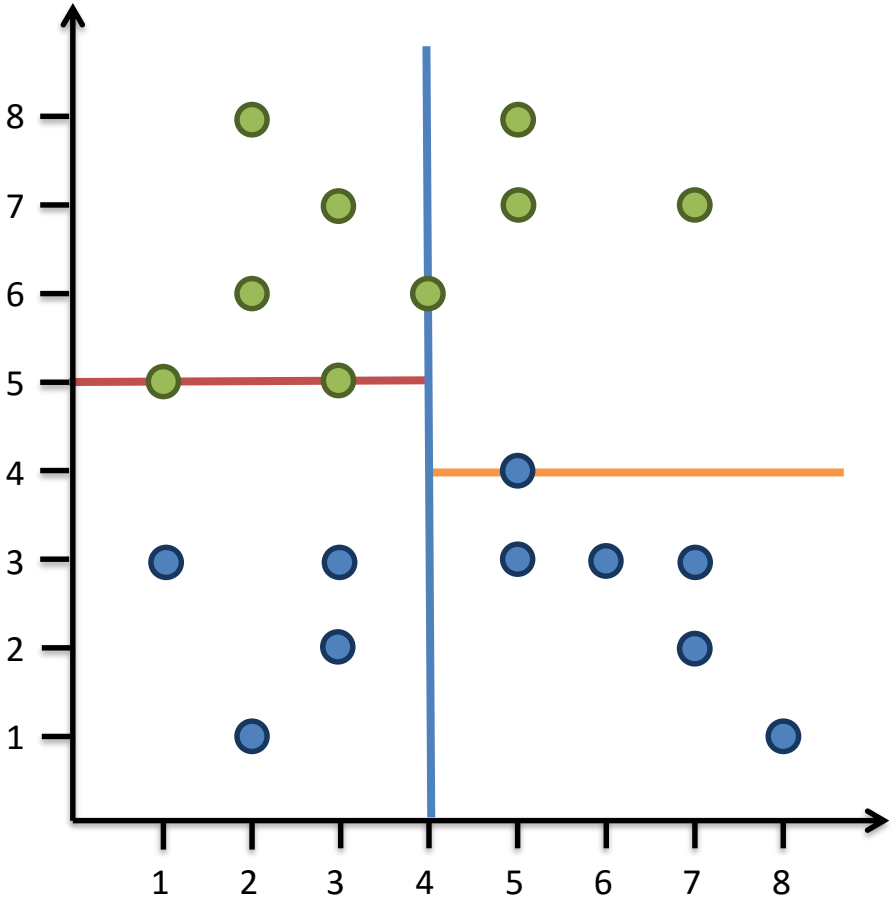
Computational complexity of finding nearest neighbours

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

- **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

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), (7,7), (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), (7,7), (8,1)

$x_2 \geq 5$

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

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

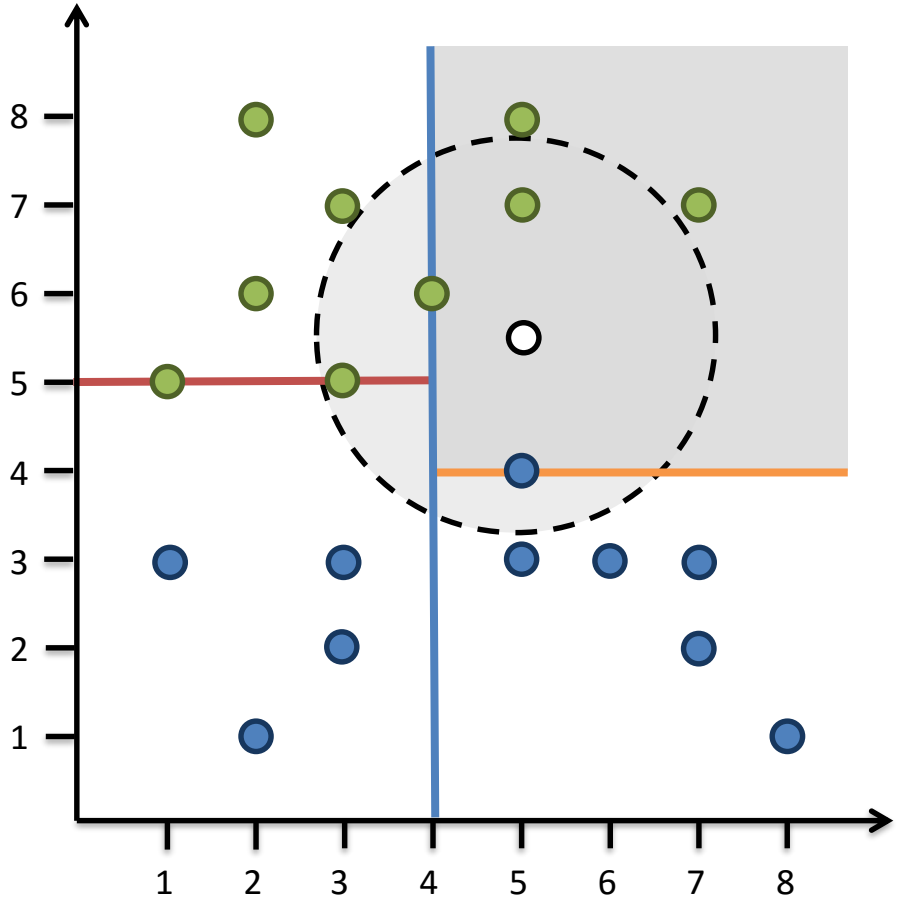
$x_2 \geq 4$

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

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

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), (7,7), (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), (7,7), (8,1)

$x_2 \geq 5$

$x_2 \geq 4$

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

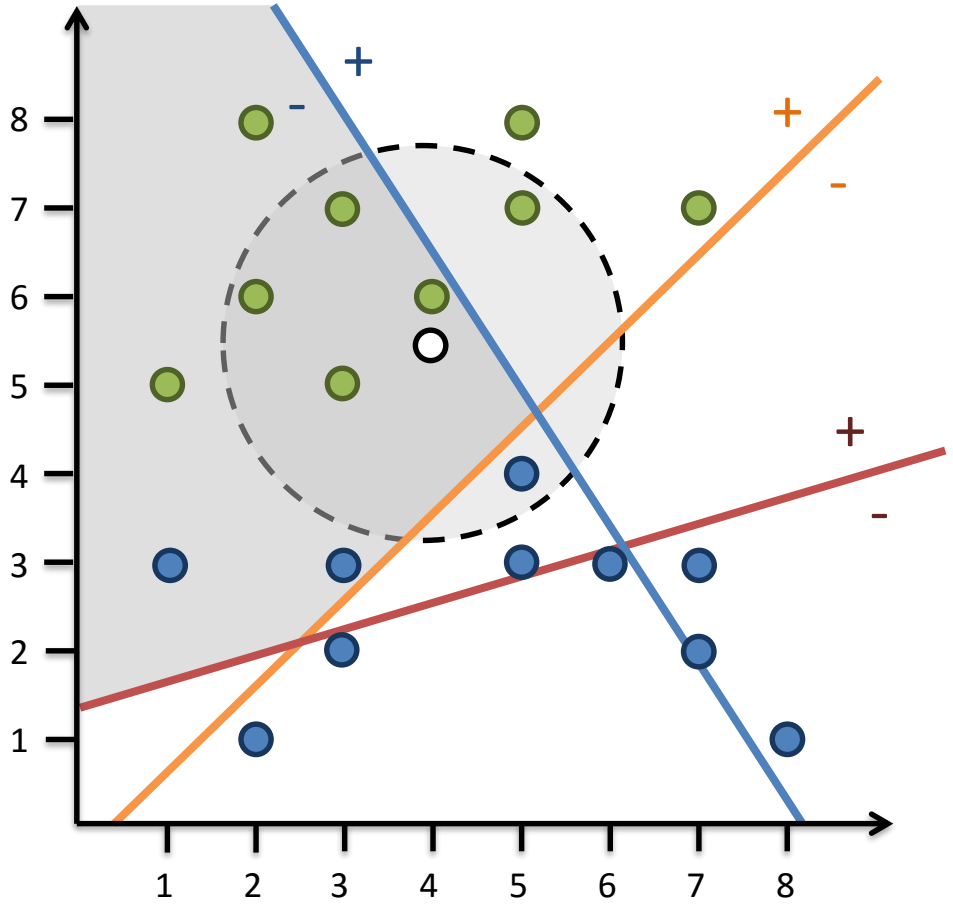
(1,5), (2,8), (2,6), (3,5), (3,7),

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

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

Locality Sensitive Hashing

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



Given a testing point:

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

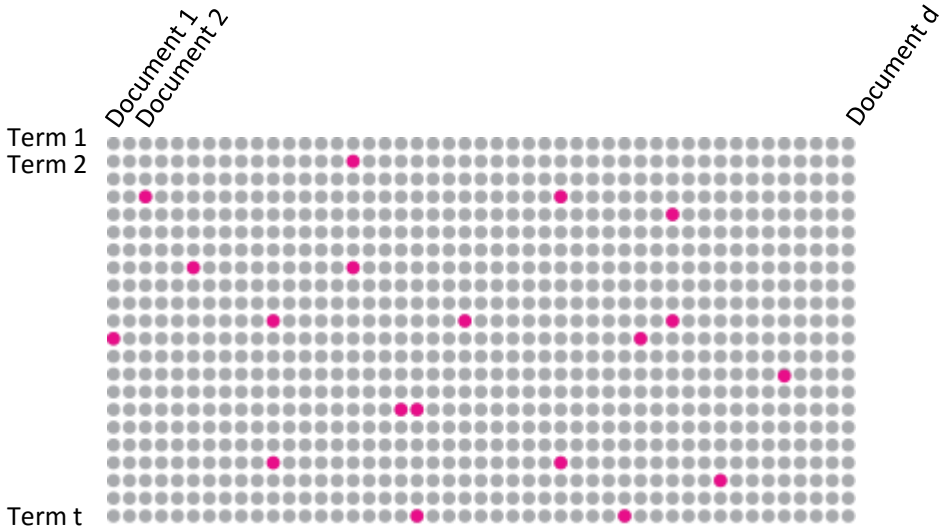
To deal with missing neighbours, the process can be repeated with different $h_1, \dots, h_k$

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

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

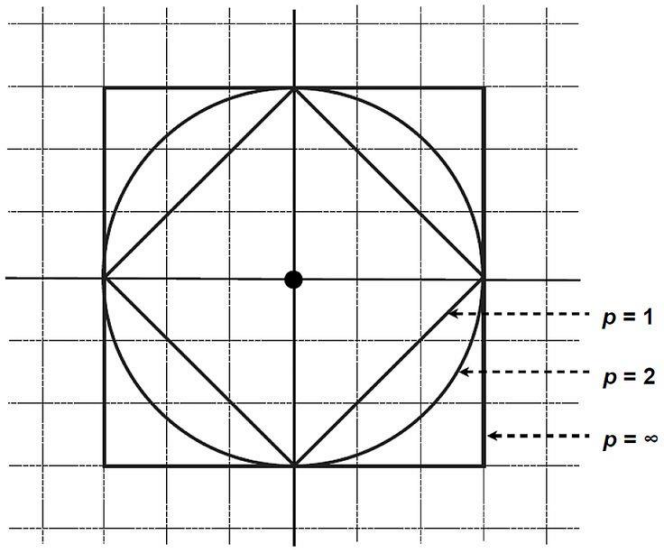
Given a set S, a function $d: S \times S \to \mathbb{R}$ is a **metric** or **distance function** if:

- $d(X, Y) \geq 0$ *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* $X, Y, Z \in S$

Which Distance Function

- Minkowski distance (p-norm)

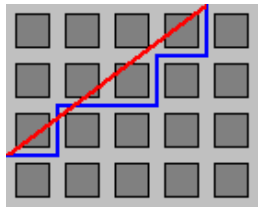
$$D(\mathbf{x}, \mathbf{x}') = \sqrt[p]{\sum_d |x_d - x'_d|^p}$$

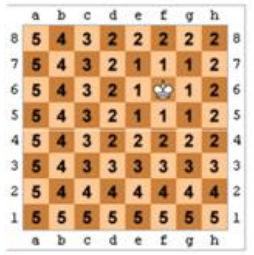


  - p=1: Manhattan (L1-norm, City-block)
  - p=2: Euclidian (L2-norm)
  - $p \to \infty$: Chebyshev (Lmax-norm), logical OR

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

  - $p \to 0$: logical AND



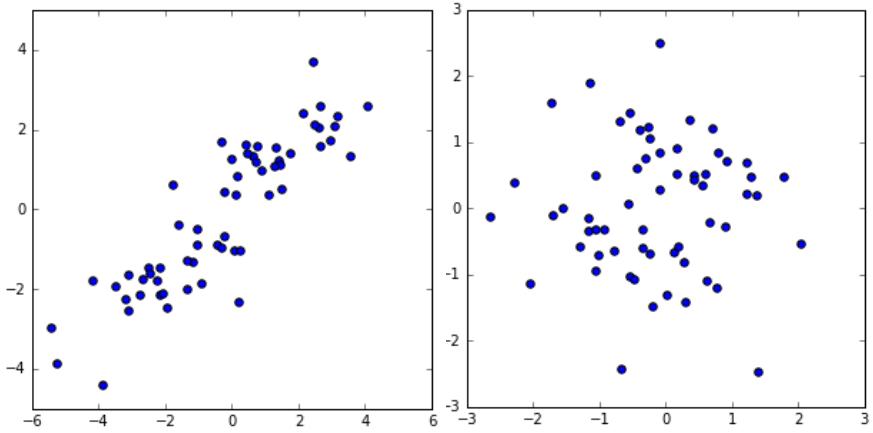


Mahalanobis distance

- 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)

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

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



Which Distance Function

- 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

Which Distance Function

- Kullback-Leibler (KL) divergence
  - Good for histograms and probability density functions ($x_d > 0, \sum_d x_d = 1$)

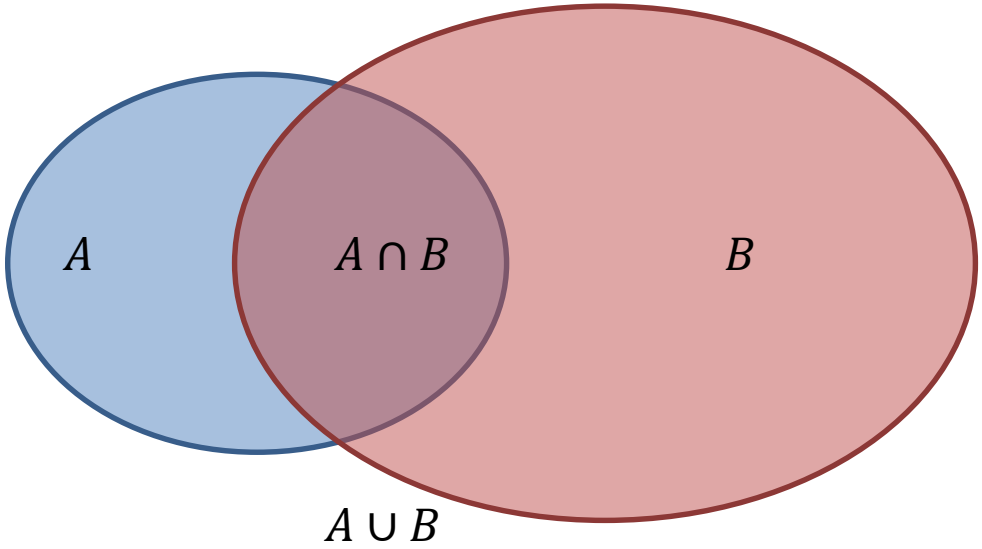
$$D(\mathbf{x}, \mathbf{x}') = -\sum_d x_d \log \frac{x_d}{x'_d}$$

  - Asymmetric function
  - A measure of information gain

Which Distance Function

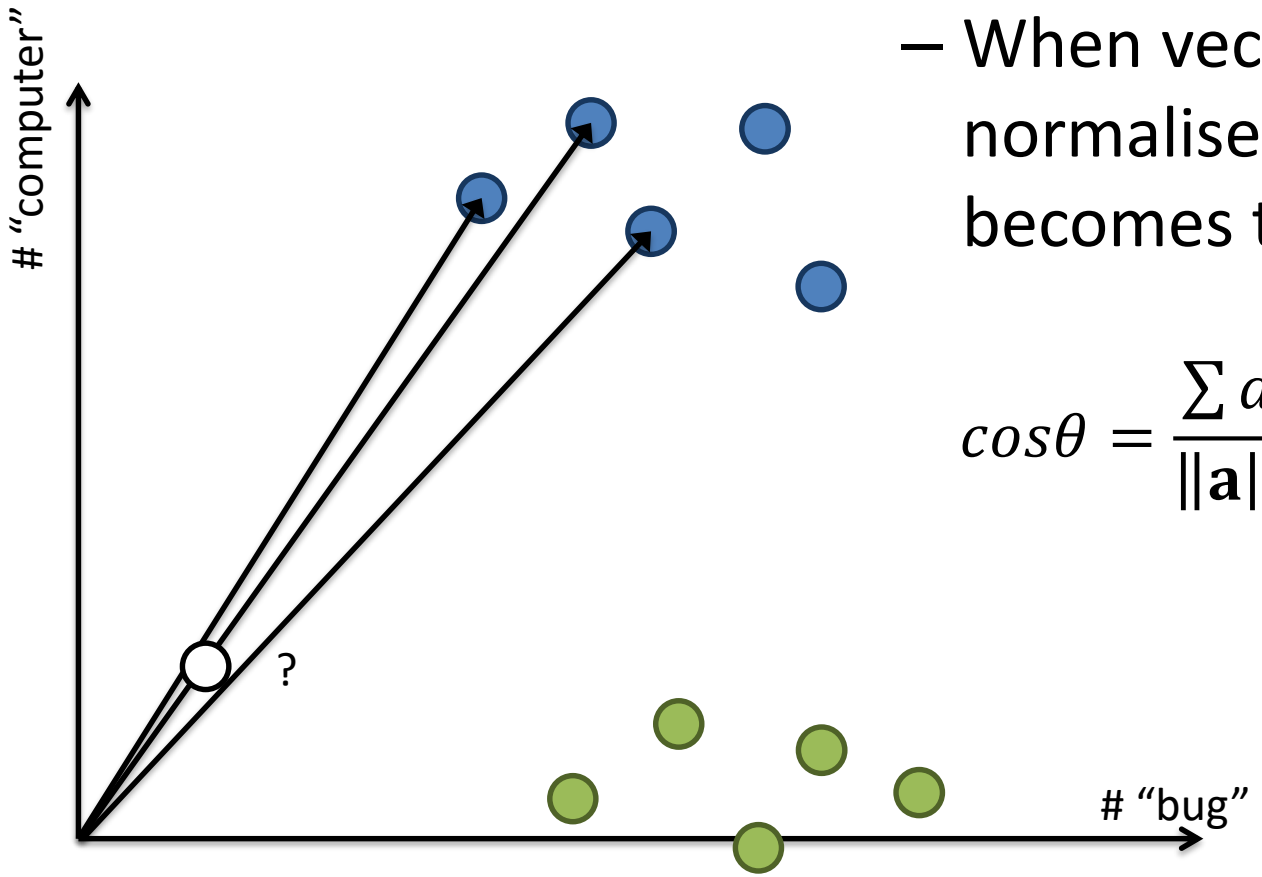
- Jaccard similarity (index)
  - Similarity between sets of things

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



Which Distance Function

Which Distance Function



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

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

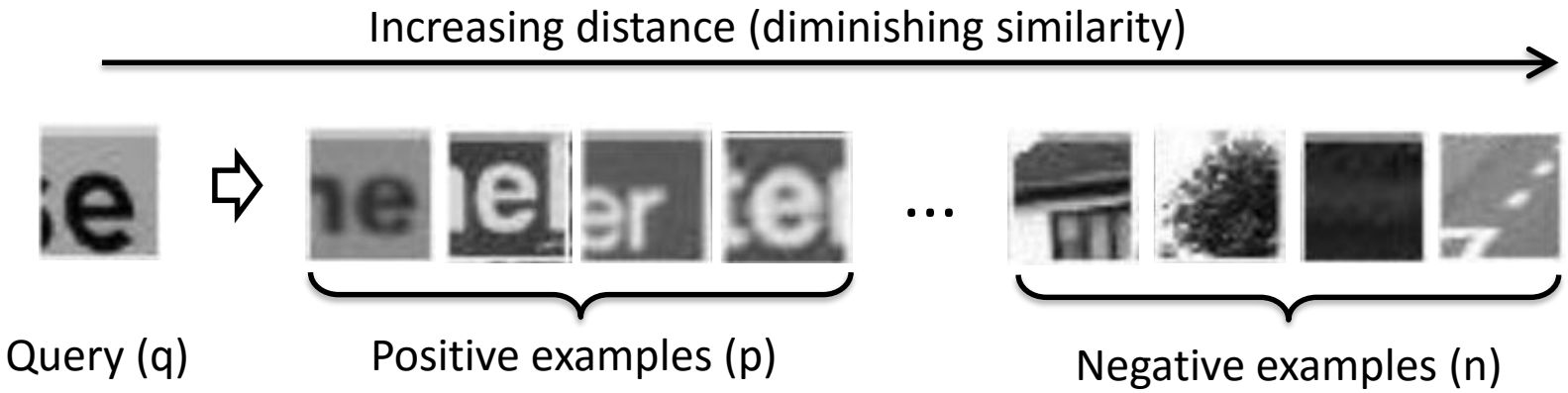
Which Distance Function

- 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
  - …

LEARNING THE DISTANCE METRIC

Supervised Metric Learning

If we use the "right" distance metric, then examples with the same label should be more similar than examples with a different label.



We would hope (assume a dot product distance metric) that:

- The similarity $S(\mathbf{q}, \mathbf{p}) = \mathbf{q}^T\mathbf{p}$ is high
- The similarity $S(\mathbf{q}, \mathbf{n}) = \mathbf{q}^T\mathbf{n}$ is low

Can we design a distance metric so that this actually happens?

Supervised Metric Learning

Select a family of distance / similarity functions. A common choice is a similarity following a bilinear form:

$$S(\mathbf{a}, \mathbf{b}) = \mathbf{a}^{\mathrm{T}}\mathbf{W}\mathbf{b} \qquad \mathbf{a}, \mathbf{b} \in \mathbb{R}^{\mathrm{D}}, \mathbf{W} \in \mathbb{R}^{D \times D}$$

Define the objective function (loss function) to optimise. In supervised semantic indexing, the following loss is defined given a set of queries ($\mathbf{q}$) and corresponding positive ($\mathbf{p}$) and negative ($\mathbf{n}$) examples:

$$L(W) = \sum_{\mathbf{q},\mathbf{p},\mathbf{n}} I[S(\mathbf{q}, \mathbf{p}) > S(\mathbf{q}, \mathbf{n})] \qquad I(a) = \begin{cases} 0 \text{ if } a = true \\ 1 \text{ if } a = false \end{cases}$$

Supervised Metric Learning

We need to estimate the parameters W that minimise the loss:

$$L(W) = \sum_{\mathbf{q},\mathbf{p},\mathbf{n}} I[S(\mathbf{q},\mathbf{p}) > S(\mathbf{q},\mathbf{n})]$$

This loss is difficult to minimise. We minimise instead a convex surrogate function (an upper bound), in this case a hinge loss:

$$L(W) = \sum_{\mathbf{q},\mathbf{p},\mathbf{n}} \max(0, 1 - \mathbf{qWp} + \mathbf{qWn})$$

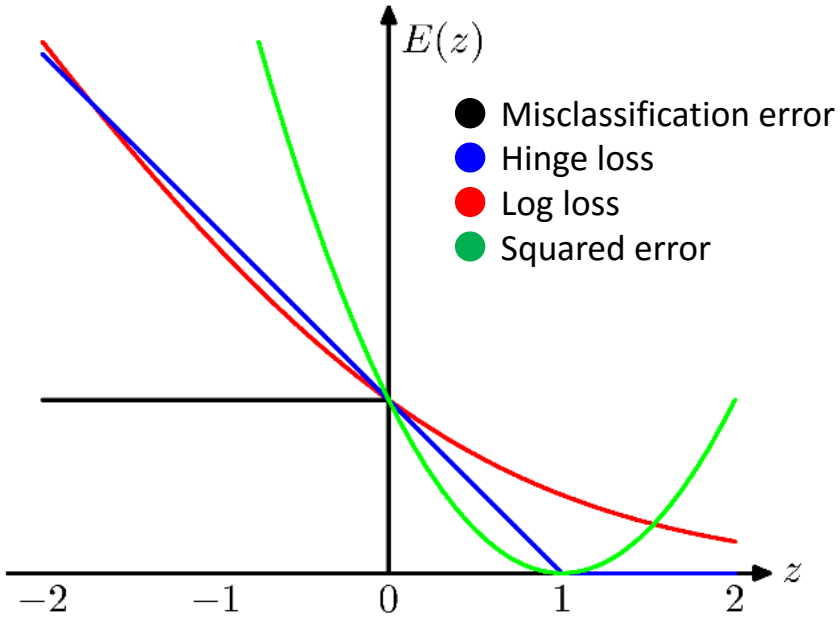


Figure from C. Bishop, "Pattern recognition and Machine Learning"

Supervised Metric Learning

A simple way to minimise this loss function is by using stochastic gradient descent:

- Sample a query ($\mathbf{q}$) example randomly, and then a positive ($\mathbf{p}$) and a negative ($\mathbf{n}$) sample subject to the query ($\mathbf{q}$)
- Perform a gradient step update using this set of $\{\mathbf{q}, \mathbf{p}, \mathbf{n}\}$

The update rule in this case is:

$$\mathbf{W} \leftarrow \mathbf{W} - \lambda \left. \frac{\partial f}{\partial \mathbf{W}} \right|_{(\mathbf{q},\mathbf{p},\mathbf{n})}$$

Which evaluates to:

$$\begin{cases} \mathbf{W} & \text{if } S(q,p) > S(q,n) + 1 \\ \mathbf{W} + \lambda \mathbf{q}(\mathbf{p} - \mathbf{n})^T & \text{otherwise} \end{cases}$$

Other Metric learning Methods

- Metric learning for nearest neighbour search
  (Weinberger and Saul, JMLR 2009)
- Information-theoretic metric learning
  (Davies et al., ICML, 2007)
- OASIS
  (Chechik et al., JMLR 2011)
- Neighbourhood “gerrymandering”
  (NIPS 2014)
- …

In general, any combination of similarity + loss (+ regularization) = different metric learning method

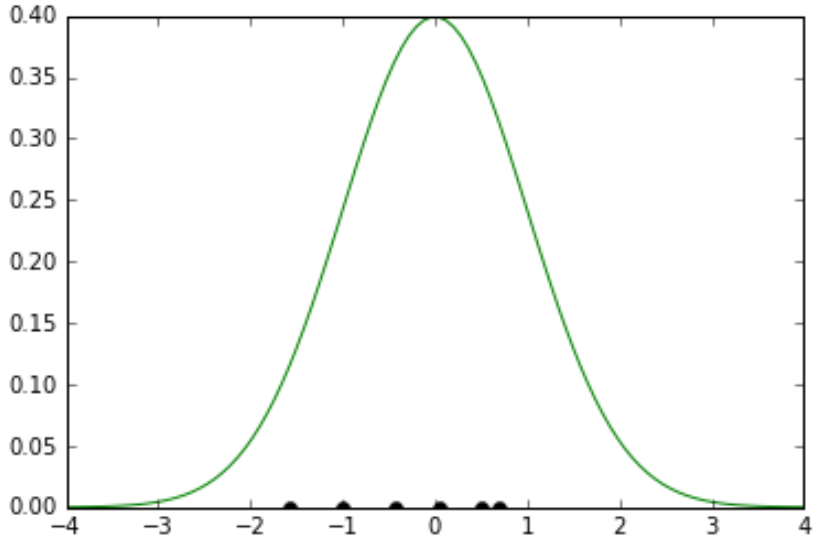
PROBABILITY DENSITY ESTIMATION: KNN, PARZEN WINDOWS, KERNELS

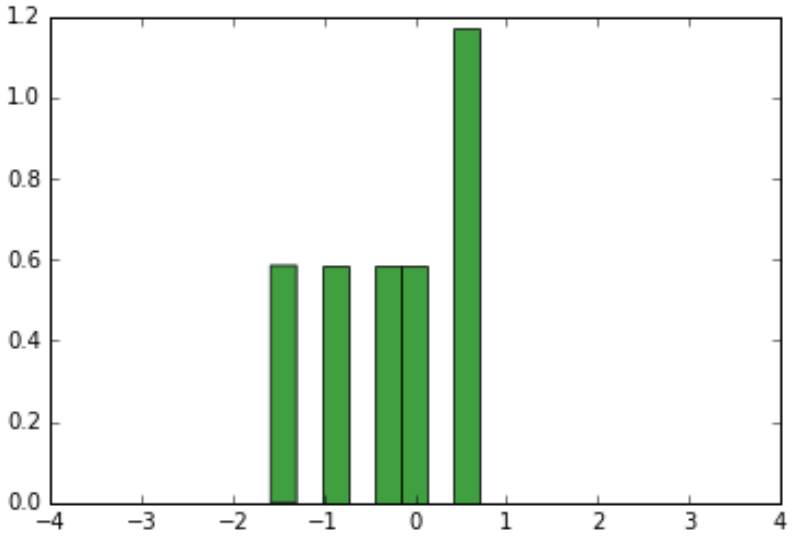
Finding the Location

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.

Probability Density Estimation

Given an independent and identically distributed sample $(x_1, x_2, \ldots, x_n)$ drawn from some distribution with an unknown density $p(x)$, we are interested in estimating the shape of this density function $p(x)$





Probability Density Estimation

Given a small region R with volume V containing x the probability mass is given by:

$$P = \int_R p(\mathbf{x}) d\mathbf{x}$$

The probability of K out of N observations falling within a within R (with volume V) is given by the binomial distribution

$$Bin(K|N,P) = \frac{N!}{K!\,(N-K)!} P^K (1-P)^{N-K}$$

For large N, this distribution will be sharply peaked around the mean and so:

$$K \simeq NP$$

If we also assume that R is sufficiently small then the probability density is roughly constant in R:

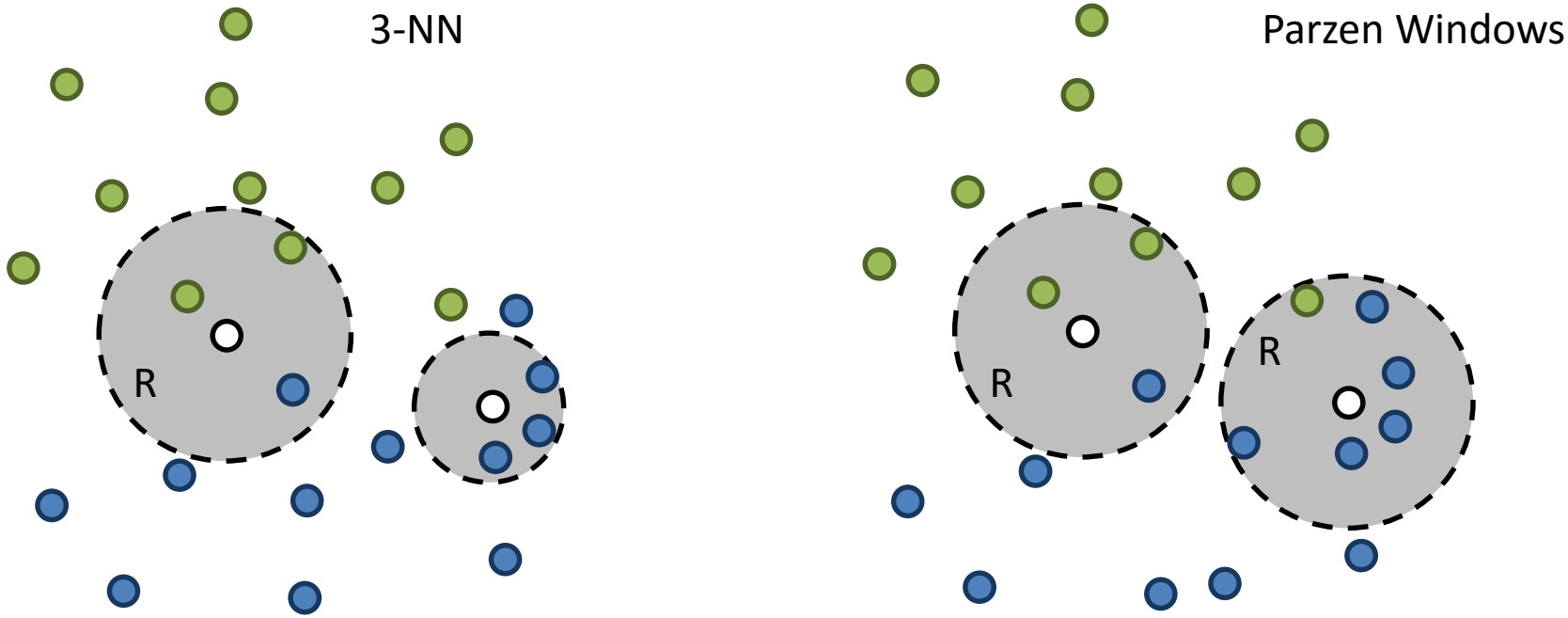
$$P \simeq p(\mathbf{x})V$$

Combining, we obtain:

$$P(x) = \frac{K}{NV}$$

This depends on two contradictory assumptions: small R vs large K.

Parzen Windows and Kernels



We can

- Fix K and determine the value of V from the data, which gives rise to the K-NN technique
- Fix V and determine K from the data, giving rise to Parzen windows the kernel approach

Parzen Windows and Kernels

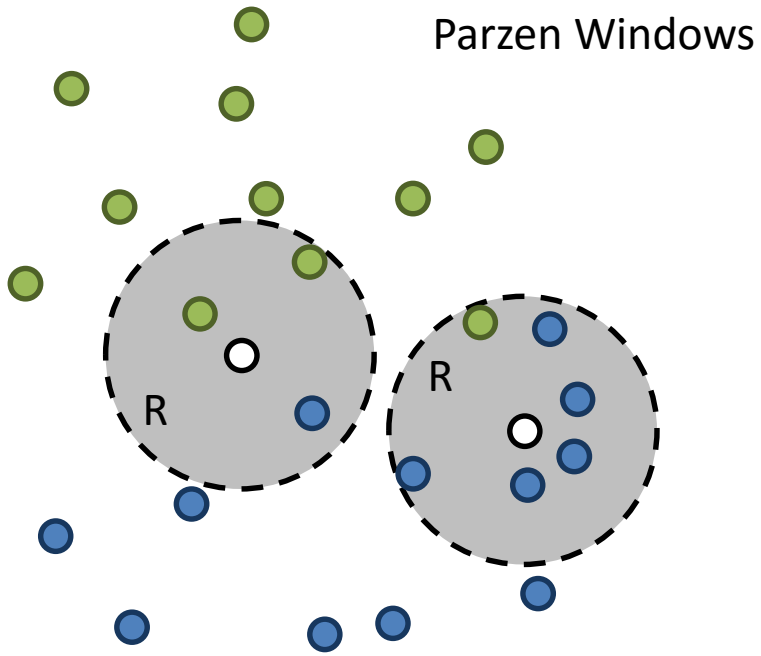
$$f(x) = sgn\left[\sum_{i:x_i \in R} y_i\right]$$

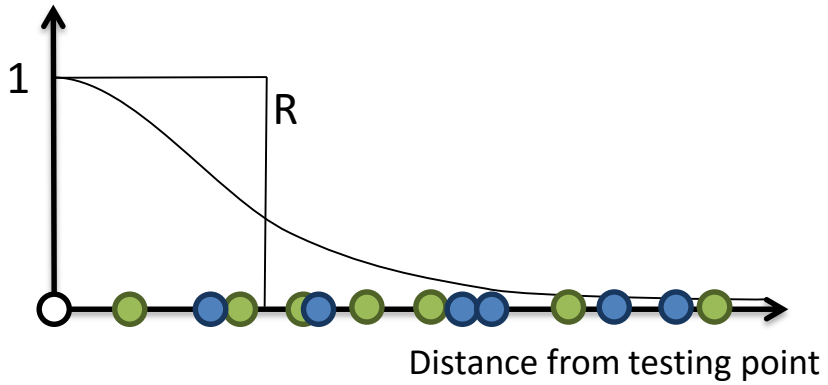
$$= sgn\left[\sum_{i} y_i \cdot 1_{\|x_i - x\| \le R}\right]$$

$$= sgn\left[\sum_{i} y_i K(x_i, x)\right]$$

$y = +1$

$y = -1$





Kernel Density Estimation

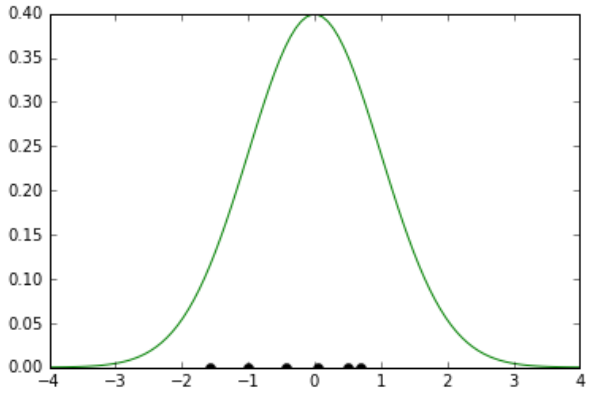
The kernel density estimator of $p(\mathbf{x})$ is:

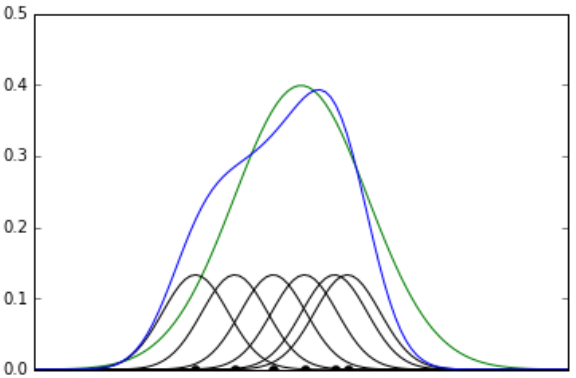
$$\hat{p}_h(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\mathbf{x} - \mathbf{x}_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

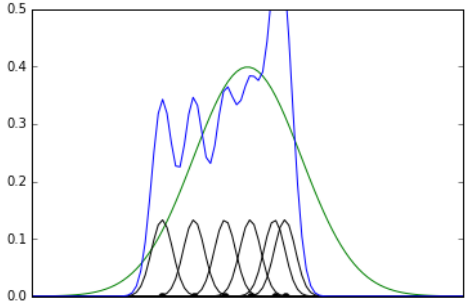
Where $K(\cdot)$ is the kernel – a non-negative function that integrates to one and has mean zero and $h > 0$ is a smoothing parameter called the bandwidth.

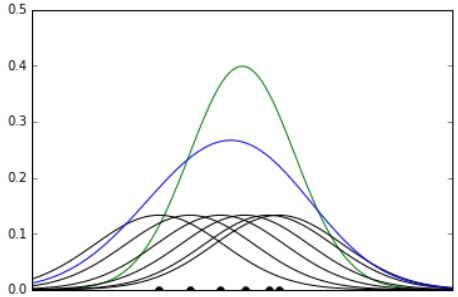
A Kernel with a subscript h is called the scaled kernel and is defined as: $K_h(x) = {}^1/_h\, K({}^x/_h)$

The bandwidth of the kernel is a free parameter which exhibits a strong influence on the resulting estimate









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

What's Next

		Mondays	Tuesdays				
		16:00 - 18:00	15:00 - 17:00				
Practical Sessions		**M**	**T**	**W**	**T**	**F**	**Lectures**
	Feb	8	9	10	11	12	Introduction and Linear Regression
P0. Introduction to Python, Linear Regression		15	16	17	18	19	Logistic Regression, Normalization
P1. Text non-text classification (Logistic Regression)		22	23	24	25	26	Regularization, Bias-variance decomposition
	Mar	29	1	2	3	4	Subspace methods, dimensionality reduction
		7	8	9	10	11	Probabilities, Bayesian inference
Discussion of intermediate deliverables / project presentations		14	15	16	17	18	Parameter Estimation, Bayesian Classification
		21	22	23	24	25	Easter Week
	Apr	28	29	30	31	1	Clustering, Gausian Mixture Models, Expectation Maximisation
P2. Feature learning (k-means clustering, NN, bag of words)		4	5	6	7	8	Nearest Neighbour Classification
		11	12	13	14	15	
		18	19	20	21	22	Kernel methods
Discussion of intermediate deliverables / project presentations		25	26	27	28	29	Support Vector Machines, Support Vector Regression
P3. Text recognition (multi-class classification using SVMs)	**May**	2	3	4	5	6	Neural Networks
		9	10	11	12	13	Advanced Topics: Deep Nets
		16	17	18	19	20	Advanced Topics: Metric Learning, Preference Learning
Final Project Presentations		23	24	25	26	27	Advanced Topics: Structural Pattern Recognition
	Jun	30	31	1	2	3	Revision

LEGEND	
	Project Follow Up
	Project presentations
	Lectures
	Project Deliverable due date
	Vacation / No Class