K-Nearest Neighbours

Adapted from material by

Ata Kaban

The University of Birmingham

This section we will learn

- K-Nearest Neighbours
- Lazy and eager learning

Instance-based learning

- One way of solving tasks of approximating discrete or real valued target functions
- Have training examples: $(x_n, f(x_n))$, $n=1..N$.
- Key idea:
  - just store the training examples
  - when a test example is given then find the closest matches

- 1-Nearest neighbour:

  Given a query instance x_q,

  - first locate the nearest training example x_n
  - then $f(x_q) := f(x_n)$

- K-Nearest neighbour:

  Given a query instance x_q,

  - first locate the k nearest training examples
  - if discrete values target function then take vote among its k nearest neighbours
    else if real valued target function then take the mean of the f values of the k nearest neighbours

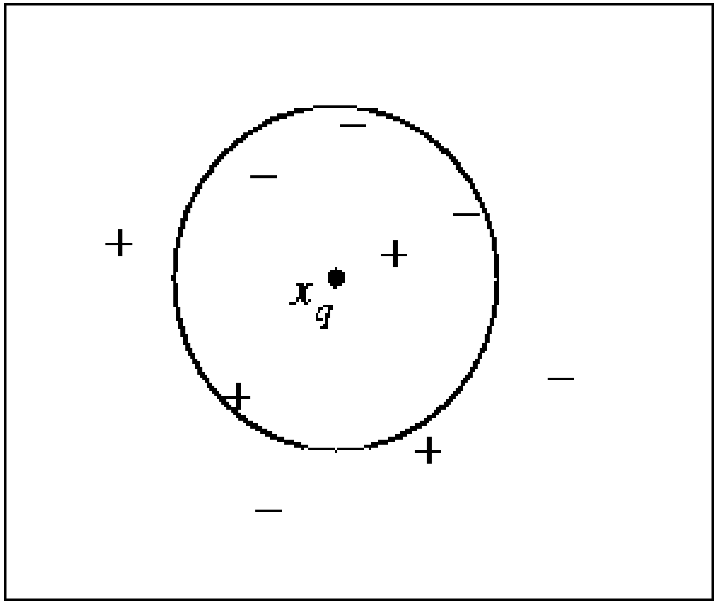
$$f(x_q) := \frac{\sum f(x_i)}{k}$$

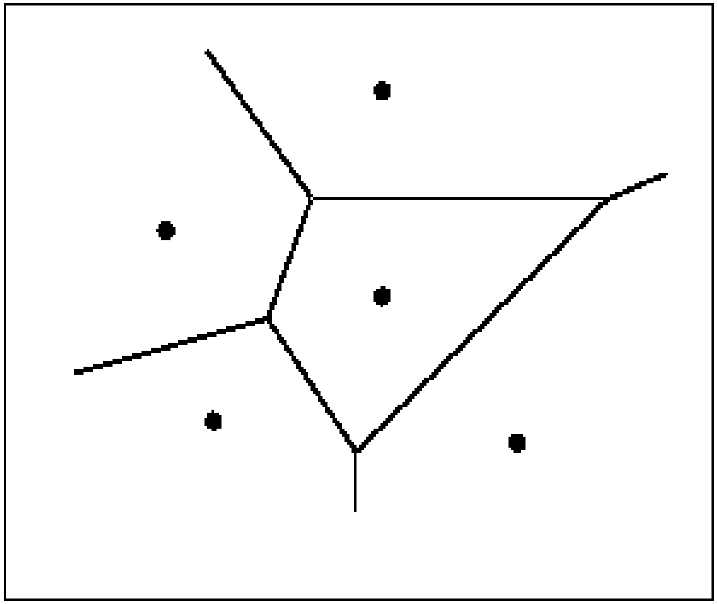
The distance between examples

- We need a measure of distance in order to know who are the neighbours
- Assume that we have T attributes for the learning problem. Then one example point $\boldsymbol{x}$ has elements $x_t \in \Re$, $t=1,\ldots T$.
- The distance between two points $\boldsymbol{x_i}$ $\boldsymbol{x_j}$ is often defined as the Euclidean distance:

$$d(x_i, x_j) = \sqrt{\sum_{t=1}^{T} [x_{ti} - x_{tj}]^2}$$

Voronoi Diagram



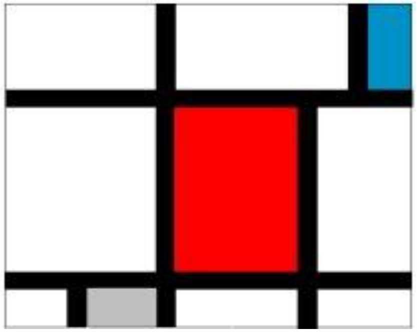


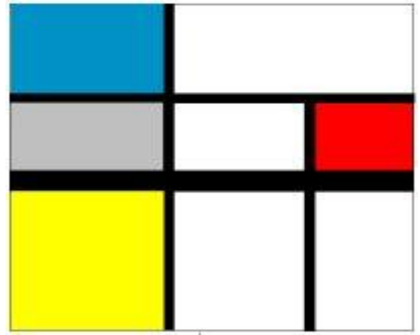
Characteristics of Instance-based-Learning

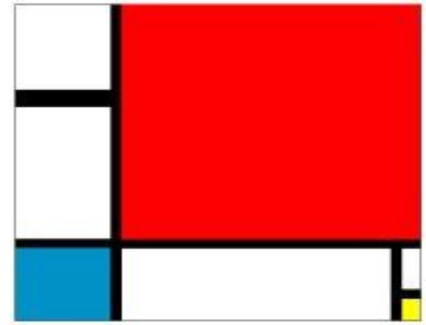
- An instance-based learner is a *lazy-learner* and does all the work when the test example is presented. This is opposed to so-called *eager-learners*, which build a parameterised compact model of the target.
- It produces *local* approximation to the target function (*different* with each test instance)

When to consider Nearest Neighbour algorithms?

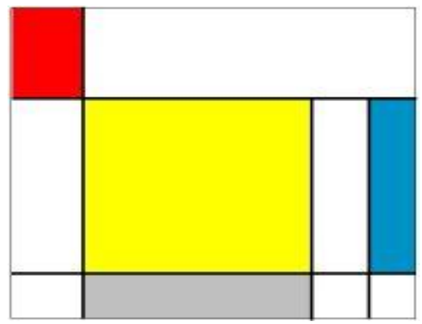
- Instances map to points in $\Re^n$
- Not more then say 20 attributes per instance
- Lots of training data
- Advantages:
  - Training is very fast
  - Can learn complex target functions
  - Don't lose information
- Disadvantages:
  - ? (will see them shortly…)

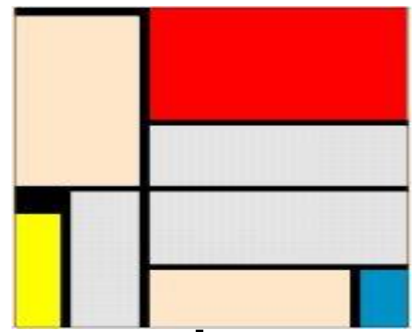

one

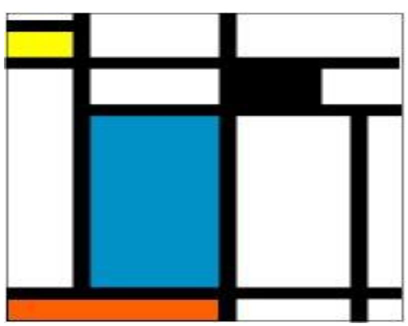

two

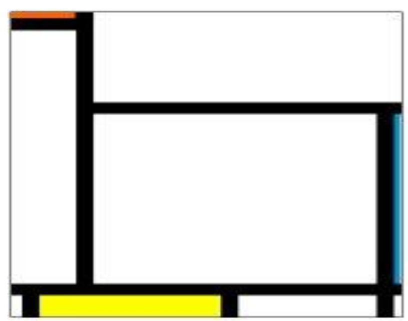

three


four


five


six


seven


Eight ?

Training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	6	1	10	4	*No*
2	4	2	8	5	*No*
3	5	2	7	4	Yes
4	5	1	8	4	Yes
5	5	1	10	5	*No*
6	6	1	8	6	Yes
7	7	1	14	5	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	7	2	9	4	

Keep data in normalised form

One way to normalise the data $a_t(x)$ to $a'_t(x)$ is

$$x_t' = \frac{x_t - \bar{x}_t}{\sigma_t}$$

$$\bar{x}_t = mean \quad of \quad t^{th} \quad attributes$$

$$\sigma_t = standard \quad deviation \quad of \quad t^{th} \quad attribute$$

To ensure equal effects from each of the feature

Normalised training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	0.632	-0.632	0.327	-1.021	*No*
2	-1.581	1.581	-0.588	0.408	*No*
3	-0.474	1.581	-1.046	-1.021	Yes
4	-0.474	-0.632	-0.588	-1.021	Yes
5	-0.474	-0.632	0.327	0.408	*No*
6	0.632	-0.632	-0.588	1.837	Yes
7	1.739	-0.632	2.157	0.408	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	1.739	1.581	-0.131	-1.021	

Distances of test instance from training data

Example	Distance of test from example	Mondrian?
1	2.517	*No*
2	3.644	*No*
3	2.395	Yes
4	3.164	Yes
5	3.472	*No*
6	3.808	Yes
7	3.490	*No*

Classification

1-NN	Yes
3-NN	Yes
5-NN	*No*
7-NN	*No*

What if the target function is real valued?

- The k-nearest neighbour algorithm would just calculate the mean of the k nearest neighbours

Variant of kNN: Distance-Weighted kNN

- We might want to weight nearer neighbours more heavily

$$f(x_q) := \frac{\sum w_i f(x_i)}{\sum w_i} \quad \text{where} \quad w_i = \frac{1}{d(x_q, x_i)^2}$$

- Then it makes sense to use *all training examples* instead of just k (Stepard's method)

Difficulties with k-nearest neighbour algorithms

- Have to calculate the distance of the test case from *all* training cases
- There may be irrelevant attributes amongst the attributes – *curse of dimensionality*

Lazy and Eager Learning

- Lazy: wait for query before generalizing
  - k-Nearest Neighbour, Case based reasoning
- Eager: generalize before seeing query
  - Radial Basis Function Networks, ID3, …
- Does it matter?
  - Eager learner must create global approximation
  - Lazy learner can create many local approximations

Summary

- K-Nearest Neighbours
- Lazy and eager learning