CS760

Instance-Based Learning

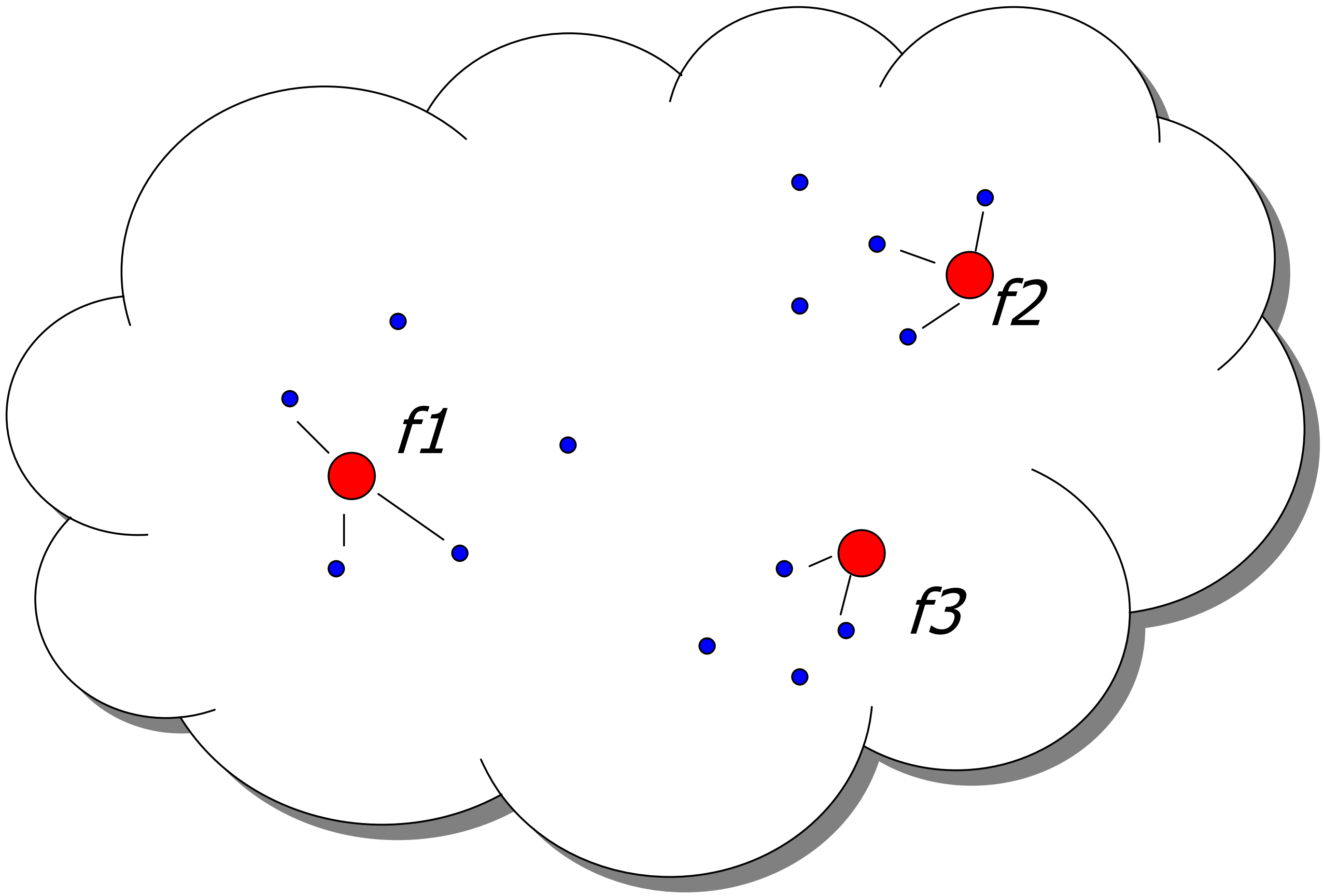
Dr. Ian Watson

 www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

Instance-based learners

- store *all* the training data
- when a new query instance is encountered, a set of related instances are retrieved from memory and used to classify the instance
- can construct a different approximation function of the target function for each distinct query instance

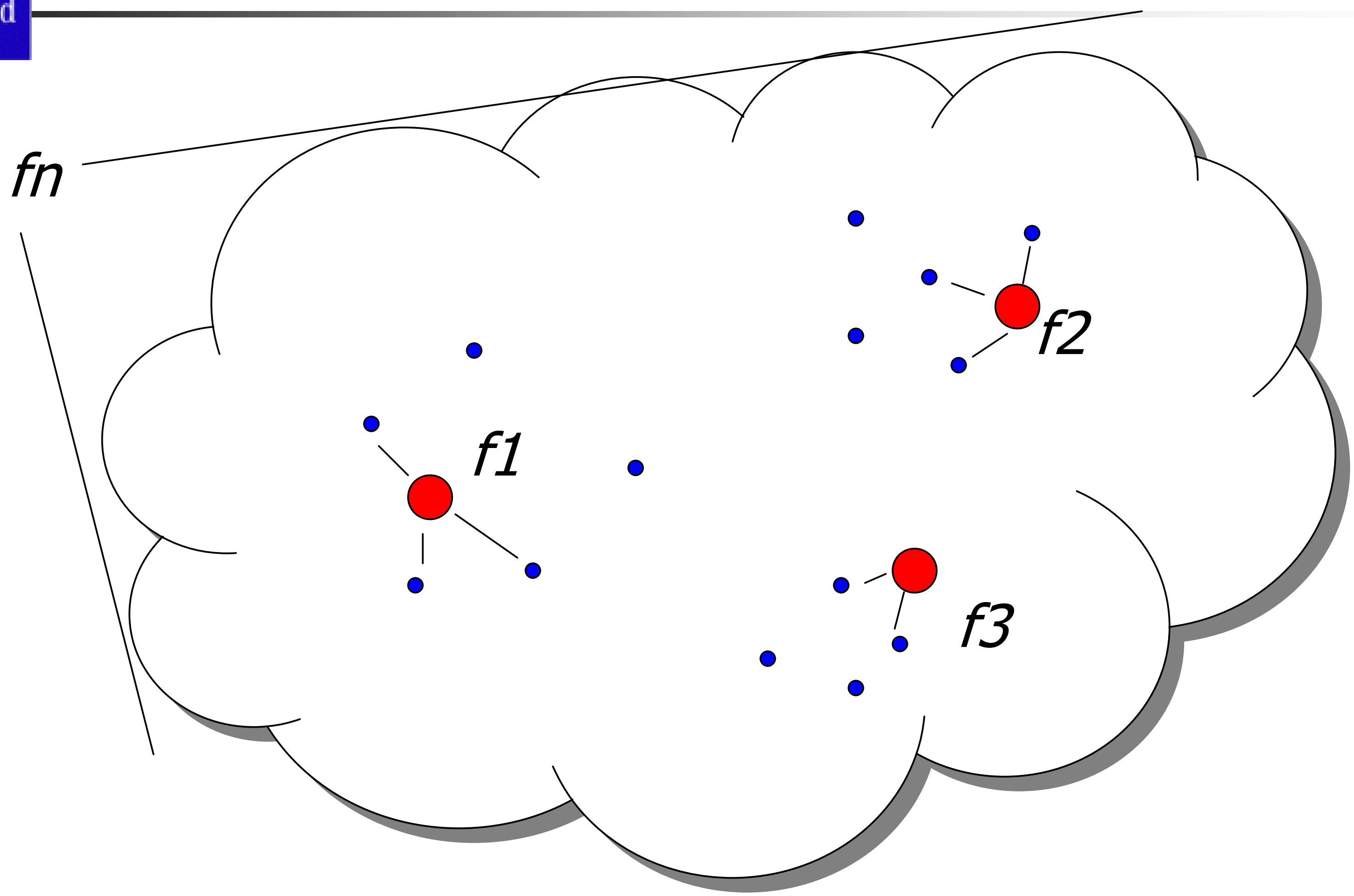
Instance-based learners



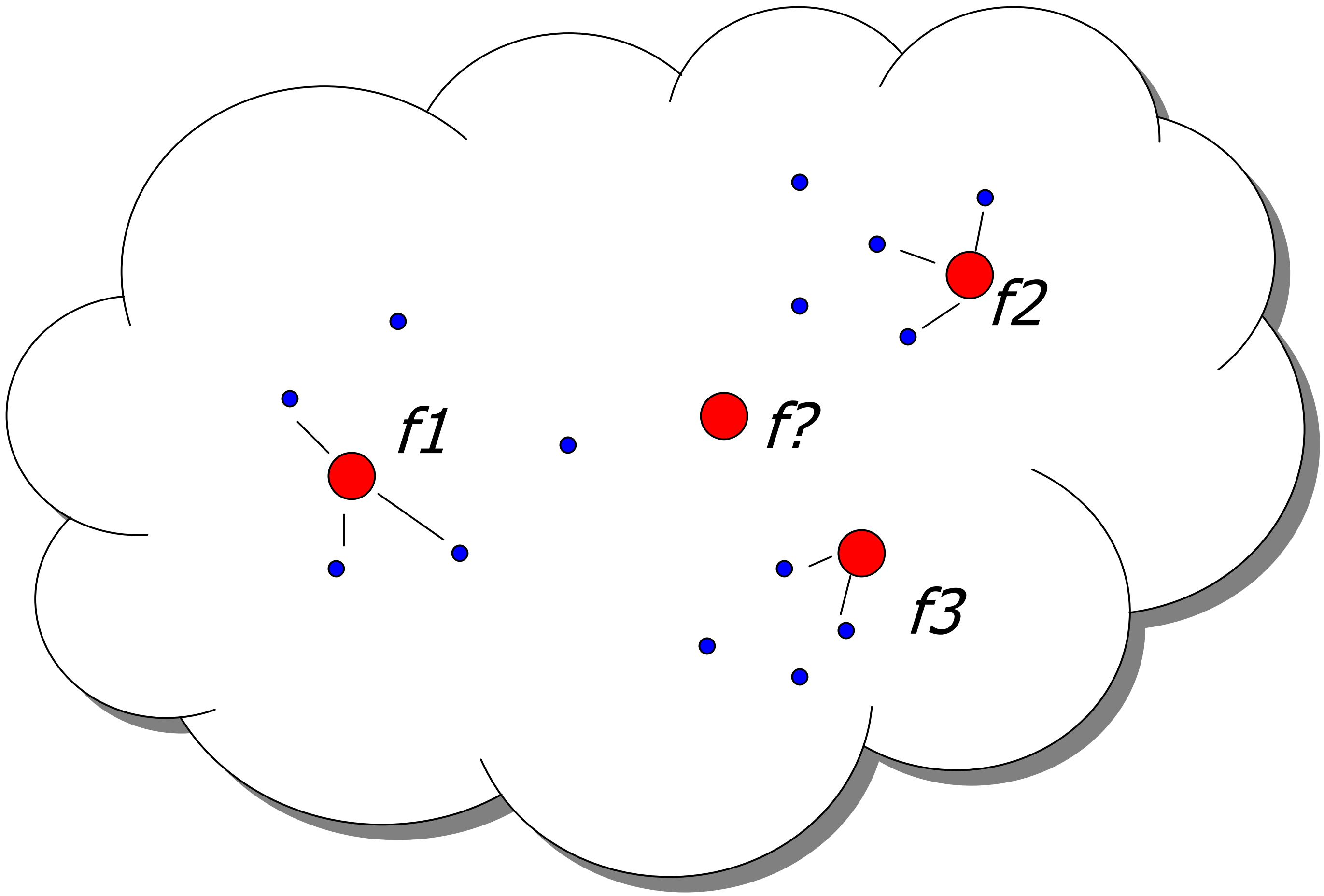
Instance-based learners

- significant advantage
- when the target function is very complex
- but can be described by a collection of less complex local approximations

Instance-based learners



Instance-based learners



  ian@cs.auckland.ac.nz

Instance-based learners

- Disadvantages
  - cost of classifying new instances can be high, so efficiently indexing training instances very important
  - uses all/many attributes in determining similar training instances...so irrelevant or redundant attributes are a problem

Instance-based learners

- Algorithms
  - Nearest neighbour (k-NN)
  - Locally weighted regression
  - Radial Bias functions (used in ANNs)

Instance-based learners

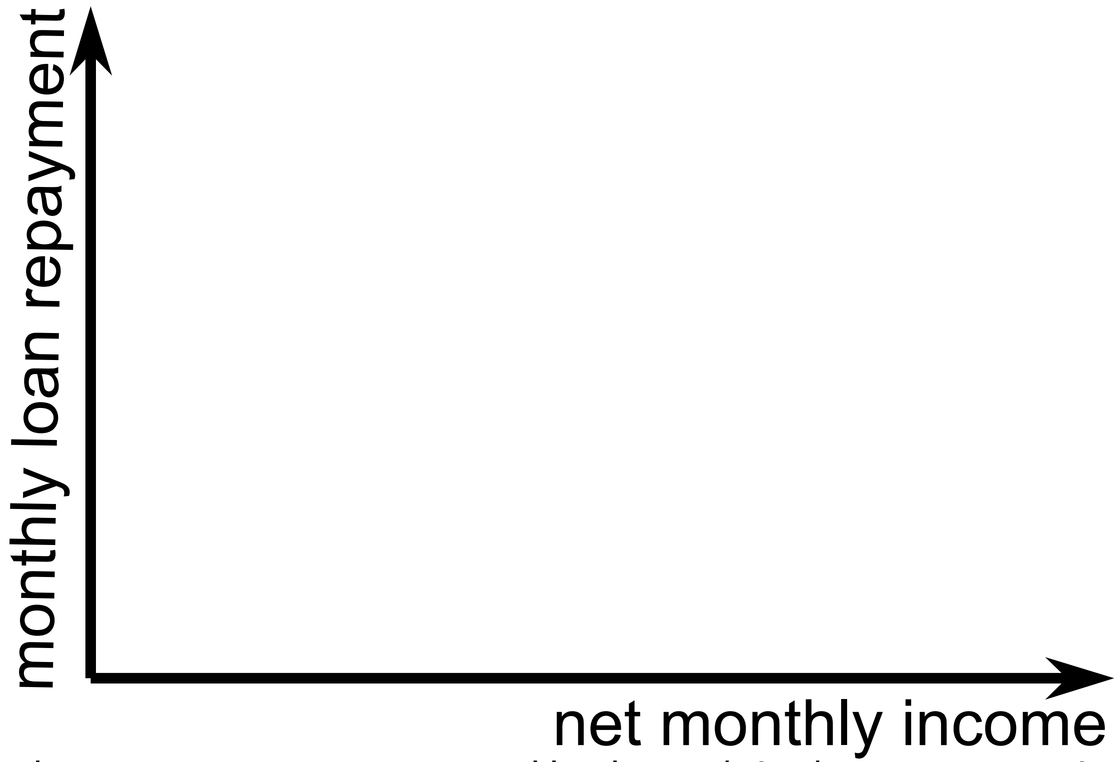
- Nearest neighbour (k-NN)
  - most basic method - all instances are points in an *n*-dimensional space
  - distance is defined as standard Euclidean distance
  - K-NN finds the *nearest* neighbours to a query in the *n*-dimensional space
  - values may be discrete or real

Nearest Neighbour

- imagine a decision with two factors that influence it
- should you grant a person a loan?
  ❶ net monthly income
  ❷ monthly loan repayment

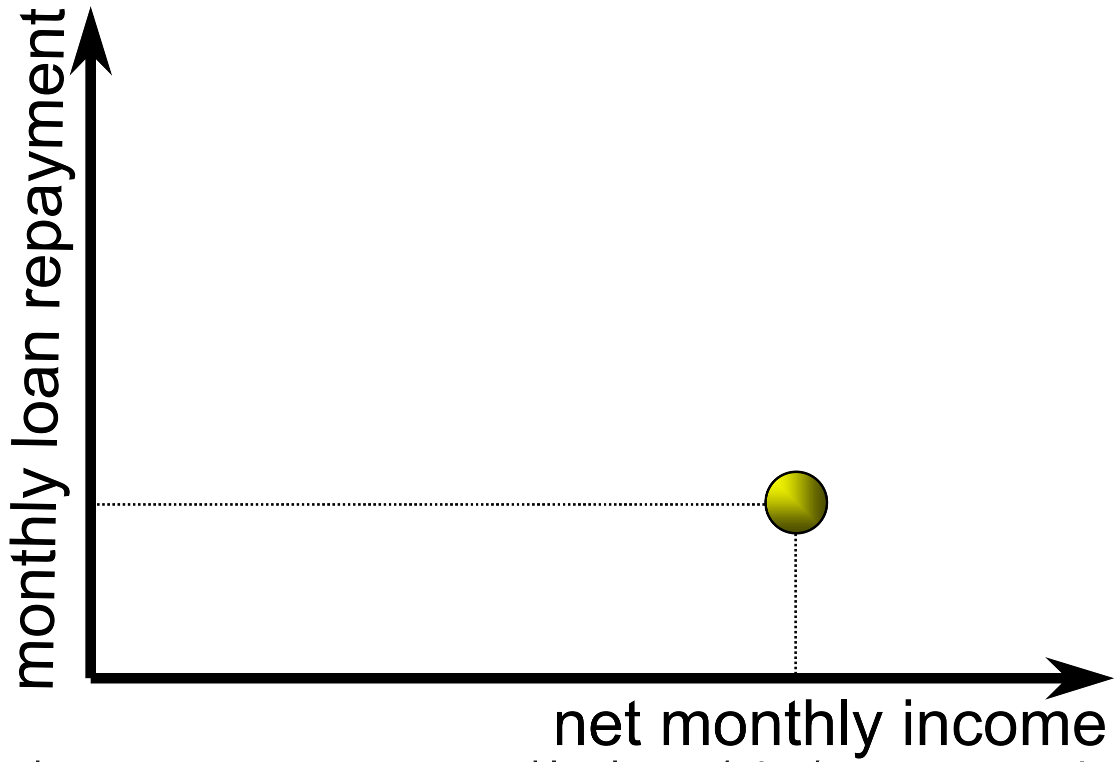
Nearest Neighbour

- these factors can be used as axes for a graph



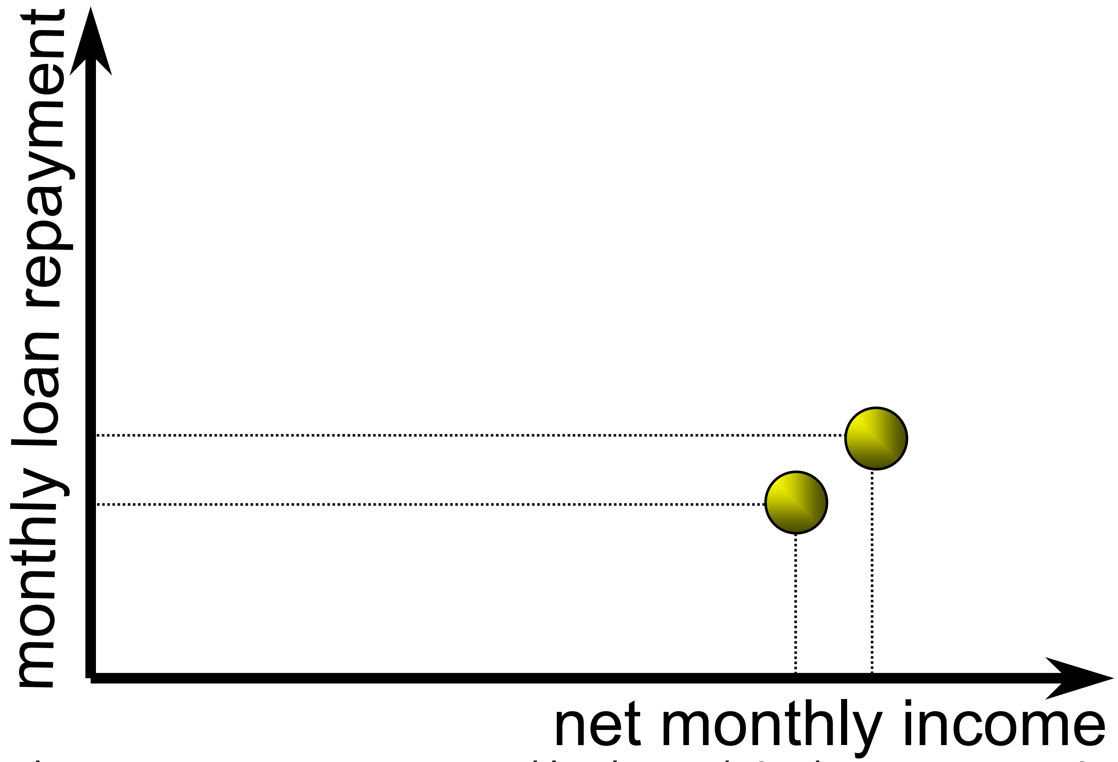
Nearest Neighbour

- a previous loan can be plotted against these axes



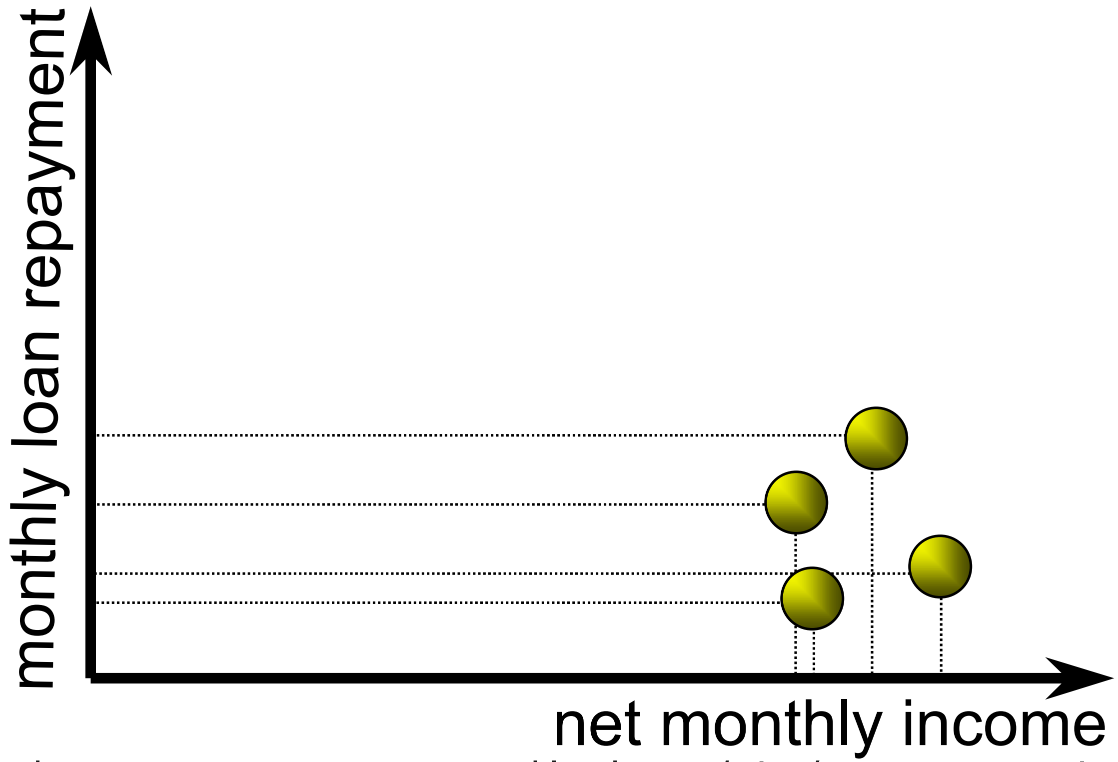
Nearest Neighbour

- and a second loan



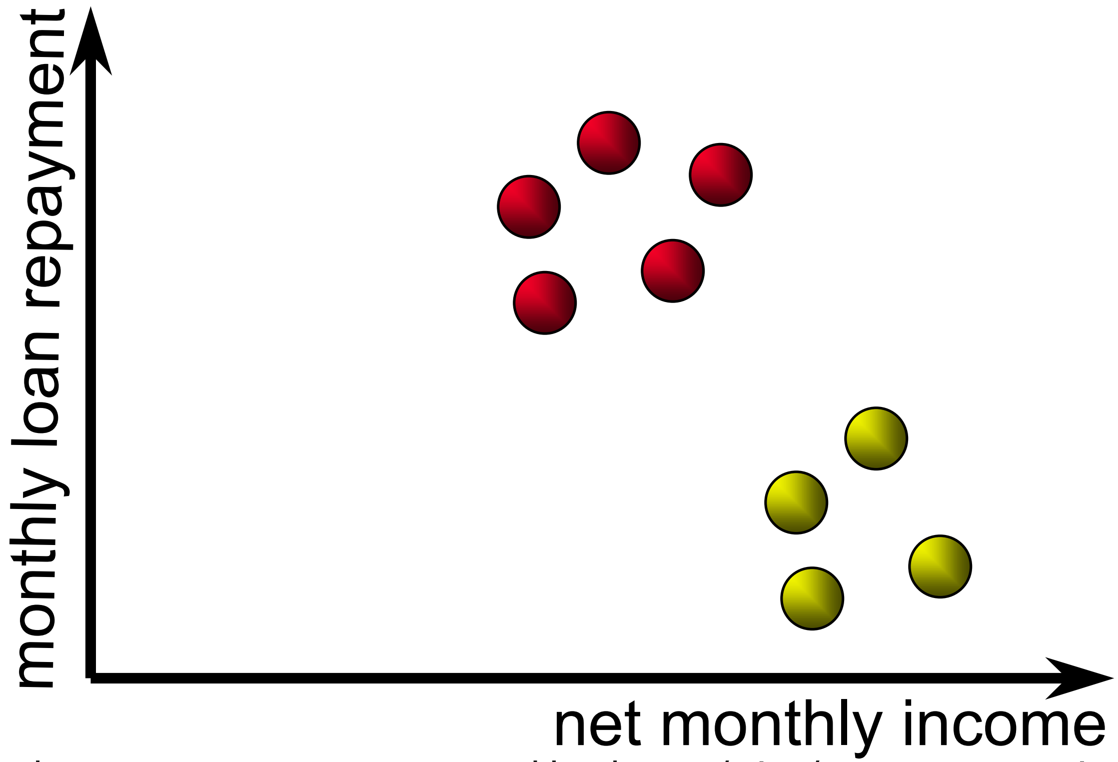
Nearest Neighbour

- and more loans



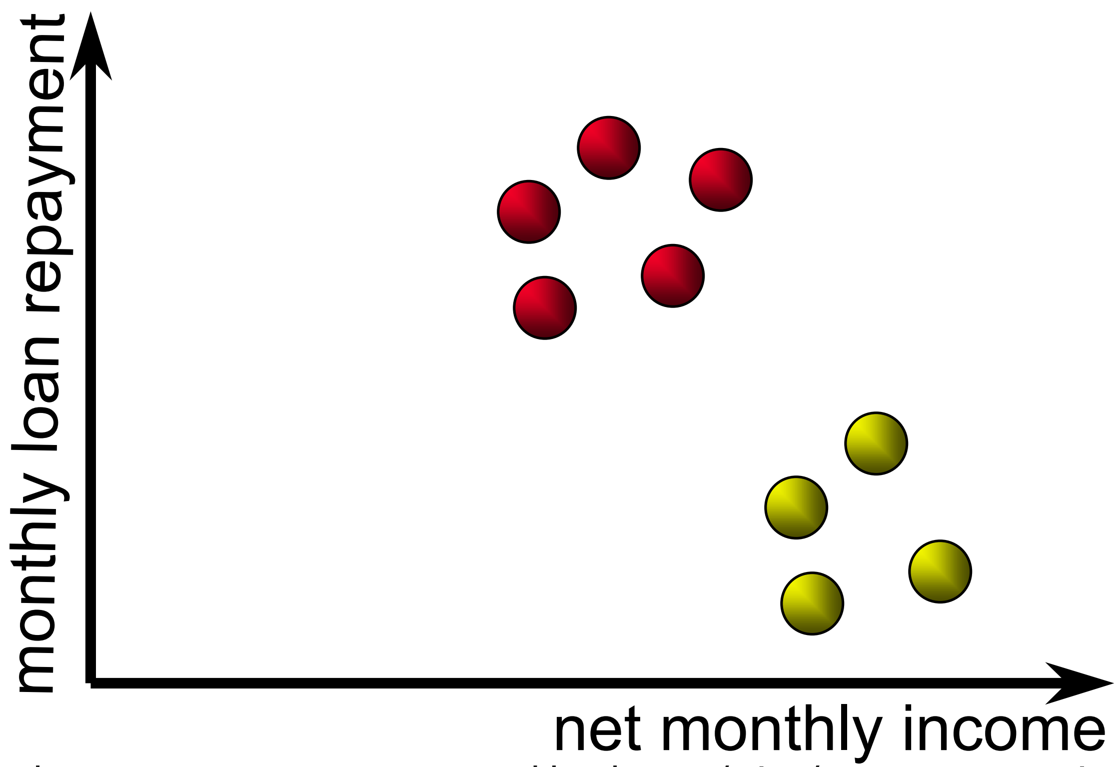
Nearest Neighbour

- and even more loans



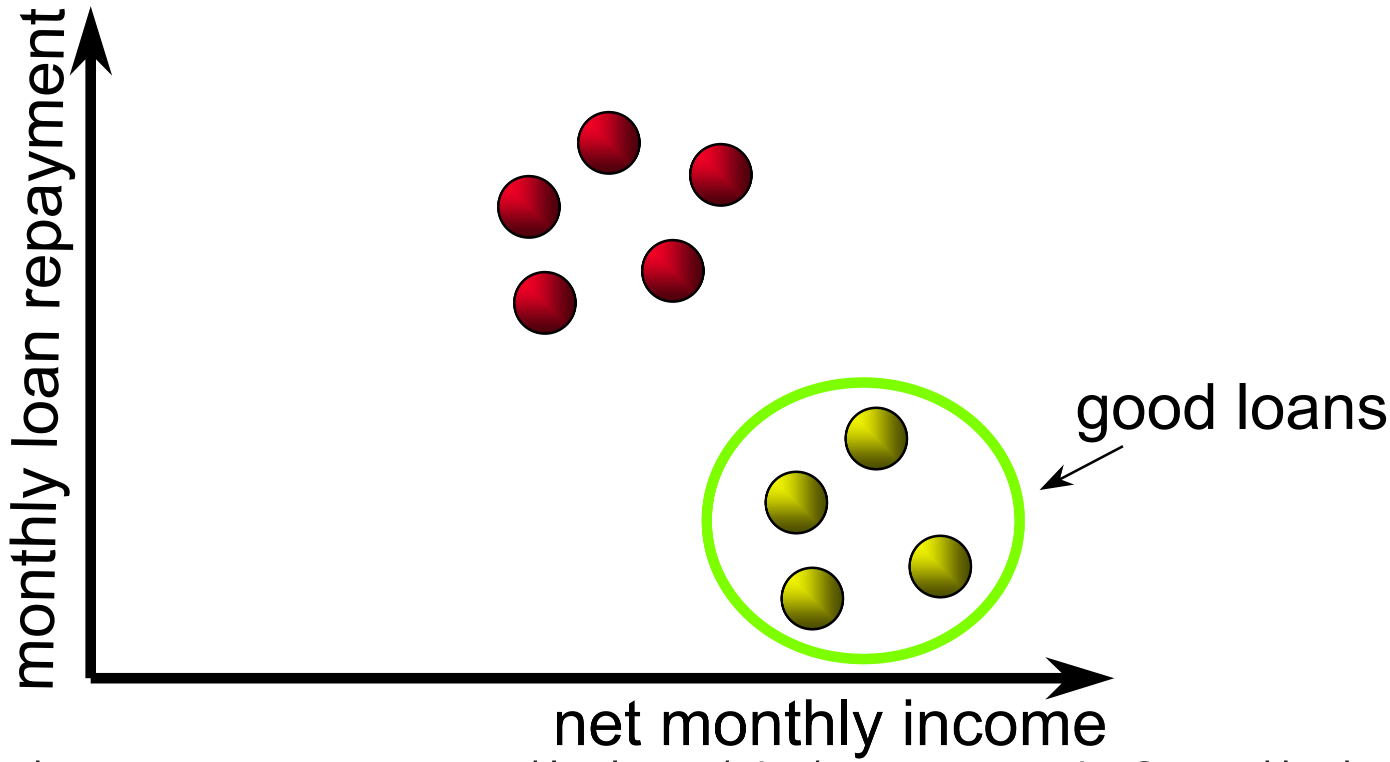
Nearest Neighbour

- past cases (loans) may form clusters



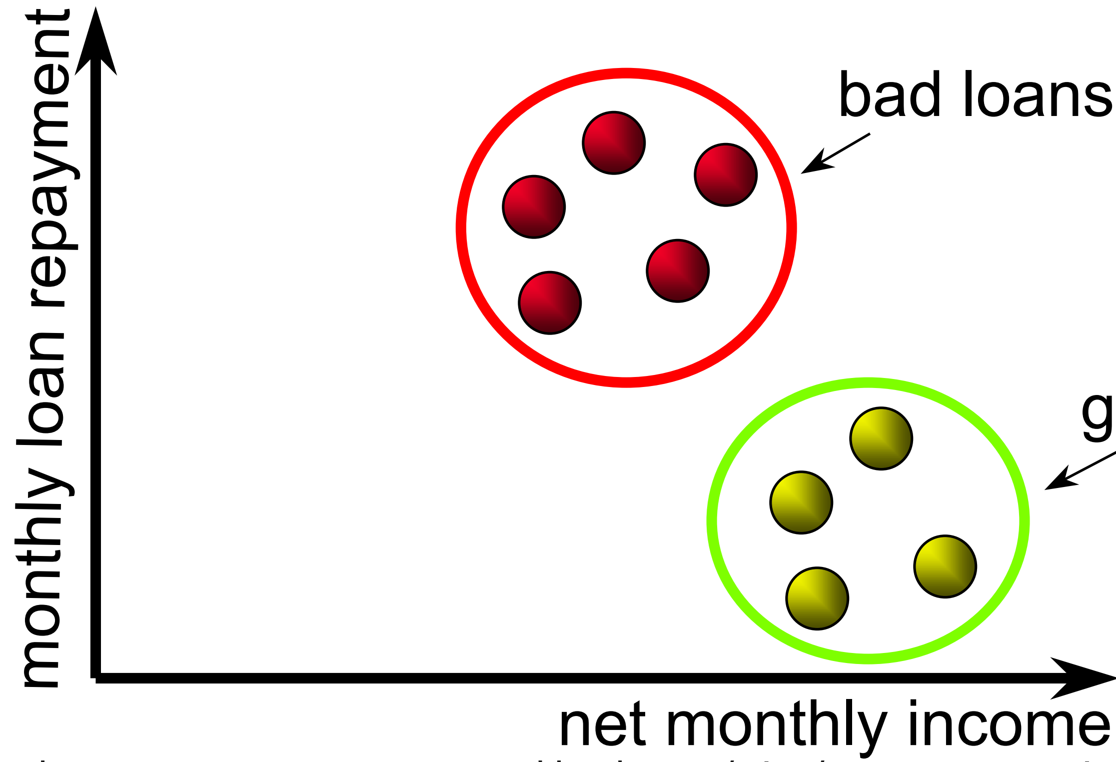
Nearest Neighbour

- past cases (loans) may tend to form clusters



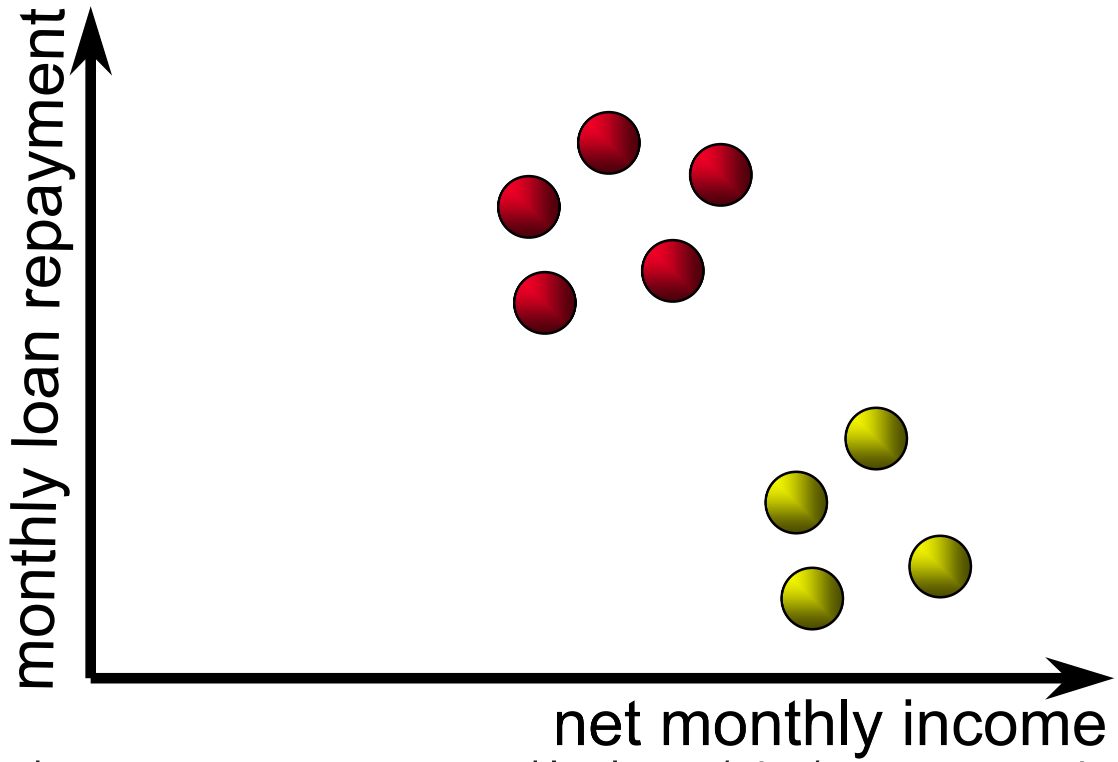
Nearest Neighbour

- past cases (loans) may tend to form clusters



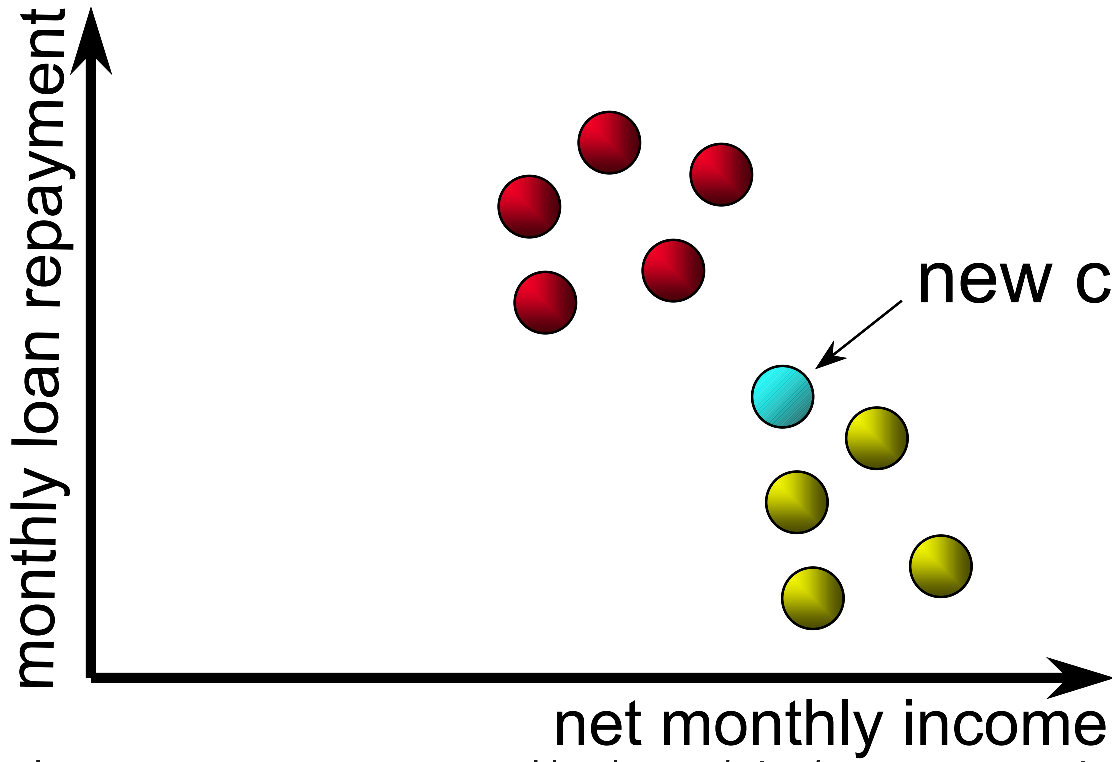
Nearest Neighbour

- a new loan prospect can be plotted on the graph



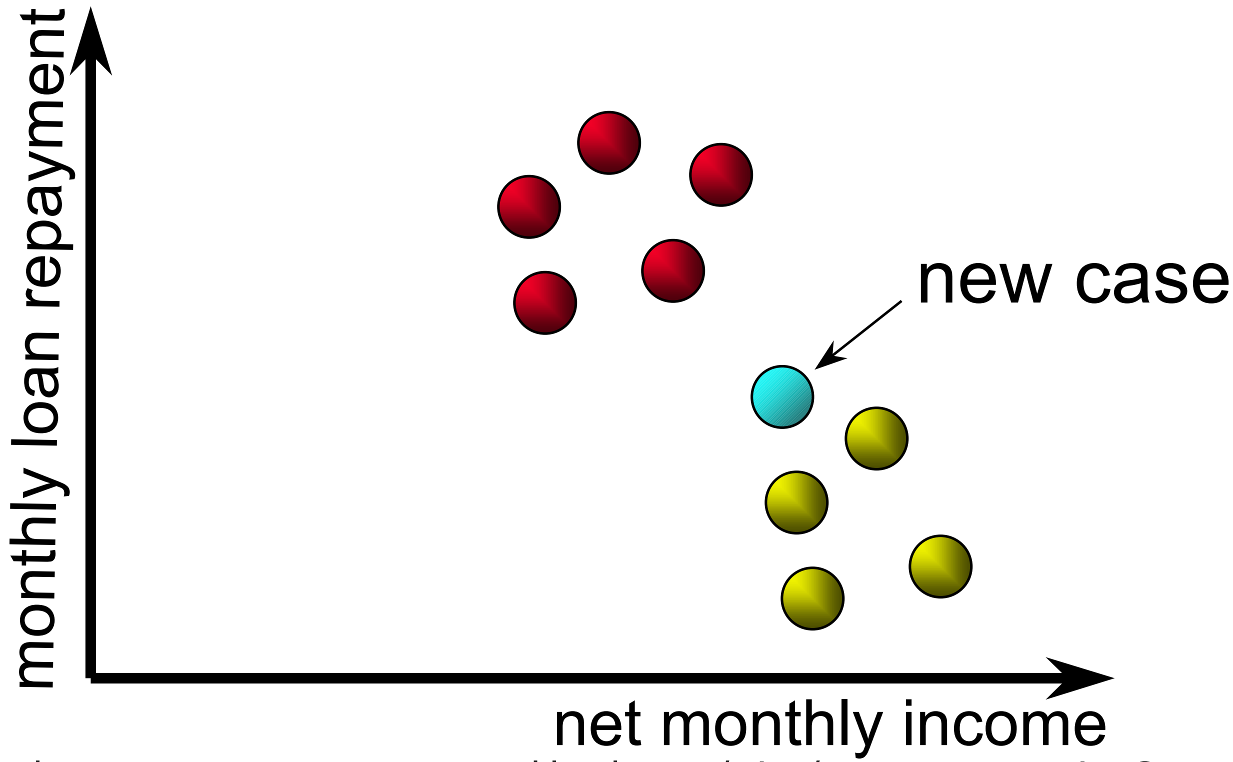
Nearest Neighbour

- a new loan prospect can be plotted on the graph



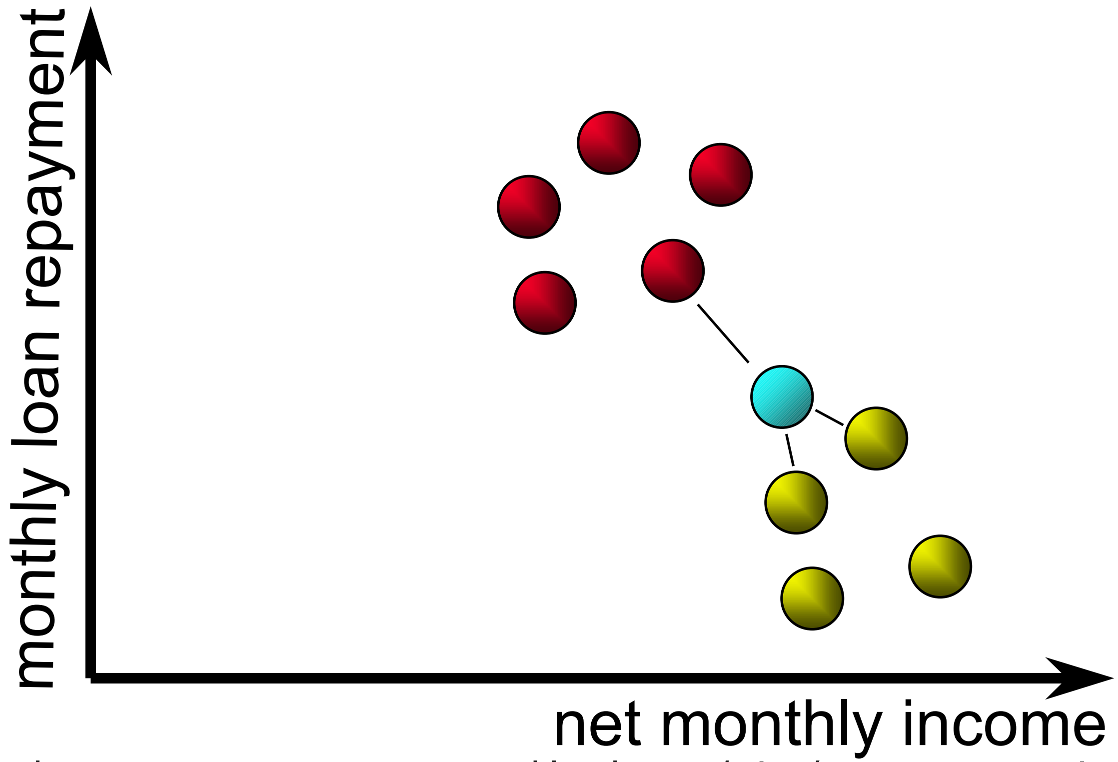
Nearest Neighbour

- and the distance to its nearest neighbours calculated



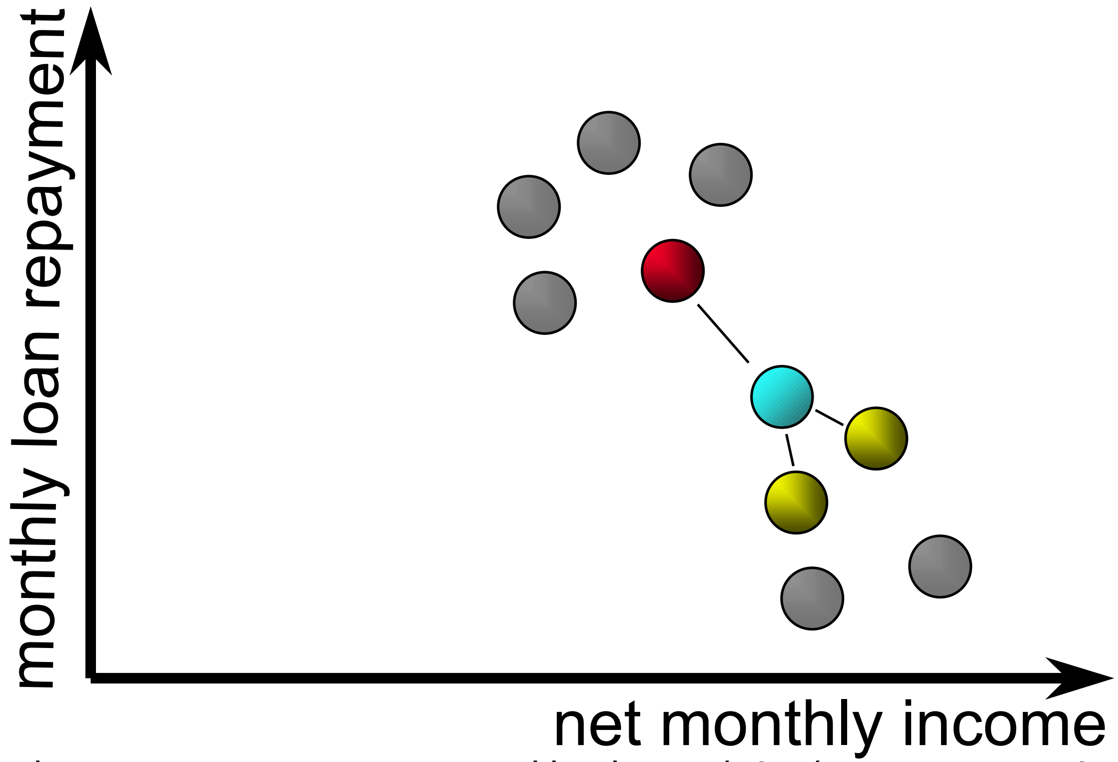
Nearest Neighbour

- and the distance to its nearest neighbours calculated



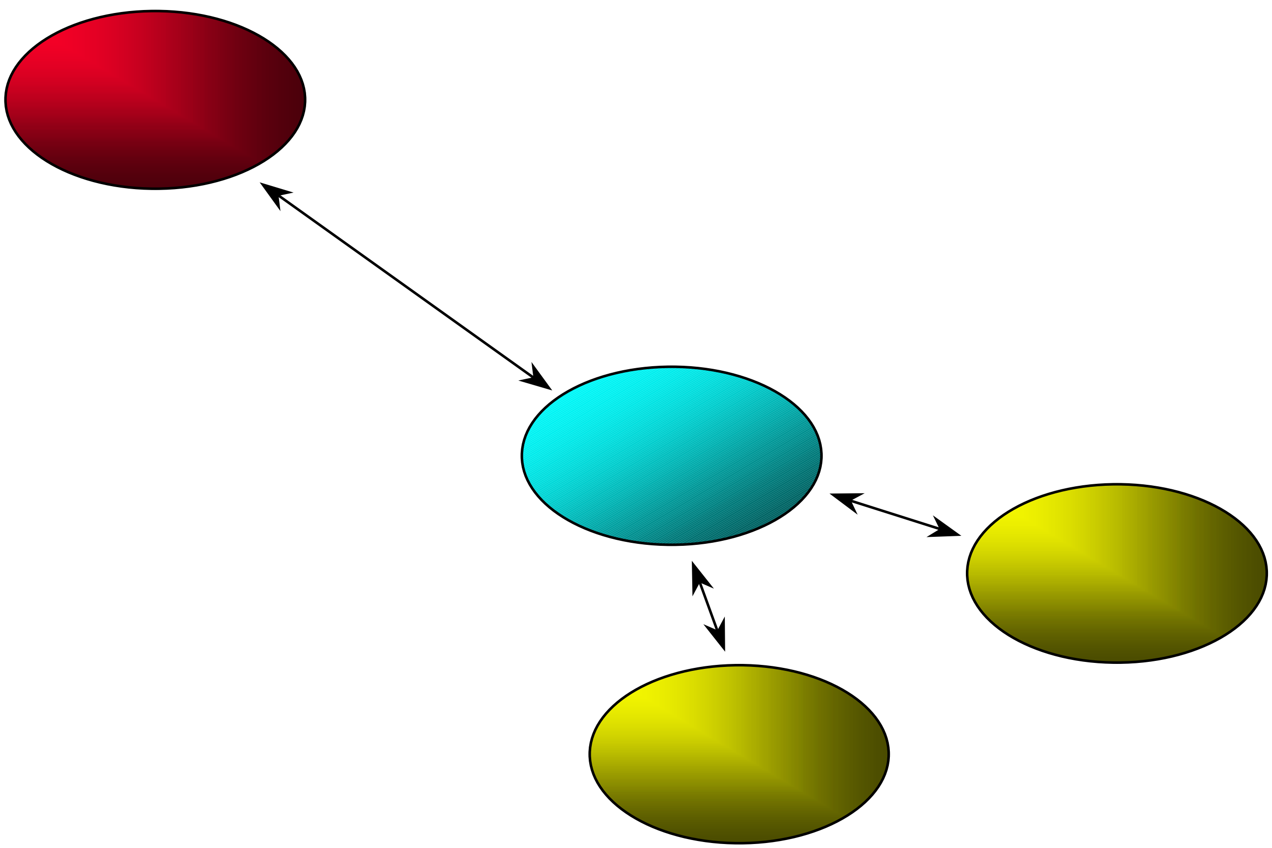
Nearest Neighbour

- and the distance to its nearest neighbours calculated



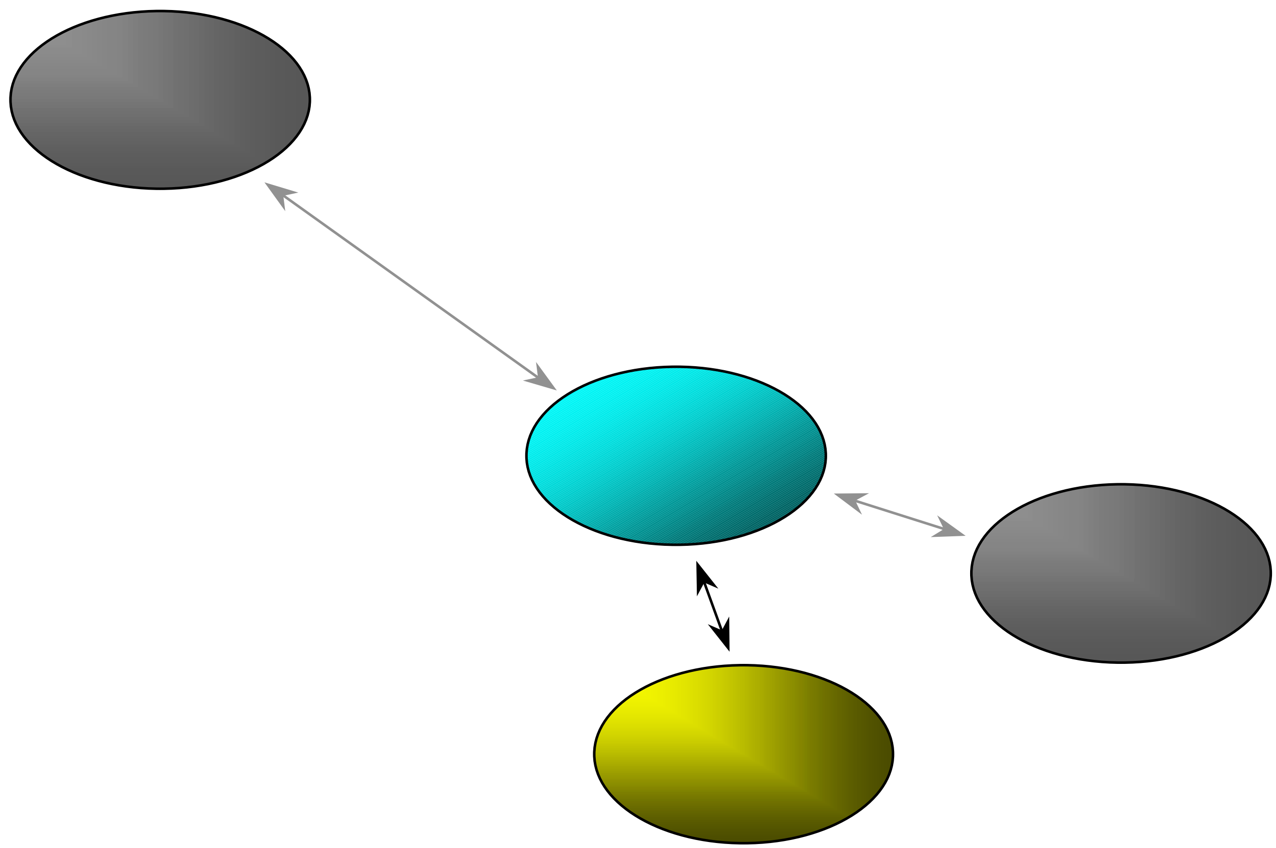
Nearest Neighbour

- the best matching past case is the closest



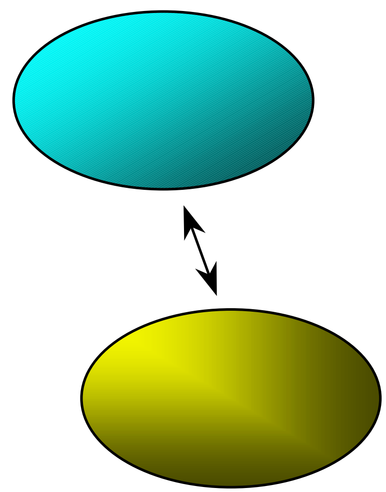
Nearest Neighbour

- the best matching past case is the closest



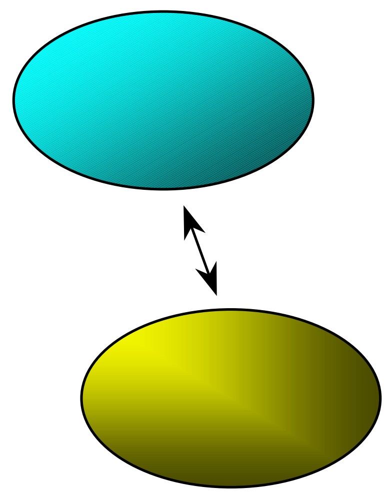
Nearest Neighbour

- this suggests a precedent



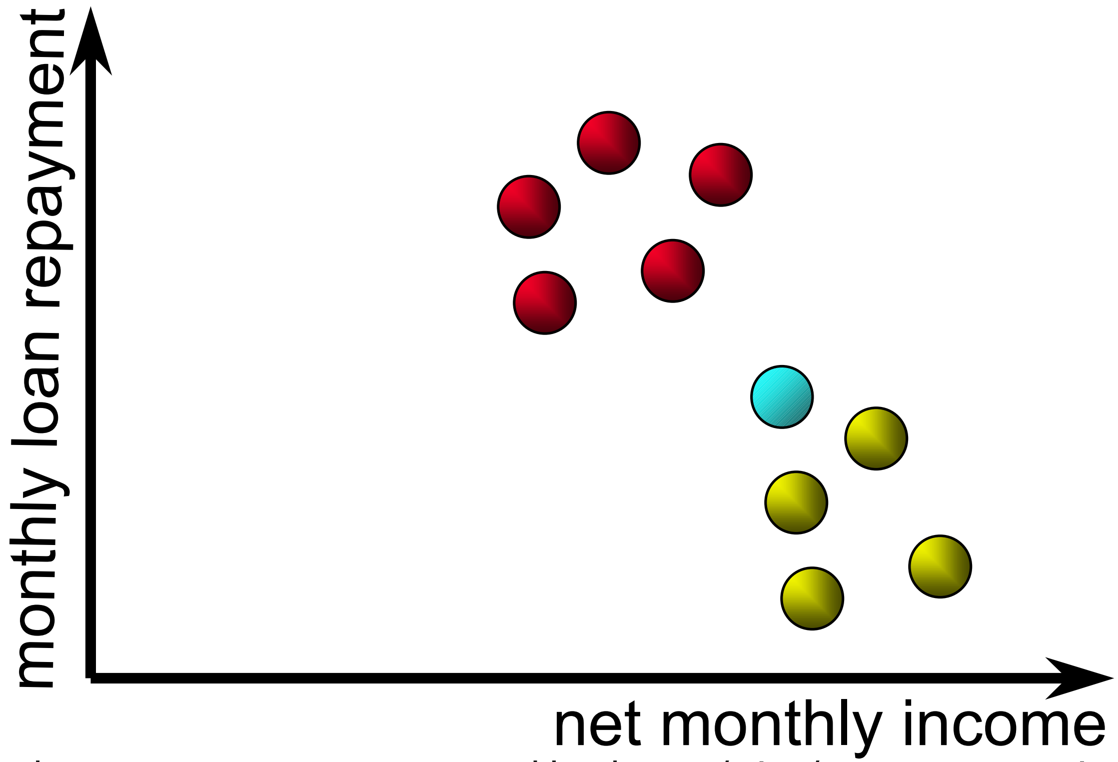
Nearest Neighbour

- this suggests a precedent
- the loan will be successful



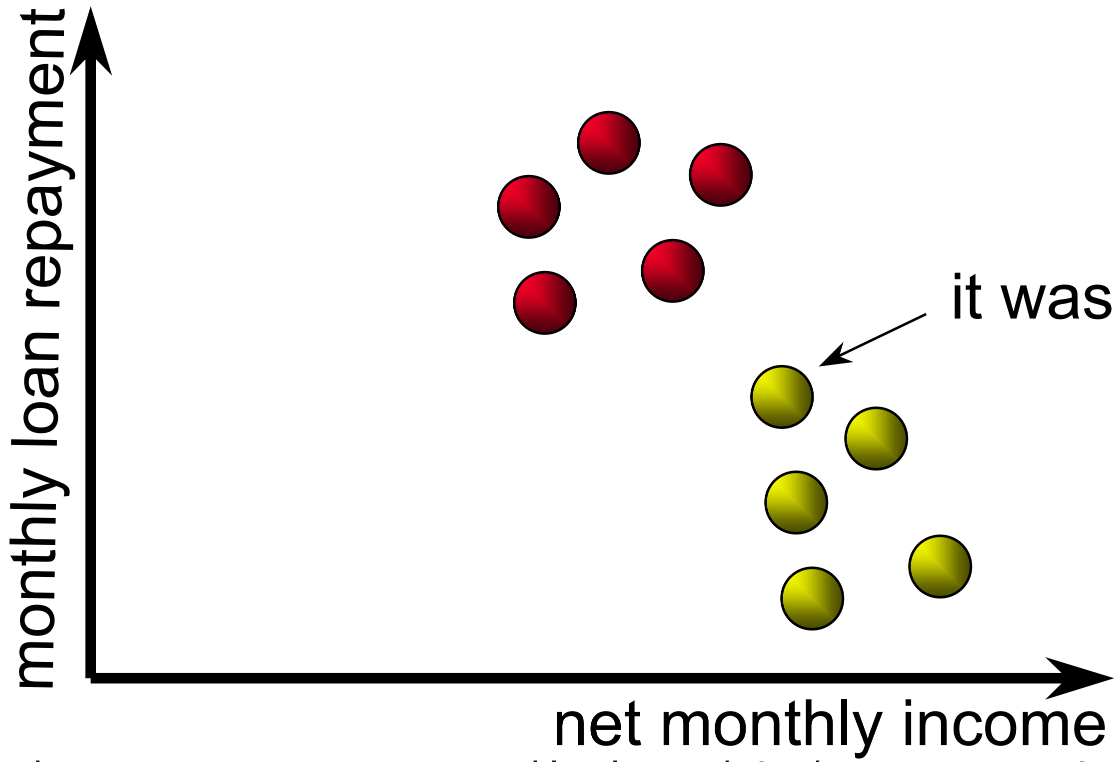
Nearest Neighbour

- over time the prediction can be validated



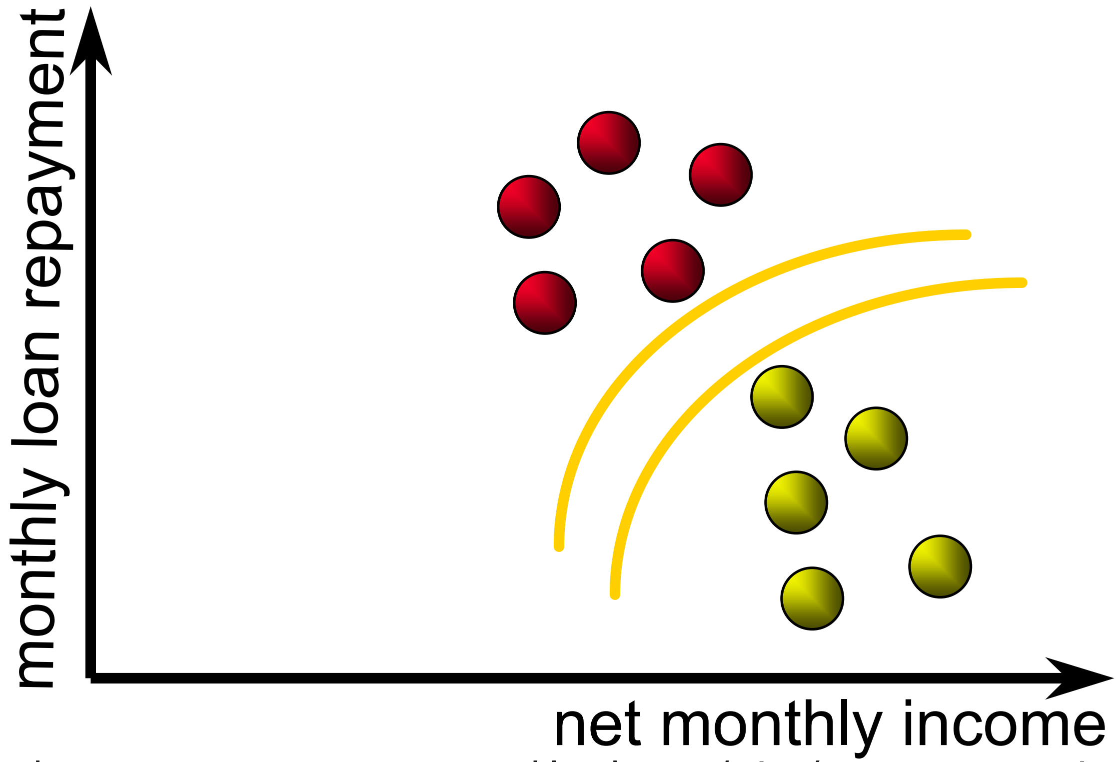
Nearest Neighbour

- over time the prediction can be validated



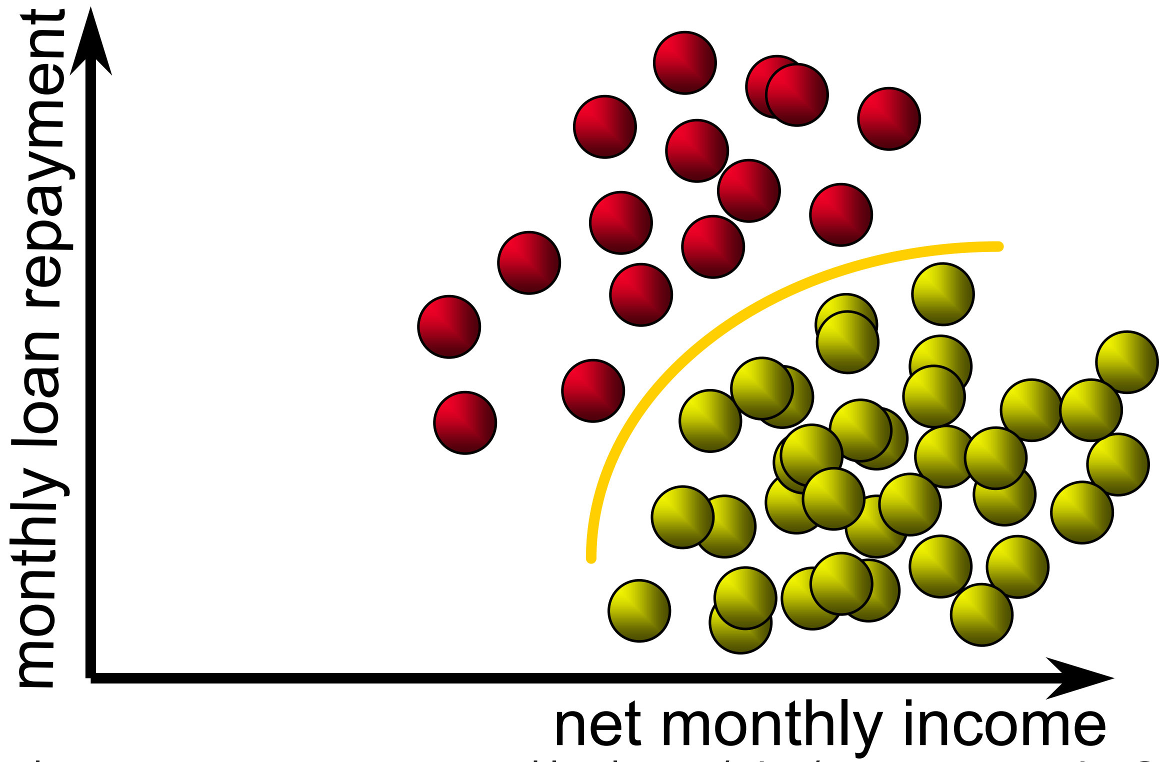
Nearest Neighbour

- the system is learning to differentiate good and bad loans better

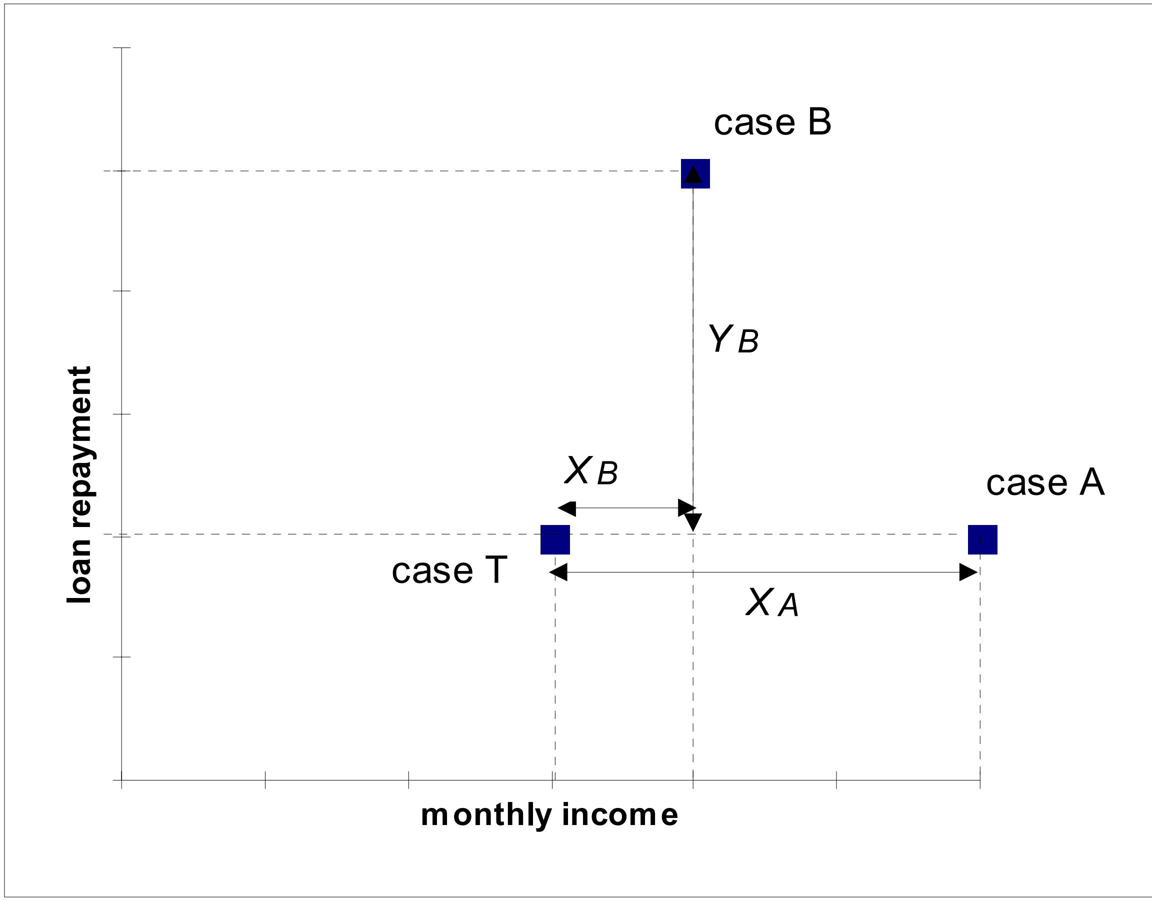


Nearest Neighbour

- as more cases are acquired its performance improves

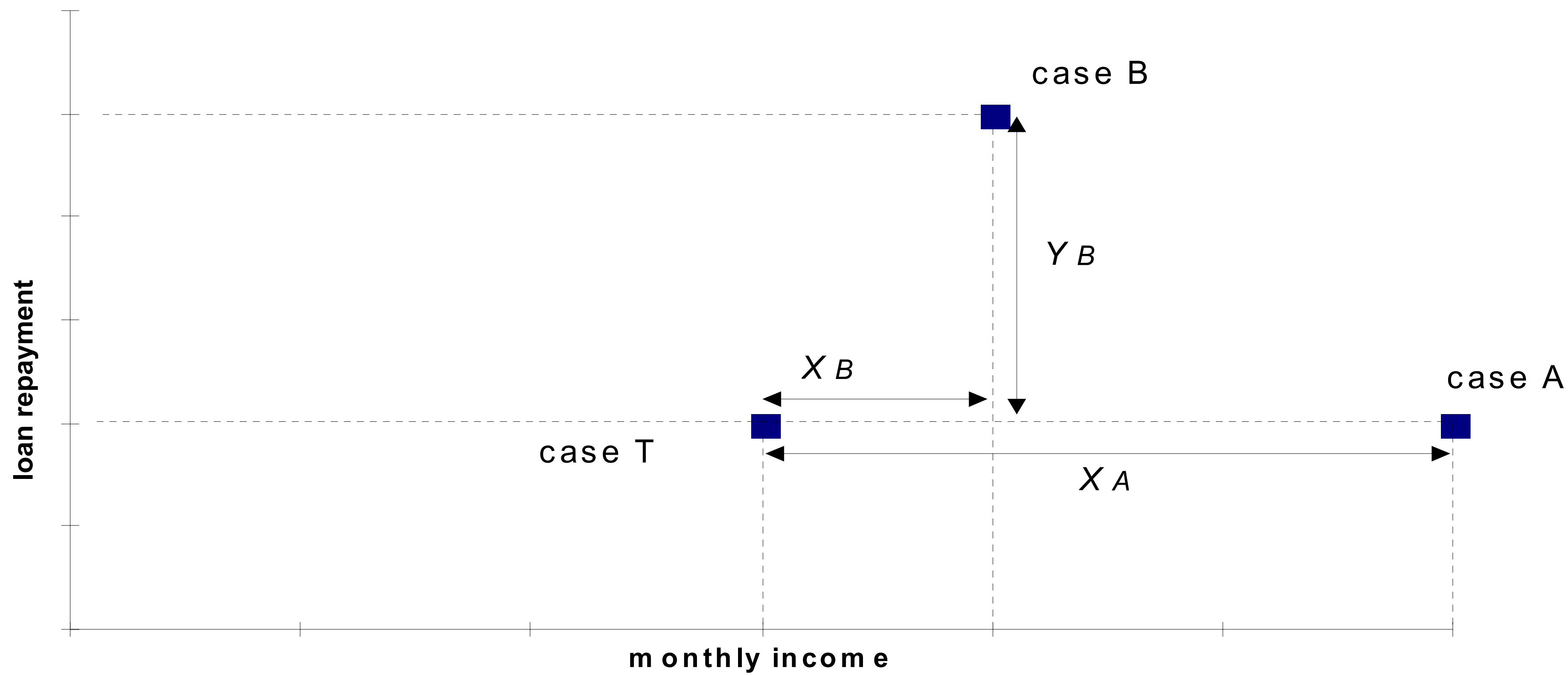


Nearest Neighbour



net monthly income

Nearest Neighbour



net monthly income

Nearest Neighbour

$$Similarity(T,S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i$$

where:
T is the target case
S is the source case
n is the number of attributes in each case
i is an individual attribute from 1 to *n*
f is a similarity function for attribute *i* in cases *T* and *S* and
w is the importance weighting of attribute *i*

net monthly income

Nearest Neighbour

- Require a similarity function for each attribute or feature (not always a trivial problem
- Requires every feature of the query to be compared to every feature of every instance/case
- Not very efficient

Nearest Neighbour

- distance weighted k-Nearest neighbour is a highly effective algorithm for many practical problems robust to noisy data if the training set is large enough
- bias is that the classification of an instance is most similar to other instances that are nearby in Euclidean distance
- But then again that's the point

Nearest Neighbour

- because distance is calculated on all attributes - irrelevant attributes are a problem - curse of dimensionality
- some approaches weight attributes to overcome this - stretching the Euclidean space – determined automatically using cross-validation
- alternatively eliminate the least relevant attributes - they used leave-one out cross-validation – ideal for IBL

Nearest Neighbour

- could locally stretch an axis...but more degrees of freedom...so more chance of overfitting…useful if problem space is not uniform…problem of over fitting
- much less common, but it is used in CBR
- efficient indexing of instances can be done with kd-trees (we'll discuss later)
- possible to pre-compute a position of each instance in the Euclidean space then simply position query in the space

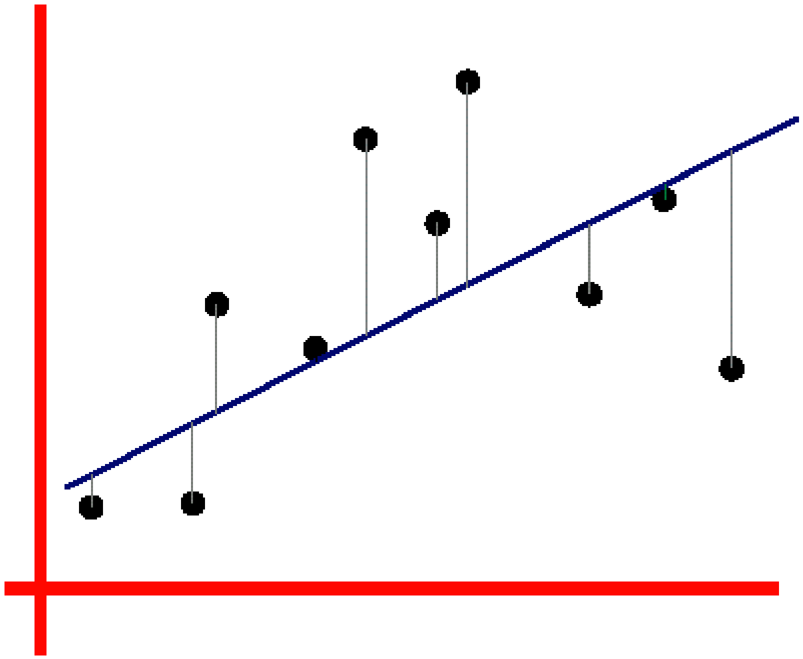
Instance-based learners

- Locally weighted regression
  - algorithm for learning continuous non-linear mappings from real-valued input vectors to real-valued output vectors.
  - particularly appropriate for learning complex highly non-linear functions of up to about 30 inputs from noisy data

Instance-based learners

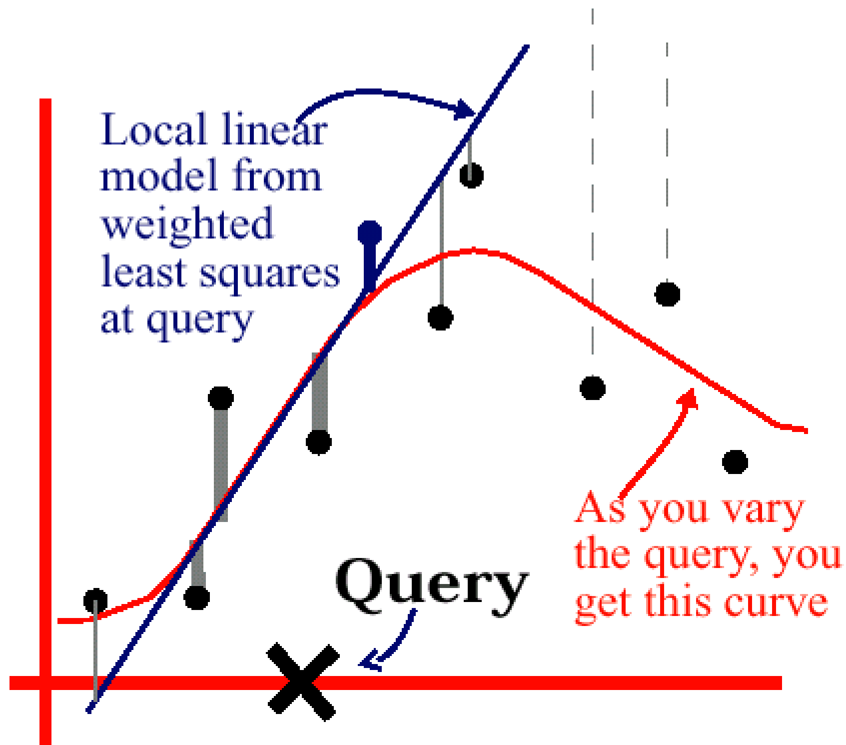
- Locally weighted regression
  - construct an approximation *f* from the training examples in the neighborhood of x_i then calculate F(xi), *f* can then be deleted
  - Assumes that each local function is a linear function
  - Computation grows linearly with # of training instance

Locally weighted regression



This graph shows a global linear regression in progress: the sum of squares of the unweighted residuals is minimized.

Locally weighted regression



This graph shows a locally weighted linear regression. The weighted sum of squared residuals is minimized, where the thickness of the lines indicates the strength of the weight.

Locally weighted regression

- During locally weighted regression a query point x_{query} is supplied
- A linear map is constructed where data points close to the query point have more weight
- A common weighting function is Gaussian
- Moving the query allows the regression algorithm to follow complex fucntions

Locally weighted regression

- broad range of methods for distance weighting the training examples range of methods for locally approximating target functions
- function is usually constant, linear, or quadratic because (1) cost of fitting more complex functions is too high and (2) simple approximations model the target function well over a sufficiently small sub-region

Lazy learning

- Most ML algorithms are *eager* learners
  - Use a training data set to
  - Generalize rules, induce a tree or a function (ANN) that can be applied to categorize future inputs
  - Processing time is done up-front before query time
  - After querying they discard any inputs

Lazy learning

- Lazy learners have three characteristics:
  - They defer processing until query/run-time
  - They discard any generated functions/answers
  - They retain the query with the stored data

Lazy vs. Eager

- Lazy learners have low computational costs at training (~0)
- But have high storage costs
- High computational costs at query
- Lazy learners can respond well to dynamic data where it would be necessary to constantly re-train an eager learner

Summary

- IBLs delay processing until prediction time they form a different local approximation for each query instance
- can model complex functions by a combination of less-complex local approximations
- information present in the training data is never lost (is this a benefit!!!)
- computationally expense to label new instances
- finding appropriate distance metric can be difficult and negative impact of irrelevant attributes