Instance-based Learning

- **Key idea**: In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, instance-based learning constructs the target function only when a new instance must be classified.

- Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance.

Sl. No.	Height	Weight	Target
1	150	50	Medium
2	155	55	Medium
3	160	60	Large
4	161	59	Large
5	158	65	Large
6	157	54	?

Instance-based Learning

- Instance based learning includes nearest neighbor and locally weighted regression methods that assume instances can be represented as points in a Euclidean space.
- It also includes case-based reasoning methods that use more complex, symbolic representations for instances.
- Instance-based methods are sometimes referred to as "lazy" learning methods because they delay processing until a new instance must be classified.
- A key advantage of this kind of delayed, or lazy, learning is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified

Instance-based Learning

- Instance-based learning methods such as nearest neighbor and locally weighted regression are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions.
- Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances are retrieved from memory and used to classify the new query instance

Advantages of Instance-based learning

1. Training is very fast
2. Learn complex target function
3. Don't lose information

Disadvantages of Instance-based learning

- The cost of classifying new instances can be high.
- In many instance-based approaches, especially nearest-neighbor approaches, they typically consider all attributes of the instances when attempting to retrieve similar training examples from memory.
- If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.

k-NEAREST NEIGHBOR LEARNING

- The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm.
- This algorithm assumes all instances correspond to points in the n-dimensional space R^n.
- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.
- The arbitrary instance $\boldsymbol{x}$ be described by the feature vector

$$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$$

where $\boldsymbol{a_r(x)}$ denotes the value of the r^{th} attribute of instance $\boldsymbol{x}$.

- Then the distance between two instances $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ is defined to be $\boldsymbol{d(x_i, x_j)}$, where

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

- In nearest-neighbor learning the target function may be either discrete-valued or real-valued.

k-Nearest Neighbour Learning

- **Assumption:** All instances, x, correspond to points in the n-dimensional space $\mathbf{R}^n$.

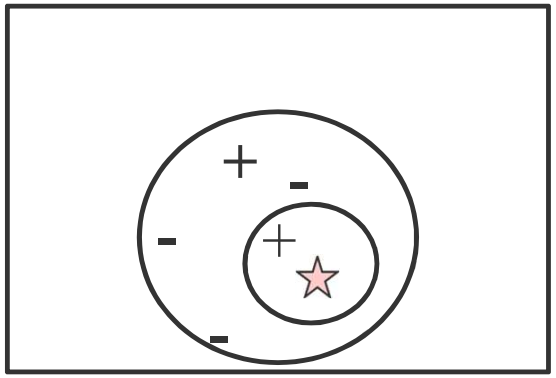
$$x = \langle a_1(x), a_2(x) \ldots a_n(x) \rangle.$$

- **Measure Used:**

$$\textit{Euclidean Distance: } d(x_i,x_j) = \sqrt{\Sigma_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

- **Training Algorithm:**
  - For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*.
- **Classification Algorithm:** Given a query instance x_q to be classified:
  - Let $x_1 \ldots x_k$ be the k instances from training_examples that are nearest to x_q.
  - Return $\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \Sigma_{i=1}^{n} \delta(v, f(x_i))$
  - where $\delta(a,b)=1$ if $a=b$ and $\delta(a,b)=0$ otherwise.

Example



: query, x_q

1-NN: +

5-NN: -

Sl. No.	Height	Weight	Target
1	150	50	Medium
2	155	55	Medium
3	160	60	Large
4	161	59	Large
5	158	65	Large
6	157	54	?

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

- The K- Nearest Neighbor algorithm for approximation a **real-valued target function** is given below $f : \Re^n \rightarrow \Re$

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Distance-Weighted Nearest Neighbour

- k-NN can be refined by weighing the contribution of the k neighbours according to their distance to the query point x_q, giving greater weight to closer neighbours.

- To do so, replace the last line of the algorithm with

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{n} w_i \delta(v, f(x_i))$$

- where $w_i = 1/d(x_q, x_i)^2$

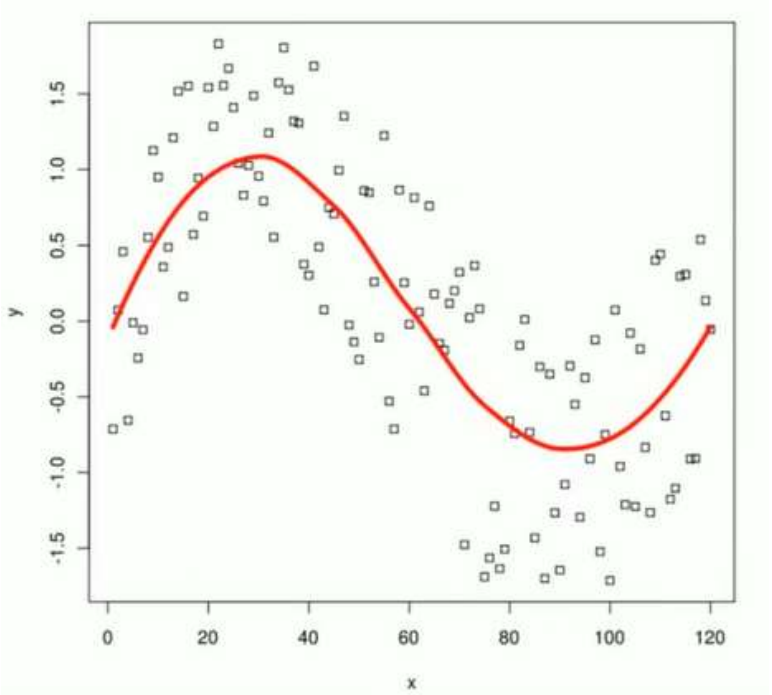
Sl. No.	Height	Weight	Target
1	150	50	Medium
2	155	55	Medium
3	160	60	Large
4	161	59	Large
5	158	65	Large
6	157	54	?

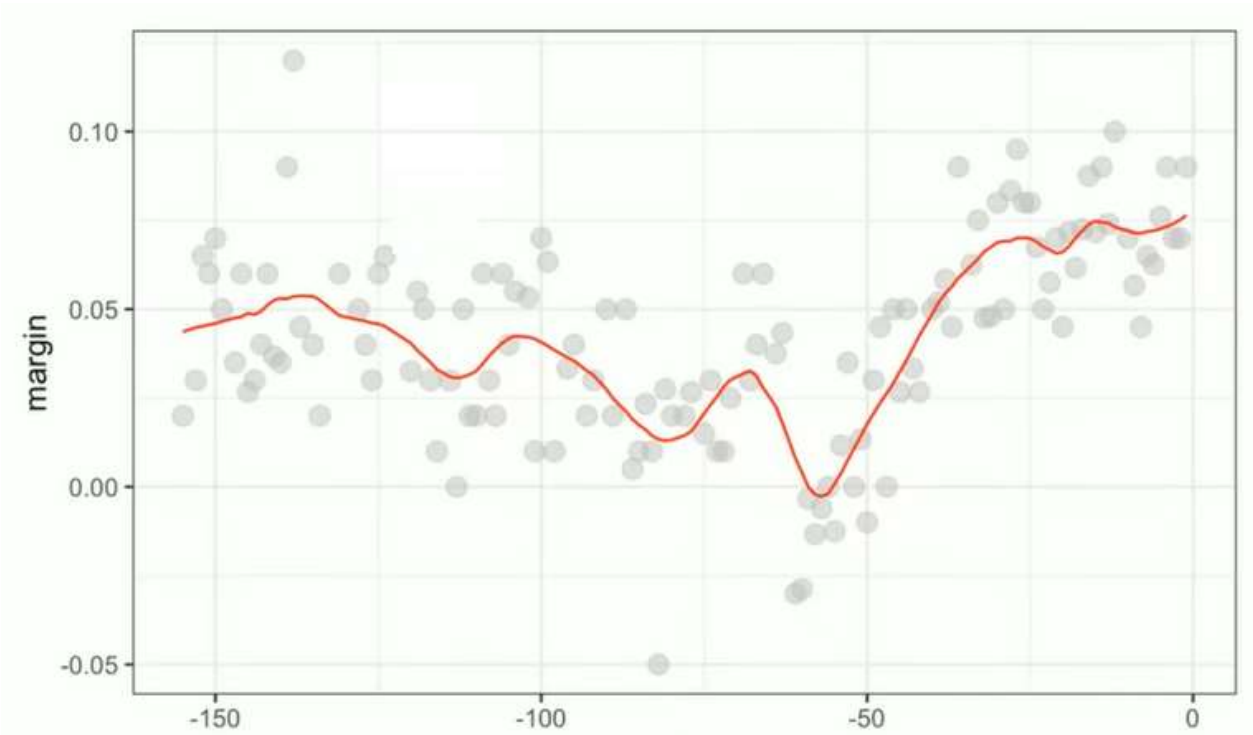
Remarks on k-NN

- k-NN can be used for ***regression*** instead of classification.
- k-NN is ***robust to noise*** and, it is generally quite a good classifier.
- k-NN's ***disadvantage*** is that it uses all attributes to classify instances
  - **Solution 1:** weigh the attributes differently (use cross-validation to determine the weights)
  - **Solution 2:** eliminate the least relevant attributes (again, use cross-validation to determine which attributes to eliminate)

LOCALLY WEIGHTED REGRESSION

- Locally weighted regression is instance based learning algorithm.
- The phrase "**locally weighted regression**" is called
  - **local** because the function is approximated based only on data near the query point,
  - **weighted** because the contribution of each training example is weighted by its distance from the query point, and
  - **regression** because this is the term used widely in the statistical learning community for the problem of approximating real-valued functions.
- Given a new query instance x_q, the general approach in locally weighted regression is to construct an approximation $\hat{f}$ that fits the training examples in the neighborhood surrounding x_q.


y
x
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0
20
40
60
80
100
120


margin
0.10
0.05
0.00
-0.05
-150
-100
-50
0

- Consider locally weighted regression in which the target function ***f*** is approximated near x_q using a linear function of the form

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \cdots + w_n a_n(x)$$

- Where, $a_i(x)$ denotes the value of the i^{th} attribute of the instance x
- Gradient descent of ANN

$$w_i \leftarrow w_i + \Delta w_i$$

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \qquad \Delta w_i = \eta \sum_{d \in D} (t_d - o_d)\, x_{id}$$

- Derived methods are used to choose weights that minimize the squared error summed over the set D of training examples using gradient descent

$$E \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

- Which led us to the gradient descent training rule

$$\Delta w_j = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) a_j(x)$$

Need to modify this procedure to derive a local approximation rather than a global one. The simple way is to redefine the error criterion E to emphasize fitting the local training examples. Three possible criteria are given below.

1. Minimize the squared error over just the k nearest neighbors:

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \qquad \text{equ(1)}$$

2. Minimize the squared error over the entire set D of training examples, while weighting the error of each training example by some decreasing function K of its distance from x_q :

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x)) \qquad \text{equ(2)}$$

3. Combine 1 and 2:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x)) \qquad \text{equ(3)}$$

- If we choose criterion three and re-derive the gradient descent rule, we obtain the following training rule

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest nbrs of } x_q} K(d(x_q, x)) \; (f(x) - \hat{f}(x)) \; a_j(x)$$

CASE-BASED REASONING

- **Instance-based methods** such as k-NEAREST NEIGHBOR and locally weighted regression share three key properties.
- **First**, they are lazy learning methods in that they defer the decision of how to generalize beyond the training data until a new query instance is observed.
- **Second**, they classify new query instances by analyzing similar instances while ignoring instances that are very different from the query.
- **Third**, they represent instances **as** real-valued points in an n-dimensional Euclidean space.

- In CBR the instances are not represented as real-valued points, but instead, they use a ***rich symbolic*** representation and the methods used to retrieve similar instances are correspondingly more elaborate.
- CBR has been applied to problems such as
  - conceptual design of mechanical devices based on a stored library of previous designs,
  - reasoning about new legal cases based on previous rulings

Case-based reasoning consists of a cycle of the following four steps:

1. **Retrieve -** Given a new case, retrieve similar cases from the case base.
2. **Reuse -** Adapt the retrieved cases to fit to the new case.
3. **Revise -** Evaluate the solution and revise it based on how well it works.
4. **Retain -** Decide whether to retain this new case in the case base.

Example:

- Help desk that users call with problems to be solved.
- When users give a description of a problem, the closest cases in the case base are retrieved.
- The diagnostic assistant could recommend some of these to the user, adapting each case to the user's particular situation.
- If one of the adapted cases works, that case is added to the case base, to be used when another user asks a similar question.
- If none of the cases found works, some other method is attempted to solve the problem, perhaps by adapting other cases or having a human help diagnose the problem.

Case based Reasoning – Solved Example

- Suppose that the database of a CBR system contains the following four cases:

Case	Monthly Income (£K)	Account Balance (£K)	Home Owner	Credit Score
1	3	2	0	2
2	2	1	1	2
3	3	2	2	4
4	0	-1	0	0

- The system is using the nearest neighbor retrieval algorithm with the following similarity function: $\boldsymbol{d(T,S) = \sum_{i=1}^{m} |T - S_i| * w_i}$
- where T is the target case, S is the source case, i is the number of a feature, and w_i are the weights. Cases with smaller values of d(T, S) are considered to be more similar.

- Consider the following new (target) case:

Case	Monthly Income (£K)	Account Balance (£K)	Home Owner	Credit Score
5	3	1	2	?

Answer the following questions:

a) Which case will the CBR system retrieve as the 'best match', if all the weights $w_i = 1$?

b) The solution that the CBR system should propose is the credit score rating. Suggest how should the solution of the retrieved case be adapted for the target case?

a) Which case will the CBR system retrieve as the 'best match', if all the weights $w_i = 1$?

Case	MI (£K)	AB (£K)	HO	Credit Score
1	3	2	0	2
2	2	1	1	2
3	3	2	2	4
4	0	-1	0	0
5	3	1	2	?

- $\boldsymbol{d(T,S) = \sum_{i=1}^{m} |T - S_i| * w_i}$
- $d(T,S1) = |3-3| + |1-2| + |2-0| = 0 + 1 + 2 = 3$
- $d(T,S2) = |3-2| + |1-1| + |2-1| = 1 + 0 + 1 = 2$
- $d(T,S3) = |3-3| + |1-2| + |2-2| = 0 + 1 + 0 = 1$
- $d(T,S4) = |3-0| + |1+1| + |2-0| = 3 + 2 + 2 = 7$
- **The most similar is Case 3.**

b) The solution that the CBR system should propose is the credit score rating. Suggest how should the solution of the retrieved case be adapted for the target case?

- **The Credit Score for Case 3 is 4.**

Case	MI (£K)	AB (£K)	H O	Credit Score
1	3	2	0	2
2	2	1	1	2
3	3	2	2	4
4	0	-1	0	0
5	3	1	2	?

- The only difference between the target case and Case 3 in the Account Balance (1 < 2).
- Based on the other cases, one can derive that the decrease in Account Balance should decrease the credit.
- Thus, the solution of Case 3 can be adapted by decreasing the value.
- The revised solution for the new case is:
- **Credit Score = 3**