Μάθηση Βασισμένη σε Στιγμιότυπα
(Instance-based Learning)

- Storing training examples
- No explicit model (= target function) construction
- Processing delayed until a new instance has to be classified
- Lazy learning / Memory-based learning
- Usually, instances are points in an Euclidean space
- Local approximation of real-valued or discrete-valued target functions
- Need for indexing of instances (kd-trees)
- k-nearest neighbor learning / locally weighted regression

A Cookie Making Example

Sugar	Flour	Temperature	Minutes	YouLike
3	5	250	120	yes
6	4	250	180	yes
5	4	200	120	no
4	2	300	120	yes
7	6	300	90	no
8	3	200	90	no
5	5	350	180	yes
7	9	250	180	yes

Do you like cookies made of 5 kgs sugar and 6 kgs flour baked at 250^oC for 120 minutes?

k-nearest Neighbor Learning

- Instances described by feature vectors $\langle a_1(x), a_2(x), \ldots, a_n(x) \rangle$, where $a_r(x)$ denotes the rth attribute of instance x

- Distance between two instances x_i and x_j

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

- Approximation of discrete-valued target functions

$$f : \Re^n \longrightarrow V, \text{ where } V = \{v_1, v_2, \ldots, v_n\}$$

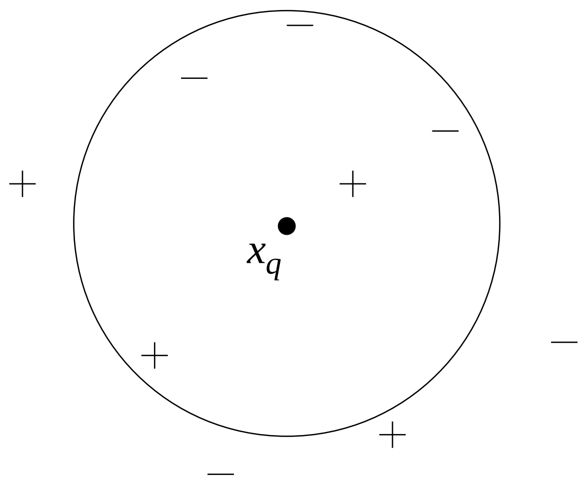
- Classification algorithm:
  - Given a query instance x_q to be classified
  - Let $x_1, x_2, \ldots, x_k$ the k instances from training examples that are nearest to x_q
  - Return

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

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise

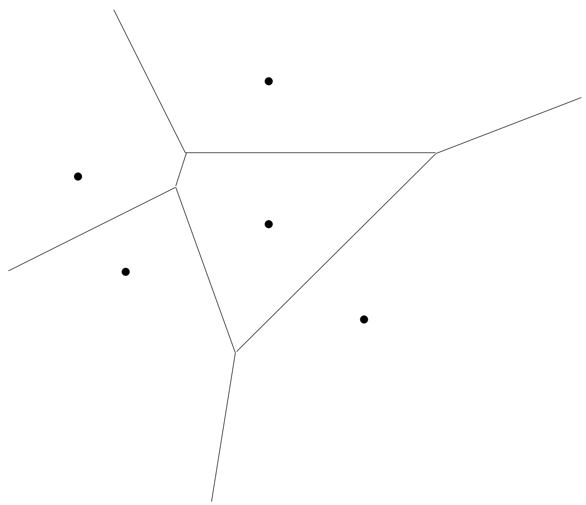
- Approximation of real-valued target functions ($f : \Re^n \longrightarrow \Re$)
  - Replace equation (1) in the classification algorithm by:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \tag{2}$$



5-nearest neighbor vs. 1-nearest neighbor

convex polyhedra surrounding training examples



- Distance-weighted nearest neighbor algorithm
  - For discrete-valued target functions replace equation (1) by:

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

  - For real-valued target functions replace equation (2) by:

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

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

- Curse of dimensionality
  - Irrelevant attributes
  - Stretching the axes
  - (Leave-one-out) cross-validation

Locally Weighted Regression

- Explicit local approximation of the target function, usually by a linear function

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

- Compute w_j in a way that the error for the query instance x_q

$$E(x_q) = \frac{1}{2} \sum_{x \in\ k\ nearest\ nbrs\ of\ x_q} (f(x) - \hat{f}(x))^2\, K(d(x_q, x))$$

  is minimized, where $K(d)$ is the kernel function, usually equal to $1/d^2$

- Computation of w_j may be done by gradient descent, giving:

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

Μάθηση κατά Bayes
(Bayesian Learning)

- Naive Bayes classifier

- Each training example is described by a conjunction of attribute values $\langle a_1, a_2, \ldots, a_n \rangle$ and the known target value (classification) of the example, from some finite set V

- Need to predict the target value of a new instance, given its attribute values

- Prediction is based on the Bayes theorem

$$P(v_j | a_1, a_2, \ldots, a_n) = \frac{P(a_1, a_2, \ldots, a_n | v_j)\, P(v_j)}{P(a_1, a_2, \ldots, a_n)}$$

A Playing Tennis Example

Outlook	Temperature	Humidity	Wind	PlayTennis
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

- The *maximum a posteriori* hypothesis v_{MAP} may be computed via the frequencies $P(v_j)$ of target values and the frequencies of combinations of attribute values for specific target values $P(a_1, a_2, \ldots, a_n | v_j)$, in the training data

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{argmax}\ P(v_j | a_1, a_2, \ldots, a_n) \\ &= \underset{v_j \in V}{argmax}\ \frac{P(a_1, a_2, \ldots, a_n | v_j)\, P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\ &= \underset{v_j \in V}{argmax}\ P(a_1, a_2, \ldots, a_n | v_j)\, P(v_j) \end{aligned}$$

- The naive Bayes classifier introduces the independence assumption

$$P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$$

  for the reduction of the required probabilities, so

$$v_{MAP}\ (\equiv v_{NB}) = \underset{v_j \in V}{argmax}\ P(v_j) \prod_i P(a_i | v_j)$$

Back to the Example

Will you play tennis when

Outlook=sunny, Temperature=cool, Humidity=high and Wind=strong?

$$\begin{aligned} v_{NB} &= \underset{v_j \in \{yes,no\}}{argmax} \; P(v_j) \quad \prod_i P(a_i|v_j) \\ &= \underset{v_j \in \{yes,no\}}{argmax} \; P(v_j) \quad P(Outlook = sunny|v_j) \; P(Temperature = cool|v_j) \\ & \qquad\qquad\qquad\qquad\quad P(Humidity = high|v_j) \; P(Wind = strong|v_j) \end{aligned}$$

$$\begin{aligned} P(yes) \; P(sunny|yes) \; P(cool|yes) \; P(high|yes) \; P(strong|yes) &= \frac{9}{14} \; \frac{2}{9} \; \frac{3}{9} \; \frac{3}{9} \; \frac{3}{9} = 0.0053 \\ P(no) \; P(sunny|no) \; P(cool|no) \; P(high|no) \; P(strong|no) &= \frac{5}{14} \; \frac{3}{5} \; \frac{1}{5} \; \frac{4}{5} \; \frac{3}{5} = 0.0206 \end{aligned}$$

Thus, $v_{NB} = no$

- Introduction of m-estimate of probability to cope with probabilities equal to 0

$$\frac{n_c + mp}{n + m}$$

  where, n is the number of examples with a specific target value, n_c the number of these examples with a specific attribute value, m is the *equivalent sample size* and p is the prior estimate of the probability we wish to determine (usually equal to $1/k$, where k is the number of possible values of the attribute under consideration)

- Application to text classification (target values v_j)
  - Attributes are word positions a_i and values are words w_k from a vocabulary
  - Assumption: Values are independent from the positions, $P(a_i = w_k|v_j) = P(w_k|v_j)$
  - Exploiting m-estimate of probability, $P(w_k|v_j) = \frac{n_k+1}{n+|Vocabulary|}$
  - Using 2/3 of 20000 Usenet articles from 20 newsgroups as training data, classification of the rest 1/3 gave 89% accuracy, while random classification should give 5%