Cybernetics and Artificial Intelligence

---

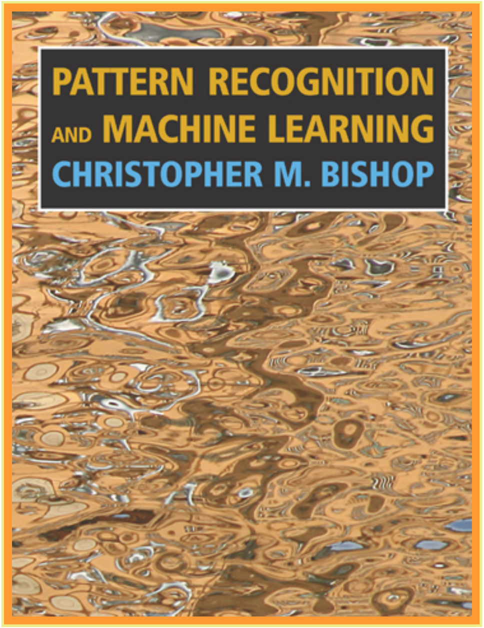
1. Probabilistic Decision Making and Classification

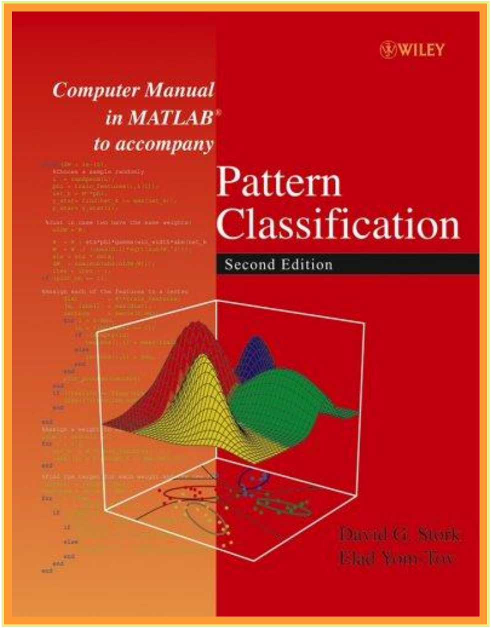
Gerstner laboratory
Dept. of Cybernetics
Czech Technical University in Prague

Daniel Novák
Thanks to: Filip Železný

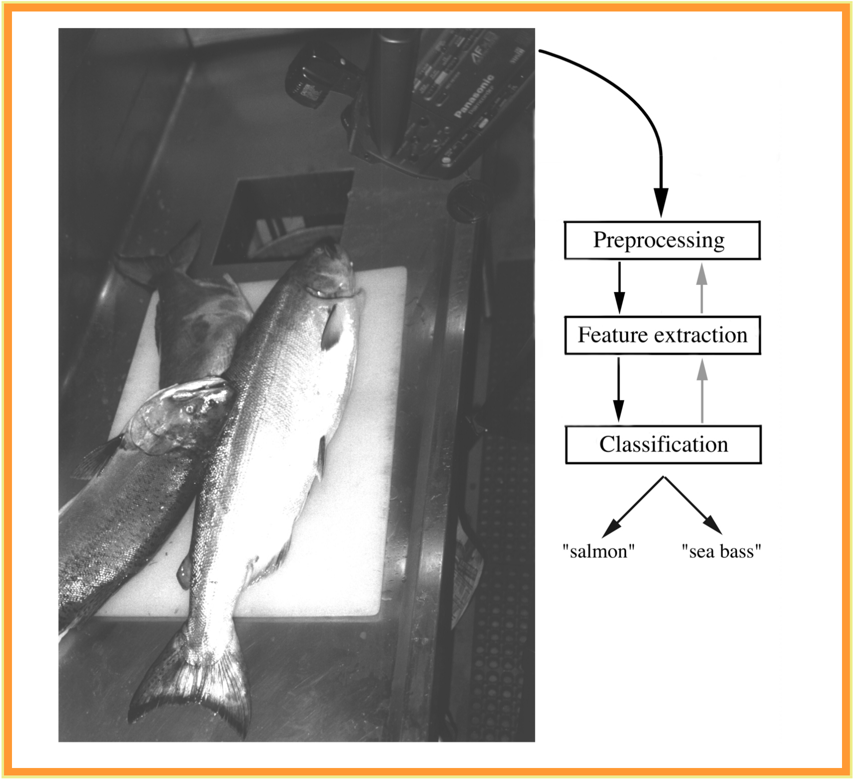
Literature, demos

- Duda, Hart, Stork: Pattern Classification `http://www.crc.ricoh.com/~stork/DHS.html`
- Ch. Bishop, Pattern Recognition and Machine Learning `http://research.microsoft.com/en-us/um/people/cmbishop/prml/`
- Kotek, Vysoký, Zdráhal: Kybernetika 1990
- Classification toolbox
  `http://stuff.mit.edu/afs/sipb.mit.edu/user/arolfe/matlab/`
- Statistical Pattern Recognition Toolbox
  `http://cmp.felk.cvut.cz/cmp/software/stprtool/`



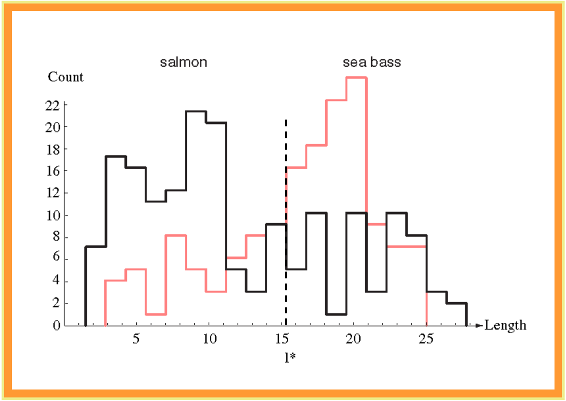


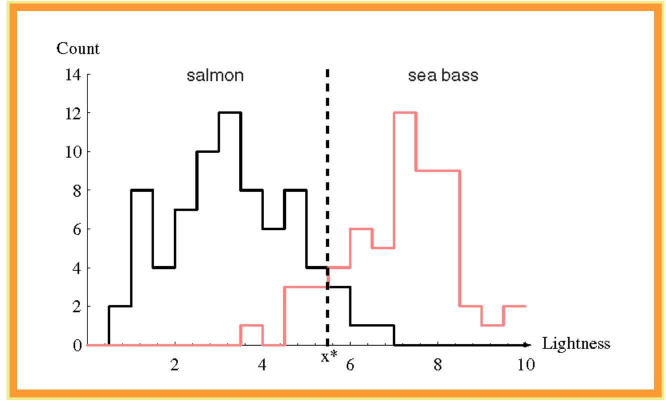
Motivation example I [Duda, Hart, Stork: Pattern Classification]



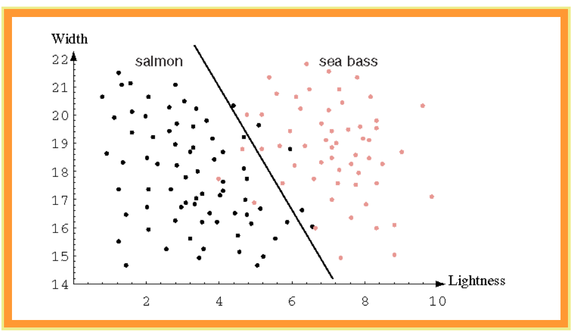
- Factory for fish processing
- — 2 classes - Detection of salmon and sea bass based on a camera
  — Features - we measure width,length, etc.
- the TASK is: FISH CLASSIFICATION

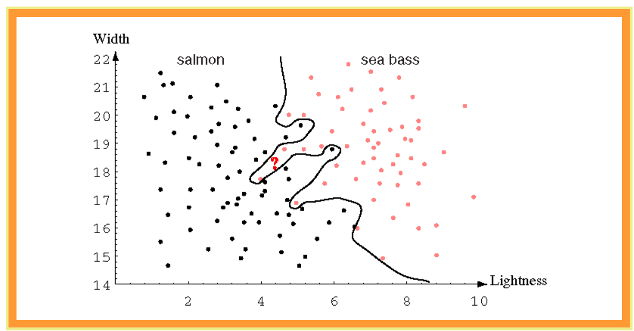
Motivation example II

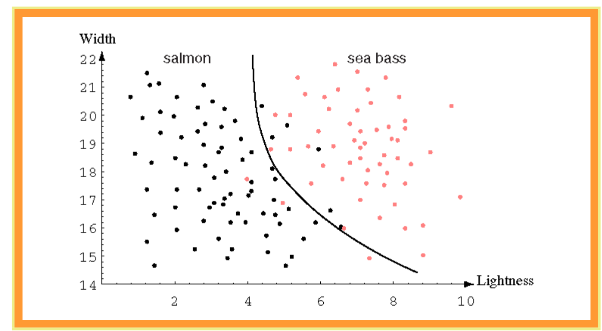




- We estimate the feature distribution using histograms
- Wrong classification due to histograms overlapping
- Improvement-feature combination







- Linnear, quadratic, k-nearest classifier
- Over-fitting, Generalization, error minimalization

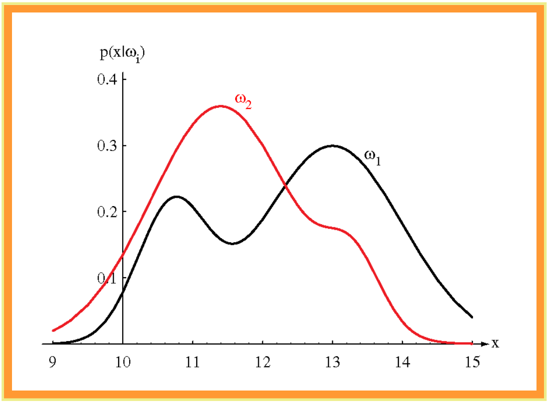
We know probability distribution

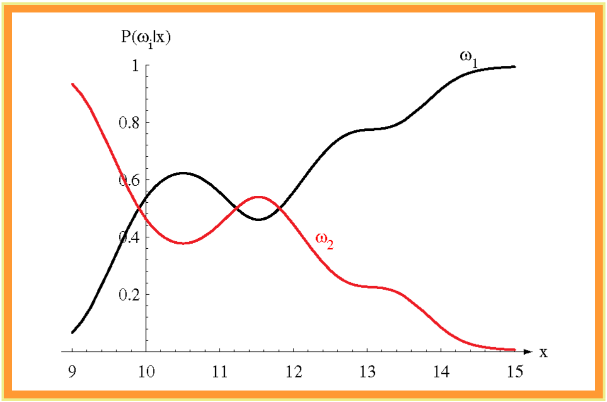
- Yes - Bayes classification
- Apriori distribution $p(s_j)$ and conditional probability $p(x|s_j)$
- Thus $p(s_j, x) = p(x|s_j)p(s_j) = p(s_j|x)p(x)$
- Bayes theorem

$$p(s_j|x) = \frac{p(x|s_j)p(s_j)}{p(x)}$$

$$posterior \propto likelihood \times prior$$

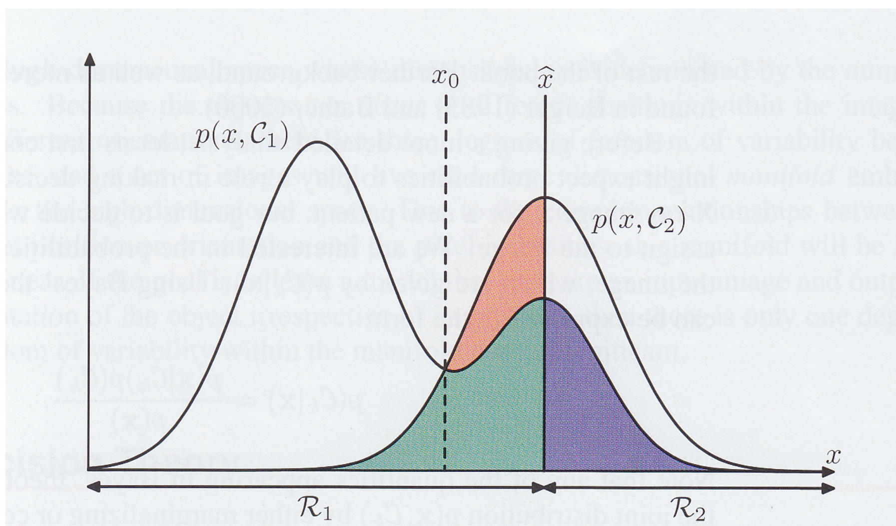
- Classification $\arg\max_j p(s_j|x)$
- $p(x|s_1) = \frac{1}{3}, p(x|s_1) = \frac{2}{3}$, (in images below $s_i = \omega_i$)





Error minimization - maximum aposterior probability (MAP)[Bishop]

- Illustrative example
- $p(s_1, x) = p(x|s_1)p(s1)$,$p(s_2, x) = p(x|s_2)p(s_2)$, see image below ($s_i = C_i$)
- Classification error: $p(error) = p(x \in R_1, C_2) + p(x \in R_2, C_1)$
- Error $p(x \in R_1, C_2)$ - Redá and green area - objects are classified as C_2 instead of C_1
- Error $p(x \in R_2, C_1)$ - blue area - objects are classified as C_1 instead of C_2
- Classification error minimization - both probabilities are overlapping in x_0 (red area will disappear)

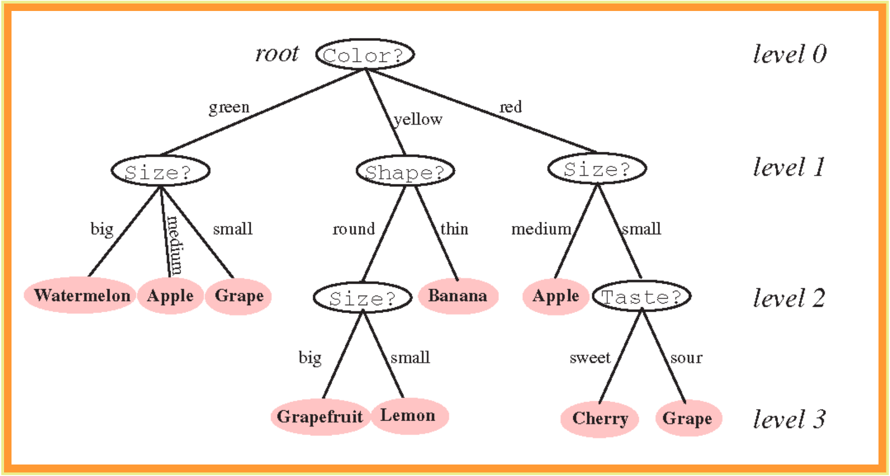


Historical note - Thomas Bayes

- Thomas Bayes - published in 1736 study *An Introduction to the Doctrine of Fluxions, and a Defence of the Mathematicians Against the Objections of the Author of the Analyst*
- Example: solution of white and black bowls using Bayes equation

Probability distribution is unknown

- Training and test data
- There are thousand classifiers - e.g. decision trees



Decision making under uncertainty

- An important feature of intelligent systems
  - make the best possible decision
  - in **uncertain** conditions.
- **Example**: Take a tram OR subway from A to B?
  - Tram: timetables imply a quicker route, but adherence uncertain.
  - Subway: longer route, but adherence almost certain.
- **Example**: where to route a letter with this ZIP?

  - 15700? 15706? 15200? 15206?
- What is the **optimal decision**?
- Both examples fall into the same framework.

Example [Kotek, Vysoký, Zdráhal: Kybernetika 1990]

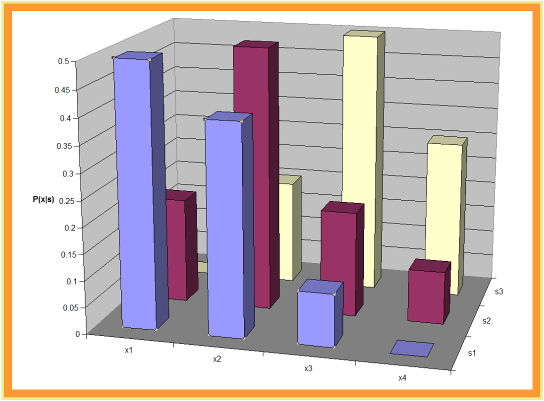
- *Wife coming back from work. Husband pondering what to cook for dinner.*
- 3 dishes **decisions** in his repertoir:
  - *nothing* . . . **don't bother cooking** ⇒ no work but makes wife upset
  - *pizza* . . . **microwave a frozen pizza** ⇒ not much work but won't impress
  - *g.T.c.* . . . **general Tso's chicken** ⇒ will make her day, but very laborious.
- Husband quantifies the degree of hassle incurred by the individual options. This depends on how wife is feeling on her way home. Her state of mind is an **uncertain state**. Let us distinguish her mood:
  - *good* . . . wife is feeling **good**.
  - *average* . . . wife **average** mooded.
  - *bad* . . . wife **bad** mooded.
- For each of the 9 possible situation (3 possible decisions $\times$ 3 possible states) the hassle is quantified by a **loss function** $l(d, s)$:

$l(s, d)$	$d =$ *nothing*	$d =$ *pizza*	$d =$ *g.T.c.*
$s =$ *good*	0	2	4
$s =$ *average*	5	3	5
$s =$ *bad*	10	9	6

Example (cont'd)

- Husband tries to estimate wife's state of mind through an experiment. He tells her he accidentally overtaped their wedding video and observes her reaction
- Anticipates 4 possible reactions:
  - *mild* . . . all right, we keep our memories.
  - *irritated* . . . how many times do I have to tell you....
  - *upset* . . . Why did I marry this guy?
  - *alarming* . . . silence
- The reaction is a measurable **attribute** (of the state of mind).
- From experience, husband knows how individual reactions are probable in each state of mind; this is captured by conditional distribution $P(x|s)$.

$P(x\|s)$	$x =$ *mild*	$x =$ *irritated*	$x =$ *upset*	$x =$ *alarming*
$s =$ *good*	0.5	0.4	0.1	0
$s =$ *average*	0.2	0.5	0.2	0.1
$s =$ *bad*	0	0.2	0.5	0.3



Decision strategy

- **Decision strategy**: a rule selecting a decision for any given value of the measured attribute(s).
- i.e. function $d = \delta(x)$.
- Example of husband's possible strategies:

$\delta(x)$	$x =$ *mild*	$x =$ *irritated*	$x =$ *upset*	$x =$ *alarming*
$\delta_1(x) =$	*nothing*	*nothing*	*pizza*	*g.T.c.*
$\delta_2(x) =$	*nothing*	*pizza*	*g.T.c.*	*g.T.c.*
$\delta_3(x) =$	*g.T.c.*	*g.T.c.*	*g.T.c.*	*g.T.c.*
$\delta_4(x) =$	*nothing*	*nothing*	*nothing*	*nothing*

- Overall, $3^4 = 81$ possible strategies (3 possible decisions for each of the 4 possible attribute values).
- How to define which strategy is best? How to sort them by quality?
- Define the **risk of a strategy** for state s: mean loss value conditioned on s.

$$R(\delta, s) = \sum_x l(s, \delta(x)) P(x|s)$$

MiniMax

- **Example:** risk of strategy δ_1 at state $s = \textit{good}$ is

$$R(\delta_1, \textit{good}) = l(\textit{good}, \delta_1(\textit{mild})) \cdot P(\textit{mild}|\textit{good}) + l(\textit{good}, \delta_1(\textit{irritated})) \cdot P(\textit{irritated}|\textit{good})$$
$$+ l(\textit{good}, \delta_1(\textit{upset})) \cdot P(\textit{upset}|\textit{good}) + l(\textit{good}, \delta_1(\textit{alarming})) \cdot P(\textit{alarming}|\textit{good})$$
$$= l(\textit{good}, \textit{nothing}) \cdot 0.5 + l(\textit{good}, \textit{nothing}) \cdot 0.4 + l(\textit{good}, \textit{pizza}) \cdot 0.1 + l(\textit{good}, \textit{g.T.c.}) \cdot 0$$
$$= 0 \cdot 0.5 + 0 \cdot 0.4 + 2 \cdot 0.1 + 4 \cdot 0 = 0.2$$

- Similarly: $R(\delta_1, \textit{average}) = 4.4$ a $R(\delta_1, \textit{good}) = 8.3$
- **Maximum risk** of strategy δ_1 (over all possible states) is thus 8.3.
- Similarly: maximum risk of strategy δ_3 is 6.
- **MiniMax** criterion: out of two strategies, whichever has a smaller maximum risk is superior.
- Thus δ_3 is better than δ_1 by MiniMax.
- The best strategy δ^* by Minimax is one that **minimizes the maximum risk**:

$$\delta^* = \arg\min_{\delta} \max_{s} R(\delta, s)$$

Bayesian decision making

- What if husband knows that wife *usually is feeling fine*? More generally: he knows how probable her state of minds are, i.e. he knows the distribution $P(s)$. For example:

	$s =$ *good*	$s =$ *average*	$s =$ *bad*
$P(s) =$	0.7	0.2	0.1

- Note that these probabilities do not influence MiniMax-based decisions.
- Given $P(s)$ we can calculate the **mean risk** of a strategy over all possible states:

$$r(\delta) = \sum_s R(\delta, s)P(s)$$

- For example.

$$r(\delta_1) = 0.2 \cdot 0.7 + 4.4 \cdot 0.2 + 8.3 \cdot 0.1 = 1.85$$

$$r(\delta_3) = 4 \cdot 0.7 + 5 \cdot 0.2 + 6 \cdot 0.1 = 4.4$$

- **Bayes criterion**: out of two strategies choose the one with smaller mean risk. From the Bayesian viewpoint δ_1 is superior to δ_3.
- In this case, contrary to MiniMax!

Bayes optimal strategy

- The **Bayes optimal strategy**: one minimizing mean risk. That is

$$\delta^* = \arg\min_{\delta} r(\delta)$$

- From $P(x|s)P(s) = P(s|x)P(x)$ (Bayes rule), we have

$$r(\delta) = \sum_s R(\delta, s)P(s) = \sum_s \sum_x l(s, \delta(x))P(x|s)P(s)$$

$$= \sum_s \sum_x l(\delta(x), s)P(s|x)P(x) = \sum_x P(x) \underbrace{\sum_s l(s, \delta(x))P(s|x)}_{\text{Conditional risk}}$$

- The optimal strategy is obtained by minimizing the conditional risk separately for each x:

$$\delta^*(x) = \arg\min_{d} \sum_s l(s, d)P(s|x)$$

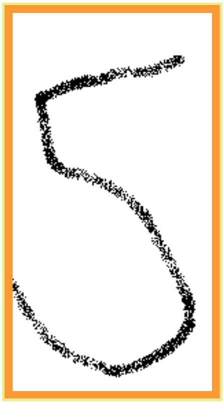
- Unlike for MiniMax, there is no need to evaluate the risk of all possible strategies. The Bayes optimal strategy can be calculated point-wise by determining the optimal decision for individual attribute values x.

Statistical decision making: wrapping up

- **Given**:
  - A set of possible **states**: $\mathcal{S}$
  - A set of possible **decisions**: $\mathcal{D}$
  - A **loss function** $l : \mathcal{D} \times \mathcal{S} \to \Re$
  - The range $\mathcal{X}$ of the **attribute**
  - Distribution $P(x|s)$, $x \in \mathcal{X}, s \in \mathcal{S}$.
- **Define**:
  - **Strategy**: function $\delta : \mathcal{X} \to \mathcal{D}$
  - **Risk of strategy** δ at state $s \in \mathcal{S}$: $R(\delta, s) = \sum_x l(s, \delta(x))P(x|s)$
- **MiniMaxov problem**:
  - Further given: admissible strategy set Δ.
  - Goal: find the optimal strategy $\delta^* = \arg\min_{\delta \in \Delta} \max_{s \in \mathcal{S}} R(\delta, s)$
- **Bayes problem**:
  - Further given: distribution $P(s)$, $s \in \mathcal{S}$.
  - Further define: **mean risk of strategy** δ: $r(\delta) = \sum_s R(\delta, s)P(s)$
  - Goal: find the optimal strategy $\delta^* = \arg\min_{\delta \in \Delta} r(\delta)$
  - Solution: $\delta^*(x) = \arg\min_d \sum_s l(s, d)P(s|x)$

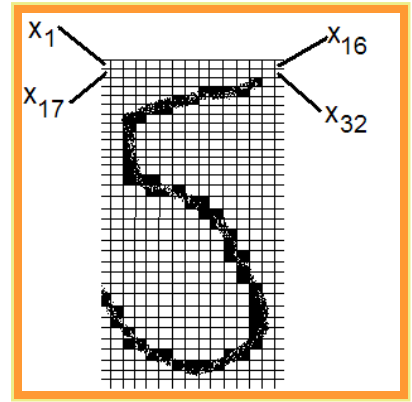
Pattern recognition

- **Example** task:



What digit is this?

can formulate as a statistical decision task



Attribute = pixel value vector.

- **Attribute-based recognition** of digits: **classification** into on of **classes** $0 \ldots 9$ by the attribute vector.
- A special case of statistical decision theory:
  - Attribute vector $\vec{x} = (x_1, x_2, \ldots)$: pixels # 1, 2, ...
  - **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \ldots 9\}$.**
  - State = actual class, Decision = recognized class.
  - Loss function:

$$l(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

- Mean risk = mean classification **error**.

Bayes classification

- Usually required: minimize mean error ⇒ Bayes classification task.
- Optimal classification of $\vec{x}$:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{l(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_s [1 - P(s|\vec{x})] = \arg\max_s P(s|\vec{x})$$

- We thus choose the most probable class for a given attribute vector.
- Usually we **are not given** $P(s|\vec{x})$ but only a finite (multi)set of
- **Training examples** $(\vec{x}_1, s_1), (\vec{x}_2, s_2), \ldots (\vec{x}_l, s_l)$ drawn i.i.d from $P(\vec{x}, s)$.
- We might want to estimate $P(s|\vec{x})$

$$P(s|\vec{x}) \approx \frac{\#\text{ examples where} \vec{x}_i = \vec{x} \text{ and } s_i = s}{\#\text{ examples where } \vec{x}_i = \vec{x}}$$

- This is usually impossible:
  - X may be uncountable ($\vec{x}$ continuous). OK, discretization possible.
  - To estimate $P(s|\vec{x})$ with a fixed accuracy, we need $O(\exp(\text{n}))$ examples (n ... width of $\vec{x}$).
  - Combinatorial curse.
  - Bayes classification provides a lower bound on classification error, but that is usually not achievable because $P(s|\vec{x})$ is not known.

Naive Bayes classification

- For efficient classification we must thus rely on additional assumptions. A basic example:
- In the **exceptional case** of **statistical independence** between $x(i)$ components for each class s it holds

$$\boxed{P(\vec{x}|s) = P(x(1)|s) \cdot P(x(2)|s) \cdot \ldots}$$

- Use simple Bayes law and maximize:

$$P(s|\vec{x}) = \frac{P(\vec{x}|s)P(s)}{P(\vec{x})} = \frac{P(s)}{P(\vec{x})} P(x(1)|s) \cdot P(x(2)|s) \cdot \ldots =$$

- No combinatorial curse in estimating $P(s)$ and $P(x(i)|s)$ separately for each i and s.
- No need to estimate $P(\vec{x})$. (Why?)
- N.B. $P(s)$ may be provided apriori.
- **Naive** $=$ when used despite statistical dependence btw. $x(i)$'s.

Neighbor-based classification

- Assumption: similar objects fall in the same class.
- *Similarity* - small *distance* in X.
- A fuction, called a **metric**: $\rho : X \times X \rightarrow \Re$ such that $\forall x, y, z$
  - $\rho(x, y) \geq 0$
  - $\rho(x, x) = 0$
  - $\rho(x, y) = \rho(y, x)$
  - $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$
- **Examples:**
  - **Euclidean metric** for $X = \Re^n$:

$$\boxed{\rho_E(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_i (x_1(i) - x_2(i))^2}}$$

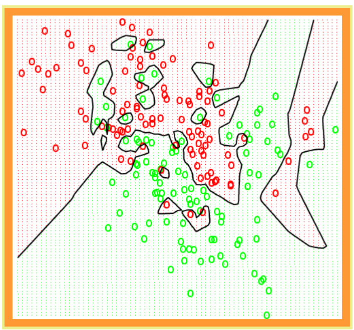
  - For $X = \{0, 1\}^n$, ρ_E^2 is equal to the **Hamming metric**, giving the number of non-equal corresponding components.

k-NN

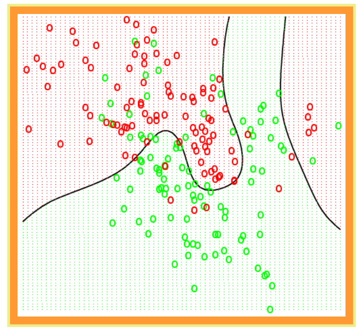
- **k-nearest neighbor classification, k-NN.**
- Given:
  - $k \in N$
  - Training examples: $(\vec{x}_1, s_1), (\vec{x}_2, s_2), \ldots (\vec{x}_l, s_l)$
  - Metric $\rho : X \times X \rightarrow \Re$
- Goal: classify $\vec{x}_{l+1}$
- Approach: choose k nearest (to $\vec{x}$ by ρ) examples. Let the majority class therein be the class for $\vec{x}_{l+1}$.

Classification flexibility

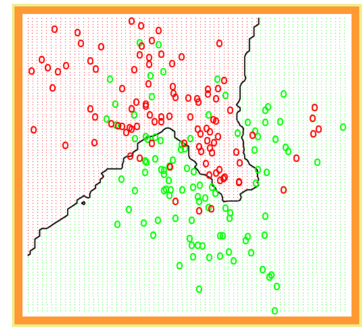
- How to choose k?
- **A general trend**: Consider a two-class problem (red/green) with noisy training examples (some s_i misclassified).



$k = 1$: Good fit of training data, small tolerance to noise.



Bayes classifier: less flexible than 1-nn, more flexible than 15-nn.



$k = 15$: Poor fit to training data. Small sensitivity to noise.

- Note: the shown Bayes classifier was constructed from **known** $P(s|\vec{x})$.
- Observation: with flexibility too large (small k) or too small (large k), one gets classifiers very different from the optimal B/C.
- Optimal k somewhere in the middle. Still pending: how to determine the best value?

Validation

- Mean risk $r(\delta)$ of classifier δ corresponds to the relative frequency of its misclassifications (convergence in the limit...), or 'error rate'.
- Define **training error** $TE(\delta)$ as the error rate on **v training data**.
- Is $TE(\delta)$ a good estimate of $r(\delta)$?
- Earlier: 1-nn is not a good classifier, despite having training error 0.
- $TE(\delta)$ is (usually) not a good estimate of $r(\delta)$ because it is biased. To estimate $r(\delta)$ in an unbiased way:
  - split available data into a **training set** $(\vec{x}_1, s_1), \ldots (\vec{x}_l, s_l)$ and an independent **testing set** $(\vec{x}_{l+1}, s_{l+1}), \ldots (\vec{x}_{l+m}, s_{l+m})$
  - (e.g. by a 75% - 25% split).
  - Construct (train) classifier on the training set.
- Error rate on the testing set is an **unbiased** estimate of $r(\delta)$.
- Unbiased does not mean accurate.