Supervised and unsupervised learning.

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

This lecture is based on the book
Ten Lectures on Statistical and Structural Pattern Recognition
by Michail I. Schlesinger and Václav Hlaváč (Kluwer, 2002).
(V české verzi kniha vyšla ve vydavatelství ČVUT v roce 1999 pod názvem
Deset přednášek z teorie statistického a strukturálního rozpoznávání).
Learning

Decision strategy design
Learning as parameter estimation
Learning as optimal strategy selection
Several surrogate criteria
Learning revisited

Unsupervised Learning

Clustering

Summary

Learning
Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \rightarrow D$ which would be optimal with respect to certain criterion.
Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \rightarrow D$ which would be optimal with respect to certain criterion.

**Bayesian decision theory** requires complete statistical information $p_{X,K}(x,k)$ of the object of interest to be known, and a suitable penalty function $W : K \times D \rightarrow \mathcal{R}$ must be provided.

**Non-Bayesian decision theory** studies tasks for which some of the above information is not available. In practical applications, typically, none of the probabilities are known! The designer is only provided with the training (multi)set $T = \{(x_1,k_1), (x_2,k_2), \ldots, (x_l,k_l)\}$ of examples.

- ✔️ It is simpler to provide good examples than to gain complete or partial statistical model, build general theories, or create explicit descriptions of concepts (hidden states).
- ✔️ The aim is to find definitions of concepts (classes, hidden states) which are
  - ✗ complete (all positive examples are satisfied), and
  - ✗ consistent (no negative examples are satisfied).
- ✔️ The training (multi)set is finite, the found concept description is only a hypothesis.
Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \rightarrow D$ which would be optimal with respect to certain criterion.

**Bayesian decision theory** requires complete statistical information $p_{X|K}(x, k)$ of the object of interest to be known, and a suitable penalty function $W : K \times D \rightarrow \mathcal{R}$ must be provided.

**Non-Bayesian decision theory** studies tasks for which some of the above information is not available. In practical applications, typically, none of the probabilities are known! The designer is only provided with the training (multi)set $T = \{(x_1, k_1), (x_2, k_2), \ldots, (x_l, k_l)\}$ of examples.

- It is simpler to provide good examples than to gain complete or partial statistical model, build general theories, or create explicit descriptions of concepts (hidden states).
- The aim is to find definitions of concepts (classes, hidden states) which are
  - complete (all positive examples are satisfied), and
  - consistent (no negative examples are satisfied).

- The training (multi)set is *finite*, the found concept description is only a *hypothesis*.

When do we need to use learning?

- When knowledge about the recognized object is insufficient to solve the PR task.
- Most often, we have insufficient knowledge about $p_{X|K}(x|k)$. 
Using an observation \( x \in X \) of an object of interest with a hidden state \( k \in K \), we should design a decision strategy \( q : X \to D \) which would be optimal with respect to certain criterion.

**Bayesian decision theory** requires complete statistical information \( p_{X|K}(x,k) \) of the object of interest to be known, and a suitable penalty function \( W : K \times D \to \mathbb{R} \) must be provided.

**Non-Bayesian decision theory** studies tasks for which some of the above information is not available. In practical applications, typically, none of the probabilities are known! The designer is only provided with the training (multi)set \( T = \{(x_1, k_1), (x_2, k_2), \ldots, (x_l, k_l)\} \) of examples.

- ✔️ It is simpler to provide good examples than to gain complete or partial statistical model, build general theories, or create explicit descriptions of concepts (hidden states).
- ✔️ The aim is to find definitions of concepts (classes, hidden states) which are
  - ✗ complete (all positive examples are satisfied), and
  - ✗ consistent (no negative examples are satisfied).
- ✔️ The training (multi)set is *finite*, the found concept description is only a *hypothesis*.

When do we need to use learning?

- ✔️ When knowledge about the recognized object is insufficient to solve the PR task.
- ✔️ Most often, we have insufficient knowledge about \( p_{X|K}(x|k) \).

How do we proceed?
1. **Assume** $p_{XK}(x,k)$ has a particular form (e.g. Gaussian, mixture of Gaussians, piece-wise constant) with a small number of parameters $\Theta_k$.

2. **Estimate** the values of parameters $\Theta_k$ using the training set $T$.

3. **Solve** the classifier design problem as if the estimated $\hat{p}_{XK}(x,k)$ was the true (and unknown) $p_{XK}(x,k)$.
Learning as parameter estimation

1. **Assume** $p_{XK}(x,k)$ has a particular form (e.g. Gaussian, mixture of Gaussians, piece-wise constant) with a small number of parameters $\Theta_k$.

2. **Estimate** the values of parameters $\Theta_k$ using the training set $T$.

3. **Solve** the classifier design problem as if the estimated $\hat{p}_{XK}(x,k)$ was the true (and unknown) $p_{XK}(x,k)$.

Pros and cons:
- ✓ If the true $p_{XK}(x,k)$ does not have the assumed form, the resulting strategy $q'(x)$ can be arbitrarily bad, even if the training set size $L$ approaches infinity.
- ✓ Implementation is often straightforward, especially if the parameters $\Theta_k$ are assumed to be independent for each class (naive bayes classifier).
Learning as optimal strategy selection

- Choose a class $Q$ of strategies $q_Θ : X \rightarrow D$. The class $Q$ is usually given as a parametrized set of strategies of the same kind, i.e. $q_Θ(x, Θ_1, \ldots, Θ_{|K|})$.

- The problem can be formulated as a non-Bayesian task with non-random interventions:
  - The unknown parameters $Θ_k$ are the non-random interventions.
  - The probabilities $p_{X|K,Θ}(x|k,Θ_k)$ must be known.
  - The solution may be e.g. such a strategy that minimizes the maximal probability of incorrect decision over all $Θ_k$, i.e. strategy that minimizes the probability of incorrect decision in case of the worst possible parameter settings.
  - But even this minimal probability may not be low enough—this happens especially in cases when the class $Q$ of strategies is too broad.
  - It is necessary to narrow the set of possible strategies using additional information—the training (multi)set $T$.

- **Learning** then amounts to selecting a particular strategy $q^*_Θ$ from the a priori known set $Q$ using the information provided as training set $T$.
  - Natural criterion for the selection of one particular strategy is the risk $R(q_Θ)$, but it cannot be computed because $p_{X,K}(x,k)$ is unknown.
  - The strategy $q^*_Θ \in Q$ is chosen by minimizing some other surrogate criterion on the training set which approximates $R(q_Θ)$.
  - The choice of the surrogate criterion determines the learning paradigm.
Several surrogate criteria

All the following surrogate criteria can be computed using the training data $T$.

Learning as parameter estimation

✔ according to the **maximum likelihood**.
✔ according to a **non-random training set**.

Learning as optimal strategy selection

✔ by **minimization of the empirical risk**.
✔ by **minimization of the structural risk**.
Several surrogate criteria

All the following surrogate criteria can be computed using the training data $T$.

Learning as parameter estimation

- according to the **maximum likelihood**.
  - The likelihood of an instance of the parameters $\Theta = (\Theta_k : k \in K)$ is the probability of $T$ given $\Theta$:
    
    $$L(\Theta) = p(T|\Theta) = \prod_{(x_i,k_i) \in T} p_K(k_i)p_X|K(x_i|k_i, \Theta_{k_i})$$

  - Learning then means to find $\Theta^*$ that maximizes the probability of $T$:
    $$\Theta^* = (\Theta^*_k : k \in K) = \arg \max_{\Theta} L(T, \Theta)$$

    which can be decomposed to
    $$\Theta^*_k = \arg \max_{\Theta_k} \sum_{x \in X} \alpha(x,k) \log p_X|K(x|k, \Theta_k),$$

    where $\alpha(x,k)$ is the frequency of the pair $(x,k)$ in $T$ (i.e. $T$ is multiset).

- The recognition is then performed according to $q_{\Theta}(x, \Theta^*)$.

- according to a **non-random training set**.

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
- by **minimization of the structural risk**.
Several surrogate criteria

All the following surrogate criteria can be computed using the training data $T$.

Learning as parameter estimation

✓ according to the **maximum likelihood**.
✓ according to a **non-random training set**.
   ✗ When random examples are not easy to obtain, e.g. in recognition of images.
   ✗ $T$ is carefully crafted by the designer:
      ✓ it should cover the whole recognized domain
      ✓ the examples should be typical (“quite probable”) prototypes

✗ Let $T(k), k \in K$, be a subset of the training set $T$ with examples for state $k$. Then

$$
\Theta^*_k = \arg \max_{\Theta_k} \min_{x \in T(k)} p_{X|K}(x|k, \Theta_k)
$$

✗ Note that the $\Theta^*$ does not depend on the frequencies of $(x, k)$ in $T$ (i.e. $T$ is a set).

Learning as optimal strategy selection

✓ by **minimization of the empirical risk**.
✓ by **minimization of the structural risk**.
Several surrogate criteria

All the following surrogate criteria can be computed using the training data $T$.

Learning as parameter estimation

✓ according to the **maximum likelihood**.
✓ according to a **non-random training set**.

Learning as optimal strategy selection

✓ by **minimization of the empirical risk**.

- The set $Q$ of parametrized strategies $q(x, \Theta)$, penalty function $W(k, d)$.
- The quality of each strategy $q \in Q$ (i.e. the quality of each parameter set $\Theta$) could be described by the risk

$$R(\Theta) = R(q) = \sum_{k \in K} \sum_{x \in X} p_{X,K}(x,k) W(k, q(x, \Theta)),$$

but $p_{X,K}$ is unknown.

✓ We thus use the **empirical risk** $R_{\text{emp}}$ (training set error):

$$R_{\text{emp}}(\Theta) = R_{\text{emp}}(q) = \frac{1}{|T|} \sum_{(x_i, k_i) \in T} W(k_i, q(x_i, \Theta)).$$

✓ Strategy $q_{\Theta}(x, \Theta^*)$ is used where $\Theta^* = \arg \min_{\Theta} R_{\text{emp}}(\Theta)$.
✓ Examples: Perceptron, neural networks (backprop.), classification trees, …

✓ by **minimization of the structural risk**.
Several surrogate criteria

All the following surrogate criteria can be computed using the training data $T$.

Learning as parameter estimation

✔ according to the **maximum likelihood**.
✔ according to a **non-random training set**.

Learning as optimal strategy selection

✔ by **minimization of the empirical risk**.
✔ by **minimization of the structural risk**.

✖ Based on Vapnik-Chervonenkis theory
✖ Examples: Optimal separating hyperplane, support vector machine (SVM)
Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method \textit{without learning}.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method \textit{with learning}. 
Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method *without learning*.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method *with learning*.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.
Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method without learning.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method with learning.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.

The solution

- is a parametrized strategy and
- its parameters are learned from the training (multi)set.
Learning revisited

Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method \textit{without learning}.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method \textit{with learning}.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.

The solution

- is a parametrized strategy and
- its parameters are learned from the training (multi)set.

The \textit{supervised learning} is a topic for several upcoming lectures:

- Decision trees and decision rules.
- Linear classifiers.
- Adaboost.
Unsupervised Learning

Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture
distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised
learning?

Summary
No learning:

- Use $q$ for recognition.

\[ x \rightarrow q(x) \rightarrow d \]
Do we need the teacher?

No learning:
✓ Use $q$ for recognition.

Supervised learning:
✓ First, use $T$ to learn $\Theta$.
✓ Then, use $q_\Theta$ for recognition.
Do we need the teacher?

No learning:

- Use $q$ for recognition.

Supervised learning:

- First, use $T$ to learn $\Theta$.
- Then, use $q_\Theta$ for recognition.

Unsupervised learning???

- First, predict the sequence $\tilde{K}^0$, i.e. $k_i^0 = R(x_i, \Theta^0), x_i \in \tilde{X}$.
- Iteratively update $\Theta^t = L(\tilde{X}, \tilde{K}^{t-1})$.

Does not work as expected, if models like perceptron are used.

Works for quadratic clustering (k-means, ISODATA, EM).
Assume:

 ✓ An object can be in one of the $|K|$ states with equal probabilities.
 ✓ All $p_{X|K}(x|k)$ are isotropic Gaussians: $p_{X|K}(x|k) = \mathcal{N}(x|\mu_k, \sigma I)$. 
K-means (ISODATA)

Assume:

✔ An object can be in one of the \(|K|\) states with equal probabilities.
✔ All \(p_{X|K}(x|k)\) are isotropic Gaussians: \(p_{X|K}(x|k) = \mathcal{N}(x|\mu_k, \sigma I)\).

Recognition:

✔ The task is to decide the state \(k\) for each \(x\), assuming all \(\mu_k\) are known.
✔ The Bayesian strategy (minimizes the probability of error):

\[
q^*(x) = \arg \min_{k \in K} (x - \mu_k)^2
\]

✔ If \(\mu_k, k \in K\), are not known, it is a parametrized strategy \(q_\Theta(x)\), where \(\Theta = (\mu_k)_{k=1}^K\).
K-means (ISODATA)

Assume:

- An object can be in one of the $|K|$ states with equal probabilities.
- All $p_{X|K}(x|k)$ are isotropic Gaussians: $p_{X|K}(x|k) = \mathcal{N}(x|\mu_k, \sigma I)$.

Recognition:

- The task is to decide the state $k$ for each $x$, assuming all $\mu_k$ are known.
- The Bayesian strategy (minimizes the probability of error):

$$ q^*(x) = \arg \min_{k \in K} (x - \mu_k)^2 $$

- If $\mu_k, k \in K$, are not known, it is a parametrized strategy $q_{\Theta}(x)$, where $\Theta = (\mu_k)_{k=1}^K$.

Learning:

- Find the maximum-likelihood estimates of $\mu_k$ based on known $(x_1, k_1), \ldots, (x_l, k_l)$:

$$ \mu_k^* = \frac{1}{|I_k|} \sum_{i \in I_k} x_i, $$

where $I_k$ is a set of indices of training examples belonging to state $k$. 
K-means algorithm

Algorithm K-means [Mac67]

✓ $K$ is the apriori given number of clusters.
✓ Algorithm:

1. Choose $K$ centroids $\mu_k$ (in almost any way, but every cluster should have at least one example.)
2. For all $x$, assign $x$ to its closest $\mu_k$.
3. Compute the new position of centroids $\mu_k$ based on all examples $x_i, i \in I_k$, in cluster $k$.
4. If the positions of centroids changed, repeat from 2.
K-means algorithm

Algorithm K-means [Mac67]

- K is the apriori given number of clusters.
- Algorithm:
  1. Choose K centroids \( \mu_k \) (in almost any way, but every cluster should have at least one example.)
  2. For all \( x \), assign \( x \) to its closest \( \mu_k \).
  3. Compute the new position of centroids \( \mu_k \) based on all examples \( x_i, i \in I_k \), in cluster \( k \).
  4. If the positions of centroids changed, repeat from 2.

Algorithm features:

- Algorithm minimizes the function (intracluster variance):

\[
J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} |x_{i,j} - c_j|^2
\]  

- Algorithm is fast, but each time it can converge to a different local optimum of \( J \).

K-means clustering: iteration 1
Learning
Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary

K-means clustering: iteration 2
K-means clustering: iteration 3
K-means clustering: iteration 4
Do we need the teacher?

K-means clustering: iteration 5
K-means clustering: iteration 6
General mixture distributions

Assume the data are samples from a distribution factorized as

\[ p_{X|K}(x, k) = p_K(k)p_{X|K}(x|k), \text{i.e.} \]

\[ p_X(x) = \sum_{k \in K} p_K(k)p_{X|K}(x|k) \]

and that the distribution is known.
General mixture distributions

Assume the data are samples from a distribution factorized as

\[ p_{X,K}(x,k) = p_K(k)p_{X|K}(x|k), \text{ i.e.} \]
\[ p_X(x) = \sum_{k \in K} p_K(k)p_{X|K}(x|k) \]

and that the distribution is known.

Recognition:

- Let’s define the result of recognition not as a single decision for some state \( k \), but rather as
- a set of posterior probabilities (sometimes called \textit{responsibilities}) for all \( k \) given \( x_i \)

\[ \gamma_k(x_i) = p_{K|X}(k|x_i) = \frac{p_{X|K}(x_i|k)p_K(k)}{\sum_{k \in K} p_{X|K}(x_i|k)p_K(k)} \]

that an object was in state \( k \) when observation \( x_i \) was made.
- The \( \gamma_k(x) \) functions can be viewed as discriminant functions.
Learning:

- Given the training multiset $T = (x_i, k_i)_{i=1}^n$ (or the respective $\gamma_k(x_i)$ instead of $k_i$),
- assume $\gamma_k(x)$ is known, $p_K(k)$ are not known, and $p_{X|K}(x|k)$ are known except the parameter values $\Theta_k$, i.e. we shall write $p_{X|K}(x|k, \Theta_k)$.
- Let the object model $m$ be a “set” of all unknown parameters $m = (p_K(k), \Theta_k)_{k \in K}$. 
Learning:

- Given the training multiset \( T = (x_i, k_i)_{i=1}^n \) (or the respective \( \gamma_k(x_i) \) instead of \( k_i \)),
- assume \( \gamma_k(x) \) is known, \( p_K(k) \) are not known, and \( p_{X|K}(x|k) \) are known except the parameter values \( \Theta_k \), i.e. we shall write \( p_{X|K}(x|k, \Theta_k) \).
- Let the object model \( m \) be a “set” of all unknown parameters \( m = (p_K(k), \Theta_k)_{k \in K} \).
- The log-likelihood of the model \( m \):

\[
\log L(m) = \log \prod_{i=1}^n p_{X|K}(x_i, k_i) = \sum_{i=1}^n \log p_K(k_i) + \sum_{i=1}^n \log p_{X|K}(x_i | k_i, \Theta_{k_i})
\]

- The log-likelihood using \( \gamma \):

\[
\log L(m) = \sum_{i=1}^n \sum_{k \in K} \gamma_k(x_i) \log p_K(k) + \sum_{i=1}^n \sum_{k \in K} \gamma_k(x_i) \log p_{X|K}(x_i | k, \Theta_k)
\]

- We search for the optimal model using maximal likelihood:

\[
m^* = (p_K^*(k), \Theta_k^*) = \arg \max_m \log L(m)
\]

- i.e. we compute

\[
p_K^*(k) = \frac{1}{n} \sum_{i=1}^n \gamma_k(x_i) \text{ and solve } k \text{ independent tasks}
\]

\[
\Theta_k^* = \arg \max_{\Theta_k} \sum_{i=1}^n \gamma_k(x_i) \log p_{X|K}(x_i | k, \Theta_k).
\]
Expectation Maximization Algorithm

Unsupervised learning algorithm [DLR77] for general mixture distributions:

1. Initialize the model parameters \( m = ((p_K(k), \Theta_k) \forall k) \).
2. Perform the **recognition** task, i.e. assuming \( m \) is known, compute
   \[
   \gamma_k(x_i) = \hat{p}_{K|X}(k|x_i) = \frac{p_K(k)p_{X|K}(x_i|k, \Theta_k)}{\sum_{j \in K} p_K(j)p_{X|K}(x_i|j, \Theta_j)}.
   \]
3. Perform the **learning** task, i.e. assuming \( \gamma_k(x_i) \) are known, update the ML estimates of the model parameters \( p_K(k) \) and \( \Theta_k \) for all \( k \):
   \[
   p_K(k) = \frac{1}{n} \sum_{i=1}^{n} \gamma_k(x_i)
   \]
   \[
   \Theta_k = \arg \max_{\Theta_k} \sum_{i=1}^{n} \gamma_k(x_i) \log p_{X|K}(x_i|k, \Theta_k)
   \]
4. Iterate 2 and 3 until the model stabilizes.
Expectation Maximization Algorithm

Unsupervised learning algorithm [DLR77] for general mixture distributions:

1. Initialize the model parameters \( m = ((p_K(k), \Theta_k) \forall k) \).

2. Perform the **recognition** task, i.e. assuming \( m \) is known, compute

\[
\gamma_k(x_i) = \hat{p}_{K|X}(k|x_i) = \frac{p_K(k)p_{X|K}(x_i|k, \Theta_k)}{\sum_{j \in K} p_K(j)p_{X|K}(x_i|j, \Theta_j)}.
\]

3. Perform the **learning** task, i.e. assuming \( \gamma_k(x_i) \) are known, update the ML estimates of the model parameters \( p_K(k) \) and \( \Theta_k \) for all \( k \):

\[
p_K(k) = \frac{1}{n} \sum_{i=1}^{n} \gamma_k(x_i)
\]

\[
\Theta_k = \arg \max_{\Theta_k} \sum_{i=1}^{n} \gamma_k(x_i) \log p_{X|K}(x_i|k, \Theta_k)
\]

4. Iterate 2 and 3 until the model stabilizes.

Features:

- The algorithm does not specify how to update \( \Theta_k \) in step 3, it depends on the chosen form of \( p_{X|K} \).
- The model created in iteration \( t \) is always at least as good as the model from iteration \( t - 1 \), i.e. \( L(m) = p(T|m) \) increases.

Each $k$th component is a Gaussian distribution:

$$
\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{D/2} |\Sigma_k|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}
$$

Gaussian Mixture Model (GMM):

$$
p(x) = \sum_{k=1}^{K} p_K(k) p_{X|K}(x|k, \Theta_k) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x|\mu_k, \Sigma_k)
$$

assuming $\sum_{k=1}^{K} \alpha_k = 1$ and $0 \leq \alpha_k \leq 1$
1. Initialize the model parameters \( m = ((p_K(k), \mu_k, \Sigma_k) \forall k) \).

2. Perform the **recognition** task as in the general case, i.e. assuming \( m \) is known, compute

\[
\gamma_k(x_i) = \hat{p}_{K|X}(k|x_i) = \frac{p_K(k)p_{X|K}(x_i|k, \Theta_k)}{\sum_{j \in K} p_K(j)p_{X|K}(x_i|j, \Theta_j)} = \frac{\alpha_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j \in K} \alpha_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}.
\]

3. Perform the **learning** task, i.e. assuming \( \gamma_k(x_i) \) are known, update the ML estimates of the model parameters \( \alpha_k, \mu_k \) and \( \Sigma_k \) for all \( k \):

\[
\begin{align*}
\alpha_k &= p_K(k) = \frac{1}{n} \sum_{i=1}^{n} \gamma_k(x_i) \\
\mu_k &= \frac{\sum_{i=1}^{n} \gamma_k(x_i)x_i}{\sum_{i=1}^{n} \gamma_k(x_i)} \\
\Sigma_k &= \frac{\sum_{i=1}^{n} \gamma_k(x_i)(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} \gamma_k(x_i)}
\end{align*}
\]

4. Iterate 2 and 3 until the model stabilizes.
Example: Source data

Source data generated from 3 Gaussians.
The data were given to the EM algorithm as an unlabeled dataset.
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary

Learning
Learning
Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Unsupervised Learning
- Do we need the teacher?
- K-means (ISODATA)
- K-means algorithm
- General mixture distributions
- EM Algorithm
- GMM

EM for GMM
- What is unsupervised learning?
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Unsupervised Learning
- Do we need the teacher?
- K-means (ISODATA)
- K-means algorithm
- General mixture distributions
- EM Algorithm
- GMM

EM for GMM
- What is unsupervised learning?

Clustering

Summary
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Learning
Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
Example: EM Iterations

Unsupervised Learning
Do we need the teacher?
K-means (ISODATA)
K-means algorithm
General mixture distributions
EM Algorithm
GMM
EM for GMM
What is unsupervised learning?
Clustering
Summary
The ground truth (left) and the EM estimate (right) are very close because

- we have enough data,
- we know the right number of components, and
- we were lucky that EM converged to the right local optimum of the likelihood function.
A strict view:

- Only those algorithms conforming to the scheme $\Theta^t = L(\tilde{X}, R(\tilde{X}, \Theta^{t-1}))$, i.e.
- only k-means, ISODATA, EM algorithm, ....
**What is unsupervised learning?**

**A strict view:**
- ✓ Only those algorithms conforming to the scheme $\Theta^t = L(\tilde{X}, R(\tilde{X}, \Theta^{t-1}))$, i.e.
- ✓ only k-means, ISODATA, EM algorithm, ... .

**A broader view:**
- ✓ any algorithm that analyses a dataset and extracts potentially usable information just on the basis of $x$, i.e. without knowing $k$.
- ✓ **Clustering** creates dissimilar groups of similar objects.
- ✓ **Vector quantization** searches for several typical prototypes, reduces amount of data.
- ✓ **Outlier detection** searches for unusual (non-probable) examples.
- ✓ **Dimensionality reduction** chooses/creates a low number of (artificial) variables that describe the data sufficiently well.
- ✓ **Feature extraction** derives new features describing the data.
What is unsupervised learning?

A strict view:
- ✔ Only those algorithms conforming to the scheme $\Theta^t = L(\tilde{X}, R(\tilde{X}, \Theta^{t-1}))$, i.e.
- ✔ only k-means, ISODATA, EM algorithm, ....

A broader view:
- ✔ any algorithm that analyses a dataset and extracts potentially usable information just on the basis of $x$, i.e. without knowing $k$.
- ✔ Clustering creates dissimilar groups of similar objects.
- ✔ Vector quantization searches for several typical prototypes, reduces amount of data.
- ✔ Outlier detection searches for unusual (non-probable) examples.
- ✔ Dimensionality reduction chooses/creates a low number of (artificial) variables that describe the data sufficiently well.
- ✔ Feature extraction derives new features describing the data.

Intuition:
- ✔ based on the “structure” hidden in the data, we want to derive new features, but
- ✔ we often do not know what these features should mean.
Clustering

Clustering

Clustering

Similarity

Clustering algorithms

Hierarchical clustering

Summary
The goal is to assign a certain set of objects into groups (called clusters) so that the objects in the same cluster are more similar to each other than objects from different clusters. (The result of clustering does not need to be a rule how to assign points to clusters).

A cluster is a set of objects which
1. are similar to each other and
2. dissimilar to objects from other clusters.

Issues:
- What do we mean by “similar” and “dissimilar”?
- What is the right number of clusters?

Cluster analysis studies algorithms for
- cluster formation, i.e. how to divide a set of objects into clusters,
- segmentation, i.e. how to describe borders of individual clusters or their prototypes (centroids),
- cluster labeling, i.e. how to assign meaningful labels to individual clusters.
**Similarity**

*Similarity* (or rather dissimilarity) is usually expressed by a distance:

- Minkowski metric:
  
  $$ d(x_1, x_2) = \left( \sum_{i=1}^{D} |x_{1,i} - x_{2,i}|^q \right)^{\frac{1}{q}} $$

  - $q = 1$: $L_1$, Manhattan, city-block, postman metric
  - $q = 2$: $L_2$, Euclidean distance
  - $q = \infty$: $L_\infty$

- Cosine distance: e.g. to assess the similarity of 2 documents (word frequency):
  
  $$ d(x_1, x_2) = \frac{x_1^T x_2}{||x_1|| ||x_2||} $$

- Mahalanobis distance: covariance matrix driven metric (see the multivariate Gaussian distribution):
  
  $$ d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)} $$
Hierarchical algorithms build the clusters incrementally using already found clusters, changing their number.
- Agglomerative (bottom-up)
- Divisive (top-down)

Partitioning algorithms refine the definition of apriori given number of clusters iteratively.
- K-means
- ISODATA
- fuzzy C-means
- EM algoritmus pro směs Gaussiánů

Graph theory methods: minimum spanning tree

Spectral clustering: based on the distance matrix, they reduce the space dimensionality, clustering is performed in the lower-dimensional space.

Kohonen maps
Hierarchical clustering

Agglomerative clustering [Joh67]

✓ $n$ objects, $n \cdot n$ distance matrix

✓ Algorithm:

1. Create $n$ clusters, each with 1 object.
2. Find the closest pair of clusters and join it.
3. Update the distance matrix for the newly created cluster.
4. Iterate 2 and 3, until there is only 1 cluster left.

✓ We need to specify what a distance between 2 clusters means!!!
Hierarchical clustering

Agglomerative clustering [Joh67]

✅ $n$ objects, $n \cdot n$ distance matrix

✅ Algorithm:
1. Create $n$ clusters, each with 1 object.
2. Find the closest pair of clusters and join it.
3. Update the distance matrix for the newly created cluster.
4. Iterate 2 and 3, until there is only 1 cluster left.

✅ We need to specify what a distance between 2 clusters means!!!

Divisive clustering
1. Assign all points to 1 big cluster.
2. Divide the cluster to several (2?) smaller clusters, e.g. using k-means.
3. Apply item 2 to each cluster that is still candidate for splitting.

Výsledek shlukování metodou hierarchického spojování
Summary
Learning: Needed when we do not have sufficient statistical info for recognition. Approaches:

- Assume $p_{X|K}$ has a certain form and use $T$ to estimate its parameters.
- Assume the right strategy is in a particular set and use $T$ to choose it.
- There are several learning paradigms depending on the choice of criterion used instead of Bayesian risk.
Learning: Needed when we do not have sufficient statistical info for recognition. Approaches:

- Assume $p_{X|K}$ has a certain form and use $T$ to estimate its parameters.
- Assume the right strategy is in a particular set and use $T$ to choose it.
- There are several learning paradigms depending on the choice of criterion used instead of Bayesian risk.

Unsupervised learning: What if we use the prediction from the recognition as the information from the teacher and re-estimate the recognizer iteratively?

- Failure for algorithm like Rosenblatt’s perceptron.
- Works for the problem of quadratic clustering.
- Expectation-Maximization algorithm!!!
Learning: Needed when we do not have sufficient statistical info for recognition.

Approaches:

✓ Assume $p_{XK}$ has a certain form and use $T$ to estimate its parameters.
✓ Assume the right strategy is in a particular set and use $T$ to choose it.
✓ There are several learning paradigms depending on the choice of criterion used instead of Bayesian risk.

Unsupervised learning: What if we use the prediction from the recognition as the information from the teacher and re-estimate the recognizer iteratively?

✓ Failure for algorithm like Rosenblatt’s perceptron.
✓ Works for the problem of quadratic clustering.
✓ Expectation-Maximization algorithm!!!

Unsupervised learning – a broader view: any analytical procedure that does not use the labels of the training data.

✓ Vector quantization, outlier/novelty detection, dimensionality reduction, feature extraction, …
✓ Clustering
  ✗ hierarchical vs. partitioning
