Introduction to Artificial Intelligence

Apriory Algorithm, Nearest Neighbor, Naive Bayes, Decision Tree, k-means

Ing. Tomas Borovicka

Department of Theoretical Computer Science (KTI), Faculty of Information Technology (FIT)
Czech Technical University in Prague (CVUT)

BIE-ZUM, LS 2013/14, 8. lecture

https://edux.fit.cvut.cz/courses/BIE-ZUM/

Summary of Previous Lecture

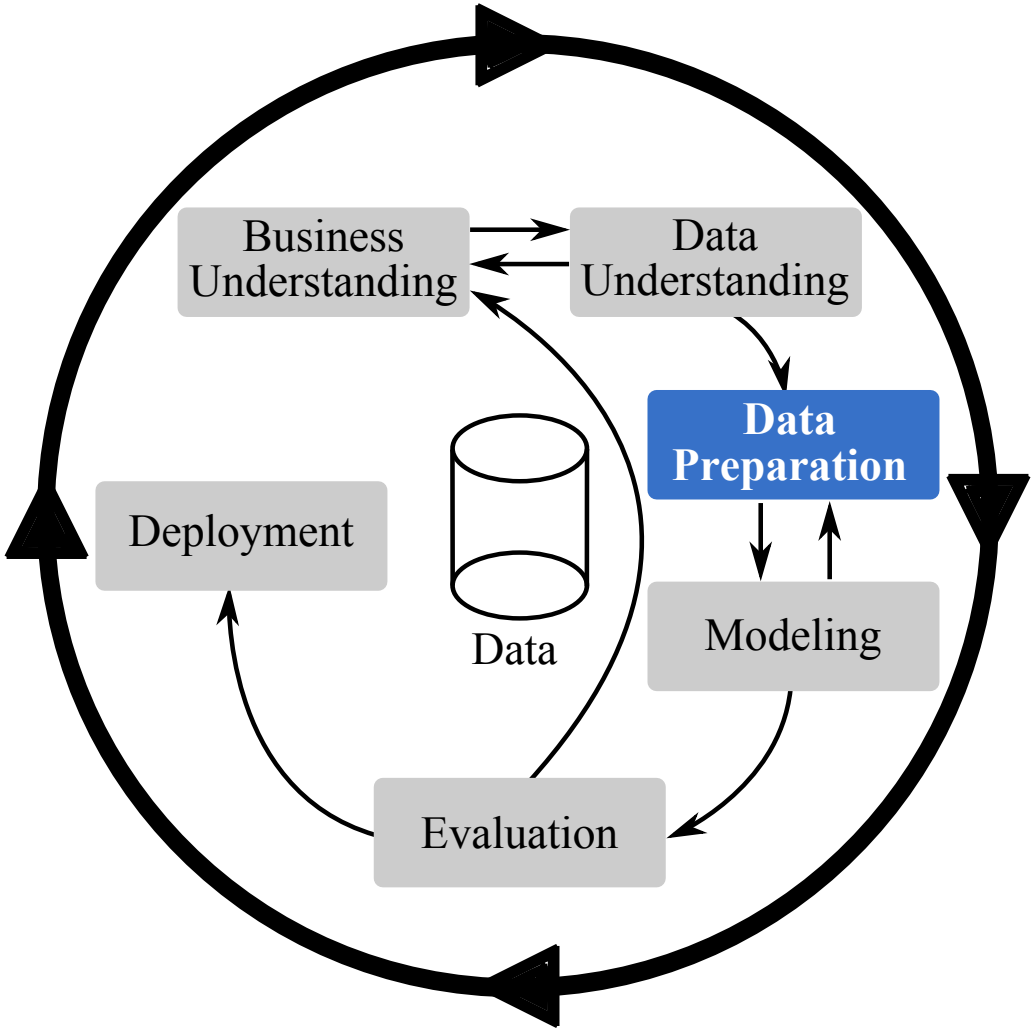


Figure : Data Mining Process by CRISP-DM

Association Rule Mining

- Unsupervised technique, learning from observations.
- Association rule mining is a method for discovering interesting relations in data.
  - Discovering regularities between products in transaction data.
- ”If – Then” relationship. If this happen, what is likely to happen next.

Generally two-step process:

1. Find all frequent itemsets.
2. Generate interesting rules from the frequent itemset.

$$\{A, B, \ldots\} \Rightarrow \{C, D, \ldots\}$$
$$\{\mathit{swimsuite}, \mathit{beachtowel}\} \Rightarrow \{\mathit{sunglasses}\}$$
$$\{\mathit{bread}, \mathit{butter}\} \Rightarrow \{\mathit{milk}\}$$

Association Rule Mining

Association rule mining problem

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of **items** and let $D = \{t_1, t_2, \ldots, t_n\}$ be a set of **transactions** called **database** where each transaction t_i is set of items such that $t_i \subseteq I$. An association rule is an implication of the form

$$A \Rightarrow B,$$

where $A, B \subset I$ and $A \cap B = \varnothing$. The set of items on the left side of the association rule (A) is called **antecedent** and the set of items on the right side (B) is called **consequent**.

Support & Confidence

Support

The support $support(A \Rightarrow B)$ is defined as the percentage of transactions in D which contain $A \cup B$ (i.e. both A and B). This is interpreted as an estimate of probability $P(A \cup B)$.

$$support(A \Rightarrow B) = P(A \cup B)$$

Confidence

The confidence $confidence(A \Rightarrow B)$ is defined as the percentage of transactions in D containing A that also contain B. This is interpreted as an estimate of the probability $P(A|B)$.

$$confidence(A \Rightarrow B) = \frac{support(A \cup B)}{support(A)}$$

Example of Rules

- Market basket transactions.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

$\{\text{Milk, Diaper}\} \Rightarrow \{\text{Beer}\}$ (s=0.4,c=0.67)
$\{\text{Milk, Beer}\} \Rightarrow \{\text{Diaper}\}$ (s=0.4,c=1.0)
$\{\text{Diaper, Beer}\} \Rightarrow \{\text{Milk}\}$ (s=0.4,c=0.67)
$\{\text{Beer}\} \Rightarrow \{\text{Milk, Diaper}\}$ (s=0.4,c=0.67)
$\{\text{Diaper}\} \Rightarrow \{\text{Milk, Beer}\}$ (s=0.4,c=0.5)
$\{\text{Milk}\} \Rightarrow \{\text{Diaper, Beer}\}$ (s=0.4,c=0.5)

Apriory Algorithm

- Mining frequent itemsets for boolean associations rules.
  1. Find the frequent itemsets.
  2. Generate strong association rules from the frequent itemsets.
- **Frequent itemsets:** the sets of items that have minimum support.
- **Apriori property:** Any subset of frequent itemset must be frequent.
  - $\forall A, B : (A \subseteq B) \Rightarrow support(A) \geq support(B)$
  - Anti-monotone property of support.
- **Strong association rules:** satisfy both minimum support and minimum confidence.

Frequent Itemsets

- Level-wise search
  - k-itemsets are used to explore (k+1)-itemsets.

1. First, the set of frequent 1-itemsets, L_1, is found.
   - All items with minimal support.
2. L_1 is used to find the set of frequent 2-itemsets, L_2.
   - Candidates are generated using join operation, $L_1 \times L_1$.
   - The set of frequent 2-itemsets is determined by minimum support.
3. L_2 is used to find the set of frequent 3-itemsets, L_3.
   - Candidates are generated using join operation, $L_2 \times L_2$.
   - The set of frequent 3-itemsets is determined by minimum support.

...

n. Until no frequent k-itemsets can be found.

Apriori Algorithm Pseudocode

Algorithm 1 Apriori Algorithm

1: C_k ... candidate itemset of size k
2: L_k ... frequent itemset of size k
3:
4: $L_1 \leftarrow \{1 - items, such\ that\ support(i) \geq \theta\}$
5: **for** $k = 1; L_k ! = \varnothing; k++$ **do**
6: $\quad C_{k+1} \leftarrow \{candidates\ generated\ from\ L_k\}$
7: $\quad L_k \leftarrow \{c : c \in C_k, support(c) \geq \theta\}$
8: **end for**
return $\bigcup_k L_k$

Strong Association Rules

- For each frequent itemset l are generated all nonempty subsets $s \subset l$.
- For each nonempty set $s \subset l$ that satisfies

$$\frac{support(l)}{support(s)} \geq \theta_{conf_{min}},$$

where $\theta_{conf_{min}}$ is minimum confidence treshold, is generated strong association rule

$$s \Rightarrow (l \setminus s).$$

Nearest Neighbor Classification

- The Nearest Neighbor is one of the simplest and oldest commonly used classification methods.
- Based on learning by similarity, i.e. on comparing given unknown instance with the instances in the training set that are similar to it.
- Holds all training instances in the memory.
- Every time it has to compare whole training set with a given unknown instance.
- Inefficient with large datasets.

1-NN

- For given unknown instance x' finds the most similar instance x^* from the training set R and classifies x' into the the class of x^*

$$x^* = \arg\min_{x \in R} ||x - x'||$$

using the Euclidean distance

$$x^* = \arg\min_{x \in R} \sqrt{\sum_j (x_j - x'_j)^2}.$$

- Unknown instance x' is classified into the same class as x^*, let y^* be the corresponding label then $y' = y^*$.

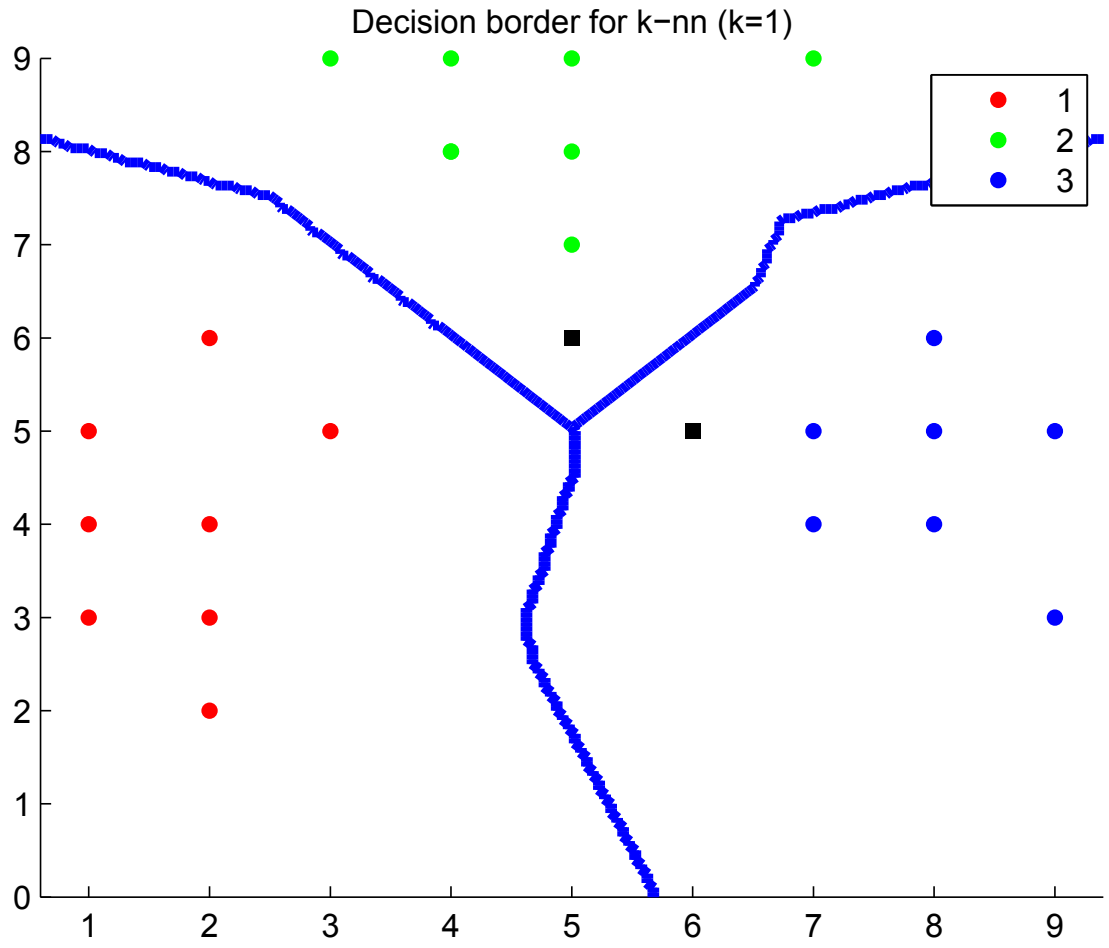
k-Nearest Neighbors

- instead of searching one most similar instance, k instances (nearest neighbors) are found
- The unknown instance is classified into the majority class of k nearest neighbors.

Let X^* be set of k nearest neighbors with corresponding classes Y^*, the unknown instance is classified into the class

$$y* = \arg\max_{y_i \in Y^*} \left| \{ Y^* = y_i \} \right| \quad (1)$$

Decision Border



Bayesian Classification

- Probabilistic methods based on Bayes theorem.
- Assumes that hypothesis that approximates target function well over observed examples will also approximate well over unobserved examples.

Bayes theorem

Let y be a hypothesis, such as that an example $\mathbf{x}$ belongs to a specific class y, Bayes theorem is

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) \cdot P(y)}{P(\mathbf{x})},$$

where $P(y)$ and $P(\mathbf{x})$ are the **prior probabilities**,
$P(\mathbf{x}|y)$ is the **posterior probability** of example x conditioned by class y and
$P(y|\mathbf{x})$ is the posterior probability of class y conditioned by example $\mathbf{x}$.

Maximum a posteriori (MAP)

- The learner considers some set of candidate hypothesis, such as that x belongs to all possible $y \in \mathcal{Y}$, and chooses the most probable hypothesis.
- The most probable hypothesis given observed data is called **Maximum a posteriori (MAP)**.

Maximum a posteriori

$$y_{MAP} = \arg\max_{y \in \mathcal{Y}} P(y|\mathbf{x})$$

$$y_{MAP} = \arg\max_{y \in \mathcal{Y}} \frac{P(\mathbf{x}|y) \cdot P(y)}{P(\mathbf{x})}$$

$$y_{MAP} = \arg\max_{y \in \mathcal{Y}} P(\mathbf{x}|y) \cdot P(y)$$

Naive Bayes Classifier

- The **naive** assumption of conditional independence of attributes.

Conditional independence assumption

$$P(\mathbf{x}|y) = P(x_1, x_2, \ldots, x_n|y) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|Y)$$
$$= \Pi_{i=1}^{n} P(x_i|y)$$

- The maximum a posteriori for naive Bayes is then

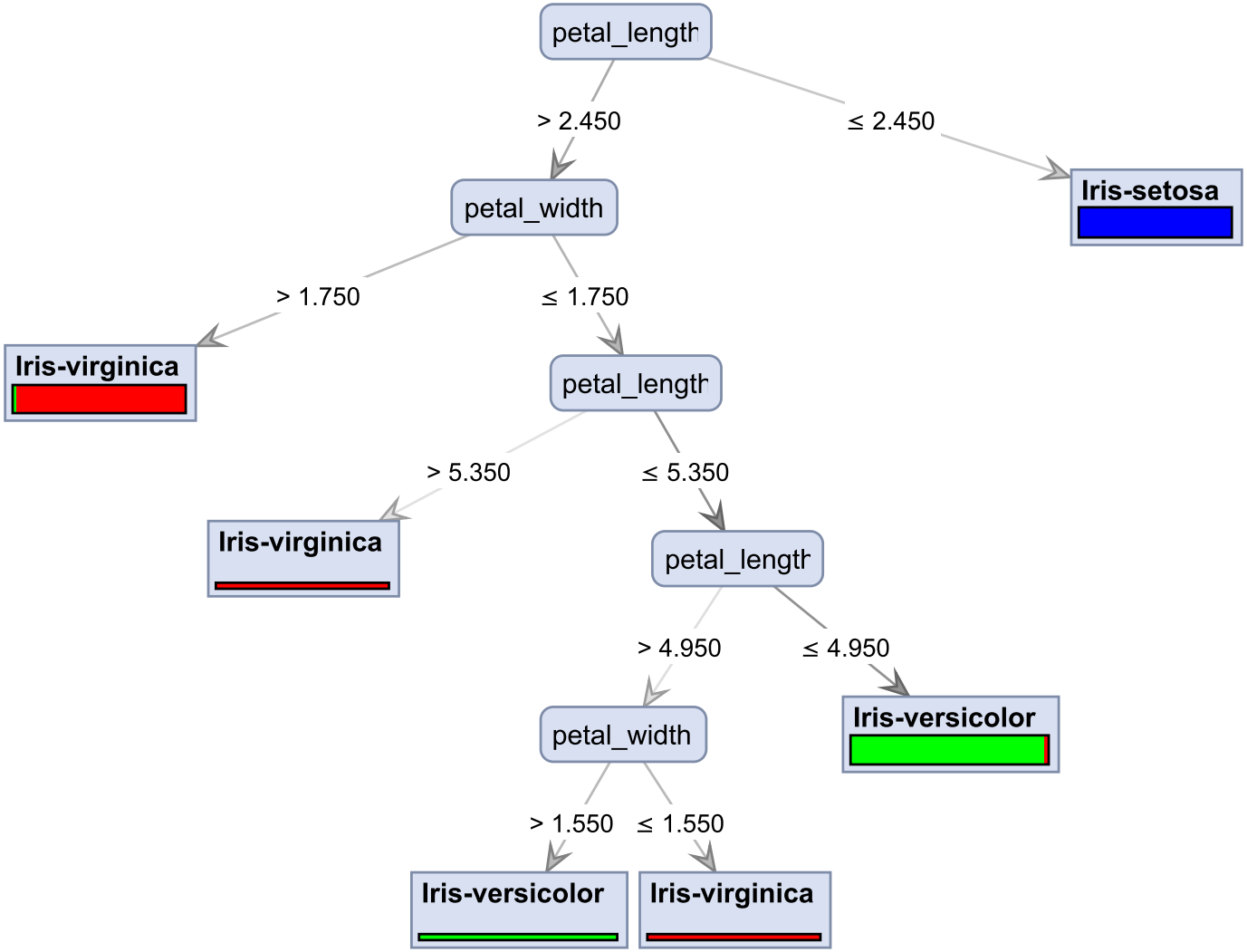
$$y_{MAP} = P(y) \cdot \Pi_{i=1}^{n} P(x_i|y)$$

- The assumption is in practice almost always violated, however, it may still find maximum probability hypothesis!
- Experiments have shown that Naive Bayes is competitive with other methods.

Decision Tree

- Very popular across various domains.
  - Simple and interpretable.
- Decision tree is represented by a nodes in the rooted tree structure (usually binary).
  - **Nodes**: attributes
  - **Edges**: attribute values
  - **Leaves**: classes
- Classification of an unknown example is performed by successive testing in internal nodes and corresponding routing towards the most appropriate leaf which contains the probability vector indicating the class.

Example: Iris Classification



Learning Decision Tree

- Finding minimal optimal decision tree is **NP-hard** task.
  - Heuristics methods are generally used for learning (greedy search).
- Decision tree is induced on a recursive partitioning of the input space.
- In each decision node we determine an attribute X_j with partitioning Φ_j that maximize the splitting criterion Ψ, formally

$$\arg\max_{j,k} \Psi(X_j, \Phi_j)$$

- **Splitting Criteria**
  - Information Gain

$$IG(X_j) = \Delta H(X_j) = H(Y) - H(Y|\phi_j)$$

  - Gain Ratio

$$GR(X_j) = \frac{H(Y) - H(Y|\phi_j)}{H(X_j)}$$

  - Gini Index

$$\Delta G(X_j) = G(Y) - G(Y|\phi_j)$$

Stopping Criteria and Pruning

Stopping Criteria

- Terminates learning of the three, when further growing would probably does not increase the decision tree performance.
  - All instances in the training set are from the same class.
  - Depth of the tree reached defined maximal limit.
  - The number of cases in the node is less than the minimum limit for inner nodes.
  - If the node were be split the number of cases in one or more terminal nodes would be less than the minimum limit.
  - The best splitting criteria is less than defined minimum threshold.

Pruning

- Pruning is technique for cutting over-fitted trees, it uses various measures to remove the least reliable branches to make simpler and more accurate decision tree.
  - Pre-Pruning
  - Post-Pruning

k-means

- One of the most popular clustering algorithms.
- The number of clusters is parameter of the algorithm.
- Uses representative point for each cluster (centroid).

k means

Let $X = \{x^1, x^2, \ldots, x^n\}$ be a set of n points in m dimensions, $X \in \mathcal{R}^m$. k-means arranges these points into k clusters by minimizing the total energy

$$E = \sum_{l=1}^{k} \sum_{x^i \in c_l} \|x^i - \mu_l\|^2,$$

where μ_l is a centroid of the points in the cluster c_l defined as

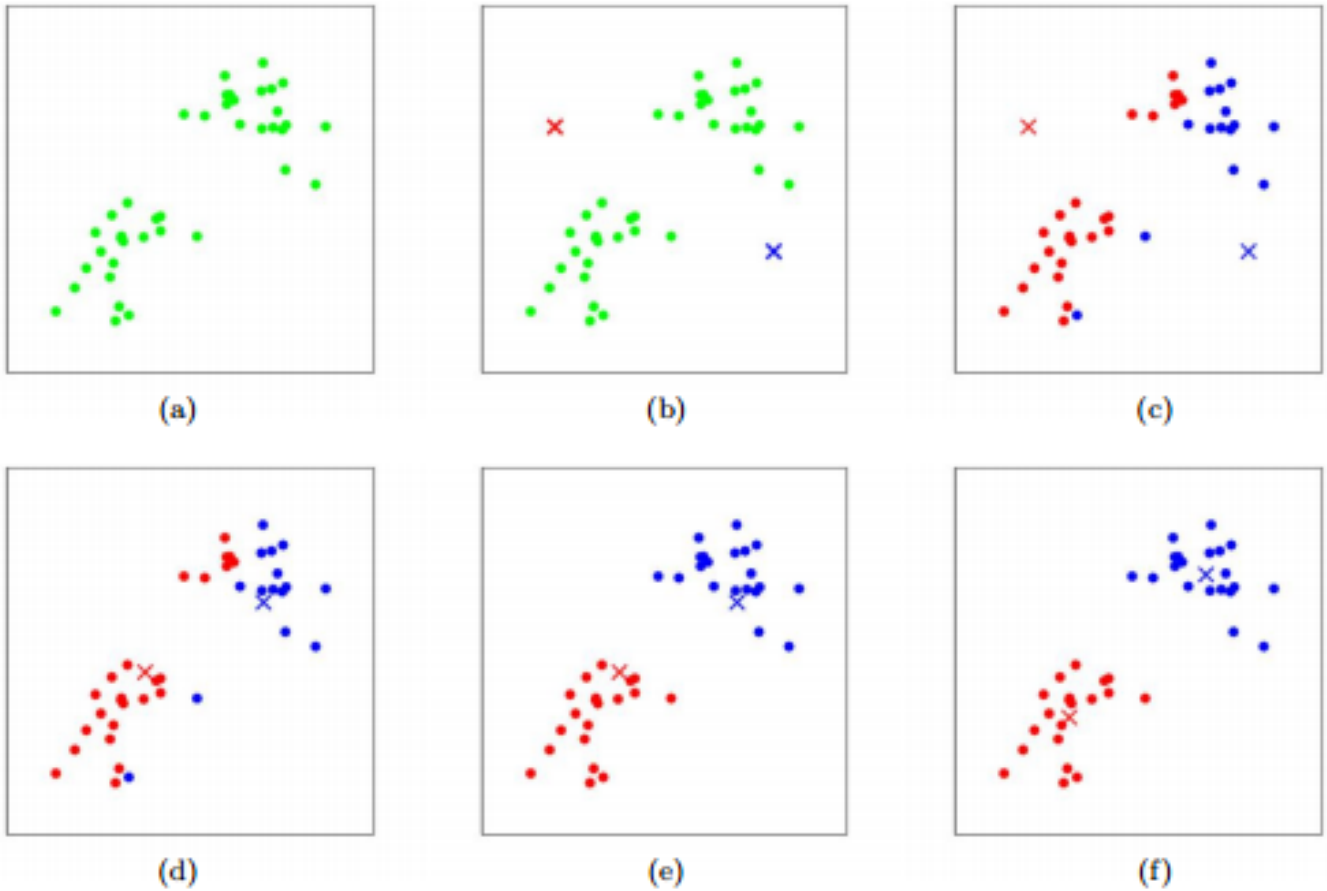
$$\mu_l = \frac{1}{|c_l|} \sum_{x^i \in c_l} x^i.$$

k-means Pseudocode

Algorithm 2 k-means Algorithm

1: Randomly initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$
2: **repeat**
3: **for all** $x^i \in X$ **do**
4: $c^i \leftarrow c_l : \arg\min_l \left\| x^i - \mu_l \right\|^2$
5: **end for**
6: **for** $c_l \in C$ **do**
7: $\mu_l = \frac{1}{|c_l|} \sum_{x^i \in c_l} x^i$
8: **end for**
9: **until** convergence

k-means Illustration



Source: Stanford University, CS221, http://stanford.edu/∼cpiech/cs221/