

Classification: Advanced Methods

Prepared by Dr. Amjad Qtaish

DS601
Data Mining

Lecture 08



Lecture: #08

Classification: Advanced Methods



Dr. Amjad Qtaish

Faculty of Informatics
Midocean University

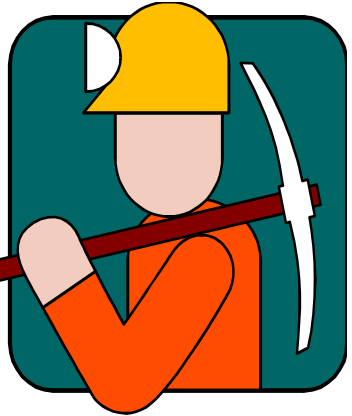




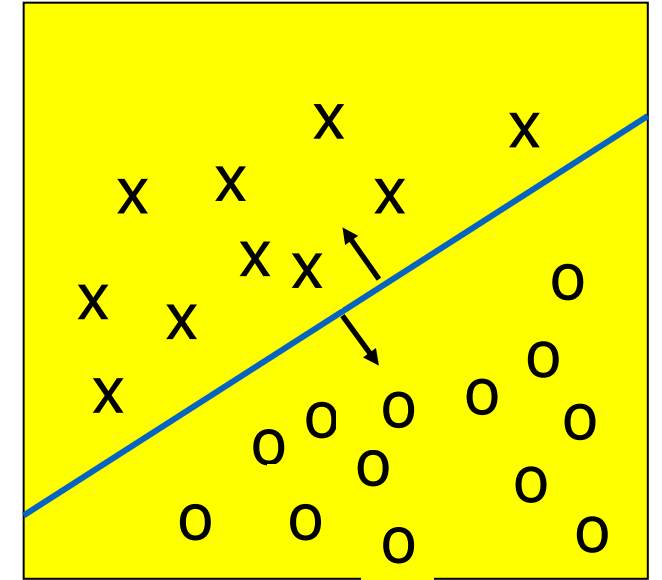
Classification: Advanced Methods

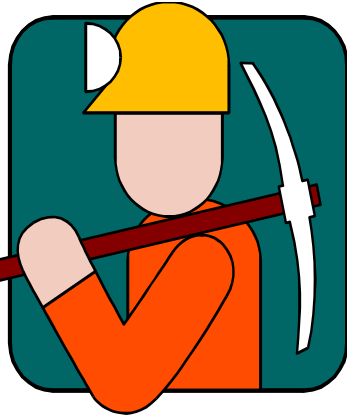
- Classification: A Mathematical Mapping 
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary





- **Classification:** predicts categorical class labels
 - E.g., Personal homepage classification
 - $x_i = (x_1, x_2, x_3, \dots)$, $y_i = +1$ or -1
 - x_1 : # of word "homepage"
 - x_2 : # of word "welcome"
- Mathematically, $x \in X = \mathbb{R}^n$, $y \in Y = \{+1, -1\}$,
 - We want to derive a function $f: X \rightarrow Y$
- Linear Classification
 - Binary Classification problem
 - Data above the red line belongs to class 'x'
 - Data below red line belongs to class 'o'
 - Examples: SVM, Perceptron, Probabilistic Classifiers





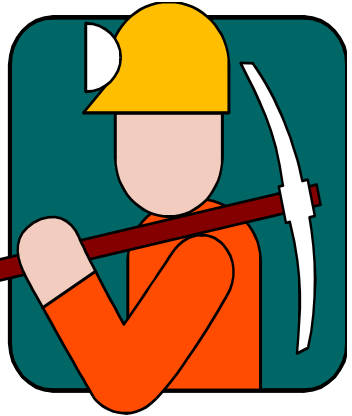
- Advantages
 - Prediction accuracy is generally high
 - As compared to Bayesian methods – in general
 - Robust, works when training examples contain errors
 - Fast evaluation of the learned target function
 - Bayesian networks are normally slow
- Criticism
 - Long training time
 - Difficult to understand the learned function (weights)
 - Bayesian networks can be used easily for pattern discovery
 - Not easy to incorporate domain knowledge
 - Easy in the form of priors on the data or distributions



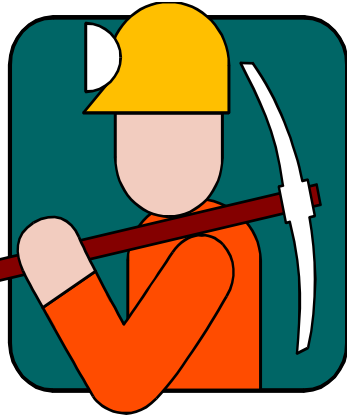
Classification: Advanced Methods

- Classification: A Mathematical Mapping
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary

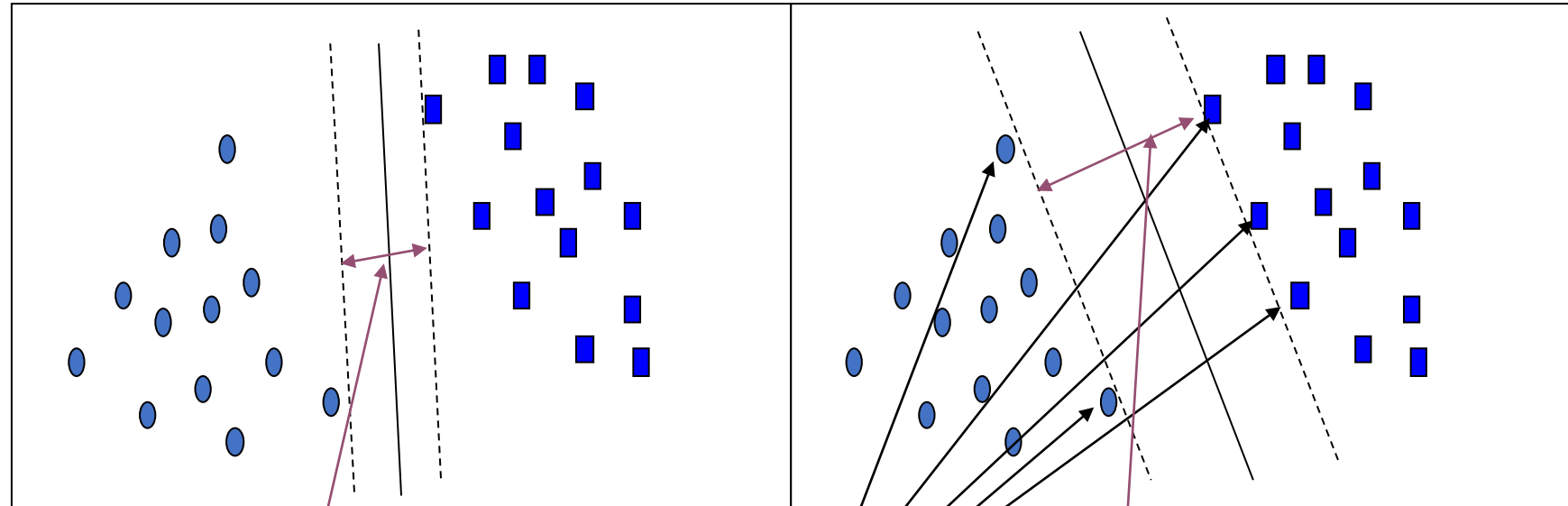
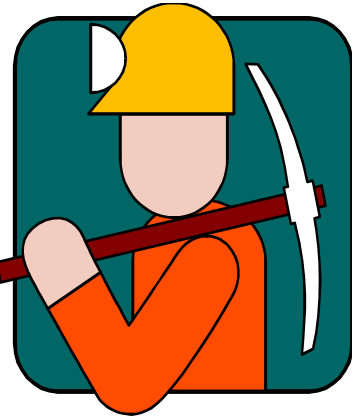




- A relatively new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., “decision boundary”)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** (“essential” training tuples) and **margins** (defined by the support vectors)



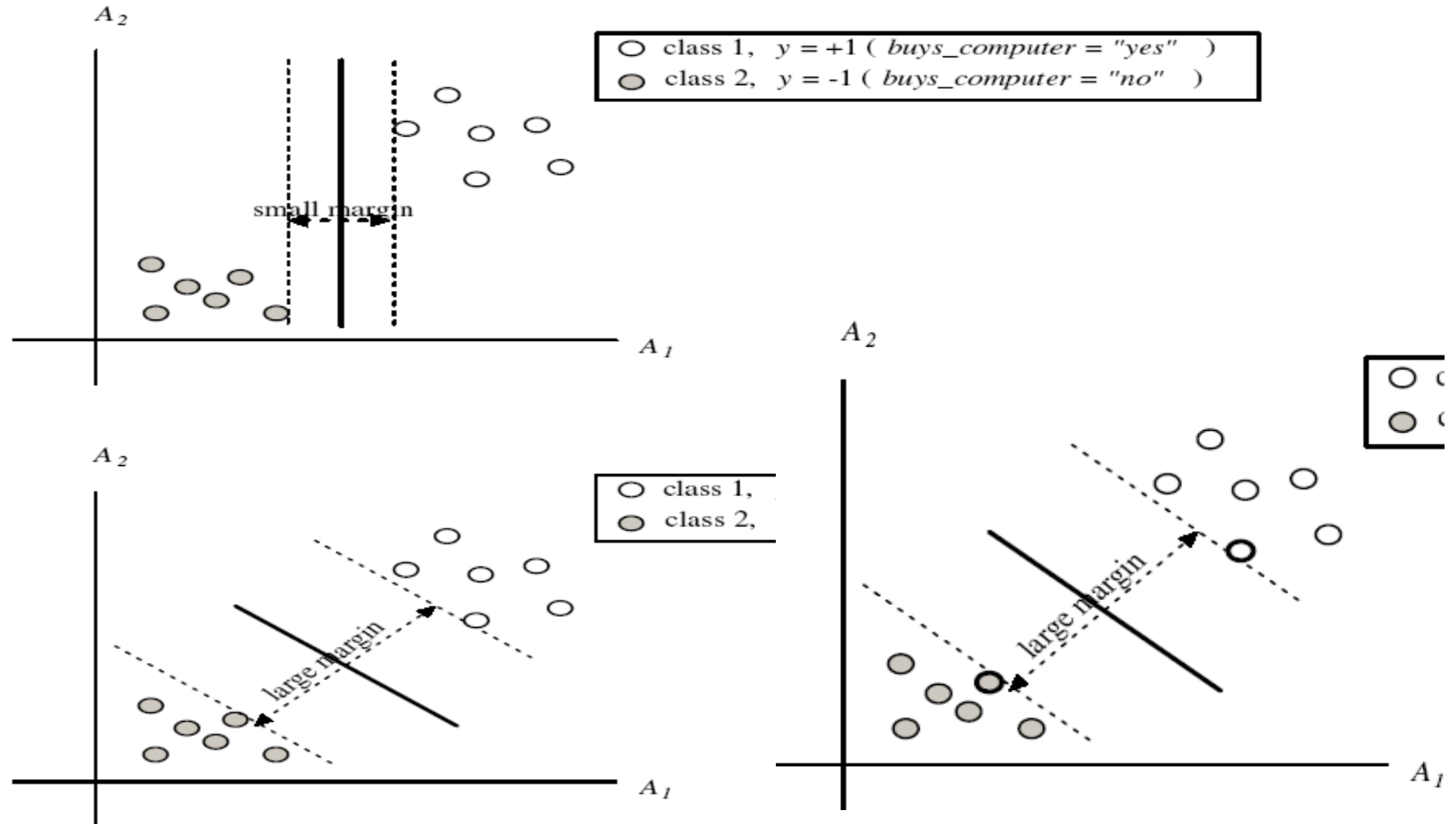
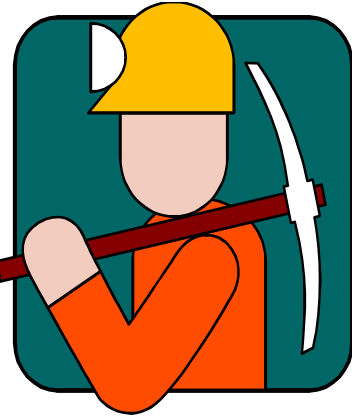
- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- **Features**: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- **Used for**: classification and numeric prediction
- **Applications**:
 - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

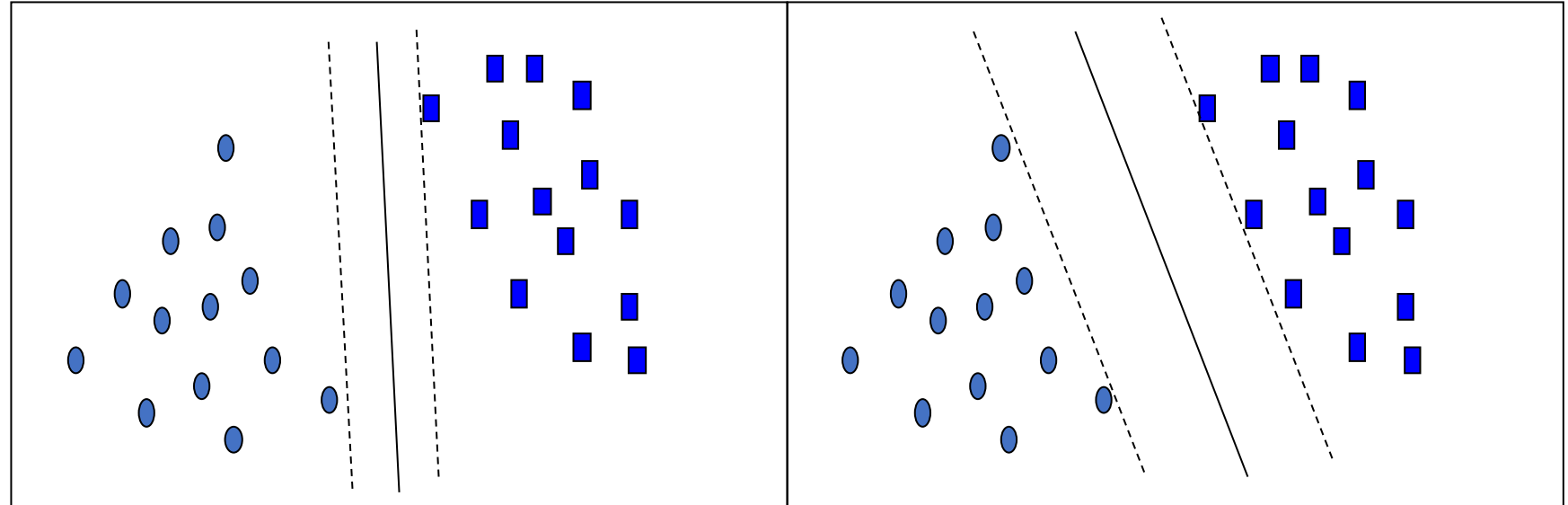
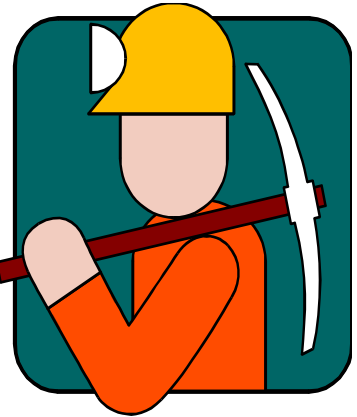


Small Margin

Large Margin

Support Vectors

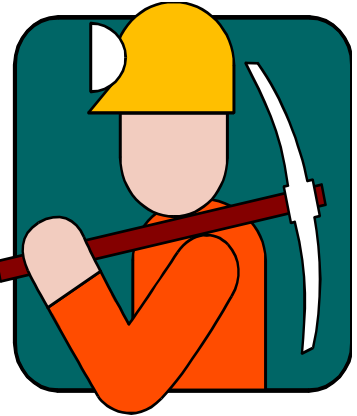




Let data D be $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

*SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)*



- A separating hyperplane can be written as

$$\mathbf{W} \bullet \mathbf{X} + b = 0$$

where $\mathbf{W} = \{w_1, w_2, \dots, w_n\}$ is a weight vector and b a scalar (bias)

- For 2-D it can be written as

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- The hyperplane defining the sides of the margin:

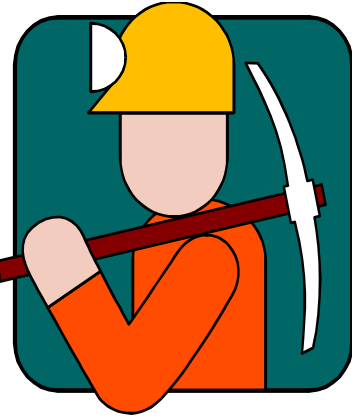
$$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1, \text{ and}$$

$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1$$

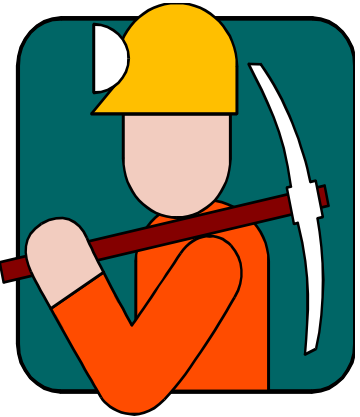
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem:
Quadratic objective function and linear constraints $\rightarrow$ *Quadratic Programming (QP)* $\rightarrow$ Lagrangian multipliers



Why Is SVM Effective on High Dimensional Data?



- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

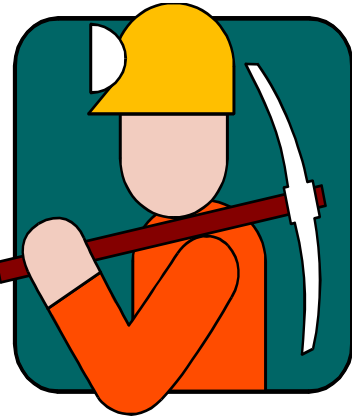


- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(\mathbf{Z}) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned} \quad \blacksquare$$

- Search for a linear separating hyperplane in the new space



- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

Polynomial kernel of degree h : $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$

Gaussian radial basis function kernel : $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$

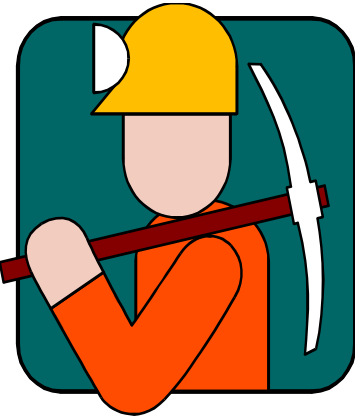
Sigmoid kernel : $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)





- **SVM**



- Deterministic algorithm
- Nice generalization properties
- Hard to learn – learned in batch mode using quadratic programming techniques
- Using kernels can learn very complex functions

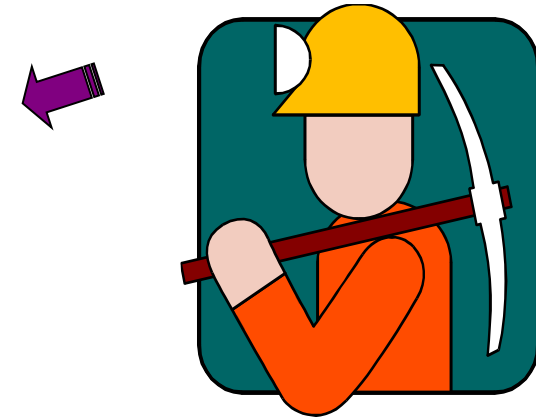
- **Neural Network**

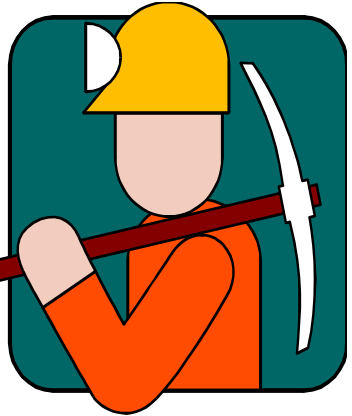
- Nondeterministic algorithm
- Generalizes well but doesn't have strong mathematical foundation
- Can easily be learned in incremental fashion
- To learn complex functions—use multilayer perceptron (nontrivial)



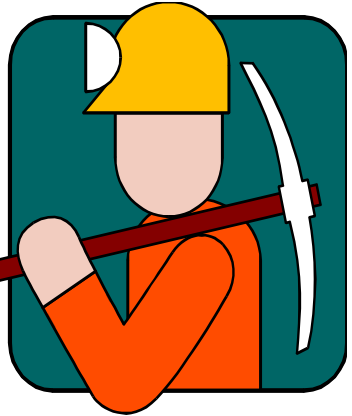
Classification: Advanced Methods

- Classification: A Mathematical Mapping
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary





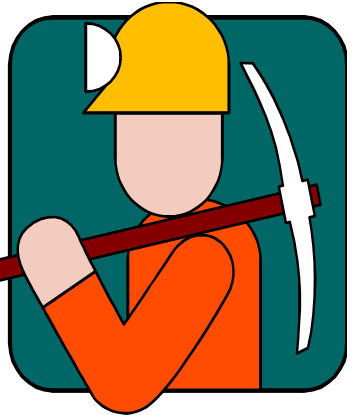
- Lazy vs. eager learning
 - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
 - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
 - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
 - Eager: must commit to a single hypothesis that covers the entire instance space



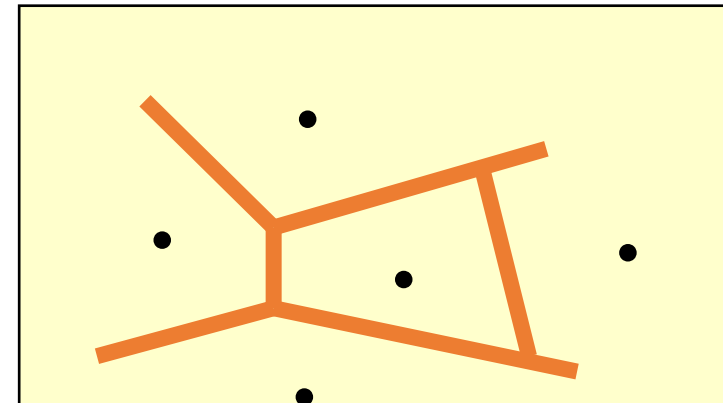
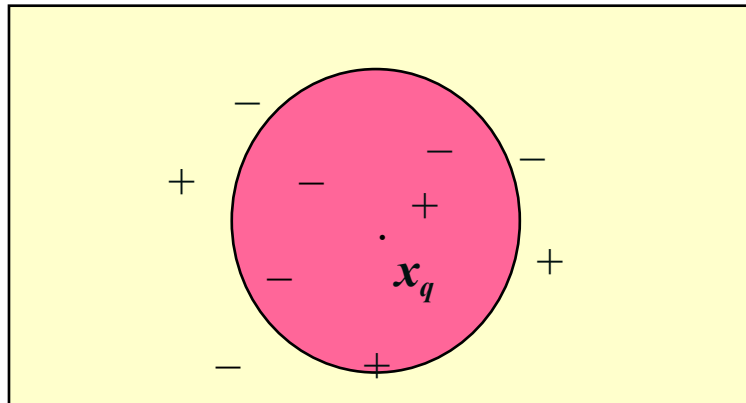
- Instance-based learning:
 - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
 - k-nearest neighbor approach
 - Instances represented as points in a Euclidean space.
 - Locally weighted regression
 - Constructs local approximation
 - Case-based reasoning
 - Uses symbolic representations and knowledge-based inference

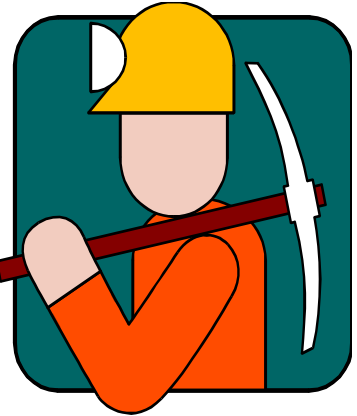


The k-Nearest Neighbor Algorithm



- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, $\text{dist}(\mathbf{X}_1, \mathbf{X}_2)$
- Target function could be discrete- or real- valued
- For discrete-valued, k -NN returns the most common value among the k training examples nearest to x_q
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples





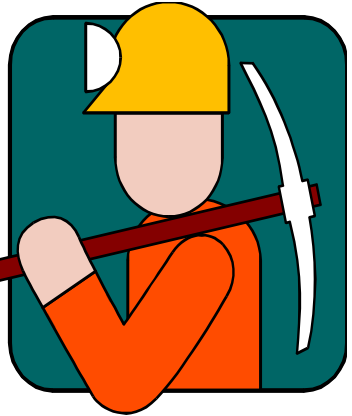
- **k-NN** for real-valued prediction for a given unknown tuple
 - Returns the mean values of the k nearest neighbors
- Distance-weighted nearest neighbor algorithm
 - Weight the contribution of each of the k neighbors according to their distance to the query x_q
 - Give greater weight to closer neighbors
- Robust to noisy data by averaging k -nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
 - To overcome it, axes stretch or elimination of the least relevant attributes



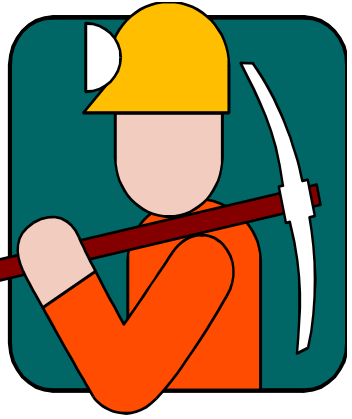
Classification: Advanced Methods

- Classification: A Mathematical Mapping
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary

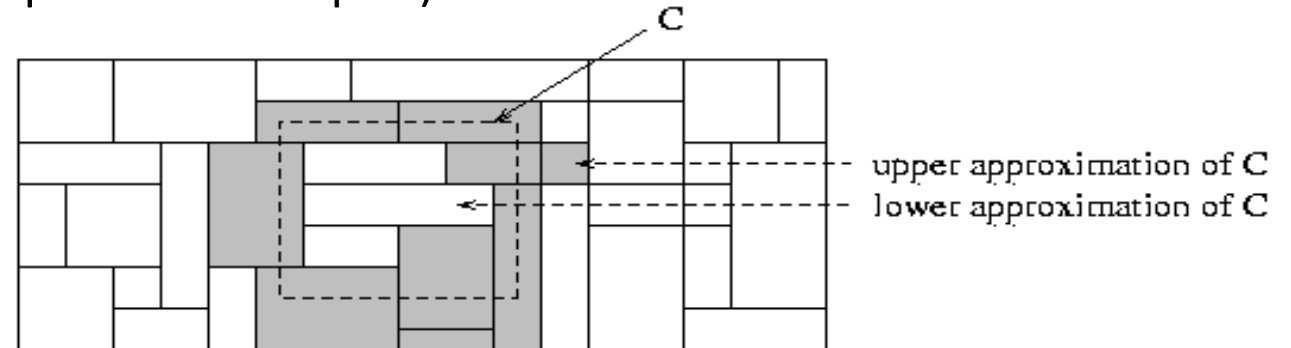


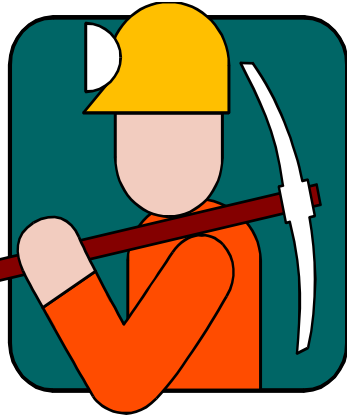


- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
 - Each rule is represented by a string of bits
 - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
 - If an attribute has $k > 2$ values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The *fitness of a rule* is represented by its classification accuracy on a set of training examples
- Offspring are generated by *crossover* and *mutation*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

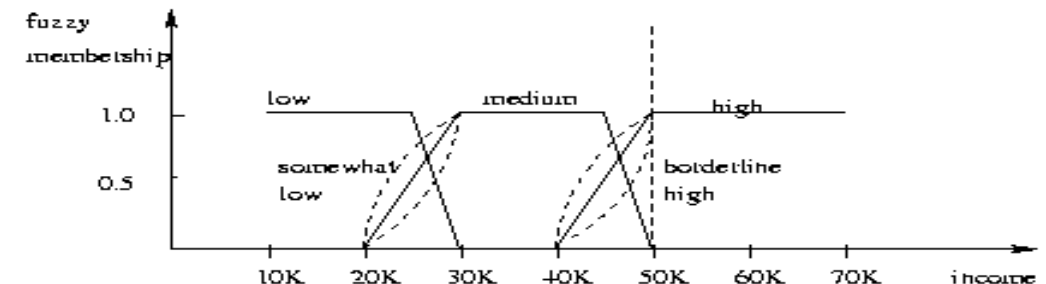


- Rough sets are used to **approximately** or “**roughly**” define **equivalent classes**
- A rough set for a given class C is approximated by two sets: a **lower approximation** (certain to be in C) and an **upper approximation** (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity





- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a *fuzzy membership graph*)
- Attribute values are converted to fuzzy values. Ex.:
 - Income, x , is assigned a **fuzzy membership value** to each of the discrete categories {low, medium, high}, e.g. \$49K belongs to “medium income” with fuzzy value 0.15 but belongs to “high income” with fuzzy value 0.96
 - Fuzzy membership values do not have to sum to 1.
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined

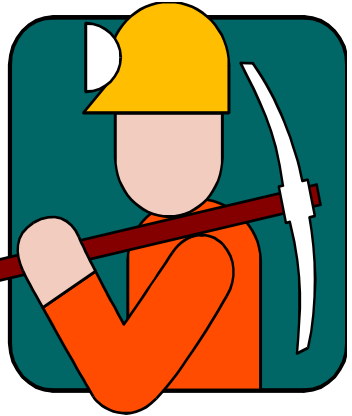




Classification: Advanced Methods

- Classification: A Mathematical Mapping
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary





- Classification involving more than two classes (i.e., > 2 Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
 - Given m classes, train m classifiers: one for each class
 - Classifier j : treat tuples in class j as *positive* & all others as *negative*
 - To classify a tuple $\mathbf{X}$, the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
 - Given m classes, construct $m(m-1)/2$ binary classifiers
 - A classifier is trained using tuples of the two classes
 - To classify a tuple $\mathbf{X}$, each classifier votes. $\mathbf{X}$ is assigned to the class with maximal vote
- Comparison
 - All-vs.-all tends to be superior to one-vs.-all
 - Problem: Binary classifier is sensitive to errors, and errors affect vote count

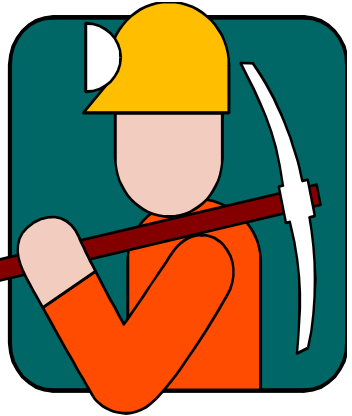


Error-Correcting Codes for Multiclass Classification

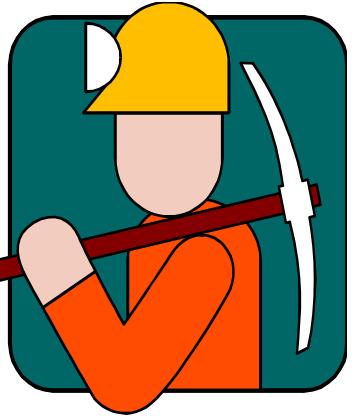


- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
- Example
 - A 7-bit codeword associated with classes 1-4
 - Given a unknown tuple $\mathbf{X}$, the 7-trained classifiers output: 0001010
 - Hamming distance: # of different bits between two codewords
 - $H(\mathbf{X}, C_1) = 5$, by checking # of bits between [1111111] & [0001010]
 - $H(\mathbf{X}, C_2) = 3$, $H(\mathbf{X}, C_3) = 3$, $H(\mathbf{X}, C_4) = 1$, thus C_4 as the label for $\mathbf{X}$
 - Error-correcting codes can correct up to $(h-1)/h$ 1-bit error, where h is the minimum Hamming distance between any two codewords
 - If we use 1-bit per class, it is equiv. to one-vs.-all approach, the code are insufficient to self-correct
 - When selecting error-correcting codes, there should be good row-wise and col.-wise separation between the codewords

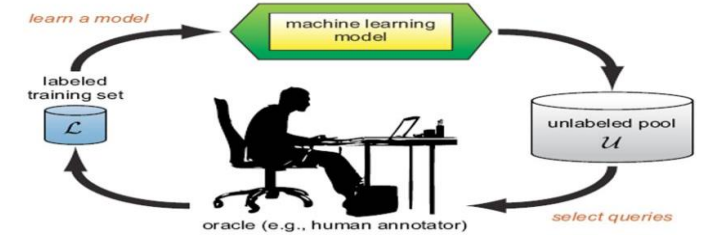
Class	Error-Corr. Codeword						
C_1	1	1	1	1	1	1	1
C_2	0	0	0	0	1	1	1
C_3	0	0	1	1	0	0	1
C_4	0	1	0	1	0	1	0



- **Semi-supervised:** Uses labeled and unlabeled data to build a classifier
- Self-training:
 - Build a classifier using the labeled data
 - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
 - Repeat the above process
 - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
 - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f_1
 - Then f_1 and f_2 are used to predict the class label for unlabeled data X
 - Teach each other: The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2 , & vice versa
- Other methods, e.g., joint probability distribution of features and labels

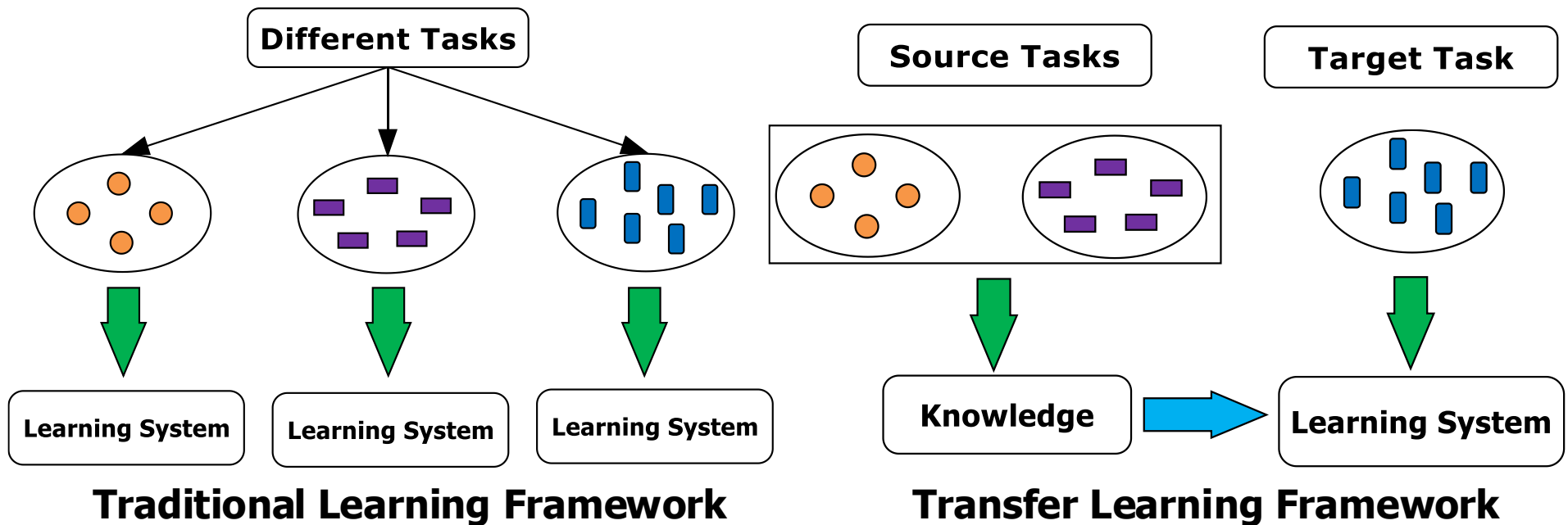
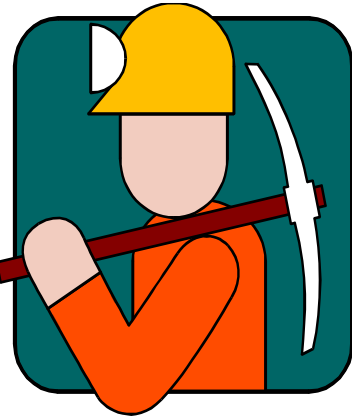


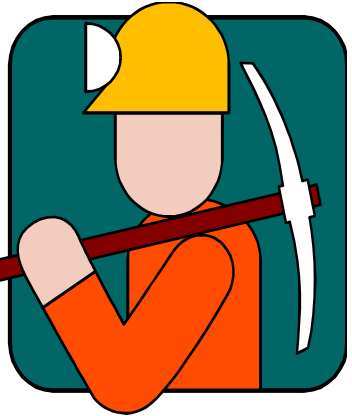
- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
 - L: a small subset of D is labeled, U: a pool of unlabeled data in D
 - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
 - The newly labeled samples are added to L, and learn a model
 - Goal: Achieve high accuracy using as few labeled data as possible
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
 - Uncertainty sampling: choose the least certain ones
 - Reduce *version space*, the subset of hypotheses consistent w. the training data
 - Reduce expected entropy over U: Find the greatest reduction in the total number of incorrect predictions





- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks





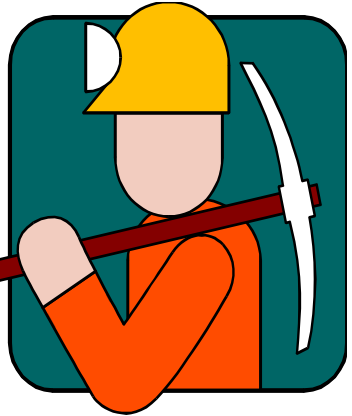
- **Applications:** Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning:* Reweight some of the data from source tasks and use it to learn the target task
- **TrAdaBoost (Transfer AdaBoost)**
 - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
 - Require only labeling a small amount of target data
 - Use source data in training: When a source tuple is misclassified, reduce the weight of such tuples so that they will have less effect on the subsequent classifier
- **Research issues**
 - Negative transfer: When it performs worse than no transfer at all
 - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
 - Large-scale transfer learning



Classification: Advanced Methods

- Classification: A Mathematical Mapping
- Support Vector Machines
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary 





- **Effective and advanced classification methods**

- Support Vector Machine (SVM)
- Other classification methods: lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches

- **Additional Topics on Classification**

- Multiclass classification
- Semi-supervised classification
- Active learning
- Transfer learning



Thank You

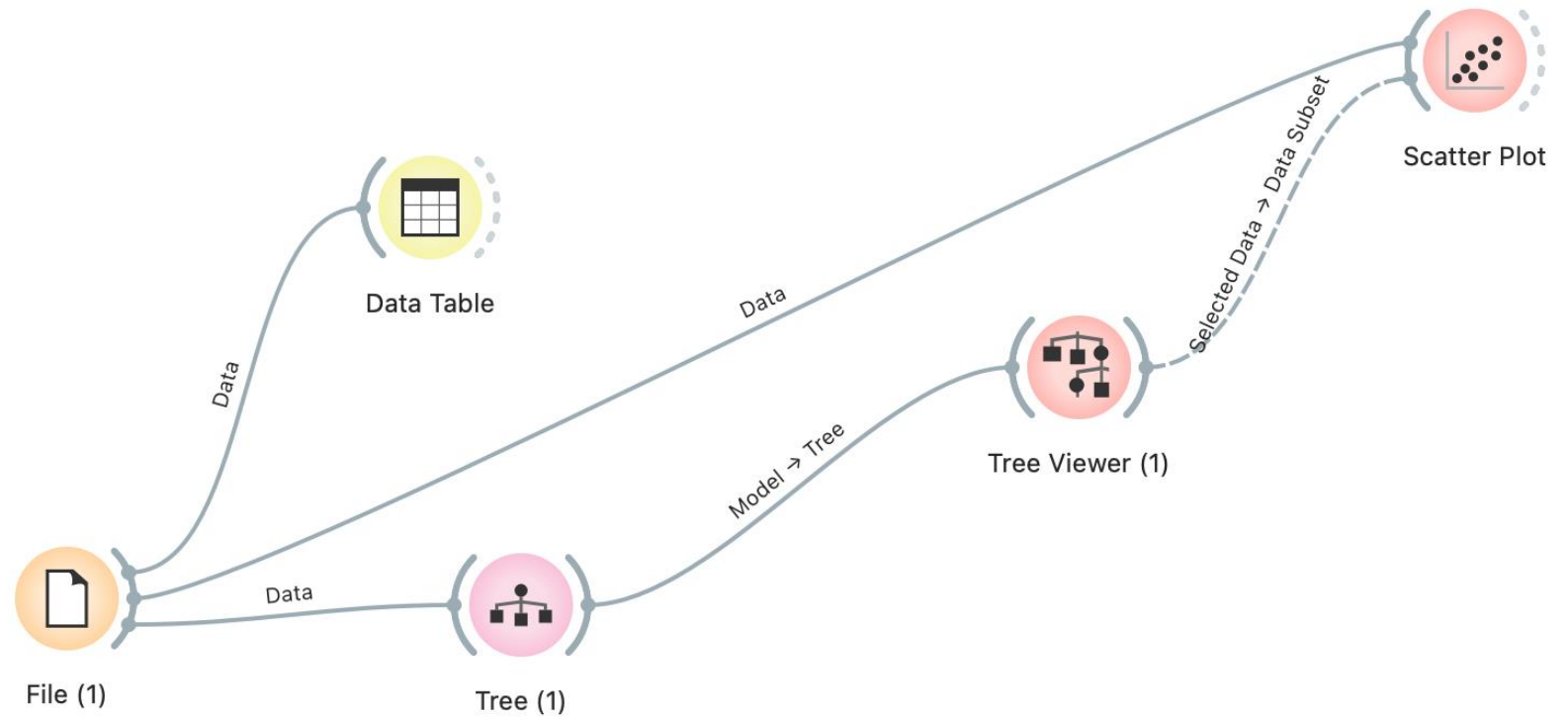
Classification



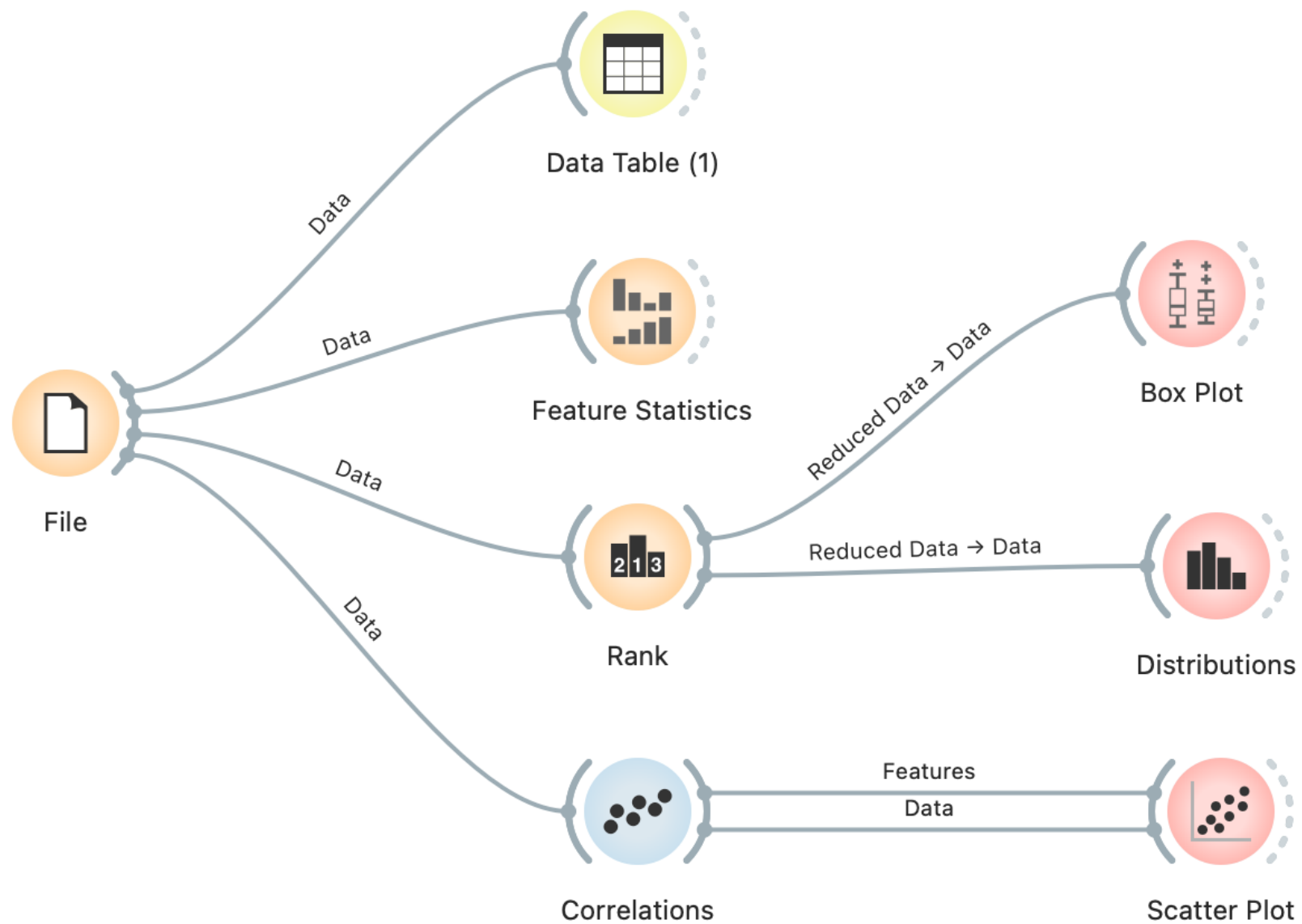
Orange Data Mining

Data Mining Software

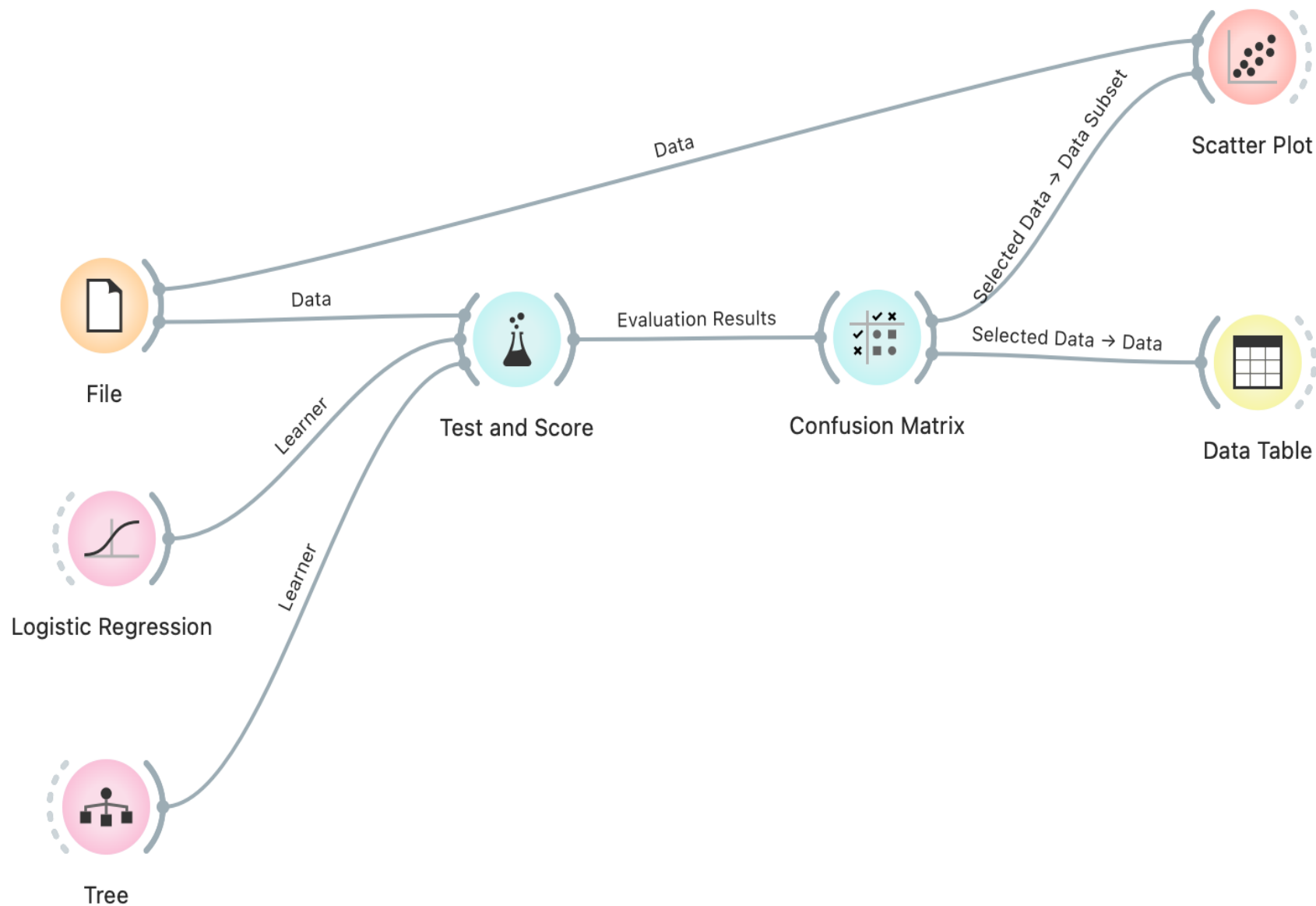
Iris Classification – Example 1



Iris Classification – Example 2



Iris Classification – Example 3



Iris Classification – Example 4

