### Definitions
- Generate knowledge from experience
- Learn from data
- Detect regularities
- Make generalizations

### Related areas
- Statistics
- Data mining
- Pattern recognition
- Artificial intelligence
- Optimization
- Complexity theory
- Information theory
- Signal theory

---

**Knowledge**

**Information**

**Data**
Supervised learning solves regression tasks

input / output \((x, y) \in \mathbb{R}^{p+q}\)

input \(x \in \mathbb{R}^p\) \rightarrow \text{regression model} \rightarrow \text{output } y \in \mathbb{R}^q

- **linear regression**
- **robust regression**
- **polynomial regression**
Universal approximators realize regression with arbitrarily small error

**Definition**

Given \( f : U \rightarrow \mathbb{R} \), with compact subset \( U \subset \mathbb{R}^n \).

Class \( F \) is universal approximator iff

\[
\forall \varepsilon > 0 \ \exists f^* \in F \ \forall x \in U \ |f(x) - f^*(x)| < \varepsilon
\]

**Takagi Sugeno system**

**perceptron**

**RBF network**
Forecasting is transformed to regression by finite unfolding in time.
Supervised learning solves classification tasks

input / output \( (x, y) \in \mathbb{R}^p \times \{1, \ldots, c\} \)

input \( x \in \mathbb{R}^p \) (features)

classification model

output \( y \in \{1, \ldots, c\} \) (class)

- linear discriminance
- nearest neighbor
- support vector machine

![Graphical representation of classification models: linear discriminance, nearest neighbor, support vector machine.]

Unrestricted © Siemens AG 2013. All rights reserved
Unsupervised learning solves clustering tasks

input (features) \( x \in \mathbb{R}^p \)

\[ y \in \{1, \ldots, c\} \quad \text{(cluster)} \]

single linkage  

\[ \begin{array}{c}
\text{c-means} \\
\text{fuzzy c-means}
\end{array} \]

\[ \begin{array}{c}
\text{output}
\end{array} \]

\[ \begin{array}{c}
\text{cluster model}
\end{array} \]
Reinforcement learning solves control tasks

\[
\text{agent} \quad \max \sum_{k=0}^{\infty} \gamma^k r_{t+k}
\]

- **state** $s_t$
- **reward** $r_t$
- **process**
- **action** $a_t$
Master’s Thesis has demonstrated a first approach to machine learning in program analysis

**Goals**

- Demonstrate application of machine learning to program analysis (within a Master’s thesis, 2011/12)
- Overall goal: extract control flow graphs from assembler code
- First step: identify procedure begins and ends (push and pull stack frames)
- First approach: train SVM classifier with labeled code repository
Suitable features are found by training and test

**Workflow**

- Selection of training data with procedure border information and test data without
- Definition and generation of suitable features
- Do
  - Train and evaluate classifier on test data
- Modify features
- Until satisfied with classification accuracy

---

**Phase 1: Training**

1. Frame detection
   - training file: known procedure borders
   - disassembly files: no procedure borders given

2. Feature Vector generation
   - Class 0 (red) = no stack frame
   - Class 1 (blue) = stack frame

---

**Phase 2: Classification**

3a Find Hyperplane

3b Classify Feature Vectors
A first automatic and sound analysis framework for executables was realized

**Experiments**

- Analysis of PowerPC assembler code
- 32 features: numbers and distances of read and write operations to registers and the return register
- class: procedure header (y/n)
- WEKA SVM with feature selection

**Results**

- run time: 2 min – 5 hours
- classification rate >> 90%