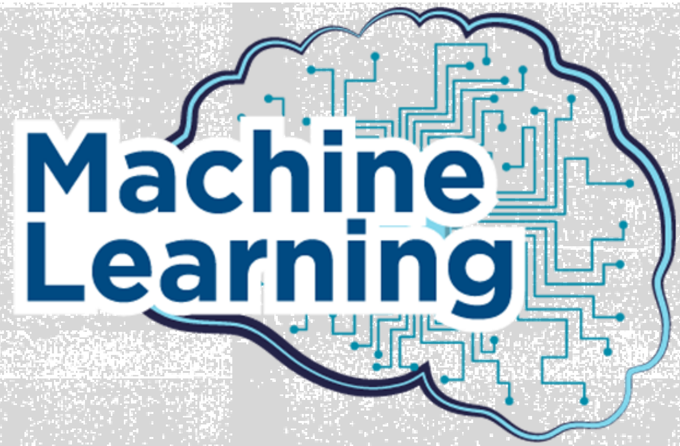
Introduction To



MACHINE LEARNING

Complete Guide

GENIAL CODE

www.genial-code.com

SOURCES

- AAAI. Machine Learning. *http://www.aaai.org/Pathfinder/html/machine.html*
- Dietterich, T. (2003). Machine Learning. *Nature Encyclopedia of Cognitive Science.*
- Doyle, P. Machine Learning. *http://www.cs.dartmouth.edu/~brd/Teaching/AI/Lectures/Summaries/learning.html*
- Dyer, C. (2004). Machine Learning. *http://www.cs.wisc.edu/~dyer/cs540/notes/learning.html*
- Mitchell, T. (1997). *Machine Learning*.
- Nilsson, N. (2004). Introduction to Machine Learning. *http://robotics.stanford.edu/people/nilsson/mlbook.html*
- Russell, S. (1997). Machine Learning. *Handbook of Perception and Cognition*, Vol. 14, Chap. 4.
- Russell, S. (2002). *Artificial Intelligence: A Modern Approach*, Chap. 18-20. *http://aima.cs.berkeley.edu*

WHAT IS LEARNING?

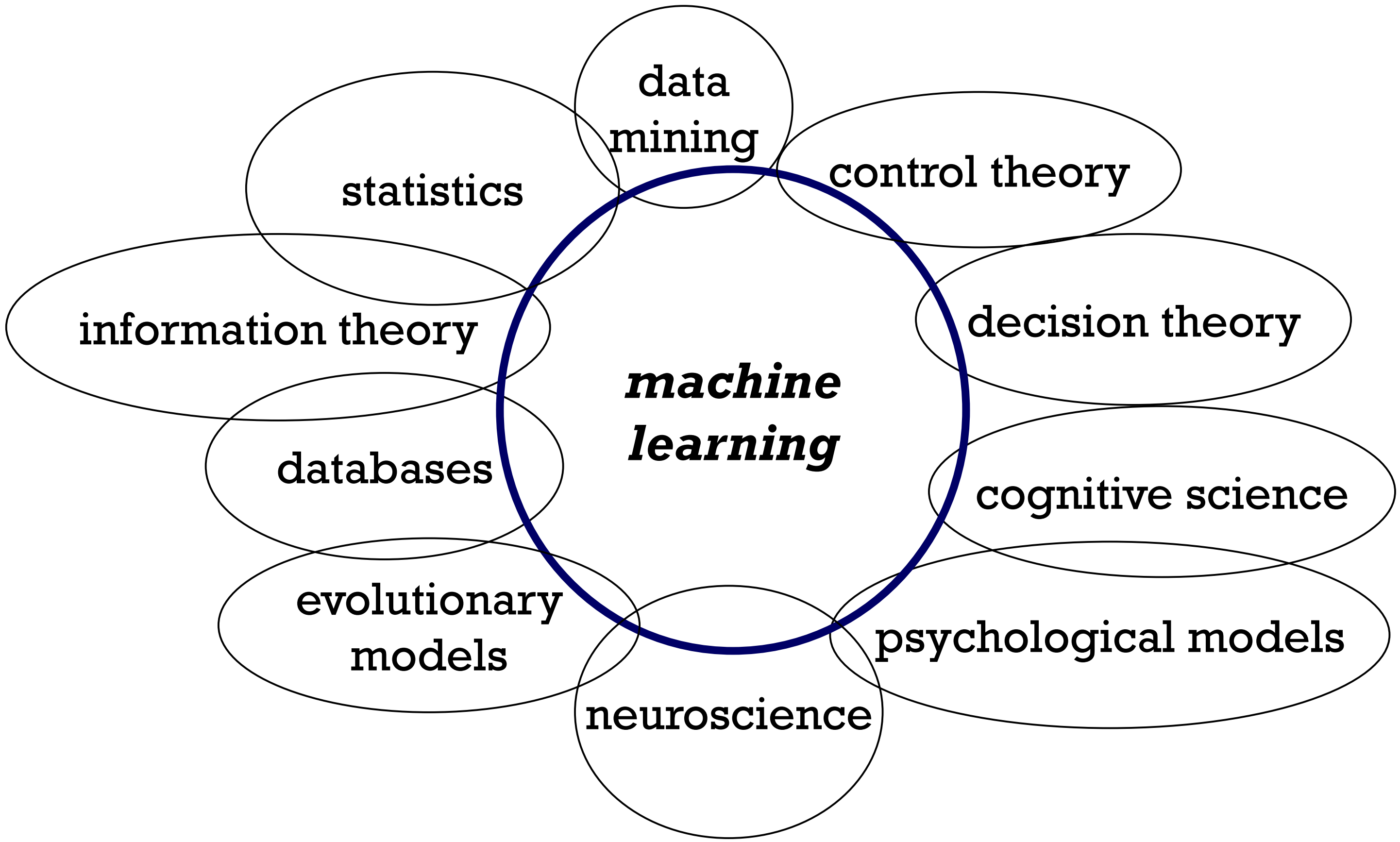
- *"Learning denotes changes in a system that ... enable a system to do the same task … more efficiently the next time."* - Herbert Simon
- *"Learning is constructing or modifying representations of what is being experienced."* - Ryszard Michalski
- *"Learning is making useful changes in our minds."* - Marvin Minsky

"Machine learning refers to a system capable of the autonomous acquisition and integration of knowledge."

WHY MACHINE LEARNING?

- No human experts
  - industrial/manufacturing control
  - mass spectrometer analysis, drug design, astronomic discovery
- Black-box human expertise
  - face/handwriting/speech recognition
  - driving a car, flying a plane
- Rapidly changing phenomena
  - credit scoring, financial modeling
  - diagnosis, fraud detection
- Need for customization/personalization
  - personalized news reader
  - movie/book recommendation

RELATED FIELDS

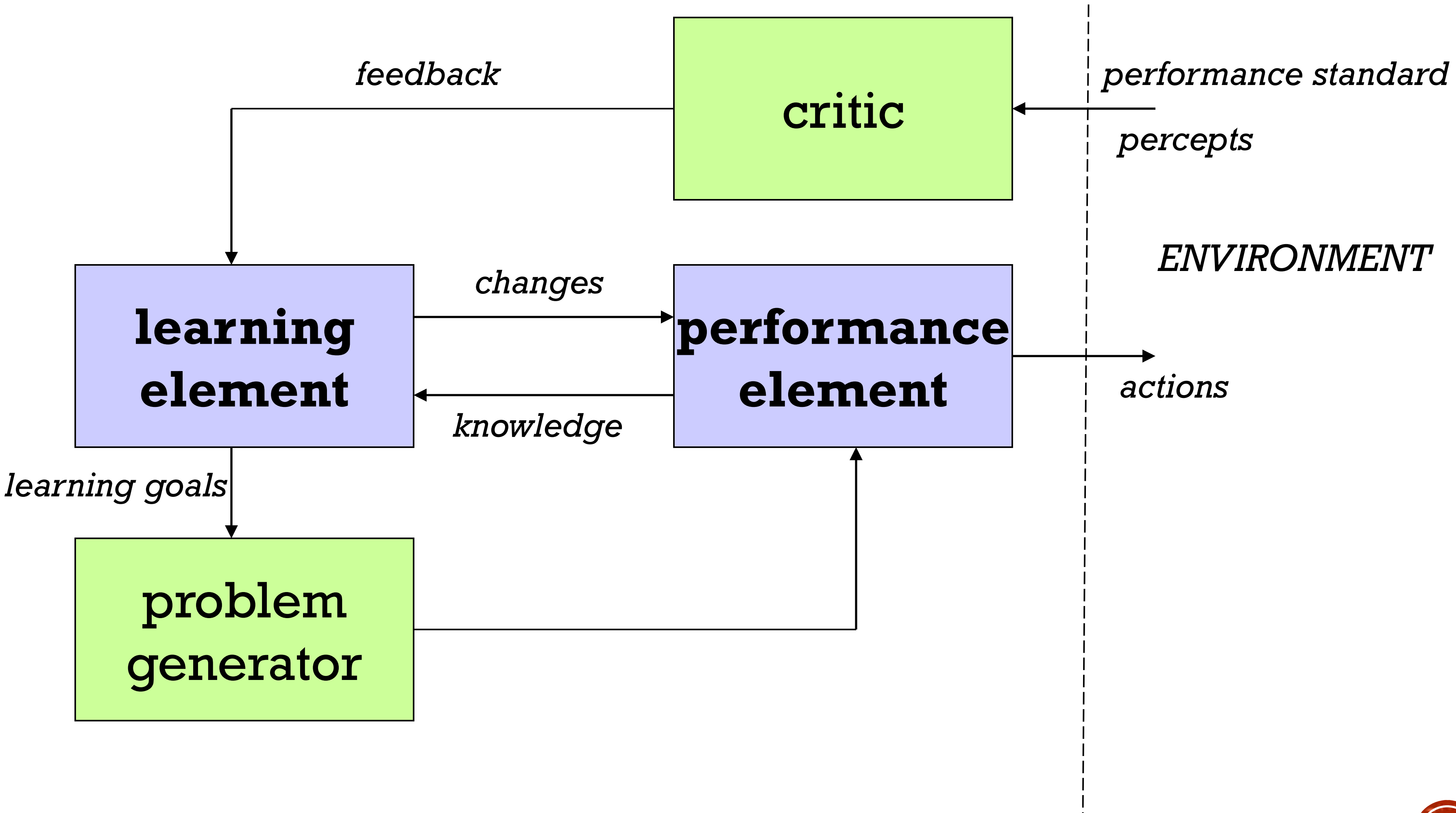


Machine learning is primarily concerned with the accuracy and effectiveness of the *computer system*.

MACHINE LEARNING PARADIGMS

- rote learning
- learning by being told (advice-taking)
- learning from examples (induction)
- learning by analogy
- speed-up learning
- concept learning
- clustering
- discovery
- …

ARCHITECTURE OF A LEARNING SYSTEM



LEARNING ELEMENT

Design affected by:

- *performance element* used
  - e.g., utility-based agent, reactive agent, logical agent
- *functional component* to be learned
  - e.g., classifier, evaluation function, perception-action function,
- *representation* of functional component
  - e.g., weighted linear function, logical theory, HMM
- *feedback* available
  - e.g., correct action, reward, relative preferences

DIMENSIONS OF LEARNING SYSTEMS

- type of feedback
  - supervised (labeled examples)
  - unsupervised (unlabeled examples)
  - reinforcement (reward)
- representation
  - attribute-based (feature vector)
  - relational (first-order logic)
- use of knowledge
  - empirical (knowledge-free)
  - analytical (knowledge-guided)

OUTLINE

- Supervised learning
  - empirical learning (knowledge-free)
    - attribute-value representation
    - logical representation
  - analytical learning (knowledge-guided)
- Reinforcement learning
- Unsupervised learning
- *Performance evaluation*
- *Computational learning theory*

INDUCTIVE (SUPERVISED) LEARNING

Basic Problem: Induce a representation of a function (a systematic relationship between inputs and outputs) from examples.

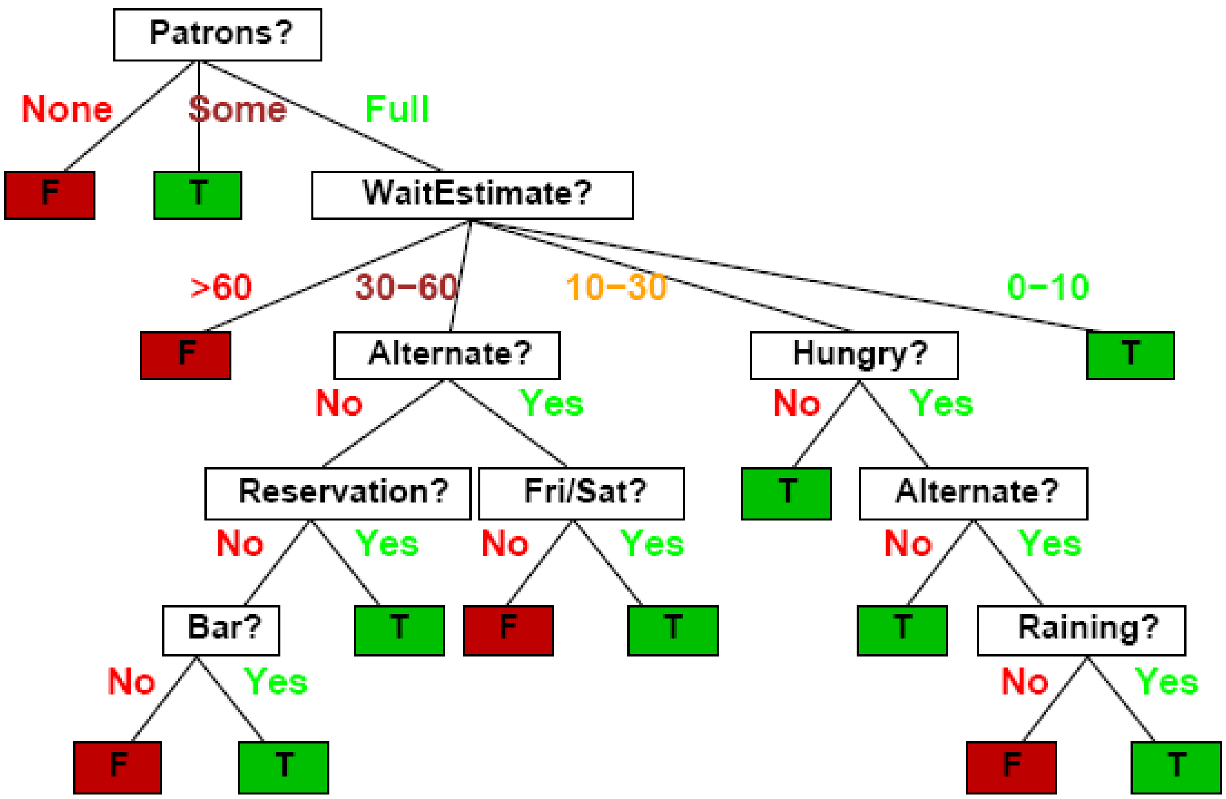
- **target function** $f\colon X \rightarrow Y$
- **example** $(x, f(x))$
- **hypothesis** $g\colon X \rightarrow Y$ *such that* $g(x) = f(x)$

x = set of attribute values (***attribute-value representation***)
x = set of logical sentences (*first-order representation*)

Y = set of discrete labels (***classification***)
$Y = \Re$ (*regression*)

DECISION TREES

Should I wait at this restaurant?



DECISION TREE INDUCTION

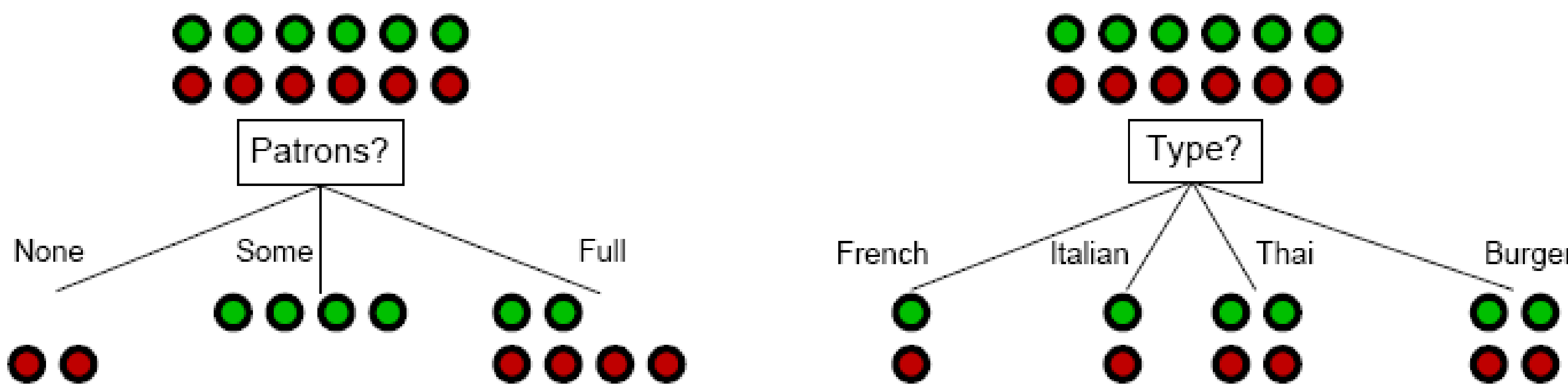
(Recursively) partition examples according to the *most important* attribute.

Key Concepts

- *entropy*
  - impurity of a set of examples (entropy = 0 if perfectly homogeneous)
  - (#bits needed to encode class of an arbitrary example)
- *information gain*
  - expected reduction in entropy caused by partitioning

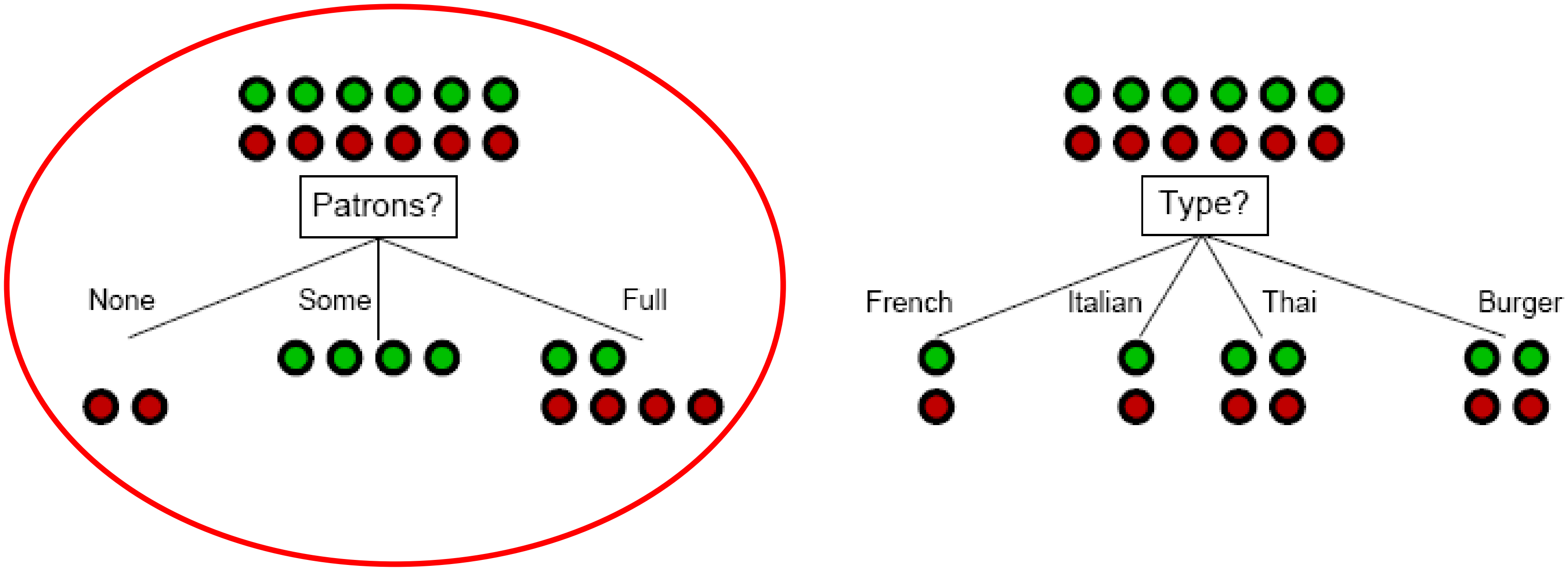
DECISION TREE INDUCTION: ATTRIBUTE SELECTION

Intuitively: *A good attribute* splits the examples into subsets that are (ideally) *all positive* or *all negative*.

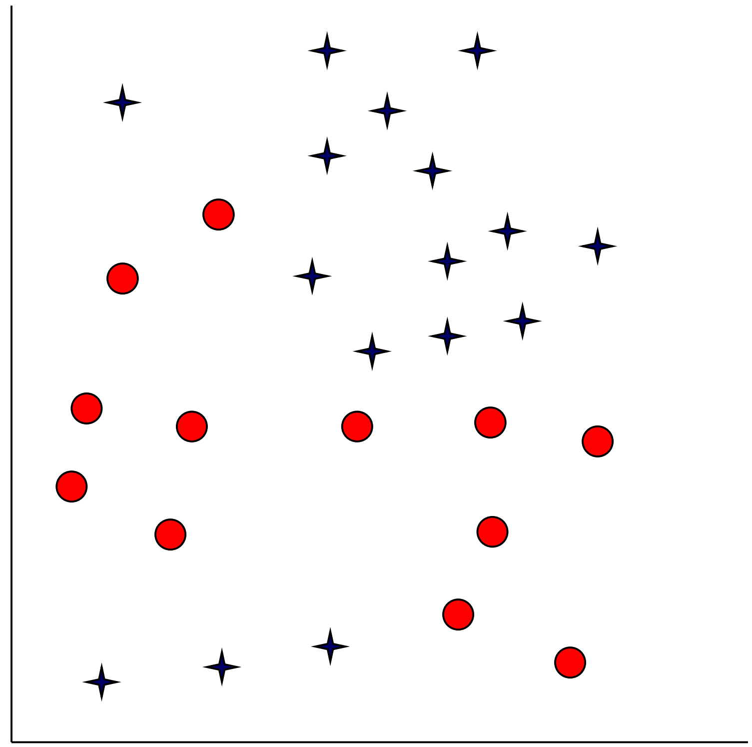


DECISION TREE INDUCTION: ATTRIBUTE SELECTION

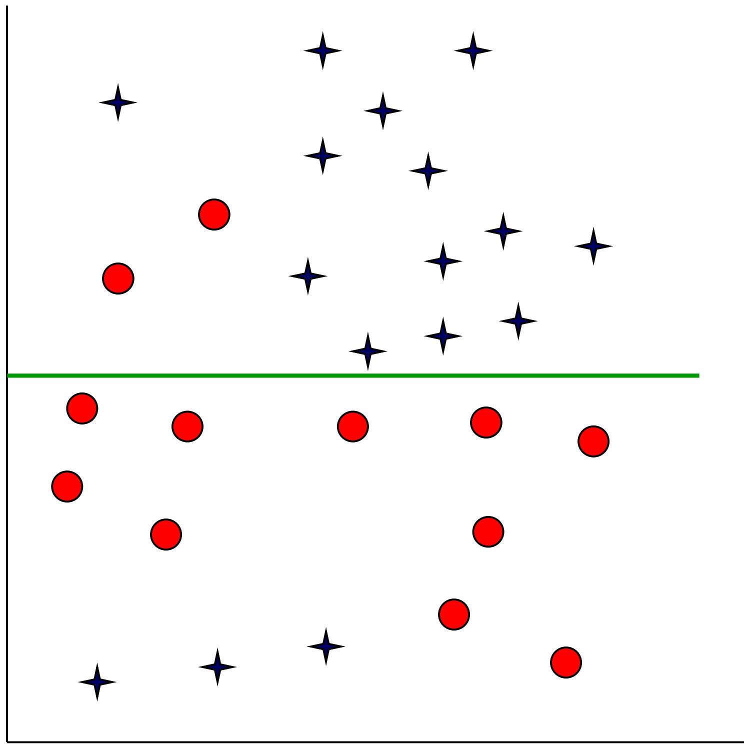
Intuitively: A *good attribute* splits the examples into subsets that are (ideally) *all positive* or *all negative*.



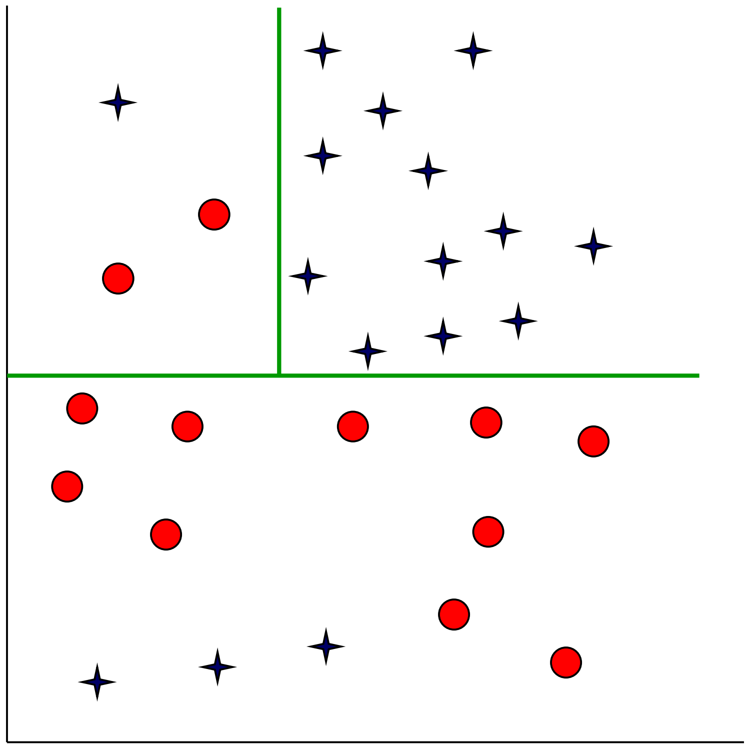
DECISION TREE INDUCTION: DECISION BOUNDARY



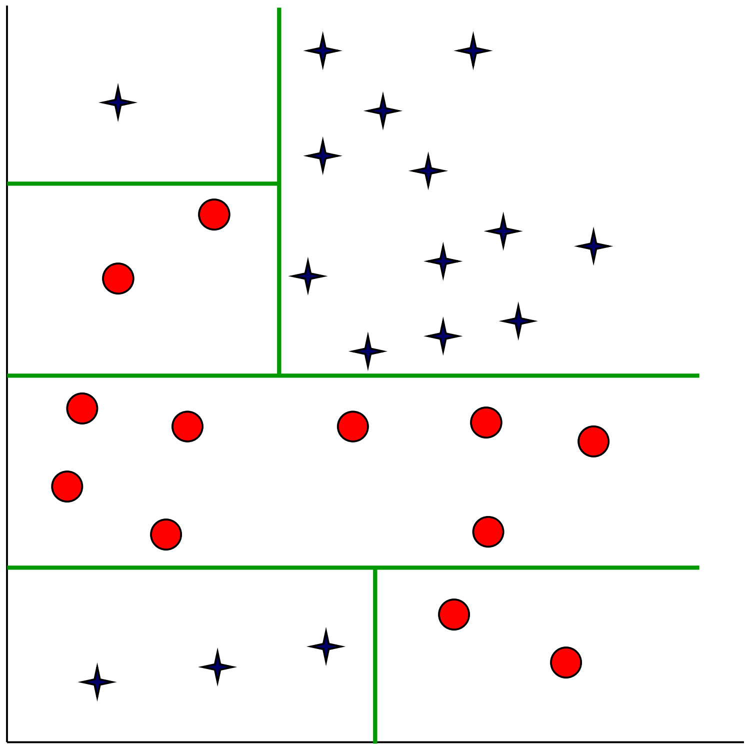
DECISION TREE INDUCTION: DECISION BOUNDARY



DECISION TREE INDUCTION: DECISION BOUNDARY

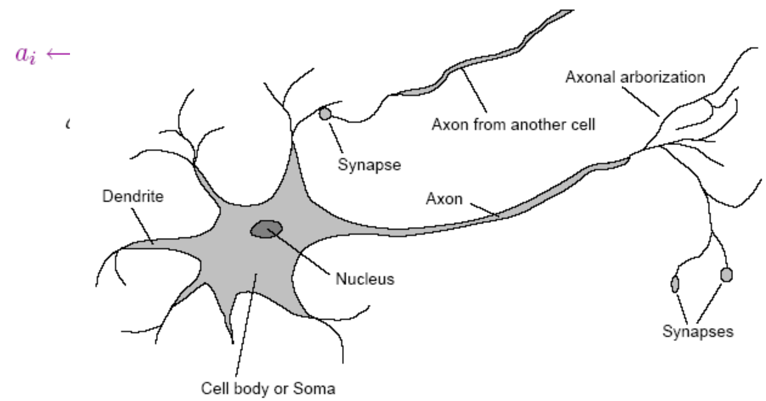


DECISION TREE INDUCTION: DECISION BOUNDARY



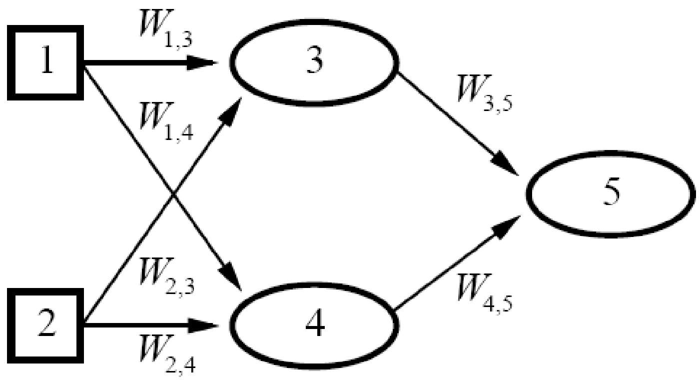
(ARTIFICIAL) NEURAL NETWORKS

- Motivation: human brain
  - massively parallel (*10^{11} neurons, ~20 types*)
  - small computational units with simple low-bandwidth communication (*10^{14} synapses, 1-10ms cycle time*)



- Realization: neural network
  - *units* (≈ neurons) connected by *directed weighted links*
  - *activation function* from inputs to output

NEURAL NETWORKS (*CONTINUED*)



$$
\begin{aligned}
a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\
&= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))
\end{aligned}
$$

- *neural network = parameterized family of nonlinear functions*
- *types*
  - *feed-forward* (acyclic): single-layer perceptrons, multi-layer networks
  - *recurrent* (cyclic): Hopfield networks, Boltzmann machines

[*connectionism, parallel distributed processing*]

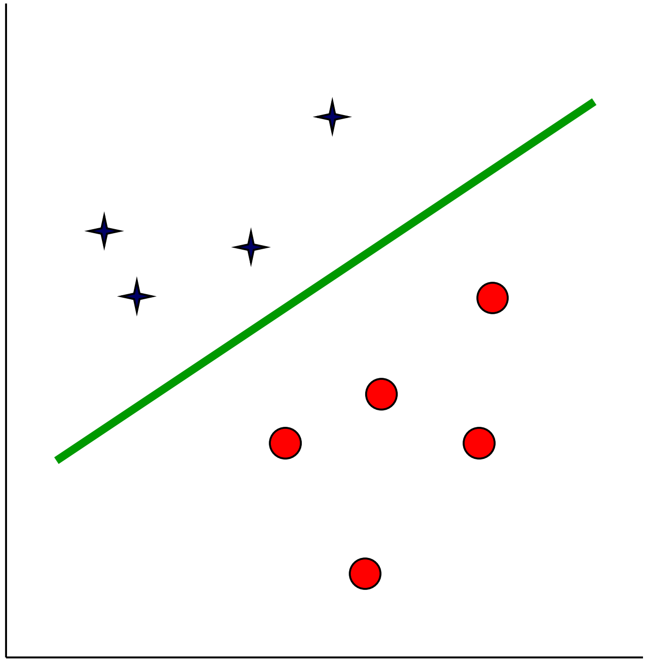
NEURAL NETWORK LEARNING

Key Idea: Adjusting the weights changes the function represented by the neural network (*learning = optimization in weight space*).

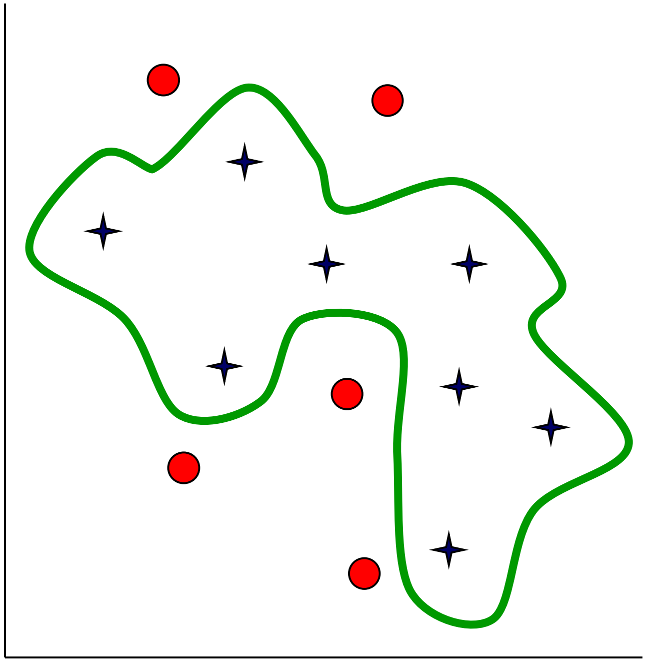
Iteratively *adjust weights* to reduce *error* (difference between network output and target output).

- Weight Update
  - *perceptron training rule*
  - *linear programming*
  - *delta rule*
  - *backpropagation*

NEURAL NETWORK LEARNING: DECISION BOUNDARY



single-layer perceptron



multi-layer network

SUPPORT VECTOR MACHINES

Kernel Trick: Map data to *higher-dimensional space* where they will be *linearly separable*.

Learning a Classifier

- optimal linear separator is one that has the *largest margin* between positive examples on one side and negative examples on the other
- = *quadratic programming optimization*

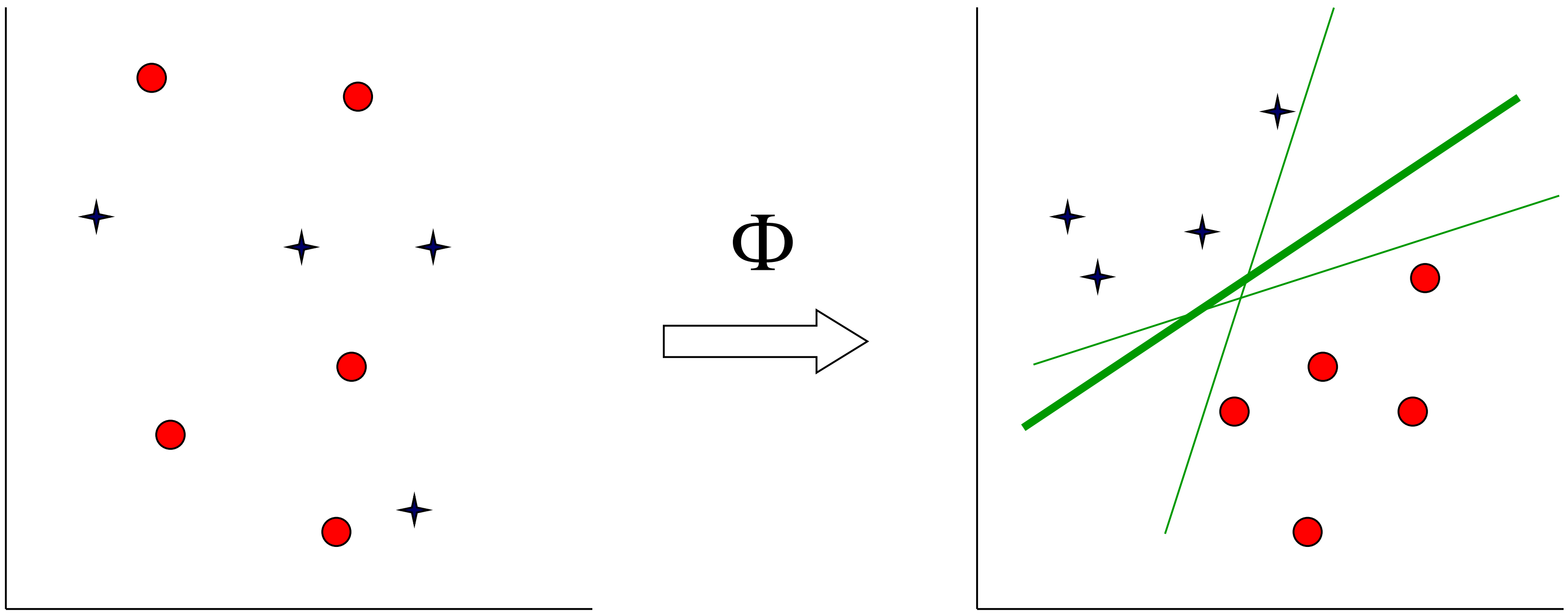
SUPPORT VECTOR MACHINES (*CONTINUED*)

Key Concept: Training data enters optimization problem in the form of *dot products* of pairs of *points*.

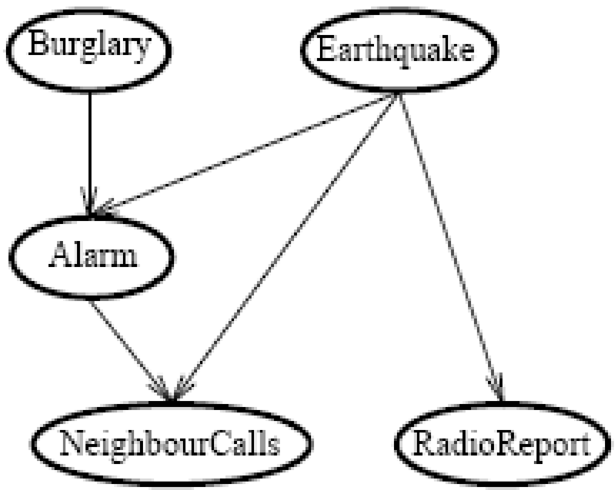
- *support vectors*
  - weights associated with data points are *zero* except for those points nearest the separator (i.e., the *support vectors*)
- *kernel function* $K(x_i,x_j)$
  - function that can be applied to pairs of points to evaluate dot products in the corresponding (higher-dimensional) feature space F (*without having to directly compute* F(x) *first*)

efficient training and *complex functions!*

SUPPORT VECTOR MACHINES: DECISION BOUNDARY



BAYESIAN NETWORKS



Network topology reflects direct *causal influence*

<u>*Basic Task*</u>: Compute probability distribution for unknown variables given observed values of other variables.

	A B	*A* $\neg$*B*	$\neg$*A B*	$\neg$*A* $\neg$*B*
C	0.9	0.3	0.5	0.1
$\neg$*C*	0.1	0.7	0.5	0.9

conditional probability table for NeighbourCalls

[*belief networks, causal networks*]

BAYESIAN NETWORK LEARNING

Key Concepts

- nodes (attributes) = random variables
- conditional independence
  - an attribute is conditionally independent of its non-descendants, given its parents
- conditional probability table
  - conditional probability distribution of an attribute given its parents
- Bayes Theorem
  - $P(h|D) = P(D|h)P(h) / P(D)$

Bayesian Network Learning (*Continued*)

Find *most probable hypothesis* given the data.

In theory: Use posterior probabilities to weight hypotheses. (*Bayes optimal classifier*)

In practice: Use single, *maximum a posteriori* (most probable) hypothesis.

Settings

- known structure, fully observable (*parameter learning*)
- unknown structure, fully observable (*structural learning*)
- known structure, hidden variables (*EM algorithm*)
- unknown structure, hidden variables (*?*)

NEAREST NEIGHBOR MODELS

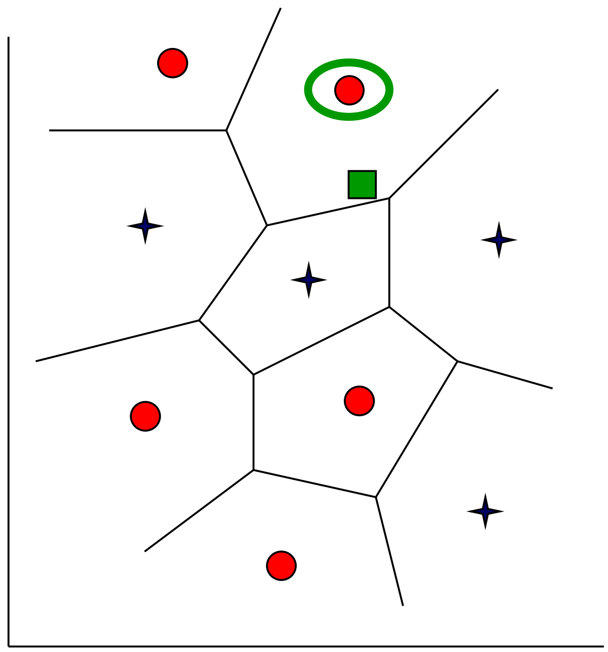
Key Idea: Properties of an input x are likely to be *similar* to those of points in the *neighborhood* of x.

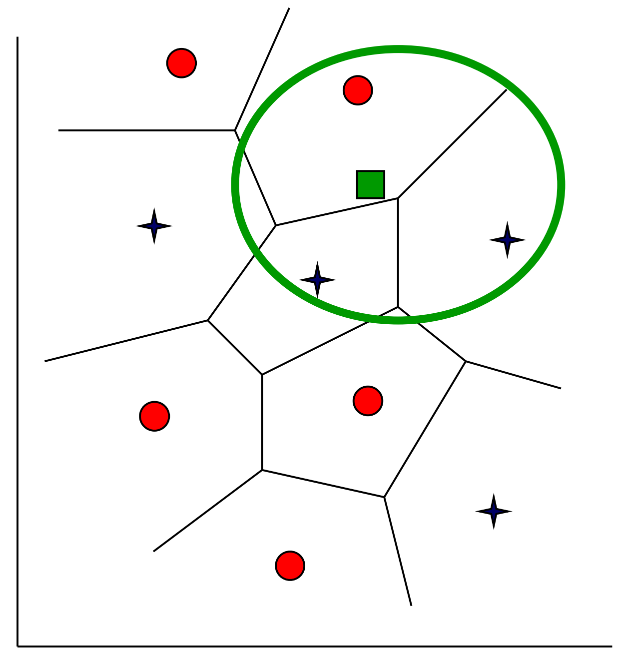
Basic Idea: Find (k) nearest neighbor(s) of x and infer target attribute value(s) of x based on corresponding attribute value(s).

Form of *non-parametric learning* where hypothesis complexity grows with data (learned model ≈ all examples seen so far)

[*instance-based learning, case-based reasoning, analogical reasoning*]

NEAREST NEIGHBOR MODEL: DECISION BOUNDARY





LEARNING LOGICAL THEORIES

Logical Formulation of Supervised Learning

- attribute $\rightarrow$ unary predicate
- instance $x \rightarrow$ logical sentence
- positive/negative classifications $\rightarrow$ sentences $Q(x_i), \neg Q(x_i)$
- training set $\rightarrow$ conjunction of all description and classification sentences

Learning Task: Find an *equivalent logical expression* for the goal predicate Q to classify examples correctly.

$$\boxed{\mathit{Hypothesis} \wedge \mathit{Descriptions} \models \mathit{Classifications}}$$

LEARNING LOGIC THEORIES: EXAMPLE

Input

- Father(Philip,Charles), Father(Philip,Anne), …
- Mother(Mum,Margaret), Mother(Mum,Elizabeth), …
- Married(Diana,Charles), Married(Elizabeth,Philip), …
- Male(Philip),Female(Anne),…
- *Grandparent(Mum,Charles),Grandparent(Elizabeth,Beatrice), ¬Grandparent(Mum,Harry),¬Grandparent(Spencer,Pete),…*

Output

- Grandparent(x,y) $\Leftrightarrow$

  $[\exists z\ \text{Mother}(x,z) \wedge \text{Mother}(z,y)] \vee [\exists z\ \text{Mother}(x,z) \wedge \text{Father}(z,y)] \vee$

  $[\exists z\ \text{Father}(x,z) \wedge \text{Mother}(z,y)] \vee [\exists z\ \text{Father}(x,z) \wedge \text{Father}(z,y)]$

LEARNING LOGIC THEORIES

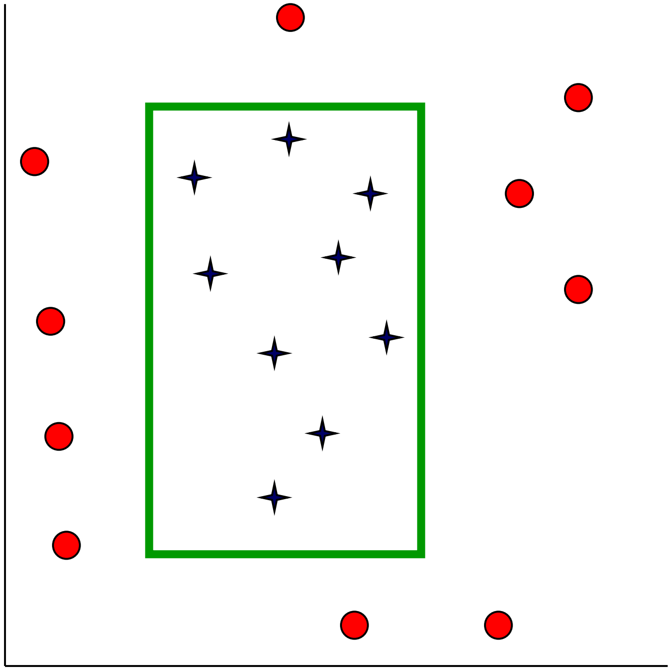
Key Concepts

- *specialization*
  - triggered by false positives (*goal: exclude negative examples*)
  - achieved by adding conditions, dropping disjuncts
- *generalization*
  - triggered by false negatives (*goal: include positive examples*)
  - achieved by dropping conditions, adding disjuncts

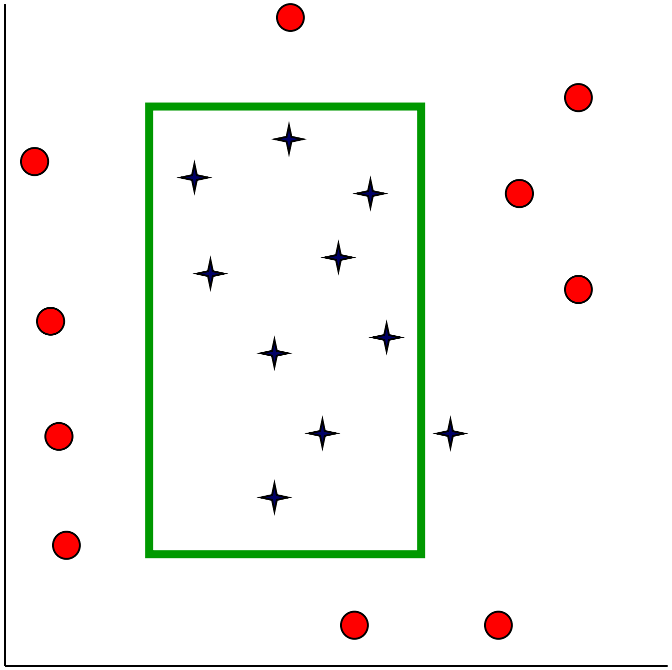
Learning

- *current-best-hypothesis*: incrementally improve single hypothesis (e.g., *sequential covering*)
- *least-commitment search*: maintain *all* hypotheses consistent with examples seen so far (e.g., *version space*)

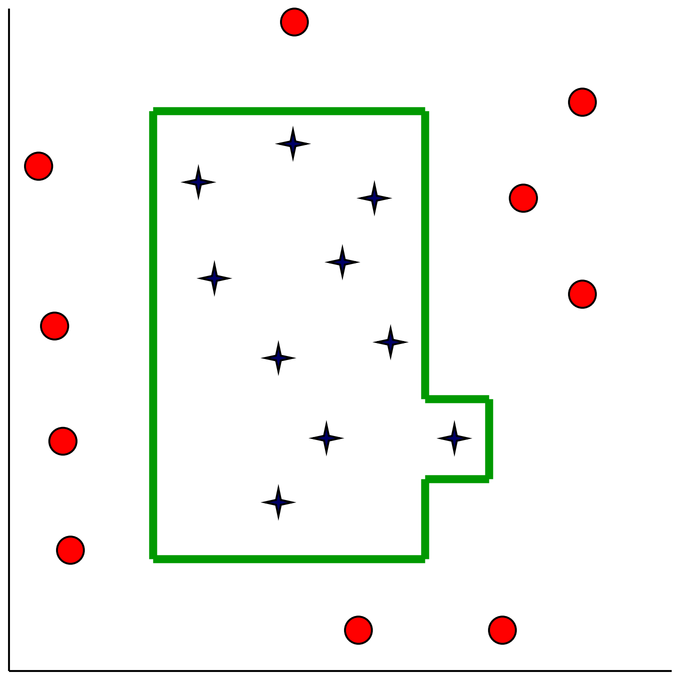
LEARNING LOGIC THEORIES: DECISION BOUNDARY



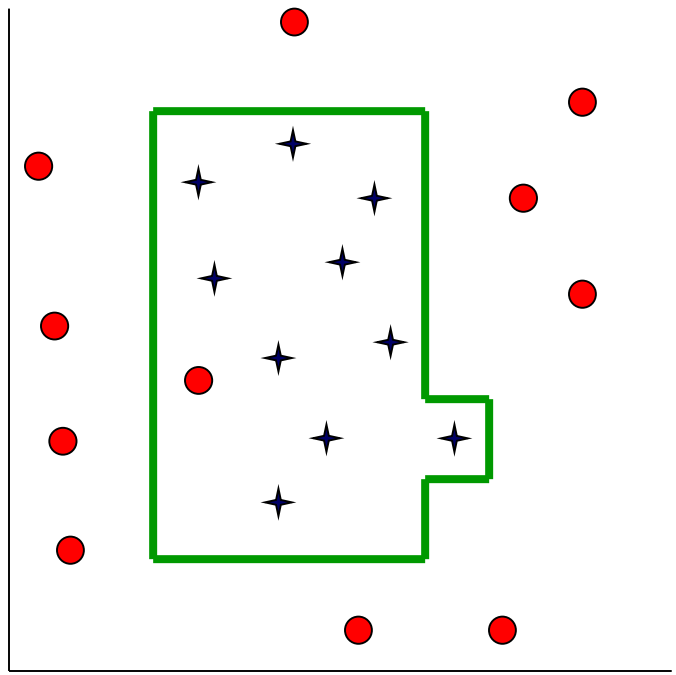
LEARNING LOGIC THEORIES: DECISION BOUNDARY



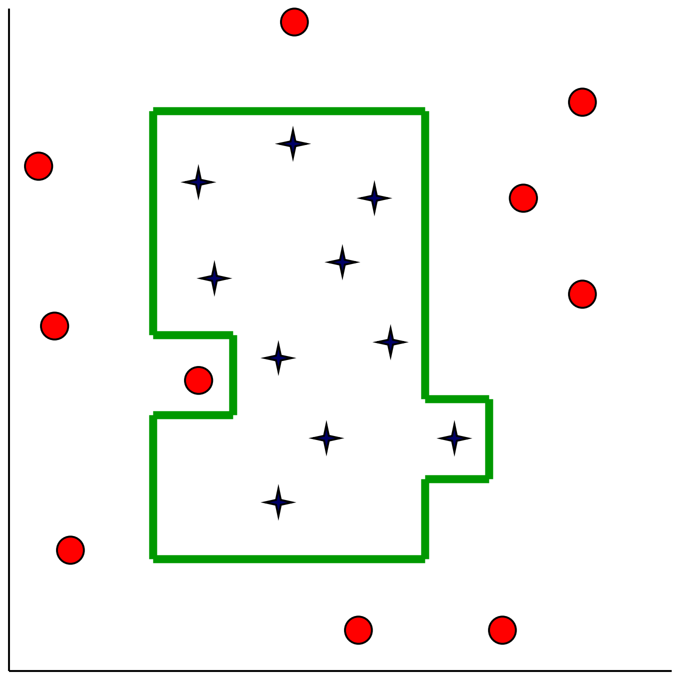
LEARNING LOGIC THEORIES: DECISION BOUNDARY



LEARNING LOGIC THEORIES: DECISION BOUNDARY



LEARNING LOGIC THEORIES: DECISION BOUNDARY



ANALYTICAL LEARNING

Prior Knowledge in Learning

Recall:

Grandparent(x,y) $\Leftrightarrow$

[$\exists$z Mother(x,z) $\wedge$ Mother)] $\vee$ [$\exists$z Mother(x,z) $\wedge$ Father(z,y)] $\vee$

[$\exists$z Father(x,z) $\wedge$ Mother(z,y)] $\vee$ [$\exists$z Father(x,z) $\wedge$ Father(z,y)]

- Suppose initial theory also included:
  - Parent(x,y) $\Leftrightarrow$ [Mother(x,y) $\vee$ Father(x,y)]
- Final Hypothesis:
  - Grandparent(x,y) $\Leftrightarrow$ [$\exists$z Parent(x,z) $\wedge$ Parent(z,y)]

Background knowledge *can dramatically reduce the size of the hypothesis (greatly simplifying the learning problem).*

EXPLANATION-BASED LEARNING

Amazed crowd of cavemen observe Zog roasting a lizard on the end of a pointed stick ("Look what Zog do!") and thereafter abandon roasting with their bare hands.

Basic Idea: Generalize by *explaining* observed instance.

- form of *speedup learning*
  - doesn't learn anything factually new from the observation
  - instead converts first-principles theories into *useful* special-purpose knowledge

- utility problem
  - cost of determining if learned knowledge is applicable may outweight benefits from its application

RELEVANCE-BASED LEARNING

Mary travels to Brazil and meets her first Brazilian (Fernando), who speaks Portuguese. She concludes that all Brazilians speak Portuguese but not that all Brazilians are named Fernando.

Basic Idea: Use knowledge of what is *relevant* to infer new properties about a new instance.

- form of *deductive learning*
  - learns a new general rule that explains observations
  - does not create knowledge outside logical content of prior knowledge and observations

KNOWLEDGE-BASED INDUCTIVE LEARNING

Medical student observes consulting session between doctor and patient at the end of which the doctor prescribes a particular medication. Student concludes that the medication is effective treatment for a particular type of infection.

Basic Idea: Use prior knowledge to *guide hypothesis generation*.

- benefits in inductive logic programming
  - only hypotheses consistent with prior knowledge and observations are considered
  - prior knowledge supports smaller (simpler) hypotheses

REINFORCEMENT LEARNING

k-armed bandit problem:

Agent is in a room with k gambling machines (one-armed bandits). When an arm is pulled, the machine pays off 1 or 0, according to some unknown probability distribution. Given a fixed number of pulls, what is the agent's (optimal) strategy?

Basic Task: Find a policy π, mapping states to actions, that maximizes (long-term) reward.

Model (*Markov Decision Process*)

- set of states S
- set of actions A
- reward function $R : S \times A \rightarrow \Re$
- state transition function $T : S \times A \rightarrow \Pi(S)$
  - $T(s,a,s')$ = probability of reaching s' when a is executed in s

REINFORCEMENT LEARNING (*CONTINUED*)

- Settings
  - fully vs. partially observable environment
  - deterministic vs. stochastic environment
  - model-based vs. model-free
  - rewards in goal state only or in any state

value of a state: expected *infinite discounted sum of reward* the agent will gain if it starts from that state and *executes the optimal policy*

Solving MDP when the model is known

- *value iteration*: find optimal value function (derive optimal policy)
- *policy iteration*: find optimal policy directly (derive value function)

REINFORCEMENT LEARNING (*CONTINUED*)

Reinforcement learning is concerned with finding an optimal policy for an MDP when the *model* (transition, reward) *is unknown*.

exploration/exploitation tradeoff

model-free reinforcement learning

- learn a controller without learning a model first
- e.g., *adaptive heuristic critic* (TD(λ)), *Q-learning*

model-based reinforcement learning

- learn a model first
- e.g., *Dyna, prioritized sweeping, RTDP*

UNSUPERVISED LEARNING

Learn patterns from (unlabeled) data.

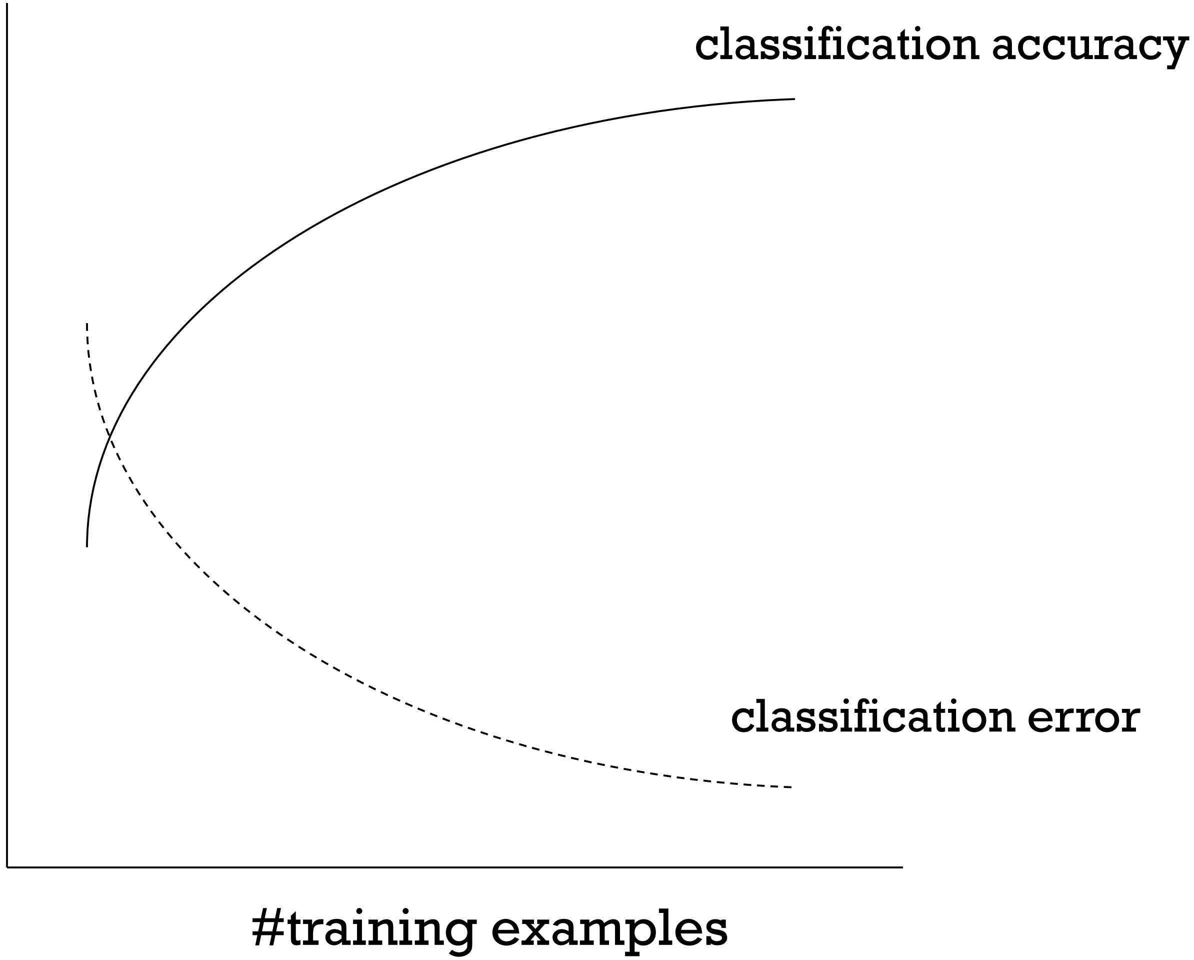
Approaches
- clustering (similarity-based)
- density estimation (e.g., EM algorithm)

Performance Tasks
- understanding and visualization
- anomaly detection
- information retrieval
- data compression

PERFORMANCE EVALUATION

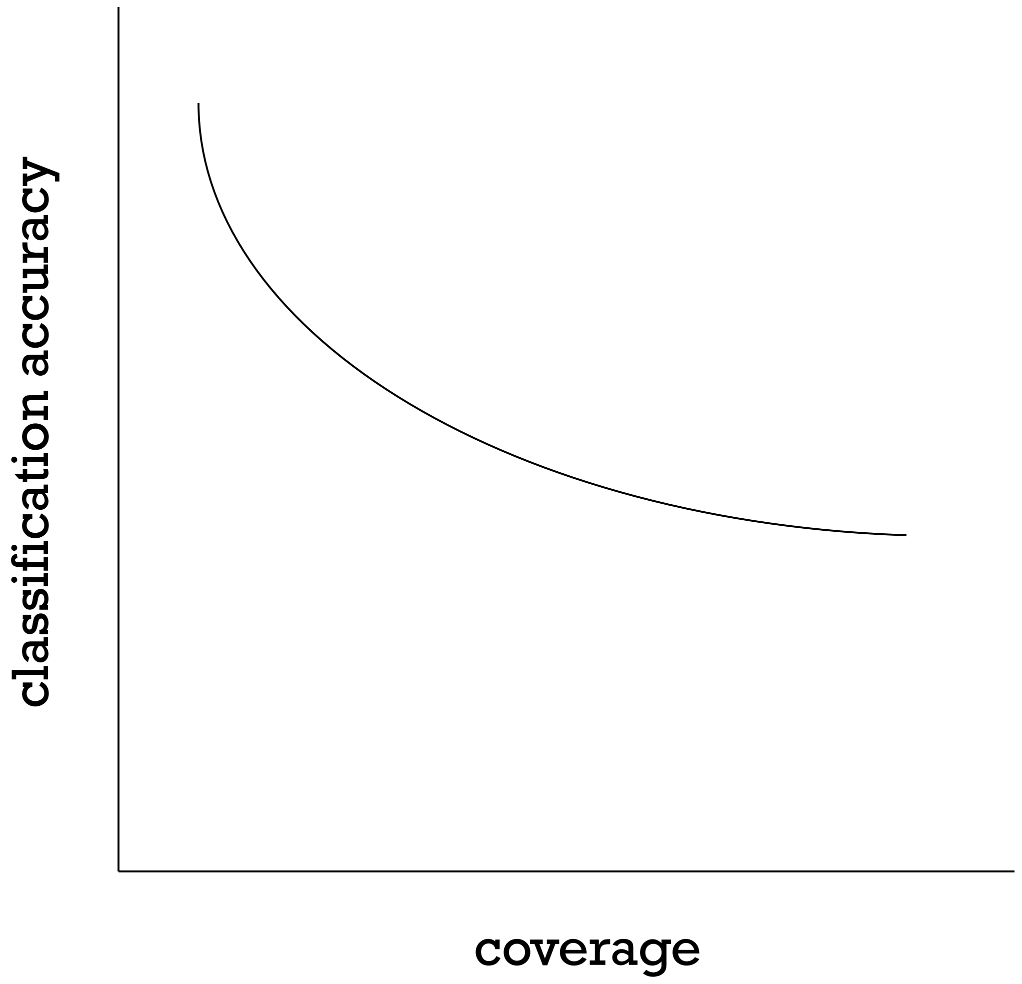
- Randomly split examples into *training set U* and *test set V*.
- Use training set to learn a hypothesis *H*.
- Measure % of *V* correctly classified by *H.*
- Repeat for different random splits and average results.

PERFORMANCE EVALUATION: LEARNING CURVES



PERFORMANCE EVALUATION: ROC CURVES

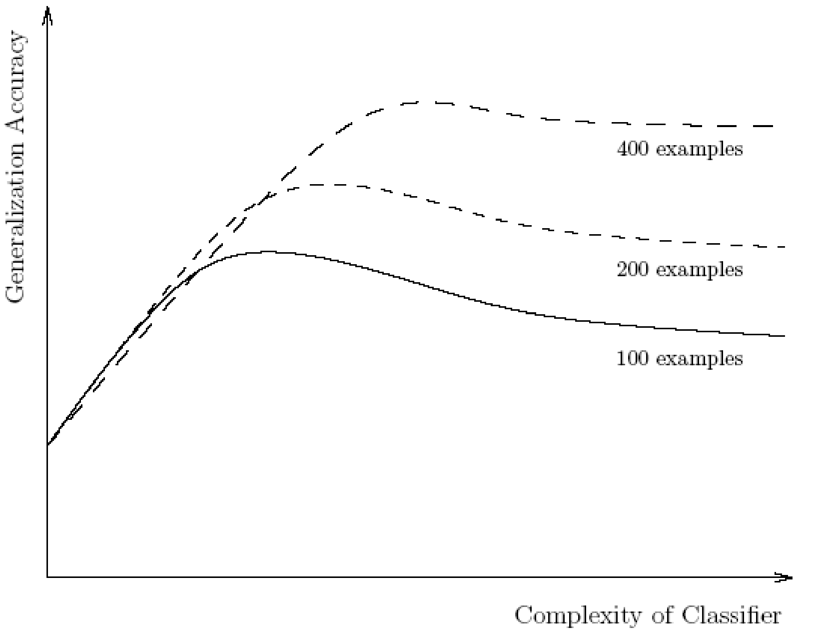
PERFORMANCE EVALUATION: ACCURACY/COVERAGE



TRIPLE TRADEOFF IN EMPIRICAL LEARNING

- size/complexity of learned classifier
- amount of training data
- generalization accuracy

bias-variance tradeoff



COMPUTATIONAL LEARNING THEORY

probably approximately correct (PAC) learning

With probability $\geq 1 - \delta$, error will be $\leq \varepsilon$.

<u>*Basic principle*</u>: Any hypothesis that is seriously wrong will almost certainly be found out with high probability after a small number of examples.

Key Concepts

$$m \geq \frac{1}{\varepsilon}(\ln \frac{1}{\delta} + \ln |H|)$$

- examples drawn from same distribution (*stationarity assumption*)
- *sample complexity* is a function of confidence, error, and *size of hypothesis space*

CURRENT MACHINE LEARNING RESEARCH

- Representation
  - data sequences
  - spatial/temporal data
  - probabilistic relational models
  - …
- Approaches
  - ensemble methods
  - cost-sensitive learning
  - active learning
  - semi-supervised learning
  - collective classification
  - …