Introduction to Statistical Learning Theory

Material para

Máster en Matemáticas y Computación

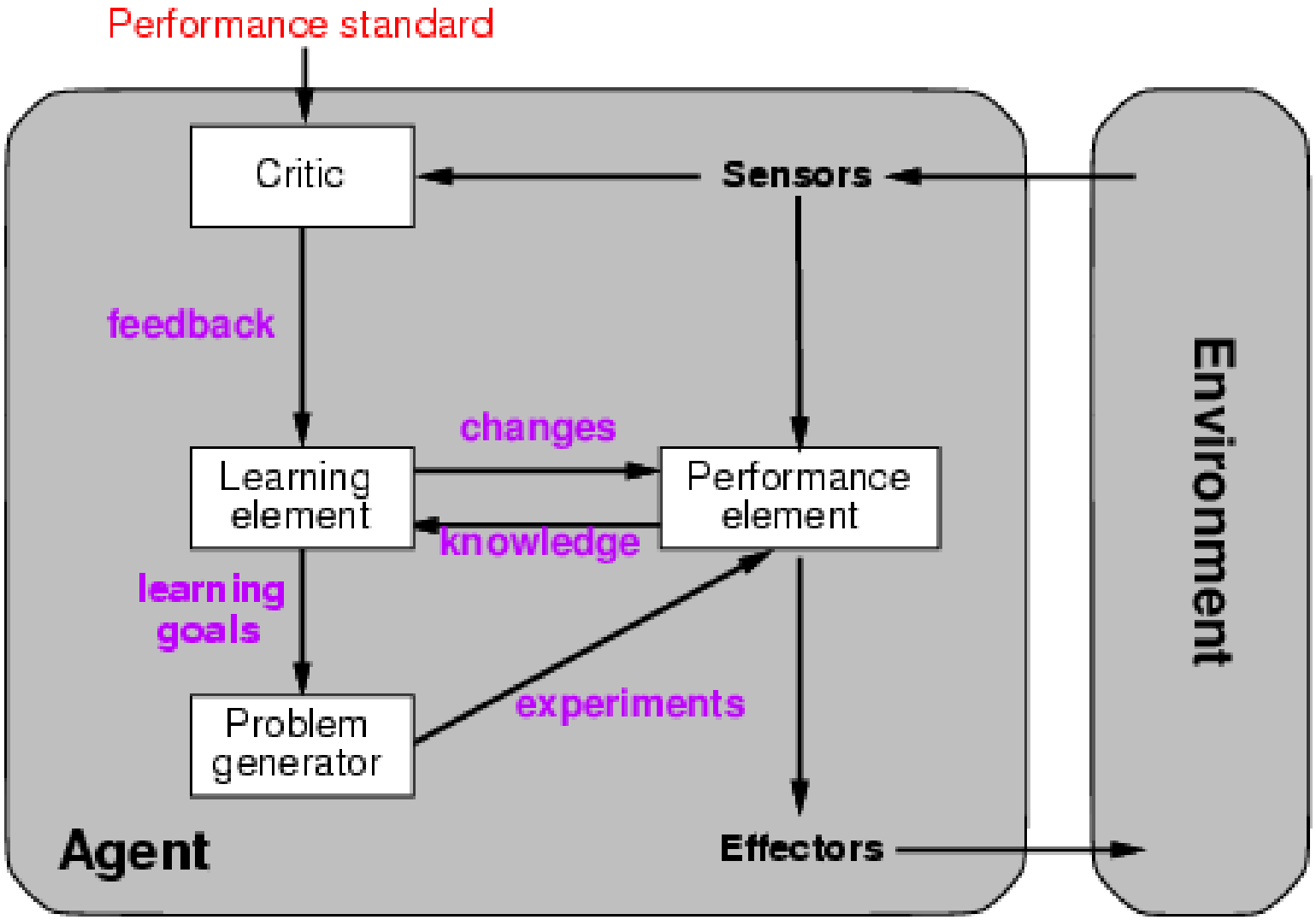
First Part: Outline

- Learning agents
- Inductive learning
- Decision tree learning

Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to wr ite it down
- Learning modifies the agent's decision mechanisms to i mprove performance

Learning agents



Learning element

- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn these components
  - What representation is used for the components

- Type of feedback:
  - Supervised learning: correct answers for each example
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards

Inductive learning

- Simplest form: learn a function from examples

f is the target function

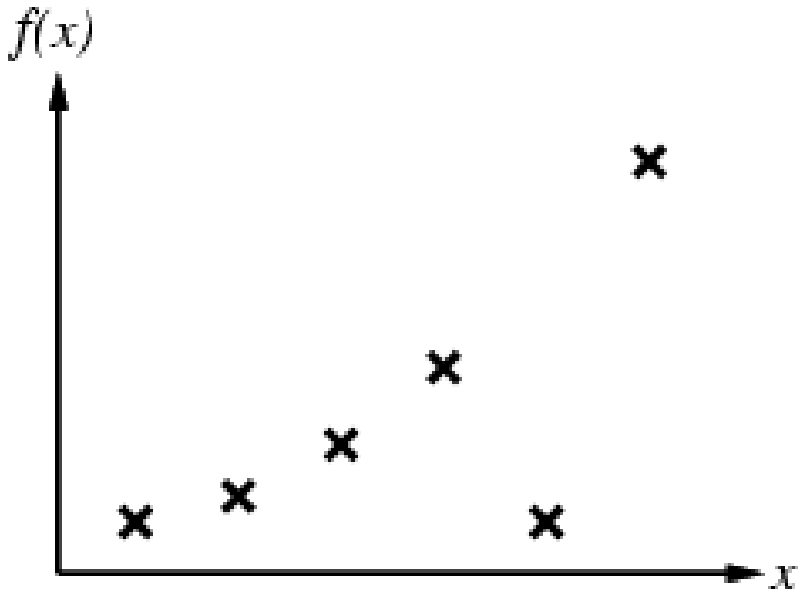
An example is a pair (*x*, *f(x)*)

Problem: find a hypothesis *h*
such that $h \approx f$
given a training set of examples

(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

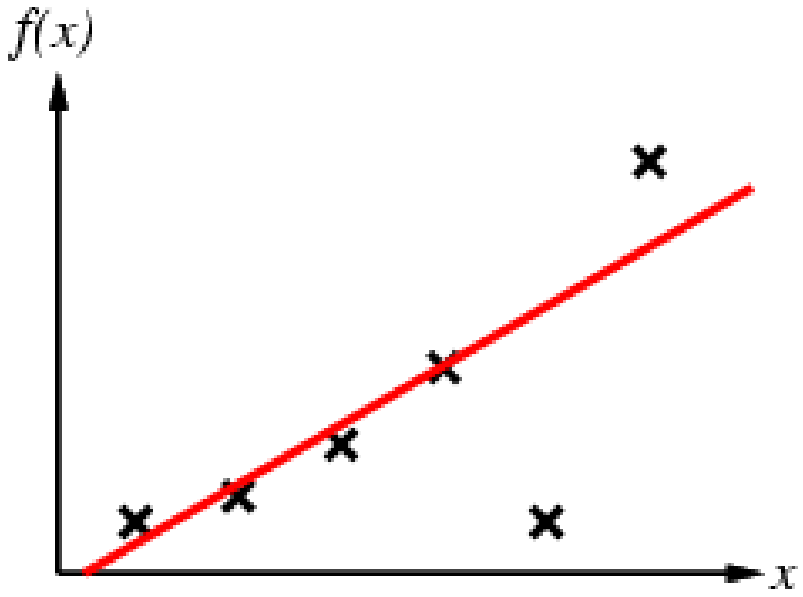
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



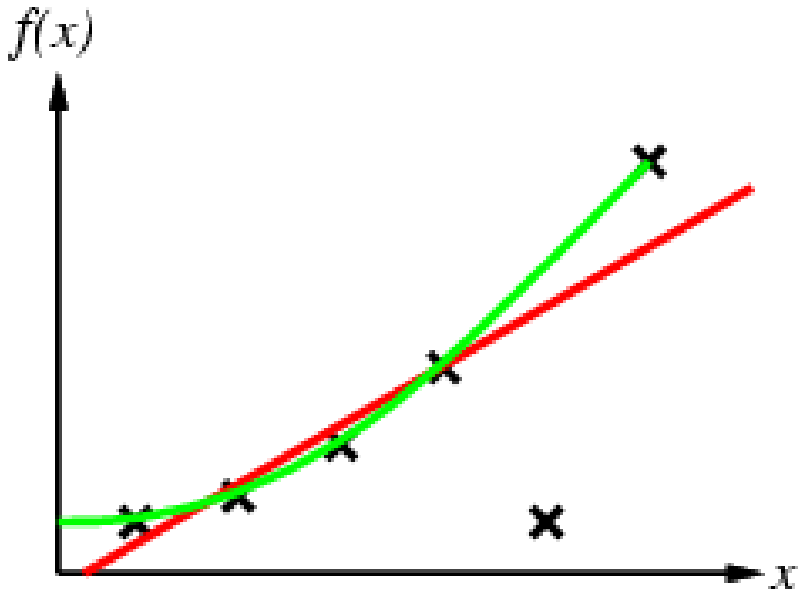
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



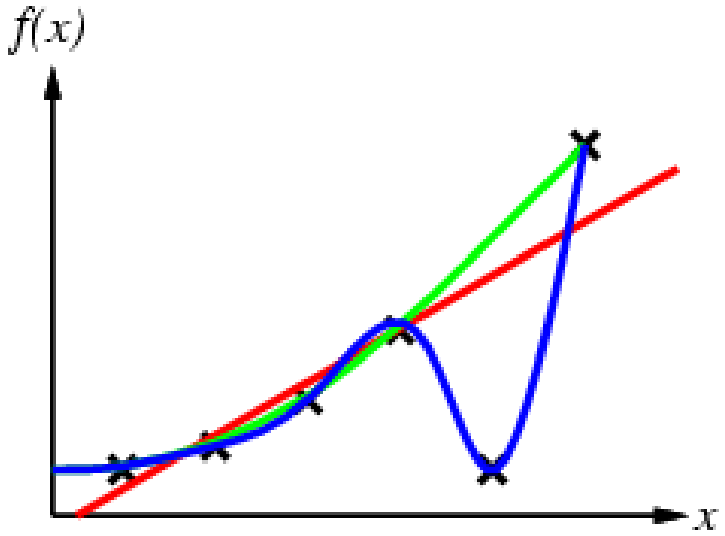
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



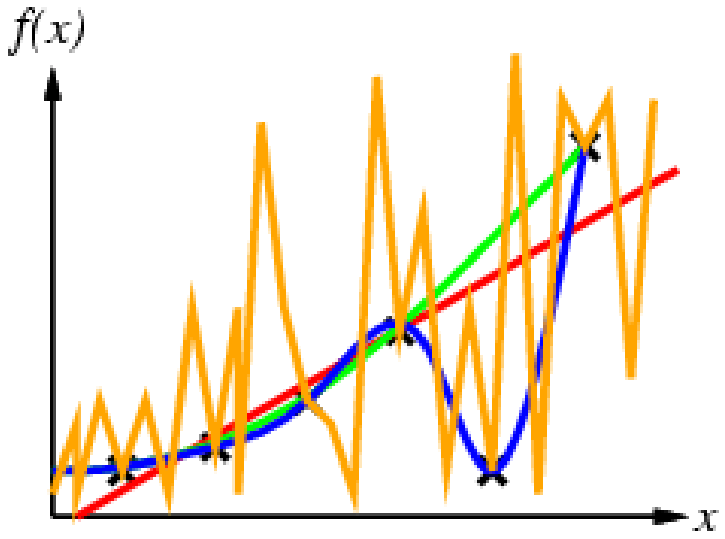
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



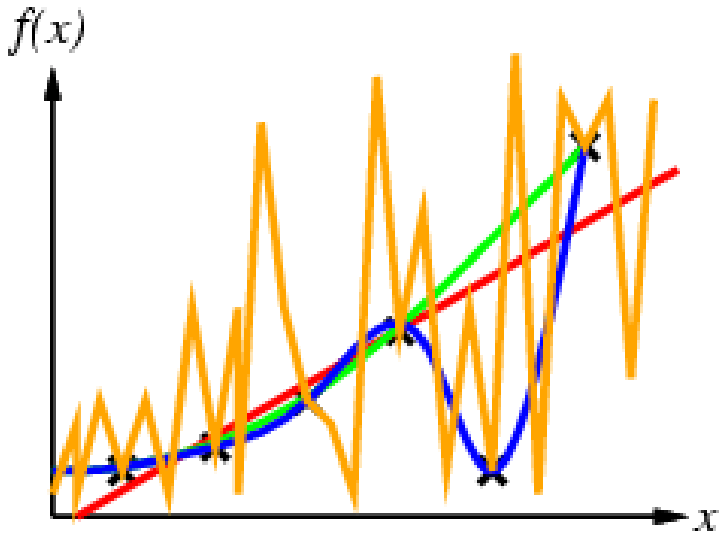
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



- Ockham's razor: prefer the simplest hypothesis consistent with data

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on th e following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

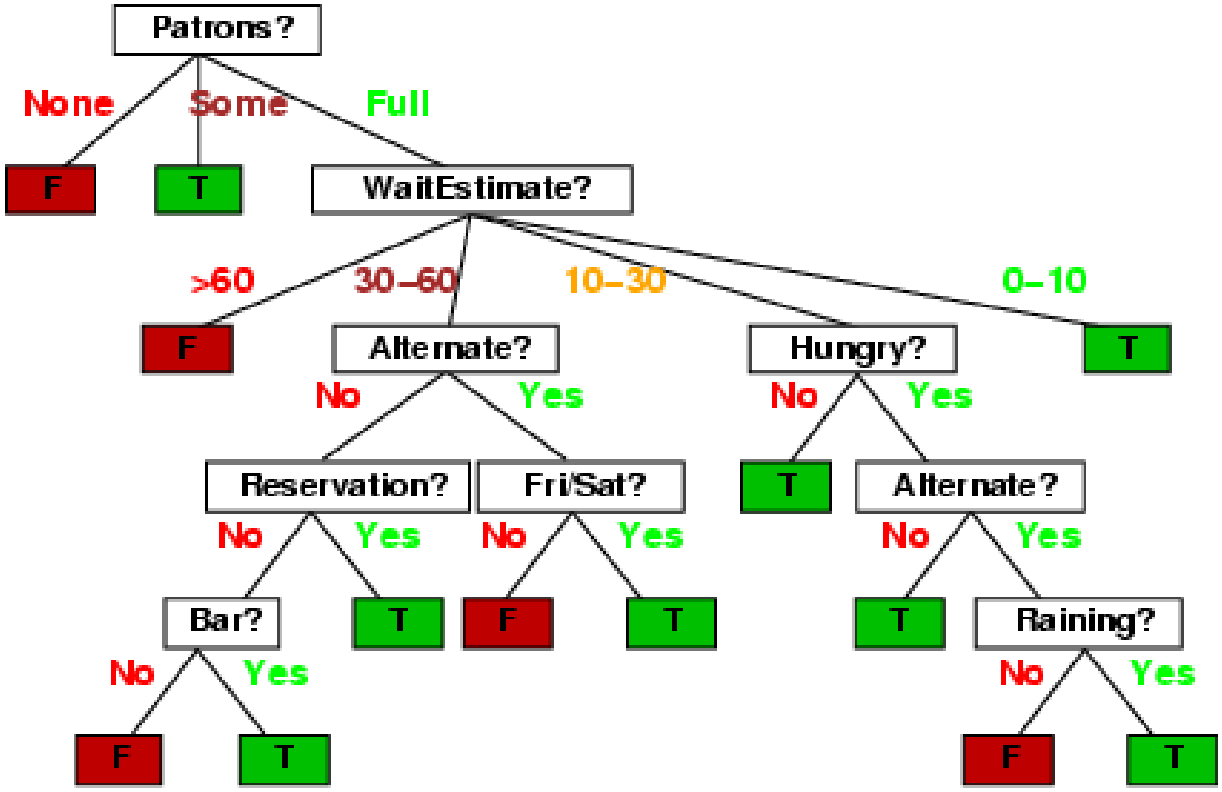
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Class

Decision trees

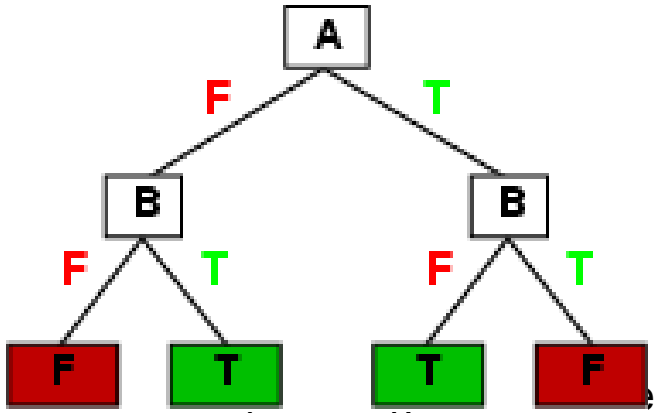
- One possible representation for hypotheses
- E.g., here is the "true" tree for deciding whether to wait:



Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivial[ly], [a] path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

- Prefer to find more compact decision trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?
= number of Boolean functions
= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., *Hungry* $\wedge$ $\neg$*Rain*)?

- Each attribute can be in (positive), in (negative), or out
  $\Rightarrow$ 3^n distinct conjunctive hypotheses
- More expressive hypothesis space
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set
    $\Rightarrow$ may get worse predictions

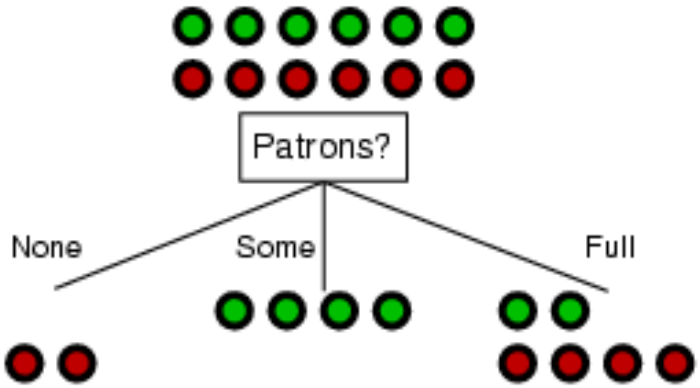
Decision tree learning

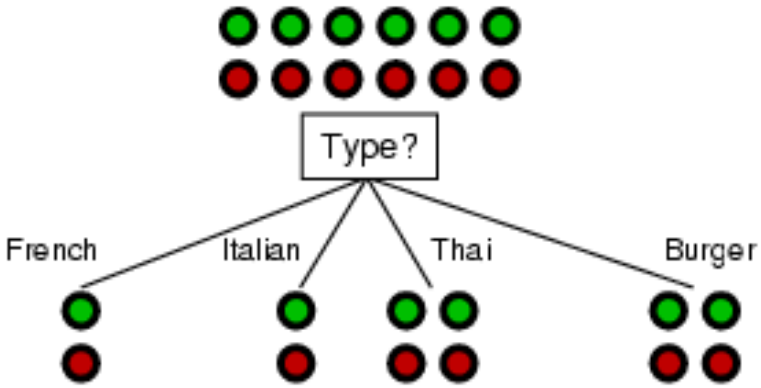
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (idea lly) "all positive" or "all negative"





Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- Information Content (Entropy):

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- For a training set containing *p* positive examples and *n* n egative examples:

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information gain

- A chosen attribute A divides the training set E into subsets $E_1, \ldots,$
- E_v according to their values for A, where A has v distinct values.
- Information Gain (IG) or reduction in entropy from the attribute test

$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A)$$

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i})$$

- Choose the attribute with the largest IG

Information gain

For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

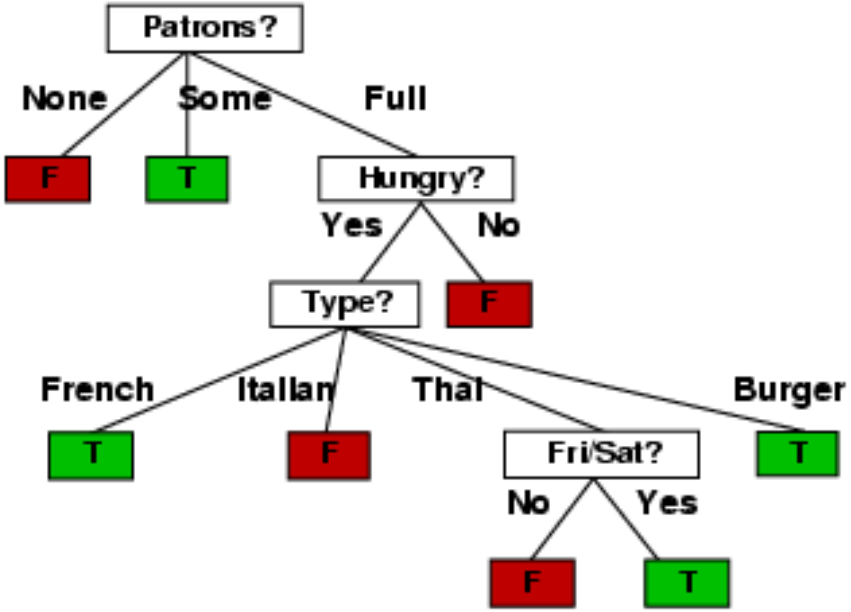
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Example contd.

- Decision tree learned from the 12 examples:

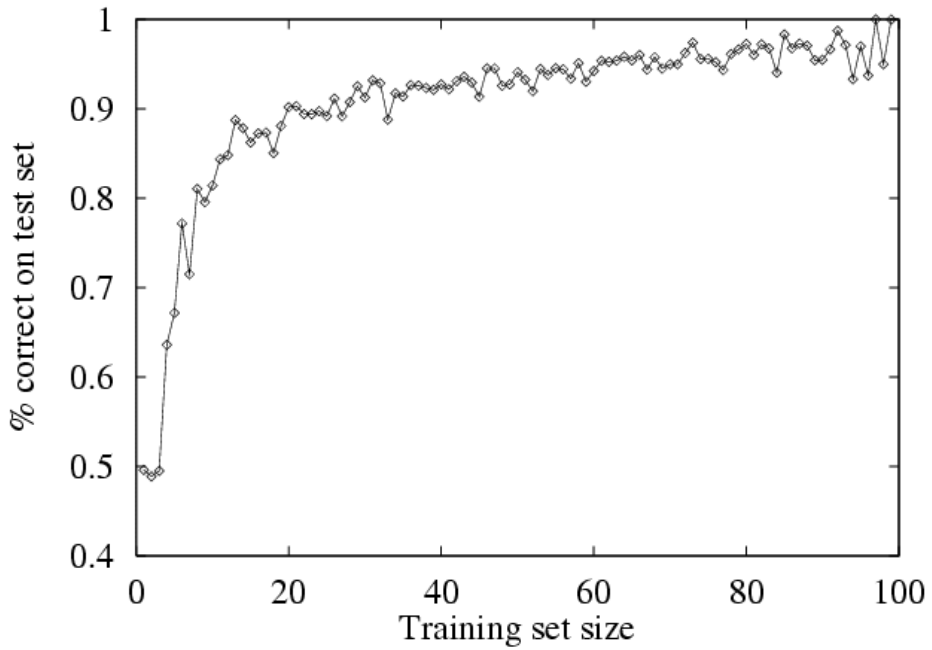


- Substantially simpler than "true" tree---a more complex hypothesis i sn't justified by small amount of data

Performance measurement

- How do we know that $h \approx f$?
  1. Use theorems of computational/statistical learning theory
  2. Try h on a new test set of examples
     (use same distribution over example space as training set)

Learning curve = % correct on test set as a function of training set size



Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set

Second Part: outline

- Learning and inference
- The role of sampling

Reference

- Based on O. Bousquet's Notes in MLSS 2003 at http://www.cmap.polytechnique.fr/%7Ebousquet/mlss_slt.pdf

Learning and Inference

- Inductive Inference Process:
  a. Observation
  b. Modeling
  c. Prediction

- This is more or less the definition of natural science.
- The goal of machine learning is to automate this process.
- The goal of Learning Theory is to formalize it.

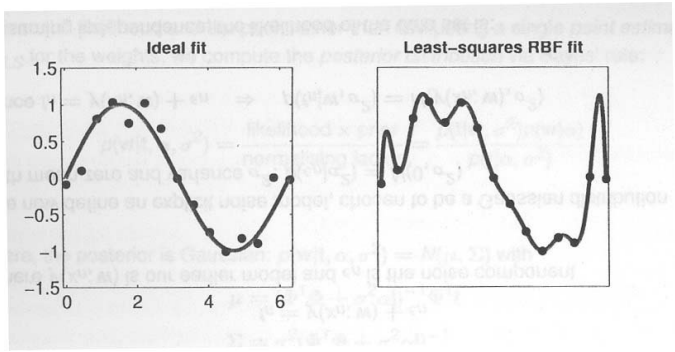
Pattern Recognition Problem

- Consider the supervised learning framework for pattern recognition.
  - Data consist of pairs (features, label).
  - Label is +1 or –1.
  - Given a dataset, a learning algorithm constructs a function which maps features to label.

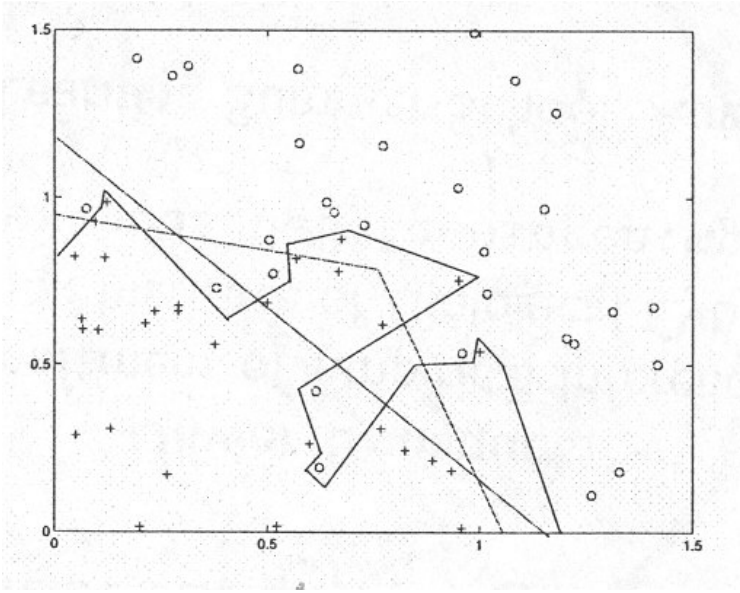
➢ Goal: Make few mistakes on unseen instances.

Need for Restriction on Model

- It is always possible to construct a function that fits the given data exactly. (For example, (google) Lagrange polynomial.)
- But is it *reasonable*? Is it *desirable*?



Ø Need a PRINCIPLE that restricts the functions a learning algorithm will construct.



No Free Lunch Principle

- No Free Lunch, or "Learning does not take place in vaccum."
  - Without assumption on how the past and the future are related, prediction is impossible.
  - Without restriction on the possible phenomenon (model, function class), generalization is impossible.

➢ Data will never replace knowledge.

Occam's Razor Principle

- *Entia non sunt multiplicanda praeter necessitatem*, or "Entities should not be multiplied beyond necessity."
- "Everything should be made as simple as possible, but not simpler." - A. Einstein
- Choose the simplest consistent model.
- Leaves the question: How to measure simplicity?
  - Description Length, number of parameters, number of hidden layers/nodes/states, etc.

➢ Statistical Learning Theory's measure of simplicity: VC dimension (among others).

General Assumptions in SLT

- Stationarity of the phenomenon
  - Observation and the structure of phenomenon do not change over time.
- Simplicity is preferred.
  - Constraints on the phenomenon.
- Probabilistic model for data
  - Data is generated from unknown but fixed probability distribution.
  - iid assumption on data

Probabilistic Model

- Data is generated from the same distribution, independently for each observation.
  - Independence implies that each new observation yields *maximum* information.
  - Each observation does give *information* about the underlying phenomenon (probability distribution), by the identity assumption.

- (Some people question this assumption.)

Probabilistic Model
Notations

- Input space $\mathcal{X}$
- output space $\mathcal{Y}$
  - For classification $\mathcal{Y}$={+1, -1}
- Assume there is an (unknown) probability distribution P on $\mathcal{X}\times\mathcal{Y}$.
- Data D={(X_i, Y_i)|i-=1…,n} is observed identically and independently according to P.

➢ Goal: Construct g:$\mathcal{X} \to \mathcal{Y}$ that predicts Y from X.

Probabilistic Model

Risk & Empirical Risk

- Risk

$$R(g)=P(g(X)\neq Y)=E(1_{g(X)\neq Y})$$

  - P is unknown so that we cannot compute this.

- Empirical Risk

$$R_n(g)=\sum_i 1_{g(X)\neq Y}/n$$

  - Dependent upon the data set.

Probabilistic Model

Bayes Classifier, Bayes Risk

- The best possible classifier is always given by

$$g^*(X)=\arg\max_Y P(Y|X)$$

and it is called the Bayes Classifier.

- Similarly, the Bayes risk is given by

$$R^*=\inf_g R(g)$$

It's the least risk that a deterministic function can have.

Probabilistic Model
Some Comments on P

- P can be decomposed as $P_X \times P(Y|X)$, i.e., the product of marginal distribution and the conditional distribution.
- In Bayesian framework, we make assumptions on P.
  - P_X is uniform, or
  - Give preference by a priori probability.
- SLT does the *worst case analysis*.

- (Some people even question the validity of the assumption that P exists.)

Basic Concepts

Empirical Risk Minimization (ERM)

- First we choose a model $\mathcal{G}$, the set of possible functions, or the function class.
- Then minimize the empirical risk *in the model:*

$$\inf_g R_n(g)$$

- Is the Bayes classifier in $\mathbb{G}$? Need it be?

Basic Concepts

Approximation & Estimation

- We compare the empirical risk $R_n(g)$ and the Bayes risk R^*.
- Decompose as follows:
  - Let $g^*=\arg\inf_{g\in\mathcal{G}} R(g)$ (sometimes called the Bayes classifier as well.)
  - $R_n(g)-R^* = \underbrace{R(g^*)-R^*}_{\text{Approximation error}} + \underbrace{R_n(g)-R(g^*)}_{\text{Estimation error}}$
  - Approximation error is completely determined by the distribution and the model.
  - Estimation error is a random variable, i.e., depends on the data.

Basic Concepts

Underfitting and Overfitting

- Underfitting
  - Model is too small to fit the data.
  - In other words, the approximation error is big.

- Overfitting
  - Artificially good agreement with the data.
  - Empirical risk is small only for given data, and gets big for other data.
  - Estimation error gets big.

Basic Concepts

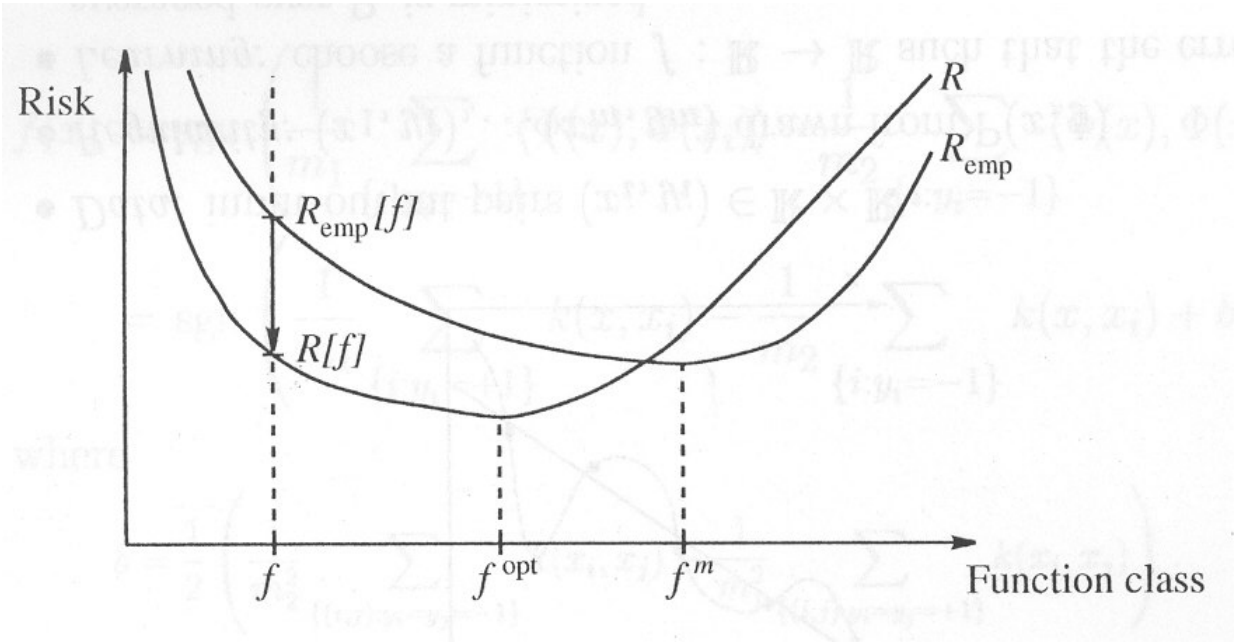
Consistency of Learning

- Given infinitum of data, can we learn any pattern?
- Law of Large Numbers: Sample mean converges to the expectation as the sample size goes to infinity. Hence, $R_n(g) \rightarrow R(g)$ as $n \rightarrow \infty$.
- Is this enough? Does this imply that ERM will result in the optimum as the data size approaches infinity? That is, is ERM always *consistent*?

➢ No.

Basic Concepts

Consistency of Learning



Ø Need a uniform version of the law of large numbers over all functions in the model.

Basic Concepts

Restriction on Function Class

- SLT take into account of the *capacity /simplicity /complexity* of the function class.
- The capacity plays a key role in the derivation of the main theorems of SLT.
- It turns out that we *must* restrict the function class in the uniform version of the law of the large numbers.
- In other words, for consistency, the function class must be restricted.

- Cf. Bayesian View: Put prior distribution on functions.

Basic Concepts

Structural Risk Minimization (SRM)

- Instead of just one model, consider a sequence of models $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \ldots$
- Minimize empirical risk in each model.
- Minimize the penalized empirical risk

$$\min_d \left(\inf_{g \in \mathcal{G}_d} R_n(g) + \text{penalty}(d) \right)$$

  - penalty(d) gives preference to models where estimation error is small.
  - penalty(d) also measures the capacity of the model.

Basic Concepts
Regularization

- A large model $\mathcal{G}$ (possibly dense).
- Regularizer $\|g\|$.
- Then minimize the regularized empirical risk

$$\inf_{g \in \mathcal{G}} (R_n(g) + \lambda \|g\|^2)$$

- Need to choose an optimal trade-off lambda
- Most methods including SRM can be viewed as regularization methods.
- Originally invented in mathematics to deal with ill-posed problems.

Measure of complexity
Growth Function

- Given the data D, let

  $\mathcal{G}_D=\{(g(X_1),\ldots,g(X_n) \mid g\in \mathcal{G}\}$

  – As far as D is concerned, $\mathcal{G}_D$ measures the effective size/classification power of $\mathcal{G}$.

- Define the growth function

  $S_{\mathcal{G}}(n)=\sup_{D \text{ of size } n} |\mathcal{G}_D|$

➢ Growth function is a measure of size/complexity, but it's hard to deal with.

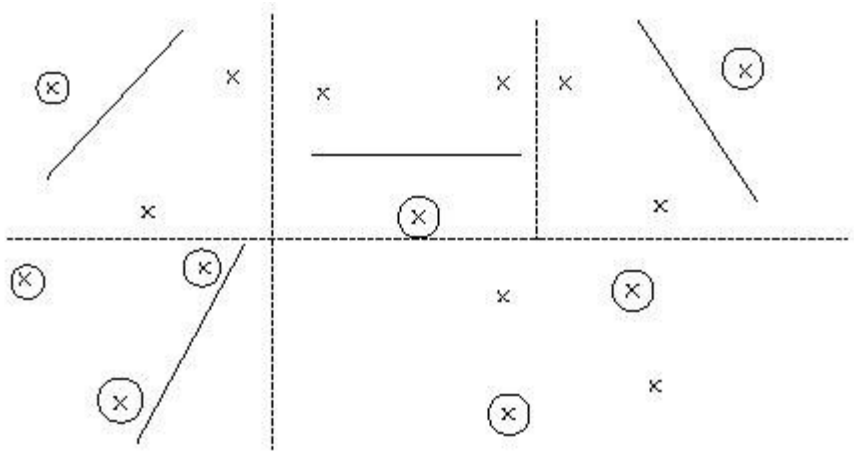
Measure of complexity

VC Dimension

- Note that $S_{\mathcal{G}}(n) \leq 2^n$.
- When $S_{\mathcal{G}}(n)=2^n$, $\mathcal{G}$ can generates any classification on (some set of) n points. In other words, $\mathcal{G}$ shatters n points.
- The VC dimension (after Vapnik-Chervonenkis) is defined as the largest n such that $S_{\mathcal{G}}(n)=2^n$.
  - It is a simplest measure of classifier complexity/capacity.
  - VC dim=n *doesn't* mean that $\mathcal{G}$ can shatter every data set of size n.
  - VC dim=n does mean that $\mathcal{G}$ can shatter *some* data set of size n.

VC Dimension
Example 1

- In $\mathbb{R}^d$, VC dimension of {all hyperplanes} is d+1.



  - We can prove that for any d+1 points *in general position* we can find hyperplanes shattering them.
  - However for d+2 points, hyperplanes cannot shatter them.

Ø Note that hyperplanes are given by $a_1x_1+\ldots+a_dx_d+a_0=0$.

Ø VC dimension == number of parameters?

VC Dimension
Example 2

- Let $\mathcal{G}$={sgn(sin(tx)) | t$\in\mathbb{R}$}, $\mathcal{X}=\mathbb{R}$.
- We can prove VC-dim($\mathcal{G}$)=∞, even though $\mathcal{G}$ is a one-parameter family!
  - The points $x_1=10^{-1}$, …, $x_n=10^{-n}$ for any integer m>0 can be shattered by choosing $t=\pi(\sum_i (1-y_i)10^i/2 + 1)$.

➢ Thus VC dimension is not just the number of parameters.

VC Bound

- One of the main theorems of SLT
- Vapnik-Chervonenkis Theorem

  Let $\mathcal{G}$ be a class of functions with *finite* VC dimension h. Then with probability at least 1-δ,

$$\text{For all } g \in \mathcal{G}, \quad R(g) \leq R_n(g) + \sqrt{\frac{h \log\left(\frac{2en}{h}\right) + \log(4/\delta)}{8n}}.$$

- So the error is of order $(h \log n / n)^{0.5}$.
- This bound is distribution-free, i.e., true for all distribution.

Use of SLT

- The bounds given by SLT are mostly useless for practical purpose.
  - Anything bigger than .5 is useless.
  - SLT bounds are either too loose because it's distribution-free, or too complicated to be explicitly computed.
- Best use of these bounds are to use them *qualitatively,* not quantitatively.
  - These bounds are robust.
  - Although loose, these bounds reflects the behavior of the error.
  - SVM, one of the strongest classifiers currently known, is conceived this way.

Computational Learning Theory; The Tradeoff between *Computational Complexity* and *Statistical Soundness*

Introduction

The complexity of leaning is measured mainly along two axis: Information and computation.

Information complexity is concerned with the generalization performance of learning;

How many training examples are needed?

How fast do learner's estimate converge to the true population parameters? Etc.

The Computational complexity concerns the computation applied to the training data to extract from it learner's predictions.

It seems that when an algorithm improves with respect to one

of these measures it deteriorates with respect to the other.

Outline of this Talk

1. Some background.

2. Survey of recent pessimistic *computational hardness* results .

3. A discussion of three different directions for solutions:
   a. The *Support Vector Machines* approach.
   b. The *Boosting* approach (an agnostic learning variant).
   c. Algorithms that are efficient for `well behaved' inputs.

The Label Prediction Problem

Formal Definition	Example
Given some domain set X	*Files of personal data of (car) drivers.*
A sample S of labeled members of X is generated by some (unknown) distribution	*Files in a sample are labeled according to driver's involvement in car accident.*
For a next point x, predict its label	*Will the current driver undergo an accident?*

Empirical Risk Minimization Paradigm

- Choose a *Hypothesis Class* H of subsets of X.
- For an input sample S, find some h in H that fits S well.
- For a new point x, predict a label according to its membership in h.

The Mathematical Justification

Assume both a training sample S and the test point (x,l)

are generated by the same distribution over $X \times \{0,1\}$ then,

If H is not too rich (e.g., has small VC-dimension),

for every h in H,

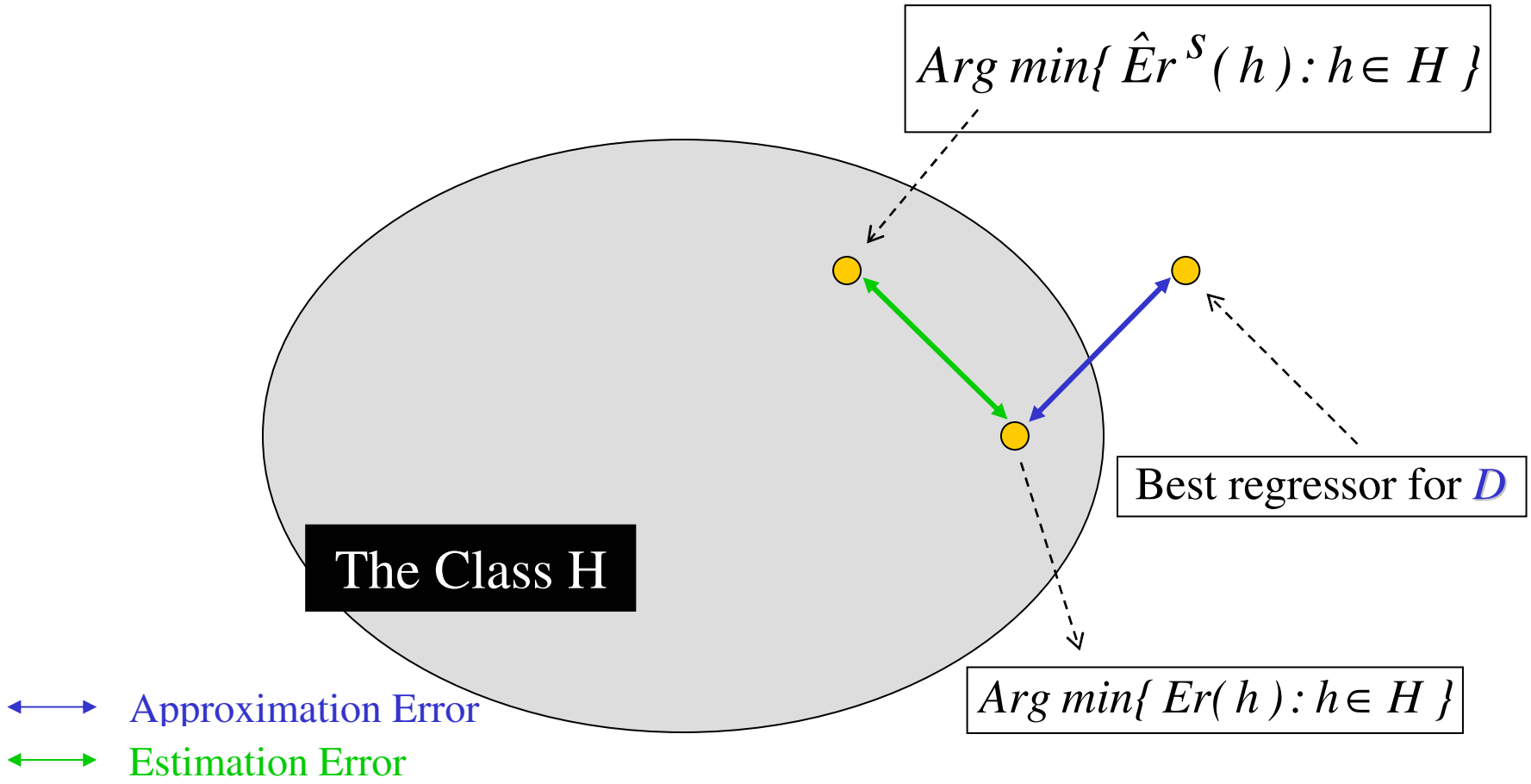
the empirical success ratio of h on the sample S

is a good estimate of its probability of success on the new x.

Two Basic Competing Models

PAC framework	Agnostic framework
Sample labels are consistent with some h in H	*No prior restriction on the sample labels*
Learner's hypothesis required to meet *absolute* upper bound on its error	*The required upper bound on the hypothesis error is only* relative *(to the best hypothesis in the class)*

The Types of Errors to be Considered

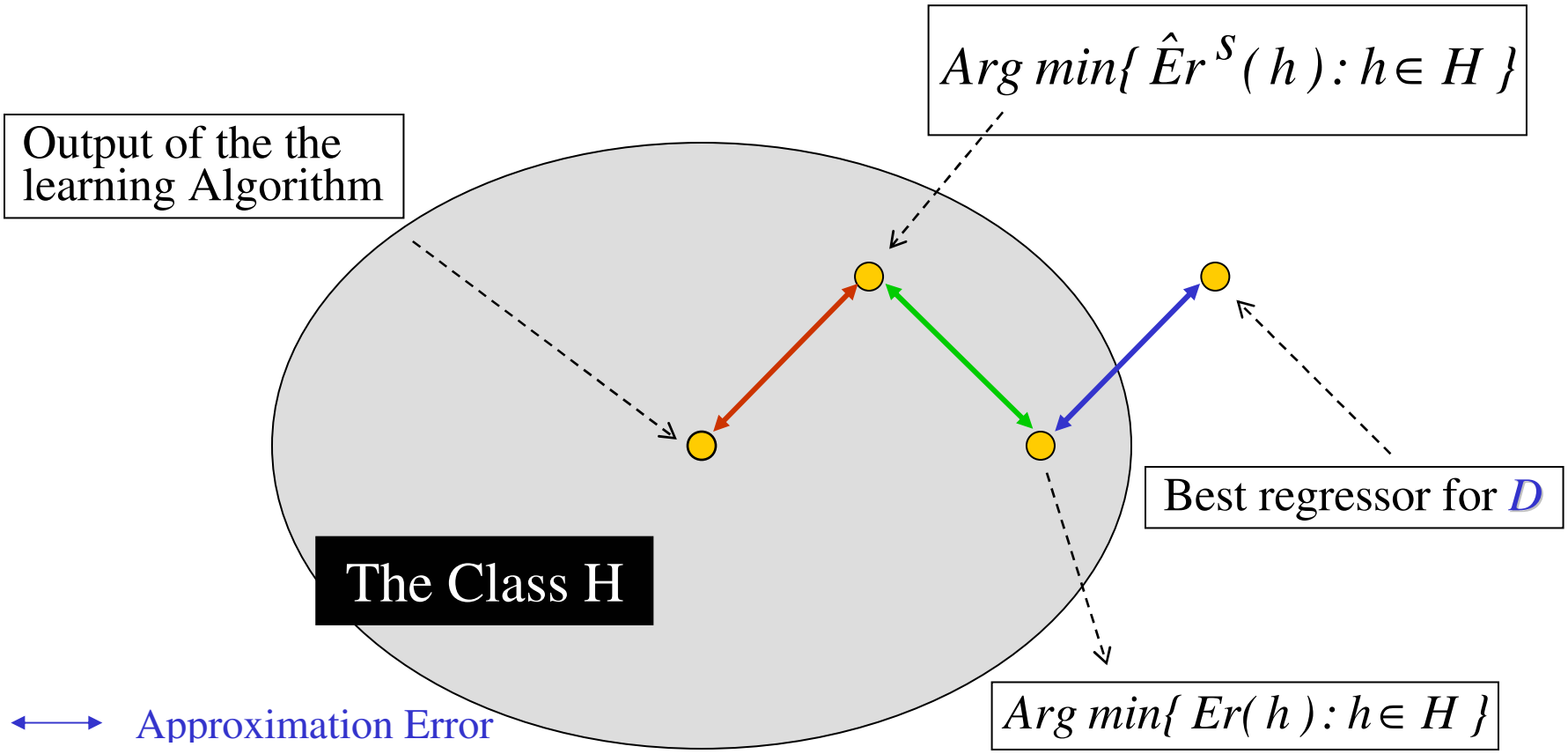


The Computational Problem

Given a class H of subsets of $\mathbb{R}^n$

- **Input**: A finite set of $\{0, 1\}$-labeled points S in $\mathbb{R}^n$.
- **Output**: Some 'hypothesis' function h in H that maximizes the number of correctly classified points of S.

The Types of Errors to be Considered



Hardness-of-Approximation Results

For each of the following classes, approximating the best agreement rate for *h in H* (on a given input sample *S*) up to some constant ratio, is *NP-hard*:

- Monomials
- Monotone Monomials
- Half-spaces
- Balls
- Axis aligned Rectangles
- Threshold NN's with constant 1st-layer width

BD-Eiron-Long

Bartlett- BD

Gaps in Our Knowledge

- The additive constants in the hardness-of-approximation results are *1%-2%.*

  They do not rule out efficient algorithms achieving, say, *90%(optimal success rate).*

- However, currently, there are no efficient algorithm whose success-rate guarantees are significantly above *50%*.

Three solution paradigms

- Boosting (adapted to the Agnostic setting).
- Kernel-Based methods
  (including Support Vectors Machines).
- Data Dependent Success Approximation
  Algorithms.

Boosting Idea: Combine Weak Learners

An algorithm is a β *weak learner* for a class H

if on every H-labeled weighted sample S,
it outputs some h in H

so that $Er_S(h) < \frac{1}{2} - \beta$

Note: The existence of **computationally efficient** weak learners is not ruled out by the above hardness results.

Boosting Solution: the Basic Result

[Schapire '89, Freund '90]

Having access to a computationally efficient weak learner,

for a P-random H-sample S, and parameters $\varepsilon, \delta > 0$,

the boosting algorithm efficiently finds some h in $Co(H)$

so that

$$Er_P(h) < \varepsilon, \text{ with prob. } > 1-\delta.$$

Note: Since h may be outside H, there is no conflict with the computational hardness results.

The Boosting Solution in Practice

- The boosting approach was embraced by practitioners of Machine Learning and applied, quite successfully, to a wide variety of real-life problems.

Theoretical Problems with the The Boosting Solution

- The boosting results assume that the input sample labeling is consistent with some function in H (the PAC framework assumption).
  In practice this is never the case.

- The boosting algorithm's success is based on having access to an efficient weak learner –
  for most classes, no such learner is known to exist.

Attempt to Recover Boosting Theory

Can one settle for weaker, realistic, assumptions?

Agnostic weak learners :

an algorithm is a β weak agnostic learner for H,
if for every labeled sample S it finds h in H s.t.

$$Er_S(h) < Er_S(Opt(H)) + \beta$$

Agnostic weak learners always exist

Revised Boosting Solution

[B-D, Long, Mansour (2001)]

There is a computationally efficient *algorithm that, having access to a* β *weak* agnostic *learner, finds* h *in* $Co(H)$ *s.t.*

$$Er_P(h) < c\, Er_P(Opt(H))^{c'}$$

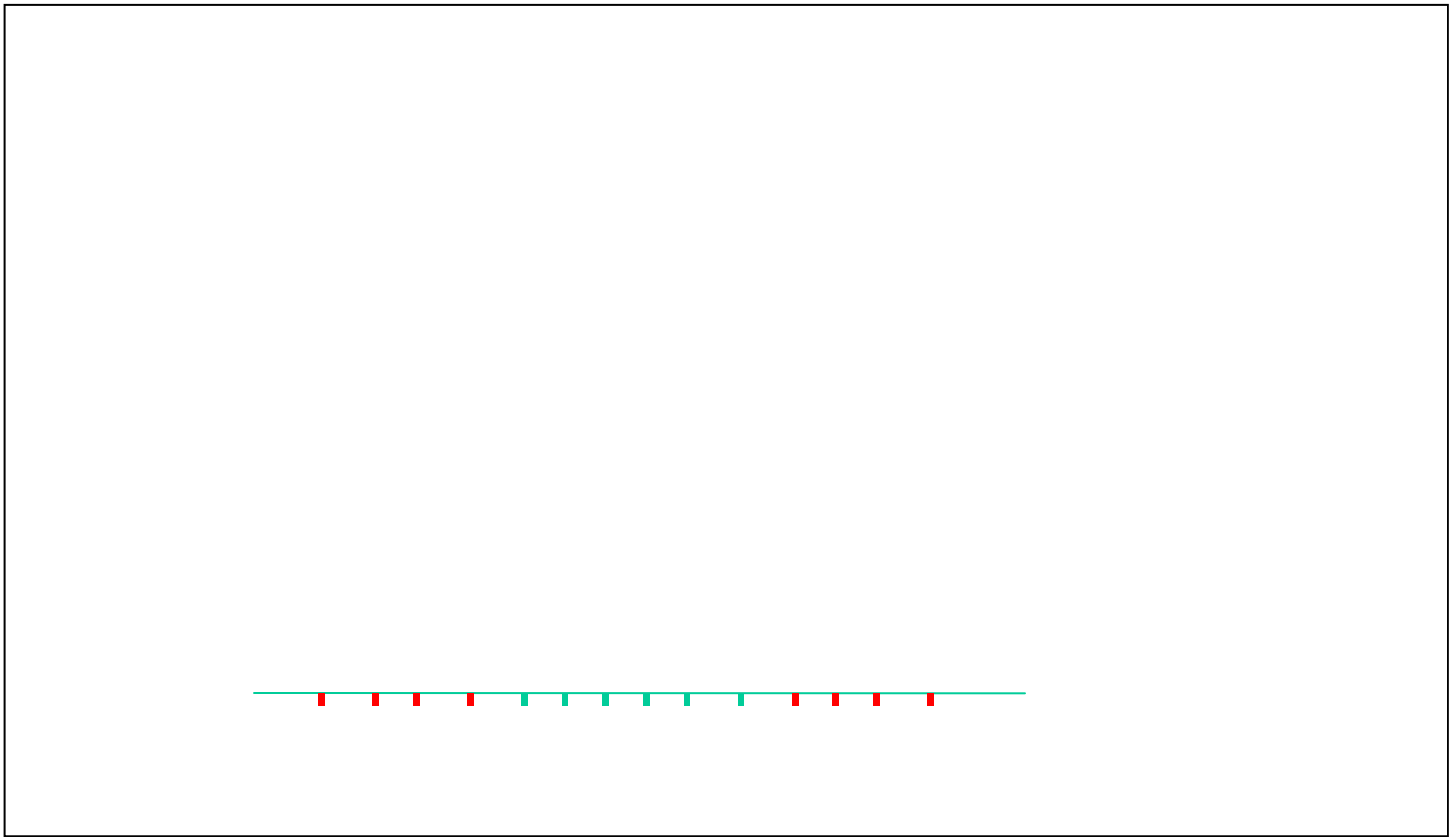
(Where c *and* c' *are constants depending only on* β*)*

The SVM Paradigm

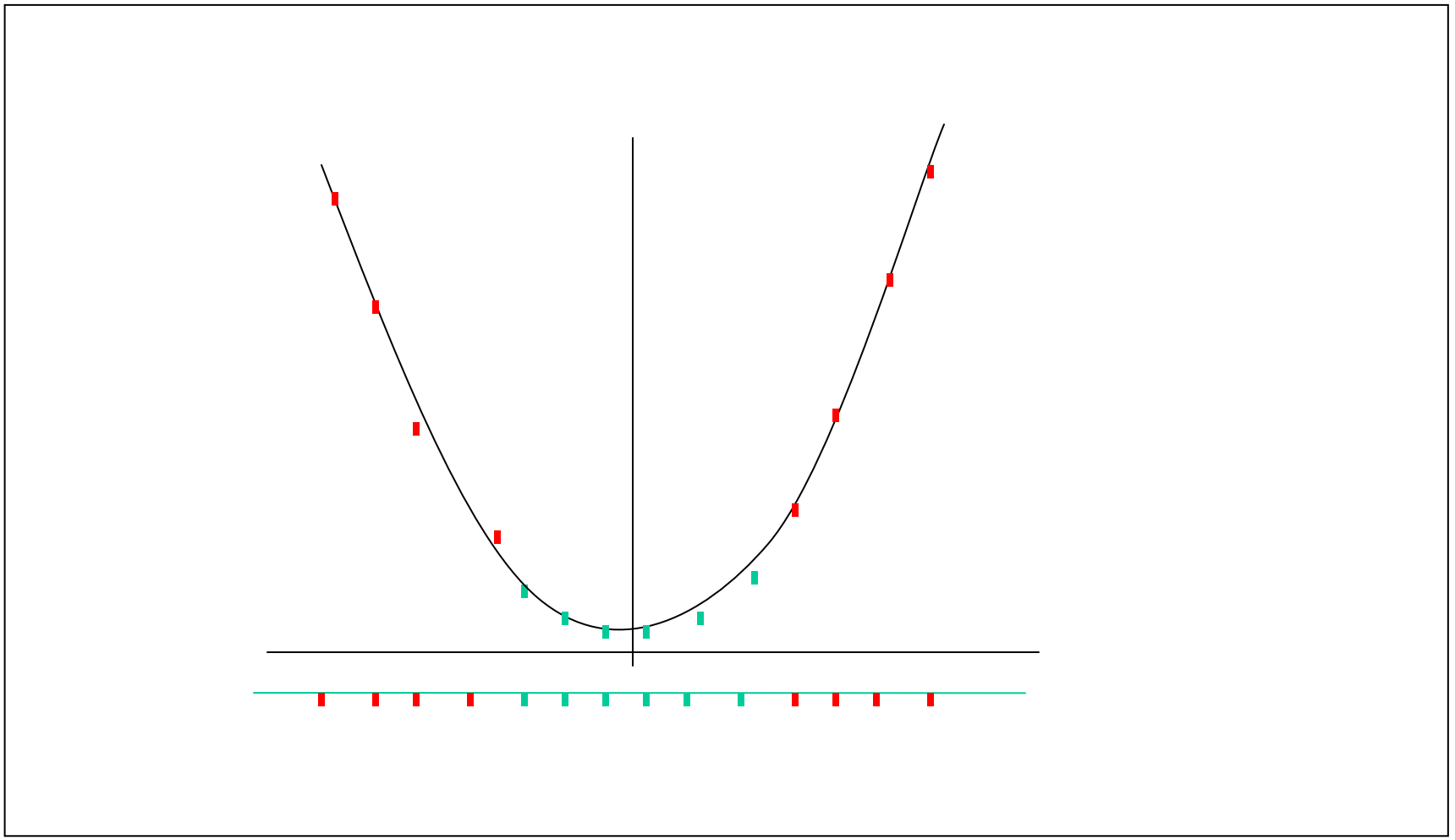
- Choose an *Embedding* of the domain X into some high dimensional Euclidean space, so that the data sample becomes (almost) linearly separable.
- Find a large-margin data-separating hyperplane in this image space, and use it for prediction.

Important gain: *When the data is separable, finding such a hyperplane is computationally feasible.*

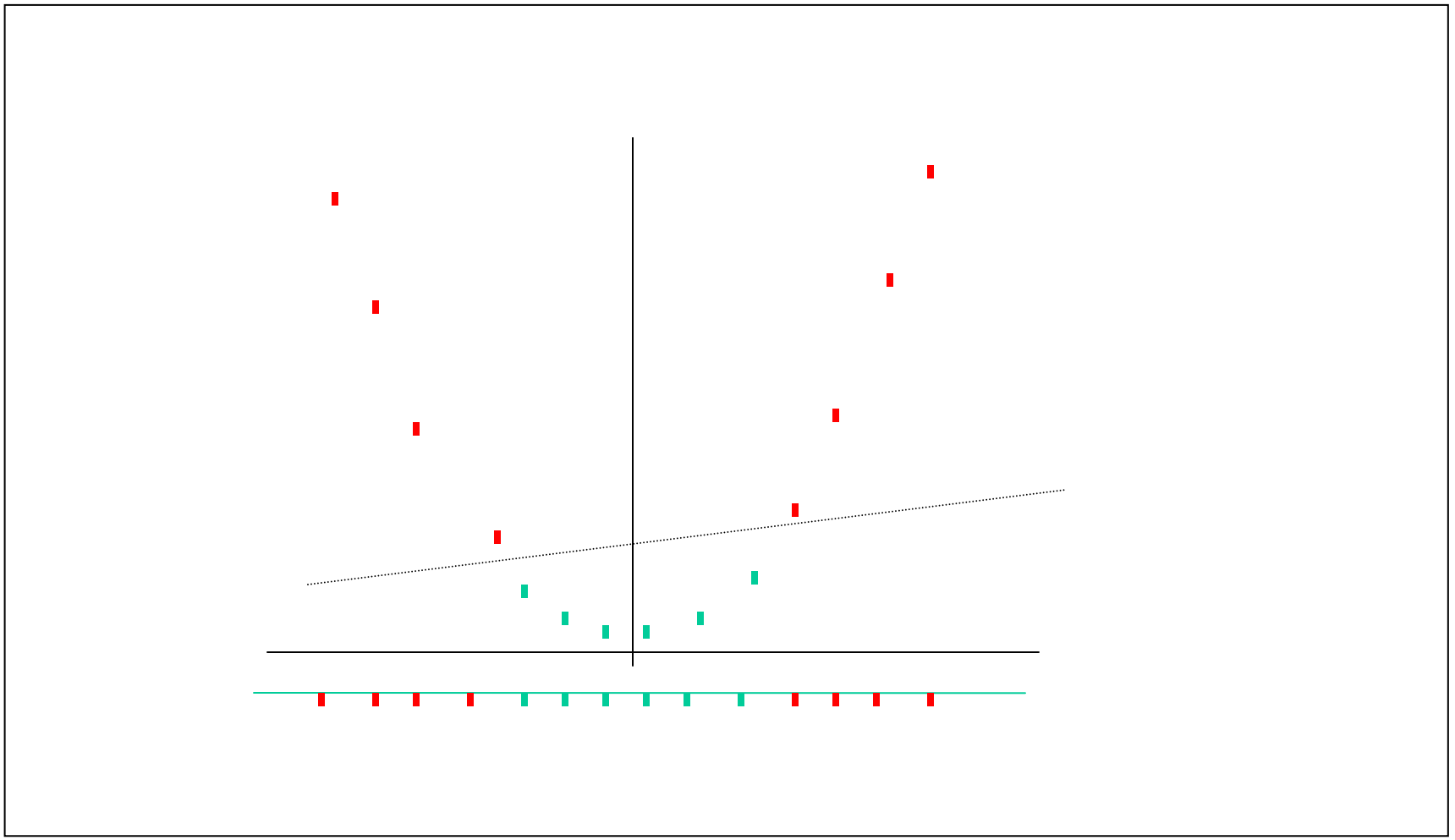
The SVM Idea: an Example



The SVM Idea: an Example



The SVM Idea: an Example



The SVM Solution in Practice

- The SVM approach is embraced by practitioners of Machine Learning and applied, very successfully, to a wide variety of real-life problems.

A Potential Problem: Generalization

- **VC-dimension bounds**: The VC-dimension of the class of half-spaces in R^n is $n+1$.

Can we guarantee low dimension of the embeddings range?

- **Margin bounds**: Regardless of the Euclidean dimension, generalization can bounded as a function of the margins of the hypothesis hyperplane.

Can one guarantee the existence of a large-margin separation?

An Inherent Limitation of SVM 's

[B-D, Eiron, Simon (2001)]

- In "most" cases the data cannot be made separable unless the mapping is into dimension $\Omega(|X|)$.

  This happens even for classes of small VC-dimension.

- For "most" classes, *no* mapping for which concept-classified data becomes separable, has large margins.

In both cases the generalization bounds are lost!

A Proposal for a Third Solution: Data-Dependent Success Approximations

With an eye on the gap between the theoretical hardness-of-approximation results and the experimental success of practical learning algorithms,

we propose a new measure of the quality of an approximation algorithm

Data Dependent Approximation:

- The required success-rate is a function of the input data.

Data Dependent Success Definition for Half-spaces

A learning algorithm A is

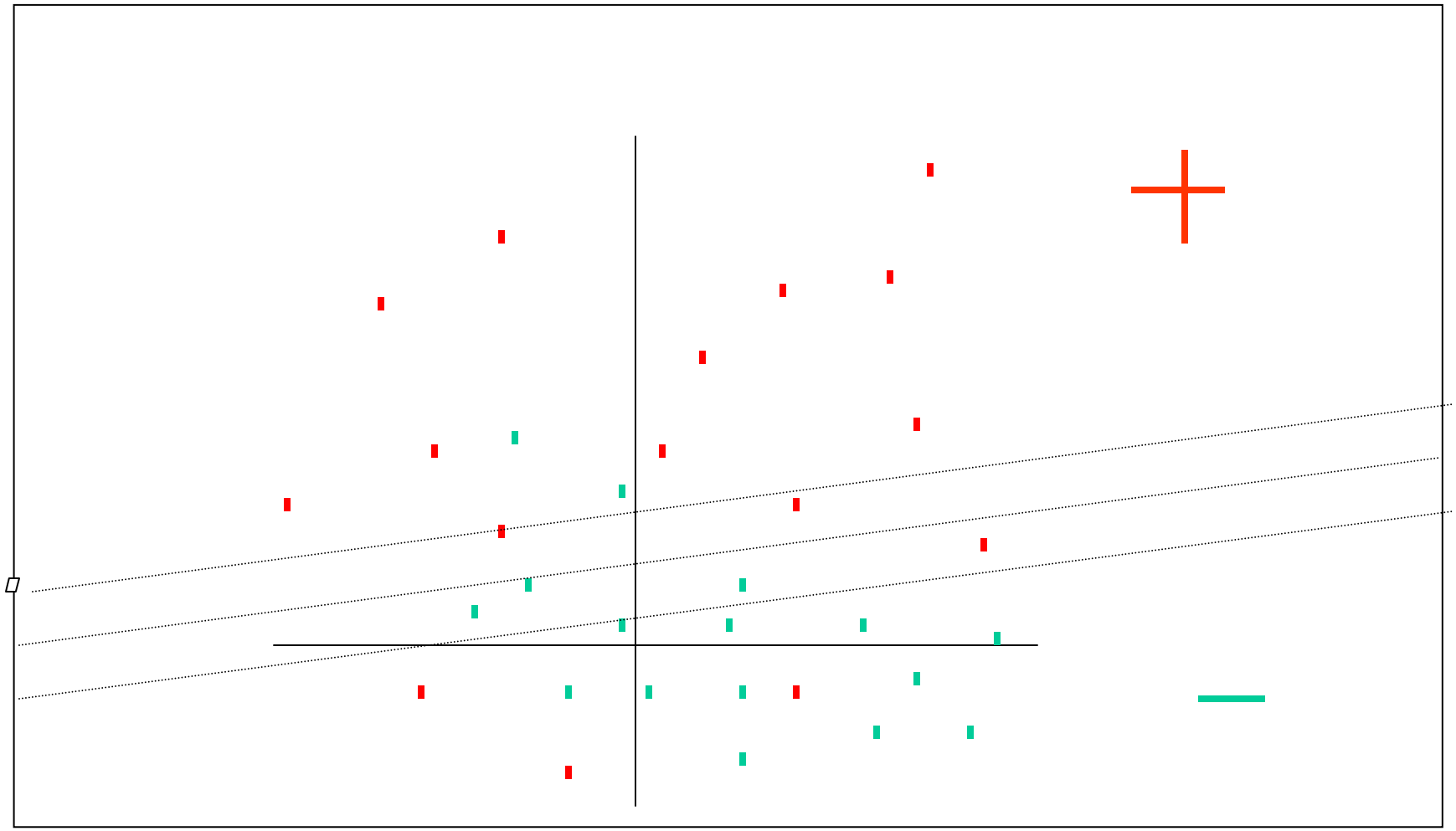
μ–*margin* successful

if, for every input $S \subseteq R^n \times \{0,1\}$,

$$|\{(x,y) \in S: A_{(S)}(x) = y\}| \geq |\{(x,y): h(x)=y \text{ and } d(h, x) > \mu\}|$$

for every half-space h.

Classifying with Margins



Some Intuition

- If there exist some optimal h which separates with generous margins, then a μ–*margin* algorithm must produce an optimal separator.

On the other hand,

- If every good separator can be degraded by small perturbations, then a μ–*margin* algorithm can settle for a hypothesis that is far from optimal.

Main Positive Result

- *For every positive μ, there is an efficient μ–margin algorithm.*

That is, the algorithm that classifies correctly as many input points as any half-space can classify correctly with margin μ.

A Crisp Threshold Phenomena

The positive result

- For every positive μ, there is a μ - margin algorithm whose running time is polynomial in $|S|$ and n.

A Complementing Hardness Result

- Unless P = NP, no algorithm can do this in time polynomial in $1/\mu$ (and in $|S|$ and n).

Some Obvious Open Questions

- *Is there a parameter that can be used to ensure good generalization for Kernel–Based (SVM-like) methods?*
- Are there efficient Agnostic Weak Learners *for potent hypothesis classes?*
- *Is there an inherent trade-off between the generalization ability and the computational complexity of algorithms?*

THE END