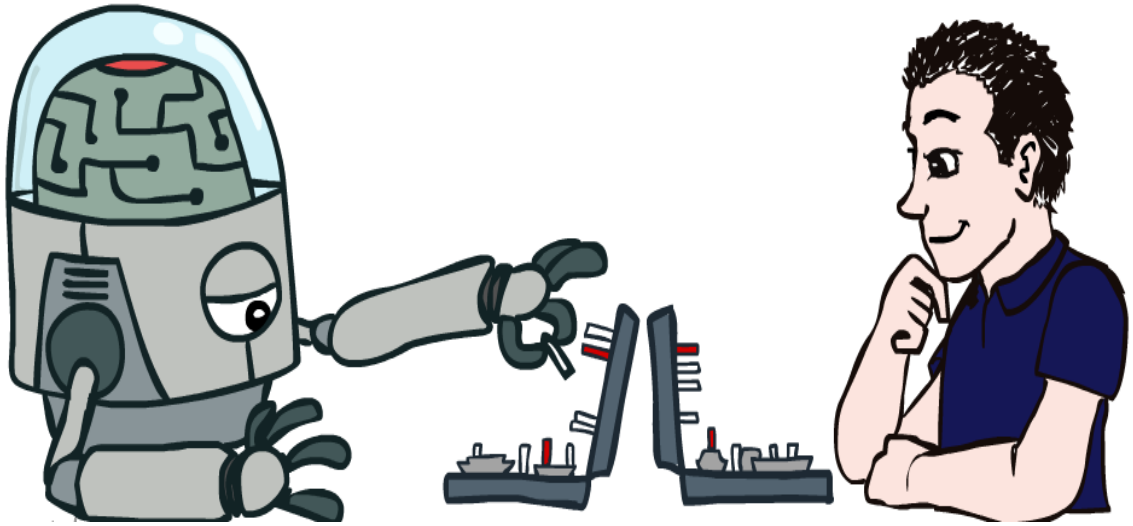
ARTIFICIAL INTELLIGENCE- CS411

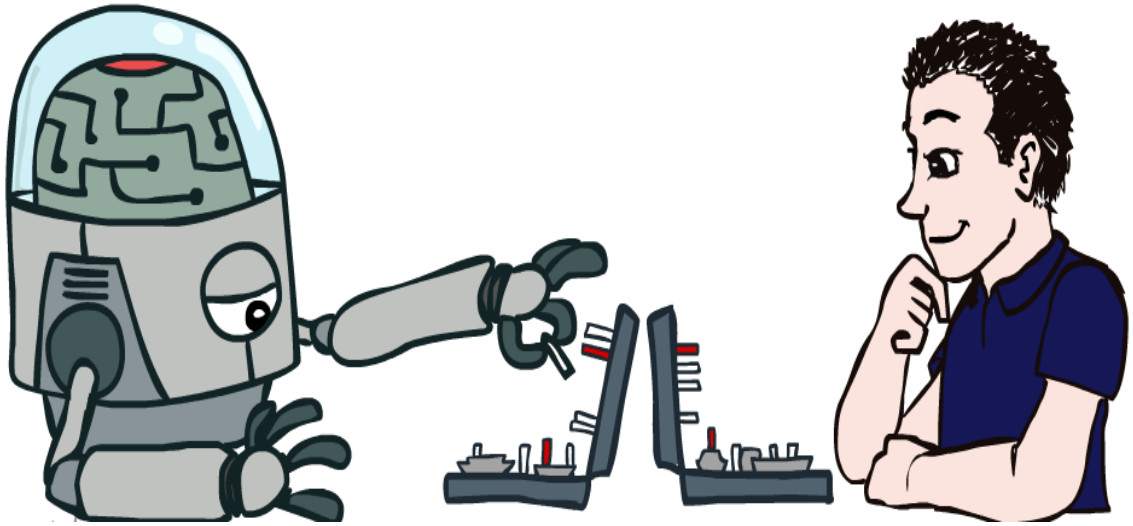
Dr. Abdulelah Algosaibi



Credit: Dr. Alaa Sagheer

ARTIFICIAL INTELLIGENCE- CS411

Dr. Abdulelah Algosaibi



Credit: Dr. Alaa Sagheer

Acknowledgement

These slides were assembled by **Credit Prof. Alaa Sagheer**, with grateful acknowledgement of the many other professors who made their course materials freely available online.

8. Learning

Fundamentals of Learning

Lecture I

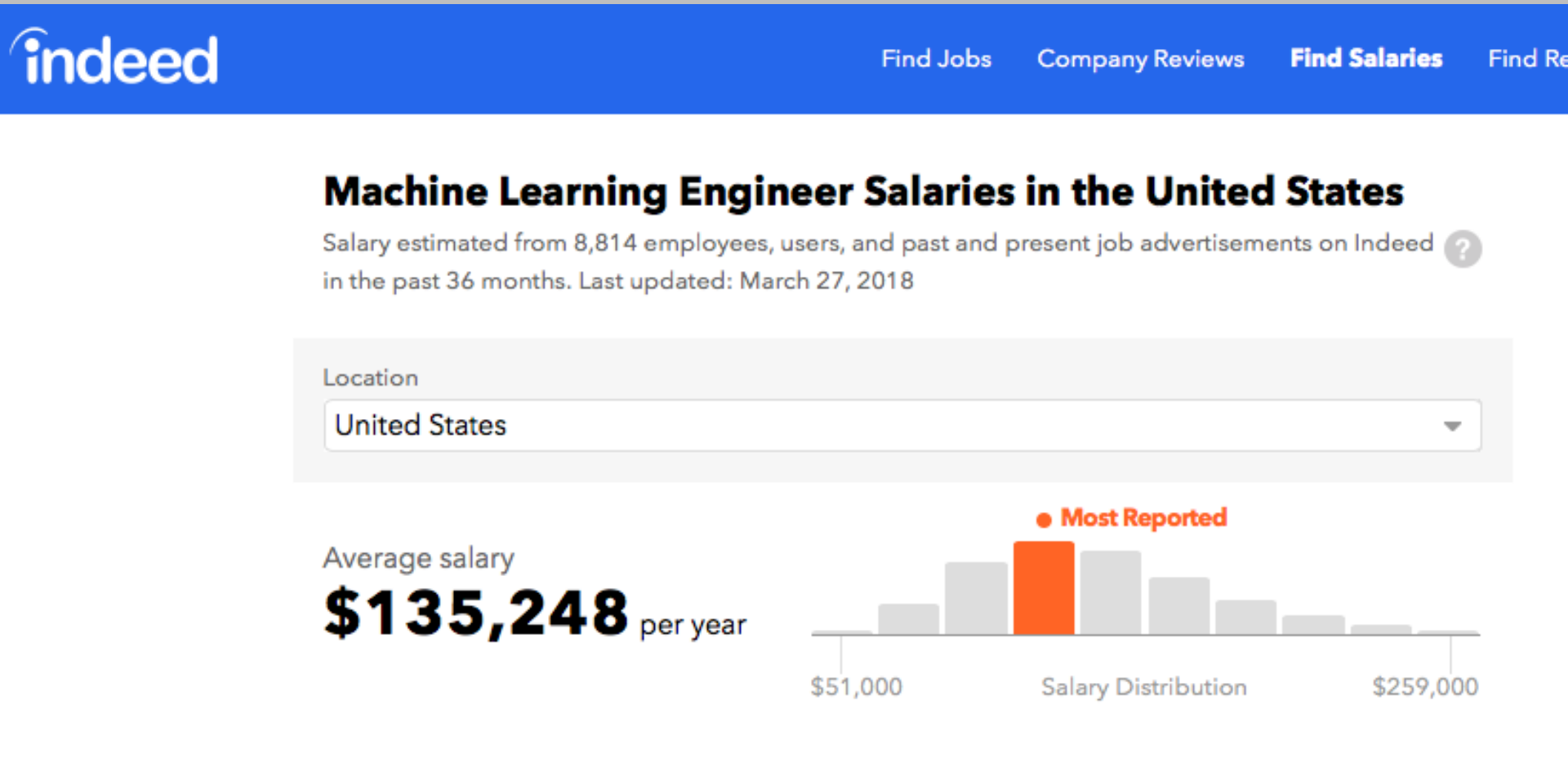
Why Should I do my Master in Machine Learning?

"Hiring a Machine Learning engineer or Data Scientist in Silicon Valley is becoming like hiring a professional athlete due to high demand by the tech giants. That's how demanding it is".

The New York Times

Why Should I do my Master in Machine Learning?

Machine Learning has become one of the most demanding skills in the workforce today, with the average salary in US reaching in average to $135,248 per year.



Introduction

- **What is Machine Learning?**
- **What is Learning?**

"Learning denotes **changes** in the system that are adaptive in the sense that they enable the system to do the same task more effectively the next time."

Progressing & Improving

- **We can easily extend this definition easily to our AI systems:**

"Machine learning denotes **automated changes** in an AI system that are adaptive in the sense that they enable the system to do the same task more effectively the next time."

- The details of Machine Learning depend on the underlying knowledge representations, e.g. learning in neural networks will be very different to learning in rule based systems or learning decision trees etc.

Introduction

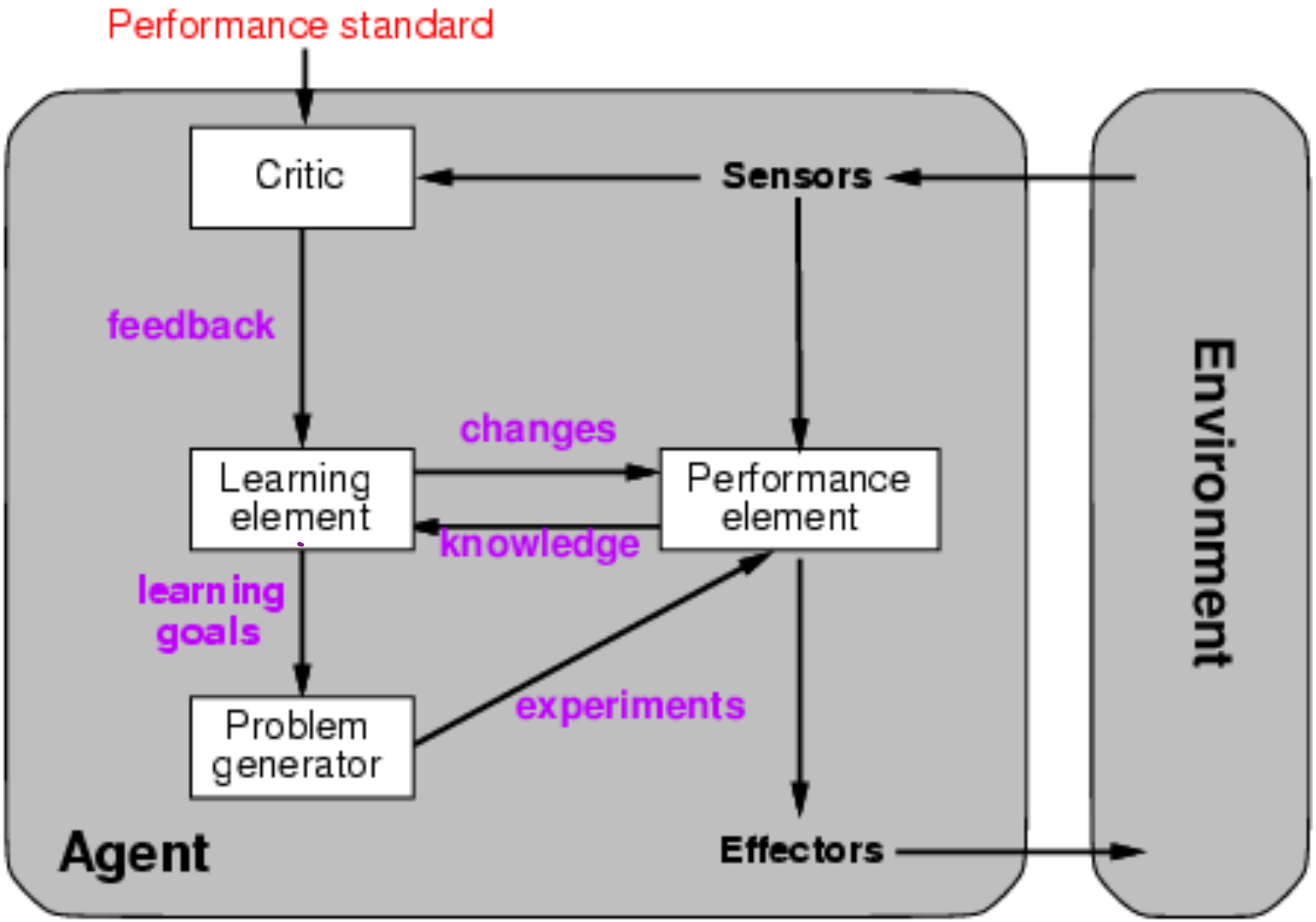
- Here, we will describe agents that can improve their behavior through efficient study of their own experiences (Learning modifies the agent's decision mechanisms to improve performance).

- The idea behind learning is that percepts should be used not only for acting, but also for improving the agent's ability to act in the future.

- Learning is essential for unknown environments, i.e., when designer lacks omniscience.

- This chapter describes different **learning approaches** such as learning from examples, inductive learning, and learning using decision trees

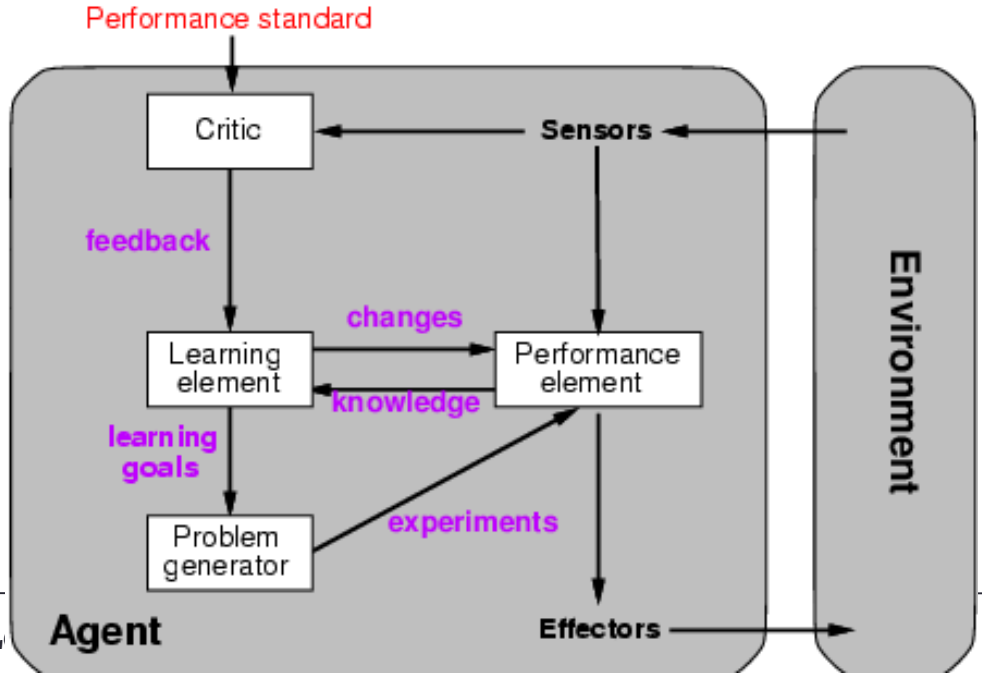
Learning agents



Learning agents

- ***Performance element***: selecting actions based on percepts
- ***Learning element***: making improvements in performance element

  (Note: Machine learning researchers have come up with a large variety of learning elements)
- ***Critic***: tell the learning element how well the agent is doing with respect to a fixed performance standard
  - It is a necessary element (the percept itself has no meaning for agent's success)
- ***Problem generator***: suggest actions that will lead to new and informative experiences
  - In case of exploring and doing suboptimal actions for the short run, this may be useful in the long run.



Types of Learning

The field of machine learning usually distinguishes THREE types based on amount of inference the system has to perform on its training data. In increasing order we have:

1. **Supervised learning**: correct answers for each example
2. **Unsupervised learning**: correct answers not given
3. **Reinforcement learning**: agents ought to take actions

Types of Learning

Fully observed action

Supervised Learning: Correct answers for each example are given

A teacher provides the correct output value of the examples.

- For fully observable environments, it will always be the case that an agent can observe the effects of its actions and hence can use supervised learning methods to learn to predict them.

- For partially observable environments, the problem is more difficult, because the immediate effects might be invisible.

essentially know the magnitude of the agent actions Not good in Partial environments

Types of Learning

not complete.

Unsupervised Learning: Correct answers not given

- Involving learning patterns in the input when no specific output values are supplied. Slowly learning the Best way.
- For example, a taxi agent might gradually develop a concept of "good traffic days" and "bad traffic days" without ever being given labeled examples of each.
- A purely unsupervised learning agent cannot learn what to do, because it has no information as to what constitutes a correct action or a desirable state.

Essentially Humans and How they develope their experiences.

Types of Learning

Reinforcement learning: correct answers not given

- Rather than being told what to do by a teacher, a reinforcement learning agent must learn from reinforcement!

- For example, a hefty wind for the car in front gives the agent some indication that its behavior is undesirable.

- Reinforcement learning typically includes the sub problem of learning how the environment works.

Learning what to do BY Learning what not to do

Supervised Learning:

where the model predicts an output variable (dependent variable) Based on the input variables (& Independent variables).

The agent is trained on labeled data meaning Input - output are provided Before hand.

examples: Decision trees, Support vector machine, neural network

Unsupervised Learning

Is a type of ML where the model learns to find patterns in the input data without outputs.

The goal is to discover underlying structure or Relationship

Example: Principal Component analysis, GANS

Reinforcement Learning:

This ML to takes actions .. to maximize a reward signal

It has no explicit Input-output pairs. And must learn from trial & error By taking actions and receiving feedback.

Examples: Robotics, Auto no mous driving

Differences

Supervised:

The Algorithm is given labelled input-output pairs and learn to predict the output variable for new inputs

Unsupervised:

The algorithm is given only input Data and Learns to find patterns.

Reinforcement:

The Algorithm learns to take Actions to Maximize a reward signal over time.

Learning agents

❖ **The availability of prior knowledge:**

- There is no doubt that prior-knowledge can help enormously in learning
- However, the majority of learning research in AI studied the case in which the agent begins with no knowledge at all about what it is trying to learn. It has access only to the examples presented by its experience, by no means this is the general case.
- Most human learning takes place in the context of a good deal of background knowledge. Some psychologists and linguists claim that even newborn babies exhibit knowledge of the world.

The Need for Learning

- We have seen that extracting knowledge from human experts, and converting it into a form useable by the inference engine of an expert system or other computer system, is an arduous and labor intensive process.

- For most large scale AI systems, it is much more efficient to give the system enough knowledge to get it started, and then leave it to learn the rest for itself. We may even end up with a system that learns to be better than a human expert.
  - *See Recent Advances in Medicine.*

- The ***general learning approach*** is to generate potential improvements, test them, and only use those that work well.

8. Learning

Inductive Learning

Lecture II

Inductive Learning

Simply..Learn a function from examples

- An algorithm for deterministic supervised learning,

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$f(1^2) = 1$
$f(2^2) = 4$
$f(3^2) = 9$
$f(4^2) = 16$
$f(5) = ?$ 25

Then Induction

Inductive Learning

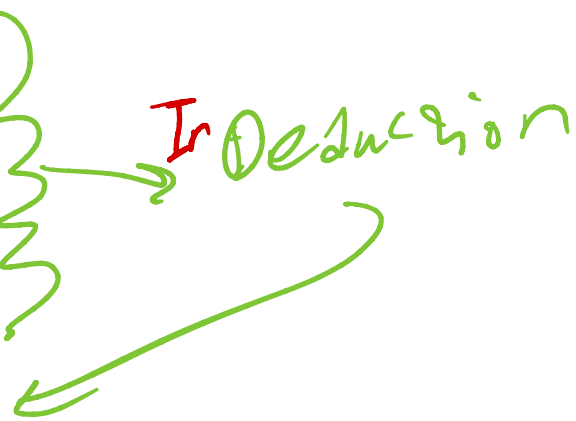
Simply..Learn a function from examples

- An algorithm for deterministic supervised learning,

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$f(1) = 1$
$f(2) = 4$
$f(3) = 9$
$f(4) = 16$
$f(5) = 25$



Inductive Learning

Simply..Learn a function from examples

- An algorithm for deterministic supervised learning,

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

f(Romeo and Juliet)	=	Shakespeare
f(Tom Sawyer)	=	Twain
f(Macbeth)	=	Shakespeare
f(Huckleberry Finn)	=	Twain
f(Othello)	=	?

Inductive Learning

Simply..Learn a function from examples

- An algorithm for deterministic supervised learning,

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

f(Romeo and Juliet) = Shakespeare
f(Tom Sawyer) = Twain
f(Macbeth) = Shakespeare
f(Huckleberry Finn) = Twain
f(Othello) = Shakespeare

Inductive Learning

Original Function

An unknown function $f : D_f \rightarrow R_f$

Inductive Learning

Original Function

An unknown function $f: D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Inductive Learning

Original Function

An unknown function $f: D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h: D_f \rightarrow R_f$

Inductive Learning

Original Function

An unknown function $f: D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h: D_f \rightarrow R_f$

Learning Goal

Pick a hypothesis h as close to f as possible

Inductive Learning

Simply..Learn a function from examples

- An algorithm for deterministic supervised learning,
- Given as input, the correct value of an unknown function for particular inputs and must try to recover the unknown function (or something close to it),
- More formally, we say that an **example** is a pair (x, f(x)), where x is the input and f(x) is the output of the function applied to x, or as follows:

f is the **target function**. An **example** is a pair (x, f(x))

Problem:

given a **training set** of examples

find a **hypothesis** h

such that $h \approx f$

Inductive Learning

f is the **target function**. An **example** is a pair (x, f(x))

Problem:

given a **training set** of examples

find a **hypothesis** ***h***

such that $h \approx f$

- This is a highly simplified model of real learning. Why?
  - Ignore the prior knowledge,
  - Assume examples are given
- Learning is a difficult. Why?
  - It is not easy to tell whether any particular ***h*** is a good approximation of ***f***
- A good hypothesis will generalize well-will predict **unseen examples** correctly. This is the fundamental problem of induction.

Inductive Learning

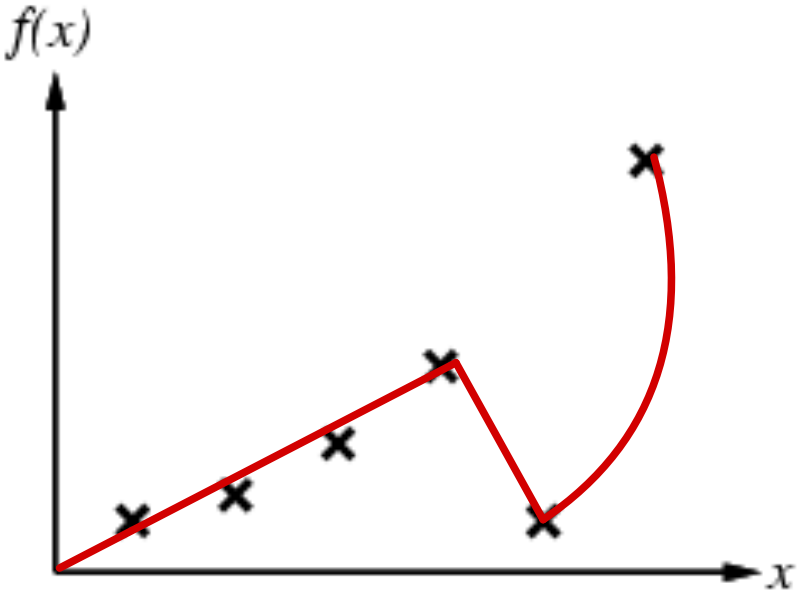
- Fitting a function of a single variable to some data points. The examples are (x, f(x)) pairs, where both x and $f(x)$ are real numbers.
- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on **all** examples)
- Many **hypotheses** possible:

The curve fitting of a set of polynomials of degree k, such as:

$3x^2 + 2$

$x^{17} - 4x^3$

and so on.

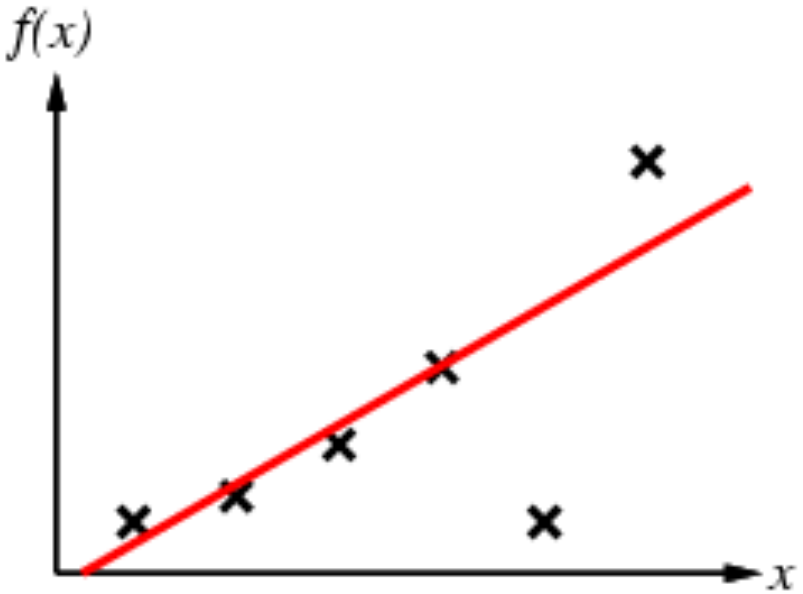


Inductive Learning

- Fitting a function of a single variable to some data points. The examples are (x, f(x)) pairs, where both x and $f(x)$ are real numbers.
- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on **all** examples)

- A Straight line (a polynomial with degree 1)

NO

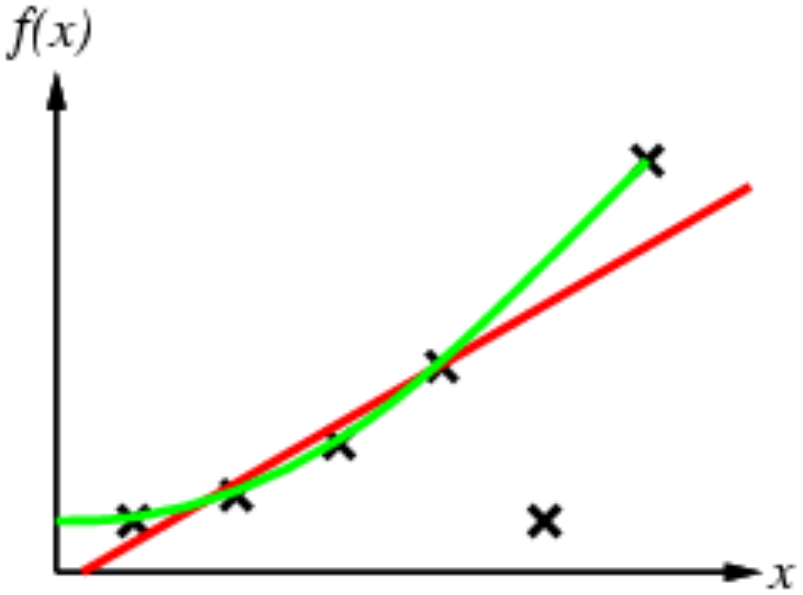


Inductive Learning

- Fitting a function of a single variable to some data points. The examples are (x, f(x)) pairs, where both x and $f(x)$ are real numbers.
- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on **all** examples)
-

❑ A high-degree polynomial

NO

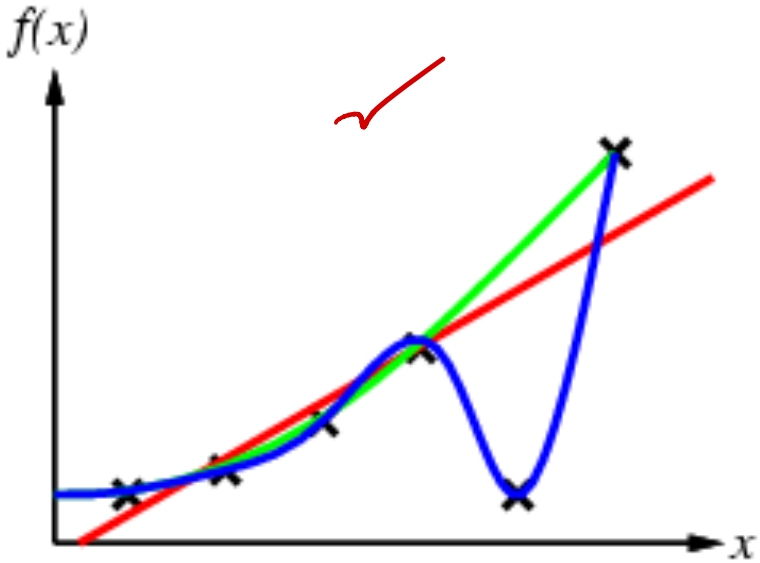


Inductive Learning

- Fitting a function of a single variable to some data points. The examples are (x, f(x)) pairs, where both x and $f(x)$ are real numbers.
- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on **all** examples)
-

❑ A high-degree polynomial

Consistent

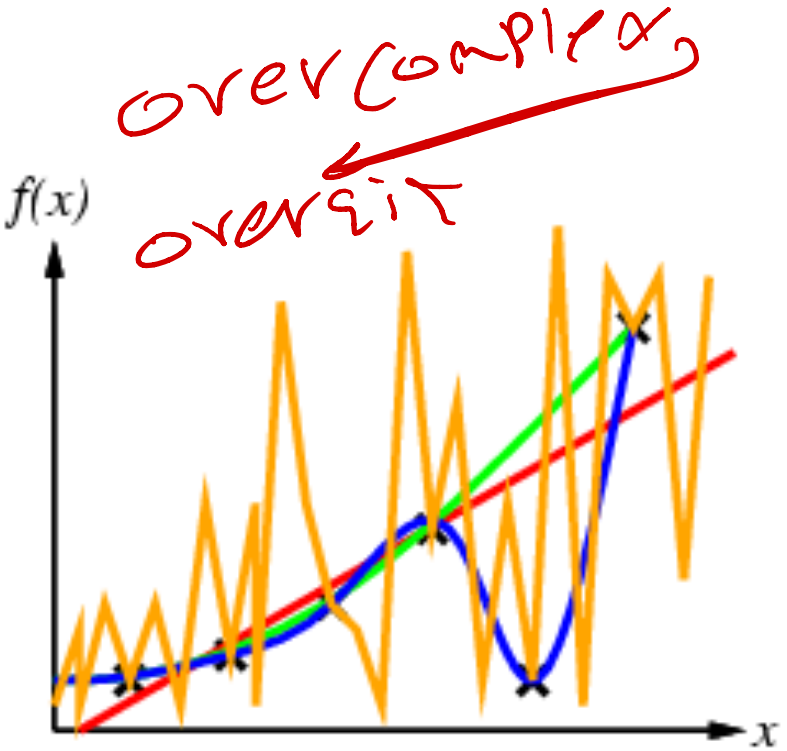


Inductive Learning

- If there are more than a consistent hypotheses! *How do we choose from among multiple consistent hypotheses?*

Goals

- Maximize number of examples where *h* agrees with *f*
- Minimize complexity of *h* function

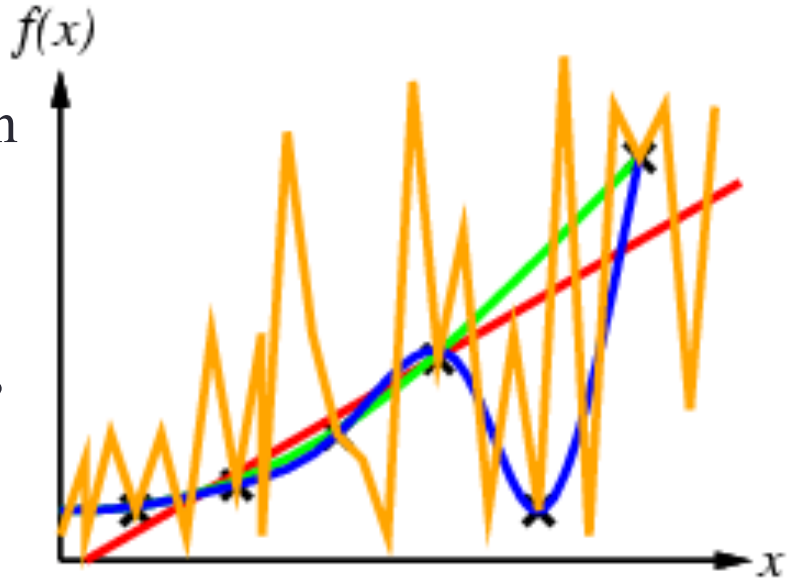


Typically a tradeoff between consistency and simplicity

Inductive Learning

- If there are more than a consistent hypotheses! *How do we choose from among multiple consistent hypotheses?*

- Ockham's razor: prefer the simplest hypothesis consistent with data,
- Because hypotheses that are no simpler than the data themselves are failing to extract any *pattern* from the data,
- Defining simplicity is not easy, but it seems reasonable to say that a degree-1 polynomial is simpler than a degree-12 polynomial.



Overfitting!

Definition

A hypothesis h has overfit the data if it uses a function that is more complex than necessary to explain the data

Examples:

$$f(1) = 2$$
$$f(2) = 4$$
$$f(3) = 6$$

Hypotheses:

$$h_1(x) = 2x$$
$$h_2(x) = x^3 - 6x^2 + 13x - 6$$

h_2 has probably overfit the data

... unless $f(x) = x^3 - 6x^2 + 13x - 6$

Lesson

Use as many parameters as we expect f to have, no more

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f: D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

- D_f = humans

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

- D_f = humans
- $R_f = \mathbb{R}$ (yearly income)

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

- D_f = humans
- $R_f = \mathbb{R}$ (yearly income)

Ex: Cancer Diagnosis

- D_f = animals

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

- D_f = humans
- $R_f = \mathbb{R}$ (yearly income)

Ex: Cancer Diagnosis

- D_f = animals
- $R_f = \{true, false\}$

Decomposing Domain Objects

Smoker identification:

$f(\text{John}) = \textit{true}$
$f(\text{Mary}) = \textit{false}$
$f(\text{Frank}) = \textit{false}$
$f(\text{Sally}) = ?$

But name is insufficient!

Need to know, e.g.

- smells like cigarettes?
- has yellow teeth?

Definition

Attributes or features are the components of a domain object believed to be important for learning the function f

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

- $D_f = \{a, aardvark, abacus, abalone, \ldots\}$
- $R_f = \{\text{NOUN}, \text{VERB}, \text{ADJ}, \text{ADV}, \text{DET}, \ldots\}$

$f(bark) = \text{NOUN}?\ \text{VERB}?$

. . . the ***bark*** *of the tree . . .*
. . . heard the dog ***bark*** *. . .*

Feature Representation

$$f([w_0 = bark, w_{-1} = the]) = \text{NOUN}$$
$$f([w_0 = bark, w_{-1} = dog]) = \text{VERB}$$

Learning Supervised Models

After problem formulation, we apply a learning algorithm

Input

A set of $(x, f(x))$ training examples

Output

A function h: $D_f \rightarrow R_f$

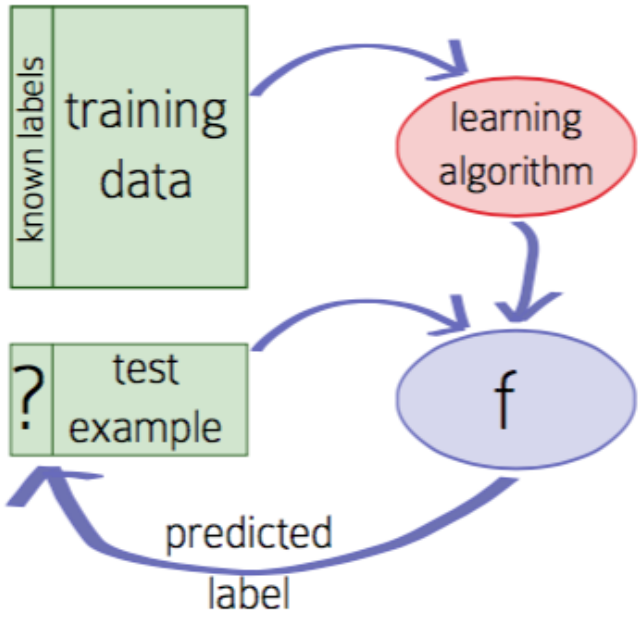
Goal

Pick an h that:

- is consistent with the training examples
- does not overfit the data (i.e. is not overly complex)

Procedure of Learning

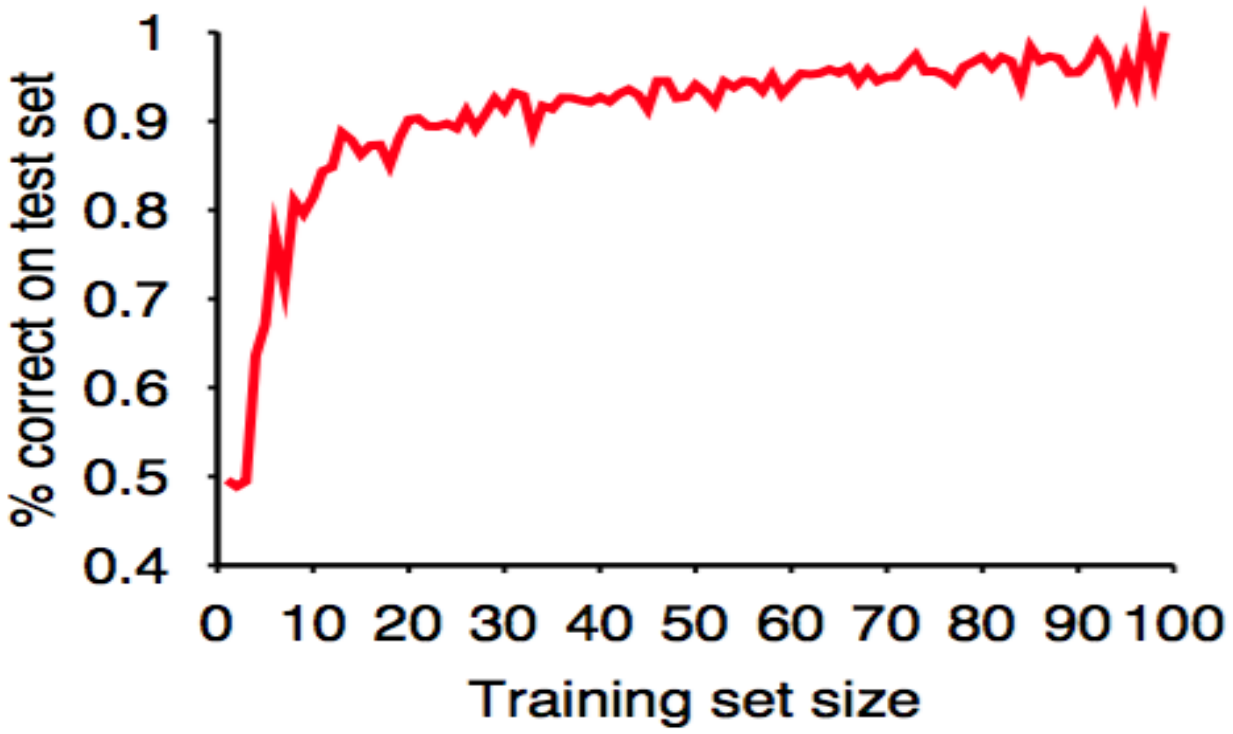
1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test}



Assessing Learning Performance

- Train on increasing fractions of training data
- Test each of the resulting hypotheses on the test data

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



Some Learning Problems

There are a large number of typical inductive learning problems. The primary difference between them is in what type of thing they're trying to predict. Here are some examples:

Regression: trying to predict a real value. For instance, predict the value of a stock tomorrow given its past performance. Or predict Alice's score on the machine learning final exam based on her homework scores.

Binary Classification: trying to predict a simple yes/no response. For instance, predict whether Alice will enjoy a course or not. Or predict whether a user review of the newest Apple product is positive or negative about the product.

Multiclass Classification: trying to put an example into one of a number of classes. For instance, predict whether a news story is about entertainment, sports, politics, religion, etc. Or predict whether a CS course is Systems, Theory, AI or Other.

Ranking: trying to put a set of objects in order of relevance. For instance, predicting what order to put web pages in, in response to a user query. Or predict Alice's ranked preferences over courses she hasn't taken.

Some Learning Problems

There are a large number of typical inductive learning problems. The primary difference between them is in what type of thing they're trying to predict.

For each of these types of machine learning problems, come up with one or two concrete examples?

It is Assignment

Inductive Learning-Summary

- One should keep in mind that the possibility or impossibility of finding a simple, consistent hypothesis depends strongly on the hypothesis space chosen.

- We say that a learning problem is realizable if the hypothesis space contains the true function; otherwise, it is unrealizable.

- Unfortunately, we cannot always tell whether a given learning problem is realizable, because the true function is not known.

- One way to get around this barrier is to use *prior knowledge* to derive a hypothesis space in which we know the true function must lie.

8. Learning

Learning Decision Trees

Lecture III

Introduction

❖ Decision tree (DT) induction is one of the simplest and most successful forms of learning algorithms. It is a good example to inductive learning, and is easy to implement. DT is closely related to the fundamental computer science notion of "divide and conquer."

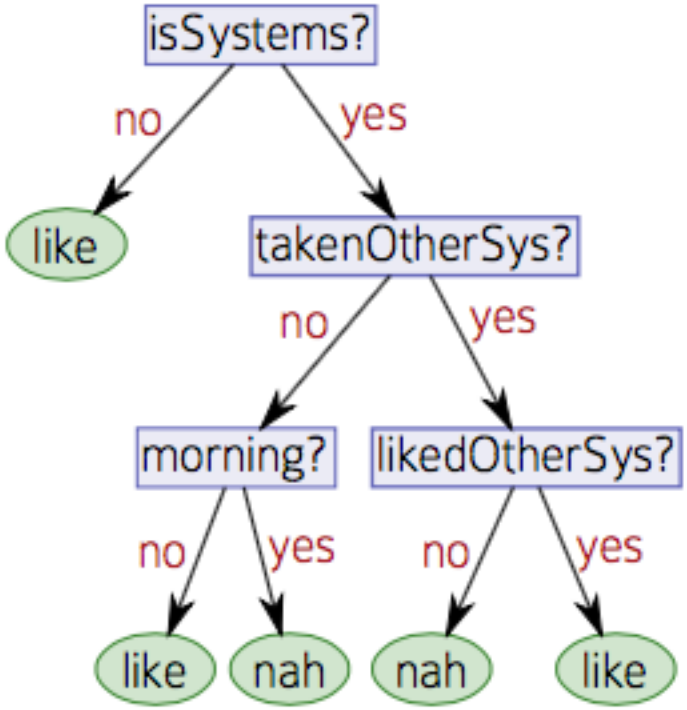
- A decision tree takes as "input" an **object** or a **situation** described by a set of **attributes** and returns a "decision" (*predicted output value for the input*),
- The output value can be **discrete** or **continuous**; learning a discrete-valued function is called classification learning, whereas learning a continuous function is called regression,
- We will concentrate on Boolean (or binary) classification, i.e. each example is classified as **true** (positive) or **false** (negative).

Simple Example

Aim: To predict whether some unknown user will enjoy some unknown course.

You must simply answer "yes" or "no.". In order to make a guess, you're allowed to ask binary questions about the user/course under consideration.

You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: *I predict this student will not like this course.*



- **Questions** are written in the internal tree nodes (rectangles) and the **guesses** are written in the leaves (ovals).
- We will refer to the questions that you can ask as **features** and the responses to these questions as **feature value** . The rating is the **label**.

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$

Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Output: Hypothesis $h \in H$ that best approximates f

Decision Trees (DT)

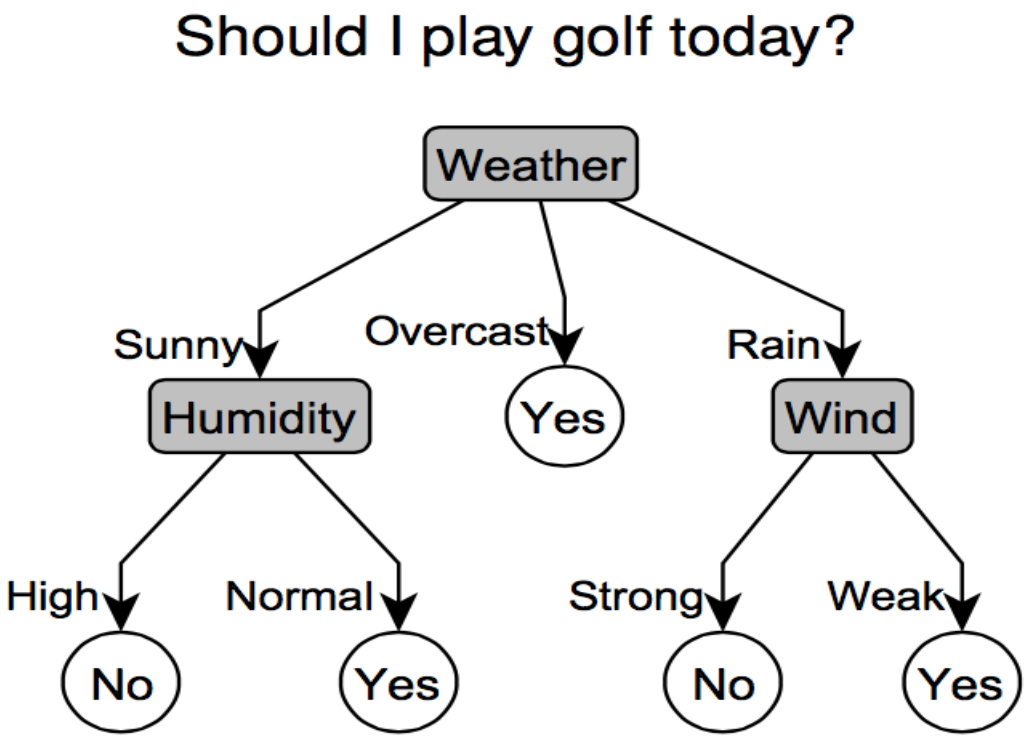
- Each **node** in the tree corresponds either a Decision node or leaf node.

❑ **Decision node: Specifies a choice or test of some attribute with 2 or more alternatives;**

→ **every decision node is part of a path to a leaf node**

→ A decision tree reaches its decision by performing a **sequence of tests**.

❑ **Leaf node:** Indicates classification of an example



Decision Trees- Example

Learning Decision Trees

Function

D_f = days

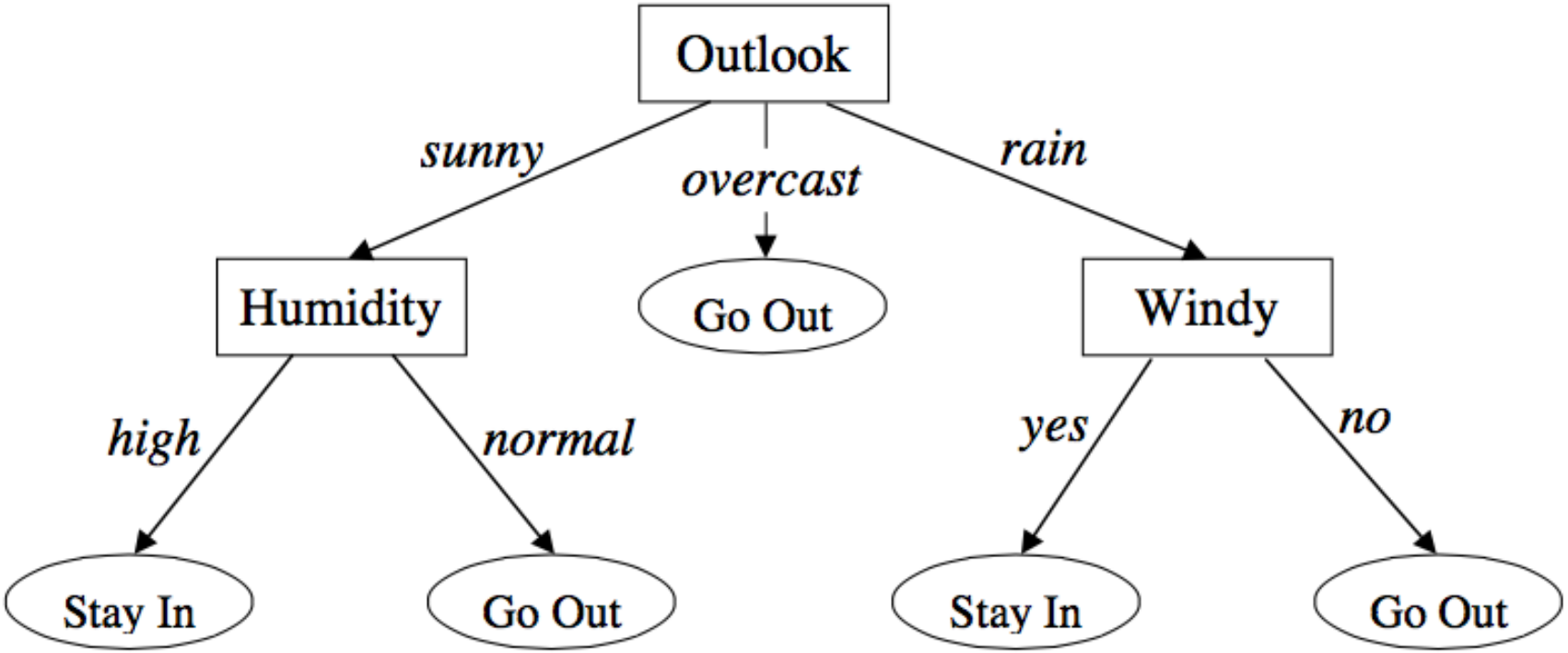
$R_f = \{Yes, No\}$

Features

- Weather
- Humidity
- Wind

DT *vs.* Rule base

Although ***decision trees*** look very different to ***rule based systems***, it is actually easy to convert a decision tree into a set of rules. For example:



DT *vs.* Rule base

Can be represented using the following rules

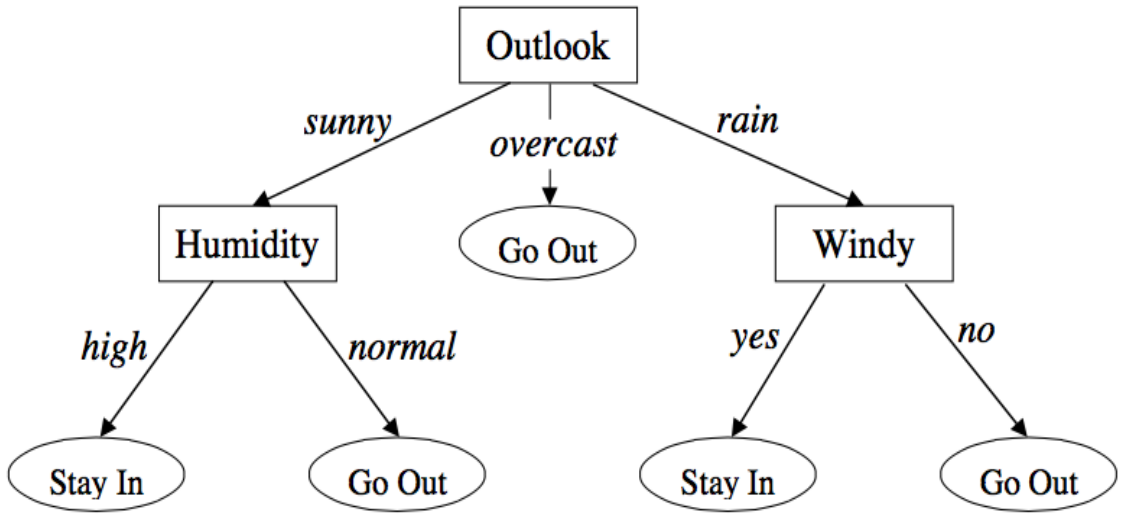
R1: IF: Outlook = *overcast*
THEN: Go Out

R2: IF: Outlook = *sunny*
Humidity = *normal*
THEN: Go Out

R3: IF: Outlook = *rain*
Windy = *no*
THEN: Go Out

R4: IF: Outlook = *sunny*
Humidity = *high*
THEN: Stay In

R5: IF: Outlook = *rain*
Windy = *yes*
THEN: Stay In



DT *vs.* Rule base

Can be represented using the following rules

R1: IF: Outlook = *overcast*
THEN: Go Out

R2: IF: Outlook = *sunny*
Humidity = *normal*
THEN: Go Out

R3: IF: Outlook = *rain*
Windy = *no*
THEN: Go Out

R4: IF: Outlook = *sunny*
Humidity = *high*
THEN: Stay In

R5: IF: Outlook = *rain*
Windy = *yes*
THEN: Stay In

The advantage of DT over Rules is that comparatively simple algorithms can derive decision trees (from training data) that are good at generalizing (i.e. classifying unseen instances).

Learning Decision Trees - Example

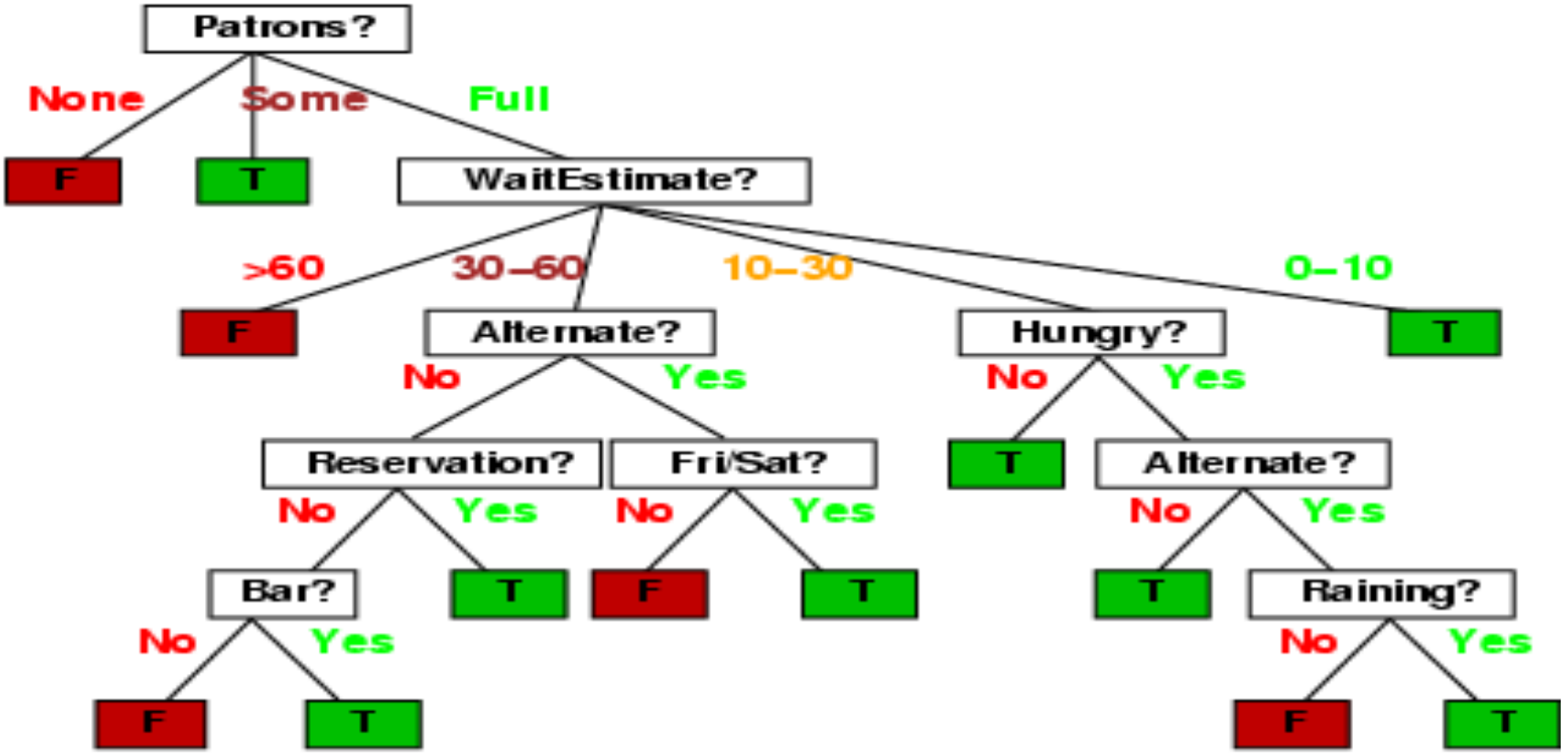
Problem: decide whether to wait for a table at a restaurant, based on the following attributes *

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. Wait Estimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Aim: to learn a definition for the **goal predicate** Will Wait

Learning Decision Trees

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



Selecting Features of Decision Trees

Feature Selection Order

- Different orders result in different trees
- "Good" features should be used before "poor" ones

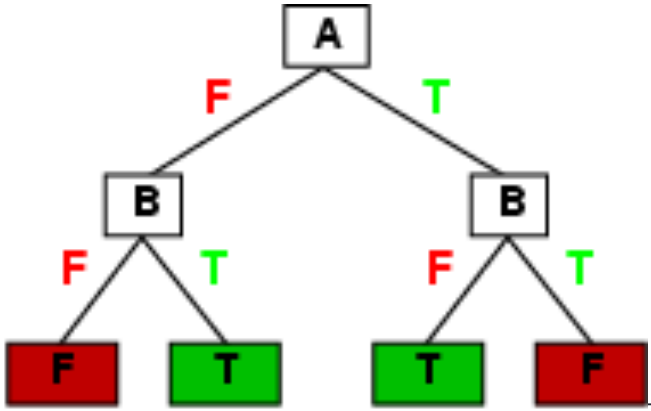
What is a "good" feature?

One whose values predict the class labels

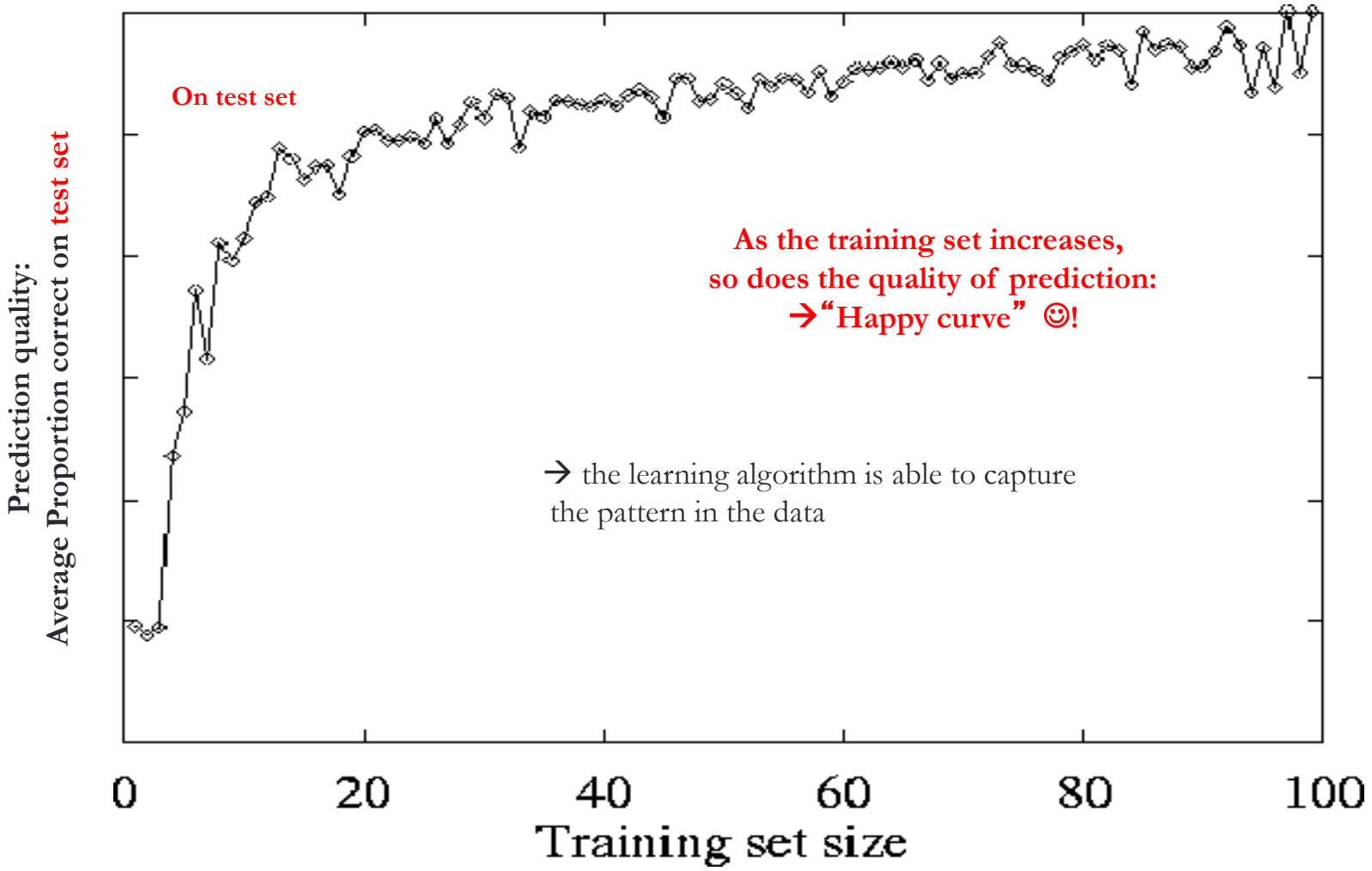
Expressiveness of Decision Trees

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:
- This can be done by having each row in the truth table for the function correspond to a path in the tree
- This would yield an exponentially large decision tree representation*. For some kinds of functions, however, this is a real problem
- **In other words, decision trees are good for some kinds of functions and bad for others.**
- Is there any kind of representation that is efficient for *all* kinds of functions? Unfortunately, No.

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



Assessing Learning Performance- Restaurant Example:



How well does DTs work?

- Many case studies have shown that decision trees are **at least** as accurate as human experts.
  - ✓ A study for **diagnosing breast cancer** had humans correctly classifying the examples 65% of the time, and the **decision tree** classified 72% correct.
  - ✓ **British Petroleum** designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system.
  - ✓ **CESSNA** designed an airplane flight controller using 90,000 examples and 20 attributes per example.

Summary of Learning using DTs

- Inducing DT is one of the most widely used learning methods in practice when Instances presented as attribute-value pairs
- Can outperform human experts in many problems
- DT learning is a particular case of supervised learning,
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Learning performance = prediction accuracy measured on test set
- **Famous algorithms are ID3, and C4.5**

Summary of Learning using DTs

Strengths

- Fast and simple to implement
- human readable
- can convert result to a set of easily interpretable rules
- empirically valid in many commercial products
- handles noisy data where data may contain errors or missing attributes

Weaknesses

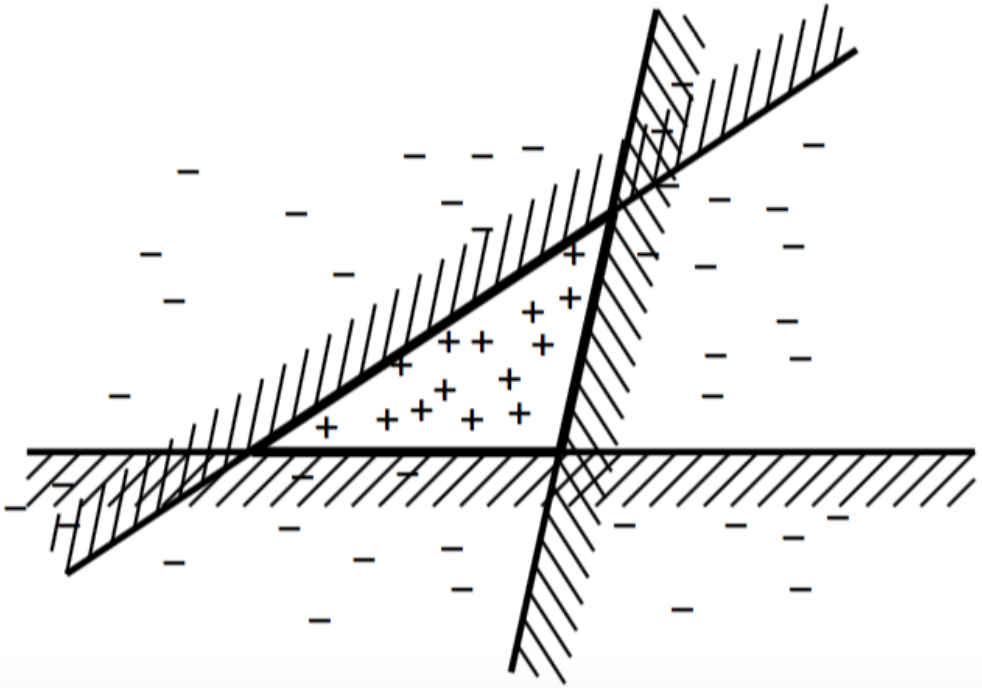
- "Univariate" splits/partitioning using only one attribute at a time so limits types of possible trees
- large decision trees may be hard to understand
- requires fixed-length feature vectors
- non-incremental (i.e., batch method)

So far we have looked in inductive learning, a single hypothesis, chosen from a hypothesis space, is used to make predictions for the output

Ensemble Learning

Key Idea

Two (or more) classifiers are better than one



So far we have looked in inductive learning, a single hypothesis, chosen from a hypothesis space, is used to make predictions for the output

Ensemble Learning

- The idea: Select a whole collection, or **ensemble,** of hypotheses from the hypothesis space and combine their predictions.

For example, we might generate a hundred different decision trees from the same training set and have them vote on the best classification for a new example.

Why ensemble learning?

Consider an ensemble of $M = 5$ hypotheses and suppose that we combine their predictions using simple majority voting. For the ensemble to misclassify a new example, *at least three of the five hypotheses have to misclassify it.* The hope is that this is **much less likely** than a misclassification by a single hypothesis.

Ensemble Learning * Boosting

- In a training set, each example has an associated weight $w_j > 0$. The higher the weight of an example, the higher is the importance attached to it during the learning of a hypothesis, **
- Boosting starts with $w_j = 1$ for all the examples. From this set, it generates the first hypothesis, h_1. This hypothesis will classify some of the training examples correctly and some incorrectly.
- We would like the next hypothesis to do better on the misclassified examples, so we increase their weights while decreasing the weights of the correctly classified examples.
- From this new weighted training set, we generate a hypothesis h_2. The process continues in this way until we have generated M hypotheses.
- The final ensemble hypothesis is a weighted-majority combination of all the M hypotheses, each weighted according to how well it performed on the training set.

Boosting

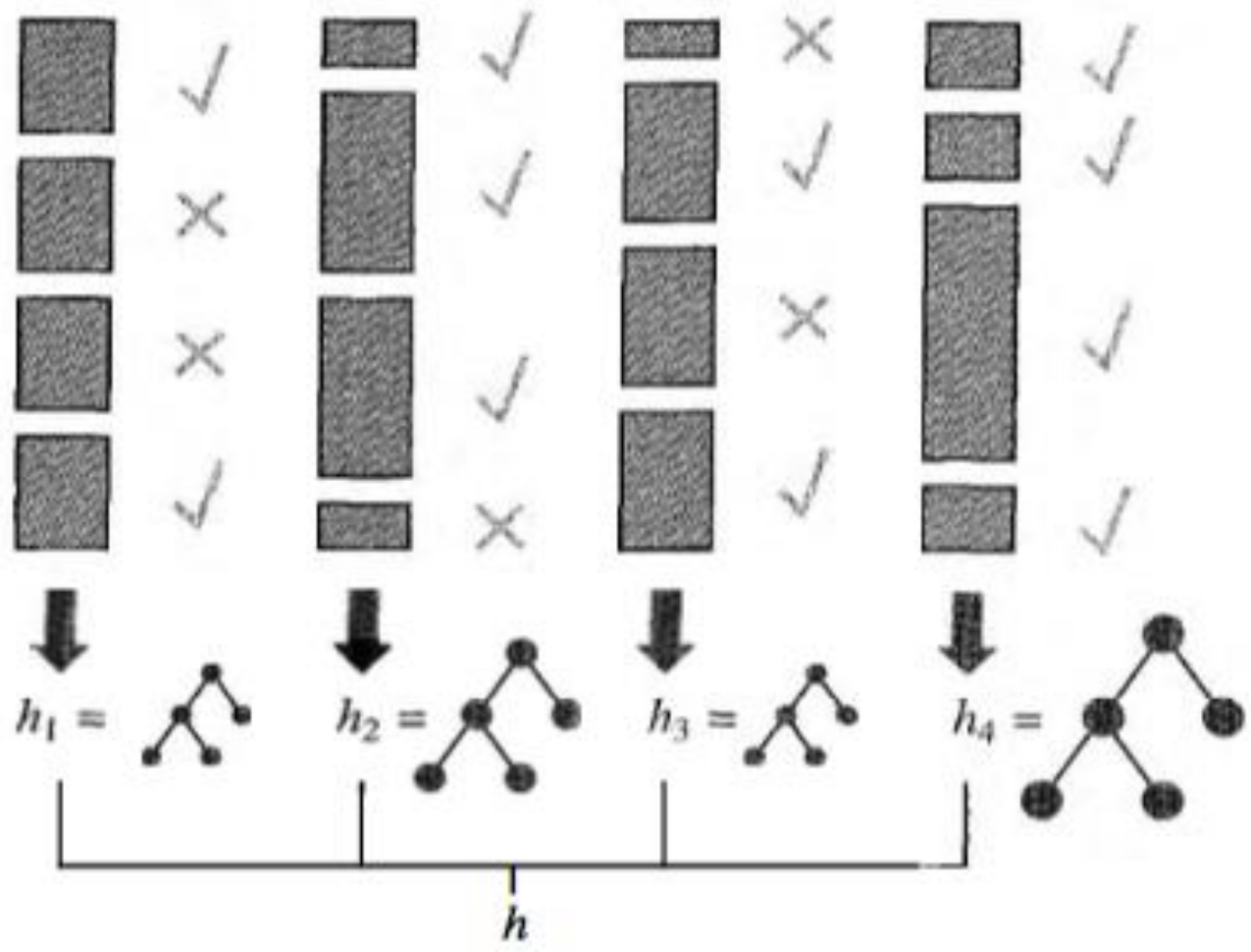
Key Ideas

- Look at examples the first classifier got wrong
- Create new models that do better on these

Boosting

1. Start with all training samples weighted equally
2. Train a model (e.g. a single feature decision stump)
3. Increase the weight of all misclassified examples
4. Repeat steps 2 and 3 until m models are generated

Ensemble Learning * Boosting



Assessing Learning Performance "Testing"

- A learning algorithm is good if it produces hypotheses that do a good job of predicting the classifications of unseen examples.

- A prediction is good if it turns out to be true, so we can assess the quality of a hypothesis by checking its predictions against the correct classification once we know it. **How do we know that $h \approx f$?**

This is done via a "Test Set"

Try h on a new test set of examples

Assessing Learning Performance

General Framework for Learning

1. Collect a large set of examples,
2. Divide it into two disjoint sets: the **training set** and the **test set**,
3. Apply the learning algorithm to the training set, generating a hypothesis h,
4. Measure the percentage of examples in the test set that are correctly classified by *h*,
5. Repeat steps 2 to 4 for different sizes of training sets and different randomly selected training sets of each size.

Summary

- Learning is needed for unknown environments, lazy designers,
- Learning agent = performance element + learning element
- Learning performance = prediction accuracy measured on test set
- Learning takes many forms, depending on the nature of the performance element, the component to be improved, and the available feedback.
- If the available feedback, either from a teacher or from the environment, provides the correct value for the examples, the learning problem is called supervised learning. The task, also called inductive learning, is then to learn a function from examples of its inputs and outputs.
- Inductive learning involves finding a consistent hypothesis that agrees with the examples. Ockham's razor suggests choosing the simplest consistent hypothesis. The difficulty of this task depends on the chosen representation.

Summary

- The performance of a learning algorithm is measured by the learning curve, which shows the prediction accuracy on the test set as a function of the training set size.
- Ensemble methods such as boosting often perform better than individual methods.
- The approximation improves as more hypotheses are added.
- Another possible explanation is that the addition of further hypotheses enables the ensemble to be more definite in its distinction between positive and negative examples, which helps it when it comes to classifying new examples.

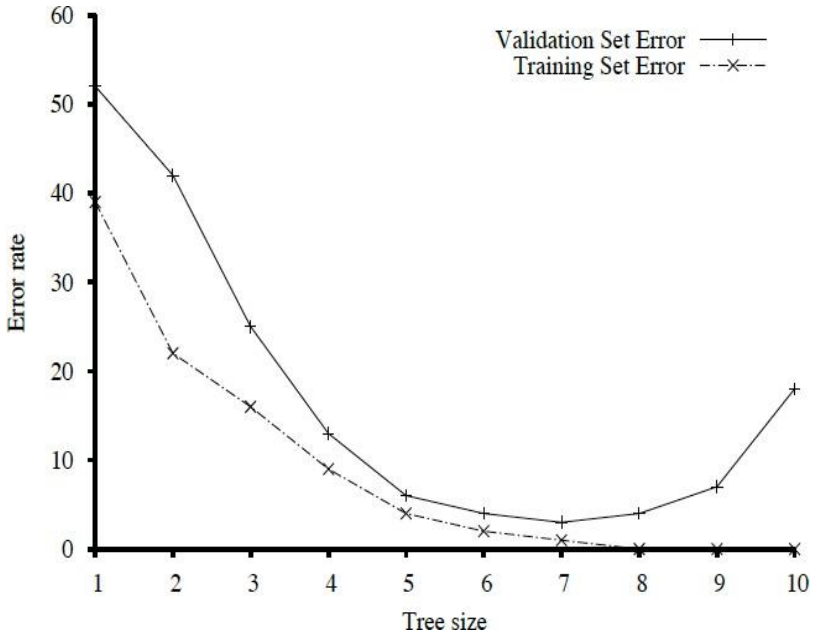
8. Learning

Regression & Classification using Linear Models

Lecture IV

Important issues

- Generalization
- Overfitting
- Cross-validation
  - Holdout cross validation
  - K-fold cross validation
  - Leave-one-out cross-validation
- Model selection



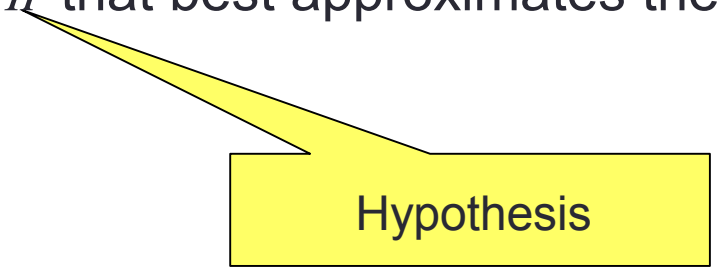
Remember!

$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ training set

Where each y_j was generated by an unknown function

$$y = f(\mathbf{x})$$

Discover a function h that best approximates the true function f



Loss Functions

Suppose the true prediction for input $\mathbf{x}$ is $f(\mathbf{x}) = y$

but the hypothesis gives $h(\mathbf{x}) = \hat{y}$

$L(\mathbf{x}, y, \hat{y}) = Utility(\text{result of using } y \text{ given input } \mathbf{x})$

$\quad - Utility(\text{result of using } \hat{y} \text{ given input } \mathbf{x})$

Simplified version: $L(y, \hat{y})$

Absolute value loss: $L_1(y, \hat{y}) = |y - \hat{y}|$

Squared error loss: $L_2(y, \hat{y}) = (y - \hat{y})^2$

0/1 loss: $L_{0/1}(y, \hat{y}) = 0$ if $y = \hat{y}$, else 1

Linear Regression?

- It is a basic and commonly used type of predictive analysis.
- The overall idea of regression is to examine two things:
  (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?

  (2) Which variables in particular are significant predictors of the outcome variable,
- These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.
- The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

Linear Regression?

- Three major uses for regression analysis are

  (1) Determining the strength of predictors,

  (2) Forecasting an effect, and

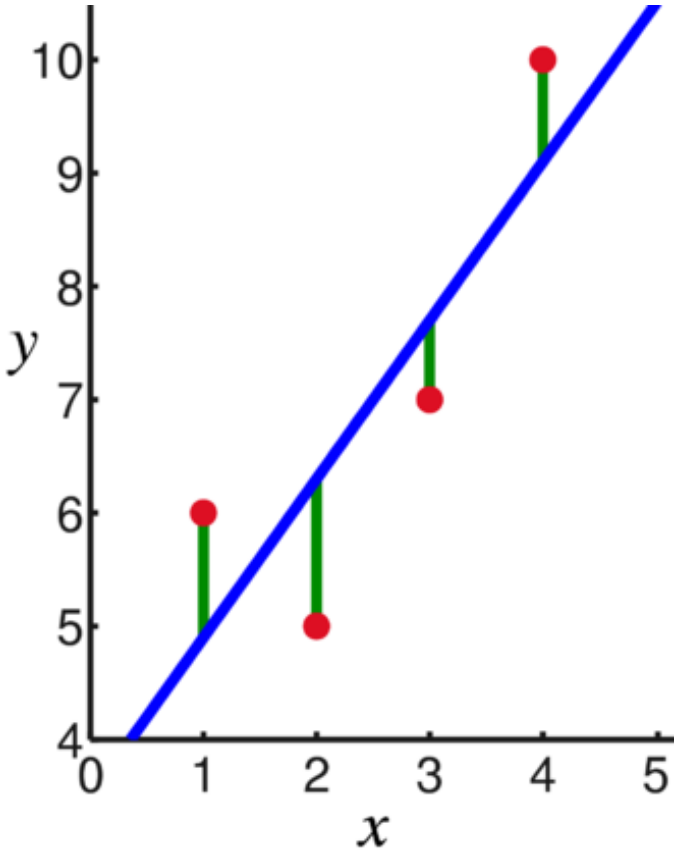
  (3) Forecasting a trend.

(1) The regression might be used to **identify** the strength of the effect that the independent variable(s) have on a dependent variable. *Typical questions are what is the strength of relationship between dose and effect, sales and marketing spending, or age and income, etc.*

(1) It can be used to **forecast** effects or impact of changes. That is, the regression analysis helps us to understand how much the dependent variable changes with a change in one or more independent variables. *A typical question is, "how much additional sales income do I get for each additional $1000 spent on marketing?"*

(2) Regression analysis **predicts** trends and future values. The regression analysis can be used to get point estimates. *A typical question is, "what will the price of gold be in 6 months?"*

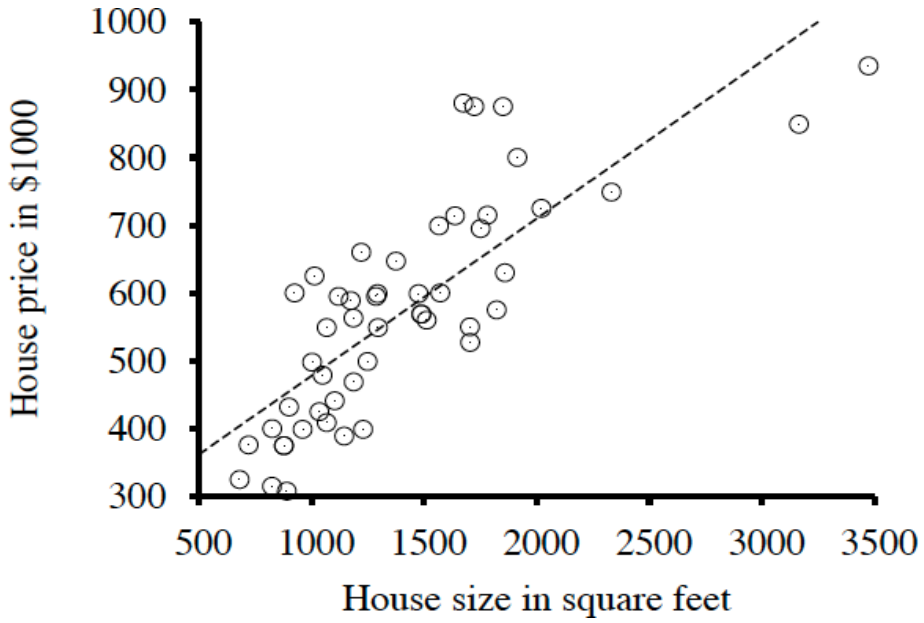
Linear Regression?



In linear regression, the observations (**red**) are assumed to be the result of random deviations (**green**) from an underlying relationship (**blue**) between a dependent variable (*y*) and an independent variable (*x*).

Univariate Linear Regression

Example



Univariate Linear Regression

$$\mathbf{w} = [w_0, w_1] \qquad \text{weight vector}$$

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

where w0 and w1 are real-valued coefficients to be learned

- **The task is to find the $h_{\mathbf{w}}$ that best fits the training data**

To fit a line to the data, we need to find the weight vector that minimizes the empirical loss, e.g .the squared loss function using L_2

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

i.e., find $\mathbf{w}^*$ such that

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

Finding w*

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

This is a general optimization search problem in a continuous weight space. We can find the weights as:

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \quad \text{and} \quad \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

i.e. The sum of the squared loss function is minimized when the partial derivatives with respect to w_0 and w_1 are zero.

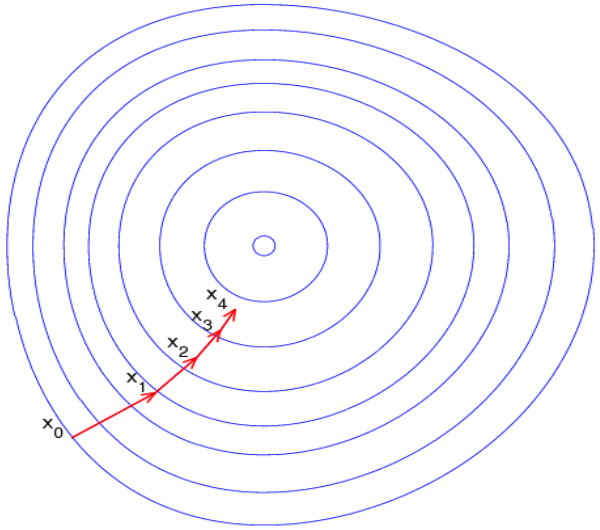
Gradient Descent

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

step size or learning rate

(a fixed or decaying constant as the learning proceeds)

- **Gradient descent** is an algorithm that minimizes functions.
- Given a function defined by a set of parameters, **gradient descent** starts with an initial set of parameter values
- iteratively moves toward a set of parameter values that minimize the function.



w ← any point in the parameter space

loop until convergence **do**

  for each w_i **in w do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

Gradient Descent

For one training example (x, y):

$$w_0 \leftarrow w_0 + \alpha(y - h_{\mathbf{w}}(x)) \quad \text{and} \quad w_1 \leftarrow w_1 + \alpha(y - h_{\mathbf{w}}(x))x$$

For *N* training examples:

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \quad \text{and} \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))x_j$$

batch gradient descent

stochastic gradient descent: take a step for one training example at a time

The Multivariate Linear Regression

$$h_{sw}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i}$$

Augmented vectors: add a feature to each $\mathbf{x}$ by tacking on a 1: $x_{j,0} = 1$

Then:

$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^T \mathbf{x}_j = \sum_i w_i x_{j,i}$$

And batch gradient descent update becomes:

$$w_i \leftarrow w_i + \alpha \sum_j (y_j - h_{\mathbf{w}}(\mathbf{x}_j)) x_{j,i}$$

Linear Classification sdlohserht drah

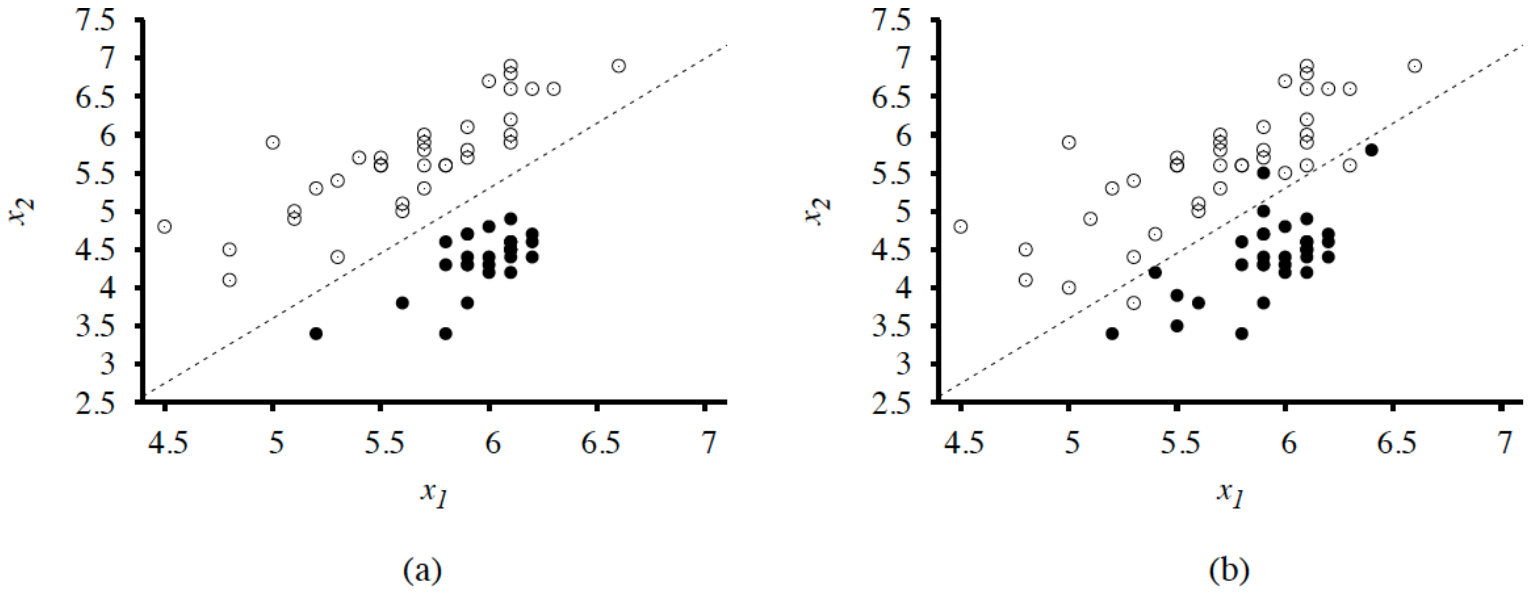


Figure 18.15 **FILES: .** (a) Plot of two seismic data parameters, body wave magnitude x_1 and surface wave magnitude x_2, for earthquakes (white circles) and nuclear explosions (black circles) occurring between 1982 and 1990 in Asia and the Middle East (?). Also shown is a decision boundary between the classes. (b) The same domain with more data points. The earthquakes and explosions are no longer linearly separable.

Linear Classification sdlohserht drah

- Decision Boundary:
  - A line (or a surface, in higher dimensions) that separates the two classes
  - In linear case: linear separator, a hyperplane

- Linearly separable:
  - data is linearly separable if the classes can be separated by a linear separator


Any questions?