Large Scale Data Analysis for Policy 90-866, Fall 2012

Course Overview

Instructor: Daniel B. Neill (neill@cs.cmu.edu)

TA: Sriram Somanchi (somanchi@cmu.edu)

Why large scale data analysis?

Critical importance of addressing global policy problems: disease pandemics, crime, terrorism, poverty, environment…

Increasing size and complexity of available data, thanks to the rapid growth of new and transformative technologies.

Much more computing power, and scalable data analysis methods, enable us to extract actionable information from all of this data.

Large scale data analysis techniques have become increasingly essential for policy analysis, and for the development of new, practical information technologies that can be directly applied **for the public good** (e.g. public health, safety, and security)

Course description and goals

This course will focus on applying large scale data analysis methods from the closely related fields of **machine learning**, **data mining**, and **artificial intelligence** to develop tools for intelligent problem solving in real-world policy applications.

We will emphasize tools that can "scale up" to real-world problems with huge amounts of high-dimensional and multivariate data.

Mountain of policy data
Huge, unstructured, hard to interpret or use for decisions

1. Translate policy questions into ML paradigms.
2. Choose and apply appropriate methods.
3. Interpret, evaluate, and use results.

Actionable knowledge of policy domain

Predict & **explain** unknown values
Model structures, relations
Detect relevant patterns

Use for decision-making, policy prescriptions, improved services

Some definitions

Machine Learning (ML) is the study of systems that improve their performance with experience (typically by **learning** from data).

Artificial Intelligence (AI) is the science of automating complex behaviors such as learning, problem solving, and decision making.

Data Mining (DM) is the process of extracting useful information from massive quantities of complex data.

I would argue that these are not three distinct fields of study! While each has a slightly different emphasis, there is a tremendous amount of overlap in the problems they are trying to solve and the techniques used to solve them.

Many of the techniques we will learn are **statistical** in nature, but are very different from classical statistics.

ML/AI/DM systems and methods:

Scale up to large, complex data
Learn and **improve** from experience
Perceive and **change** the environment
Interact with humans or other agents
Explain inferences and decisions
Discover new and useful patterns

Structure of the course

- 11 lectures divided into three modules (Prediction, Modeling, and Detection).
- Grades will be based on:
  - Class participation: 5%
  - Project plan: 10%, Project progress report: 10%
  - Final project report: 25%, Final presentation: 10%
  - Final exam: 40%
- Course projects will be done in three person teams.
- See syllabus on Blackboard for:
  - Instructor information (e-mails, office hours, etc.)
  - Detailed course schedule (subject to change!)
  - Description of projects and practice exercises.
  - Course policies (cheating, late work, re-grades)

Course syllabus

- Introduction to Large Scale Data Analysis
  - Incorporates methods from **machine learning**, data mining, artificial intelligence.
  - Goals, problem paradigms, and software tools
- Module I (**Prediction**)
  - Classification and regression (making, explaining predictions)
  - Rule-based, case-based, and model-based learning.
- Module II (**Modeling**)
  - Representation and heuristic search
  - Clustering (modeling group structure of data)
  - Bayesian networks (modeling probabilistic relationships)
- Module III (**Detection**)
  - Anomaly Detection (detecting outliers, novelties, etc.)
  - Pattern Detection (e.g. event surveillance, anomalous patterns)
  - Applications to biosurveillance, crime prevention, etc.
  - Guest “mini-lectures” from the Event and Pattern Detection Lab.

How is ML relevant for policy?

ML provides a powerful set of **tools** for intelligent problem-solving.

Building sophisticated models that combine data and prior knowledge to enable intelligent decisions.

Automating tasks such as prediction and detection to reduce human effort.

Scaling up to large, complex problems by focusing user attention on relevant aspects.

Using ML to analyze data and guide policy decisions.

Proposing policy initiatives to reduce the amount and impact of violent crime in urban areas.

Predicting the adoption rate of new technology in developing countries.

Analyzing which factors influence congressional votes or court decisions

Using ML in information systems to improve public services

Health care: diagnosis, drug prescribing
Law enforcement: crime forecasting
Public health: epidemic detection/response
Urban planning: optimizing facility location
Homeland security: detecting terrorism

Analyzing impacts of ML technology adoption on society

Internet search and e-commerce
Data mining (security vs. privacy)
Automated drug discovery
Industrial and companion robots
Ethical and legal issues

Advertisement: MLP@CMU

We are working to build a comprehensive curriculum in **machine learning and policy (MLP)** here at CMU.

Goals of the MLP initiative: increase collaboration between ML and PP researchers, train new researchers with deep knowledge of both areas, and encourage a widely shared focus on using ML to benefit the public good.

Here are some of the many ways you can get involved:

Joint Ph.D. Program in Machine Learning and Public Policy (MLD & Heinz)
Ph.D. in Information Systems + M.S. in Machine Learning

Large Scale Data Analysis for Policy: introduction to ML for PPM students.
Research Seminar in Machine Learning & Policy: for ML/Heinz Ph.D. students.
Special Topics in Machine Learning and Policy: Event and Pattern Detection, ML for Developing World, Wisdom of Crowds, **Mining Massive Datasets**

Workshop on Machine Learning and Policy Research & Education
Research Labs: Event and Pattern Detection Lab, Auton Laboratory, iLab
Center for Science and Technology in Human Rights, many others…

Common ML paradigms: prediction

In **prediction**, we are interested in explaining a specific attribute of the data in terms of the other attributes.

Classification: predict a discrete value

"What disease does this patient have, given his symptoms?"

Regression: estimate a numeric value

"How is a country's literacy rate affected by various social programs?"

Two main goals of prediction

Guessing unknown values for specific instances (e.g. diagnosing a given patient)

Explaining predictions of both known and unknown instances (providing relevant examples, a set of decision rules, or class-specific models).

Example 1: What socio-economic factors lead to increased prevalence of diarrheal illness in a developing country?

Example 2: Developing a system to diagnose a patient's risk of diabetes and related complications, for improved medical decision-making.

Common ML paradigms: modeling

In **modeling**, we are interested in describing the underlying relationships between many attributes and many entities.

Our goal is to produce models of the "entire data" (not just specific attributes or examples) that accurately reflect underlying complexity, yet are simple, understandable by humans, and usable for decision-making.

Relations between entities	Relations between variables
Identifying link, group, and network structures	Identifying significant positive and negative correlations
Partitioning or "clustering" data into subgroups	Visualizing dependence structure between multiple variables

Example 1: Can we visualize the dependencies between various diet-related risk factors and health outcomes?

Example 2: Can we better explain consumer purchasing behavior by identifying subgroups and incorporating social network ties?

Common ML paradigms: detection

In **detection**, we are interested in identifying relevant patterns in massive, complex datasets.

Main goal: focus the user's attention on a potentially relevant subset of the data.

a) Automatically detect relevant individual records, or groups of records.

b) Characterize and explain the pattern (type of pattern, H_0 and H_1 models, etc.)

c) Present the pattern to the user.

Some common detection tasks

Detecting **anomalous** records or groups

Discovering **novelties** (e.g. new drugs)

Detecting **clusters** in space or time

Removing **noise** or **errors** in data

Detecting **specific patterns** (e.g. fraud)

Detecting emerging **events** which may require rapid responses.

Example 1: Detect emerging outbreaks of disease using electronic public health data from hospitals and pharmacies.

Example 2: How common are patterns of fraudulent behavior on various e-commerce sites, and how can we deal with online fraud?

Software tools for data analysis

Many different ML software tools are available for performing different large scale data analysis tasks.

Weka data mining toolkit

Free Java open-source software, available for download at:
http://www.cs.waikato.ac.nz/ml/weka/
Weka contains classifiers, clustering, data visualization and preprocessing tools; command-line and graphical interfaces.

ASL (Accelerated Statistical Learning)

Machine learning software available for download at:
http://www.autonlab.org

Programs such as **Excel** and **Minitab** are useful for preprocessing and exploratory data analysis.

If you have some programming experience and want to implement your own ML methods, you could use **Matlab**, **R**, **Java**, **C++**, etc.

Large Scale Data Analysis for Policy 90-866, Fall 2012

Module I: Prediction (Classification and Regression)

Data set representation

Our dataset consists of a set of **data records** $\{x_i\}$.

Each record has values for a set of **attributes** $\{A_j\}$.

Each data record x_i has a **value** v_{ij} for each attribute A_j.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Attributes can be real-valued (a number) or discrete-valued (a class).

Some attribute values may be missing (represented here by ???).

The prediction problem

Our dataset consists of a set of **data records** $\{x_i\}$.

Each record has values for a set of **attributes** $\{A_j\}$.

Each data record x_i has a **value** v_{ij} for each attribute A_j.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

The goal of prediction is to guess the missing value of some attribute for a given data point, given the other attributes for that point, as well as the rest of the dataset.

The prediction problem

Our dataset consists of a set of **data records** $\{x_i\}$.

Each record has values for a set of **attributes** $\{A_j\}$.

Each data record x_i has a **value** v_{ij} for each attribute A_j.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

If we are predicting a discrete value (e.g. heart attack risk), this is a classification problem.

If we are predicting a real value (e.g. blood pressure), this is a regression problem.

The prediction problem

Our dataset consists of a set of **data records** $\{x_i\}$.

Each record has values for a set of **attributes** $\{A_j\}$.

Each data record x_i has a **value** v_{ij} for each attribute A_j.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Let A_p denote the attribute we are trying to predict. Assume that all records either a) have no missing values, or b) have only A_p missing. We call the first set **training records**, and the second set **test records**.

The prediction problem

Our dataset consists of a set of **data records** $\{x_i\}$.

Each record has values for a set of **attributes** $\{A_j\}$.

Each data record x_i has a **value** v_{ij} for each attribute A_j.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Our goal is to <u>accurately</u> predict the missing values of A_p for each test record, using the training data.

<u>Classification</u>
Maximize proportion of correct predictions

<u>Regression</u>
Minimize mean squared error

Three approaches to prediction

In rule-based learning, we infer a set of rules from the training data, and use these rules for prediction of missing values.

If (Diabetes = Yes) or (Systolic BP > 180) then Risk = High, else Risk = Low.

(So what should we predict for John?)

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Rule-based learning makes it easy to see which attributes are most relevant for a given prediction.

It is commonly used for medical diagnosis, failure troubleshooting, and sequential decision-making.

Three approaches to prediction

In instance-based learning, we choose a set of training examples that are most similar to the given example, and use these for prediction.

John is more similar to Kathy than to Bob.

(So what should we predict for John?)

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Instance-based learning makes it easy to see which examples are most relevant for a given prediction.

It is commonly used for many applications including information retrieval and recommender systems.

Three approaches to prediction

In model-based learning, we build a (possibly complex) model for each class, and infer the probability of each class given the data.

High risk individuals tend to be male, high BMI, high BP, diabetic. Low risk individuals tend to be female, low BMI, low BP, no diabetes.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Model-based learning enables us to gain a detailed understanding of the characteristics of each class.

Since class probabilities can be predicted, it is commonly used for risk analysis and decision support.

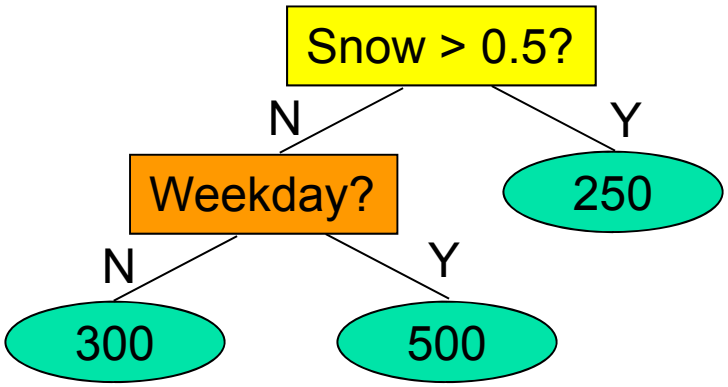
This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Large Scale Data Analysis for Policy 90-866, Fall 2012

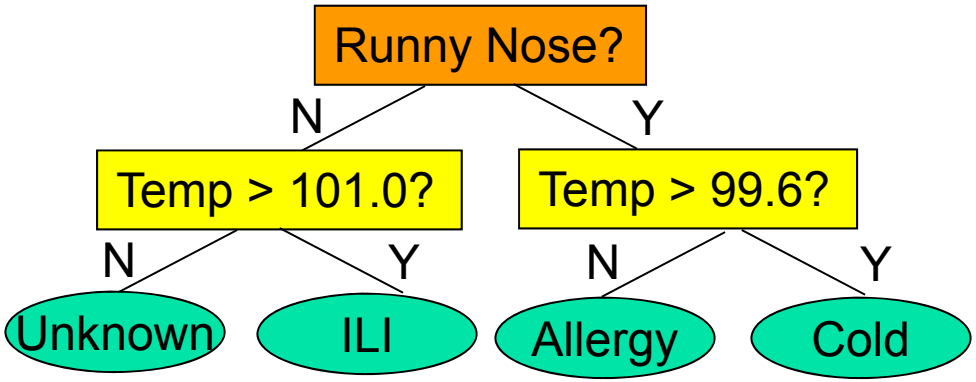
Lecture 2: Rule-Based Learning (Decision Trees)

Rule-based learning with decision trees

- A decision tree is a set of rules that can be learned from data and used to predict an unknown value.
  - Unknown real value (regression): What is the expected incidence of car thefts in NYC on a given day?
  - Unknown category value (classification): What type of illness does patient X have, given their symptoms and demographic data?



How many thefts on Tuesday, January 3 (0.2 inches of snow)?



What do we predict for a patient with Temp = 100 and a runny nose?

Learning binary decision trees

Example dataset:
Predicting whether a car is fuel-efficient, given its number of cylinders (4, 6, or 8), weight (light, medium, or heavy), and horsepower (real-valued).

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).

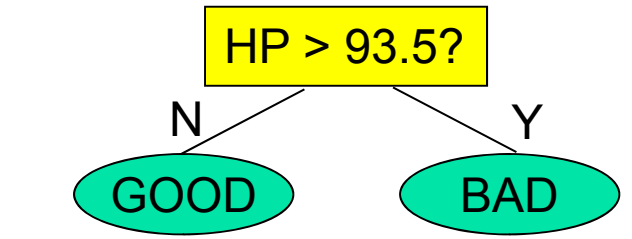
(5 good, 15 bad)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.



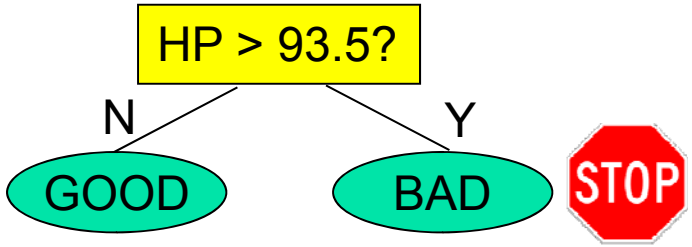
(5 good, 3 bad) (0 good, 12 bad)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.



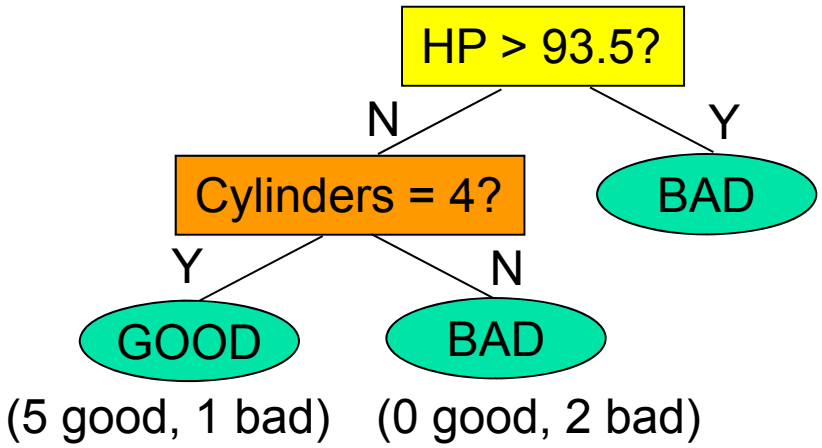
(5 good, 3 bad) (0 good, 12 bad)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.

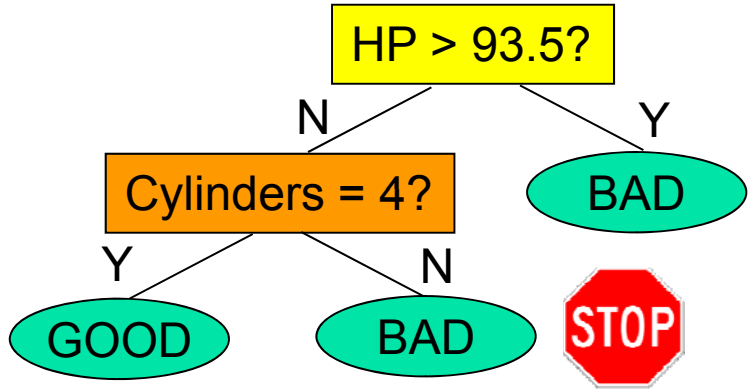


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.

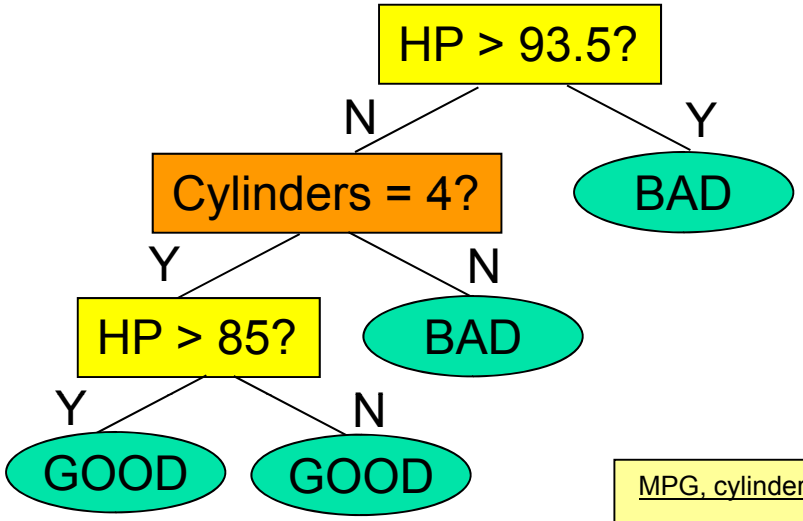


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.



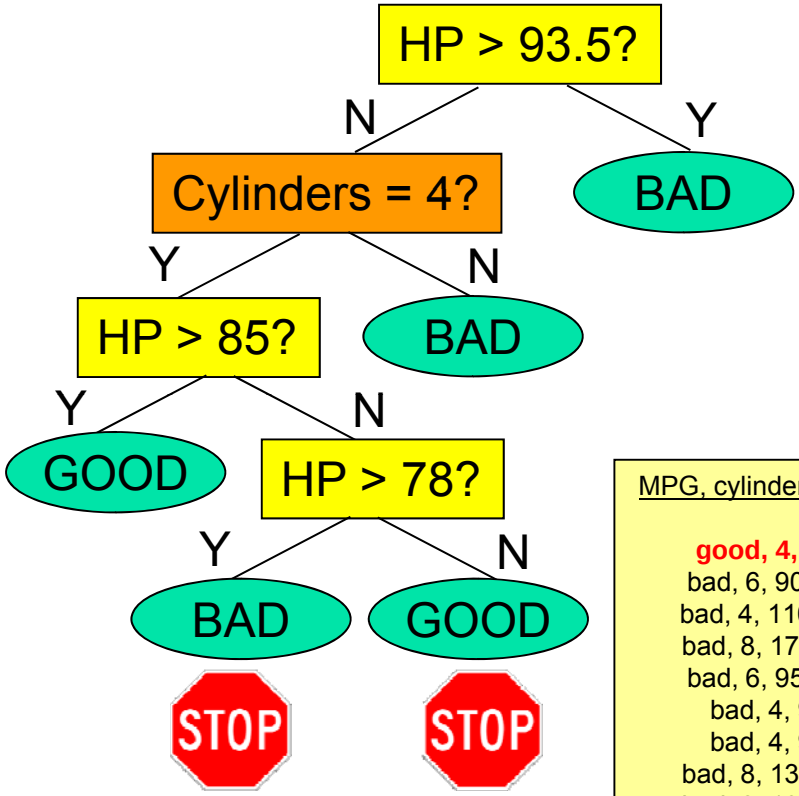
(3 good, 0 bad) **(2 good, 1 bad)**

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.



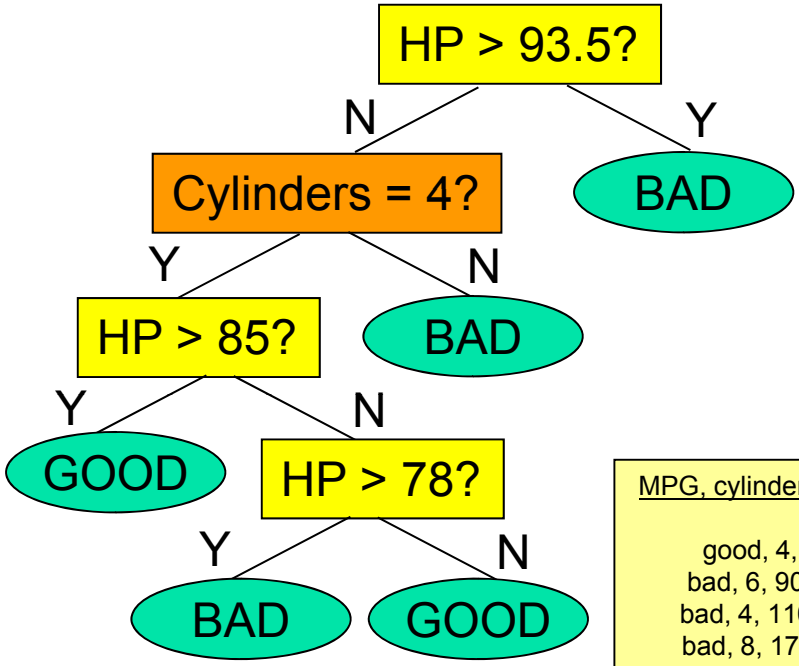
MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
  - **All outputs same?**
  - **All inputs same?**

bad, 4, 81, light
good, 4, 81, light

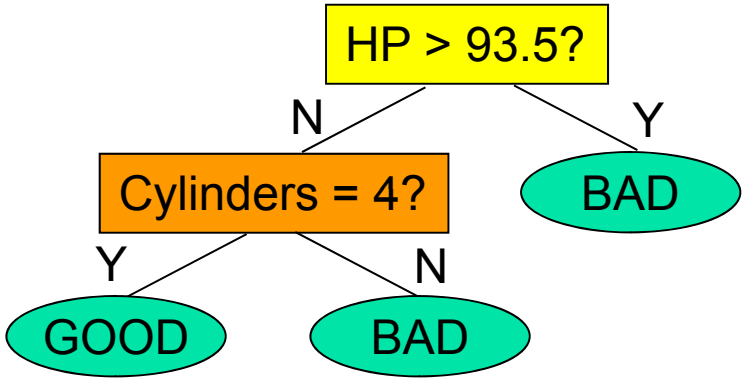


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
- Step 4: Prune the tree to remove irrelevant rules.

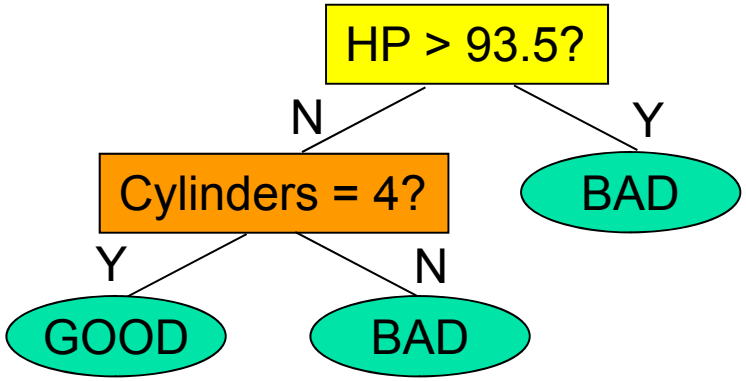


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the "best" binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
- Step 4: Prune the tree to remove irrelevant rules.



Question 1: How to choose the best decision rule for a given node?

Question 2: How to prune the tree (and why bother?)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We can use any input attribute to split.
  - If discrete: choose a class, split into = and ≠.
  - If real: choose a threshold, split into > and ≤.
- To choose a threshold for a real attribute: sort the values, and use midpoints.

Split	Group Y	Group N
Cylinders = 4?	5+ / 4-	0+ / 11-
Cylinders = 6?	0+ / 6-	5+ / 9-
Cylinders = 8?	0+ / 5-	5+ / 10-
HP > 78?	2+ / 0-	3+ / 15-
HP > 87?	2+ / 2-	3+ / 13-
HP > 89.5?	3+ / 2-	2+ / 13-
HP > 91?	3+ / 3-	2+ / 12-
HP > 93.5?	5+ / 3-	0+ / 12-
Weight = light?	3+ / 3-	2+ / 12-
Weight = medium?	2+ / 6-	3+ / 9-
Weight = heavy?	0+ / 6-	5+ / 9-

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We can use any input attribute to split.
  - If discrete: choose a class, split into = and ≠.
  - If real: choose a threshold, split into > and ≤.
- To choose a threshold for a real attribute: sort the values, and use midpoints.
- Choose the split with highest information gain.

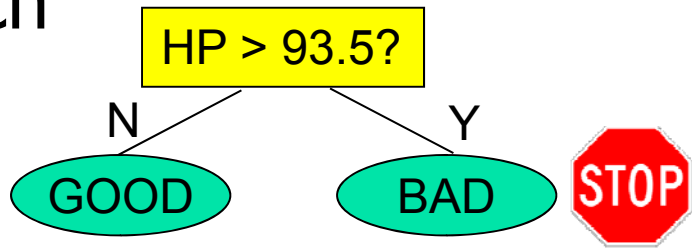
Split	Group Y	Group N	Gain
Cylinders = 4?	5+ / 4-	0+ / 11-	0.365
Cylinders = 6?	0+ / 6-	5+ / 9-	0.153
Cylinders = 8?	0+ / 5-	5+ / 10-	0.123
HP > 78?	2+ / 0-	3+ / 15-	0.226
HP > 87?	2+ / 2-	3+ / 13-	0.054
HP > 89.5?	3+ / 2-	2+ / 13-	0.144
HP > 91?	3+ / 3-	2+ / 12-	0.097
HP > 93.5?	**5+ / 3-**	**0+ / 12-**	**0.430**
Weight = light?	3+ / 3-	2+ / 12-	0.097
Weight = medium?	2+ / 6-	3+ / 9-	0.000
Weight = heavy?	0+ / 6-	5+ / 9-	0.153

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



(5 good, 3 bad) (0 good, 12 bad)

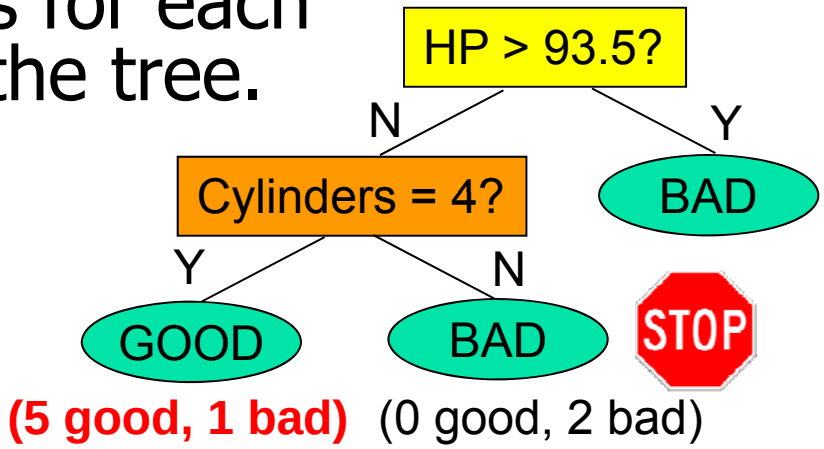
Split	Group Y	Group N	Gain
Cylinders = 4?	**5+ / 1-**	**0+ / 2-**	**0.467**
HP > 78?	2+ / 0-	3+ / 3-	0.204
HP > 87?	2+ / 2-	3+ / 1-	0.049
HP > 89.5?	3+ / 2-	2+ / 1-	0.003
HP > 91?	3+ / 3-	2+ / 0-	0.204
Weight = light?	3+ / 1-	2+ / 2-	0.049

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



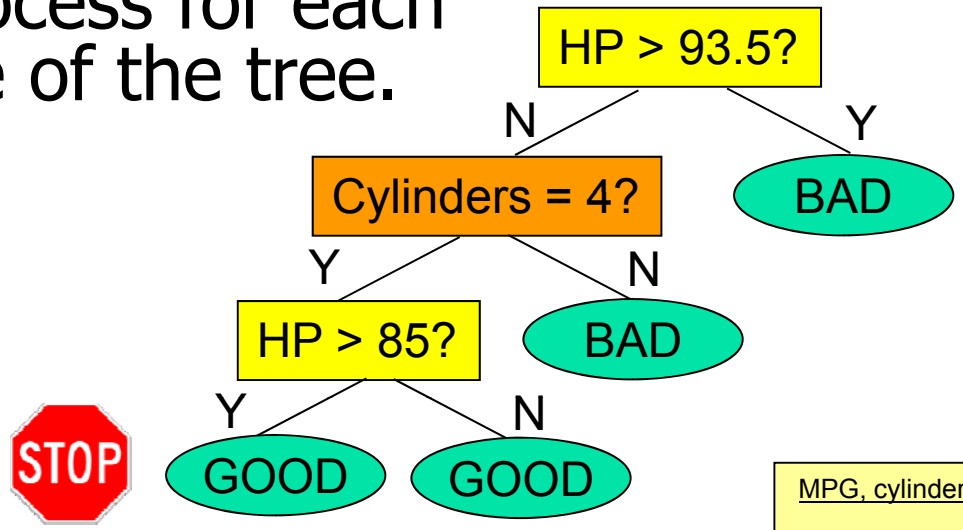
Split	Group Y	Group N	Gain
HP > 78?	3+ / 1-	2+ / 0-	0.109
HP > 85?	**3+ / 0-**	**2+ / 1-**	**0.191**
Weight = light?	3+ / 1-	2+ / 0-	0.109

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



(3 good, 0 bad) **(2 good, 1 bad)**

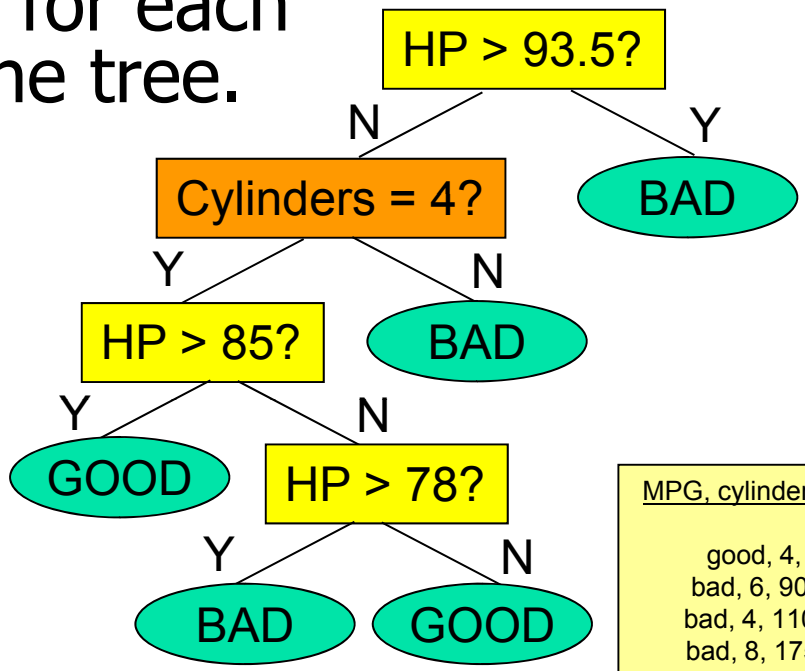
Split	Group Y	Group N	Gain
HP > 78?	**0+ / 1-**	**2+ / 0-**	**0.918**

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

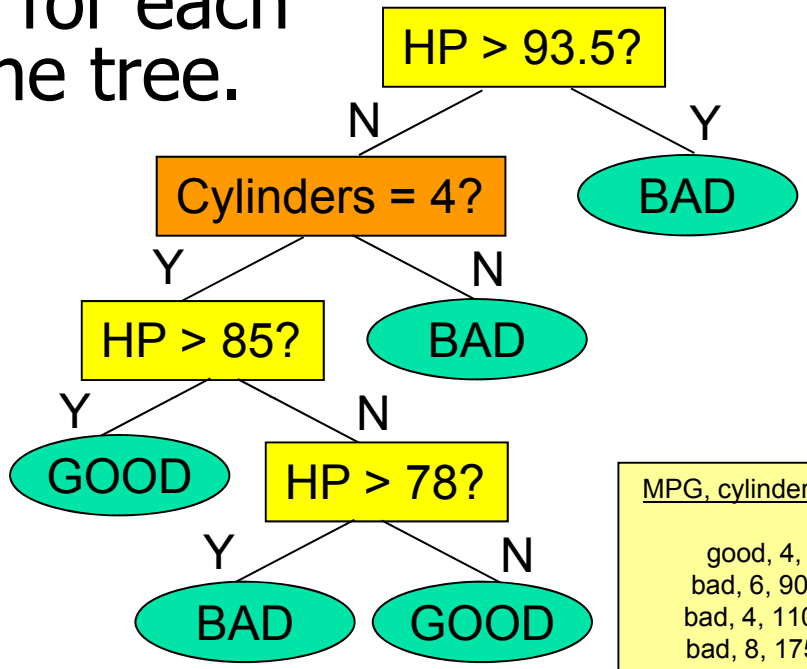
Group Y	Group N
A+ / B-	C+ / D-

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

Group Y	Group N
A+ / B-	C+ / D-

$$\text{Gain} = \frac{F((A+C),(B+D)) - F(A,B) - F(C,D)}{A+B+C+D}$$

where: $$F(X,Y) = X\log_2\frac{X+Y}{X} + Y\log_2\frac{X+Y}{Y}$$

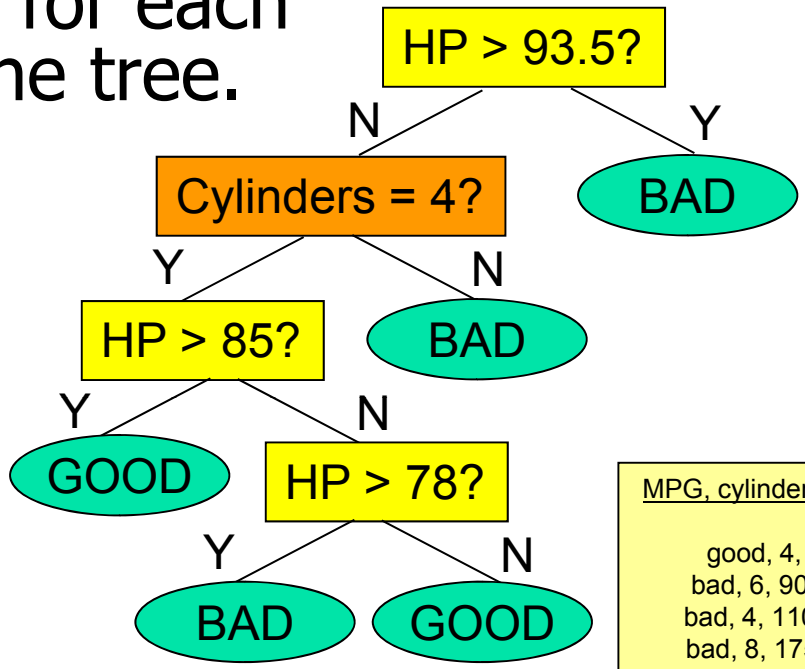
(You don't have to memorize this formula.)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

Group Y	Group N	Gain
3+ / 1-	6+ / 2-	0.000
8+ / 1-	1+ / 2-	0.204
9+ / 0-	0+ / 3-	0.811

Intuitively, the information gain is large when the proportions of positive examples in the two groups are very different, and zero when they are the same.

MPG, cylinders, HP, weight

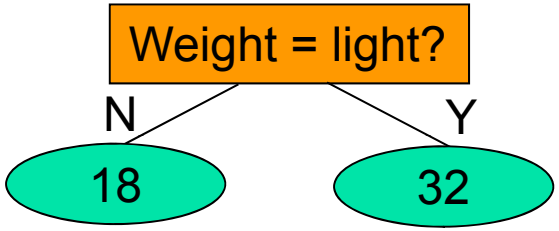
good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- If we are trying to predict a real-valued attribute (i.e. doing regression), minimize the sum of squared errors instead of maximizing information gain.

MPG, cylinders, HP, weight
32, 4, 75, light
20, 6, 95, medium
20, 4, 115, medium
14, 6, 95, medium

Split	Group Y	Group N	Total SSE
Cylinders = 4?	Predict: 26 SSE: 72	Predict: 17 SSE: 18	90
Weight = light?	**Predict: 32 SSE: 0**	**Predict: 18 SSE: 24**	**24**
HP > 85?	**Predict: 18 SSE: 24**	**Predict: 32 SSE: 0**	**24**
HP > 105?	Predict: 20 SSE: 0	Predict: 22 SSE: 168	168

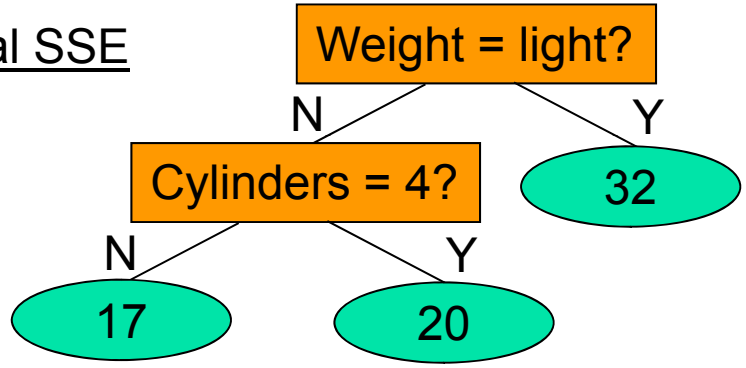


Choosing a decision rule

- If we are trying to predict a real-valued attribute (i.e. doing regression), minimize the sum of squared errors instead of maximizing information gain.

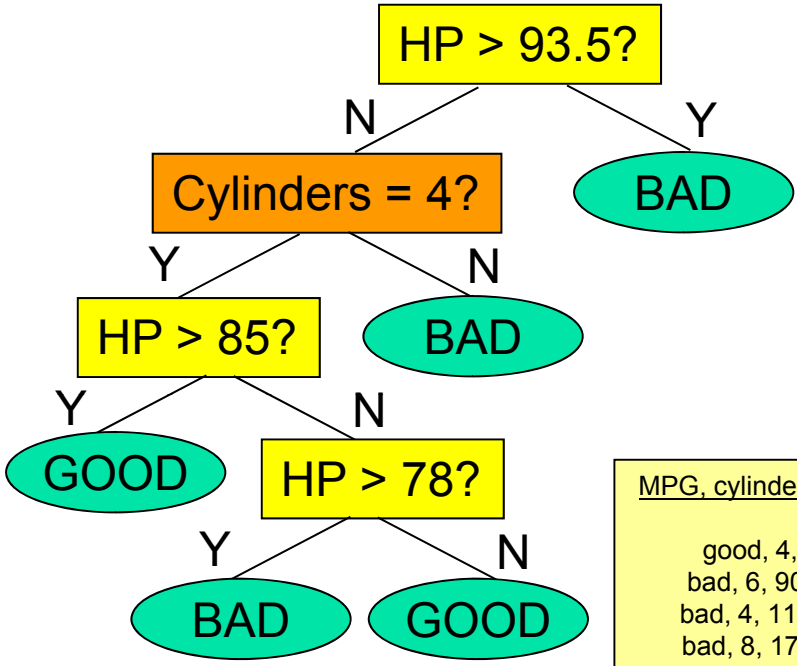
MPG, cylinders, HP, weight
32, 4, 75, light
20, 6, 95, medium
20, 4, 115, medium
14, 6, 95, medium

Split	Group Y	Group N	Total SSE
Cylinders = 4?	Predict: 20 SSE: 0	Predict: 17 SSE: 18	18
HP > 105?	Predict: 20 SSE: 0	Predict: 17 SSE: 18	18



Pruning decision trees to prevent overfitting

- Notice that the unpruned decision tree classifies every training example perfectly.
- This will always be the case (unless there are records with the same input values and different output values, as in the regression example).
- Does this mean we've found the best tree?



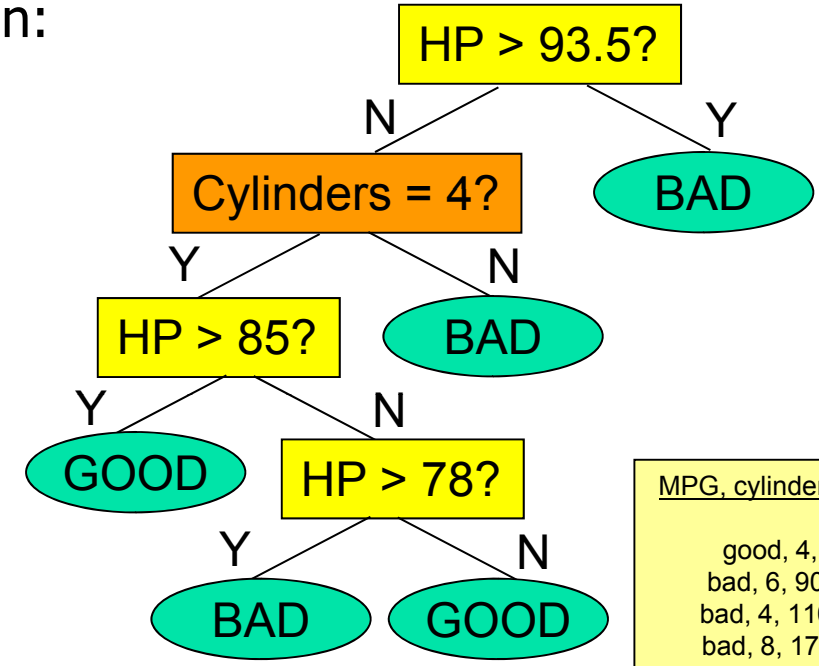
MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the "test set").
  - Learn a tree using the rest of the data (the "training set").
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

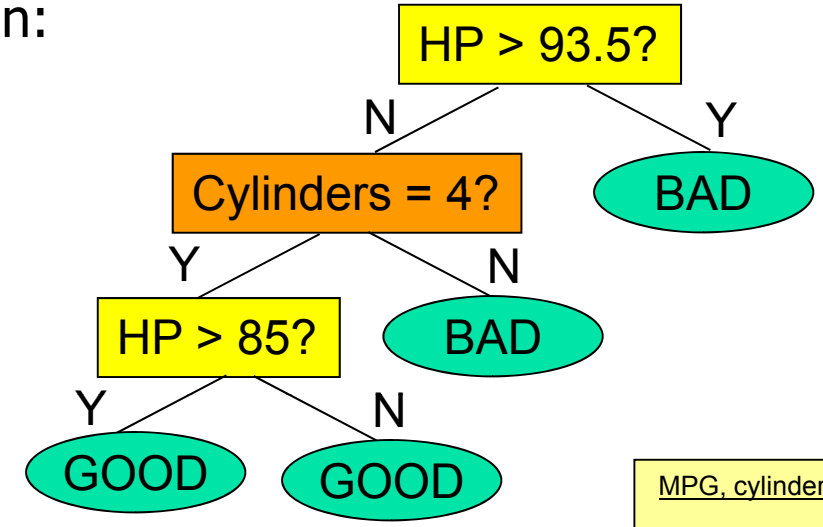
Training set (20 examples):
100% correct, 0% incorrect

Test set (100 examples):
86% correct, 14% incorrect

What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the "test set").
  - Learn a tree using the rest of the data (the "training set").
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

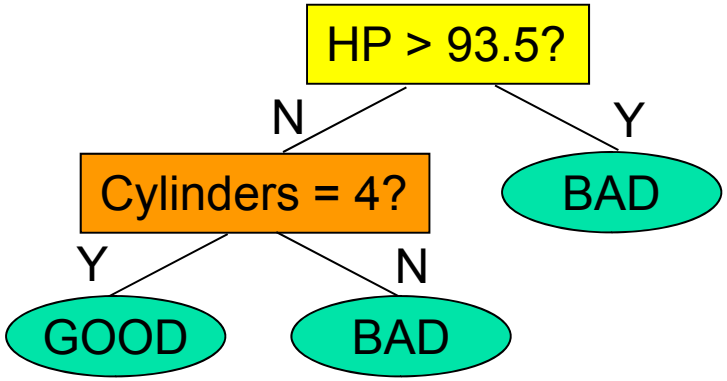
What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the "test set").
  - Learn a tree using the rest of the data (the "training set").
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

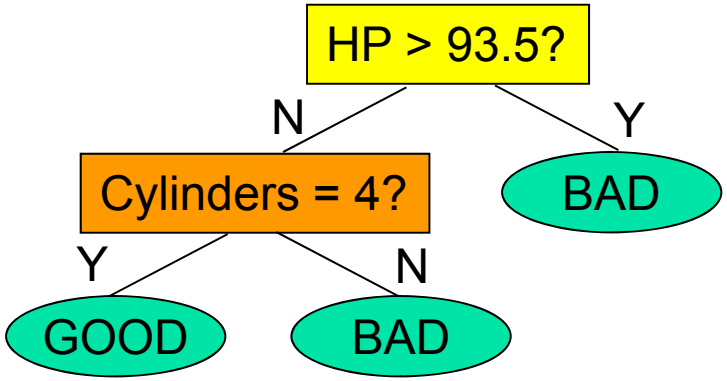
What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the "test set").
  - Learn a tree using the rest of the data (the "training set").
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

If we pruned Cylinders = 4, test set error would increase to 16%, so stop here!

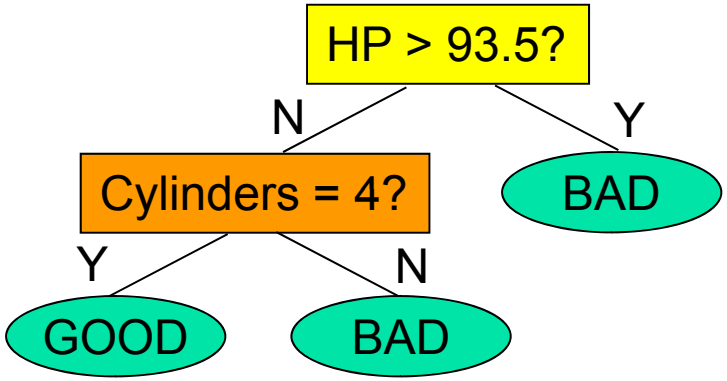
What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

MPG, cylinders, HP, weight

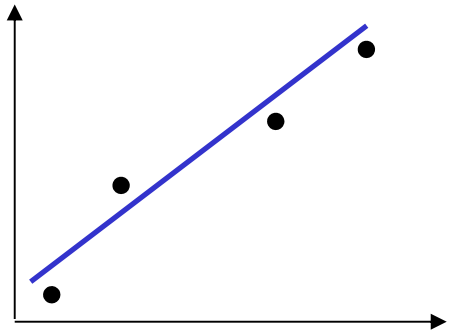
good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

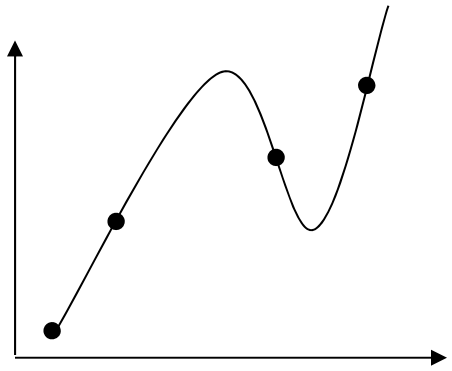
- Q: Why does pruning the tree reduce test set error?
- A: Because the unpruned tree was overfitting the training data (paying attention to parts of the data that are not relevant for prediction).



Here's another example of overfitting, this time for regression:



A line fits pretty well…



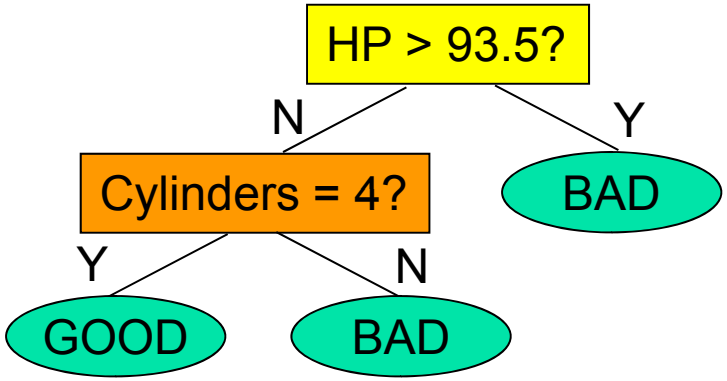
A cubic is probably overfitting.

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

- Q: What if we don't have enough data points?
- A: Another way to prevent overfitting is to do a certain kind of significance test (chi-squared) for each node, and prune any nodes that are not significant.



MPG, cylinders, HP, weight

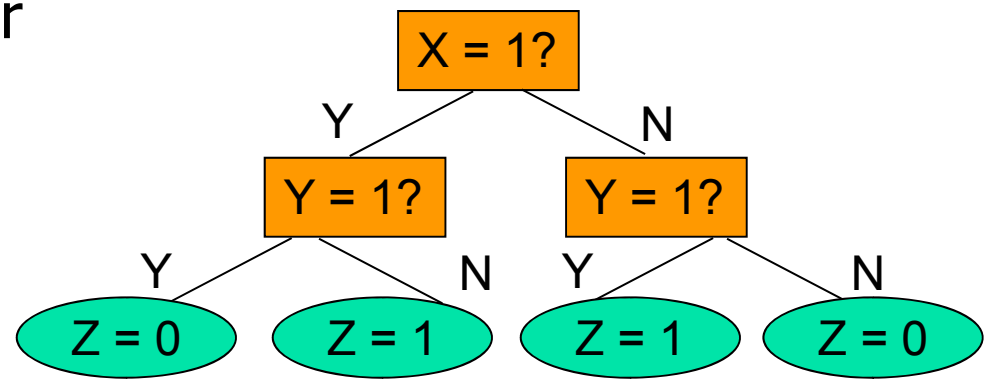
good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

- Q: Why bother building the whole tree, if we're just going to prune it?
- A: We could do significance testing while building the tree, but for many datasets, post-pruning gives better performance.

Consider the XOR function:

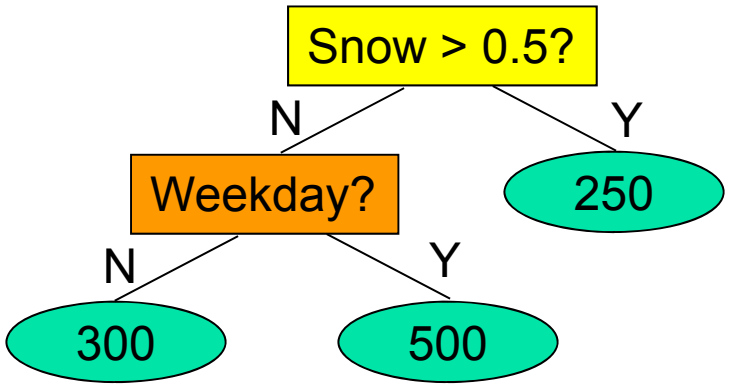
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

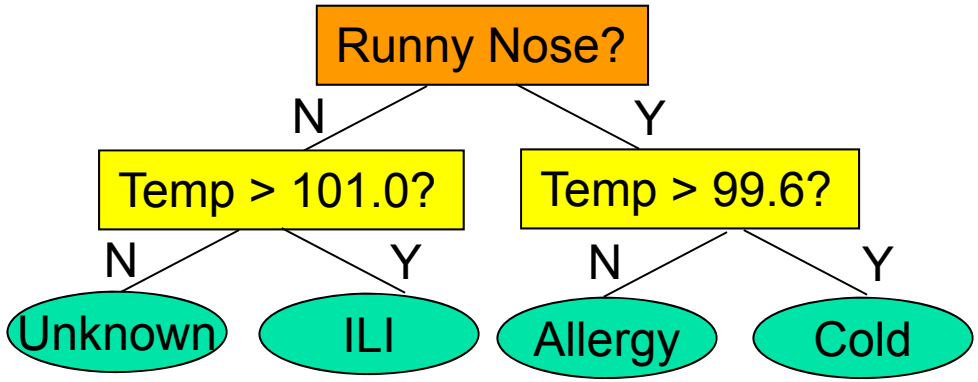


Each second level split has high information gain. The first level split has no information gain but is needed to make the second level splits possible.

Some advantages of decision trees

- Easy to learn the tree automatically from a dataset.
- Very easy to predict the target value given the tree.
- Generally good classification performance (though some fancier methods may do better, depending on the dataset).
- Can do both classification and regression, and can use both real and discrete inputs.
- Gives an idea of which variables are important in predicting the target value.
  - More important variables tend to be toward the top of the tree.
  - Unimportant variables are not included.





When to use decision trees?

- We have a dataset, with some attribute we want to predict.
  - We can do either classification or regression.
  - Datasets with lots of attributes, and/or lots of records, are okay.
  - Datasets with discrete or real values, or both, are okay.
- We want to be able to explain our prediction using a simple and interpretable set of rules.
  - We want to distinguish relevant from irrelevant attributes.
  - We want to provide a set of decision steps (e.g. "expert system" for medical diagnosis, don't want to perform irrelevant tests).
  - If all we care about is prediction accuracy and not interpretability, we might want to use some "black box" classifier instead (e.g. neural network, support vector machine).
- Performance is better when a tree structure makes sense.
  - We can learn non-linear functions of the input attributes.
  - But each decision step only considers one attribute at a time, so it's hard to learn non-stepwise functions accurately (e.g. $Z = X + Y$).

Advanced topic: combining classifiers

- On many datasets, we can achieve better predictions if, rather than learning a single decision tree, we learn many trees and then let them vote on the prediction.
  - One method (**bagging**): learn each tree from a randomly selected sample of the training set, and give each tree an equal vote.
  - Better method (**boosting**): learn each tree by reweighting the training set to emphasize examples that previous trees got wrong, then give "better" trees more voting power.
  - Disadvantage: harder to explain the prediction.

Some references

- L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*, Wadsworth, 1984.
- J.R. Quinlan. *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, 1993.
- T.M. Mitchell. *Machine Learning*, McGraw-Hill, 1997. (Chapter 3)
- T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*, Springer-Verlag, 2001.
- A.W. Moore. *Decision Trees*. Available at http://www.cs.cmu.edu/~awm/tutorials.

This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Large Scale Data Analysis for Policy 90-866, Fall 2012

Lecture 3: Instance-Based Learning (k-NN and Kernel Regression)

Instance-based learning

- So far, we've learned one way to do prediction: learn a (tree-structured) set of rules.
- Here's a remarkably simple alternative:
  - Choose a few instances from the dataset that are similar to the instance that we're trying to predict.

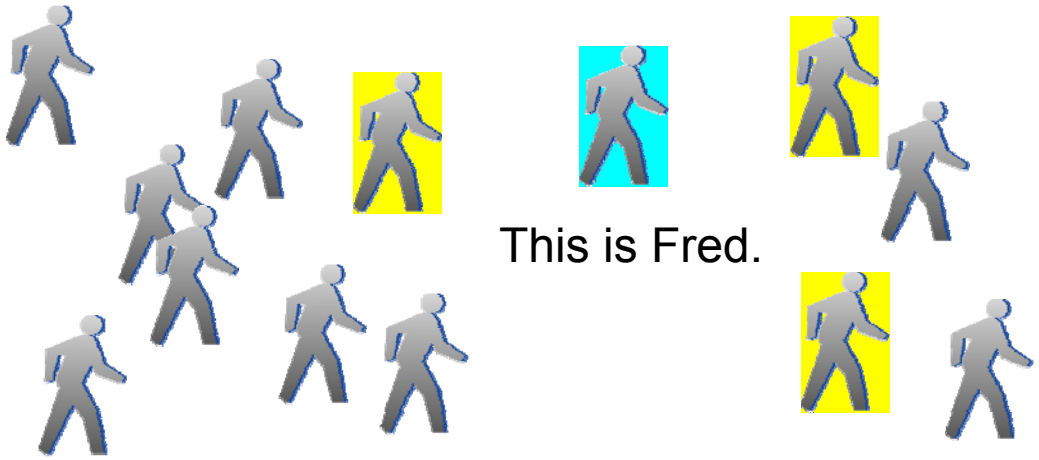
Example: Predicting Fred's income and his political party, based on demographics.

Here are a bunch of other people whose income and party are known.

Instance-based learning

- So far, we've learned one way to do prediction: learn a (tree-structured) set of rules.
- Here's a remarkably simple alternative:
  - Choose a few instances from the dataset that are similar to the instance that we're trying to predict.

Example: Predicting Fred's income and his political party, based on demographics.



John, Bob, and Alice are most like Fred.

J: Democrat, $60K
B: Republican, $70K
A: Democrat, $80K

So what about Fred?

Instance-based learning

- Also called memory-based, example-based, case-based, neighbor-based, similarity-based, local…
- To perform prediction in instance-based learning, we need to decide two things:
  - How to find "similar" examples in the dataset?

We need to define a distance metric $D(x_i, x_j)$ that gives the distance between any two data points x_i and x_j.

  - How to use these examples for prediction?

One answer: For a target data point x, choose the k points closest to x. Then predict the majority class (classification) or average (regression).

k-nearest neighbors (k-NN)

Distance metrics

- If data points have only real-valued attributes, use the Euclidean distance:

$$D(x_i, x_j) = \| x_i - x_j \| = \sqrt{\sum_{A_k} (v_{ik} - v_{jk})^2}$$

Example 1 (MPG dataset, 3 real-valued input attributes)

Car 1: weight = 2500, displacement = 250, horsepower = 100
Car 2: weight = 3000, displacement = 150, horsepower = 150
Car 3: weight = 3050, displacement = 250, horsepower = 100

Which car is more similar to Car 1?

$$D(x_1, x_2) = \sqrt{(2500 - 3000)^2 + (250 - 150)^2 + (100 - 150)^2} \approx 512$$

$$D(x_1, x_3) = \sqrt{(2500 - 3050)^2 + (250 - 250)^2 + (100 - 100)^2} = 550$$

Distance metrics

- If data points have only real-valued attributes, use the Euclidean distance:

$$D(x_i, x_j) = \| x_i - x_j \| = \sqrt{\sum_A (x_{i,k} - x_{j,k})^2}$$

Maybe we should scale the attributes differently!

Let's normalize each attribute value by subtracting the mean value of that attribute then dividing by the standard deviation.

$D(x_1, x_2) = \sqrt{(2500 - 3000)^2 + (250 - 150)^2 + (100 - 150)^2} \approx 512$

$D(x_1, x_3) = \sqrt{(2500 - 3050)^2 + (250 - 250)^2 + (100 - 100)^2} = 550$

Distance metrics

- If data points have only real-valued attributes, use the Euclidean distance:

$$D(x_i, x_j) = \| x_i - x_j \| = \sqrt{\sum_{A_k} (v_{ik} - v_{jk})^2}$$

Example 2 (MPG dataset, 3 scaled real-valued input attributes)

Car 1: weight = -0.20, displacement = 0.32, horsepower = -0.51
Car 2: weight = 0.75, displacement = -0.90, horsepower = -0.19
Car 3: weight = 0.84, displacement = 0.32, horsepower = -0.51

For example, Car 3's weight is 0.84 standard deviations higher than the mean weight of the entire dataset.

Now Car 3 is more similar to Car 1:

$D(x_1, x_2) = 1.58$
$D(x_1, x_3) = 1.04$

Distance metrics

- If data points have only discrete-valued attributes, use the Hamming distance:

  $D(x_i, x_j)$ = # of attributes for which x_i and x_j are different.

Example 3 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium
Car 2: cylinders = 4, origin = asia, weight = heavy
Car 3: cylinders = 6, origin = europe, weight = heavy

$D(x_1, x_2) = 2$ $D(x_1, x_3) = 3$ $D(x_2, x_3) = 2$

What about datasets with both discrete and real attributes?
(Hint: Hamming $D(x_i, x_j) = \Sigma_k$ {1 if $v_{ik} = v_{jk}$, 0 otherwise})

Distance metrics

- If data points have only discrete-valued attributes, use the Hamming distance:

  $D(x_i, x_j)$ = # of attributes for which x_i and x_j are different.

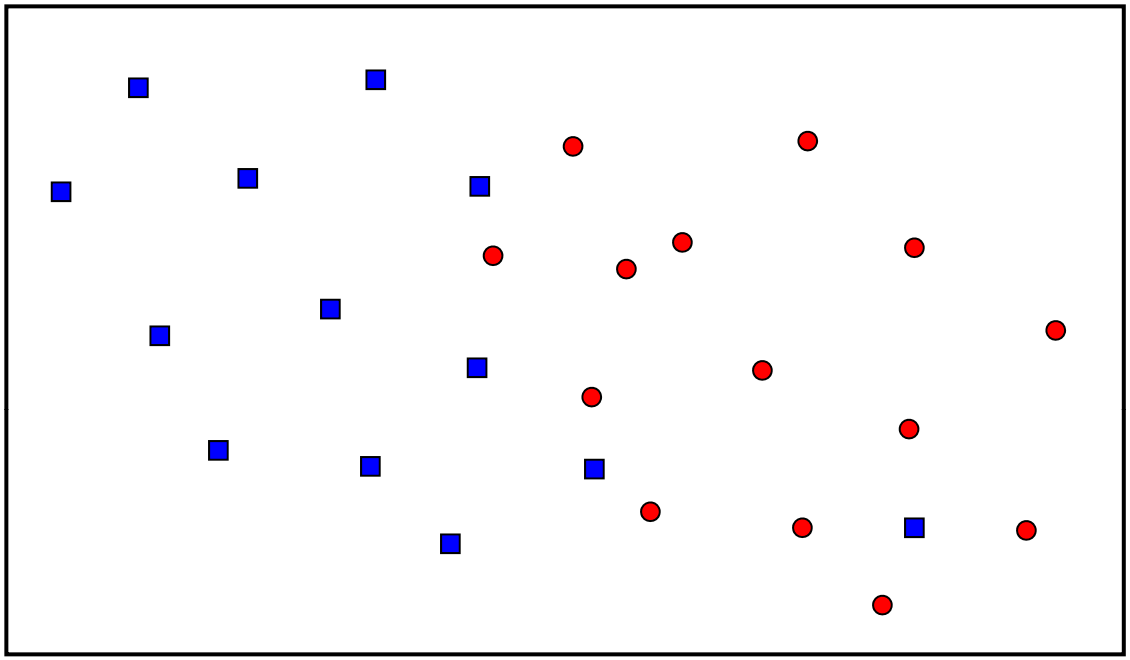
Example 3 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium
Car 2: cylinders = 4, origin = asia, weight = heavy
Car 3: cylinders = 6, origin = europe, weight = heavy

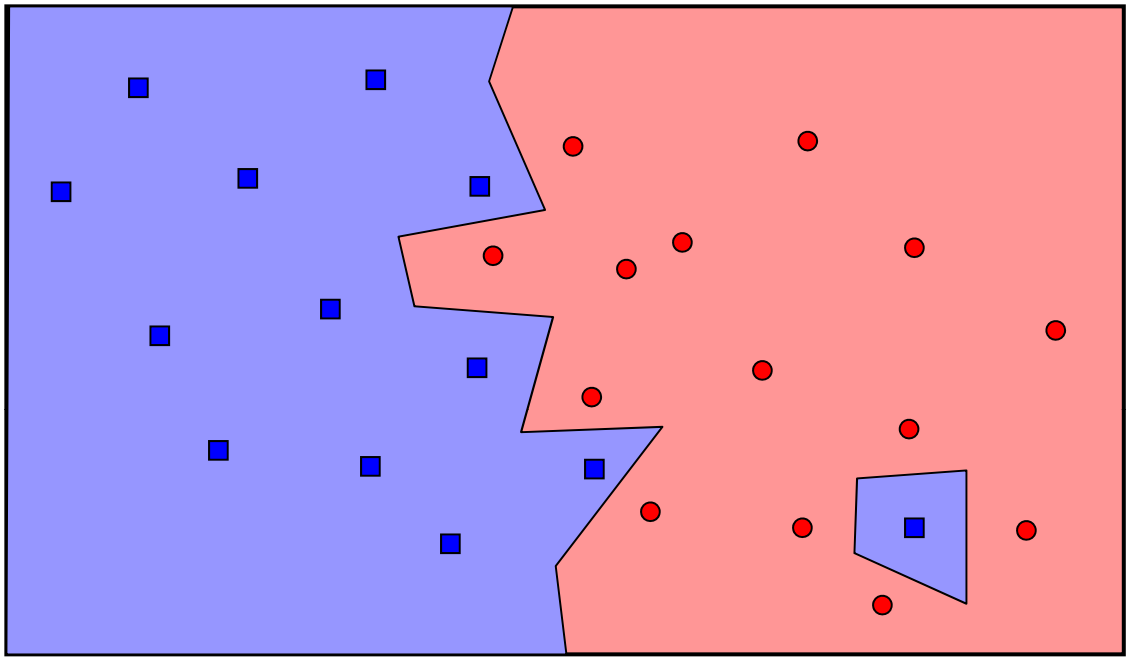
$D(x_1, x_2) = 2$ $D(x_1, x_3) = 3$ $D(x_2, x_3) = 2$

Many other distance metrics are possible… and several recent papers suggest ways of learning a good distance metric from data.

Example: 1-NN classification in 2D



Example: 1-NN classification in 2D

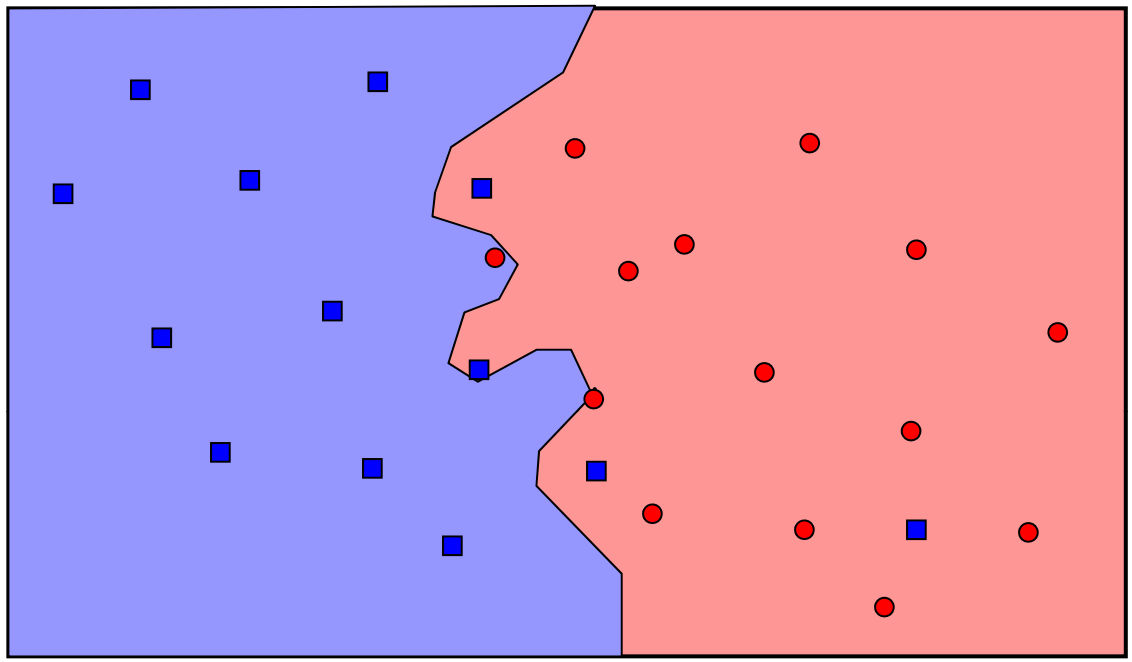


Notice that 1-NN fits the training data perfectly. Is this a good thing?

Yes: can learn sharp decision boundaries for complex functions.
No: will overfit the data if there is noise, outliers, or overlap between classes.

Let's try a larger value of k…

Example: 3-NN classification in 2D



Notice that 3-NN ignores the outlier point, and has a smoother boundary.

Larger values of k tend to have better test set performance when the data is noisy, if we have enough data points.

It's not good if k is too large: what would happen for 27-NN?

How to choose k?

- The value of k (number of neighbors to use) must be specified in advance.
- But we can consider multiple values of k and then choose the best one.

Option 1: Choose the value of k with lowest error on the training data.

BAD IDEA! We will always choose k = 1 and overfit the training data.

Option 2: Hold out a separate set of test data, and choose the value of k with lowest error on the test data.

This is OK, but it still wastes data (typically 2/3 train, 1/3 test).

So how can we avoid overfitting without wasting so much data?

The solution: cross-validation

- Randomly partition the data into R groups.
- For each group:
  - Train the model on the other R – 1 groups.
  - Test on the held out group (compute proportion correct/incorrect)
- Compute average error over the R groups. This is called **R-fold cross-validation error**, and is a good measure of prediction performance on unseen data.

Typically R = 10 or 20.

Advantages: Each "fold" uses almost all of the data. Averaging across multiple folds reduces variance.

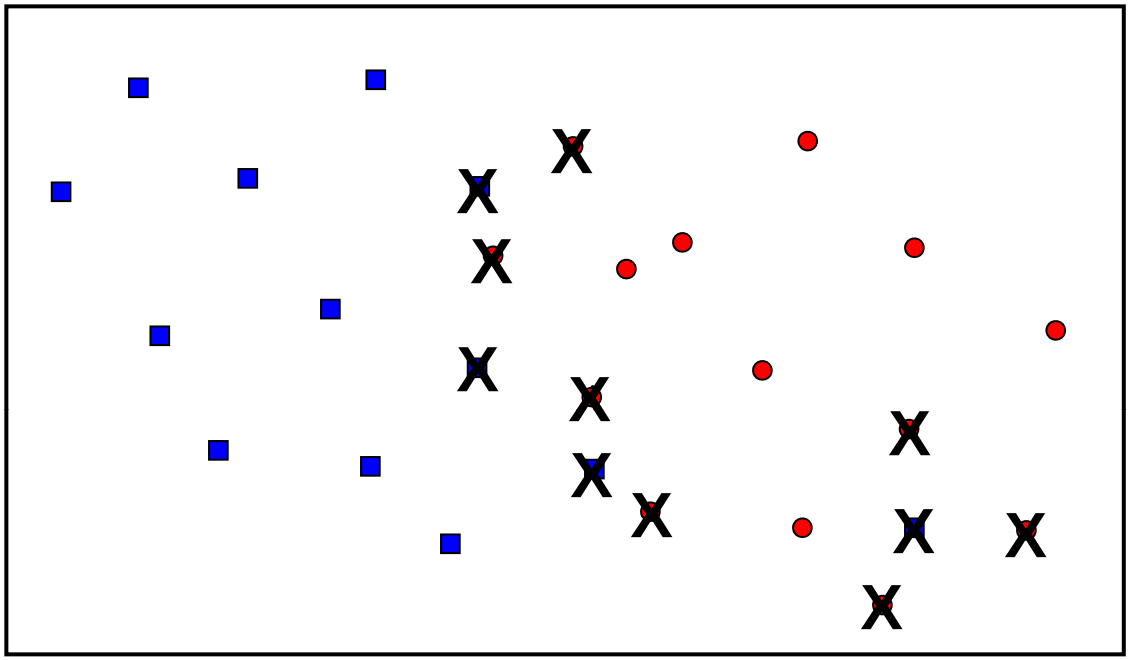
Disadvantage: More computationally expensive (we must learn and evaluate R different models).

Extreme case: Let number of groups = number of data points. This is called **leave-one-out cross-validation**.

But wouldn't that be computationally infeasible?

Not for k-nearest neighbors!!!

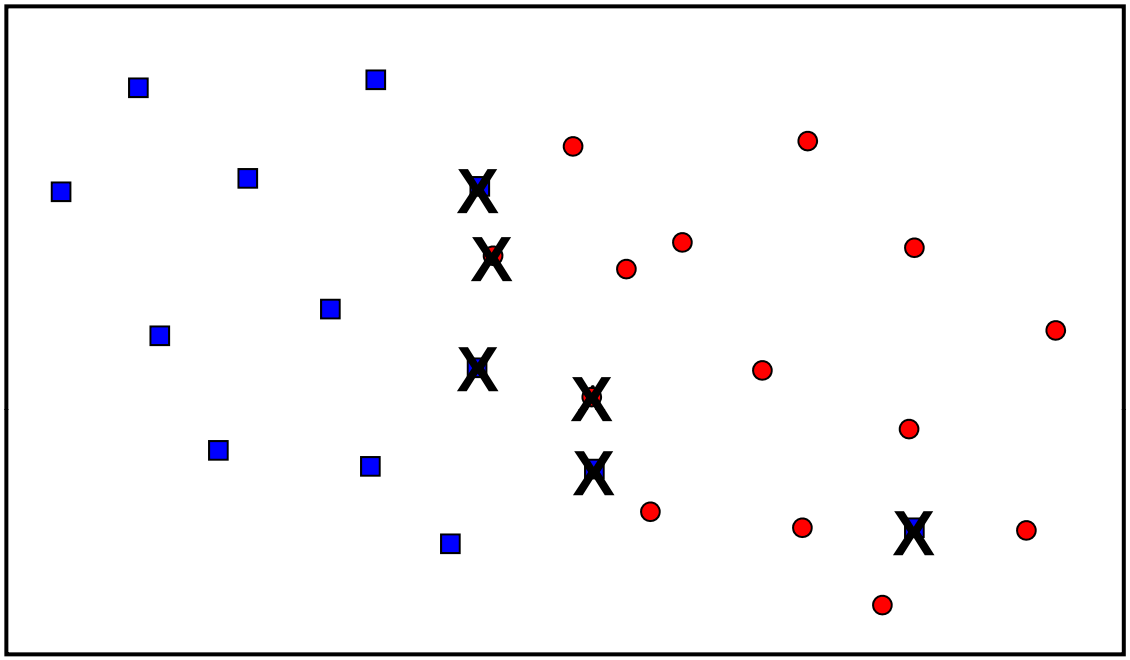
Leave-one-out cross-validation



To do LOOCV for k-nearest neighbor: classify each data point using all other points, and count the proportion classified correctly.

1-NN: 16 / 27 correct

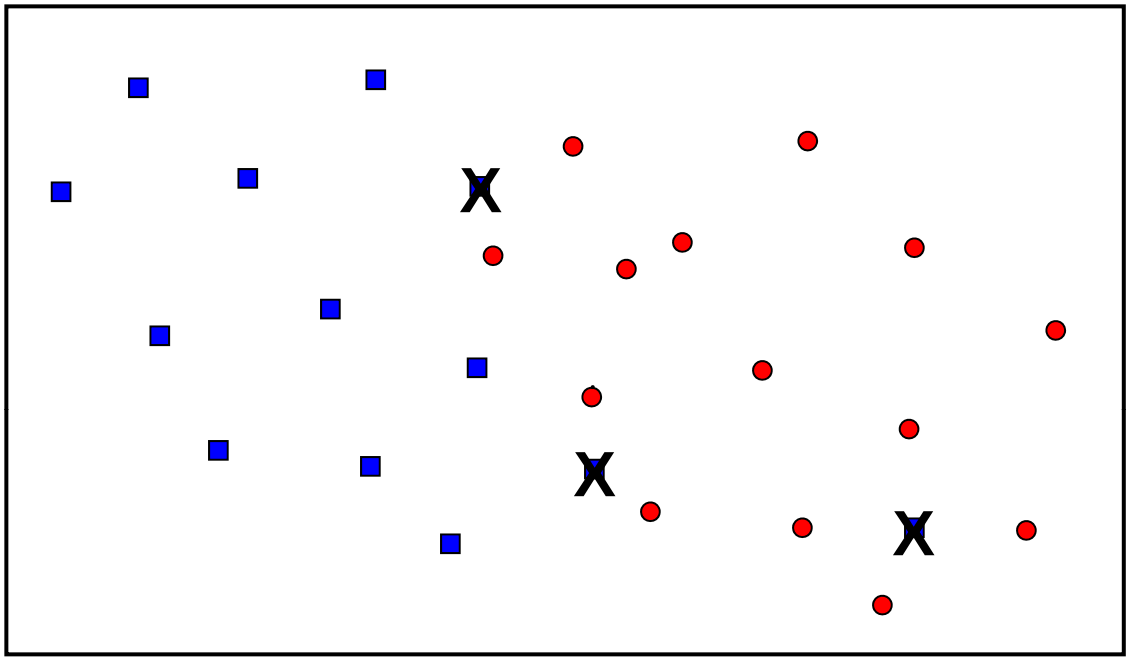
Leave-one-out cross-validation



To do LOOCV for k-nearest neighbor: classify each data point using all other points, and count the proportion classified correctly.

1-NN: 16 / 27 correct
3-NN: 21 / 27 correct

Leave-one-out cross-validation



To do LOOCV for k-nearest neighbor: classify each data point using all other points, and count the proportion classified correctly.

1-NN: 16 / 27 correct
3-NN: 21 / 27 correct
5-NN: 23 / 27 correct
7-NN: 24 / 27 correct
9-NN: 23 / 27 correct

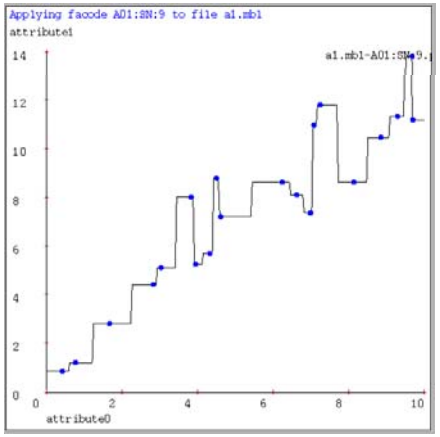
11-NN: 23 / 27 correct
13-NN: 22 / 27 correct
15-NN: 23 / 27 correct
17-NN: 22 / 27 correct
19-NN: 23 / 27 correct

21-NN: 23 / 27 correct
23-NN: 21 / 27 correct
25-NN: 12 / 27 correct

Instance-based learning for regression

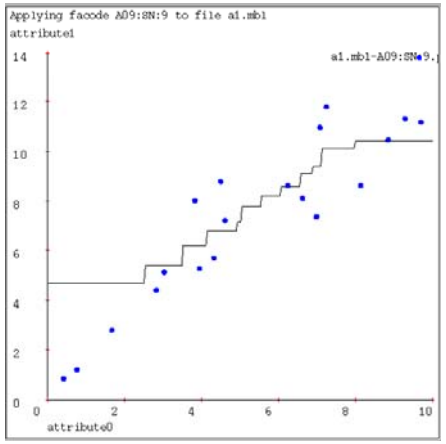
- For k-NN, instead of predicting the majority class, we predict the average value of the k nearest neighbors.
- Another option: use all points for prediction, but weight each one proportional to some rapidly decreasing function of distance (e.g. exponential, Gaussian).

kernel regression



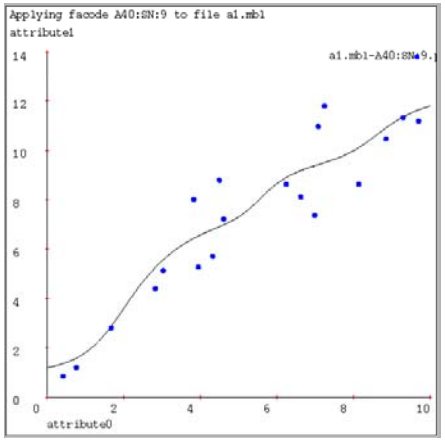
1-NN

(very bumpy, fits noise)



9-NN

(bumpy, smeared ends)



KR

(nice, smooth prediction)

(Thanks to Andrew Moore for these examples!)

Example: kernel regression

- Let's go back to the MPG example, and attempt to predict Car 1's MPG using Car 2 and Car 3.
  - Hamming distance: $D(x_1, x_2) = 2$, and $D(x_1, x_3) = 3$.
- Assume an exponential kernel with bandwidth b: each point is weighted proportional to $e^{-D/b}$.

Example 4 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium, mpg = ???
Car 2: cylinders = 4, origin = asia, weight = heavy, mpg = 20
Car 3: cylinders = 6, origin = europe, weight = heavy, mpg = 10

Let's predict Car 1's MPG, assuming a bandwidth of b = 1.

Relative weight of Car 2 = e^{-2} = .135
Relative weight of Car 3 = e^{-3} = .050
Estimate for MPG of Car 1 = (.135 x 20 + .050 x 10) / (.135 + .050) = 17.31

Example: kernel regression

- Let's go back to the MPG example, and attempt to predict Car 1's MPG using Car 2 and Car 3.
  - Hamming distance: $D(x_1, x_2) = 2$, and $D(x_1, x_3) = 3$.
- Assume an exponential kernel with bandwidth b: each point is weighted proportional to $e^{-D/b}$.

Example 4 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium, mpg = ???
Car 2: cylinders = 4, origin = asia, weight = heavy, mpg = 20
Car 3: cylinders = 6, origin = europe, weight = heavy, mpg = 10

What happens if we increase the bandwidth to b = 2?

Relative weight of Car 2 = e^{-1} = .368
Relative weight of Car 3 = $e^{-3/2}$ = .223
Estimate for MPG of Car 1 = (.368 x 20 + .223 x 10) / (.368 + .223) = 16.22

Example: kernel regression

- Let's go back to the MPG example, and attempt to predict Car 1's MPG using Car 2 and Car 3.
  - Hamming distance: $D(x_1, x_2) = 2$, and $D(x_1, x_3) = 3$.
- Assume an exponential kernel with bandwidth b: each point is weighted proportional to $e^{-D/b}$.

Example 4 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium, mpg = ???
Car 2: cylinders = 4, origin = asia, weight = heavy, mpg = 20
Car 3: cylinders = 6, origin = europe, weight = heavy, mpg = 10

Extreme case 1: If bandwidth is very large, all points have equal weights.
Estimate for MPG of Car 1 = 15.

Extreme case 2: If bandwidth is very small, almost all weight is on the 1-NN.
Estimate for MPG of Car 1 = 20.

Example: kernel regression

- Let's go back to the MPG example, and attempt to predict Car 1's MPG using Car 2 and Car 3.
  - Hamming distance: $D(x_1, x_2) = 2$, and $D(x_1, x_3) = 3$.
- Assume an exponential kernel with bandwidth b: each point is weigh

We can choose a good bandwidth by cross-validation. This is just like choosing k for k-NN, but we minimize the sum of squared errors.

Extreme case 1: If bandwidth [illegible] points have equal weights.
Estimate for MPG of Car 1 = 15.

Extreme case 2: If bandwidth is very small, almost all weight is on the 1-NN.
Estimate for MPG of Car 1 = 20.

Example: kernel regression

- Let's go back to the MPG example, and attempt to predict Car 1's MPG using Car 2 and Car 3.
  - Hamming distance: $D(x_1, x_2) = 2$, and $D(x_1, x_3) = 3$.
- Assume an exponential kernel with bandwidth b: each point is weighted proportional to $e^{-D/b}$.

Example 4 (MPG dataset, 3 discrete-valued input attributes)

Car 1: cylinders = 4, origin = america, weight = medium, mpg = ???
Car 2: cylinders = 4, origin = asia, weight = heavy, mpg = 20
Car 3: cylinders = 6, origin = europe, weight = heavy, mpg = 10

Kernel regression is just a locally weighted average; in fact, many other regression techniques can also be locally weighted.

One popular choice is locally weighted linear regression: see Andrew Moore's lecture on "Memory-Based Learning" for details.

Advantages of instance-based learning

- Easy to implement and apply to many problems.
  - Even complex structures such as time series or images can be predicted: you just need a distance metric!
  - Since no global model is learned, training and cross-validation are very fast.
- Can easily learn complex functions.
  - By combining many local models, we can approximate even very complex global models.
- Can explain a prediction by presenting the set of k-nearest neighbors.

Disadvantages of instance-based learning

- No models, rules, or overview of the dataset.
  - Hard to gain an overall understanding of prediction.
- Needs the entire training set to make a prediction.
  - Requires a lot of memory.
  - Prediction is slow for large training sets (run time is proportional to the number of training examples).
- Must choose parameters (k, bandwidth) carefully:
  - If too local (e.g. 1-NN), may overfit the noise.
  - If not local enough (e.g. all-NN), may lose the signal.
- The "curse of dimensionality":
  - As number of dimensions increases, average distance between data points becomes more uniform, and neighbors do not provide much information.

When to use instance-based learning?

- We have a dataset, a distance metric, and some attribute we want to predict.
  - We can do either classification or regression.
  - Datasets with discrete or real values, or both, are okay.
- We want to be able to explain each prediction by showing similar examples (but we don't need decision rules or a model of the data).
- Performance is best for low-dimensional data and when we have sufficiently many training examples.

Use instance-based learning when you have lots of data, not many input variables, and you expect a very complex non-linear function of the data.

Some references

- T.M. Mitchell. *Machine Learning*, McGraw Hill, 1997. (Chapter 8)
- A.W. Moore. *Memory-based learning*, available at: http://www.cs.cmu.edu/~awm/tutorials
- Jerry Zhu has a nice k-NN demo applet, available at: http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html
- Vizier software for locally weighted learning, available at: http://www.cs.cmu.edu/~awm/vizier/

Large Scale Data Analysis for Policy 90-866, Fall 2012

Lecture 4: Model-Based Learning (Bayesian Classifiers)

Model-based learning

- Now we've seen two ways to predict unknown values in our dataset:
  - Learning a set of decision rules. ← Rule-based learning
  - Using "similar" data points. ← Instance-based learning
- Let's focus on classification, and ask the question, "How likely is this data point to be in each class?"
- This question is hard to answer accurately using the methods that we've seen so far.

Solution: learn a **probabilistic model** for each class, then compute the class posterior probabilities using Bayes' Theorem.

Using Bayes' Theorem

- Posterior probability that data record x_i belongs to class C_j:

Likelihood of record x_i's attribute values if it belongs to class C_j

$$\Pr(C_j \mid x_i) = \frac{\Pr(x_i \mid C_j)\Pr(C_j)}{\sum_{C_k} \Pr(x_i \mid C_k)\Pr(C_k)}$$

Prior probability of class C_j

Normalize probabilities (must sum to 1)

- Simple example: {has stripes, doesn't swim}

Class	Prior	Likelihood	Unnormalized posterior
Horse	Pr(Horse) = 0.4	Pr(stripes, no swim | Horse) = 0.050	0.4 x 0.050 = 0.020
Zebra	Pr(Zebra) = 0.1	Pr(stripes, no swim | Zebra) = 0.950	0.1 x 0.950 = 0.095
Fish	Pr(Fish) = 0.5	Pr(stripes, no swim | Fish) = 0.002	0.5 x 0.002 = 0.001
			0.116

Using Bayes' Theorem

- Posterior probability that data record x_i belongs to class C_j:

Likelihood of record x_i's attribute values if it belongs to class C_j

$$\Pr(C_j \mid x_i) = \frac{\Pr(x_i \mid C_j)\Pr(C_j)}{\sum_{C_k} \Pr(x_i \mid C_k)\Pr(C_k)}$$

Prior probability of class C_j

Normalize probabilities (must sum to 1)

- Simple example: {has stripes, doesn't swim}

Class	Prior	Likelihood	**Normalized posterior**
Horse	Pr(Horse) = 0.4	Pr(stripes, no swim | Horse) = 0.050	**0.020 / 0.116 = 0.172**
Zebra	Pr(Zebra) = 0.1	Pr(stripes, no swim | Zebra) = 0.950	**0.095 / 0.116 = 0.819**
Fish	Pr(Fish) = 0.5	Pr(stripes, no swim | Fish) = 0.002	**0.001 / 0.116 = 0.009**

Using Bayes' Theorem

- Posterior probability that data record x_i belongs to class C_j:

Likelihood of record x_i's attribute values if it belongs to class C_j

$$\Pr(C_j \mid x_i) = \frac{\Pr(x_i \mid C_j)\Pr(C_j)}{\sum_{C_k} \Pr(x_i \mid C_k)\Pr(C_k)}$$

Prior probability of class C_j

Normalize probabilities (must sum to 1)

So how can we obtain the priors and likelihoods?

1. Prior knowledge (ask a domain expert).
2. **Learn** the priors and likelihoods from a representative "training" dataset. →

200 horses, 50 zebras, 250 fish

Pr(horse) = 200 / 500 = 0.4
Pr(zebra) = 50 / 500 = 0.1
Pr(fish) = 250 / 500 = 0.5

Using Bayes' Theorem

- Posterior probability that data record x_i belongs to class C_j:

Likelihood of record x_i's attribute values if it belongs to class C_j

$$\Pr(C_j \mid x_i) = \frac{\Pr(x_i \mid C_j)\Pr(C_j)}{\sum_{C_k} \Pr(x_i \mid C_k)\Pr(C_k)}$$

Prior probability of class C_j

Normalize probabilities (must sum to 1)

So how can we obtain the priors and likelihoods?

1. Prior knowledge (ask a domain expert).
2. **Learn** the priors and likelihoods from a representative "training" dataset.

Dataset of N records

$$\Pr(C_k) = \frac{\#\{\text{class} = C_k\}}{N}$$

("Maximum likelihood estimation")

Using Bayes' Theorem

- Posterior probability that data record x_i belongs to class C_j:

Likelihood of record x_i's attribute values if it belongs to class C_j

$$\Pr(C_j \mid x_i) = \frac{\Pr(x_i \mid C_j)\Pr(C_j)}{\sum_{C_k} \Pr(x_i \mid C_k)\Pr(C_k)}$$

Prior probability of class C_j

Normalize probabilities (must sum to 1)

So how can we obtain the priors and likelihoods?

To learn the likelihoods $\Pr(x_i \mid C_k)$, we need a model of how the data is generated, for each class C_k.

In this lecture, we will consider one very simple (but very useful) way of building these models.

"naïve Bayes"

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = High)
Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = Low)

Pr(Gender, BMI, Diabetes | Risk) = Pr(Gender | Risk) Pr(BMI | Risk) Pr(Diabetes | Risk)

Naïve Bayes assumption: All attribute values are conditionally independent given the class.

$$\Pr(x_i \mid C_k) = \prod_{j=1..J} \Pr(A_j = v_{ij} \mid C = C_k)$$

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

$$\Pr(\text{Gender} = \text{Male}, \text{BMI} = 26, \text{Diabetes} = \text{Yes} \mid \text{Risk} = \text{High})$$
$$\Pr(\text{Gender} = \text{Male}, \text{BMI} = 26, \text{Diabetes} = \text{Yes} \mid \text{Risk} = \text{Low})$$

$$\Pr(\text{Gender}, \text{BMI}, \text{Diabetes} \mid \text{Risk}) = \Pr(\text{Gender} \mid \text{Risk}) \Pr(\text{BMI} \mid \text{Risk}) \Pr(\text{Diabetes} \mid \text{Risk})$$

	$\Pr(A_j = v_{ij} \mid C = C_k)$			$\Pr(x_i \mid C = C_k)$
Class	Gender = Male	BMI = 26	Diabetes = Yes	Total likelihood
Risk = Low	0.44	0.01	0.03	1.32×10^{-4}
Risk = High	0.53	0.02	0.45	4.77×10^{-3}

$\Pr(\text{Gender} = \text{Male} \mid \text{Risk} = \text{High}) = 0.53$

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

$$\Pr(\text{Gender} = \text{Male}, \text{BMI} = 26, \text{Diabetes} = \text{Yes} \mid \text{Risk} = \text{High})$$
$$\Pr(\text{Gender} = \text{Male}, \text{BMI} = 26, \text{Diabetes} = \text{Yes} \mid \text{Risk} = \text{Low})$$

$$\Pr(\text{Gender}, \text{BMI}, \text{Diabetes} \mid \text{Risk}) = \Pr(\text{Gender} \mid \text{Risk}) \Pr(\text{BMI} \mid \text{Risk}) \Pr(\text{Diabetes} \mid \text{Risk})$$

Now we can use Bayes' Theorem:

Class	Total likelihood	Prior	Unnormalized posterior	Posterior
Risk = Low	1.32×10^{-4}	0.9	1.19×10^{-4}	1.19 / 5.96 = 0.2
Risk = High	4.77×10^{-3}	0.1	4.77×10^{-4}	4.77 / 5.96 = 0.8
			5.96×10^{-4}	

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = High)
Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = Low)

Pr(Gender, BMI, Diabetes | Risk) = Pr(Gender | Risk) Pr(BMI | Risk) Pr(Diabetes | Risk)

Finally, we can use the posteriors to make decisions, given costs of each outcome:

Class	Posterior	Cost(Treat)	Cost(Don't treat)
Risk = Low	0.2	$10,000	$1,000
Risk = High	0.8	$10,000	$100,000
Choose lower expected cost:		$10,000	$80,200

0.8($100,000) + 0.2($1,000) = $80,200

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = High)
Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = Low)

Pr(Gender, BMI, Diabetes | Risk) = Pr(Gender | Risk) Pr(BMI | Risk) Pr(Diabetes | Risk)

Question 1: How to estimate the conditional probability of a discrete attribute?

$$\Pr(\text{Gender} = \text{Male} \mid \text{Risk} = \text{High}) = \frac{\#\{\text{Gender} = \text{Male AND Risk} = \text{High}\}}{\#\{\text{Risk} = \text{High}\}}$$

For example, if there were 500 training records with Risk = High, and 265 of these also had Gender = Male, our probability estimate would be 265 / 500 = 0.53.

Model-based classification

Dataset representation: records x_i, attributes A_j, values v_{ij}

Gender	BMI	Diabetes?	Heart attack risk
Male	26	Yes	**???**

To decide whether the patient's risk is high or low, we must first compute the likelihood of seeing his attribute values for both high-risk and low-risk groups:

Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = High)
Pr(Gender = Male, BMI = 26, Diabetes = Yes | Risk = Low)

Pr(Gender, BMI, Diabetes | Risk) = Pr(Gender | Risk) Pr(BMI | Risk) Pr(Diabetes | Risk)

Question 2: How to estimate the conditional probability* of a real-valued attribute?

$$f(\text{BMI} = 26 \mid \text{Risk} = \text{High}) = ???$$

Solution: learn a Gaussian distribution from all of the training records with Risk = High, and use this distribution to estimate the probability.

← This is called Gaussian Naïve Bayes classification.

*Technically, we are estimating the probability density f(x), not the probability of drawing x.

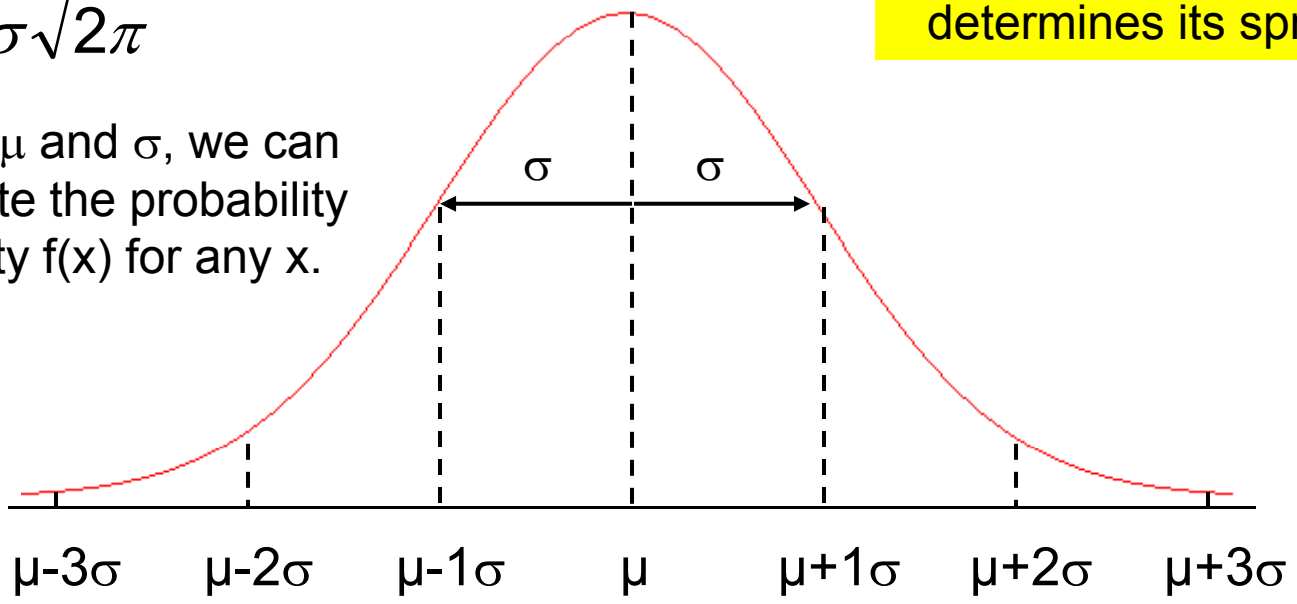
Review of the Gaussian distribution

- Also called "normal distribution" or "bell curve".
- A good approximation for many real-world distributions.
  - Central Limit Theorem: The sum (or mean) of sufficiently many i.i.d. samples from any distribution is approximately Gaussian.
- A symmetric, unimodal distribution N(μ, σ), determined by its mean μ and standard deviation σ:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ determines the center of the distribution, and σ determines its spread.

Given μ and σ, we can compute the probability density f(x) for any x.



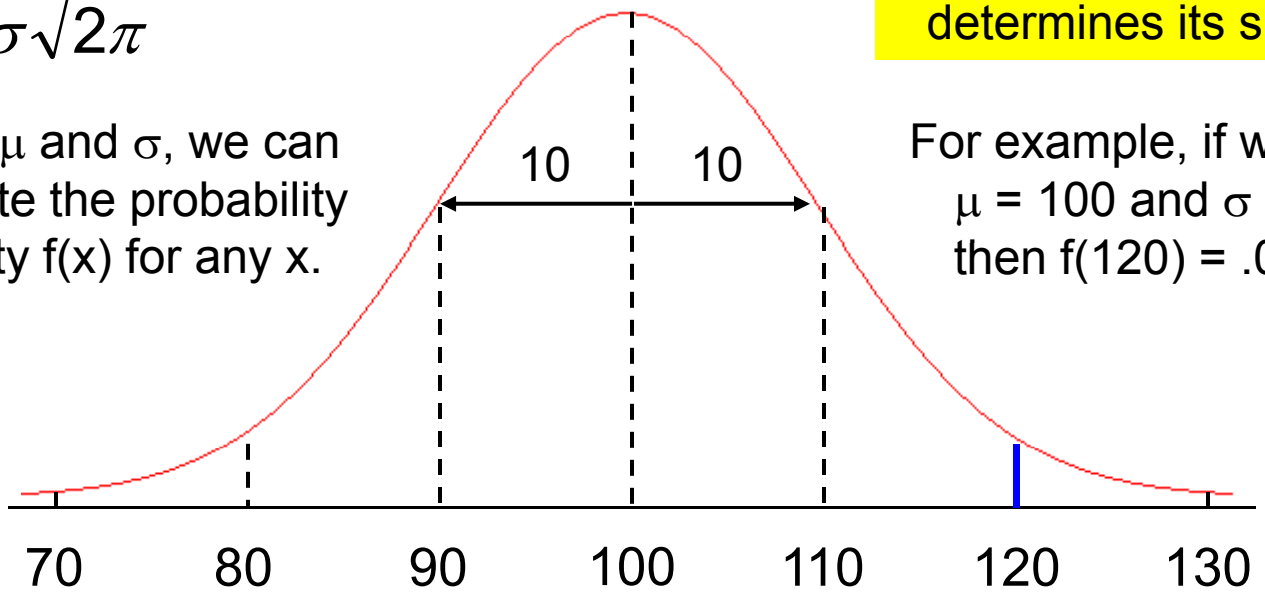
Review of the Gaussian distribution

- Also called "normal distribution" or "bell curve".
- A good approximation for many real-world distributions.
  - Central Limit Theorem: The sum (or mean) of sufficiently many i.i.d. samples from any distribution is approximately Gaussian.
- A symmetric, unimodal distribution N(μ, σ), determined by its mean μ and standard deviation σ:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ determines the center of the distribution, and σ determines its spread.

Given μ and σ, we can compute the probability density f(x) for any x.



For example, if we have μ = 100 and σ = 10, then f(120) = .0054.

Learning Gaussian distributions

Gaussians are simple to learn from training data:

μ = sample mean of x_i $\qquad \mu = \frac{\sum x_i}{N}$

σ = sample standard deviation of x_i $\qquad \sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N-1}}$

For example, consider the distribution of body mass index (BMI) for patients with low and high heart attack risks respectively:

Low risk (4500 patients): mean = 20.0, standard deviation = 2.6
High risk (500 patients): mean = 32.0, standard deviation = 3.1

Probability density of BMI = 26 for Low risk group: f(x = 26 | N(20, 2.6)) = .01
Probability density of BMI = 26 for High risk group: f(x = 26 | N(32, 3.1)) = .02

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Summary of Naïve Bayes

Step 1: Learn a **class-conditional model** for each attribute for each class, using the training data.

Discrete-valued attribute: $Pr(A_j = v_{ij} \mid C = C_k)$
Real-valued attribute: μ and σ of A_j for $C = C_k$

	Gender	BMI	Diabetes
Risk = Low	Pr(Male) = 0.44 Pr(Female) = 0.56	μ = 20 σ = 2.6	Pr(Diabetes) = 0.03 Pr(No diabetes) = 0.97
Risk = High	Pr(Male) = 0.53 Pr(Female) = 0.47	μ = 32 σ = 3.1	Pr(Diabetes) = 0.45 Pr(No diabetes) = 0.55

Step 2: To classify a given test record, first compute the likelihood of each of its attributes given each class, and multiply to obtain the total likelihood.

	Gender = Male	BMI = 26	Diabetes = Yes	Total likelihood
Risk = Low	0.44	0.01	0.03	1.32×10^{-4}
Risk = High	0.53	0.02	0.45	4.77×10^{-3}

Summary of Naïve Bayes

Step 1: Learn a **class-conditional model** for each attribute for each class, using the training data.

Discrete-valued attribute: $\Pr(A_j = v_{ij} \mid C = C_k)$
Real-valued attribute: μ and σ of A_j for $C = C_k$

	Gender	BMI	Diabetes
Risk = Low	Pr(Male) = 0.44 Pr(Female) = 0.56	μ = 20 σ = 2.6	Pr(Diabetes) = 0.03 Pr(No diabetes) = 0.97
Risk = High	Pr(Male) = 0.53 Pr(Female) = 0.47	μ = 32 σ = 3.1	Pr(Diabetes) = 0.45 Pr(No diabetes) = 0.55

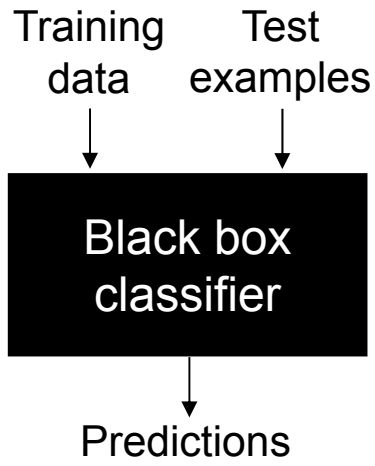
Step 3: Then obtain the prior probability of each class from the training data, and combine this with the likelihood using Bayes' Theorem.

	Total likelihood	Prior	Unnormalized posterior	Posterior
Risk = Low	1.32×10^{-4}	0.9	1.19×10^{-4}	0.2
Risk = High	4.77×10^{-3}	0.1	4.77×10^{-4}	0.8

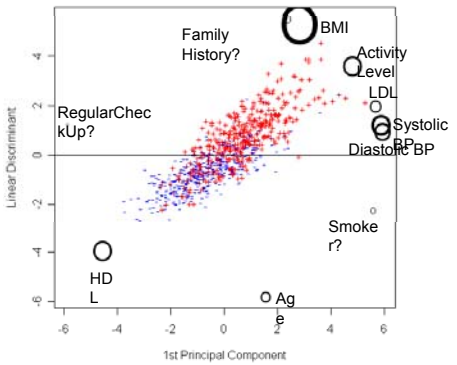
When to use model-based learning

- We have a dataset, with some class we want to predict.
  - We can only do classification, not regression!
  - Datasets with lots of attributes, and/or lots of records, are okay.
  - Datasets with discrete or real values, or both, are okay.
  - We can predict the posterior probability of each class, and use these probabilities to make decisions.
- We want to create an interpretable model of each class, and understand how each attribute affects our predictions.
  - For a discrete attribute, which values are common for each class?
  - For a real attribute, what are μ and σ for each class?
  - For a given test record, compare the class-conditional likelihoods for each attribute.
- Performance is better when the given model makes sense.
  - Naïve Bayes assumes all attributes are conditionally independent given the class. This assumption does surprisingly well in practice!
  - We can also use a more complicated model for each class, such as a Bayesian network (to be discussed in a future lecture).

Advanced topics



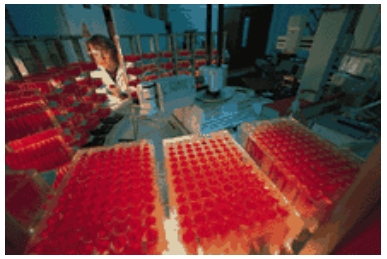
1. **Black box classifiers** such as neural networks and support vector machines often achieve higher prediction accuracy, but do not produce easily interpretable models.



2. We often want to use only a subset of relevant attributes for prediction ("**feature selection**") or to create a small set of new attributes that more accurately represent the data ("**feature extraction**").

3. In **active learning**, we are given a large number of unlabeled data points, and must decide which ones will be the most informative to label.

For example, in drug discovery, we have a large set of chemical compounds, and must choose the most promising to test for clinical relevance.



Drug discovery

Some references

- T.M. Mitchell. *Machine Learning*, McGraw Hill, 1997. (Chapter 6)
- A.W. Moore. Probability for data miners, available at: http://www.autonlab.org/tutorials/prob.html
- If you need a refresher on conditional probability and Bayes' Rule, see my slides on Blackboard.
- Auton Lab Fast Classifiers are available at: http://www.autonlab.org/autonweb/16478.html