Data Mining (CSE5312/CSEG312)

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science & Engineering
Sogang University

Time & Place

- Class Time: **TR 10:30∼11:45**
- Room: K304

Instructor

- Name: Jihoon Yang
- Email: yangjh@sogang.ac.kr
- Office & Hours: AS805; MWF 10:00-12:00
- URL: http://eclass.sogang.ac.kr

- TA: Bokjin Jung (youmeky5@naver.com, AS808)

Grading Policy

- Two in-class exams: Midterm(20%) + Final(30%) = 50%
- Project: 25%
- Problem sets, Lab assignments, Quizzes, Participation in class: 25%

- Will grade on the curve
- Late submissions are not accepted except for extenuating circumstances
- All assignments are individual work unless specified otherwise
  - You may *discuss* material needed for doing the assignments, but should **NEVER share/post/email** solutions with others
  - Academic dishonesty will cause failure in the course

Projects

- Team size: 1∼3
- Topic: Any **active** competitions (e.g. Kaggle, KDDCUP, RecSys)
- 1 page description of the Competition and Task due: March 26
- 1∼2 page intermediate progress report: April 30
- Presentation on June 15 or 17
- Final **full** paper/report (with a CD containing all codes) due by **June 18**, 5 p.m.
- Final report should be in the form of a journal/conference paper
- **Evaluation criteria**: Rank in the leaderboard (Competitions) & clarity of the writeup
  - In top 10% of leaderboard, International paper acceptance/in leaderboard, submission: ★★★★★/★★★
  - Domestic paper acceptance/submission: ★★★/★★
  - Report: ★

References

- Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer.
- Murphy, K. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.
- Duda, R., Hart, P., and Stork, D. (2001). Pattern Classification. Wiley.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning. Springer.
- Mitchell, T. (1997). Machine Learning. McGraw-Hill.

Lecture Schedule (tentative)

1. Introduction
2. Association Rule Mining
3. Decision Tree Learning (including Information Theory, C4.5)
4. Bayesian Learning (including Bayes Decision Theory, Naive Bayes, Bayes nets)
5. Artificial Neural Networks (including Perceptron, MLP, SVM)
6. Clustering
7. Ensemble Learning
8. Deep Learning
9. Reinforcement Learning
10. Final exam: **June 10**

Overview

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science & Engineering
Sogang University

Data Mining

- *We are drowning in information and starving for knowledge* – John Naisbitt
- We are entering the era of **big data**
  - Gartner's definition of 3Vs: Volume, Velocity, Variety
  - Big Data initiative of Obama in 2012, $200M
- Refers to *extracting* or *mining* knowledge from large amounts of data; So large or complex for traditional data processing applications are inadequate
- Technological Driving Factors
  - Larger, cheaper memory
  - Faster, cheaper processors
  - Success of DBs and the Web
  - New ideas in machine learning/statistics

우리는 정보에 빠져 지식에 굶주리고 있다. - John Naisbitt
지금 우리는 빅데이터 시대를 맞이하고 있다.
Gatner의 3 V = 크기, 속도, 다양성
2012년 오바마는 빅데이터에 2억달러 투자

데이터 마이닝이란 많은 양의 데이터에서 지식을 추출하거나 채굴하는 것을 말한다.
기존 데이터 처리 에플리케이션의 경우 너무 크거나 복잡하기에 부적절 하다.

기술적 추진 요인에는 다음이 따른다
더 크로 저렴한 메모리
더 빠르고 저렴한 프로세서
DB와 웹의 성공
기계학습, 통계의 새로운 아이디어

Statistics vs. Data Mining vs. Machine Learning

- Traditional statistics
  - First hypothesize, then collect data, then analyse
  - Often model-oriented (strong parametric models)
- Data mining
  - Few if any a priori hypotheses
  - Data is usually already collected a priori
  - Analysis is typically data-driven not hypothesis-driven
  - Often algorithm-oriented rather than model-oriented
  - Statistical ideas are very useful in data mining (e.g. in validating whether discovered knowledge is useful)
  - Data mining relies heavily on ideas from machine learning
  - More emphasis on scalability (e.g. algorithms that can work on outside main memory data, or streaming data)
  - Somewhat more application-oriented: higher visibility in industry and in public, while ML is somewhat more theoretical, research oriented (i.e. emphasis on automating the discovery of regularities from data, characterizing what can be learned and under what conditions, obtaining guarantees regarding quality of learned models)

통계 vs 데이터 마이닝 vs 기계학습

전통적인 통계
- 먼저 가설을 세우고 데이터를 수집한 뒤 분석
- 종종 모델 지향적(강력한 파라메트릭 모델)

데이터 마이닝
- 선험적 가설이 있는 경우는 거의 없다.
- 데이터는 일반적으로 이미 사전에 수집되있다.
- 분석은 일반적으로 가설 기반이 아닌 데이터 기반이다.
- 종종 모델보다 알고리즘을 지향
- 통계적 아이디어는 데이터 마이닝에 매우 유용합니다 (예 :
발견 된 지식이 유용한 지 여부)
- 데이터 마이닝은 머신 러닝의 아이디어에 크게 의존합니다.
- 확장성에 더 강조 (예 : 외부에서 작동 할 수있는 알고리즘)
주 메모리 데이터 또는 스트리밍 데이터)
- 다소 애플리케이션 지향적 : 업계 및 업계에서 더 높은 가시성
ML은 다소 이론적이며 연구 지향적입니다 (예 :
데이터에서 규칙성을 발견하는 것에 대한 자동화에 중점
어떤 조건에서 무엇을 배울 수 있는지에 대해 특성화.
학습 된 모델의 품질에 대한 보증 획득)

Machine Learning

- Algorithms or computation or information processing provide for study of cognition and life what calculus provided for physics
- We have a theory of intelligent behavior when we have precise information processing models (computer programs) that produce such behaviour
- We will have a theory of **learning** when we have precise information processing models of learning (computer programs that learn from experience)

기계 학습
알고리즘, 계산 또는 정보 처리는 미적분학이 물리학에 제공한 인식과 생활의 연구 등을 제공한다.

우리는 지능적인 행동에 대한 이론을 가지고 있다.

정보 처리 모델 (컴퓨터 프로그램)
그런 행동
정확한 정보가있을 때 학습 이론을 갖게됩니다.
학습 모델 처리 (
경험)

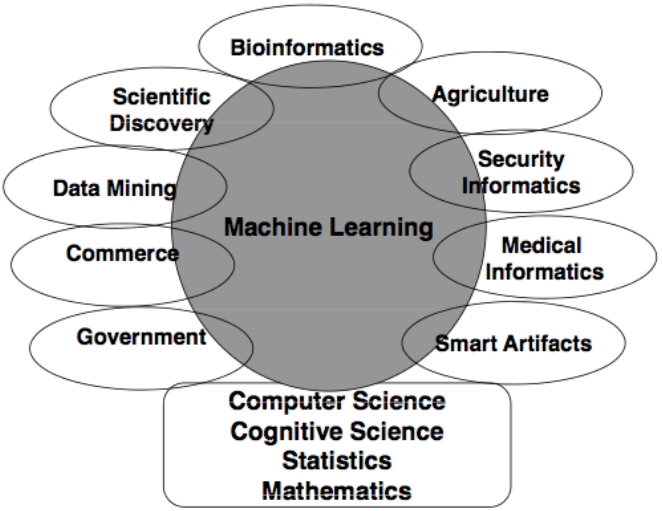
Why should machines learn?

- Intelligent behavior requires knowledge
- Explicitly specifying the knowledge needed for specific tasks is hard, and often infeasible
- Some tasks are best specified by examples (e.g. medical diagnosis, credit risk assessment)
- Buried in large volumes of data are useful predictive relationships (data mining)
- Machine learning is most useful when
  - The structure of the task is not well understood but a representative dataset is available
  - Task (or its parameters) change dynamically
- If we can program computers to learn from experience, we can
  - Dramatically enhance the usability of software (e.g. personalised information assistants)
  - Dramatically reduce the cost of software development (e.g. for medical diagnosis)
  - Automate data driven discovery (e.g. bioinformatics, social informatics)

ML Applications

- Medical diagnosis/image analysis (e.g. pneumonia)
- Spam filtering, fraud detection (e.g. credit cards, phone calls)
- Search and recommendation (e.g. google, amazon)
- Automatic speech recognition & speaker verification
- Locating/tracking/identifying objects in images & videos (e.g. faces)
- Printed and handwritten text parsing
- Driving computer players in games
- Computational molecular biology (e.g. gene expression analysis)
- Autonomous driving
- ...

ML in Context



What is ML?

- A program M is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance as measured by P on tasks in T in an environment Z with experience E

Examples

1. T: cancer diagnosis
   E: a set of diagnosed cases
   P: accuracy of diagnosis on new cases
   Z: noisy measurements, occasionally misdiagnosed training cases
   M: a program that runs on a general purpose computer

What is ML?

2. T: annotating protein sequences with function labels
   E: a data set of annotated protein sequences
   P: score on a test set not seen during training (e.g. accuracy of annotations)

3. T: driving on the interstate
   E: a sequence of sensor measurements and driving actions recorded while observing an expert driver
   P: mean distance traveled before an error as judged by a human expert

Canonical learning problems

- *Supervised learning*: given examples of inputs and corresponding desired outputs, predict outputs on future inputs
  - Classification
  - Regression
  - Time series prediction
- *Unsupervised learning*: given only inputs, automatically discover representations, features, structures, etc.
  - Clustering
  - Outlier detection
  - Compression
- *Reinforcement learning*: given sequences of inputs, actions from a fixed set, and scalar rewards/punishments, learn to select actions in a way that maximises expected reward

Machine Learning

- Learning involves synthesis or adaption of computational structures:
  - Classifiers
  - Functions
  - Logic Programs
  - Rules
  - Grammars
  - Probability distributions
  - Action policies

ML = Inference + Data Structures + Algorithms

Learning input-output functions

- *Target function* f: unknown to the learner – $f \in F$
- Learner's *hypothesis* about what f might be – $h \in H$, *Hypothesis space*
- *Instance space* X: domain of f, h
- *Output space* Y: range of f, h
- *Example*: an ordered pair (x, y) where $x \in X$ and $f(x) = y \in Y$
- F and H *may* or *may not* be the same!
- *Training set* E: a multi-set of examples
- *Learning algorithm* L: a procedure which given some E, outputs an $h \in H$

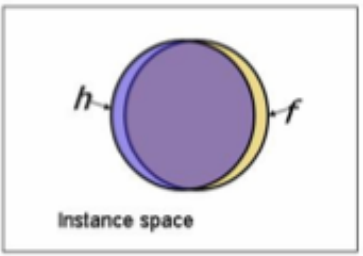
Learning input-output functions

- Training regime
  - Batch
  - Online
  - Distributed
    - Vertical fragmentation
    - Horizontal fragmentation
- Noise
  - Attribute noise
  - Classification noise
  - Both

Inductive learning

- **Premise**: A hypothesis (e.g. a classifier) that is consistent with a sufficiently large number of representative training examples is likely to accurately classify novel instances drawn from the same universe
- We can prove this is an optimal approach (under appropriate assumptions)
- When the number of examples is limited, the learner needs to be smarter (e.g. find a concise hypothesis that is consistent with the data)

Measuring classifier performance



Measuring classifier performance

N: Total number of instances in the data set
$TP(c)$: True Positives for class c, $FP(c)$: False Positives for class c
$TN(c)$: True Negatives for class c, $FN(c)$: False Negatives for class c
TP: True Positives over all classes

$$Accuracy = TP/N$$
$$Precision/Specificity(c) = \frac{TP(c)}{TP(c)+FP(c)}$$
$$Recall/Sensitivity(c) = \frac{TP(c)}{TP(c)+FN(c)}$$
$$False\ Alarm(c) = \frac{FP(c)}{TP(c)+FP(c)} = 1 - Precision(c)$$

Inductive bias

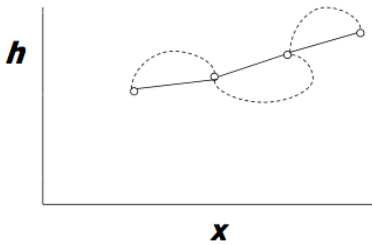
- Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let $D_c = \{\langle x, c(x)\rangle\}$ be an arbitrary set of training examples of c. Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on the data D_c.
- The *inductive bias* of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

- In other words, the set of assumptions, that together with the training data, deductively justify the classifications assigned by the learner to future instances

Function learning and bias

Example



There is an infinite number of functions that match any finite number of training examples!

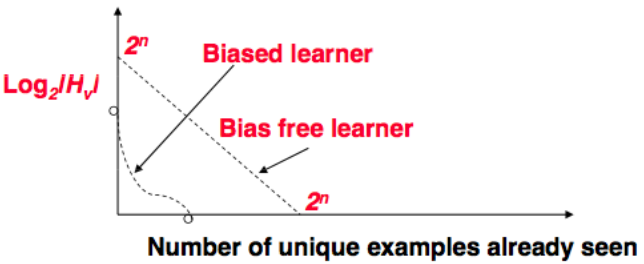
Bias free function learning is impossible!

Learning and bias

Suppose H = set of all n-input Boolean functions.
Suppose the learner is unbiassed. Then

$$|H| = 2^{2^n}$$

H_V = *version space* – the subset of H not yet ruled out by the learner



Learning and bias

- Weaker bias
  $\rightarrow$ more open to experience, flexible
  $\rightarrow$ more expressive hypothesis representation

- *Occam's razor*
  - Simple hypotheses preferred
  - Linear fit preferred to quadratic fit assuming both yield relatively good fit over the training examples

- *Learning in practice requires a trade-off between complexity of hypothesis and goodness of fit*; How this trade-off is done affects the learner's ability to generalise

Course Objectives

- Understand, implement, and use ML algorithms to solve practical problems
- Make intelligent choices among learning algorithms for specific applications
- Formulate and solve new ML problems combining or adapting elements of existing algorithms
- Analyze learning algorithms (e.g. performance guarantees) and distinguish between easy and hard learning problems
- Gain adequate background to understand current literature
- Gain an understanding of the current state of the art in ML
- Learn to conduct original research in ML

Association Rule Mining

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science & Engineering
Sogang University

APYiori

Motivation

- Market basket analysis:
  - "Which groups or sets of items are customers likely to purchase on a given trip to the store?"
  - Will help plan marketing or advertising strategies (e.g. store layouts) & catalog design

Definitions

- *Support* of a rule $X \rightarrow Y$:

$$support(X, Y) = \frac{\#\{\text{customers who bought } X \text{ and } Y\}}{\#\{\text{customers}\}}$$

- *Confidence* of a rule $X \rightarrow Y$:

$$confidence(X \rightarrow Y) = \frac{\#\{\text{customers who bought } X \text{ and } Y\}}{\#\{\text{customers who bought X}\}}$$

- *Itemset*: a set of items; k-itemset: an itemset with k items
- Occurrence frequency of an itemset (aka frequency, *support count*, count of the itemset): number of transactions that contain the itemset
- Minimum support count: the number of transactions required for the itemset to satisfy minimum support

Finding frequent item sets: Apriori Algorithm

(Agrawal and Srikant, 1994)

Notations:

- L_k: the set of *frequent* k-itemsets
- C_k: the set of *candidate* k-itemsets

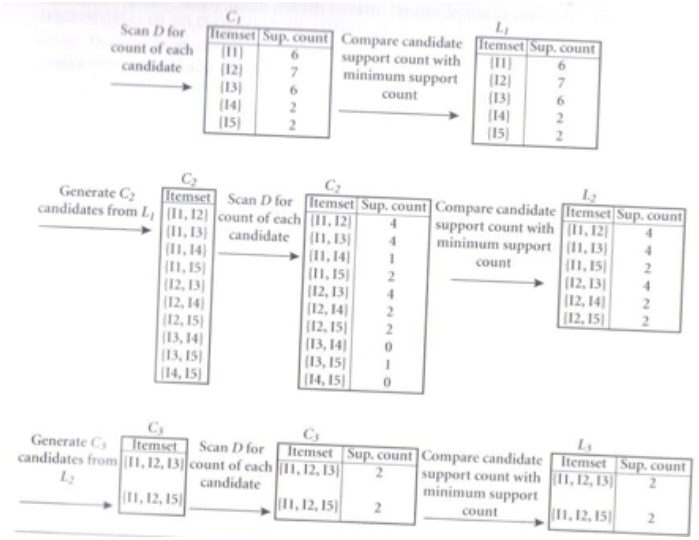
Example

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

~ 최종 out put

Apriori Example

(minimum support count = 2, minimum support threshold = 2/9 = 22%)



Apriori Algorithm

Input: Database, D, of transactions; minimum support threshold, min_sup.

Output: L, frequent itemsets in D.

Method:

```
(1)  L1 = find_frequent_1-itemsets(D);
(2)  for (k = 2; Lk−1 ≠ φ; k++) {
(3)      Ck = apriori_gen(Lk−1, min_sup);
(4)      for each transaction t ∈ D { // scan D for counts
(5)          Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)          for each candidate c ∈ Ct
(7)              c.count++;
(8)      }
(9)      Lk = {c ∈ Ck|c.count ≥ min_sup}
(10) }
(11) return L = ∪k Lk;

procedure apriori_gen(Lk−1: frequent (k − 1)-itemsets; min_sup: minimum support threshold)
(1)  for each itemset l1 ∈ Lk−1
(2)      for each itemset l2 ∈ Lk−1
(3)          if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)              c = l1 ⋈ l2; // join step: generate candidates
(5)              if has_infrequent_subset(c, Lk−1) then
(6)                  delete c; // prune step: remove unfruitful candidate
(7)              else add c to Ck;
(8)          }
(9)  return Ck;

procedure has_infrequent_subset(c: candidate k-itemset; Lk−1: frequent (k − 1)-itemsets);
          // use prior knowledge
(1)  for each (k − 1)-subset s of c
(2)      if s ∉ Lk−1 then
(3)          return TRUE;
(4)  return FALSE;
```

Generating association rules from frequent item sets

- $confidence(A \rightarrow B) = \frac{support\_count(A \cup B)}{support\_count(A)}$

- Steps for rule generation:
  1. For each frequent itemset l, generate all nonempty subsets of l
  2. For every nonempty subset s of l, output the rule

$$s \rightarrow (l - s) \text{ if } \frac{support\_count(l)}{support\_count(s)} \geq min\_conf$$

  where min_conf is the minimum confidence threshold

- A rule is *strong* if it satisfies both minimum support and minimum confidence

Rule generation example

Consider the frequent itemset $l = \{I1, I2, I5\}$ from previous example

$\Rightarrow$

Nonempty subsets of l are

$$\{I1,I2\}, \{I1, I5\}, \{I2,I5\}, \{I1\}, \{I2\}, \{I5\}$$

$\Rightarrow$

The resulting association rules are:

$I1 \wedge I2 \rightarrow I5$, $confidence = 2/4 = 50\%$

$I1 \wedge I5 \rightarrow I2$, $confidence = 2/2 = 100\%$

$I2 \wedge I5 \rightarrow I1$, $confidence = 2/2 = 100\%$

$I1 \rightarrow I2 \wedge I5$, $confidence = 2/6 = 33\%$

$I2 \rightarrow I1 \wedge I5$, $confidence = 2/7 = 29\%$

$I5 \rightarrow I1 \wedge I2$, $confidence = 2/2 = 100\%$

$\Rightarrow$

The 2nd, 3rd, and the last rules are strong and output when $min\_conf = 70\%$

Improvements

- Hashing itemset counts
- Reducing the number of transactions scanned in future iterations
- Data sampling
- Frequent-pattern growth (FP-growth)

SOGANG UNIVERSITY

Information Theory & Decision Trees

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science & Engineering
Sogang University

Email: yangjh@sogang.ac.kr

Decision tree classifiers

- Decision tree representation for modeling dependencies among input variables using elements of information theory
- How to learn decision tree from data
- Over-fitting and how to minimize it
- How to deal with missing values in the data
- Learning decision trees from distributed data
- Learning decision trees at multiple levels of abstraction

SOGANG UNIVERSITY

Decision tree representation

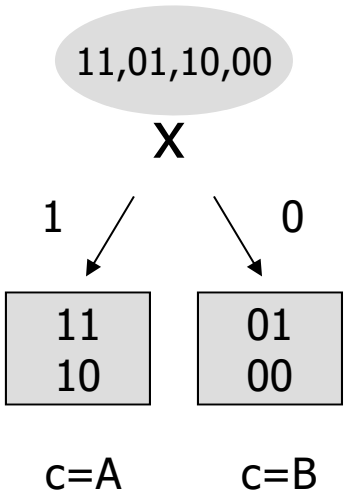
- In the simplest case
  - Each internal node tests on an attribute
  - Each branch corresponds to an attribute value
  - Each leaf node corresponds to a class label

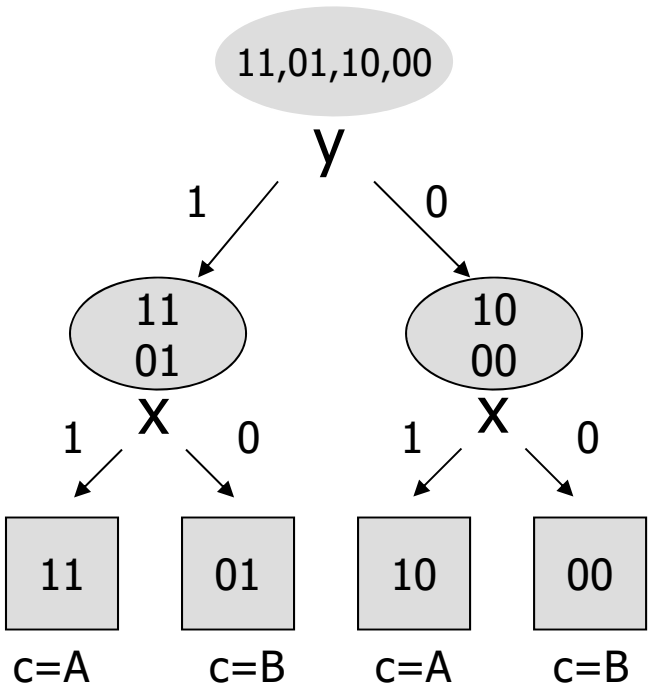
- In general
  - Each internal node corresponds to a test (on input instances) with mutually exclusive and exhaustive outcomes – tests may be univariate or multivariate
  - Each branch corresponds to an outcome of a test
  - Each leaf node corresponds to a class label

Decision tree representation

	Attributes		Class
	x	y	c
1	1	1	A
2	0	1	B
3	1	0	A
4	0	0	B

Examples





Data set ***Tree 1*** ***Tree 2***

- **Should we choose Tree 1 or Tree 2? Why?**

SOGANG UNIVERSITY

Decision tree representation

- **Any boolean function can be represented by a decision tree**
- **Any function** $f : A_1 \times A_2 \times \cdots \times A_n \rightarrow C$
  where each *Ai* is the domain of the *i*th attribute and *C* is a discrete set of values (class labels) can be represented by a decision tree
- **In general, the inputs need not be discrete valued**

Learning decision tree classifiers

- Decision trees are especially well suited for representing simple rules for classifying instances that are described by discrete attribute values

- Decision tree learning algorithms
  - Implement Ockham's razor as a preference bias (simpler decision trees are preferred over more complex trees)
  - Are relatively efficient – linear in the size of the decision tree and the size of the data set
  - Produce comprehensible results
  - Are often among the first to be tried on a new data set

Learning decision tree classifiers

- Ockham's razor recommends that we pick the simplest decision tree that is consistent with the training set
- Simplest tree is one that takes the fewest bits to encode (why? – information theory)
- There are far too many trees that are consistent with a training set
- Searching for the simplest tree that is consistent with the training set is not typically computationally feasible
- Solution
  - Use a greedy algorithm – not guaranteed to find the simplest tree – but works well in practice
  - Or restrict the space of hypothesis to a subset of simple trees

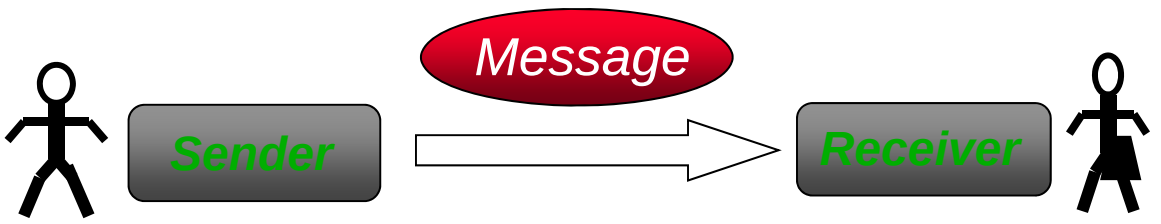
SOGANG UNIVERSITY

Information – some intuitions

- Information *reduces uncertainty*
- Information is *relative* – to what you already know
- Information content of a message is related to *how surprising the message is*
- Information *depends on context*

SOGANG UNIVERSITY

Information and Uncertainty

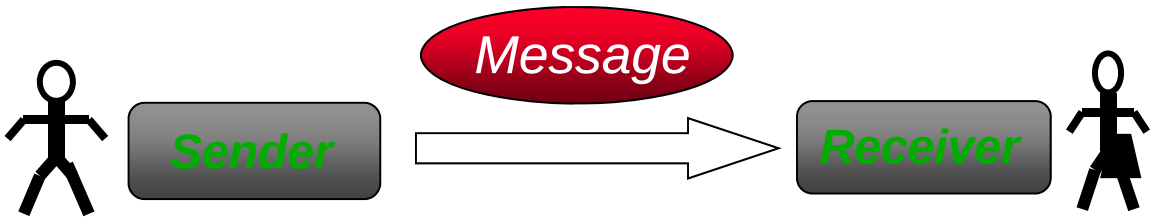


- **You are stuck inside. You send me out to report back to you on what the weather is like. I do not lie, so you trust me. You and I are both generally familiar with the weather in Korea.**

- **On a *July* afternoon in *Korea*, I walk into the room and tell you it is *hot* outside**

- **On a *January* afternoon in *Korea*, I walk into the room and tell you it is *hot* outside**

Information and Uncertainty



- **How much information does a message contain?**
- **If my message to you describes a scenario that you expect with certainty, the information content of the message for you is zero**
- ***The more surprising the message to the receiver, the greater the amount of information conveyed by the message***
- **What does it mean for a message to be surprising?**

Information and Uncertainty

- **Suppose I have a coin with *heads on both sides* and you know that I have a coin with heads on both sides; I toss the coin, and without showing you the outcome, tell you that it came up heads**

→

how much information did I give you?

- **Suppose I have a *fair* coin and you know that I have a fair coin; I toss the coin, and without showing you the outcome, tell you that it came up heads**

→

how much information did I give you?

Information & Coding Theory

- Without loss of generality, assume that messages are binary – made of 0s and 1s
- Conveying the outcome of a fair coin toss requires 1 bit of information – need to identify one out of two equally likely outcomes
- Conveying the outcome of an experiment with 8 equally likely outcomes requires 3 bits..
- Conveying an outcome of that is certain takes 0 bits
- In general, if an outcome has a probability *p*, the information content of the corresponding message is

$$I(p) = -\log_2 p, \; I(0) = 0$$

SOGANG UNIVERSITY

Information is Subjective

- Suppose there are 3 agents – Kim, Lee, Park, in a world where a dice has been tossed; Kim observes the outcome is a "6" and whispers to Lee that the outcome is "even" but Park knows nothing about the outcome
- Probability assigned by Lee to the event "6" is a subjective measure of Lee's belief about the state of the world
- Information gained by Kim by looking at the outcome of the dice = $log_2 6$ bits
- Information conveyed by Kim to Lee = $log_2 6 - log_2 3$ bits
- Information conveyed by Kim to Park = 0 bits

Information and Shannon Entropy

- **Suppose we have a message that conveys the result of a random experiment with *m* possible discrete outcomes, with probabilities**

$$p_1, p_2, \ldots, p_m$$

- **The expected information content of such a message is called the entropy of the probability distribution**

$$H(p_1, p_2, \ldots, p_m) = \sum_{i=1}^{m} p_i I(p_i)$$

$$I(p_i) = -\log_2 p_i \text{ provided } p_i \neq 0$$

$$I(p_i) = 0 \text{ otherwise}$$

Entropy and Coding Theory

- **Entropy is a measure of uncertainty associated with a random variable**

- ***Noiseless coding theorem*: the Shannon entropy is the minimum average message length, in bits, that must be sent to communicate the true value of a random variable to a recipient**

Shannon's entropy as a measure of information

Let $\vec{P} = (p_1 \ .... \ p_n)$ be a discrete probability distribution

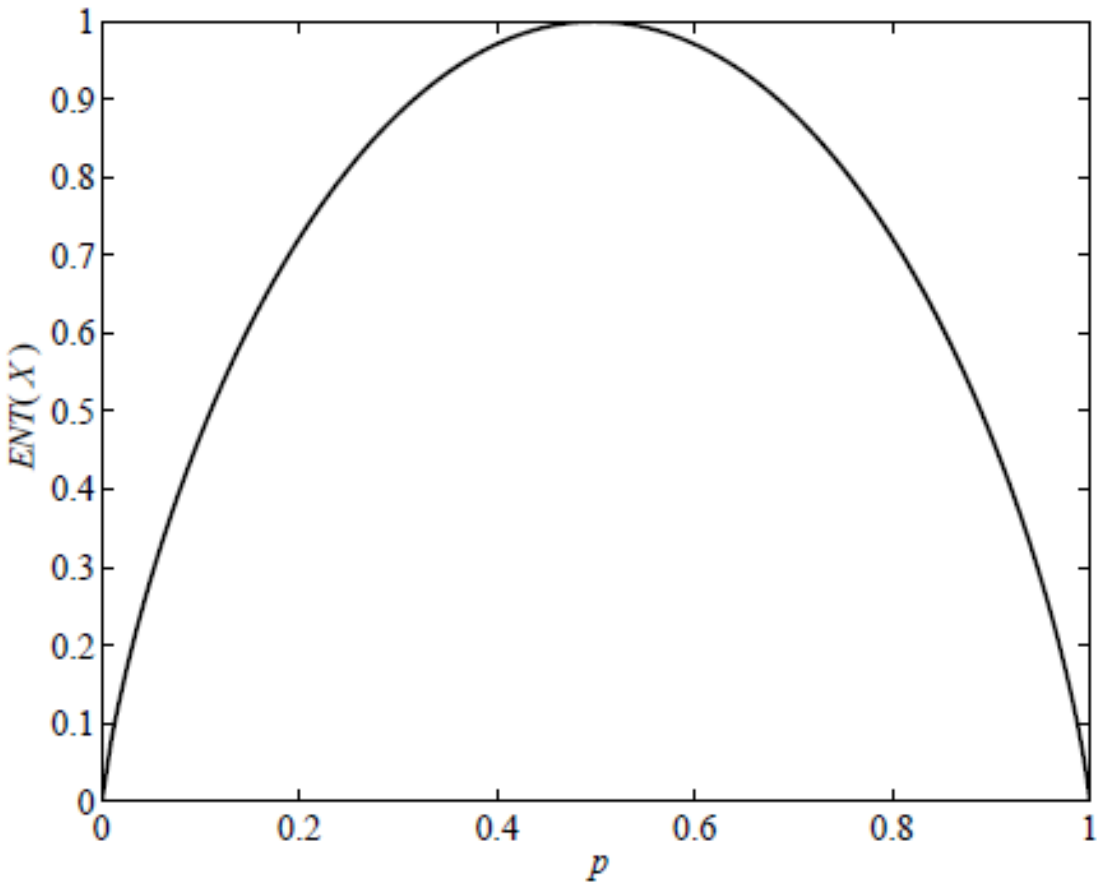
The entropy of the distribution P is given by

$$H(\vec{P}) = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$H\left(\frac{1}{2},\frac{1}{2}\right) = -\sum_{i=1}^{2} p_i \log_2(p_i) = -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1\,bit$$

$$H(0,1) = -\sum_{i=1}^{2} p_i \log_2(p_i) = -1I(1) - 0I(0) = 0\,bit$$

The entropy for a binary variable



Properties of Shannon's entropy

a. $\forall \vec{P} \quad H(\vec{P}) \geq 0$

If there are N possible outcomes, $H(\vec{P}) \leq \log_2 N$

b. If $\forall i \; p_i = \frac{1}{N}, H(\vec{P}) = \log_2 N$

c. If $\exists i$ such that $p_i = 1, H(\vec{P}) = 0$

d. $H(\vec{P})$ is a continuous function of $\vec{P}$

Shannon's entropy as a measure of information

- For any distribution $\vec{P}$, $H\left(\vec{P}\right)$ is the optimal number of binary questions required on average to determine an outcome drawn from ***P***
- We can extend these ideas to talk about how much information is conveyed by the observation of the outcome of one experiment about the possible outcomes of another (*mutual information*)
- We can also quantify to the difference between two probability distribution (*Kullback-Leibler divergence* or *relative entropy*)

Coding theory perspective

- Suppose you and I both know the distribution $\vec{P}$
- I choose an outcome according to $\vec{P}$
- Suppose I want to send you a message about the outcome
- You and I could agree in advance on the questions
- I can simply send you the answers
- Optimal message length on average is $H\left(\vec{P}\right)$
- This generalizes to noisy communication

Entropy and Coding Theory

x	a	b	c	d	e	f	g	h
P(x)	1/2	1/4	1/8	1/16	1/64	1/64	1/64	1/64
code	0	10	110	1110	111100	111101	111110	111111

$$H(x) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - 4\left(\frac{1}{64}\log_2\frac{1}{64}\right)$$
$$= 2 \text{ bits}$$

- **Average code length =** $\frac{1}{2}\times 1 + \frac{1}{4}\times 2 + \frac{1}{8}\times 3 + \frac{1}{16}\times 4 + 4\times\frac{1}{64}\times 6 = 2 \text{ bits}$

Entropy of random variables and sets of random variables

For a random variable X taking values $a_1 \ldots . a_n$,

$$H(X) = -\sum_X P(X)\log_2 P(X)$$

$$= -\sum_{i=1}^{n} P(X = a_i)\log_2 P(X = a_i)$$

If $\mathbf{X}$ is a set of random variables,

$$H(\mathbf{X}) = -\sum_{\mathbf{X}} P(\mathbf{X})\log_2 P(\mathbf{X})$$

Joint entropy and conditional entropy

For a random variable X and Y, the joint entropy

$$H(X,Y) = -\sum_{X,Y} P(X,Y)\log_2 P(X,Y)$$

Conditional entropy of X given Y

$$H(X \mid Y) = \sum_{Y} P(Y) H(X \mid Y = a)$$

$$= -\sum_{Y} P(Y) \sum_{X} P(X \mid Y = a)\log_2 P(X \mid Y = a)$$

$$= -\sum_{X,Y} P(X,Y)\log_2 P(X \mid Y)$$

Joint entropy and conditional entropy

Some useful results:

$$\left.\begin{aligned} H(X,Y) &\le H(X)+H(Y) \\ H(Y \mid X) &\le H(Y) \end{aligned}\right\} \text{(When do we have equality?)}$$

The entropy never increases after conditioning
→observing value of *X* reduces our uncertainty about *Y*, unless independent

Chain rule for entropy

$$\begin{aligned} H(X,Y) &= H(X)+H(Y \mid X) \\ &= H(Y)+H(X \mid Y) \end{aligned}$$

SOGANG UNIVERSITY

Example of entropy calculations

$P(X = H; Y = H) = 0.2; P(X = H; Y = T) = 0.4$

$P(X = T; Y = H) = 0.3; P(X = T; Y = T) = 0.1$

→

$H(X,Y)=-0.2\log_2 0.2+\ldots\approx 1.85$

$P(X = H) = 0.6$, so $H(X) = 0.97$

$P(Y = H) = 0.5$, so $H(Y) = 1.0$

$P(Y = H \mid X = H) = 0.2/0.6 = 0.333$

$P(Y = T \mid X = H) = 1-0.333=0.667$

$P(Y = H \mid X = T)= 0.3/0.4=0.75$

$P(Y = T \mid X = T)=0.1/0.4=0.25$

→

$H(Y|X)\approx 0.88$

Mutual information

For random variables X and Y,

the average mutual information between X and Y

$$I(X,Y) = H(X) + H(Y) - H(X,Y)$$

Or by using chain rule $H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$,

$$I(X,Y) = H(X) - H(X \mid Y)$$

$$I(X,Y) = H(Y) - H(Y \mid X)$$

In terms of probability distributions,

$$I(X,Y) = \sum_{X,Y} P(X=a, Y=b) \log_2 \frac{P(X=a, Y=b)}{P(X=a)P(Y=b)}$$

- **Mutual information measures the extent to which observing one variable will reduce the uncertainty in another**
- **Mutual information is nonnegative, and equal to zero iff *X* and *Y* are independent**

Relative entropy

Let P and Q be two distributions over random variable X.

The relative entropy (Kullback - Leibler distance)

is a measure of "distance" from P to Q.

$$D(P \| Q) = \sum_X P(X) \log_2 \left(\frac{P(X)}{Q(X)} \right)$$

Note $D(P \| Q) \neq D(Q \| P)$

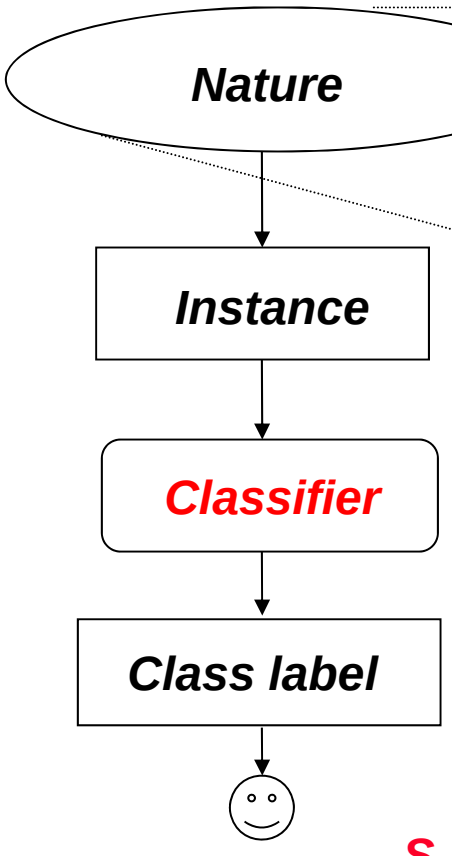
$D(P \| Q) \geq 0$

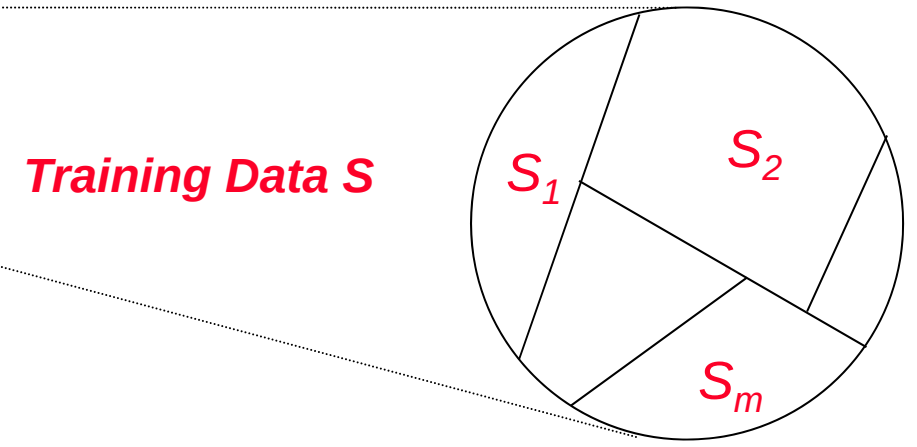
$D(P \| P) = 0$

(KL-divergence is not a true distance measure in that it is not symmetric)

Learning decision tree classifiers

- On average, the information needed to convey the class membership of a random instance drawn from nature is $H(\vec{P})$





$$H\left(\vec{P}\right) = -\sum_{i=1}^{m}\left(\hat{p}_i\right)\log_2\left(\hat{p}_i\right) = \hat{H}(X)$$

where $\hat{P}$ is an estimate of P

and X is a random variable with distribution P

S_i is the multi-set of examples belonging to class C_i

SOGANG UNIVERSITY

Learning decision tree classifiers

- The task of the learner then is to extract the needed information from the training set and store it in the form of a decision tree for classification

- *Information gain* based decision tree learner

  Start with the entire training set at the root

    Recursively add nodes to the tree
      corresponding to tests that yield the
      greatest expected reduction in entropy
      (or the largest expected information gain)
    until some termination criterion is met
      (e.g. the training data at every leaf node has zero entropy)

Learning decision tree classifiers

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

Base cases:

-uniform example classification

-empty examples: majority classification at the node's parent

-empty attributes: use a majority vote?

Which attribute to choose?

The expected amount of **information** provided by an attribute.

The **entropy** or uncertainty about the class

$$H(\omega) = H(\langle P_1, \ldots, P_c \rangle) = \sum_{i=1}^{c} - P_i \log_2 P_i$$

Suppose we have p positive and n negative examples in the training set E at the root

$$H(\omega) = H(E) = H(\langle p/(p+n), n/(p+n) \rangle)$$

A chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values. Let E_i have p_i positive and n_i negative examples. The conditional entropy

$$H(\omega | A = a_i) = H(E_i) = H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i) \rangle)$$

Which attribute to choose?

The conditional entropy, the remaining information needed or the average uncertainty about the class after observing the value of A

$$H(\omega|A) = Remainder(A) = \sum_{i=1}^{v} \frac{|E_i|}{|E|} H(E_i) = \Sigma_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

Information Gain (mutual information) or reduction in entropy from the attribute test:

$$Gain(A) = I(\omega, A) = H(E) - Remainder(A)$$

Choose the attribute with the largest Information Gain

$\Longrightarrow$ choose the attribute that minimizes the remaining information needed

Learning decision tree classifiers – Example

Instances –

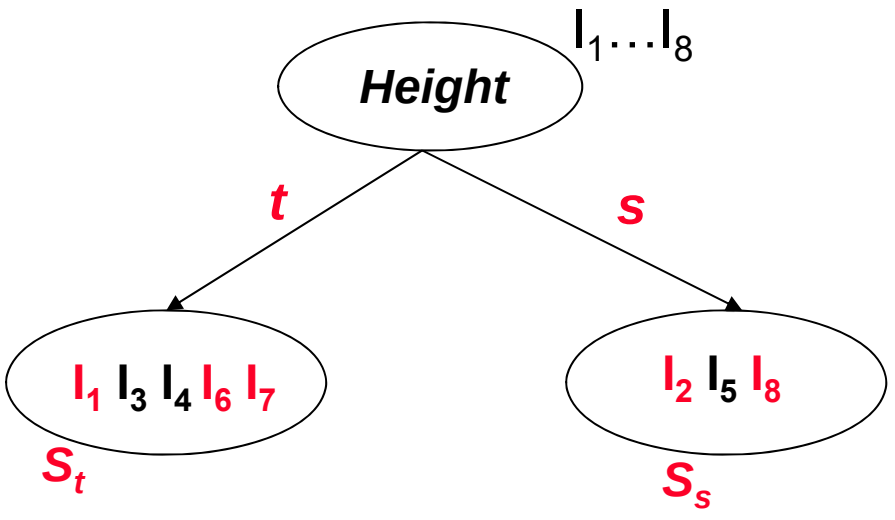
ordered 3-tuples of attribute values corresponding to

Height (tall, short)
Hair (dark, blonde, red)
Eye (blue, brown)

Training Data

Instance	Class label
I_1 (t, d, l)	+
I_2 (s, d, l)	+
I_3 (t, b, l)	–
I_4 (t, r, l)	–
I_5 (s, b, l)	–
I_6 (t, b, w)	+
I_7 (t, d, w)	+
I_8 (s, b, w)	+

Learning decision tree classifiers – Example



$$\hat{H}(X) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = 0.954 bits$$

$$\hat{H}(X \mid Height = t) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 bits$$

$$\hat{H}(X \mid Height = s) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918 bits$$

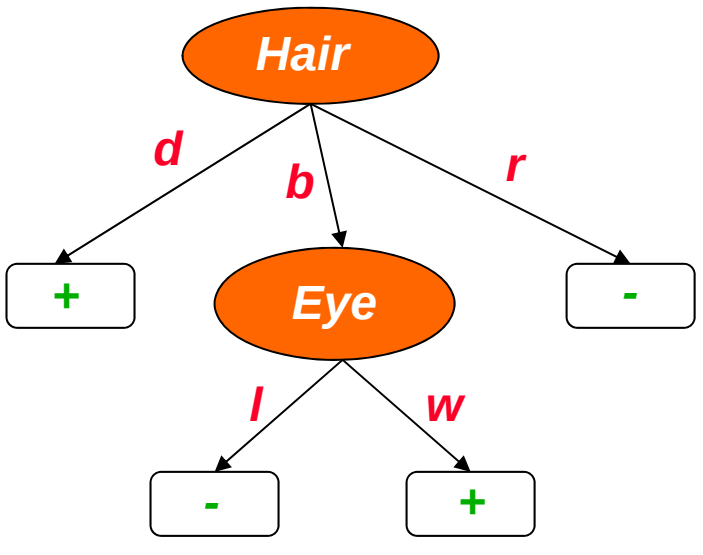
$$\hat{H}(X \mid Height) = \frac{5}{8}\hat{H}(X \mid Height = t) + \frac{3}{8}\hat{H}(X \mid Height = s) = \frac{5}{8}(0.971) + \frac{3}{8}(0.918) = 0.95 bits$$

Similarly, $\hat{H}(X \mid Hair) = \frac{3}{8}\hat{H}(X \mid Hair = d) + \frac{4}{8}\hat{H}(X \mid Hair = b) + \frac{1}{8}\hat{H}(Hair = r) = 0.5 bits$ and

$$\hat{H}(X \mid Eye) = 0.607 bits$$

Hair is the most informative because it yields the largest reduction in entropy; Test on the value of _Hair_ is chosen to correspond to the root of the decision tree

Learning decision tree classifiers – Example



Compare the result with Naïve Bayes

- **In practice, we need some way to prune the tree to avoid overfitting the training data**

Learning, generalization, overfitting

- **Consider the error of a hypothesis *h* over**
  - **Training data:** $Error_{Train}(h)$
  - **Entire distribution *D* of data:** $Error_{D}(h)$

- **Hypothesis** $h \in H$ **over fits training data if there is an alternative hypothesis** $h' \in H$ **such that**

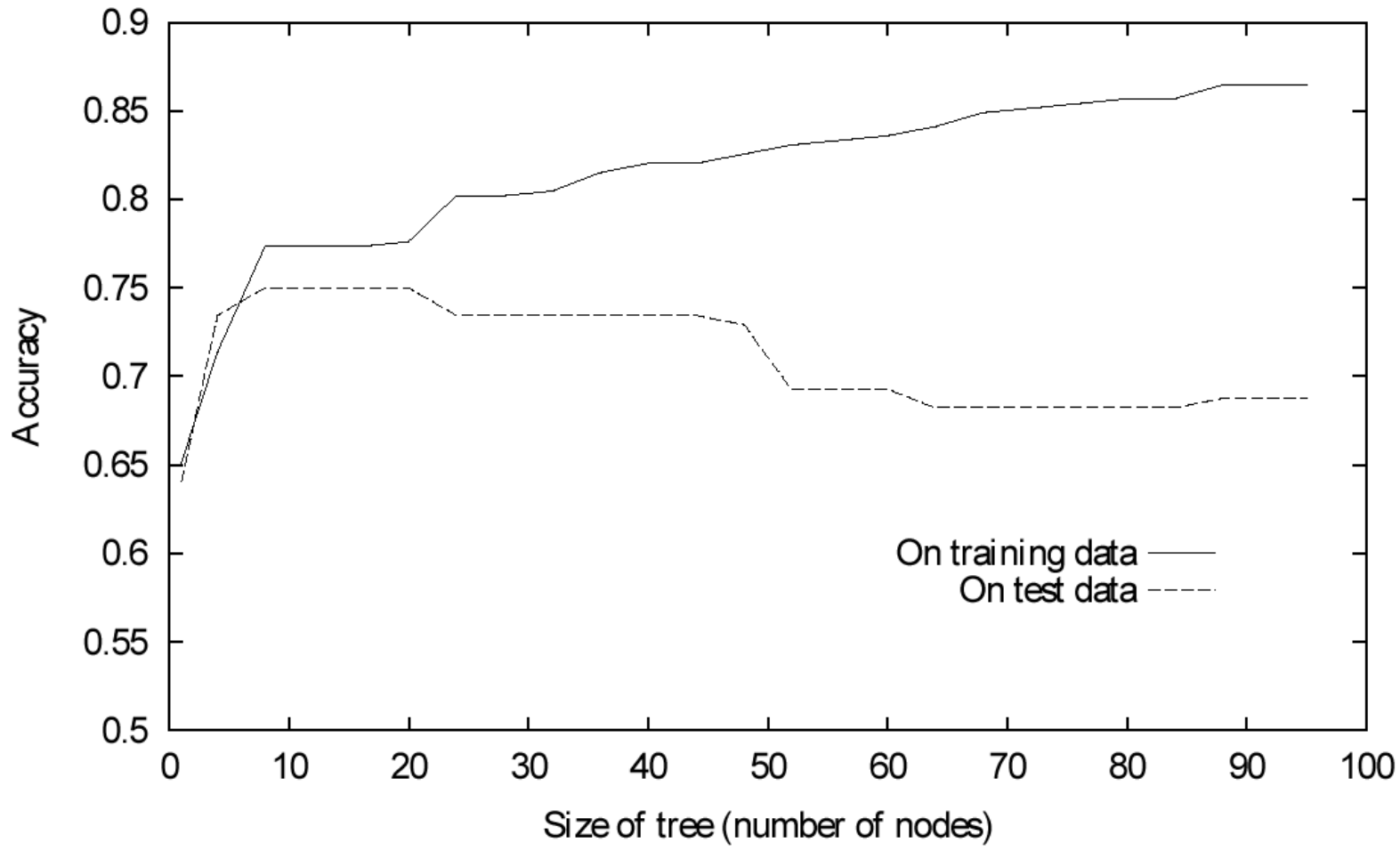
$$Error_{Train}(h) < Error_{Train}(h')$$

and

$$Error_{D}(h) > Error_{D}(h')$$

Overfitting in decision tree learning

(e.g. diabetes dataset)



Causes of overfitting

- As we move further away from the root, the data set used to choose the best test becomes smaller → poor estimates of entropy
- Noisy examples can further exacerbate overfitting

SOGANG UNIVERSITY

Minimizing overfitting

- Use roughly the same size sample at every node to estimate entropy – when there is a large data set from which we can sample
- Stop when further split fails to yield statistically significant information gain (estimated from validation set)
- Grow full tree, then prune
- Minimize *size (tree) + size (exceptions (tree))*

SOGANG UNIVERSITY

Reduced error pruning

- Each decision node in the tree is considered as a candidate for pruning

- Pruning a decision node consists of
  - Removing the sub tree rooted at that node,
  - Making it a leaf node, and
  - Assigning it the most common label at that node or storing the class distribution at that node (for probabilistic classification)

SOGANG UNIVERSITY

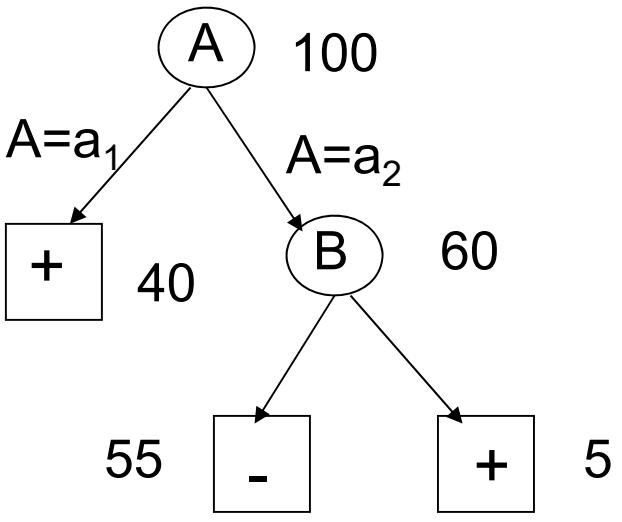
Reduced error pruning

- Do until further pruning is harmful:
  1. Evaluate impact on *validation* set of pruning each candidate node
  2. Greedily select a node which *most improves* the performance on the *validation* set when the sub tree rooted at that node is pruned

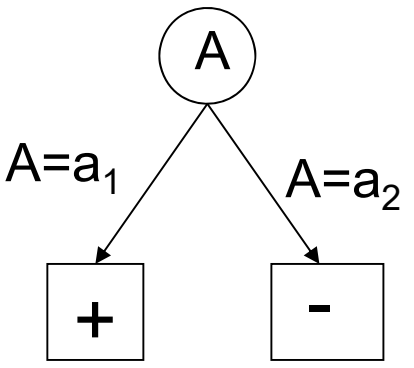
- Drawback
  - holding back the validation set limits the amount of training data available
  - Not desirable when data set is small

Reduced error pruning – Example

Node	Accuracy gain by Pruning
A	-20%
B	+10%



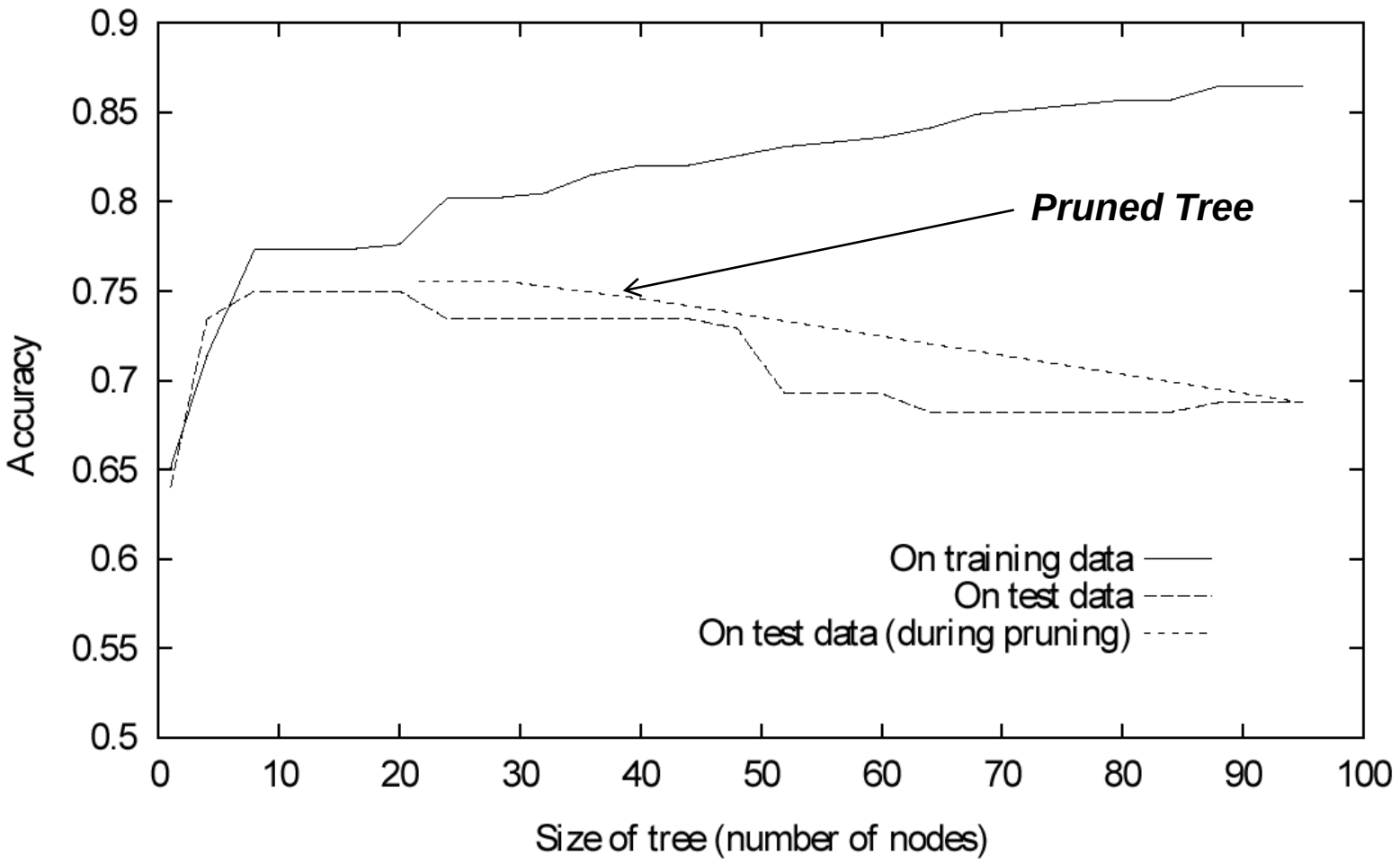
***Before* Pruning**



***After* Pruning**

SOGANG UNIVERSITY

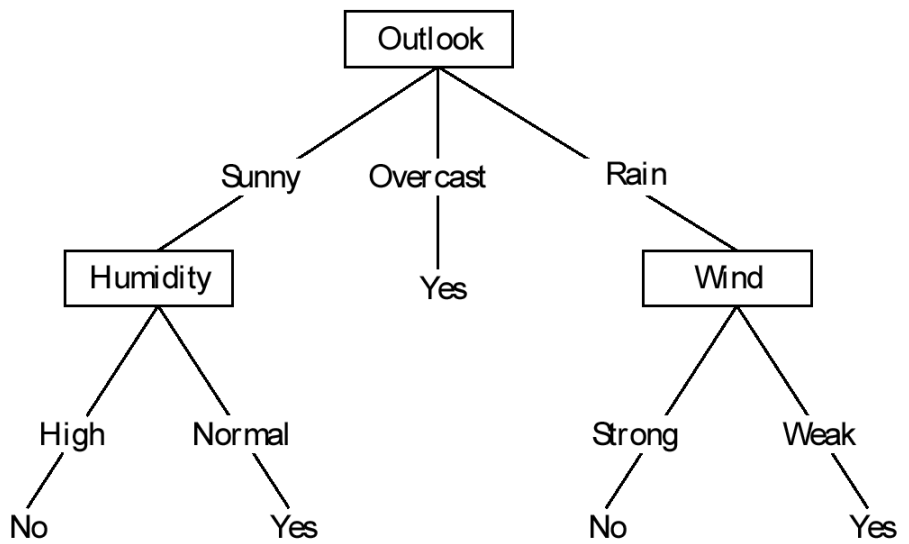
Reduced error pruning



SOGANG UNIVERSITY

Rule post-pruning

- **Convert tree to equivalent set of rules**



IF (*Outlook* = *Sunny*) ∧ (*Humidity* = *High*)
THEN *PlayTennis* = *No*
IF (*Outlook* = *Sunny*) ∧ (*Humidity* = *Normal*)
THEN *PlayTennis* = *Yes*
...

SOGANG UNIVERSITY

Rule post-pruning

1. Convert tree to equivalent set of rules

2. Prune each rule independently of others by dropping a condition at a time if doing so does not reduce estimated accuracy (at the desired confidence level)

3. Sort final rules in order of lowest to highest error for classifying new instances

- Advantage – can potentially correct bad choices made close to the root

- Post pruning based on validation set is the most commonly used method in practice

Classification of instances

- Unique classification – possible when each leaf has zero entropy and there are no missing attribute values

- Most likely or probabilistic classification – based on distribution of classes at a node when there are no missing attributes

Handling different types of attribute values

- **Types of attributes**
  - **Nominal – values are *names***
  - **Ordinal – values are *ordered***
  - **Cardinal (numeric) – values are *numbers* (hence ordered)**
  - **…**

SOGANG UNIVERSITY

Handling numeric attributes

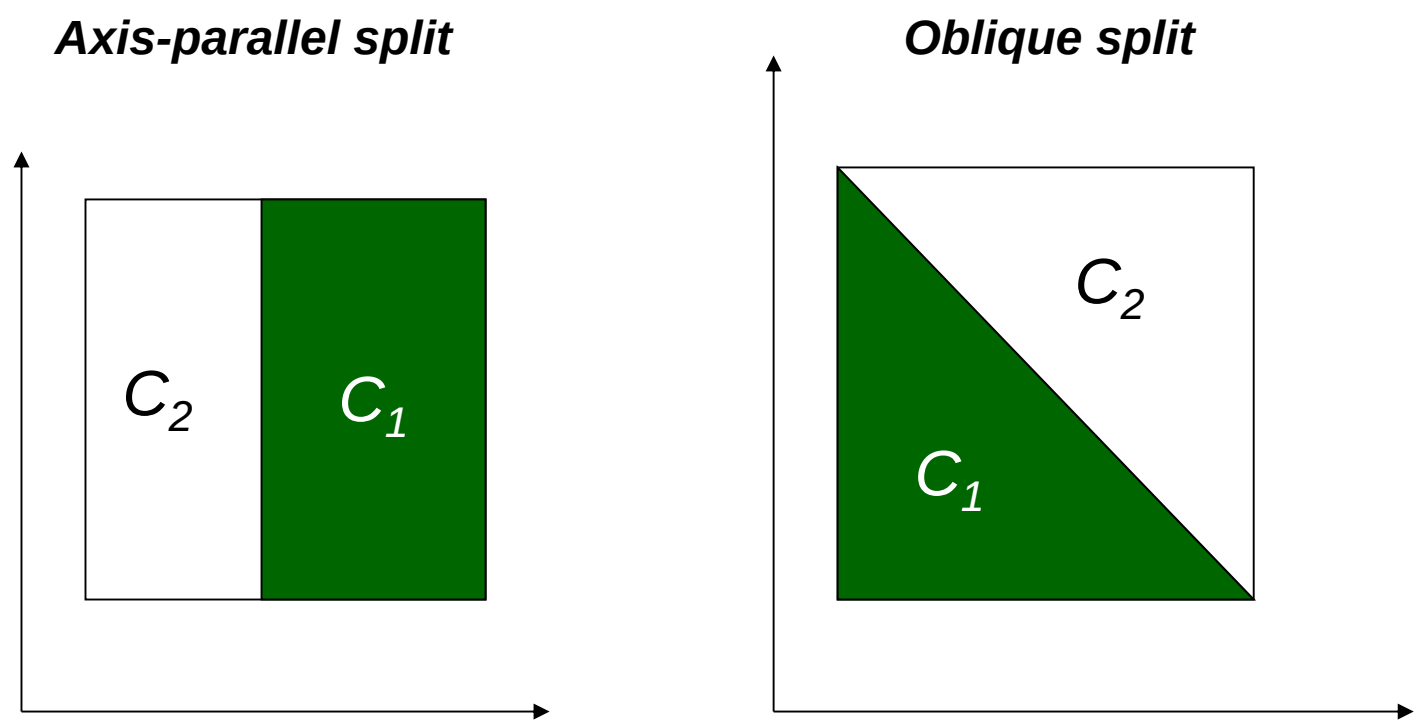
Attribute T	40	48	50	54	60	70
Class	N	N	Y	Y	Y	N

Candidate splits $T > \frac{(48+50)}{2}?$ $T > \frac{(60+70)}{2}?$

$$E(S \mid T > 49?) = \frac{2}{6}(0) + \frac{4}{6}\left(-\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right)\right)$$

- **Sort instances by value of numeric attribute under consideration**
- **For each attribute, find the test which yields the lowest entropy**
- **Greedily choose the best test across all attributes**

Handling numeric attributes



- **Oblique splits cannot be realized by univariate tests**

Two-way versus multi-way splits

- **Entropy criterion favors many-valued attributes**
  - **Pathological behavior – what if in a medical diagnosis data set, social security number is one of the candidate attributes?**

- **Solutions**
  - **Only two-way splits (CART): *A* = *value* versus *A* = ~*value***

  - **Gain ratio (C4.5)**

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{|Values(A)|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Alternative split criteria

- **Consider split of set *S* based on attribute *A***

$$Impurity(S \mid A) = \sum_{j=1}^{|Values(A)|} Impurity(S_j)$$

$$\text{Entropy} \quad Impurity(Z) = \sum_{i=1}^{Classes} -\frac{|Z_i|}{|Z|} \log_2 \frac{|Z_i|}{|Z|}$$

$$\text{Gini} \quad Impurity(Z) = \sum_{i \neq j} \left(\frac{|Z_i|}{|Z|}\right)\left(\frac{|Z_j|}{|Z|}\right) = \frac{1}{2}\left[1 - \sum_{i=1}^{Classes} \left(\frac{|Z_i|}{|Z|}\right)^2\right]$$

(expected rate of error if class label is picked randomly according to distribution of instances in a set)

SOGANG UNIVERSITY

Incorporating attribute costs

- Not all attribute measurements are equally costly or risky
- Example: In medical diagnosis
  - Blood-test has cost $150
  - Exploratory-surgery may have a cost of $3000
- Goal: Learn a decision tree classifier which minimizes cost of classification
  - Tan and Schlimmer (1990)

$$\frac{Gain^2(S,A)}{Cost(A)}$$

  - Nunez (1988)

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

where w $\in$ [0, 1] determines importance of cost

Incorporating different misclassification costs for different classes

- **Not all misclassifications are equally costly:**

  An occasional false alarm about a nuclear power plant meltdown is less costly than the failure to alert when there is a chance of a meltdown

- **Weighted Gini Impurity**

$$Impurity(S) = \sum_{ij} \lambda_{ij} \left(\frac{|S_i|}{|S|} \right) \left(\frac{|S_j|}{|S|} \right)$$

λ_{ij} is the cost of wrongly assigning an instance belonging to class *i* to class *j*

Dealing with missing attribute values (Solution 1)

- Sometimes, the fact that an attribute value is missing might itself be informative
  - Missing blood sugar level might imply that the physician had reason not to measure it
- Introduce a new value (one per attribute) to denote a missing value
- Decision tree construction and use of tree for classification proceed as before

Dealing with missing attribute values (Solution 2)

- **During decision tree construction**
  - Replace a missing attribute value in a training example with the most frequent value found among the instances at the node

- **During use of tree for classification**
  - Replace a missing attribute value in an instance to be classified with the most frequent value found among the training instances at the node

Dealing with missing attribute values (Solution 3)

- **During decision tree construction**
  - Replace a missing attribute value in a training example with the most frequent value found among the instances at the node that have the same class label as the training example

- **During use of tree for classification**
  - Same as solution 2

Dealing with missing attribute values (Solution 4)

- **During decision tree construction**
  - Generate several fractionally weighted training examples based on the distribution of values for the corresponding attribute at the node

- **During use of tree for classification**
  - Generate multiple *instances* by assigning candidate values for the missing attribute based on the distribution of instances at the node
  - Sort each such instance through the tree to generate candidate labels and assign the most probable class label or probabilistically assign class label

- **Used in C4.5**

See5/C5.0 [1997]

- Boosting
- New data types (e.g. dates), N/A values, variable misclassification costs, attribute pre-filtering
- Unordered rulesets: all applicable rules are found and voted
- Improved scalability: multi-threading, multi-core/CPUs

SOGANG UNIVERSITY

Summary of decision trees

- **Simple**
- **Fast (linear in size of the tree, linear in the size of the training set, linear in the number of attributes)**
- **Produce easy to interpret rules**
- **Good for generating simple predictive rules from data with lots of attributes**
- **Popular extensions: GBDT(Friedman, 2001), XGBoost(Chen & Guestrin, 2016), LightGBM(Ke *et al*., 2017)**