Learning from Examples

(Part A)

Contents

Introduction

- Learning is essential for unknown environments
  - i.e., when designer cannot anticipate all possible situations
- Learning is useful as a system construction method
  - i.e., expose the agent to reality rather than trying to write it down
  - No way to program to recognize the faces of family members other than using learning algorithms
- Learning modifies the agent's decision mechanisms to improve performance

- Supervised vs. unsupervised learning

Supervised Learning

- Learning task:
  - Given a training set of N example input-output pairs

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

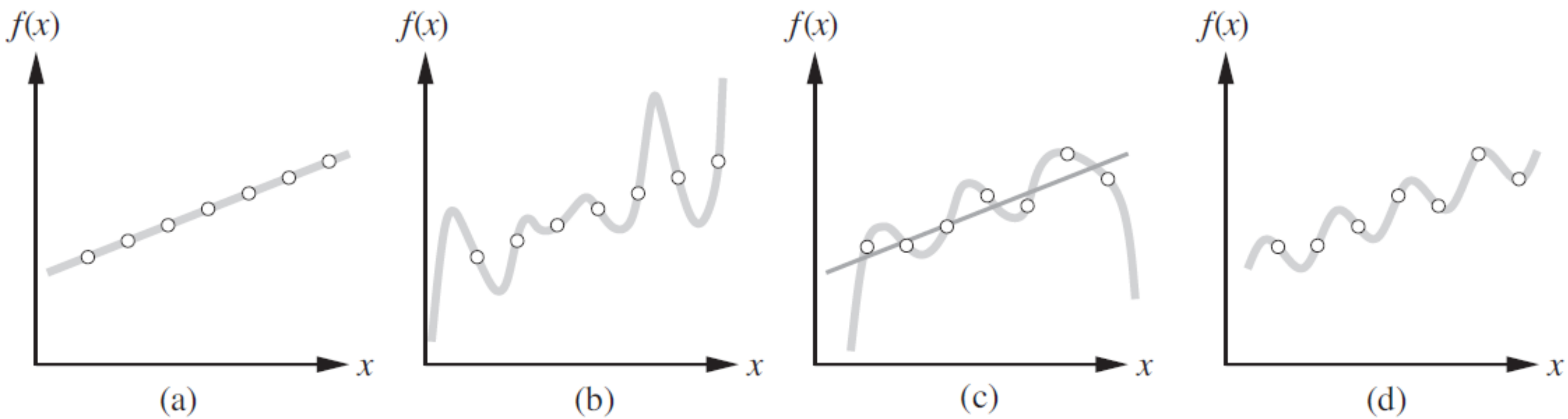
    where each y_j is generated by an unknown function $y = f(x)$, search for a hypothesis h such that $h \approx f$

- Learning problems:
  - We have to learn $\mathbf{P}(Y \mid x)$ when f is stochastic
  - Classification when y is one of a finite set of values
  - Regression when y is a number

- Unsupervised learning receives only inputs without any output

Supervised Learning

Example: curve fitting (regression)



- Construct/adjust h to agree with f on training set
  - h is consistent if it agrees with f on all examples
  - h generalizes well if it correctly predicts y for test set
- Ockham's razor: Prefer the simplest hypothesis consistent with data (maximize a combination of consistency and simplicity)

Supervised Learning

- Supervised learning chooses from the hypothesis space $\mathcal{H}$ the hypothesis h^* that is most probable given the data:

$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data)$$

Bayes' rule:

$$P(a \mid b) = \frac{P(b \mid a)P(a)}{P(b)}$$

- By Bayes' rule this is equivalent to

$$h^* = \arg\max_{h \in \mathcal{H}} P(data \mid h)P(h)$$

  - $P(h)$ is low for complex hypothesis □

Learning Decision Trees

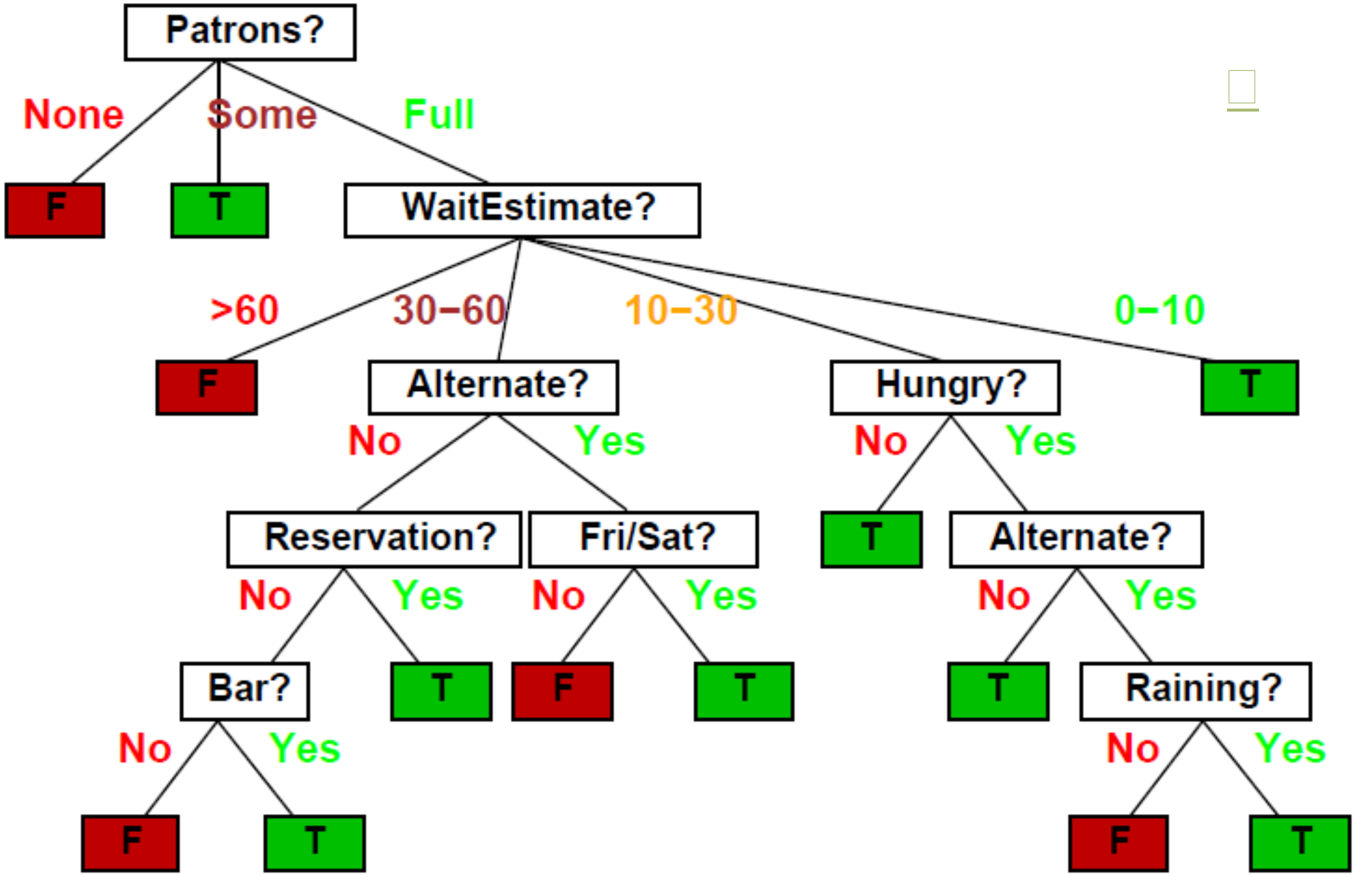
- Examples are described by attribute values (Boolean, discrete, continuous, etc.)

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

- Classification of examples is positive (T) or negative (F)

Decision Tree Representation

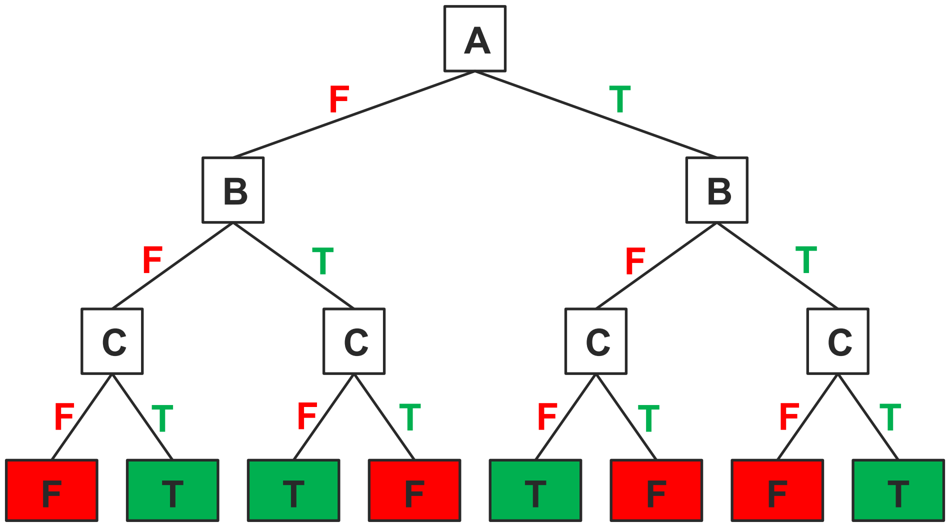
- One possible representation for hypotheses (e.g., "true" tree)



Expressiveness of Decision Trees

- Decision trees can express any function of the input attributes
  - For Boolean functions, truth table row ≡ path to leaf:

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



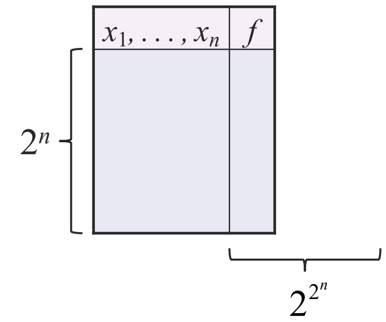
- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f is nondeterministic in x)
  - It probably won't generalize to new examples
- Prefer to find more compact decision trees

Hypothesis Spaces

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

  = number of Boolean functions

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

  - With 6 Boolean attributes, there are more than 1.8×10^{19} trees
- How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??
  - Each attribute can be in (positive), in (negative), or out

    $\rightarrow 3^n$ distinct conjunctive hypotheses
- More expressive hypothesis space
  - Increases chance that target function can be expressed
  - Increases number of hypotheses consistent with training set

    $\rightarrow$ may get worse predictions

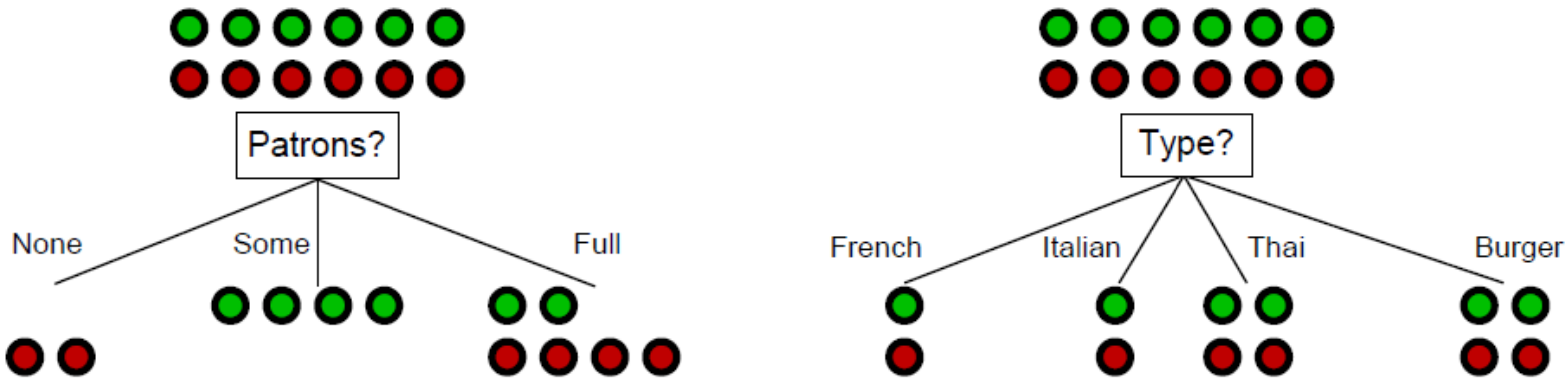
Inducing Decision Trees from Examples

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

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*)
returns a tree
  if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
  else
    $A \leftarrow \operatorname{arg\,max}_{a \in attributes}$ IMPORTANCE(*a*, *examples*)
    tree ← a new decision tree with root test A
    for each value v_k of A **do**
      $exs \leftarrow \{e \mid e \in examples$ **and** $e.A = v_k\}$
      subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* – A, *examples*)
      add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
    return *tree*

Choosing Attribute Tests

- Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”



- *Patrons*? is a better choice—gives more information about the classification

Choosing Attribute Tests

- Basic intuition behind the information theory:
  - Observing an unlikely event is more informative than observing a likely event
  - A guaranteed event should have no information content
  - Independent events have additive information
- The self-information of an event $X = x$ is defined to be $-\log_2 P(X = x)$ bits (more uncertain event has more self-information)
- We can quantify the amount of uncertainty in an entire probability distribution $\langle P(x_1), \ldots, P(x_n) \rangle$ using the entropy:

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \lg P_i$$

  - Entropy is the expected value of self information over the entire distribution

Choosing Attribute Tests

<u>Example</u>: Coin tossing

- The distribution of Head and Tail of an unbiased coin is $\langle \frac{1}{2}, \frac{1}{2} \rangle$

  The self-information of each event is $-\lg \frac{1}{2} = 1$ bit

  So the entropy becomes $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$ bit
- The entropy of the distribution $\langle \frac{3}{4}, \frac{1}{4} \rangle$ of a biased coin:

  $\frac{3}{4} \cdot (-\lg \frac{3}{4}) + \frac{1}{4} \cdot (-\lg \frac{1}{4}) \approx 0.81$ bit
- When an unbiased coin is tossed twice

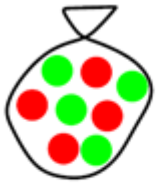
  The distribution of HH, HT, TH, TT is $\langle \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \rangle$

  The self-information of each event is $-\lg \frac{1}{4} = 2$ bit

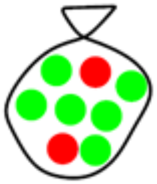
  Therefore, the entropy is $\frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = 2$ bit

Choosing Attribute Tests

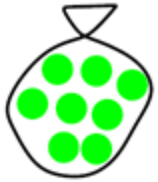
- Entropy can be deemed as a measure of impurity



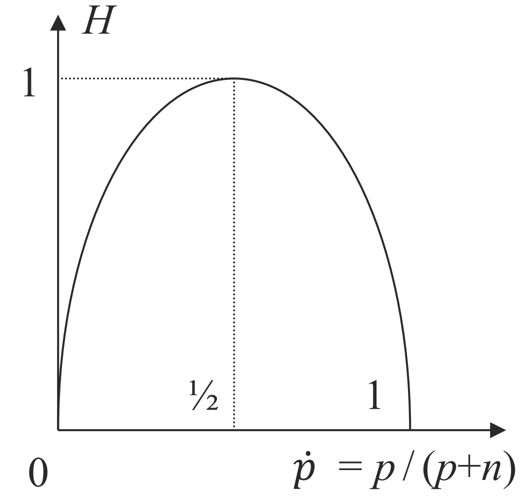
$$H\left(\left\langle \frac{1}{2}, \frac{1}{2} \right\rangle\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$



$$H\left(\left\langle \frac{1}{4}, \frac{3}{4} \right\rangle\right) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} \approx 0.81$$

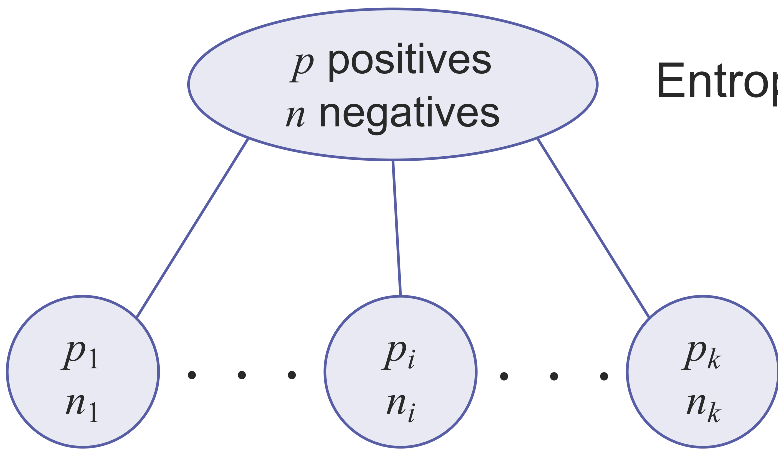


$$H(\langle 0,1 \rangle) = 0$$



Choosing Attribute Tests

- Calculation of information gain for attribute selection:



Entropy of the root: $E = H\left(\left\langle \frac{p}{p+n}, \frac{n}{p+n} \right\rangle\right)$

Entropy of the i-th child:

$$e_i = H\left(\left\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right\rangle\right)$$

Average of child entropies: $e = \sum_{i=1}^{k} \frac{p_i+n_i}{p+n} e_i$

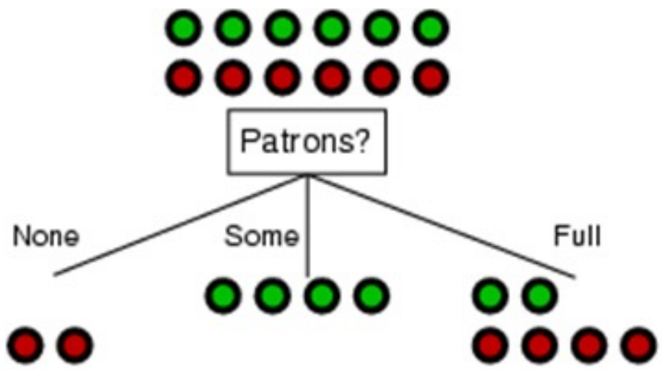
Information gain $= E - e$

- We choose the attribute that maximizes the information gain

Choosing Attribute Tests

Example: Information gain when Patrons? is selected

$$H\left(\left\langle \frac{1}{2}, \frac{1}{2} \right\rangle\right) = 1$$

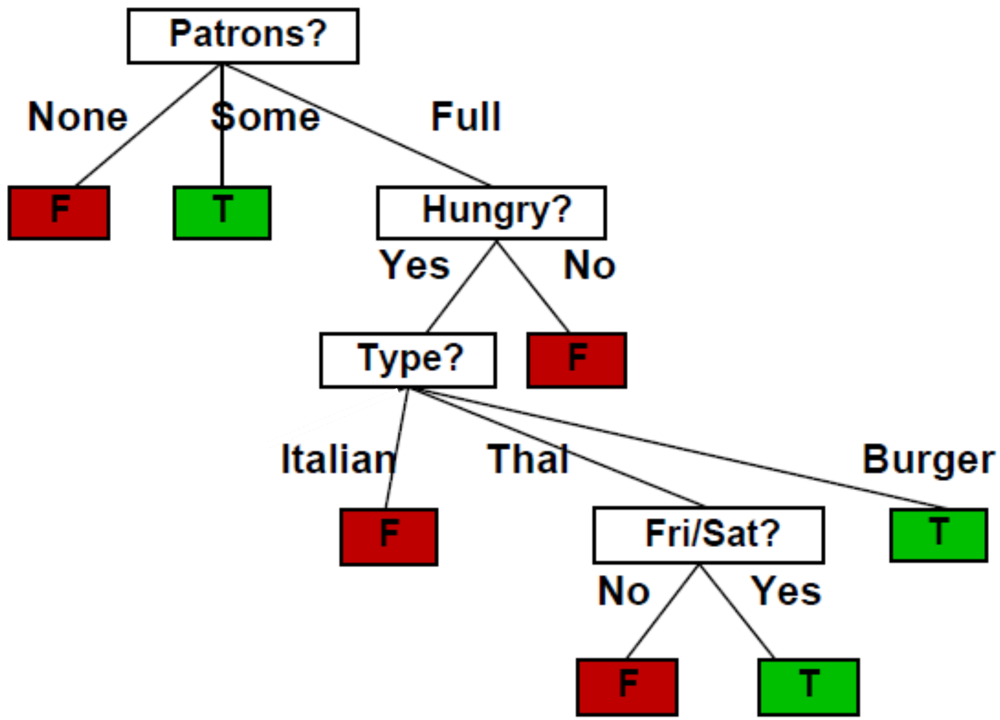


$$\frac{2}{12}\cdot 0 + \frac{4}{12}\cdot 0 + \frac{6}{12}\cdot H\left(\left\langle \frac{1}{3}, \frac{2}{3} \right\rangle\right) = \frac{1}{2}\cdot\left(-\frac{1}{3}\lg\frac{1}{3} - \frac{2}{3}\lg\frac{2}{3}\right) \approx 0.459$$

Information gain $= 1 - 0.459 = 0.541$

Choosing Attribute Tests

- Decision tree learned from the 12 examples:



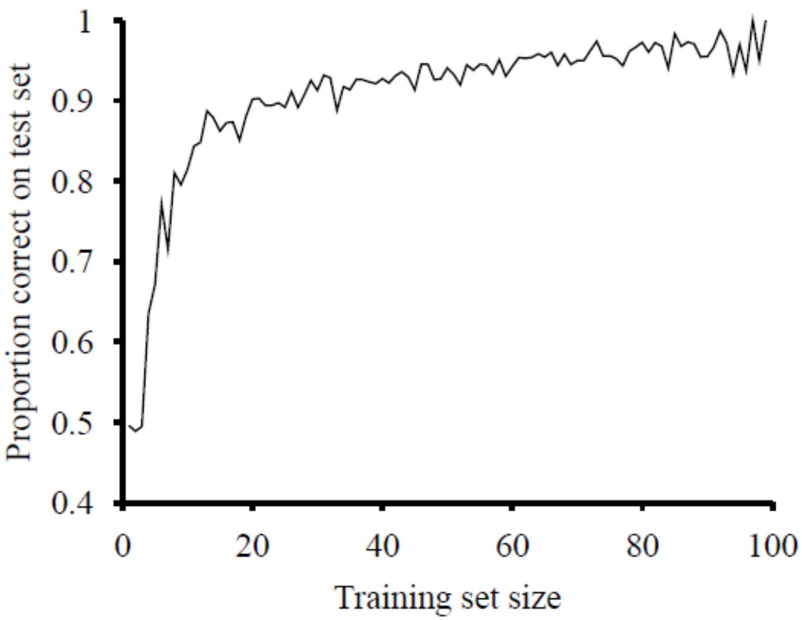
- Substantially simpler than “true” tree—a more complex hypothesis isn't justified by small amount of data

Performance Measurement

- How do we know that $h \approx f$?
  - Try h on a new test set of examples

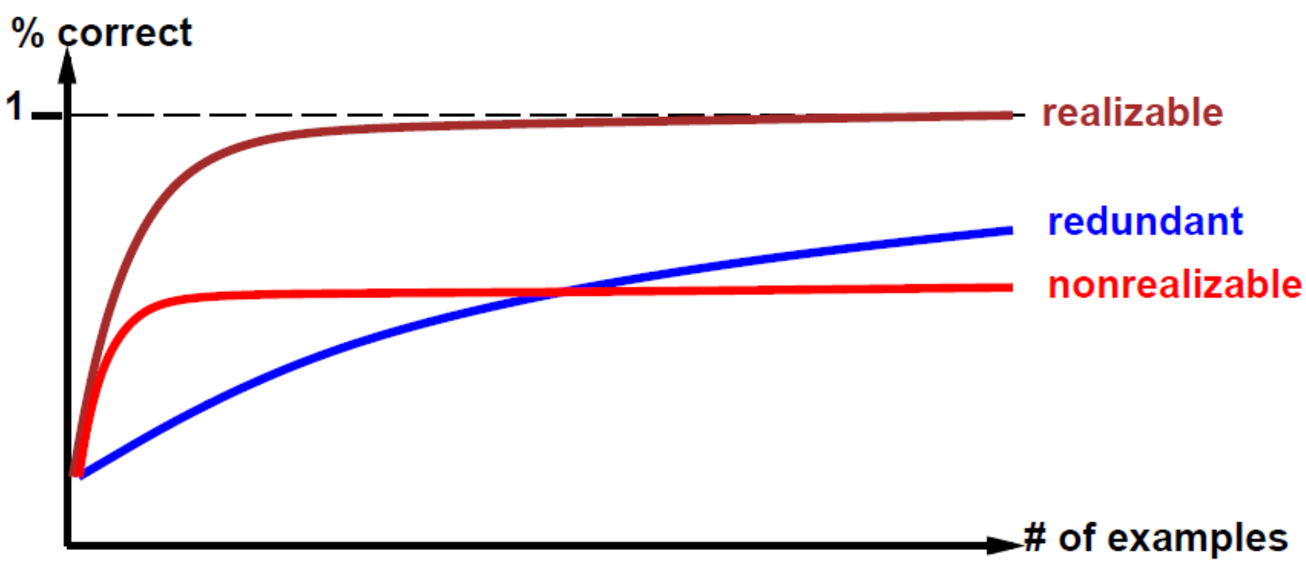
    (use same distribution over example space as training set)
- Learning curve

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



Performance Measurement

- Learning curve depends on
  - Realizable (can express target function) vs. non-realizable
    - Non-realizability can be due to missing attributes or restricted hypothesis class (e.g., linear function)
  - Redundant expressiveness (e.g., loads of irrelevant attributes)



Generalization and Overfitting

- Pruning:
  - Proceeds from the leaves and works back up: ☞
    - Look at a test node with only leaf descendants
    - Eliminate irrelevant test and replace it with a leaf node
    - Repeat until relevant test is seen

- How do we detect irrelevant test?
  - An irrelevant test would split examples into subsets that each have roughly the same proportion of positive examples as the whole set, $p/(p+n)$, and so the information gain will be close to zero

Generalization and Overfitting

- Statistical **significance test**:
  - **Null hypothesis** (assumes true irrelevance)

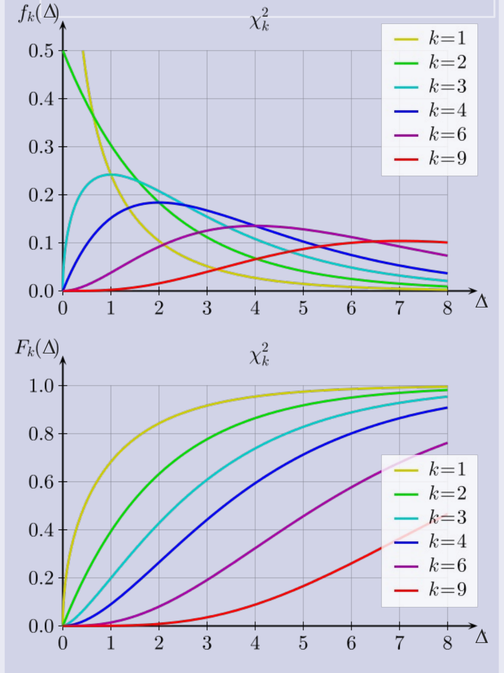
$$\hat{p}_k = \frac{p}{p+n} \times (p_k + n_k) \qquad \hat{n}_k = \frac{n}{p+n} \times (p_k + n_k)$$

  - The total deviation

$$\Delta = \sum_{k=1}^{d} \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}$$

    is distributed according to χ^2 (chi-squared) distribution with $d-1$ degrees of freedom

χ^2–distribution with k degrees of freedom

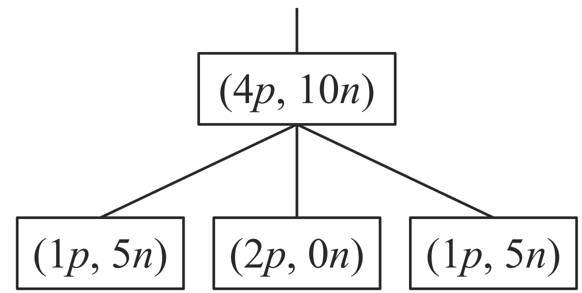


- **χ^2 pruning**: Look up χ^2 table, e.g., with 3 degrees of freedom ($d = 4$)
  - $\Delta \geq 7.82 \rightarrow$ reject the null hypothesis at the 5% level (i.e., test is relevant, so do not prune)
  - $\Delta \geq 11.35 \rightarrow$ reject at the 1% level (harder to be relevant)

Generalization and Overfitting

Example: χ^2 pruning with 2 degrees of freedom

- $\Delta \geq 4.60 \rightarrow$ reject at the 10% level
- $\Delta \geq 5.99 \rightarrow$ reject at the 5% level



$$\hat{p}_1 = \hat{p}_3 = \frac{4}{14} \cdot 6 = \frac{12}{7} \qquad \hat{n}_1 = \hat{n}_3 = \frac{10}{14} \cdot 6 = \frac{30}{7}$$

$$\hat{p}_2 = \frac{4}{14} \cdot 2 = \frac{4}{7} \qquad \hat{n}_2 = \frac{10}{14} \cdot 2 = \frac{10}{7}$$

$$\Delta = 2 \cdot \left(\frac{\left(1 - \frac{12}{7}\right)^2}{\frac{12}{7}} + \frac{\left(5 - \frac{30}{7}\right)^2}{\frac{30}{7}} \right) + \left(\frac{\left(2 - \frac{4}{7}\right)^2}{\frac{4}{7}} + \frac{\left(0 - \frac{10}{7}\right)^2}{\frac{10}{7}} \right) \approx 5.83$$

- We prune at 5% level, but not at 10% level

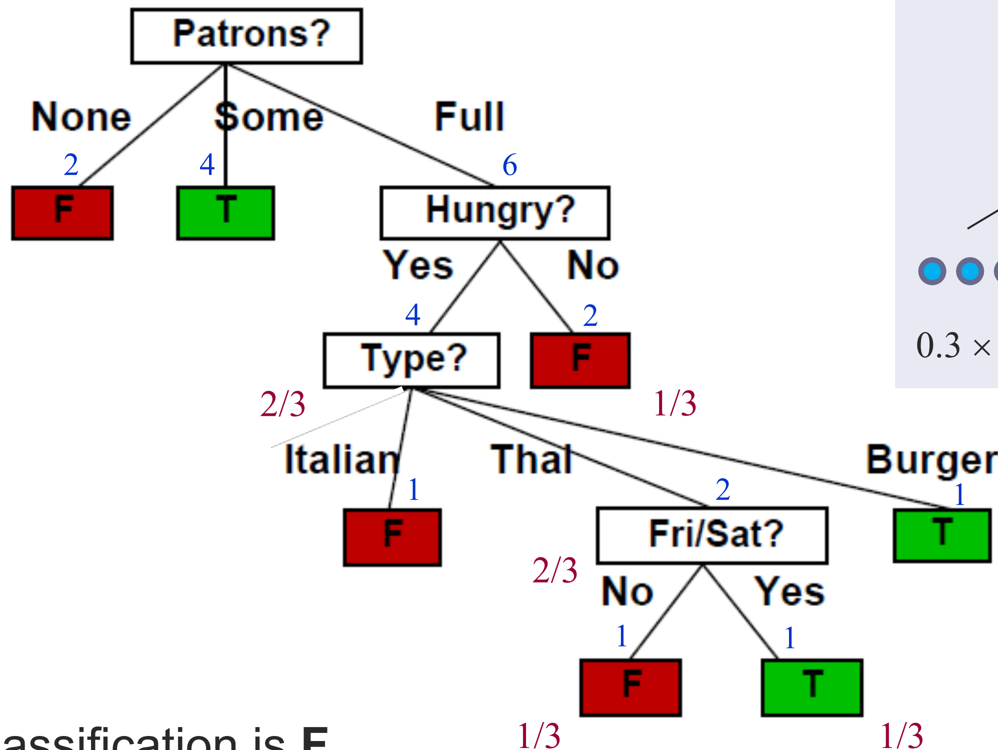
Missing Data

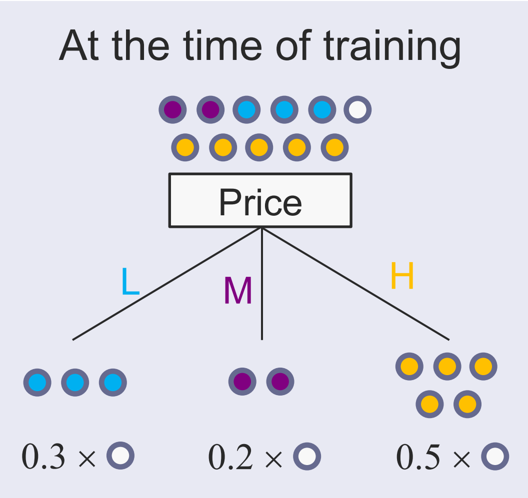
- When the absence of a value is of some significance:
  - Treat as another possible value in its own right
- Otherwise (at the time of test):
  - Split the instance into pieces and send part of it down each branch in proportion to the number of training instances going down that branch
  - The decision at the leaf nodes must be recombined using the weights that have percolated to the leaves
- At the time of training:
  - Split into pieces and send down each branch in the same proportion as the known instances go down the various branches

Missing Data

Example: Classification of an example with some data missing

$$\langle Patron = Full, Type = Thai \rangle$$





The classification is **F**

Multivalued Attributes

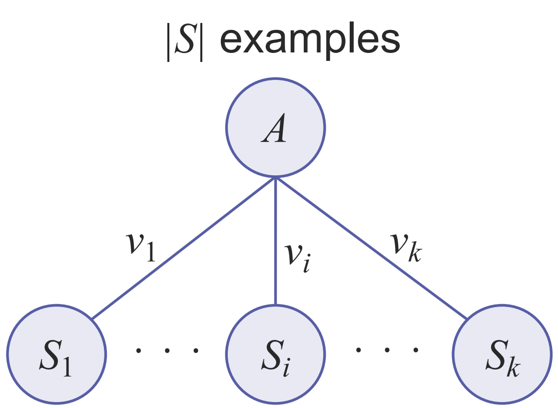
- Information gain measure can be misleading
  - An attribute such as *StudentID* has a different value for every example

    $\rightarrow$ highest information gain
- Use gain ratio instead

  gain ratio = (information gain) / (split information)

When an attribute A splits a set of examples S into n subsets

$$SplitInfo(S, A) = -\sum_{k=1}^{n} \frac{|S_k|}{|S|} \lg \frac{|S_k|}{|S|}$$



- *Splitinfo* gets larger as n gets larger

Numeric Input Attributes

- Restaurant data with a numeric attribute:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	40	F	T	French	0–10	T
X_2	T	F	F	T	Full	9	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	8	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	12	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	45	F	T	French	>60	F
X_6	F	T	F	T	Some	25	T	T	Italian	0–10	T
X_7	F	T	F	F	None	8	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	20	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	10	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	35	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	9	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	10	F	F	Burger	30–60	T

Numeric Input Attributes

- Restricted to binary splits

8	9	10	12	20	25	35	40	45
T F	F F	F T	T	T	T	F	T	F

Calculate information gain for splitting at 30

$$\text{info}([5,4],[1,2]) = \frac{9}{12} H\left(\left\langle \frac{5}{9}, \frac{4}{9} \right\rangle\right) + \frac{3}{12} H\left(\left\langle \frac{1}{3}, \frac{2}{3} \right\rangle\right) = 0.973 \text{ bits}$$

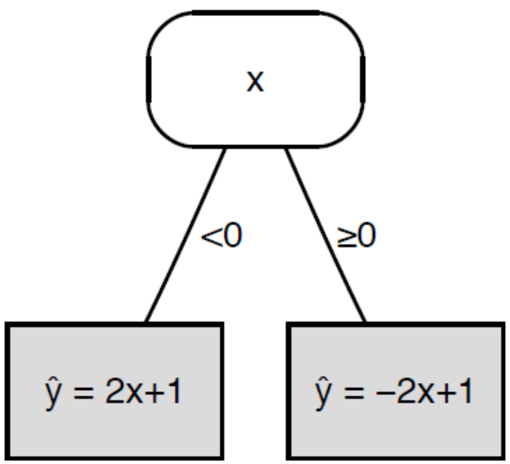
- Test all the possible split points and then split at the point with the lowest entropy

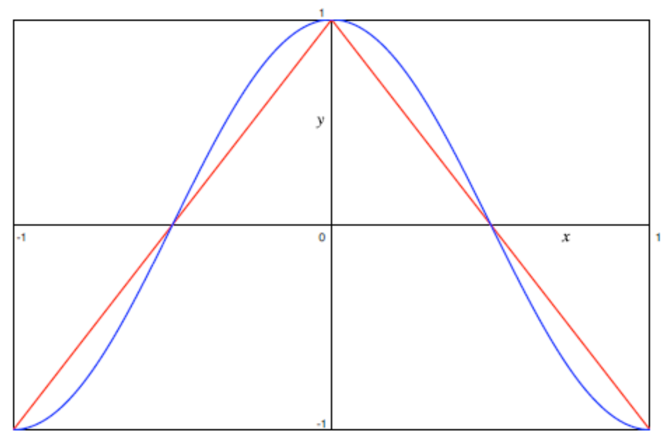
Numeric Input Attributes

- Numeric attributes can be tested many times (successive splits) on a path from the root to the leaf
  - Tests on a single numeric attribute are not located together but can be scattered along the path
  - Trees can be messy and difficult to understand
  - For a more readable tree, allow a multiway test by prediscretizing the numeric attributes

Numeric Output Attributes

- Model tree:
  - Each leaf has a linear function of some subset of numerical attributes
  - The learning algorithm must decide when to stop splitting and begin applying linear regression over the attributes



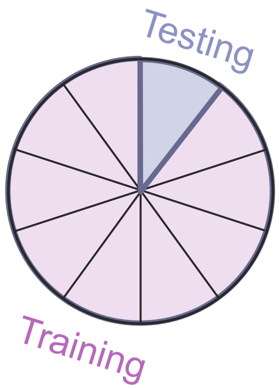


Evaluating and Choosing the Best Hypothesis

- To learn h that fits future data best
  - Assume stationarity (future = present): examples are i.i.d
  - Measure error rate on a set of data not seen yet
- Be careful about peeking
  - Do not use test-set performance to both choose a hypothesis and evaluate it
- Holdout cross-validation (separate training and test sets)
  - Large test set $\rightarrow$ poor hypothesis
  - Large training set $\rightarrow$ poor estimate of error rate

Evaluating and Choosing the Best Hypothesis

- *k*-fold cross-validation:
  - Partition the data into k equal folds (optionally with stratification)
  - Each fold in turn is used for testing, and the remainder for training
  - The k error rates are averaged (better estimate than a single score)



- $k = 5$ or 10 gives an estimate that is statistically likely to be accurate
  - At a cost of 5 to 10 times longer computation time

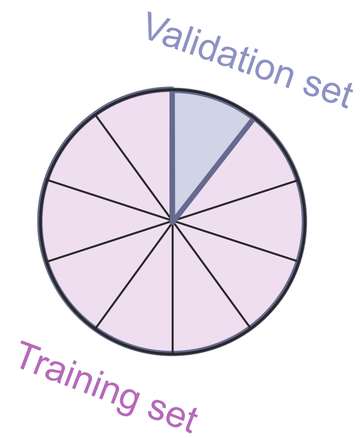
- Leave-one-out cross-validation (LOOCV): $k = \#$ examples

Evaluating and Choosing the Best Hypothesis

- To avoid peeking in doing model selection

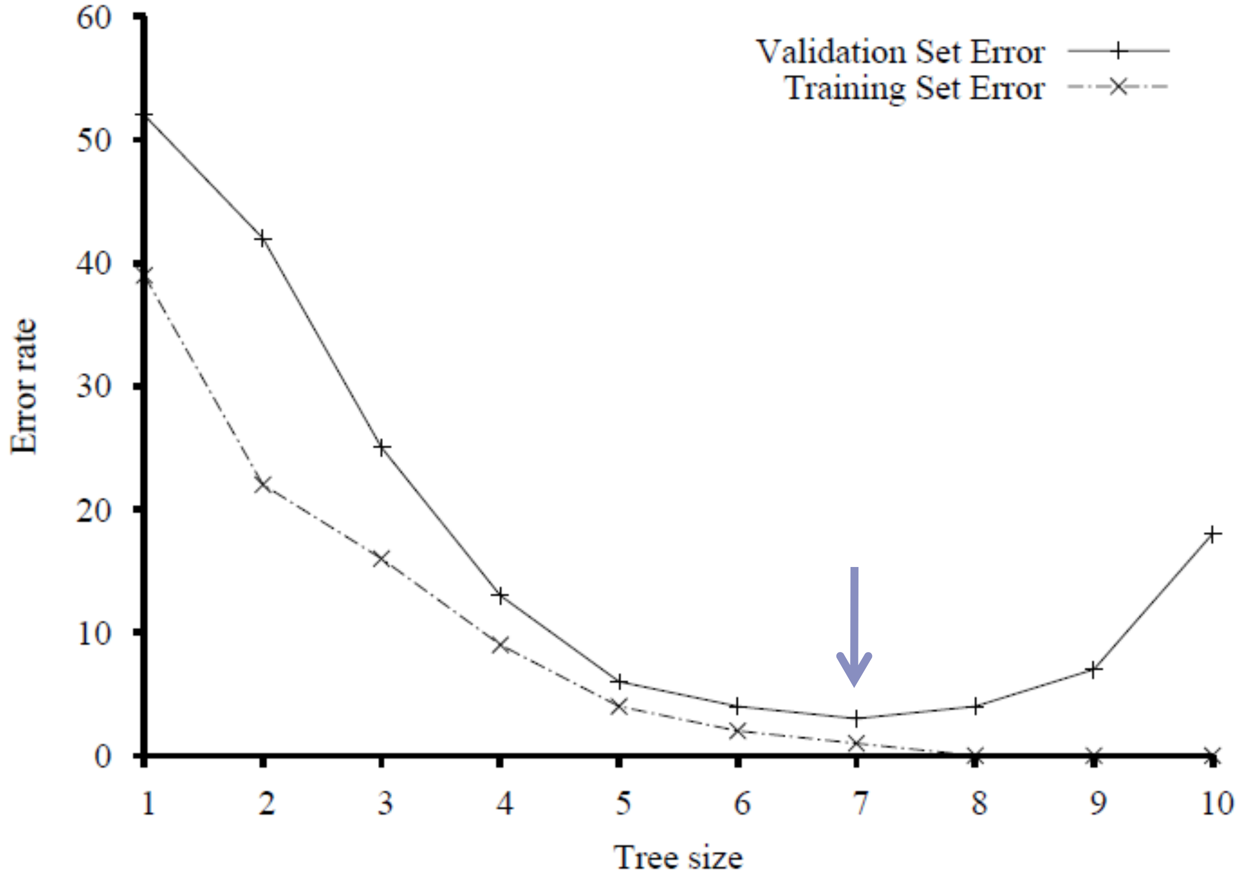
  (e.g., choosing the degree of the polynomial)

  - Lock away the test set
  - Divide the available data (without the test set) into a training set and a validation set to measure performance on unseen data as a way of selecting a good hypothesis



Model Selection: Complexity vs. Goodness of Fit

- Find a right-sized hypothesis that gives the lowest validation set error



Model Selection: Complexity vs. Goodness of Fit

function CROSS-VALIDATION-WRAPPER(*Learner*, *k*, *examples*) **returns** a hypothesis

local variables: *errT*, an array, indexed by *size*, storing training-set error rates
errV, an array, indexed by *size*, storing validation-set error rates

for *size* = 1 **to** ∞ **do**
errT[*size*], *errV*[*size*] ← CROSS-VALIDATION(*Learner*, *size*, *k*, *examples*)
if *errT* has converged **then do**
best_size ← the value of *size* with minimum *errV*[*size*]
return *Learner*(*best_size*, *examples*)

Number of folds

- Cross-validation finds the best *size* with the lowest error
- Then, a hypothesis of that *size* is generated using all the data (without holding out any of it)
- The returned hypothesis should be evaluated on a separate test set

From Error Rates to Loss

- Loss function:
  - Expresses utilities lost by predicting $h(x) = \hat{y}$ when the correct answer is $f(x) = y$:

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

  - A simplified version $L(y, \hat{y})$ independent of x is often used

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

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

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

  - Note: $0/1$ loss cannot account for the following case

$$L(spam, nospam) = 1, \quad L(nospam, spam) = 10$$

From Error Rates to Loss

- The expected generalization loss for a hypothesis h:

  ($\mathcal{E}$: set of all possible input-output examples)

  $$GenLoss_L(h) = \sum_{(x,y)\in\mathcal{E}} L(y, h(x))P(x, y)$$

  and the best hypothesis, h^*, is the one with minimum $GenLoss$

  $$h^* = \arg\min_{h\in\mathrm{H}} GenLoss_L(h)$$

- Since $P(x, y)$ is unknown, we can only estimate by empirical loss on a set E of N examples,

  $$EmpLoss_{L,E}(h) = \frac{1}{N}\sum_{(x,y)\in E} L(y, h(x))$$

  The estimated best hypothesis is

  $$\hat{h}^* = \arg\min_{h\in\mathrm{H}} EmpLoss_{L,E}(h)$$

From Error Rates to Loss

- Sources of loss:
  - Unrealizability (Bias):
    - $f \notin \mathcal{H}$
  - Variance:
    - Different hypotheses from different training sets (variance $\rightarrow 0$ as $|E| \rightarrow \infty$, if $f \in \mathcal{H}$)
  - Noise:
    - Often the observed labels y are the result of unknown attributes
  - Computational complexity:
    - Intractable to search the whole $\mathcal{H}$ when $\mathcal{H}$ is huge (especially for large-scale learning with millions of examples)

Regularization

- Explicit penalization of complex hypotheses on a training set

  $$Cost(h) = EmpLoss(h) + \lambda \underbrace{Complexity(h)}_{\text{regularization function}}$$

  $$\hat{h}^* = \underset{h \in \mathcal{H}}{\arg\min}\, Cost(h)$$

  - Often used for linear regression, where a good regularization function is the sum of the squares of the coefficients
  - We need cross-validation search with different λ rather than *size*
- MDL (minimum description length) approach: ☞
  - Both the empirical loss and the complexity are measured in bits (without λ):

    $$h^* = \underset{h \in \mathcal{H}}{\arg\max}\, P(data \mid h) P(h)$$

    Bits for encoding hypothesis + Bits for encoding error

    $(-\lg P(h))$ $(-\lg P(\text{data} \mid h))$
- Feature selection (e.g., χ^2 pruning) can also simplify models