CS5100 Foundations of Artificial Intelligence

Module 05 Lesson 08

Machine Learning 1

Some images and slides are used from CS188 UC Berkeley/AIMA with permission
All materials available at http://ai.berkely.edu / http://aima.cs.berkeley.edu

Caveats

- We are getting a quick introduction to Machine Learning (ML) here.
  - It’s marvelous, it’s exciting! ML in action is fun!
  - But this is just an introduction. Using this you can solve some ML problems.
  - It is not a substitute for a full ML course or more.
- Machine Learning is not magic
  - Don’t expect all problems to be solved when you wave the ML wand!
  - Think about how humans solve the problem.
  - Learn from the solutions
  - Think about explainability and bias
- Data is important
  - You need good, clean data.
  - Often the data preparation and cleaning could take 60-80% of the time to solve a problem.

Overview

- Learning Agents
- Supervised Learning
- Decision Trees
- Choosing Attributes and Entropy
- Choosing the Best Hypothesis

Learning Agents

Learning essential to navigate
unknown environments and situations

- e.g., a robot navigating an unknown space

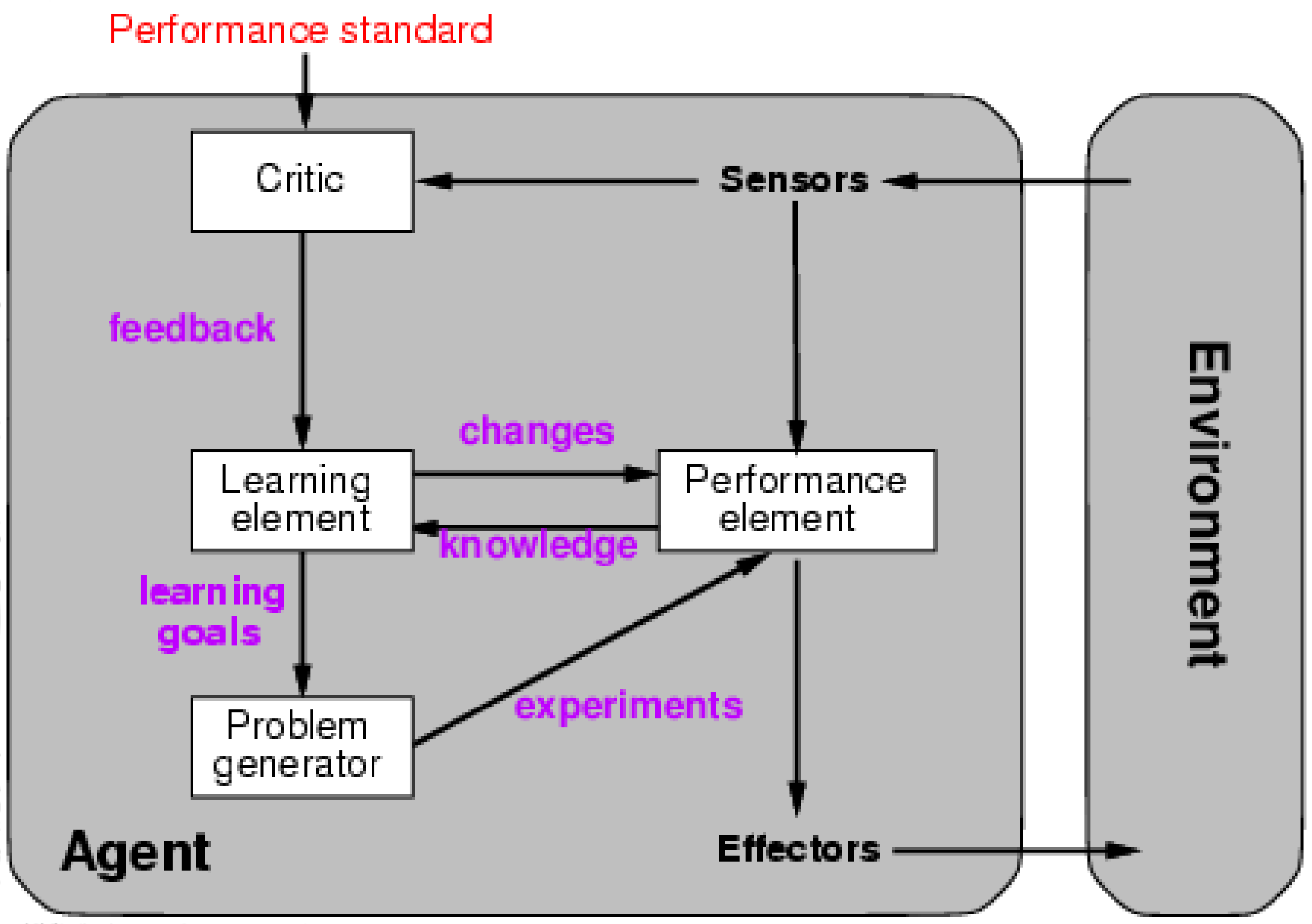
Learning useful to deal with changes over time

- e.g., GPS navigation systems dealing with traffic issues

Learning helps overcome human inability to solve problems

- e.g., learning to translate between languages

Learning Agents



Learning-Element Design

- Any component of a learning agent can be improved.

  What you can improve, and by how much depends on:
  - Which *component* is to be improved
  - What *prior knowledge* the agent has
  - What *feedback* is available to learn from
  - What *representation* is used

Learning Types & Feedback

Types of learning based on feedback available:

Unsupervised learning: correct answers not given
- E.g., clustering (bad traffic roads vs. better ones)

Reinforcement learning: occasional rewards
- E.g., wins in games

Supervised learning: correct answers for each example
- E.g., classification based on labeled examples (E.g., identifying spam)

Our Focus

Semi-supervised learning
- When you have issues because of noisy labels, or lack of labels

Supervised Learning .. 1

Simplest form: learn a function from examples

What is the next in the series?

1. 2, 4, 6, 8, …
2. 1, 1, 2, 3, 5, 8,…
3. 1, 2, 4, 8, …
4. 1, 4, 7, …

5. A, E, I, …
6. E, G, B, D, … (F, A, …)
7. 19, 23, 29, 31, …

8. 1,2, …
9. A, C, …

Supervised Learning .. 2 solutions

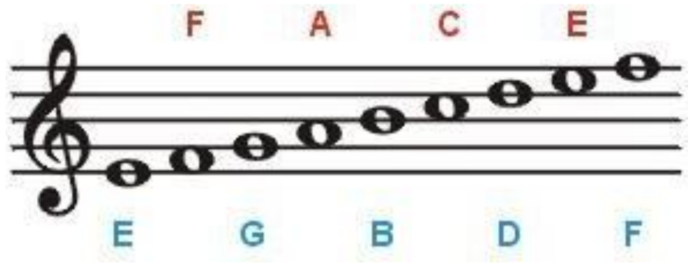
Simplest form: learn a function from examples

What is the next in the series?

1. 2, 4, 6, 8, … 10, 12… [even numbers, f(x) = 2*x]
2. 1, 1, 2, 3, 5, 8,… 13, 21, … [Fibonacci series f(n) = f(n-1) + f(n-2)]
3. 1, 2, 4, 8, … 16, 32… [powers of 2, f(x) = 2^x]
4. 1, 4, 7, … 10, 13… [f(x) = 3*x + 1]

5. A, E, I, … O, U [English vowels]
6. E, G, B, D, … F (F, A, C, E) [treble clef notes]
7. 19, 23, 29, 31, … 37, 41, 43… [primes]

8. 1,2, … ???
9. A, C, … ???



Supervised Learning ..3
Problem specification & notation

Simplest form: learn a function from N examples
$(x_1, y_1), (x_2, y_2)\ldots(x_N, y_N)$

Each y_i generated by an unknown function $y = f(x)$

f is the (true) target function

Problem: find a hypothesis *h* such that $h \approx f$,
given a training set of examples
(and measure accuracy on a test set of examples)

Supervised Learning ..4
Generalization & Consistency

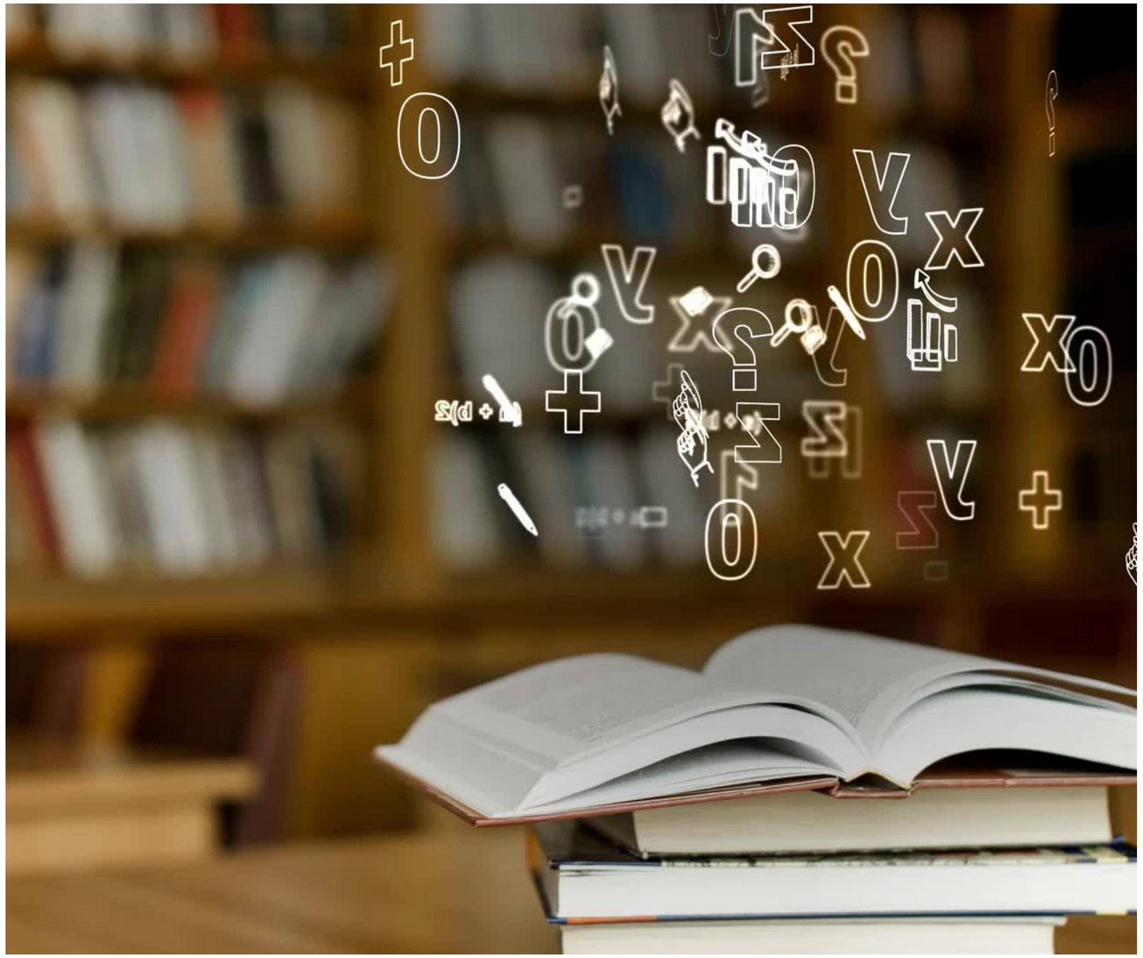
Highly simplified model of real learning

- Ignores prior knowledge
- Assumes examples are given

Hypothesis h **generalizes** well if it correctly predicts y for novel (new) examples

- Generalization: key concept in ML

h is **consistent** if it agrees with *f* on all examples



Supervised Learning ..5
Hypothesis Space

Function h selected from a Hypothesis Space $\mathcal{H}$

Learning problem is realizable if

hypothesis space $\mathcal{H}$ contains the true function f

Issue: Which h to choose from *multiple consistent hypotheses* (functions) in the hypothesis space?

Supervised Learning ..6
Classification vs. Regression

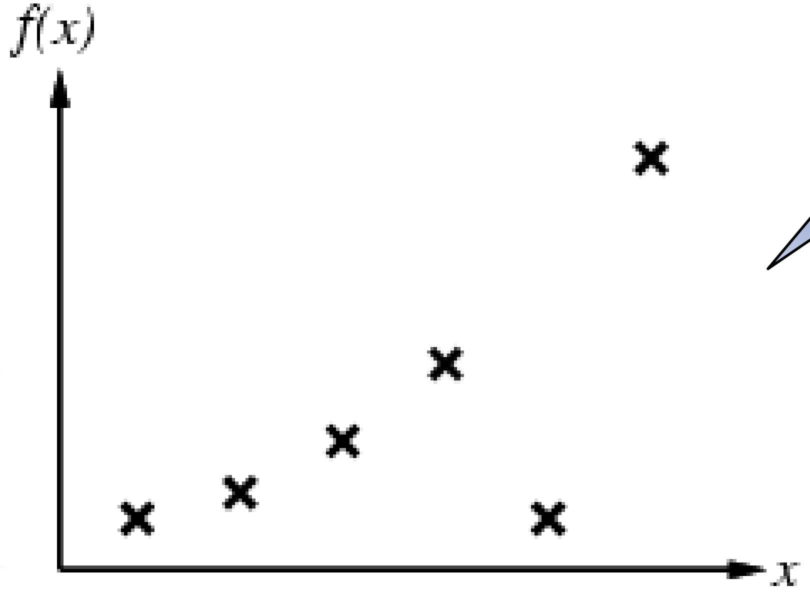
If output y is a discrete value like spam, cloudy etc.,
then it is a **classification** problem

If output y is a number (E.g., predicted high temperature, or stock price),
then it is a **regression** problem

Supervised Learning Method ..1

Construct/adjust *h* to agree with *f* on training set

[via curve fitting]:



Suggestions?

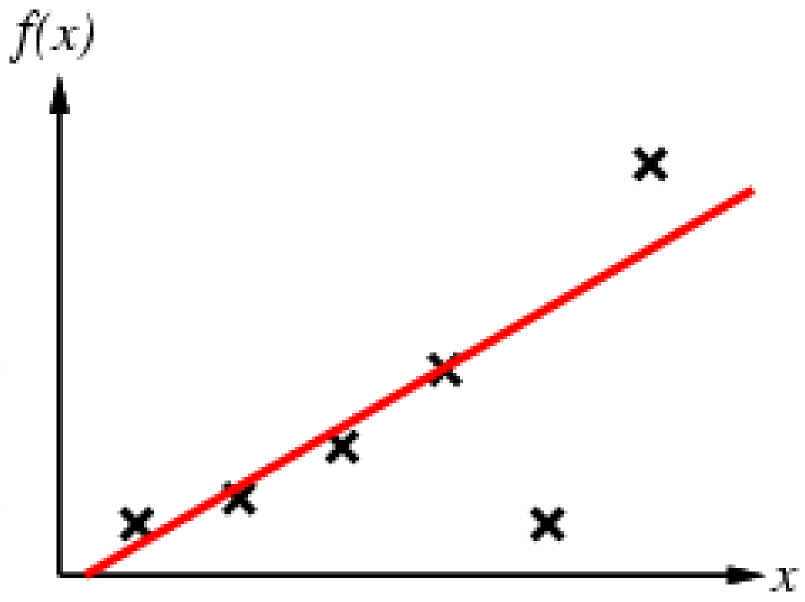
Supervised Learning Method ..2

Construct/adjust *h* to agree with *f* on training set

- *h* is consistent if it agrees with *f* on all examples

E.g., curve fitting: straight line (approximation)

y = mx + c



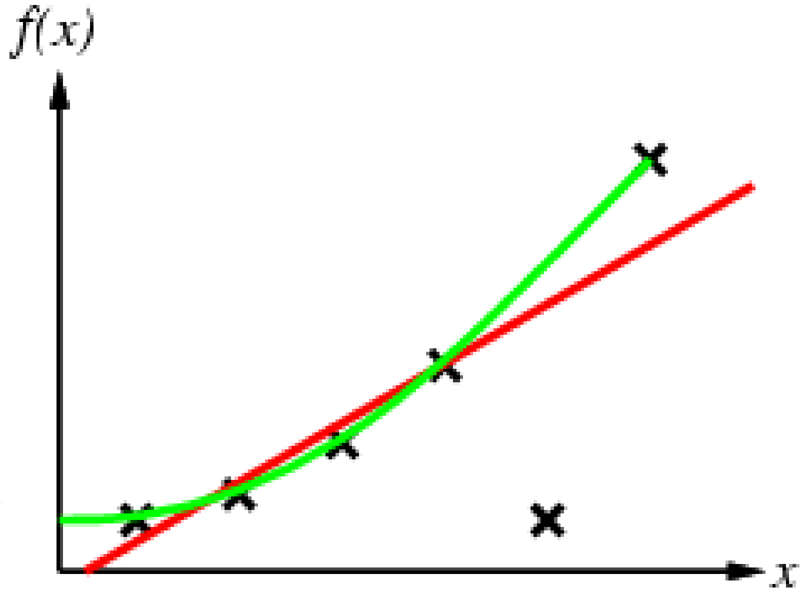
Supervised Learning Method ..3

Construct/adjust *h* to agree with *f* on training set

- *h* is consistent if it agrees with *f* on all examples

E.g., curve fitting: try more complicated (higher-degree)

polynomial curves -- more ‘expressive’



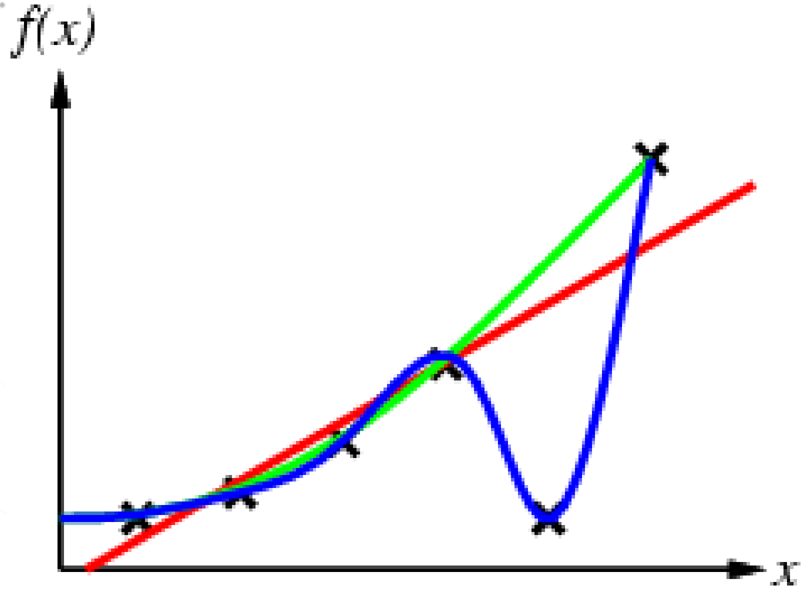
$$a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0 .$$

Supervised Learning Method ..4

Construct/adjust *h* to agree with *f* on training set

- *h* is consistent if it agrees with *f* on all examples

A higher-degree polynomial, maybe add a sin() component…

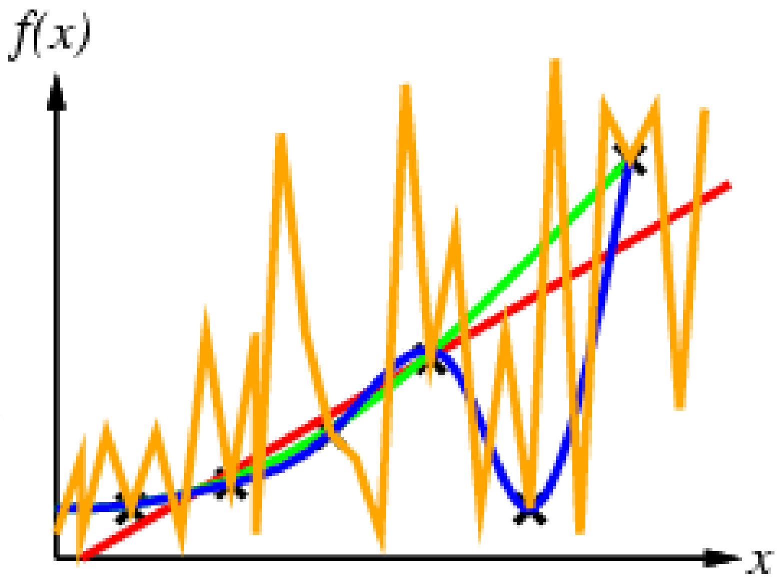


Supervised Learning Method ..5

Construct/adjust *h* to agree with *f* on training set

- *h* is consistent if it agrees with *f* on all examples

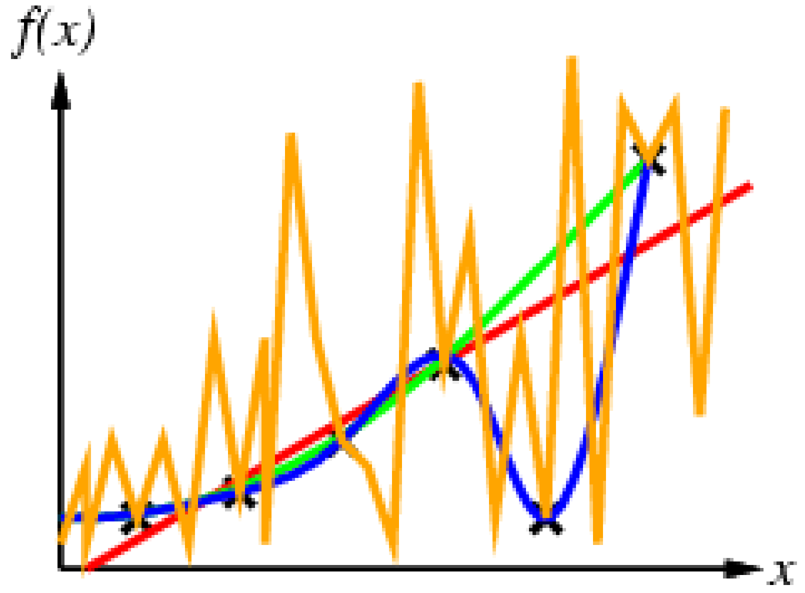
Even higher-degree polynomial



Supervised Learning Method ..6

Construct/adjust *h* to agree with *f* on training set

- *h* is consistent if it agrees with *f* on all examples



Which curve (hypothesis) to choose?

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

Supervised Learning Method ..7

Trade-off between
complex hypotheses that fit the training data well,
and simpler hypotheses that may generalize better

Also:
Trade-off between
expressiveness of hypothesis space, and
complexity of finding a good hypothesis there

Decision Trees ..1 Intuition

Problem: decide whether to wait for a table at a restaurant

What would you base your decision on?

How would you decide?

Decision Trees ..2

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

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

6 Booleans,
2 categorical with 3 values each,
2 categorical with 4 values each

$2^6 \times 3 \times 3 \times 4 \times 4 = 9216$ combinations

Attribute-based Representations

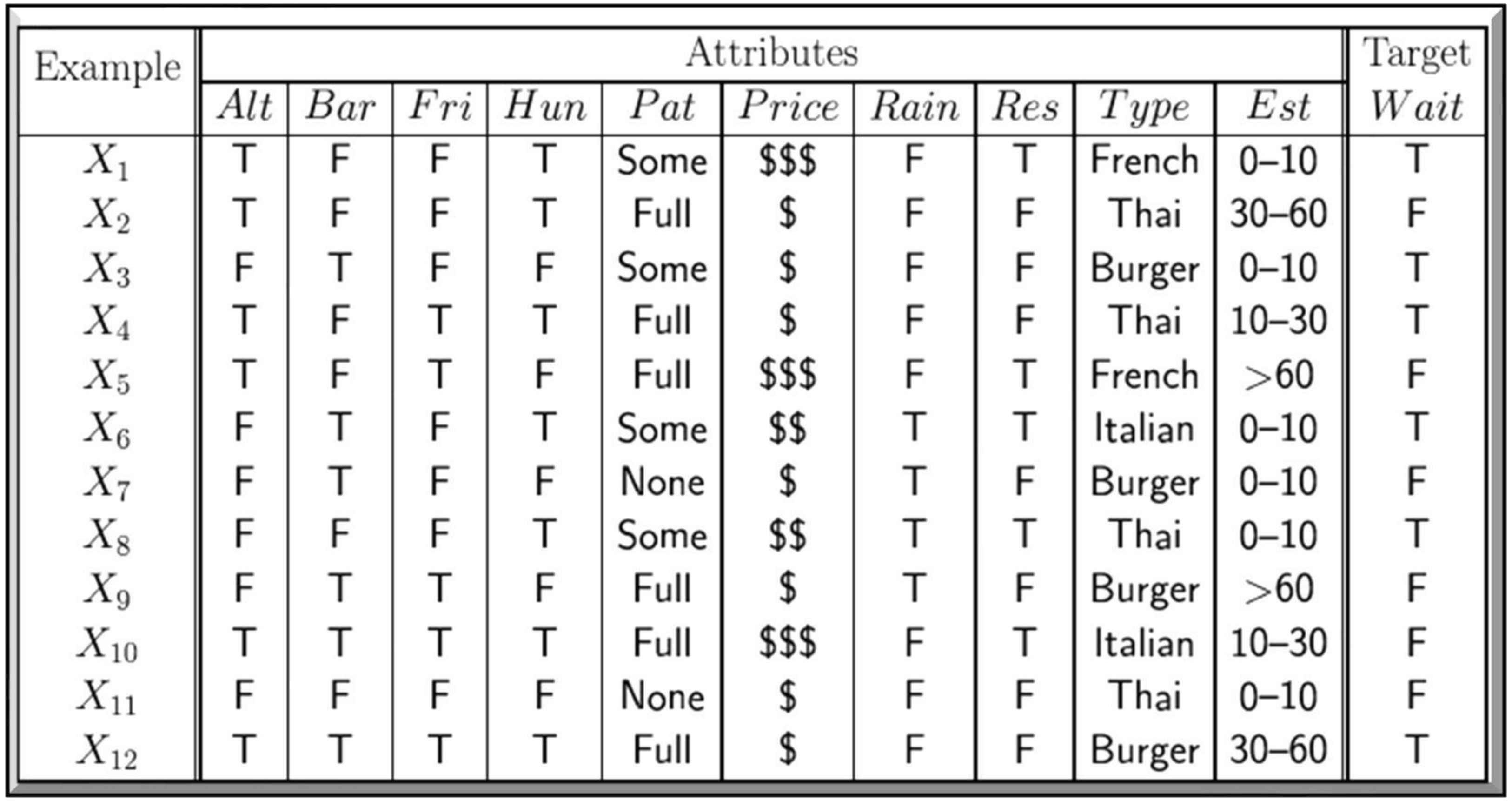
Each row is an example

Examples described by attribute values (Boolean, discrete, continuous)

E.g., situations where I will or won't wait for a table:

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

Only 12 of 9216 possible combinations!



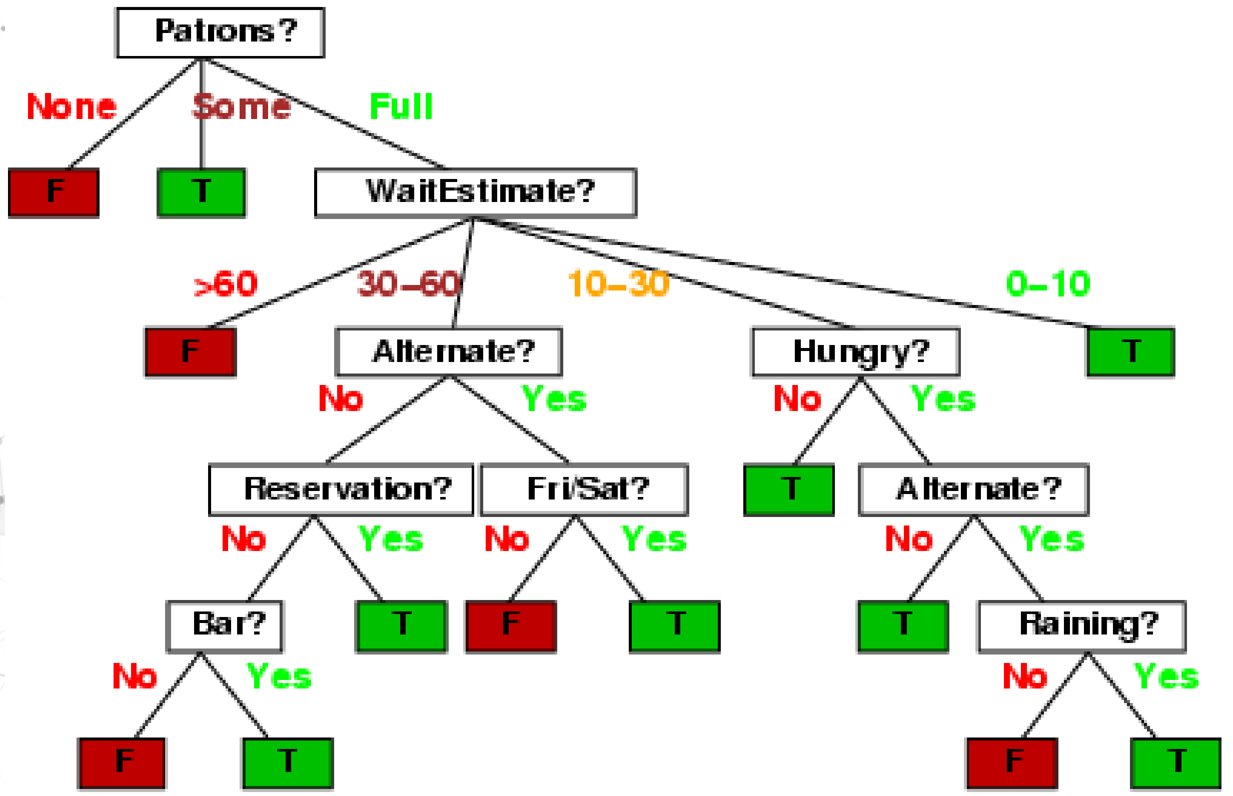
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6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

Decision Trees ..3

Does the order of attributes matter?

One possible representation for hypotheses (ignoring Price, Type)

-- "true" tree for deciding whether to wait:

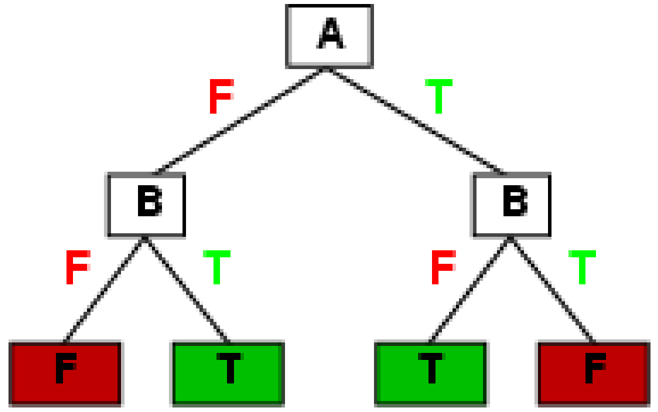


Decision Tree Expressiveness

Decision trees can express any function of the input attributes,
but some functions (e.g., majority) harder to represent

E.g., for Boolean functions, truth table row → path to leaf:

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



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*)
but it probably won't generalize to new examples

Prefer to find more compact decision trees (that generalize)

Size of Hypothesis Spaces

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

= number of distinct truth tables we can write down

Number of rows in a truth table with n attributes = 2^n

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

Can be huge
E.g., with 6 Boolean attributes, there are

18,446,744,073,709,551,616 trees

Hypothesis Spaces & Expressiveness

More expressive hypothesis space

increases chance that target function can be expressed,

for example, with sin, sqrt , e^x etc. functions

increases number of hypotheses consistent with training set

➔ but may result in worse predictions

Decision Tree Learning algorithm .. 1

Aim: find a small tree, consistent with the training examples (**X**,y)

Idea: Greedy algorithm:

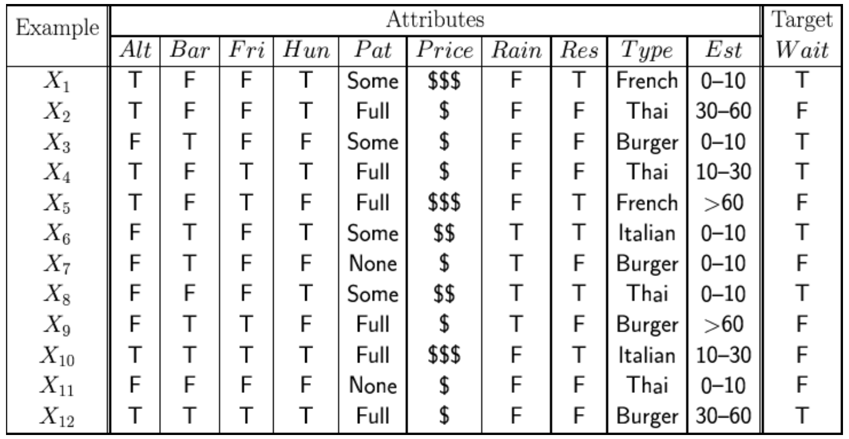
(Recursively) choose ***most significant*** attribute as root of (sub)tree

However: What *is* the most significant attribute?

One that makes the most difference to classifying an example

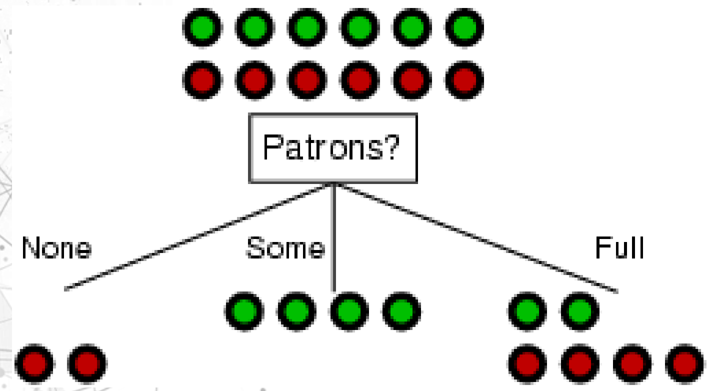
Might lead to correct classifications with fewer tests,
shorter paths and therefore a shallower tree

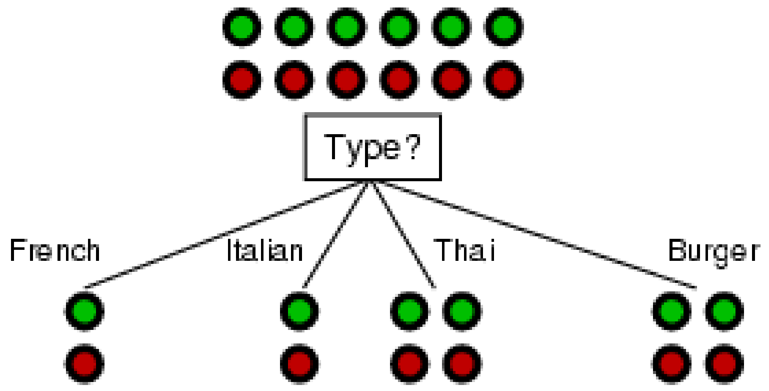
Choosing an Attribute



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

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





Which is a better choice?

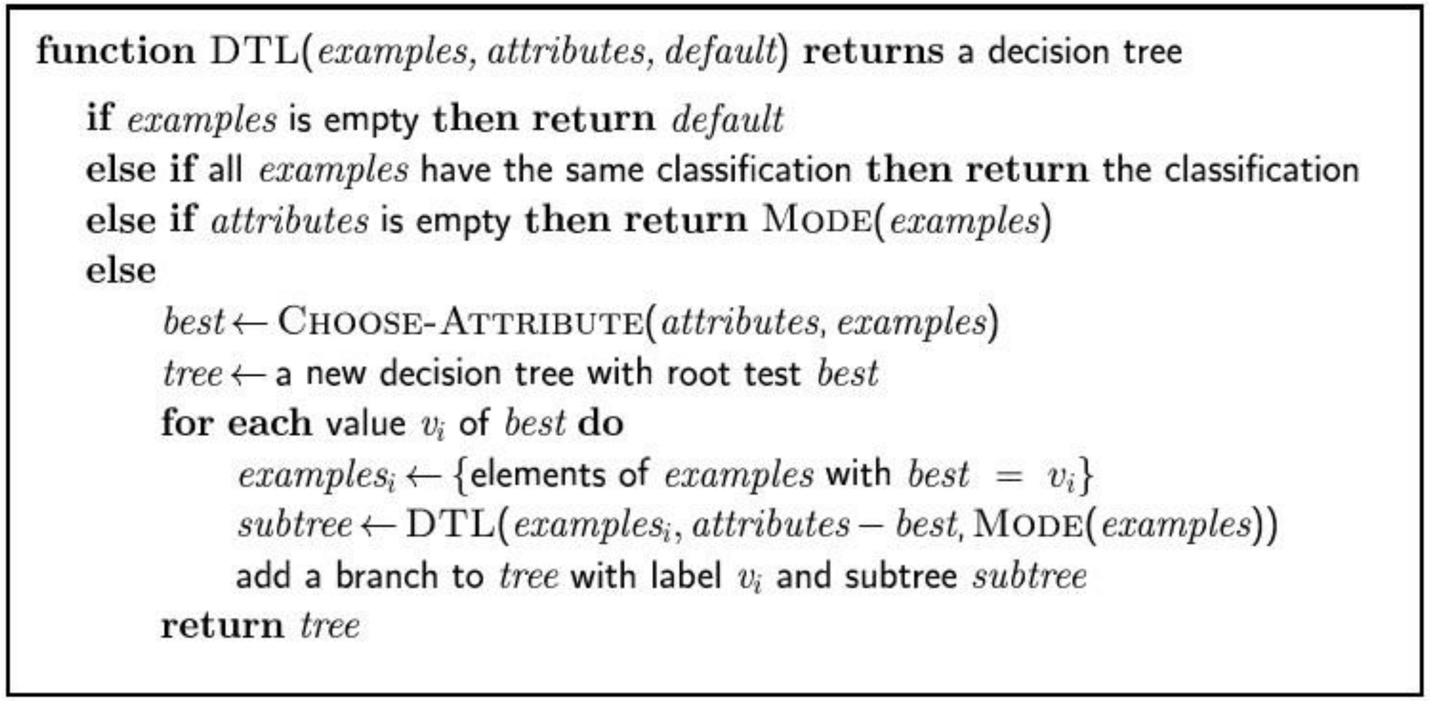
Patrons?

Decision Tree Learning algorithm .. 2

Aim: find a small tree, consistent with the training examples (**X**,y)

Idea: Greedy algorithm:

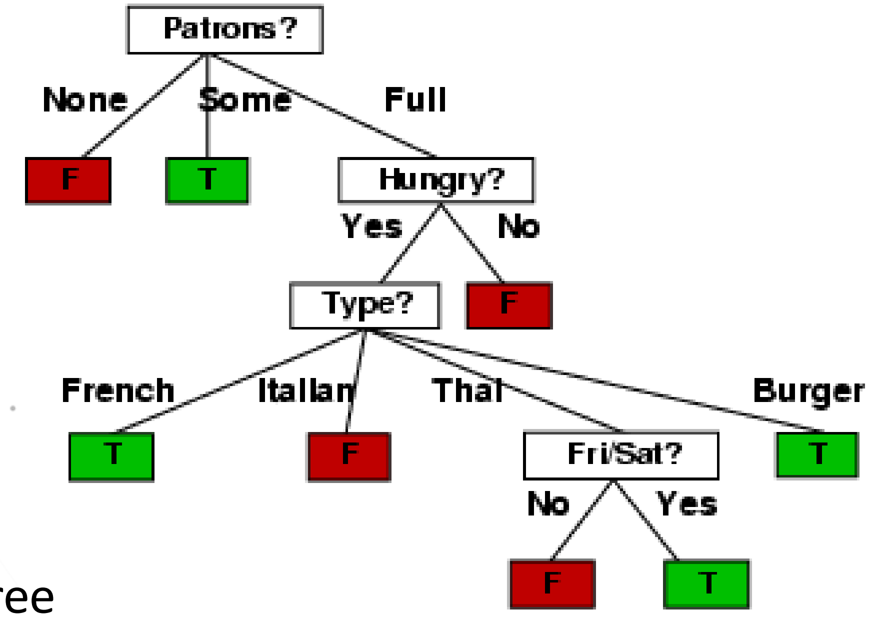
(Recursively) choose ***most significant*** attribute as root of (sub)tree



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

Decision Tree Learned

Decision tree learned from the 12 examples (rows in table):

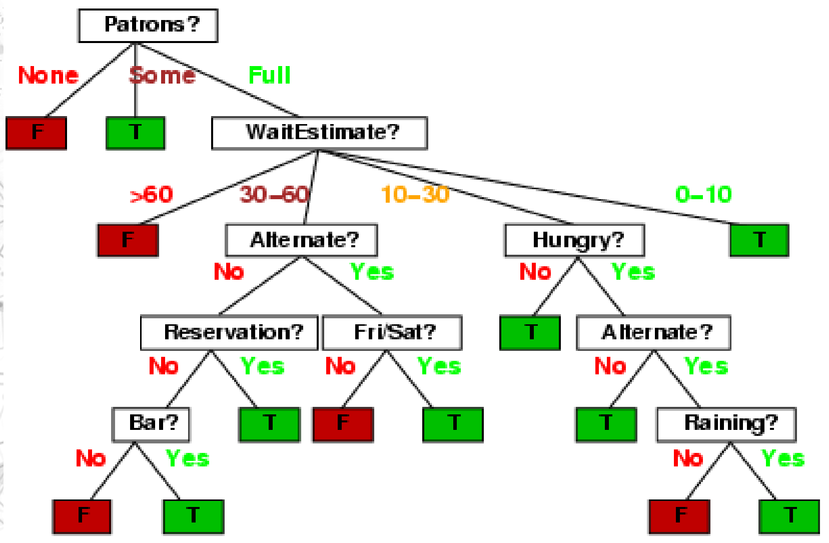


Much simpler than “true” tree

More complex hypothesis not justified by small amount of data

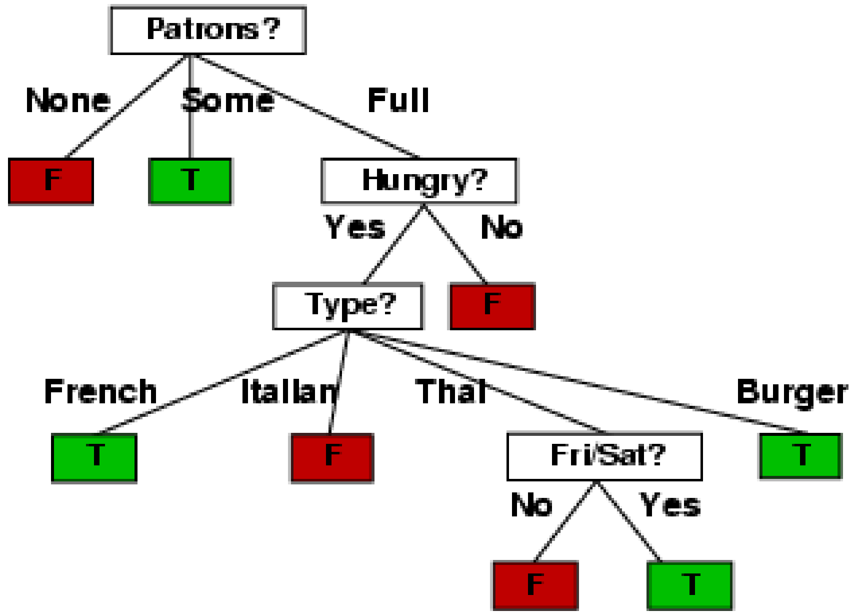
Decision Tree
True vs. Learned

'True' Tree*



* ignoring Price, Type

Learned Tree



Entropy

How do we implement `Choose-Attribute` in the DTL algorithm?

- Good attributes divide up universe clearly into sets
  that are each (almost) all of one class, E.g., all positives or all negatives
- Bad attributes have a mix E.g., mix of positives & negatives

How do we quantify this?

➔ use Entropy

Entropy:

- Measure of disorder (Physics)
- Measure of uncertainty (Information Theory)

Entropy Intuition

Random variable with one value
(E.g., coin, always ‘Heads’)

- No uncertainty. Hence, 0 entropy

Fair coin, which comes up heads or tails
with equal probability

- “1 bit” of entropy

4-sided dice

- “2 bits” of entropy

Biased coin, which comes up heads 99% of the time?

- Very little uncertainty, close to 0 bits

Entropy Definition

Given a random variable V, with values v_1, v_2… v_k, each with probability $P(v_i)$,

Entropy $H(V) = \Sigma_{i=1} P(v_i) * (1/\log_2 P(v_i))$

$= - \Sigma_{i=1} P(v_i) * \log_2 P(v_i)$ [Note -ve sign; also note log to the base 2]

Examples

- H(fair coin) = $-(0.5 \log_2 (0.5) + 0.5 \log_2 (0.5)) = 1$ bit
- H(biased coin) = $-(0.99 \log_2 (0.99) + 0.01 \log_2 (0.01)) \sim 0.08$ bit

Note: Other books use I(V) instead of H(V)

Entropy
Additional Definitions ..1

Entropy H(V) =

$- \Sigma_{i=1} P(v_i) * \log_2 P(v_i)$

Probabilities: q, (1-q)

Define entropy of Boolean random variable that is true with probability q:

$$B(q) = -(q * \log_2 (q) + (1 - q) * \log_2 (1 - q))$$

E.g., H(biased coin) = B(0.99) ~ 0.08 bit

Entropy
Additional Definitions ..2

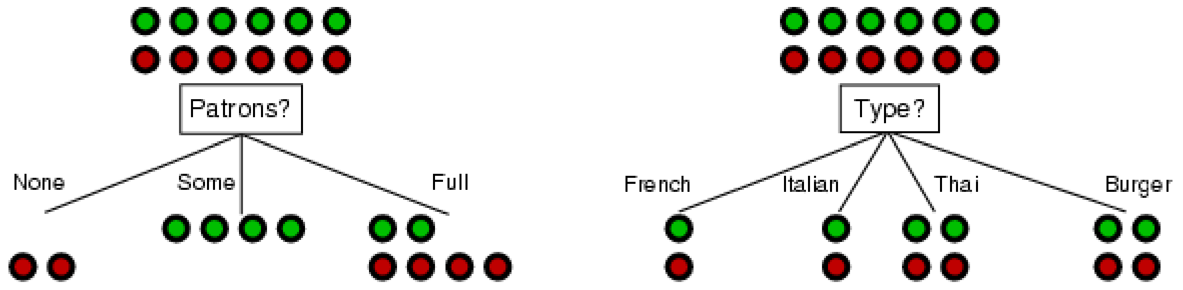
Given a training set with p positive examples,
and n negative examples,
Probability of positive example: p/(p+n)

- H(Goal) = B(p/(p+n))
- Example: p = n = 6, H(Goal) = B(6/12) = B(0.5) = 1 bit

Testing a single attribute A gives some information, reducing overall entropy a little bit

- How much? Look at entropy *after* testing A

Entropy Remainder



- Consider an attribute A, with d distinct values, dividing the training set into subsets E_1, E_2… E_d.
- Each subset E_k has p_k positive and n_k negative examples.
- If we choose to go along $E_{k,}$ we will need $B(p_k/(p_k+n_k))$ bits of info to answer the question, to see how much entropy remains.
- A randomly chosen example from the training set has the kth value for attribute, that is, is in E_k with probability $(p_k + n_k) / (p + n)$
- Hence expected entropy remaining after testing attribute A is:

$$\text{Remainder}(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right)$$

Entropy
Information Gain

Information Gain (IG) or expected reduction in entropy from the attribute test:

Gain(A) = H(Goal) – Reminder(A)

= B(p/(p+n)) – Remainder(A)

This gives us the rule to implement `Choose-Attribute`: Choose the attribute with the largest Gain!

Information Gain Restaurant Example

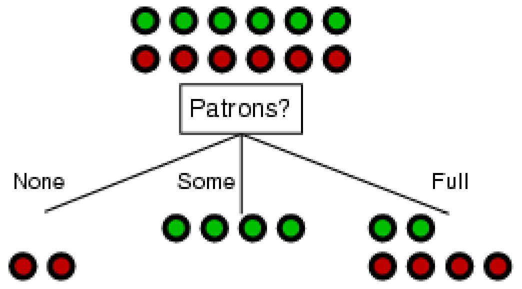
- For the training set, $p = n = 6$, $H(Goal) = 1$ bit
- Gain for the attributes *Patrons* and *Type is*:

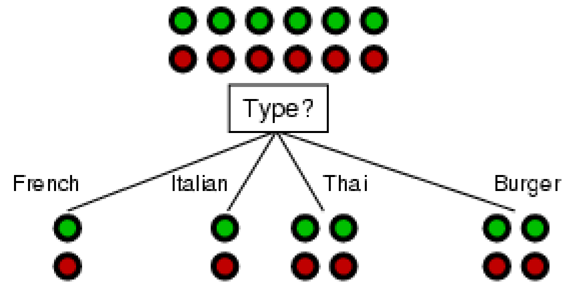
None Some Full

$$\text{Gain(Patrons)} = 1 - [\boxed{\frac{2}{12} * B(\frac{0}{2})} + \boxed{\frac{4}{12} * B(\frac{4}{4})} + \boxed{\frac{6}{12} * B(\frac{2}{6})}] \sim 0.541$$

$$\text{Gain(Type)} = 1 - [\frac{2}{12} * B(\frac{1}{2}) + \frac{2}{12} * B(\frac{1}{2}) + \frac{4}{12} * B(\frac{2}{4}) + \frac{4}{12} * B(\frac{2}{4})] = 0$$

- *Patrons* has the highest Gain of all attributes and therefore, chosen by the DTL algorithm as the root
- Further attributes chosen similarly, based on Gain





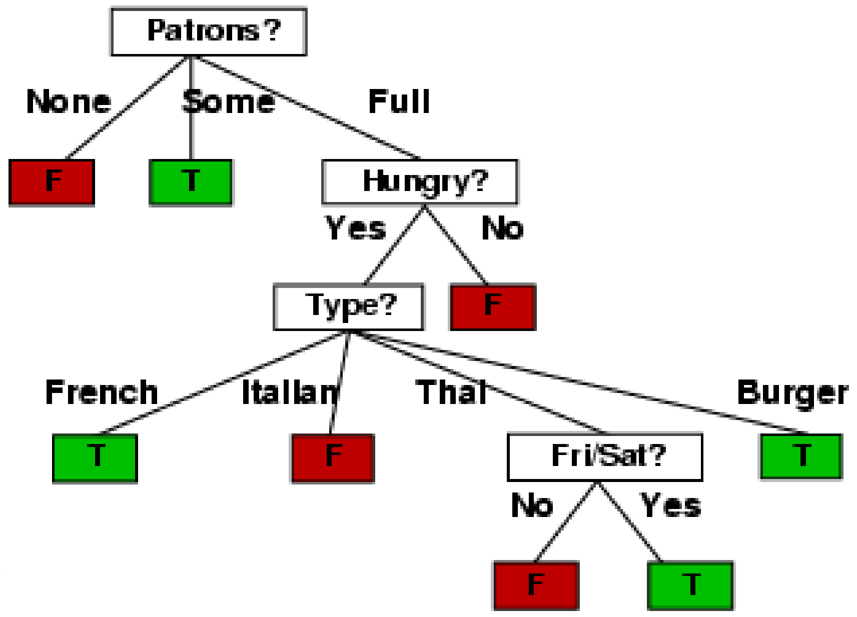
$$H(Goal) = B(p/(p+n))$$

$$\text{Remainder}(A) = \sigma_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{p_k + n_k})$$

https://www.youtube.com/watch?v=_PG-jJKB_do

Learned Decision Tree

Decision tree learned from the 12 examples (same as before):



Overfitting

- Sometimes the DTL algorithm will see patterns where none exist. Will create noisy branches – in a process termed overfitting
- Like examples we saw in curve-fitting using high-degree polynomials
- Overfitting:
  - More likely as hypothesis space and number of input attributes grows
  - Less likely as number of examples increases
- Can combat overfitting with decision tree pruning, chopping off nodes which are clearly irrelevant

Broadening applicability of decision trees

Continuous/integer-valued attributes

- Instead of a Yes-No split as in Boolean variables, or a split along categorical values, need a split in value
  - E.g., Instead of nodes with attribute = value, we may have attribute > value etc.
- Split point found where information gain is highest
- Finding split points usually the most expensive part of
  real-world decision DT applications

Regression trees

- Like classification trees, but has a linear function of numerical attributes at each leaf node

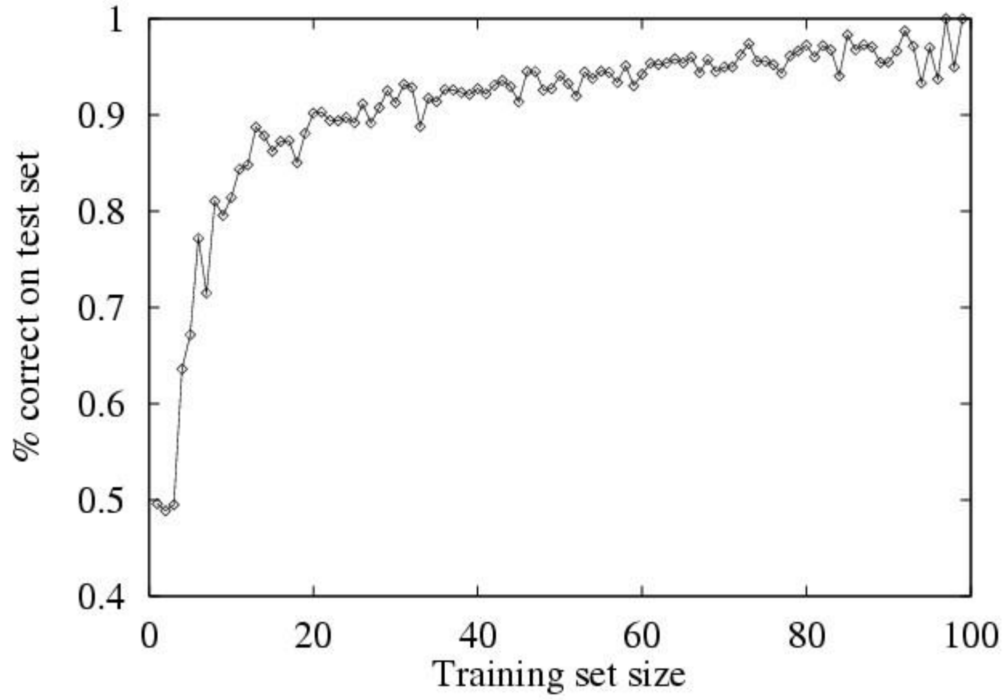
Performance Measurement

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



Decision Trees Summary

Several commercial and open-source decision tree systems available

Thousands of decision tree systems developed

Great advantage: being able to explain reasoning behind a decision (unlike NNs, SVM etc.) -- often a requirement

Evaluating Hypotheses ..1

Goal: Learn a hypothesis that *fits future data best*

Future data?

- Stationarity Assumption: there is a probability distribution over examples that is the same over time: $P(E_j) = P(E_{j+1}) = P(E_{j+2})$

- Each example data point is sampled from that distribution, and is independent of previous examples

$$P(E_j) = P(E_j \mid E_{j-1}, E_{j-2}, \ldots)$$

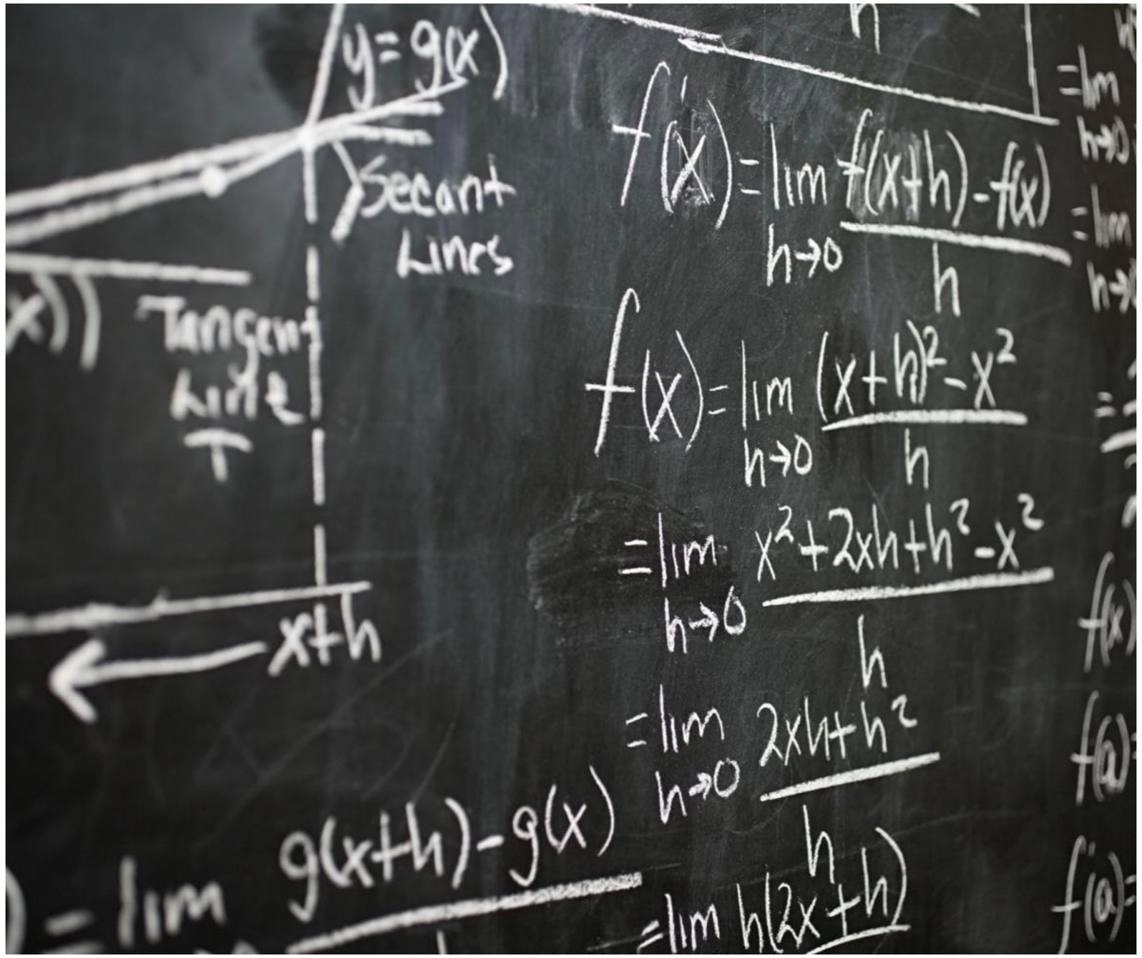
- Each example has an identical prior probability distribution
  ➔ examples satisfying these are
  i.i.d.: independent and identically distributed

Evaluating Hypotheses ..2

Best fit?

- Error rate: Proportion of mistakes made,
  i.e. where h(x) != y
  for a given example (x,y)

- Low error rate on training set does not mean the hypothesis generalizes well.
  - Need to test on unseen examples – cross-validate!


y=g(x)
Secant Lines
Tangent Line
x+h
$f'(x)=\lim_{h\to0}\frac{f(x+h)-f(x)}{h}$
$f'(x)=\lim_{h\to0}\frac{(x+h)^2-x^2}{h}$
$=\lim_{h\to0}\frac{x^2+2xh+h^2-x^2}{h}$
$=\lim_{h\to0}\frac{2xh+h^2}{h}$
$=\lim h(2x+h)$
$\lim g(x+h)-g(x)$

Cross-validation methods ..1

Holdout cross-validation

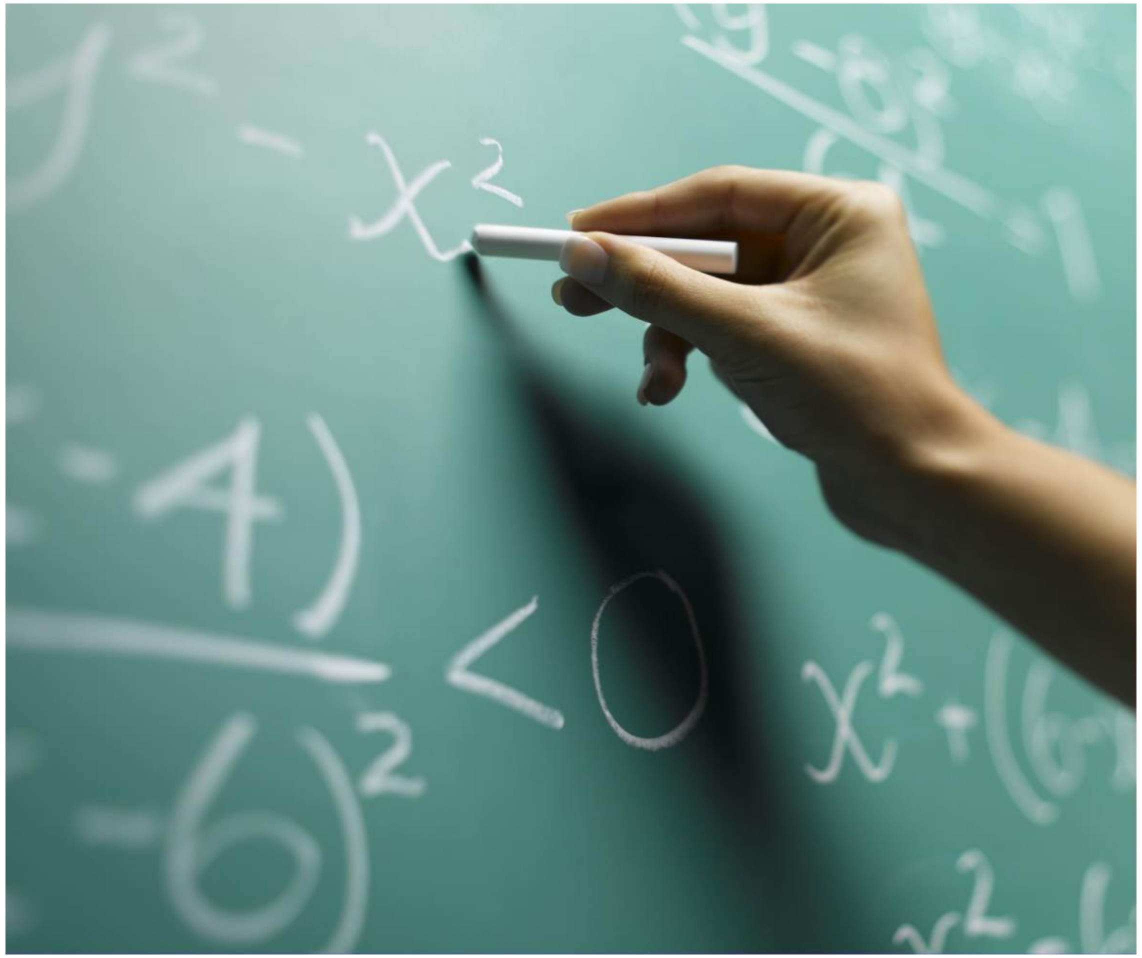
1. Split available examples into training data and test data
2. Learn h from the training data, and compute accuracy against the test data

Disadvantage:

Not all example data is used for learning h

- If 50% used for test, then hypothesis may be poor
- If 10% used for test, then may not get a good measure of accuracy

Usually split is 2:1, 3:1 or 4:1.



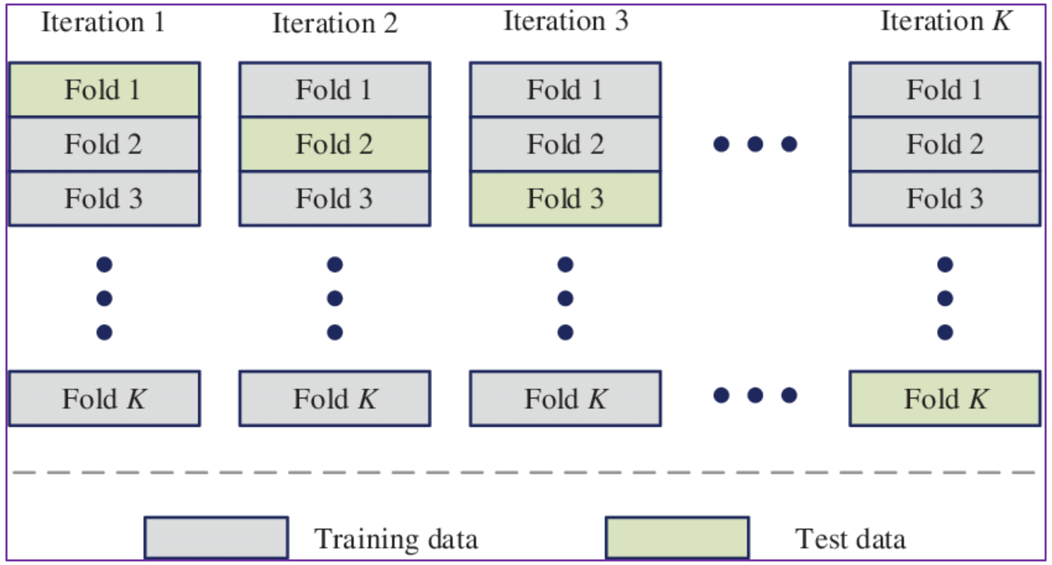
Cross-validation methods ..2

k-fold cross-validation

- Tries to make better use of all data
  1. Keep test data aside
  2. Split rest of data into k equal subsets
  3. Perform k rounds of training, for i in 1 to k:
     a. In each round, test on the subset i,
        after training on the remaining (k-1) subsets
     b. Average accuracy scores across all k rounds;
        should be a better estimate

Usual values of k are 3, 5, 10

Disadvantage: k-times computation time

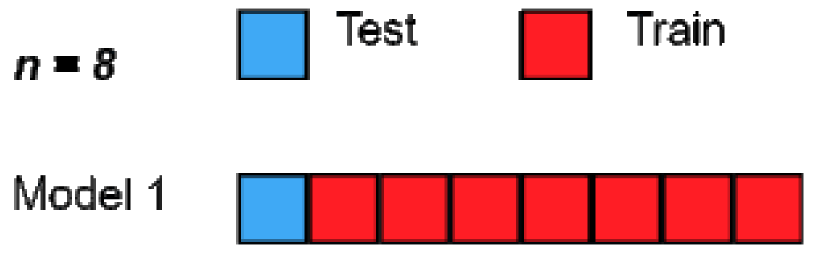


Cross-validation methods ..3

Leave-one-out cross-validation (LOOCV)

- Version of k-fold cross-validation, where k = n
- That is, leave sample for testing, and train on rest; do this n times, and average scores across the n trials

Disadvantage: n-times computation time



From https://en.wikipedia.org/wiki/Cross-validation_(statistics)

“Peeking” at test data

Users may inadvertently “peek” at test data

How can this occur? Say…

1. User may use some parameters to fine-tune learning algorithm
2. May generate hypotheses for various values of these parameters, and evaluate these against test set
3. May choose best hypothesis based on these tests.
4. But if hypothesis chosen based on perf against test data… test data info has leaked into learning algorithm

➔ User has peeked at test data !

Validation Sets

How do we avoid peeking?

Best to have a validation set in addition to training and test sets

- Learn using training data
- Refine by testing on validation data
- Test on really held-out test data

Could have train:validation:test in ratio 3:1:1

Model Selection
Complexity vs. Goodness of fit

Finding best hypothesis:

Model selection to define the hypothesis space

+

Optimization to find best hypothesis in that space

Start with simple models parameterized by size, move to more complex models, evaluate to find the right space

Size can be #nodes

Generate hypothesis at optimal size, evaluate on held-out data

Loss functions

Loss function $L(x, y, Y)$ is utility lost by predicting $h(x) = Y$, when the correct answer is $f(x) = y$.

- Simpler version $L(y,Y)$

Not all errors or losses are the same; may be worse to classify ok mail as spam, rather than spam as non-spam

Can specify

- Absolute loss: $L_1(y,Y) = |y - Y|$
- Squared error loss $L_2(y,Y) = (y - Y)^2$
- 0/1 loss $L_{0/1}(y,Y) = 0$ if $y = Y$, else 1

Can define generalization loss, estimate empirical loss

Empirical loss for a set of examples is a weighted sum of losses on examples

Best hypothesis is one with minimum empirical loss

Why does best hypothesis differ from true function?

Unrealizability: f may not be realizable, not in $\mathcal{H}$, or may not be as preferred

Variance: Different hypotheses returned for different example sets,
leading to different accuracies on test data.
Variance decreases as the number of examples increases

Noise: f may be nondeterministic or noisy;
different values of f(x) returned every time x occurs.
May be because of attributes not listed in x.

Computational Complexity: Can be intractable to search hypothesis space.
May be able to only explore part of the space,
or do a greedy search, so we only approximate f.

With large-scale learning, generalization error today dominated by limits of computation.
So though we can find h close to f, may have to settle for an approximation.

Regularization

Can combine empirical loss and a term for complexity of hypothesis into a total cost

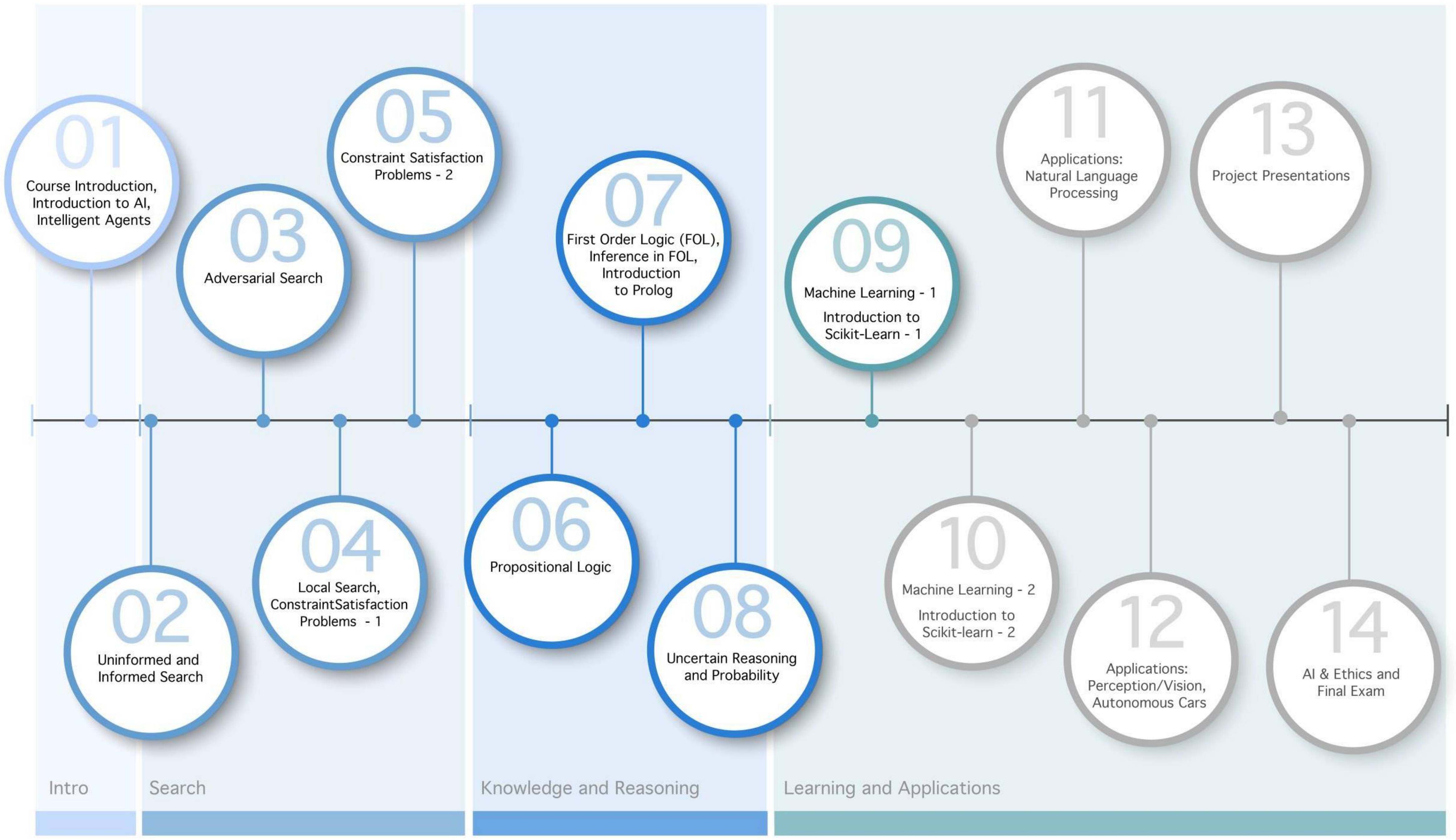
- Cost(h) = EmpLoss(h) + λ * Complexity(h)

Best hypothesis is the one with minimum Cost.

Can use cross-validation with different values of λ

Regularization: process of penalizing complex hypotheses

- Good regularization function (complexity term) for polynomials =
  sum of squares of coefficients


01
Course Introduction, Introduction to AI, Intelligent Agents
02
Uninformed and Informed Search
03
Adversarial Search
04
Local Search, ConstraintSatisfaction Problems - 1
05
Constraint Satisfaction Problems - 2
06
Propositional Logic
07
First Order Logic (FOL), Inference in FOL, Introduction to Prolog
08
Uncertain Reasoning and Probability
09
Machine Learning - 1
Introduction to Scikit-Learn - 1
10
Machine Learning - 2
Introduction to Scikit-learn - 2
11
Applications: Natural Language Processing
12
Applications: Perception/Vision, Autonomous Cars
13
Project Presentations
14
AI & Ethics and Final Exam
Intro
Search
Knowledge and Reasoning
Learning and Applications