CSCI 561 - Foundation for Artificial Intelligence

Discussion Section (Week 11)

PROF WEI-MIN SHEN SHEN@ISI.EDU

What is Learning?

No hard and fast definition

- Prototype (or fuzzy) definition

Modification of a system

- by itself
  - Not simply programming/surgery by others
  - Although others (teachers?) may be involved
- that improves its performance
  - Not random or detrimental changes (forgetting, etc.)
  - Although some learning may, at least temporarily, cause a performance decrement
    - Or the impact on performance may become apparent only later, or not at all if the right situation does not arise
- for the long term
  - Not simply a transient change

Why Learning?

Critical for autonomy

- i.e., for systems operating without human intervention

Essential for unknown environments

- i.e., when designer lacks omniscience

Useful for system construction

- i.e., expose agent to reality rather than writing it down

Central piece of human cognition

- Understanding learning is key in cognitive modeling

But considered undesirable in some applications

- *Unpredictable outcomes, unverifiable (if learning is online) and untrustable*

Rote Learning (Memorizatio

Duck
Goose
Pig
Elephant
Horse
Azalea
Computer

Perhaps the simplest form of learning conceptually

- Given a list of items to remember,
- Learn the list so that can respond to queries about it
    - Recognition: Was Horse in list? Rhino? Goose? Azalea? Came
    - Recall: What items did you see?
    - Cued Recall: What animals did you see?

Relatively simple to implement in computers (except cued)

- Can improve accuracy by remembering what is perceived
- Can improve efficiency by *caching* computations

Core research topic in human learning (semantic memory)

- Memorization is a relatively difficult skill for people

Generalization

- A major step beyond rote learning is to create *generalizations* across items seen

- Adds some *understanding* to memorization

- One aspect of larger *assimilation* process, of relating new knowledge/items/experiences to what is already known

Typical generalization/learning problems

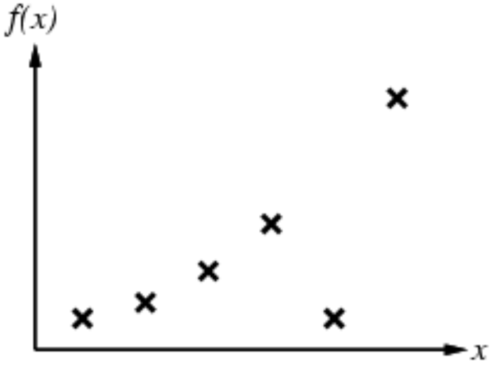
- *Clustering* (*unsupervised* learning)
  - uses commonalities and differences among items to group them in some useful way
- *Concept formation* (*supervised* or *inductive* learning**)**
  - uses commonalities among items assigned to a class, and differences with items assigned to other classes, to develop general definitions for class membership
- *Semi-supervised learning*
  - uses a small number of classifed examples with a larger number of unclassified ones
- *Reinforcement learning*
  - acquires general *policies* for controlling behavior from external reinforcement signals concerning the utility of states reached in environment

Approaches tend to be *knowledge lean* in general

Inductive Learning

Learn a *target function f* from classified *examples*

- Each example is a pair (*x, f*(*x*))
- If both *x* and *f* are numeric, this is traditional function fitting
- AI traditionally focused on cases where *x* is described discretely/symbolically and *f* is either Boolean or discrete
- Modern methods may do both

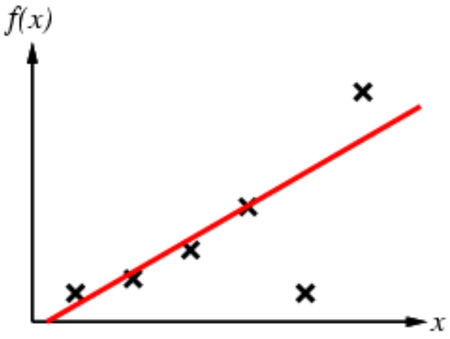


f(Robin: WarmB & LayE & Fly)=Bird
f(Bat: WarmB & ¬LayE & Fly)=Mammal
f(Pteranodon: ColdB & LayE & Fly)=Reptile
f(Ostrich: WarmB & LayE & ¬Fly)=Bird

Inductive Learning

Problem:

- **Find a hypothesis *h* in *Hypothesis Space* (H), e.g.**
  - Univariate polynomial functions
  - Boolean feature vectors
- Such that *h* ≈ *f*
- Given *training set* of examples



WarmB & LayE ⇒ Bird

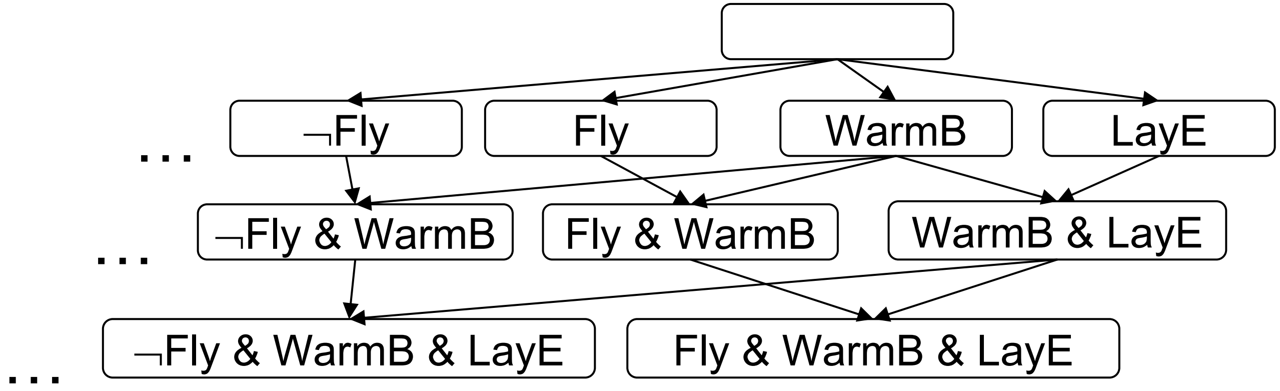
Inductive Learning Method

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

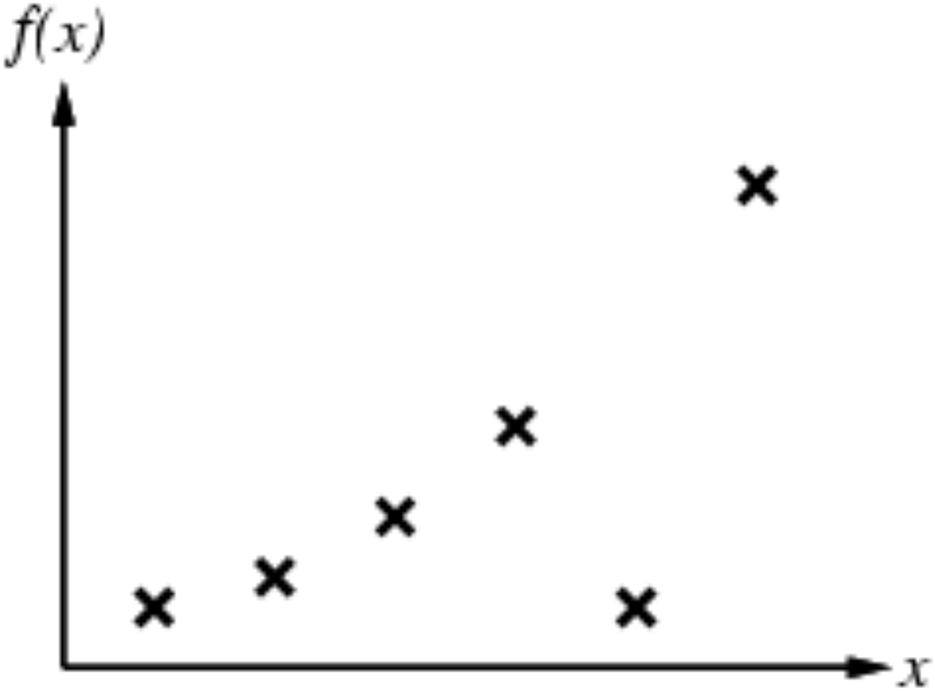
- *h* is *consistent* if it agrees with *f* on all examples
- *f* is *realizable* in **H** if there is some *h*∈**H** that exactly represents *f*
- Although, often must be satisfied with best approximation

Generally search through H until find a “good” *h*

- If **H** is defined via a concept description language there is usually an implicit generalization hierarchy
    - Can search this hierarchy from specific to general, or vice versa
- Or there may be a measure of simplicity on **H** so that can search from simple to complex
    - Using *Ockham’s razor* to choose simplest consistent, or good, *h*

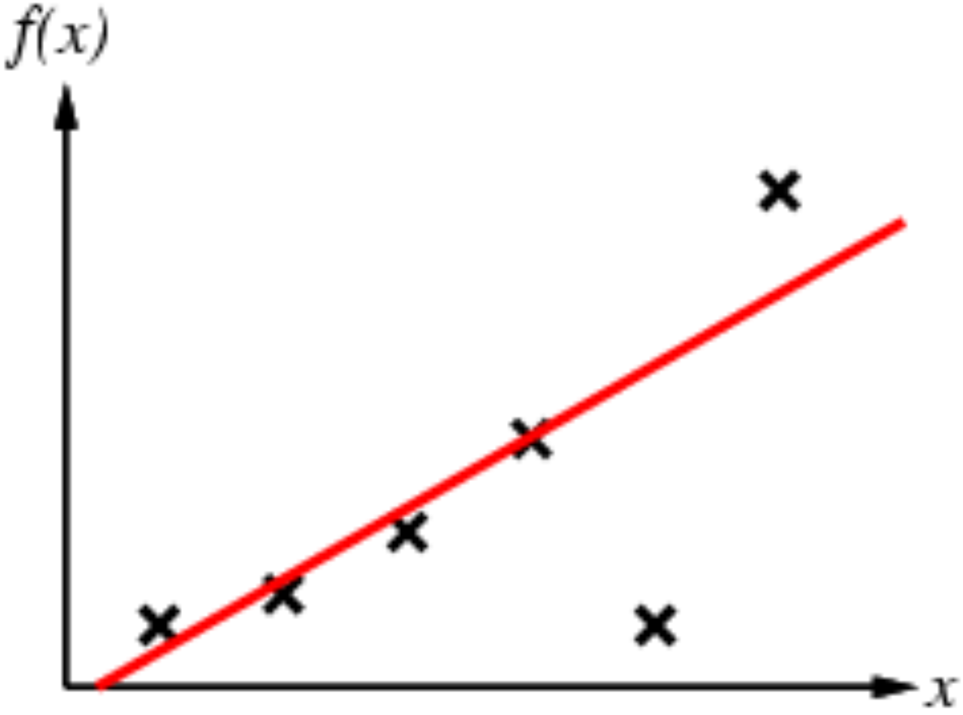


Curve Fitting From Simple to Complex



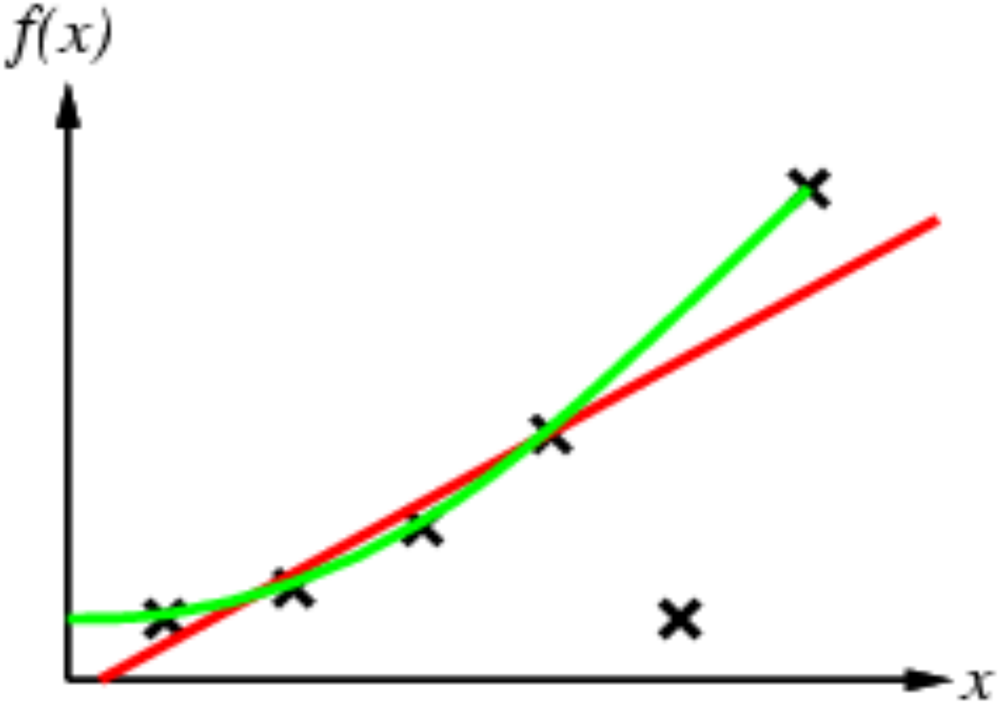
Curve Fitting From Simple to Complex

Linear



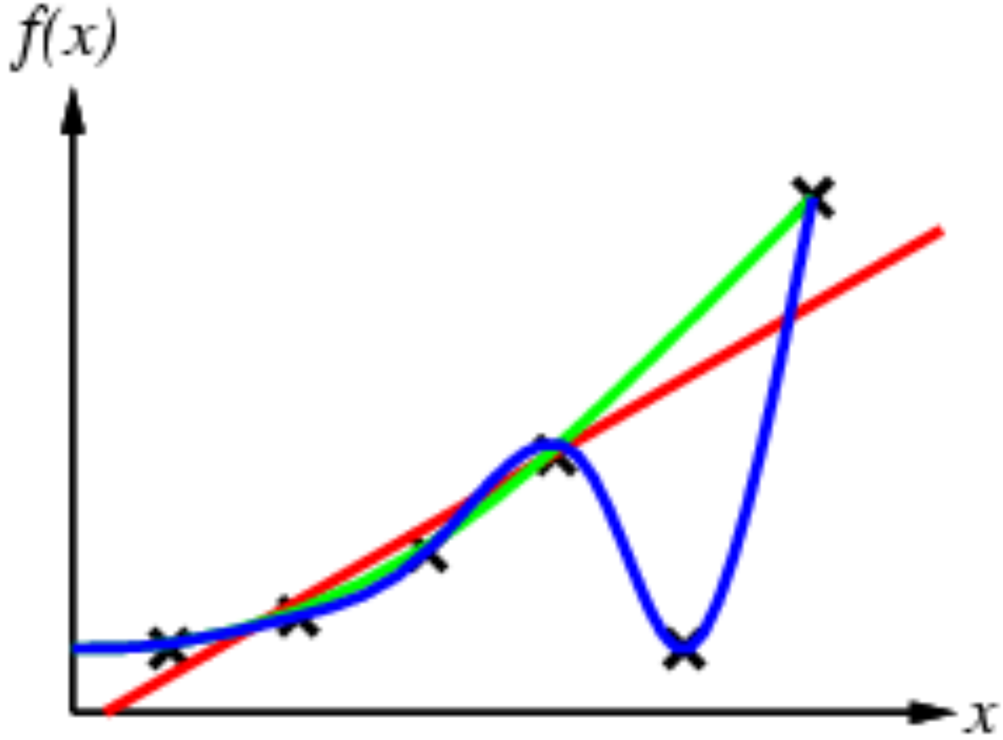
Curve Fitting From Simple to Complex

Quadratic

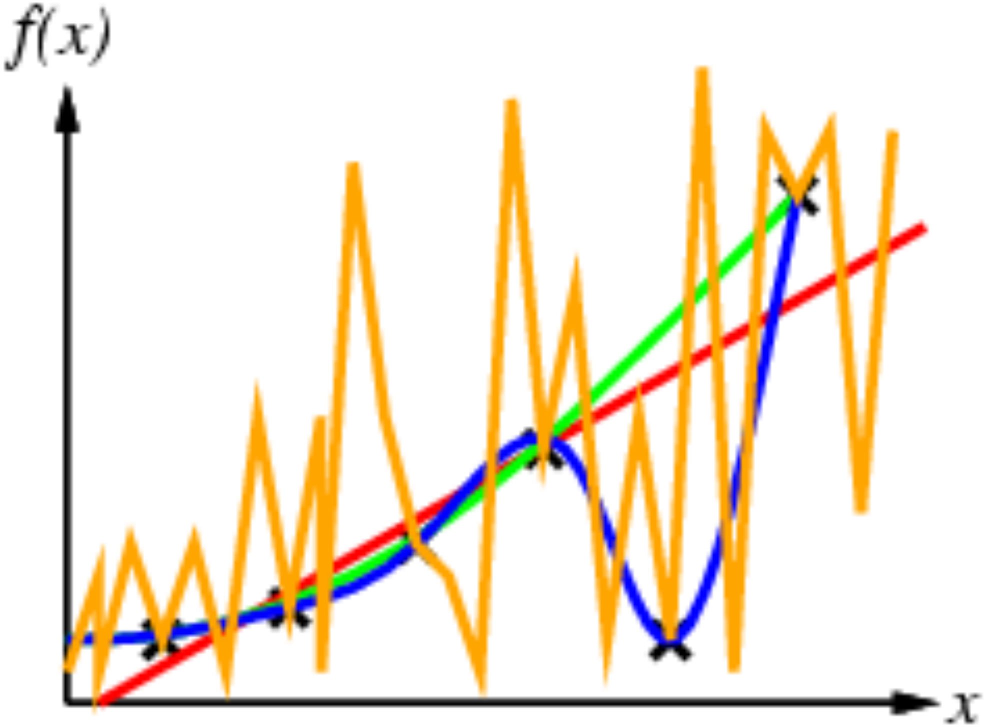


If willing to treat remaining point as noise, could stop here

CURVE FITTING FROM SIMPLE TO COMPLEX



Curve Fitting From Simple to Complex



By Ockham's razor, should prefer blue to orange

What makes for a good ML problem?

Labeled examples

Customer churn: records of current customers (loyal) + former customers who left (churn)

Fraud detection: examples of fraud and not fraud

Relevant features

Customer information: age, sex, zip code, historic spending patterns, etc.

Transaction information: amount, previous transactions, customer information, etc.

Uncertainty/error tolerance

Identifying some customers who are leaving is better than current system

Automated fraud identification still verified by human user

Example: predicting housing prices

Square Footage	ZipCode	Price
2000	48075	200,000
3000	48075	300,000
4000	48075	400,000
5000	48075	500,000
2000	98052	400,000
3000	98052	800,000
4000	98052	1,500,000
5000	98052	3,000,000

Example Induction Problem

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

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

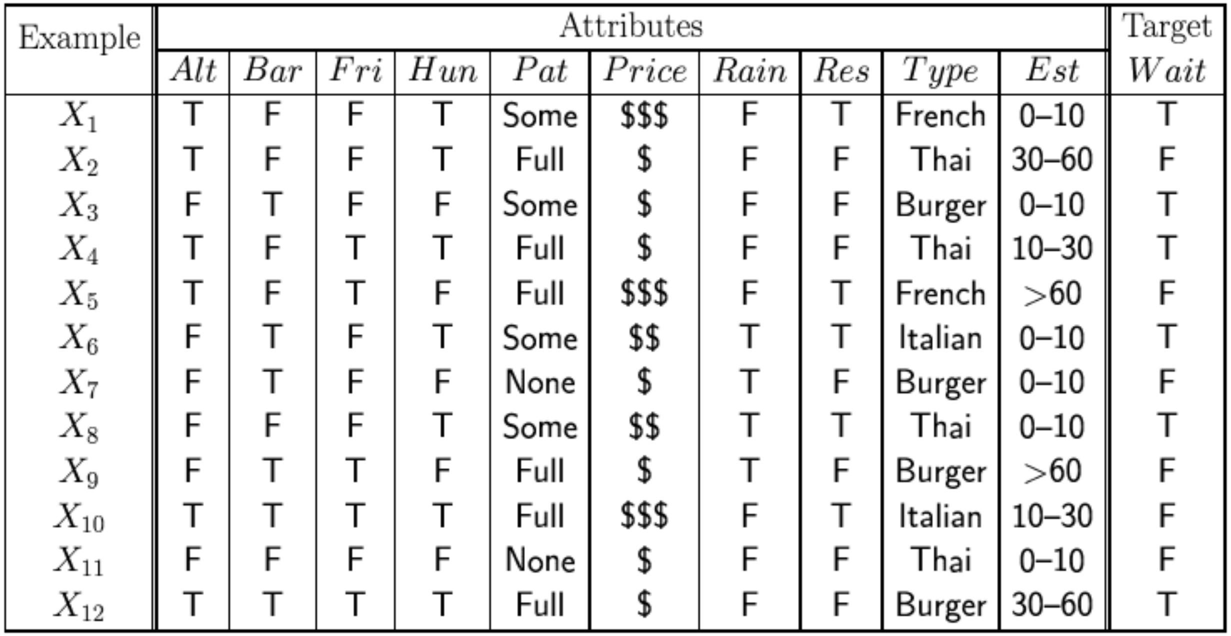
Attribute-Based Representations

Examples described by *attributes* and *values*

- Values may be Boolean, discrete, or continuous

Example situations where will/won't wait for a table:

- *Classification* of examples as *positive* (T) or *negative* (F)



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

Want a general way of deciding when to wait or not

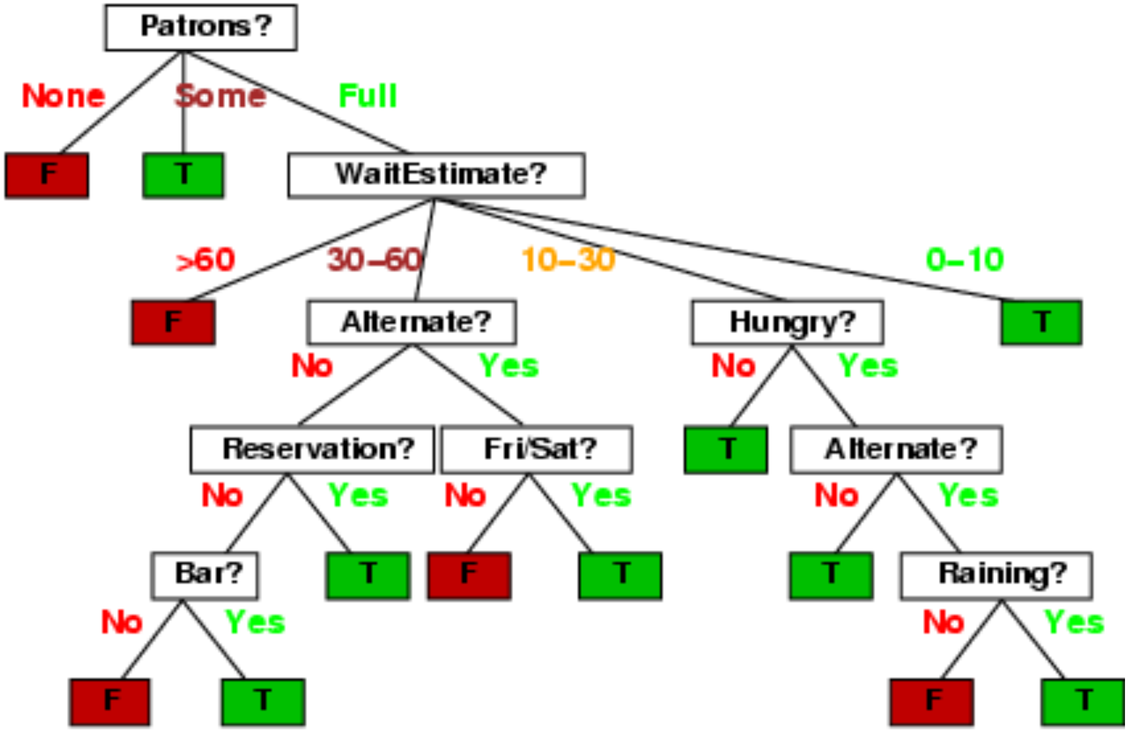
- E.g., generalize from specific examples to a general rule (or tree)

Decision Trees

One possible representation for hypotheses

- Each non-leaf node splits on an attribute, w/ a branch per value
- Leaf nodes specify the *class* for any instance reaching them
- *Can view each path from root to terminal as a rule*

Here is a hand-engineered tree for deciding on waiting:



Note: This is a performance model rather than a learning model, but it can make a good target for inductive learning

Hypothesis Spaces

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

= number of Boolean functions over *n* symbols

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

E.g., 6 Boolean attributes implies 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses?

E.g., $Hungry \wedge \neg Rain$?

Each attribute can be positive, negative, or omitted

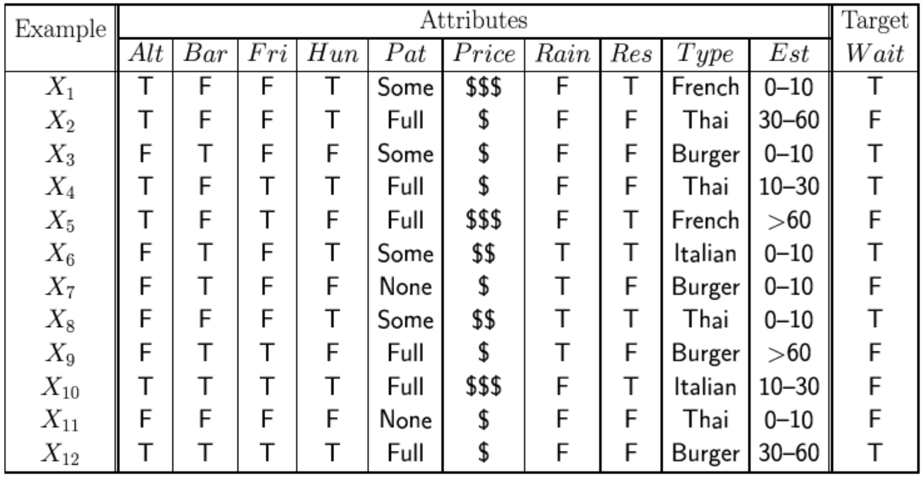
⇒ 3^n distinct conjunctive hypotheses

A more expressive hypothesis space implies

- Increased chance that target function can be expressed
- Increased number of hypotheses consistent with training set

⇒ May make learning slower and yield worse predictions

Sample Trace



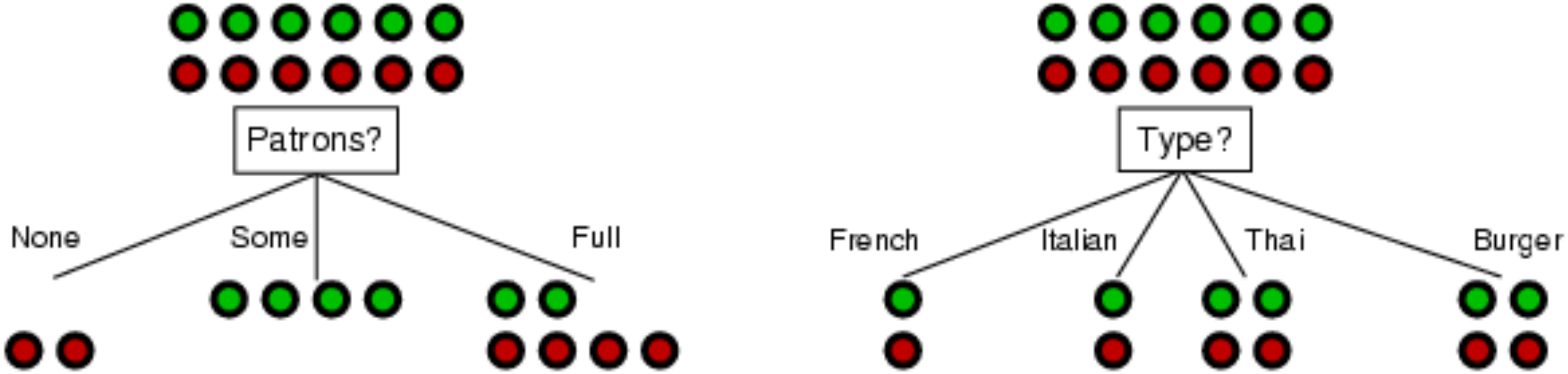
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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

Choosing an Attribute

Idea: An ideal attribute splits the examples into subsets that are "all positive" or "all negative"

If can’t get perfect split, the closer you are to a perfect split the closer you are to a consistent hypothesis

- *Greedy strategy*: as pure a split as possible at each point



***Patrons* is a better choice**

Using Information Theory for Choose-Attribute

Consider *information* required for a decision

- One *bit* is enough to answer a yes/no question if there is no other information about it available
  - E.g., determining heads vs. tails for a fair coin

***Approach*: Choose attribute so as to minimize the remaining information that will be required to answer the question once have split on attribute**

Result is *Information Gain* for splitting on that attribute

Information in an Answer

The amount of *information* contained in an answer given prior probabilities of the possible values is:

$$I(P(v_1), \ldots , P(v_n)) = \Sigma_n -P(v_i) \log_2 P(v_i)$$

For a training set with *p* positive and *n* negative examples:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2 \frac{p}{p+n} - \frac{n}{p+n}\log_2 \frac{n}{p+n}$$

- For a fair coin: $(-1/2\log_2 1/2 - 1/2\log_2 1/2) = 1$ bit

Information Gain

- A chosen attribute *A* divides the training set *E* into *v* subsets E_1, … , E_v according to the values of these examples for *A* (where *A* has v distinct values) On average, the amount of remaining information to answer after splitting on A is:

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- So, the *Information Gain* (IG) from testing attribute A is the total information required before split on A, minus the information required after split on A:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- ***Choose the attribute with the largest IG***
  - *Minimizes total expected information required for answer*

Information Gain Example

For Restaurant training set, $p = n$ = *6*

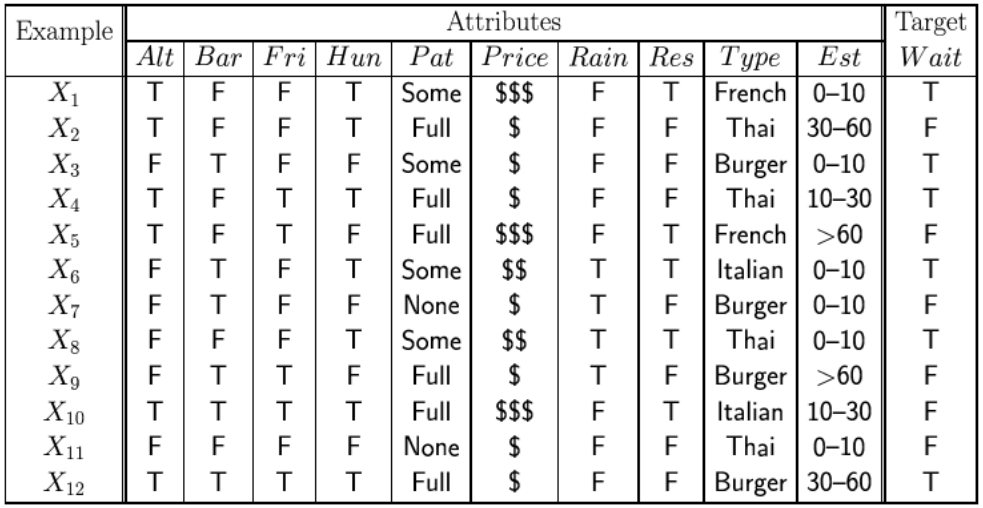
So, I(*6/12, 6/12*) = *1* bit

Consider attributes *Patrons* and *Type* (and others too):

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6},\frac{4}{6})] = 0.541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{4}{12} I(\frac{2}{4},\frac{2}{4}) + \frac{4}{12} I(\frac{2}{4},\frac{2}{4})] = 0 \text{ bits}$$

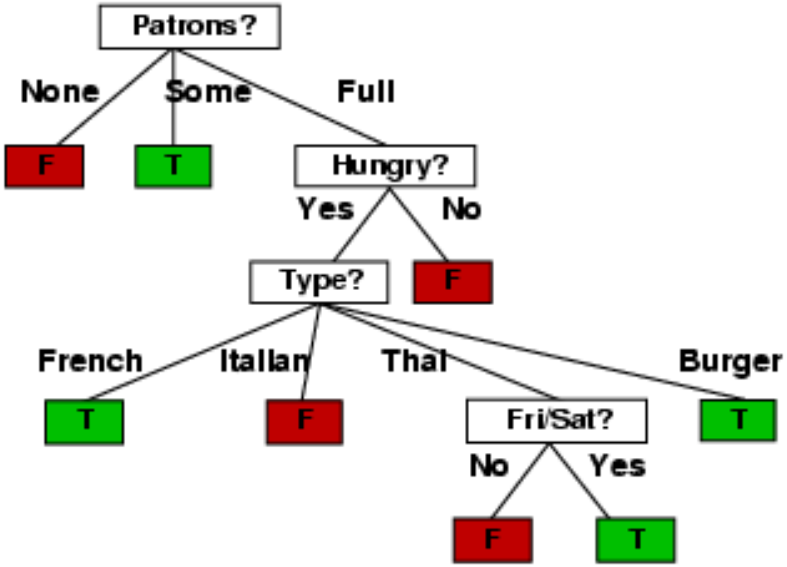
Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root of the tree

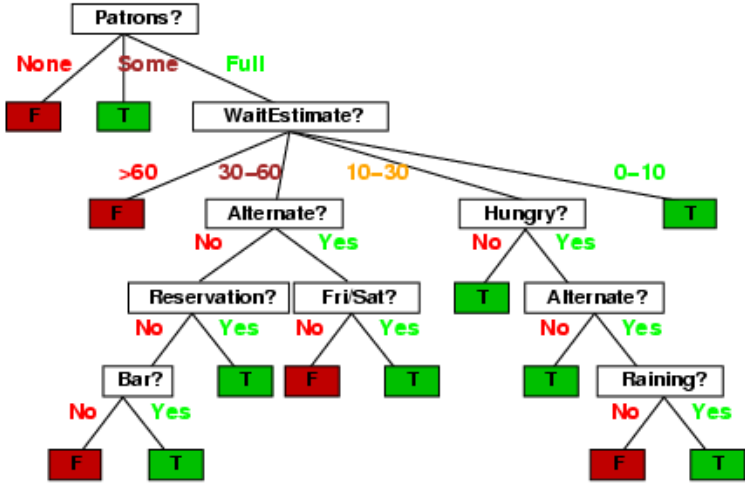


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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

Example (Cont.)

Decision tree learned from the 12 examples:

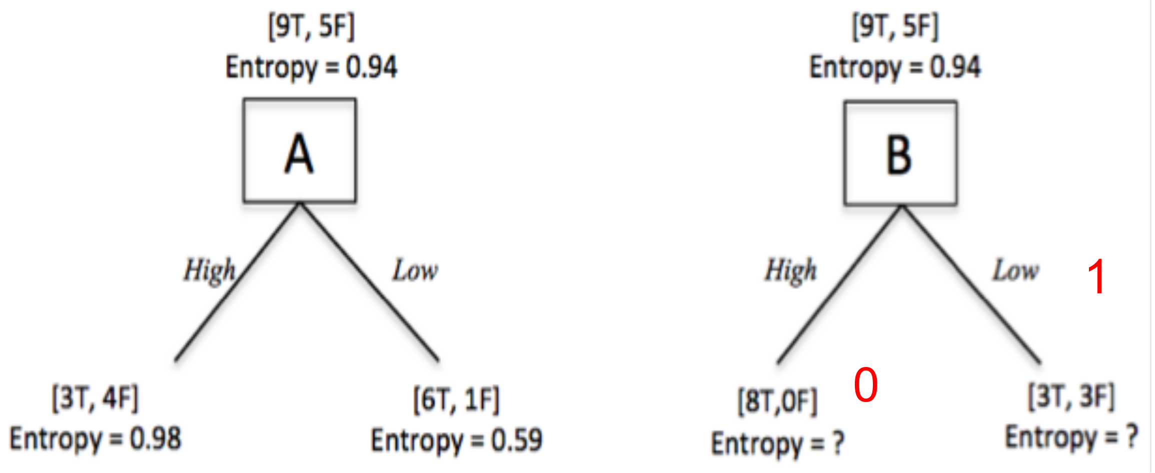




Substantially simpler than tree created by hand

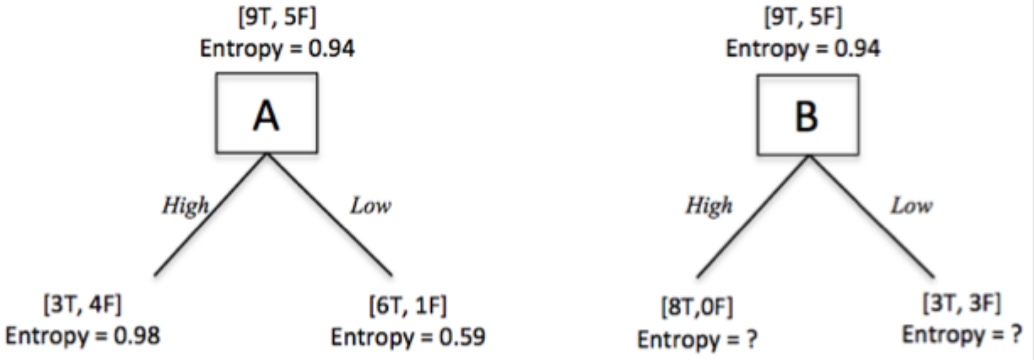
- More complex hypothesis isn’t justified by small amount of data available

Decision Trees



$$I(\frac{p}{p+n},\frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Decision Tree



2. Calculate the information gain (or change in entropy), in bits, for each attribute.

$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A) \qquad remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

IG(A) = .94 – (7/14 I(3/7,4/7) + 7/14 I(6/7,1/7)) = .94 -.5(.98+.59)

IG(B) = .94 – (8/14 I(8/8,0/8) + 6/14 I(3/6,3/6)) = .94 – 3/7

3. Which one of the two attributes A and B is a better classifier according to the decision tree learning (DTL) algorithm?

B

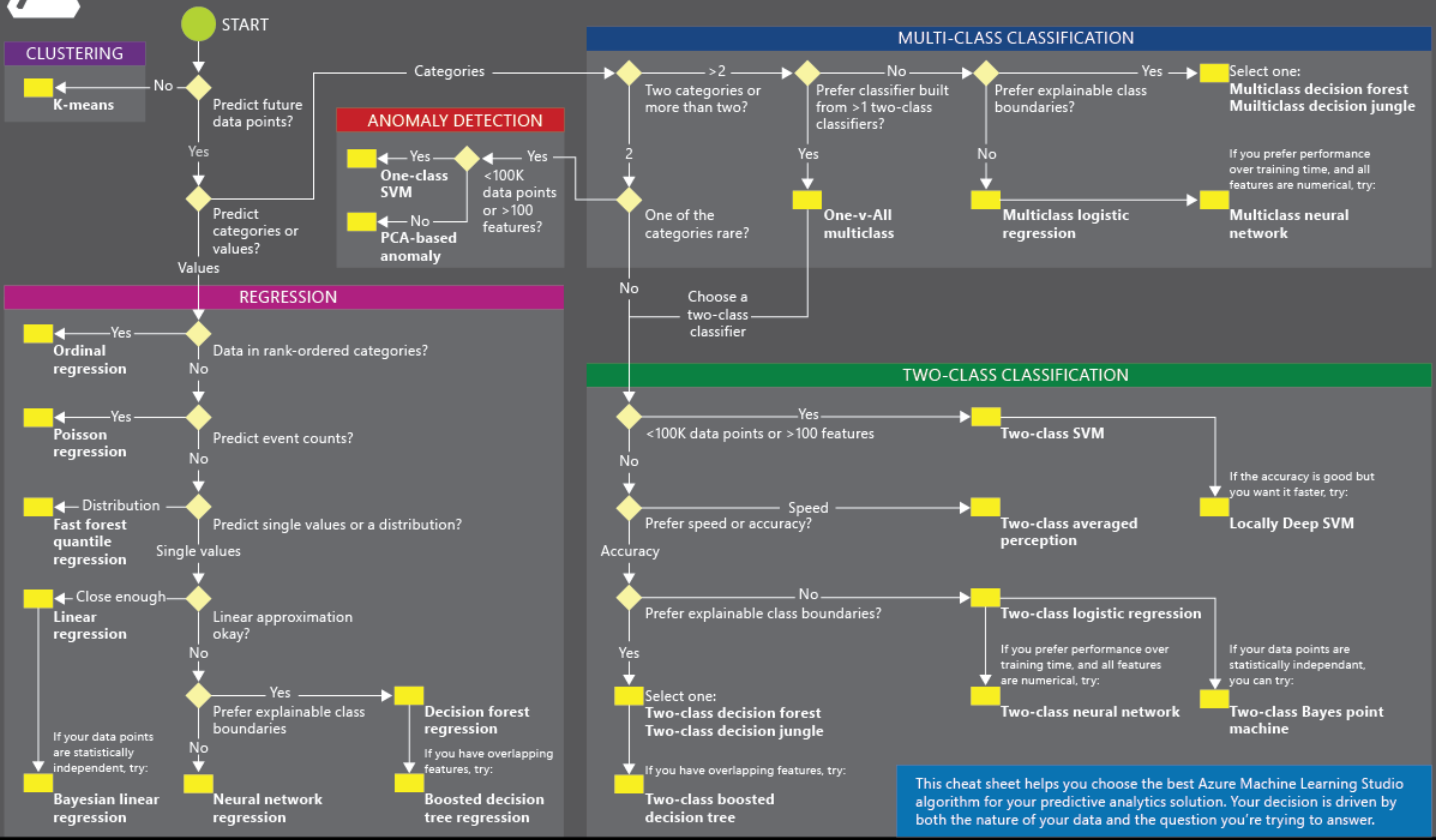
What you should know?

- What is learning? Why is learning important? What are some of the performance measures for learning?
- What is generalization? What is bias? What is Occam's Razor? Why is they important for learning?
- What is the difference between supervised, semi-supervised, and unsupervised learning?
- What is overfitting? When does this happen? Why do we want to avoid this?
- Cross-validation? How is this addressing over overfitting?

Want more?

Try exercise 18.5-7 in AIMA

Microsoft Azure Machine Learning: Algorithm Cheat Sheet



START

Predict future data points?

No

Yes

CLUSTERING

K-means

Predict categories or values?

Categories

Values

ANOMALY DETECTION

<100K data points or >100 features?

Yes

No

One-class SVM

PCA-based anomaly

MULTI-CLASS CLASSIFICATION

Two categories or more than two?

>2

2

Prefer classifier built from >1 two-class classifiers?

Yes

No

One-v-All multiclass

Prefer explainable class boundaries?

Yes

No

Select one:
Multiclass decision forest
Muilticlass decision jungle

Multiclass logistic regression

If you prefer performance over training time, and all features are numerical, try:

Multiclass neural network

One of the categories rare?

Yes

No

Choose a two-class classifier

REGRESSION

Data in rank-ordered categories?

Yes

No

Ordinal regression

Predict event counts?

Yes

No

Poisson regression

Predict single values or a distribution?

Distribution

Single values

Fast forest quantile regression

Linear approximation okay?

Close enough

No

Linear regression

If your data points are statistically independent, try:

Bayesian linear regression

Prefer explainable class boundaries

Yes

No

Decision forest regression

If you have overlapping features, try:

Boosted decision tree regression

Neural network regression

TWO-CLASS CLASSIFICATION

<100K data points or >100 features

Yes

No

Two-class SVM

If the accuracy is good but you want it faster, try:

Locally Deep SVM

Prefer speed or accuracy?

Speed

Accuracy

Two-class averaged perception

Prefer explainable class boundaries?

No

Yes

Two-class logistic regression

If you prefer performance over training time, and all features are numerical, try:

Two-class neural network

If your data points are statistically independant, you can try:

Two-class Bayes point machine

Select one:
Two-class decision forest
Two-class decision jungle

If you have overlapping features, try:

Two-class boosted decision tree

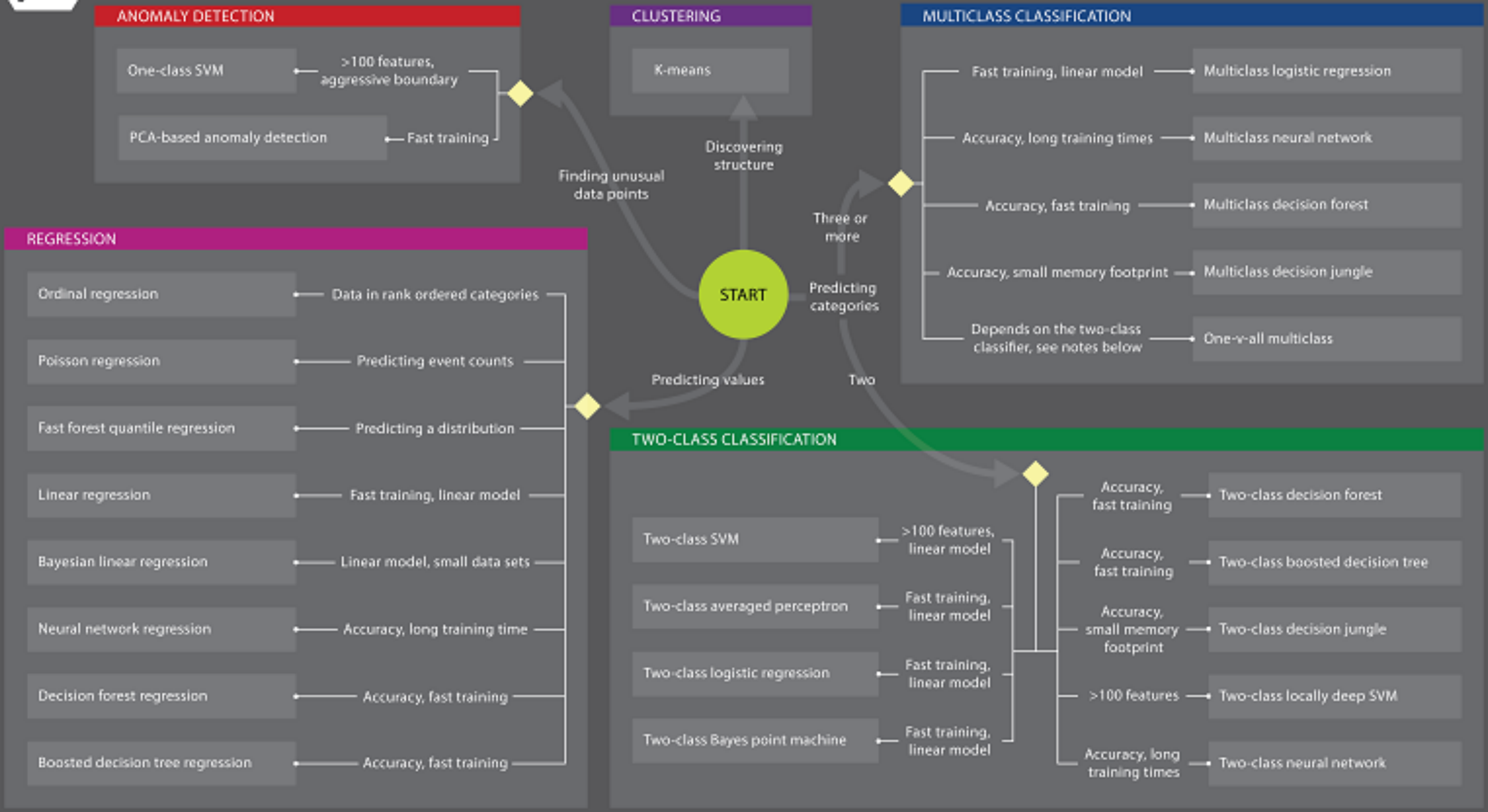
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.

Created by the Azure Machine Learning Team   Email: AzurePoster@microsoft.com   Download this poster: http://aka.ms/MLCheatSheet

Microsoft

Microsoft Azure Machine Learning: Algorithm Cheat Sheet

This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.



START

Finding unusual data points → ANOMALY DETECTION

Discovering structure → CLUSTERING

Predicting categories: Three or more → MULTICLASS CLASSIFICATION; Two → TWO-CLASS CLASSIFICATION

Predicting values → REGRESSION

ANOMALY DETECTION

- >100 features, aggressive boundary → One-class SVM
- Fast training → PCA-based anomaly detection

CLUSTERING

- Discovering structure → K-means

MULTICLASS CLASSIFICATION

- Fast training, linear model → Multiclass logistic regression
- Accuracy, long training times → Multiclass neural network
- Accuracy, fast training → Multiclass decision forest
- Accuracy, small memory footprint → Multiclass decision jungle
- Depends on the two-class classifier, see notes below → One-v-all multiclass

REGRESSION

- Data in rank ordered categories → Ordinal regression
- Predicting event counts → Poisson regression
- Predicting a distribution → Fast forest quantile regression
- Fast training, linear model → Linear regression
- Linear model, small data sets → Bayesian linear regression
- Accuracy, long training time → Neural network regression
- Accuracy, fast training → Decision forest regression
- Accuracy, fast training → Boosted decision tree regression

TWO-CLASS CLASSIFICATION

- >100 features, linear model → Two-class SVM
- Fast training, linear model → Two-class averaged perceptron
- Fast training, linear model → Two-class logistic regression
- Fast training, linear model → Two-class Bayes point machine
- Accuracy, fast training → Two-class decision forest
- Accuracy, fast training → Two-class boosted decision tree
- Accuracy, small memory footprint → Two-class decision jungle
- >100 features → Two-class locally deep SVM
- Accuracy, long training times → Two-class neural network

Created by the Azure Machine Learning Team   Email: AzurePoster@microsoft.com   Download this poster: http://aka.ms/MLCheatSheet

Microsoft