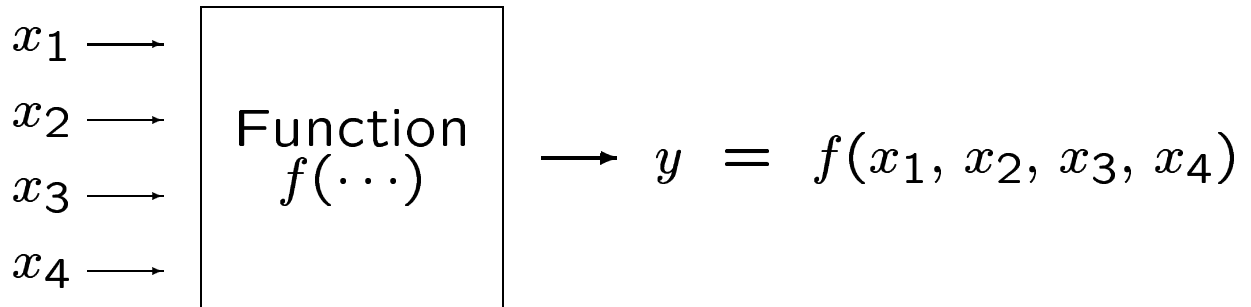


Classification Task

Real World



- **Credit risk assessment**

x : Properties of customer and proposed purchase.

$f(x)$: Approve purchase or not.

- **Disease diagnosis**

x : Properties of patient (symptoms, lab tests)

$f(x)$: Disease (or maybe, recommended therapy)

- **Face recognition**

x : Bitmap picture of person's face

$f(x)$: Name of person

- **Spam Detection**

x : Email message $f(x)$: Spam or not spam

Difficult to get Classifier

- $\neg\exists$ **human expert**

x: Bond graph for a new molecule

f(x): Predicted binding strength to AIDS protease molecule.

- **humans (can perform task) but can't describe how they do it.**

x: Bitmap picture of hand-written character

f(x): Ascii code of the character

- **... desired function changes frequently**

x: Description of stock prices and trades for last 10 days

f(x): Recommended stock transactions

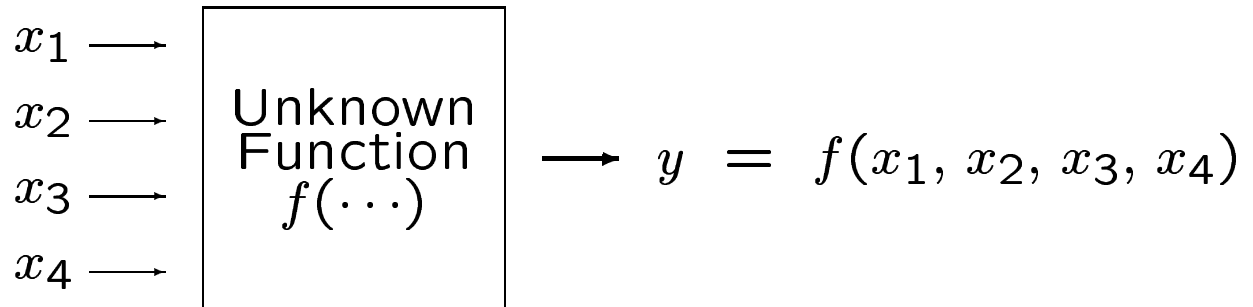
- **Each user u needs customized function f_u**

x: Incoming email message.

f(x): Importance score for presenting to user
(or deleting without presenting)

Answer: Learn Classifier!

- Real World

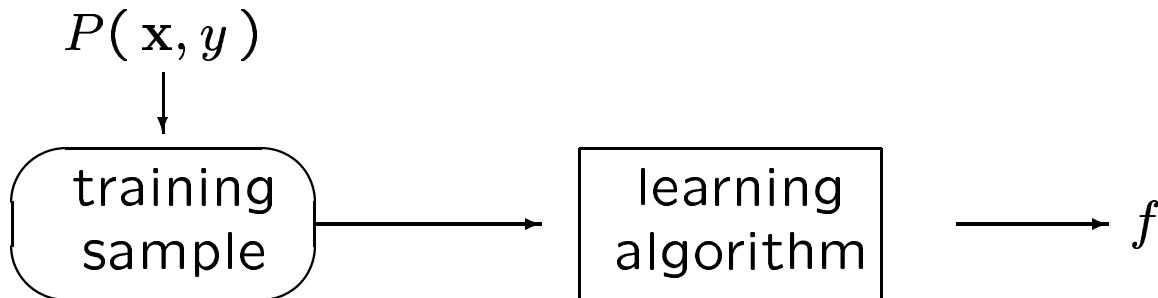


- Observations of World

<i>Example</i>	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

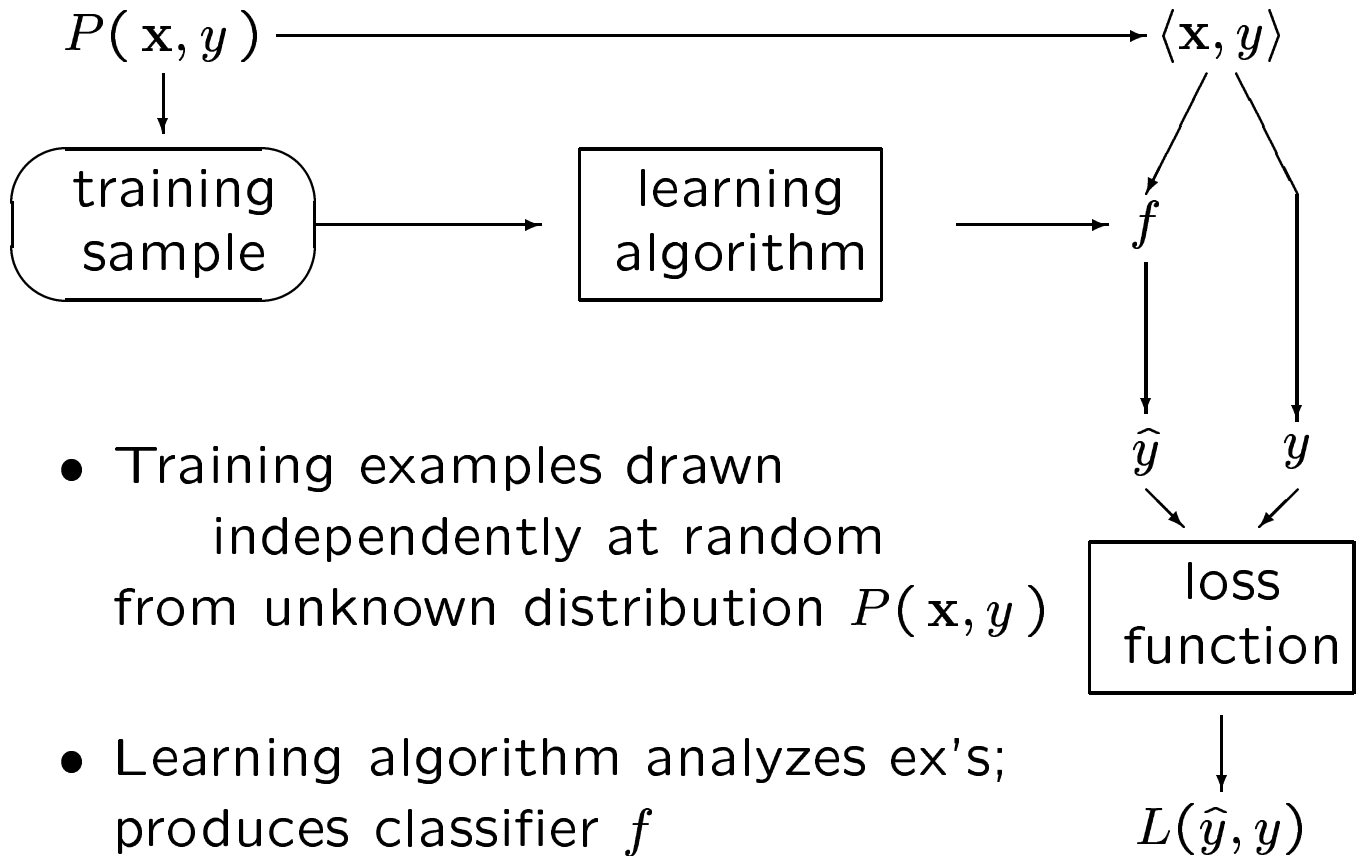
- Learn approximation $\hat{f}(x_1, x_2, x_3, x_4)$
- Use it to compute $\hat{f}(1, 1, 1, 0)$?

Formal Setting



- Training examples drawn independently at random from unknown distribution $P(\mathbf{x}, y)$
- Learning algorithm analyzes ex's; produces classifier f

Formal Setting



- Training examples drawn independently at random from unknown distribution $P(\mathbf{x}, y)$
- Learning algorithm analyzes ex's; produces classifier f
- For new data point $\langle \mathbf{x}, y \rangle$ drawn from P , classifier is given \mathbf{x} and predicts $\hat{y} = f(\mathbf{x})$
... with loss, $L(\hat{y}, y)$
- Goal of learning algorithm:
Find f that minimizes expected loss.

Formal Version of Spam Detection

- $P(\mathbf{x}, y)$: distribution of email messages \mathbf{x} and their true labels y (spam or not spam)
- **training sample**: set of email messages that have been labeled by the user
- **learning algorithm**: ... this course ...
- f : classifier produced by learning alg
- **test point**: A new email message \mathbf{x} (with a true, but hidden, label $y \in \{\text{spam}, \text{not_spam}\}$)
- loss function $L(\hat{y}, y)$:

predicted label \hat{y}	true label y	
	spam	not spam
spam	0	10
not spam	1	0

Main Approaches to Machine Learning

1. **Learn classifier directly:** $f : \mathbf{X} \mapsto Y$,
where Y is discrete (e.g., $Y = \{+1, -1\}$)

2. **Learn regression function:** $r : \mathbf{X} \mapsto \mathcal{R}$
 c_1 if $r(\mathbf{x}) > \lambda$ c_2 if $r(\mathbf{x}) \leq \lambda$

3. **Learn conditional distribution:** $P(y | \mathbf{x})$

4. **Learn joint distribution:** $P(y, \mathbf{x})$

- **Linear Models:**

- (a) **Learn classifier:** Perceptron Algorithm

- (b) **Learn regression function:** LMS

- (c) **Learn conditional distribution:** Logistic Regression

- (d) **Learn joint distribution:** Linear discriminant analysis

Infering Classifier f from $P(y | \mathbf{x})$

- Give \mathbf{x} , predict class y that minimizes the expected loss:

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{argmin}_{\hat{y}} E_{y|\mathbf{x}} [L(\hat{y}, y)] \\ &= \operatorname{argmin}_{\hat{y}} \sum_y P(y | \mathbf{x}) L(\hat{y}, y) \end{aligned}$$

Eg: For specific email message \mathbf{x}

$$P(y = \text{spam} | \mathbf{x}) = 0.6$$

$$P(y = \text{not_spam} | \mathbf{x}) = 0.4$$

(Note $P(y = \text{spam} | \mathbf{x}) > P(y = \text{not_spam} | \mathbf{x})$)

What is optimal prediction \hat{y} ?

– Expected loss of $\hat{y} = \text{spam}$:

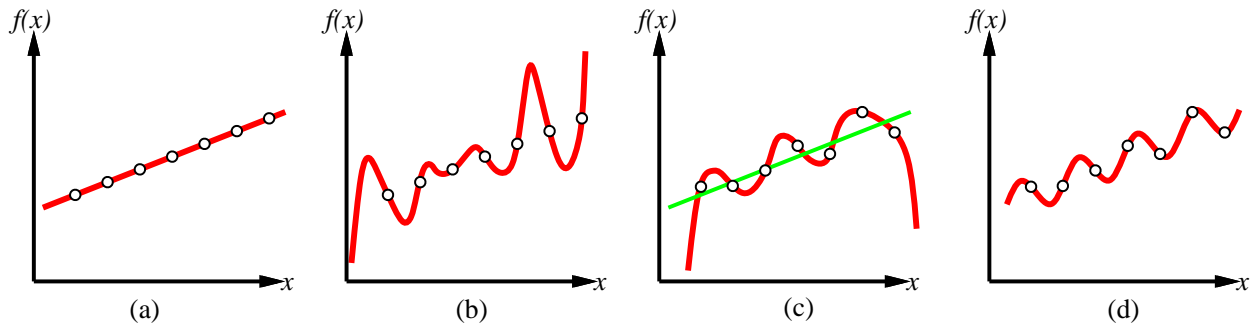
$$0 \times 0.6 + 10 \times 0.4 = 4$$

– Expected loss of $\hat{y} = \text{not_spam}$:

$$1 \times 0.6 + 0 \times 0.4 = 0.6$$

\Rightarrow optimal prediction is “not_spam”

Goal: Generalization



- Which hypothesis is best?
- Depends. . .
 - Is data noisy?
 - What is known about $f(\cdot)$?
 - . . . piece-wise linear, smooth, . . .

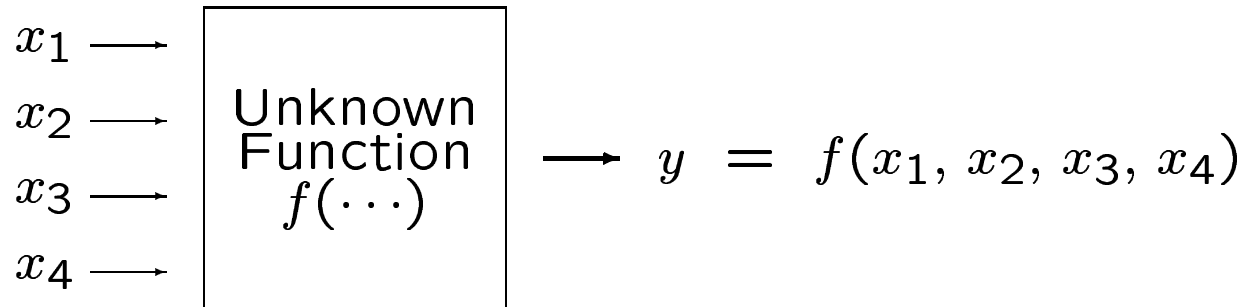
Goal: Good performance over *distribution*
including UNSEEN data

\neq simply fitting observed data
(Otherwise: just memorize observations!)

\Rightarrow Need “bias” . . .

A Supervised Learning Problem

- Real World



- Observations of World

<i>Example</i>	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

- Learn approximation $\hat{f}(x_1, x_2, x_3, x_4)$
- Use it to compute $\hat{f}(1, 1, 1, 0)$?

Hypothesis Space

x_1	x_2	x_3	x_4	y
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

- **Complete Ignorance:** $\exists 2^{2^4} = 65536$ boolean functions over 4 inputs.
- Can NOT determine which is correct until see *every* possible i/o pair

Hypothesis Space, con't

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

- After 7 examples, still 2^9 possibilities H' !

- What is $f(1, 1, 1, 1)$?

$$\exists 128 f \in H' \text{ s.t. } f(1, 1, 1, 1) = 1$$

$$\exists 128 f' \in H' \text{ s.t. } f'(1, 1, 1, 1) = 0$$

(Ie, $\forall f \in H' \text{ s.t. } f(1, 1, 1, 1) = 1$

$\exists f' \in H' \text{ s.t. } f'(1, 1, 1, 1) = 0$)

Solution: Restrict Hypothesis Space

By applying prior knowledge or by guessing, choose a space of hypotheses H that is *smaller* than space of all possible functions:

- simple conjunctive rules

- m-of-n rules

- linear functions

- multivariate Gaussian joint probability distributions

- ...

Hypothesis Spaces (2)

- **Simple Rules:** \exists 16 simple conjunctive rules.
but none fit the 7 examples:

<i>Rule</i>	<i>Counterexample id</i>
$\Rightarrow y$	1
$x_1 \Rightarrow y$	3
$x_2 \Rightarrow y$	2
$x_3 \Rightarrow y$	1
$x_4 \Rightarrow y$	7
$x_1 \wedge x_2 \Rightarrow y$	3
$x_1 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \Rightarrow y$	3
$x_2 \wedge x_4 \Rightarrow y$	3
$x_3 \wedge x_4 \Rightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3

NOTE: No simple rule explains data!

Hypothesis Space (3)

- **m-of-n rules:** \exists 32 simple m-of-n rules
(includes simple conjunctions, clauses)

<i>Variables</i>	<i>Counterexample id</i>			
	1-of	2-of	3-of	4-of
$\{x_1\}$	3	—	—	—
$\{x_2\}$	2	—	—	—
$\{x_3\}$	1	—	—	—
$\{x_4\}$	7	—	—	—
$\{x_1, x_2\}$	3	3	—	—
$\{x_1, x_3\}$	4	3	—	—
$\{x_1, x_4\}$	6	3	—	—
$\{x_2, x_3\}$	2	3	—	—
$\{x_2, x_4\}$	2	3	—	—
$\{x_3, x_4\}$	4	4	—	—
$\{x_1, x_2, x_3\}$	1	3	3	—
$\{x_1, x_2, x_4\}$	2	3	3	—
$\{x_1, x_3, x_4\}$	1	***	3	—
$\{x_2, x_3, x_4\}$	1	5	3	—
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

NOTE: 31/32 inconsistent w/ ≥ 1 example
1 never matches!

Hypothesis Space (4)

- Consider LIST of simple Rules?
evaluated sequentially . . .

- Set of 1 rule?
No

- Set of 2 rules?
No

- Set of 3 rules?
Yes:

$$\begin{array}{lcl} x_2 \wedge x_4 & \Rightarrow & \neg y \\ x_4 & \Rightarrow & y \\ & \Rightarrow & \neg y \end{array}$$

“Bias” of Learning Algorithms

- Learning algorithm embodies some “bias” to prefer one hypothesis over another
... ideally: matched to assumptions/environment
- Two types of bias:
 - **restriction bias or language bias**
Specifies what hypothesis space is searched
(*Eg, Gaussian, Mixture of Gaussians, Decision Trees, Piece-wise linear functions, ...*)
 - **preference bias or search bias**
specifies how hypothesis space is explored
⇒ leads to different (first) answer
- Tradeoff: Suppose $A \subset B$
 - + : B more likely to include *correct* hypothesis
 - : B more likely to include *INCORRECT* hypothesis

(Size of H , VC Dimension of H)

Two Views of Learning

- **Learning \equiv removing remaining uncertainty**

If know target function is m -of- n function

\Rightarrow could use training ex's to identify which function

- **Learning \equiv “growing” good hypothesis**

Start w/very small class C_0 , see if ok hyp $\in C_0$

If not, enlarge to $C_1 \supset C_0$; see if ok hyp $\in C_1$

... until find hyp that fits the data.

- Either way:

**Need alg that finds hypothesis,
from given class,
that “fits” the data.**

Terminology

Labeled example: Example of form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$

Labeled sample: Set of $\{ \langle \mathbf{x}_i, f(\mathbf{x}_i) \rangle \}$

Classifier: Discrete-valued function.

Possible values $f(x) \in \{1, \dots, K\}$ called “classes”;
“class labels”

Concept: Boolean function.

x s.t. $f(x) = 1$ called “positive examples”

x s.t. $f(x) = 0$ called “negative examples”

Target function (target concept): “True function” f generating the labels

Hypothesis: Proposed function h believed to be similar to f .

Hypothesis Space: Space of all hypotheses that can, in principle, be output by a learning algorithm

Key Issues in Machine Learning

- **What are good hypothesis spaces?**
Which spaces are useful in practical applications; why?
- **What algorithms can work with these spaces?**
∃ general design principles for machine learning alg's?
- **How can we optimize accuracy on future data points?**
Avoiding “overfitting”.
- **How can we have confidence in results?**
How much training data is required to find accurate hypotheses? (statistical question)
- **Are some learning problems computationally intractable?**
(computational question)
- **How can we formulate application problems as machine learning problems?**
(engineering question)

A Framework for Hypothesis Spaces

Size: Is size of hypothesis space
flat or stratified?

Flat spaces: easier to understand

Stratified spaces are generally more useful.

Stratified spaces introduce “overfitting”.

Randomness: Is each hypothesis determin-
istic or stochastic?

Affects hypotheses evaluation:

Deterministic: training example either $\left\{ \begin{array}{l} \text{consistent} \\ \text{inconsistent} \end{array} \right.$

Stochastic: training example either $\left\{ \begin{array}{l} \text{more} \\ \text{less} \end{array} \right\}$ likely

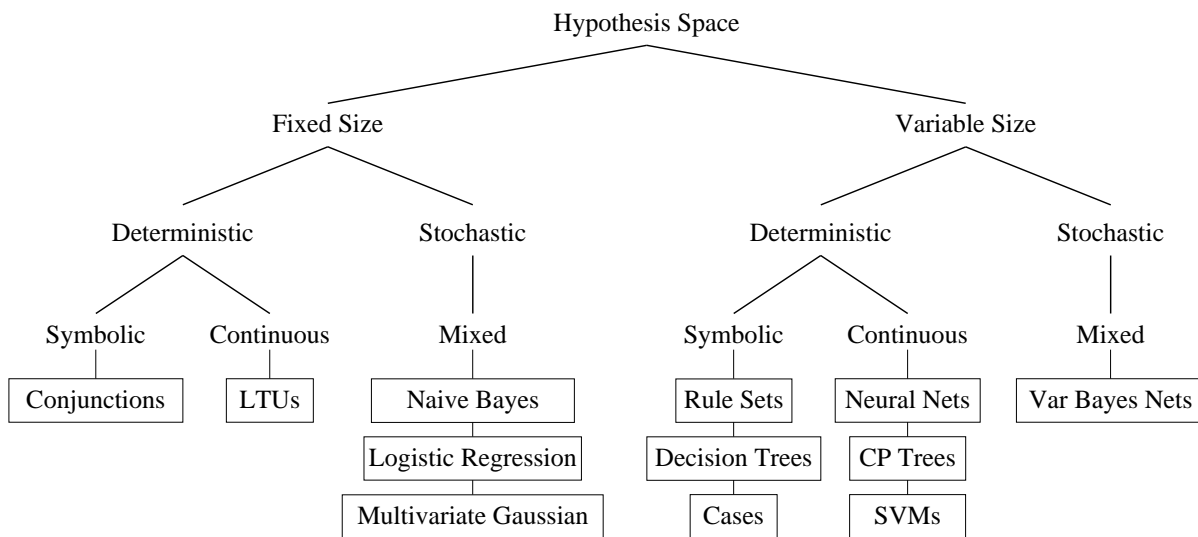
Parameterization: Is each hypothesis de-
scribed by symbolic (discrete) choices, or
by continuous parameters (or both)?

Typically. . . find

discrete parameters by combinatorial search;

continuous parameters by numerical search.

Framework for Hypothesis Spaces (2)



Framework for Learning Algorithms

Search Procedure

- **Direction Computation:** solve for hypothesis directly.
- **Local Search:** start with initial hypothesis, make small improvements until a local optimum.
- **Constructive Search:** start with empty hypothesis, gradually add structure to it until local optimum.

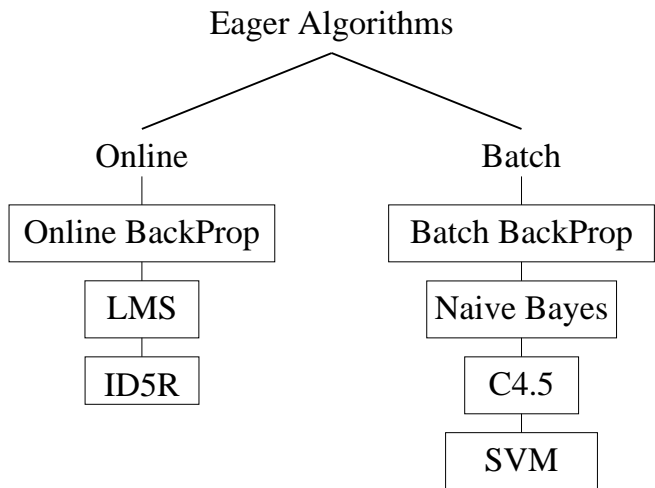
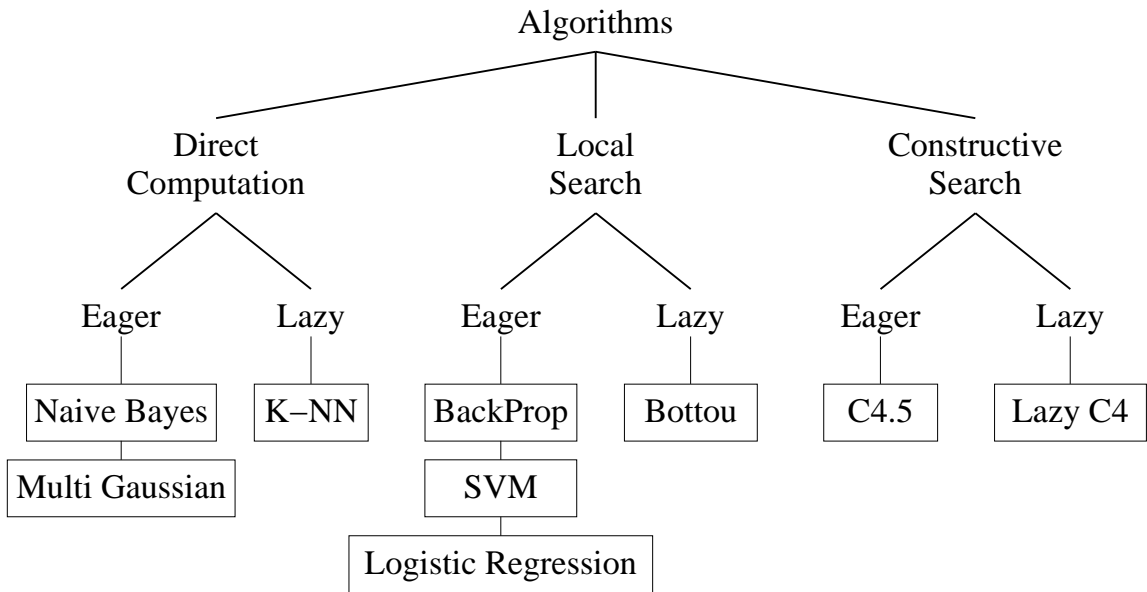
Timing

- **Eager:** Analyze training data and construct an explicit hypothesis.
- **Lazy:** Store training data and wait until a test data point is presented, then construct ad hoc hypothesis to classify that one data point.

Online vs. Batch (for eager algorithms)

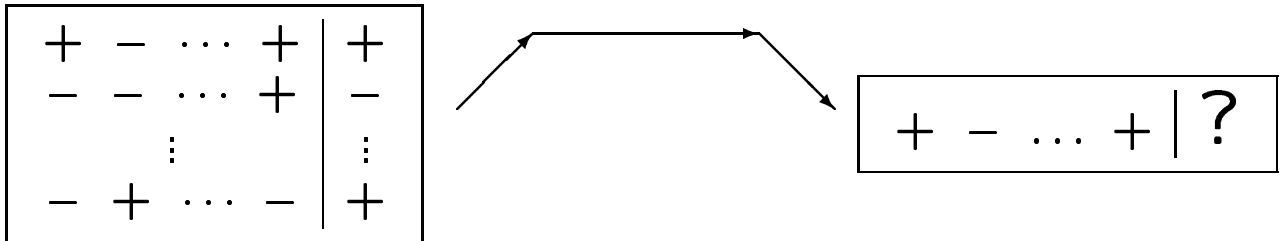
- **Online:** Analyze each training example as it is presented.
- **Batch:** Collect training examples, analyze them, output an hypothesis.

Framework for Learning Alg's (2)



“Train-then-Use” Model

Task: Use labeled training examples $S = \{\langle x_i, c(x_i) \rangle\}_{i=1}^n$ to classify x_{n+1}, \dots



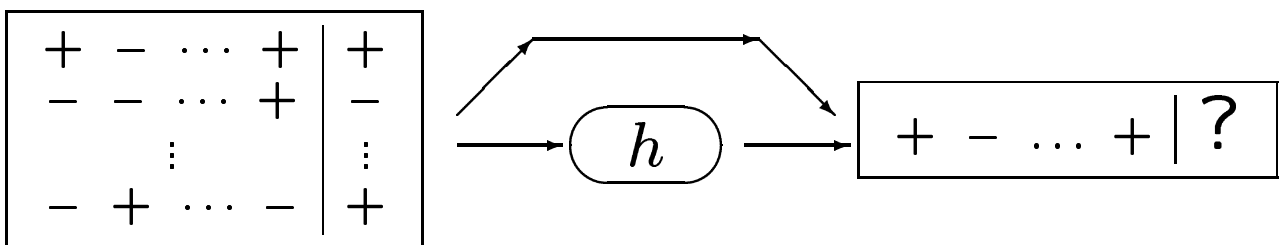
2-step Approach:

Learner: Training Data \mapsto Hypothesis

$$\{\langle x_i, c(x_i) \rangle\}_{i=1..n} \mapsto h(\cdot)$$

Classifier: Instance \mapsto Label

$$h: \mathcal{X} \mapsto \mathcal{C}$$



Source of Examples $\langle x_1, \dots, x_n \rangle$:

- 1: Suggested by learner
- 2: Suggested by helpful teacher
- 3: Randomly produced