Inductive Learning

Anurag Nagar

Recap

- Inductive learning - generalize from a limited set of training data
- Training data is labeled
- You would like to estimate true separating function f
- Attributes of data (i.e. features) are important

Learning from data

- Given: a set of labeled training examples:
  <x, f(x)>
  Global f(x) is unknown to us
  Distribution of x is unknown to us
- Find: An <u>approximation</u> of f(x)

Appropriate situations

- **Credit risk assessment**

  $\mathbf{x}$: Properties of customer and proposed purchase.

  $f(\mathbf{x})$: Approve purchase or not.

- **Disease diagnosis**

  $\mathbf{x}$: Properties of patient (symptoms, lab tests)

  $f(\mathbf{x})$: Disease (or maybe, recommended therapy)

- **Face recognition**

  $\mathbf{x}$: Bitmap picture of person's face

  $f(\mathbf{x})$: Name of the person.

Learning

- Improving with experience (E) at some task (T) with respect to some performance measure (P).
- Experience = Training data
  Task = Any classification task (for this class, at least)
  Performance Measure = Error value
  -> difference between true value and predicted value.

Examples of Learning

- Learning to play checkers:
- T: Play checkers
- P: % of games won
- E: opportunity to play against computer or self

- Most important thing -> How does a learner learn concepts from training (E)

Model Representation

What are you given in supervised learning?

A set of training examples and their labels $(x^{(i)}, y^{(i)})$

** It is assumed $y^{(i)}$ is generated by a true function $f(x)$ **

What do you do with the training data?

Feed it to a learning algorithm that learns a function h, that is an approximation to f

Model Representation

How do you know if h is good?

We measure the error (overall) by using h

Example:

Error = | f(x) – h(x) | or

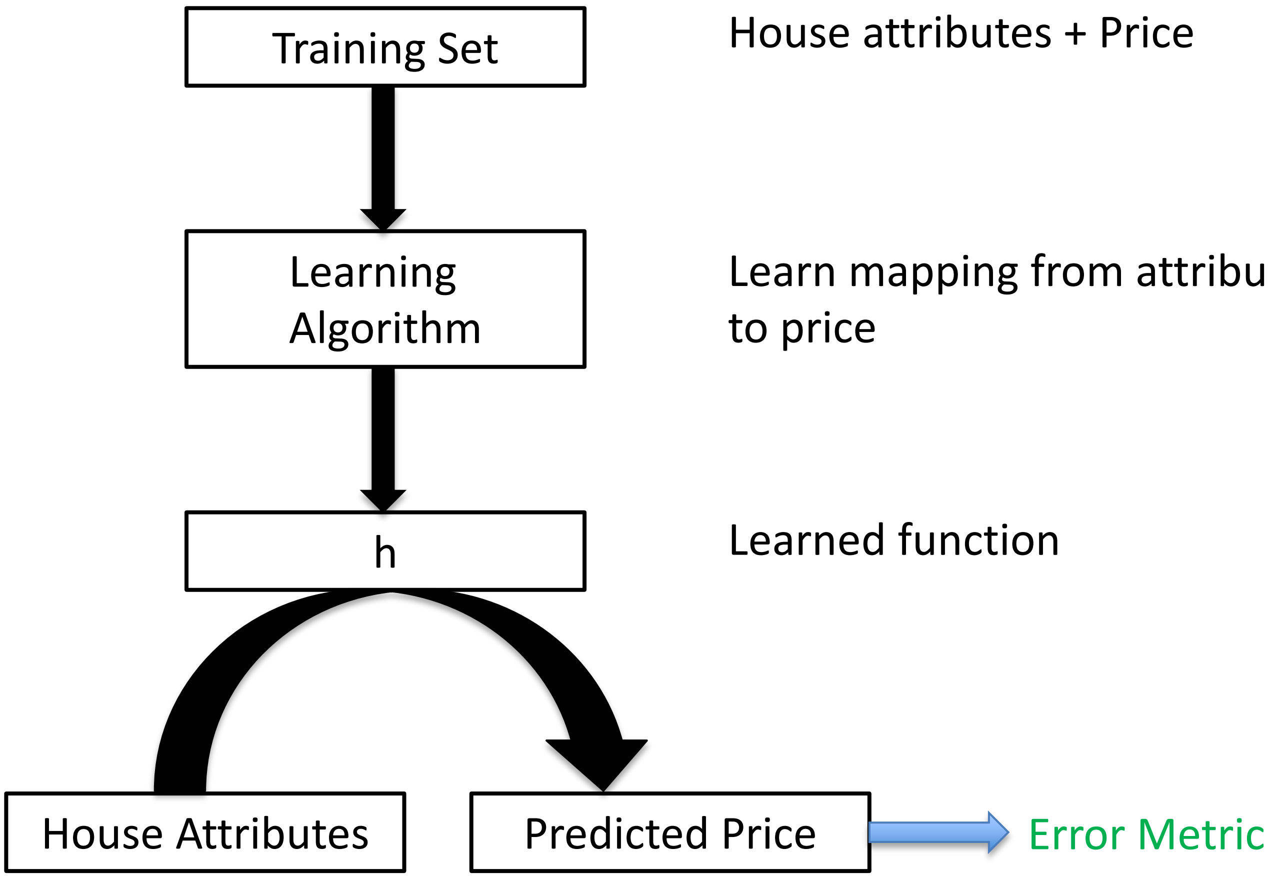
Error = 1/(2m)* $(f(x) - h(x))^2$

Think:

Is more training data good?

Always?

Learning Process



House attributes + Price

Learn mapping from attributes to price

Learned function

use it as a guide
After each training instance, refine h so that value of error metric goes down.

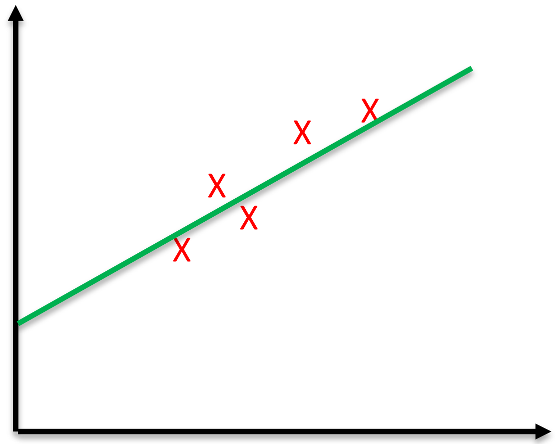
LINEAR REGRESSION

Learning a linear function

- Suppose we want to learn a function of the form:

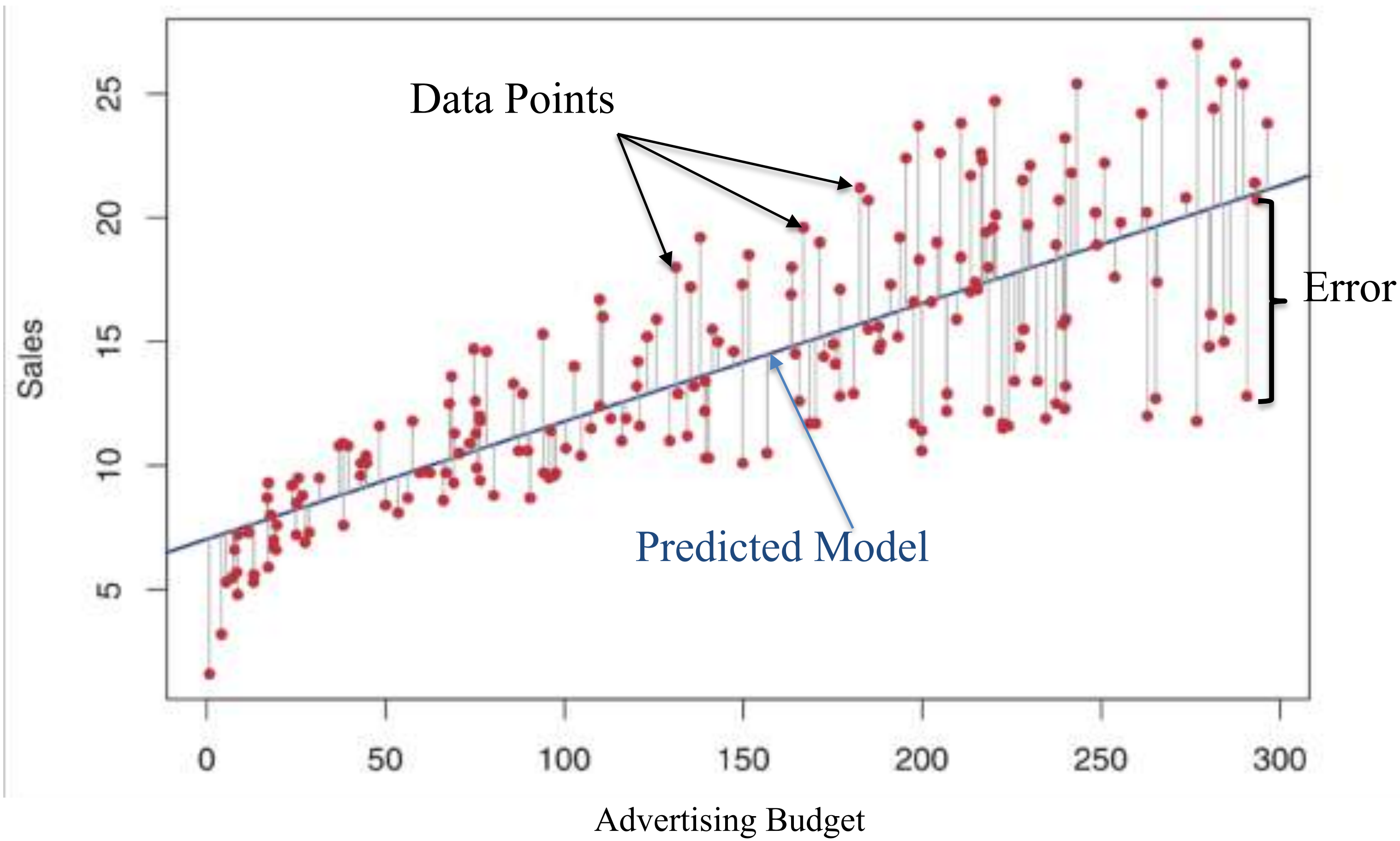
$$h_\theta(x) = \theta_0 + \theta_1 x$$

to represent house price. Let's say it's a one-D problem and only independent attribute is house size x.



Linear Regression

- Linear Regression – find best model that fits a continuous (i.e. real valued) output



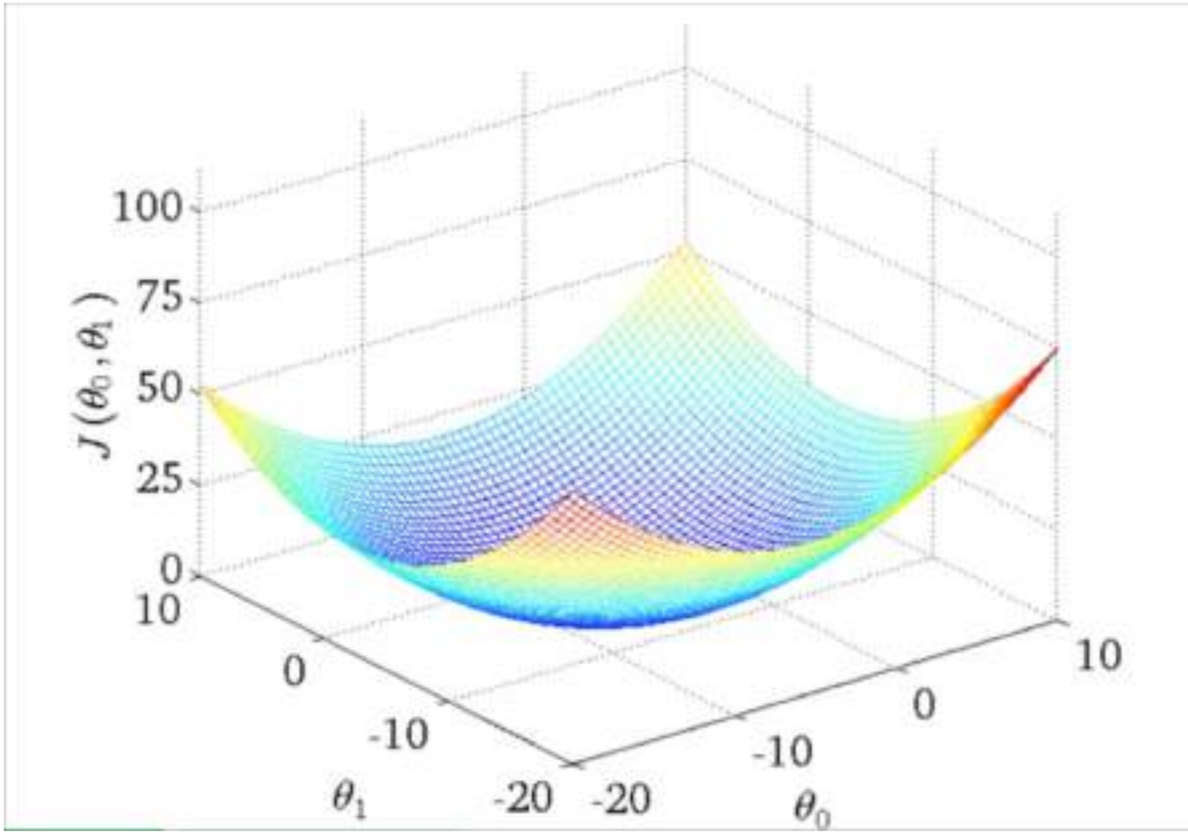
Error Function

- Our aim can be stated as:
  Choose parameters θ_0 and θ_1 such that our hypothesis $h_\theta(x)$ is as close to y for our training examples.
- Mathematically, choose parameters such that the following is minimized (called error or cost function). m is the number of training instances

$$E(\theta) = J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2$$

Error function

- How does J vary wrt the parameters
- Contour plot
- We are looking for the minima
- How do we get there?



Gradient Descent

- Given a function J of parameters Θ, how do we find its minimum or maximum.
- Gradient Descent is a very powerful and popular algorithm.
- Widely used in machine learning
- In many cases, analytical solution is not possible, so we have to randomly take steps in search of minimum.

Gradient Descent

- Aim: We have a function $J(\theta_0, \theta_1)$, and we want

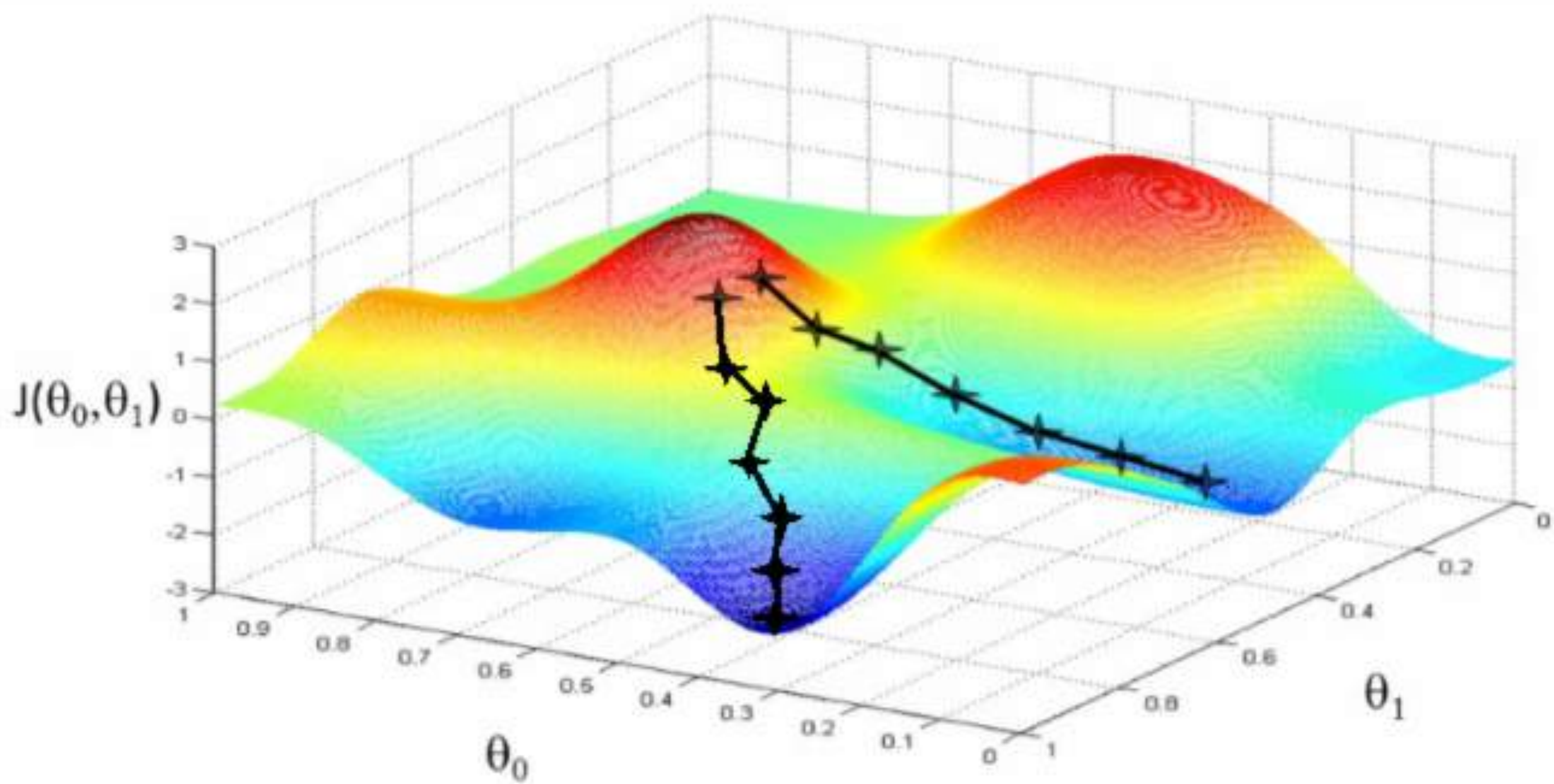
$$\underset{\theta_0\ \theta_1}{\operatorname{argmin}}\ J(\theta_0, \theta_1)$$

STEPS:

- Start with some random values
- Keep changing these values such that you achieve a reduction in J

Gradient Descent

- Imagine a man at a random point on the mountains.
- He needs to reach the city by walking randomly



Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

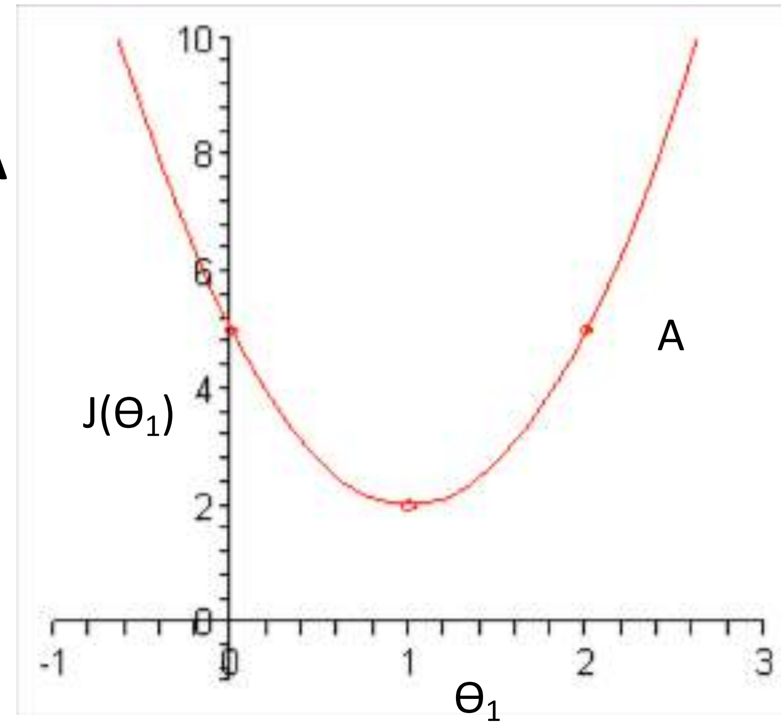
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\theta_1 := \text{temp1}$

α is called the learning rate
Intuition: It is how big a step you are taking.

Illustration

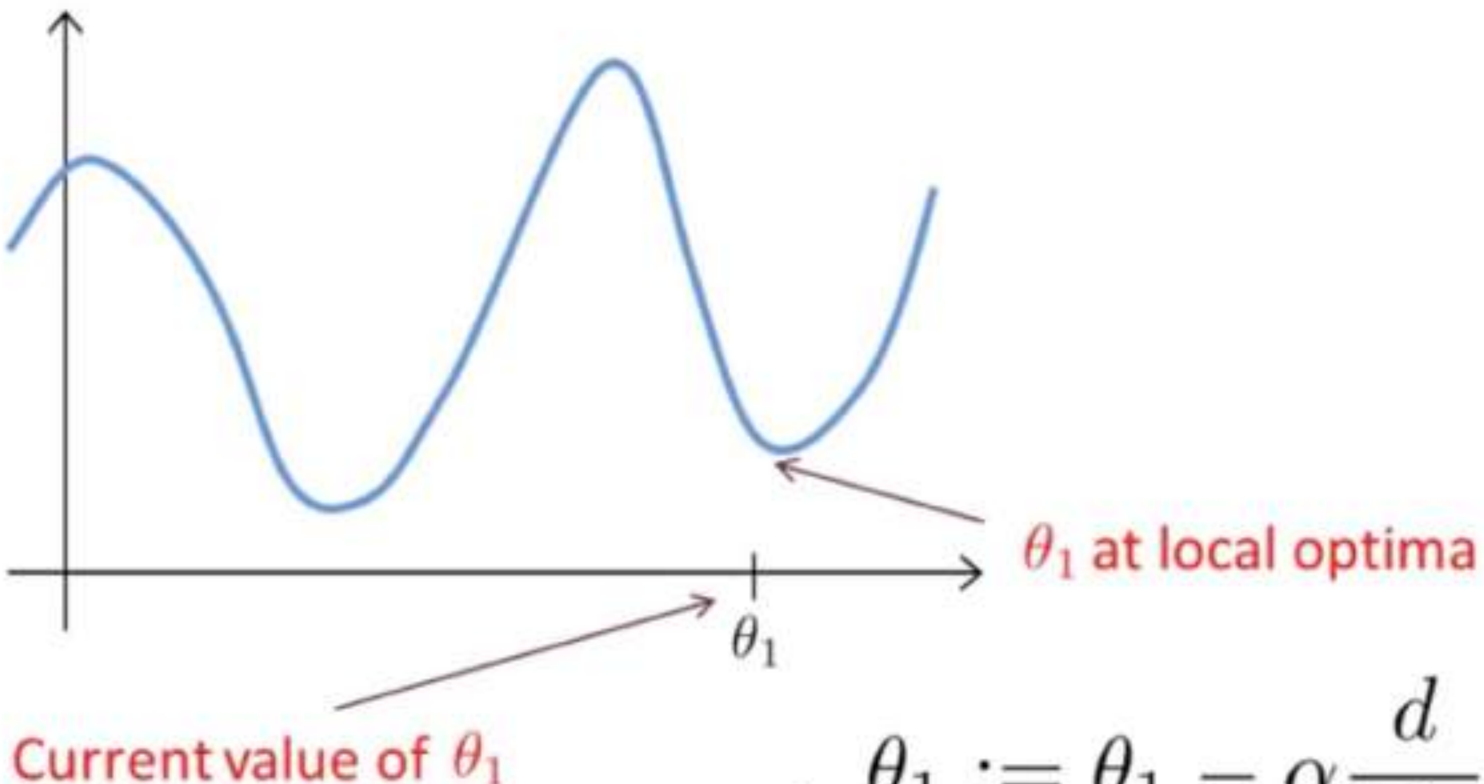
- In the curve on the right, imagine you are at point A
- The slope there is positive
- Update rule:

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$



since $\frac{\partial J}{\partial \theta_1}$ is positive and α is always positive, we would move towards left.

Local Minima can be a problem



$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for Linear Regression

Square meters	Bedrooms	Floors	Age of building (years)	Price in 1000€
x1	*x2*	*x3*	*x4*	*y*
200	5	1	45	460
131	3	2	40	232
142	3	2	30	315
756	2	1	36	178
...	...	...	...	...

- **Notation**

n – number of features (here $n = 4$)

$x^{(i)}$ – input features of ith training example

$x_j^{(i)}$ – feature j in ith training example

$$x^{(3)} = \begin{bmatrix} 142 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_1^{(4)} = 756$$

Gradient Descent for Linear Regression

Hypothesis representation

- $\mathrm{h}_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- **More compact**

$$\mathrm{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$ **with definition** $\mathrm{x}_0 := 1$

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \theta^T x$$

Gradient Descent for Linear Regression

Gradient descent for multiple variables

- **Generalized cost function** $\mathrm{J}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- **Generalized gradient descent**

$$
\begin{aligned}
&\textbf{while} \ \ \text{not converged:} \\
&\qquad \textbf{for} \ \ \text{all } j: \\
&\qquad\qquad tmp_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\
&\qquad \theta := \begin{bmatrix} tmp_0 \\ \vdots \\ tmp_n \end{bmatrix}
\end{aligned}
$$

Gradient Descent for Linear Regression

Partial derivative of cost function for multiple variables

- **Calculating the partial derivative**

$$
\begin{aligned}
\frac{\partial}{\partial\theta_j} J(\theta) &= \frac{\partial}{\partial\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\theta_0 x_0^{(i)} + \cdots + \theta_n x_n^{(i)}\right) - y^{(i)}\right)^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}
\end{aligned}
$$

Gradient Descent for Linear Regression

Gradient descent for multiple variables

- **Simplified gradient descent**

$$
\begin{aligned}
&\textbf{while not } \text{converged}: \\
&\quad \textbf{for } \text{all } j: \\
&\qquad tmp_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \\
&\quad \theta := \begin{bmatrix} tmp_0 \\ \vdots \\ tmp_n \end{bmatrix}
\end{aligned}
$$

Regression Evaluation Metrics

- Suppose we propose a linear model:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

  where ϵ represents the error.

- The coefficients β_0 and β_1 need to be estimated from the data (using gradient descent or other computational or analytical techniques).
- Let's suppose our estimates are $\widehat{\beta_0}$ and $\widehat{\beta_1}$, then the predicted value would be:

$$\hat{y} = \widehat{\beta_0} + \widehat{\beta_1} X$$

Regression Evaluation Metrics

- $e_i = \ \widehat{y_i} - y_i$ represents the residual or error for the i^th data point.
- Residual sum of square (RSS) is defined as:

$$RSS = e_1^2 + e_2^2 + \cdots + e_n^2$$

- By minimizing the RSS, we can arrive at the estimates $\widehat{\beta_0}$ and $\widehat{\beta_1}$

Regression Evaluation Metrics

- Standard error of estimation is defined as:

$$SE\left(\widehat{\beta_1}\right)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

and

$$SE\left(\widehat{\beta_0}\right)^2 = \sigma^2[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}]$$

where

$\sigma^2 = var(e)$ and $\overline{x_i} = \frac{\sum_{i=1}^{n} x_i}{n}$

Regression Evaluation Metrics

- These standard errors can be used to come up with confidence intervals(CI).
- A 95% CI means that there is 95% probability that the true value of an estimate will be within a specified interval.
- We can estimate the true parameter with 95% probability that it will be in the range:

$$[\widehat{\beta_i} - 2\ SE_i, \widehat{\beta_i} + 2\ SE_i]$$

Regression Hypothesis Testing

- SE can also be used for hypothesis testing.
- The general format of hypothesis testing between Y and X is:
  H_0: null hypothesis: There is no relation between Y and X
  H_1: alternate hypothesis: There is a relation between Y and X
- For example, if Y is price of a house, and X is the house size. You would like to check if there is a relation between Y and X.
  The null hypothesis will say there is no relation between price and size.

Regression Hypothesis Testing

- You would like to come up with evidence that the null hypothesis can be rejected and your hypothesis (i.e. there is a relation between Y and X) can be validated.
- If Y is the house price and X_1 is the house size, and we would like to check if $Y = \beta_0 + \beta_1 X_1$

Null hypothesis: $\beta_1 = 0$

Alternate hypothesis: $\beta_1 \neq 0$

We would like to come up with evidence that can reject the null hypothesis.

Regression Hypothesis Testing

- T-statistic of an estimate is given by:

$$t = \frac{\widehat{\beta_1} - 0}{SE(\widehat{\beta_1})}$$

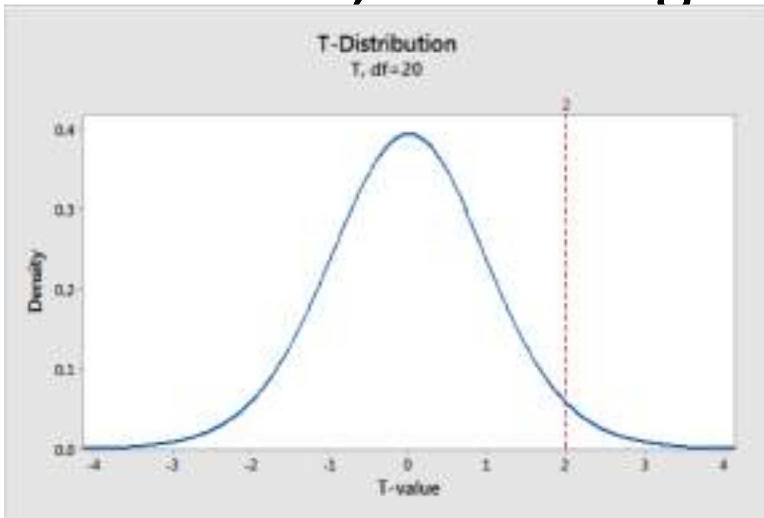
  this will have a t-distribution with n-2 degrees of freedom.

- T-statistics compares difference between two populations. Larger T-values indicate significant difference.

- In our case, we compare two populations: $\beta_1 = 0$ and $\beta_1 \neq 0$.

Regression Hypothesis Testing

- T-distributions assume the null hypothesis is true.
- Below is a plot which describes the probability of each t-value, assuming the null hypothesis.



Larger t-values are not common, if null hypothesis (H_0) is true.

- A t-value of 0 indicates a very high probability of null hypothesis being true.
- A large t-value indicates a rare case, if null hypothesis is true

Regression Hypothesis Testing

- p-value indicates the probability of observing the t-value with the null hypothesis.
- You would like the p-value to be small, if you would like to reject null hypothesis.
- Suppose we are doing a study of sales(Y) against TV ad spending. We get the results below:

	Coefficient	Std. Error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

A small p-value indicates that null hypothesis can be rejected.

Another evaluation metric

- We would like to check what fraction of data variance is explained by the model.
- R^2 statistic measures this:

$$R^2 = 1 - \frac{RSS}{TSS}$$

where RSS is the residual sum of squares (defined earlier) and TSS is the total sum of squares: $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$

Regression Packages

- For Python, see https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html

- For R see, http://r-statistics.co/Linear-Regression.html

Practice Question

- Consider the problem of predicting the number of A grades that a student at UTD will obtain in second year of M.S. based on the number of A grades obtained in the first year of M.S. course.

  Below is the data:

x	y
3	2
1	2
0	1
4	3

x represents the number of A grades in 1st year
y represents the number of A grades in 2nd year

You decide to use a hypothesis of the form $h_\theta(x) = \theta_0 + \theta_1 x$ where θ_0=0 and θ_1=1. Find the value of the squared error?

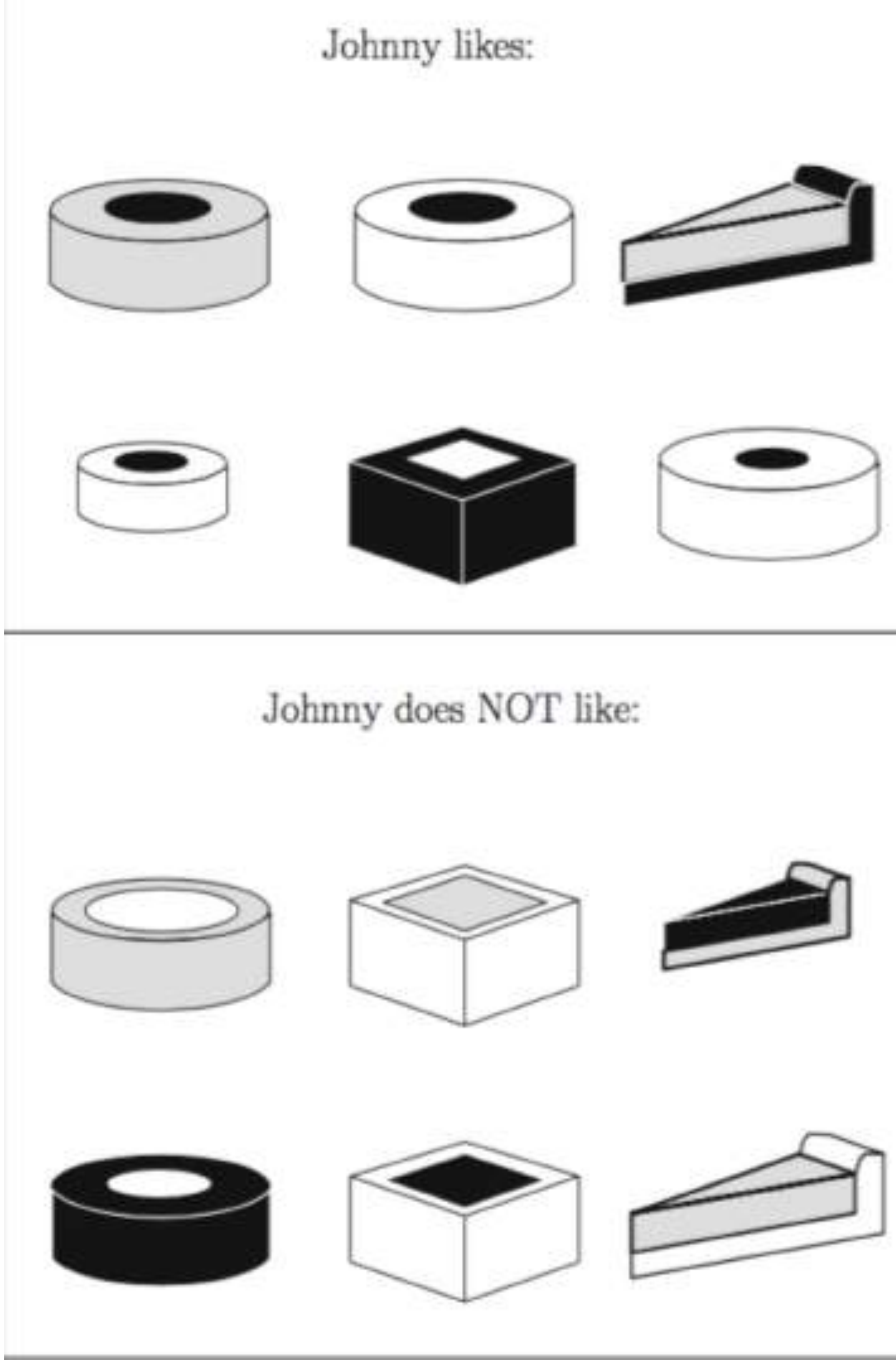
CLASSIFICATION

Classification example

- Learn which pies Johnny likes.

Another example

- Learn which pies Johnny likes.



Another example

- ML Notation

Attributes

Class Labels

Instances

example	shape	crust		filling		class
		size	shade	size	shade	
ex1	circle	thick	gray	thick	dark	pos
ex2	circle	thick	white	thick	dark	pos
ex3	triangle	thick	dark	thick	gray	pos
ex4	circle	thin	white	thin	dark	pos
ex5	square	thick	dark	thin	white	pos
ex6	circle	thick	white	thin	dark	pos
ex7	circle	thick	gray	thick	white	neg
ex8	square	thick	white	thick	gray	neg
ex9	triangle	thin	gray	thin	dark	neg
ex10	circle	thick	dark	thick	white	neg
ex11	square	thick	white	thick	dark	neg
ex12	triangle	thick	white	thick	gray	neg

Values of the attributes

- In the “pies” domain, there are five attributes:
  - shape (circle, triangle, and square),
  - crust-size (thin or thick),
  - crust-shade (white, gray, or dark),
  - filling-size (thin or thick),
  - filling-shade (white, gray, or dark).
- Question -> How many possible instances (types of pies) can you have?
- How many ways of labeling them can you have?

Instance Space

- The size of the *instance space* is
  3 x 2 x 3 x 2 x 3 = 108 different examples.
- You present Johnny a pie from this instance space => Johnny has two choices
  -> like or dislike

Class Labeling

Pie 1 -> 2 choices
Pie 2 -> 2 choices
….
….
Pie 108 -> 2 choices

Total ways of labeling = 2^{108}

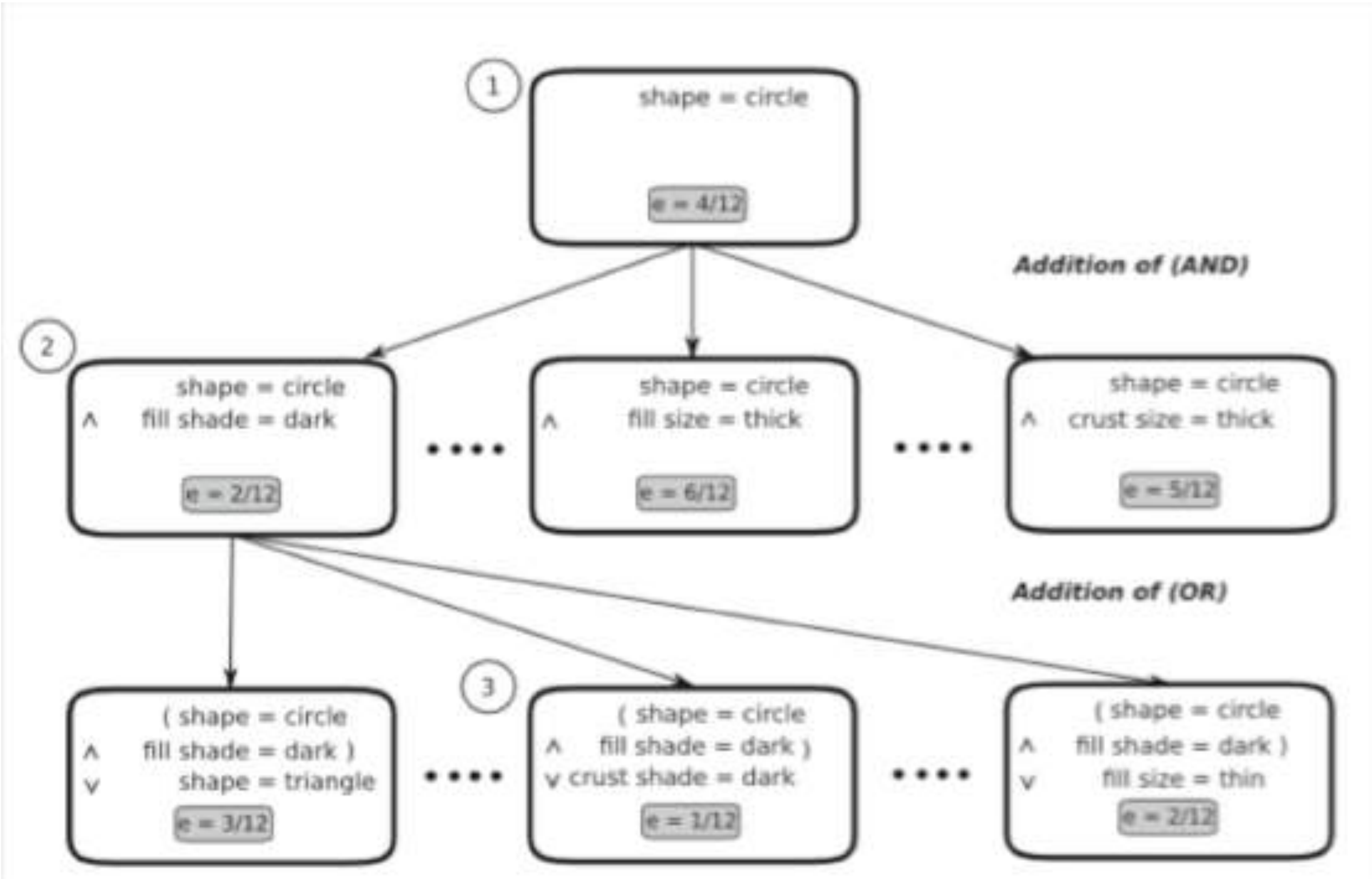
If you really wanted to know Johnny's choices, you would have to find out which of these labelings apply to him.

Hypothesis Generation

We won't try to learn all possible labellings. We will restrict ourselves to certain types
e.g. Conjunctions and Disjunctions

- You can make hypothesis in the form of conjunctions:
  (Shape = circle) ^ (Fill Shade = dark)
- or a combination of conjunctions and disjunctions:
  (Shape = circle ^ Fill Shade = dark) V
  (Fill Size = thin)

Finding Errors in Hypotheses



Hypothesis Evaluation

- In short, we are searching through the space of all possible hypothesis for a hypothesis that best matches the training data perfectly.

Hypotheses Boundaries

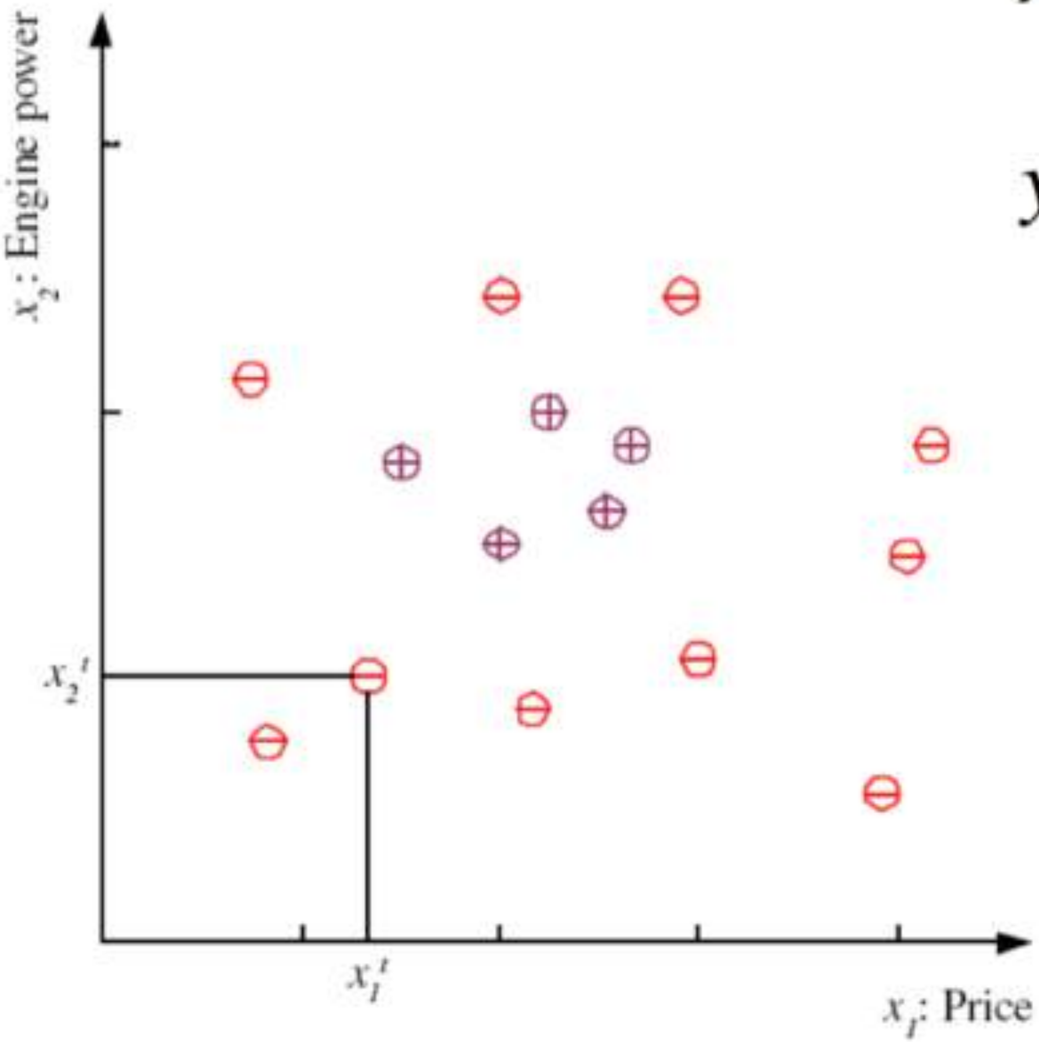
- Can we find the most general and most specific boundaries of hypotheses?
- Let's look at an example.

Example

- Trying to classify a car as “family car”
  f(x) = 1 if x is family car
  f(x) = 0 otherwise

$$x = \left(x_1 \; x_2 \right)^T$$

x1: price
x2: engine power

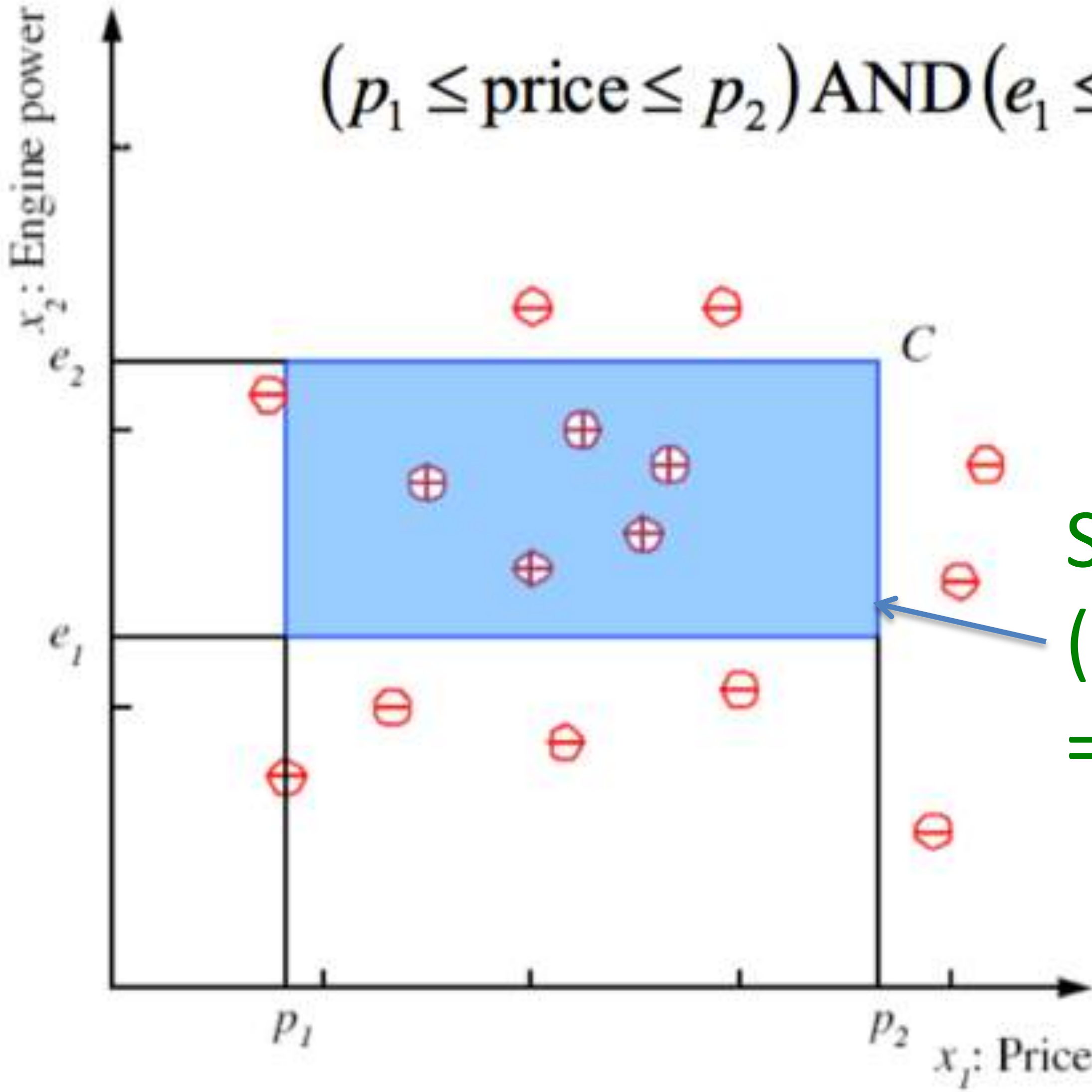


$$\mathcal{X} = \{\mathbf{x}^t, y^t\}_{t=1}^{N}$$

$$y = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

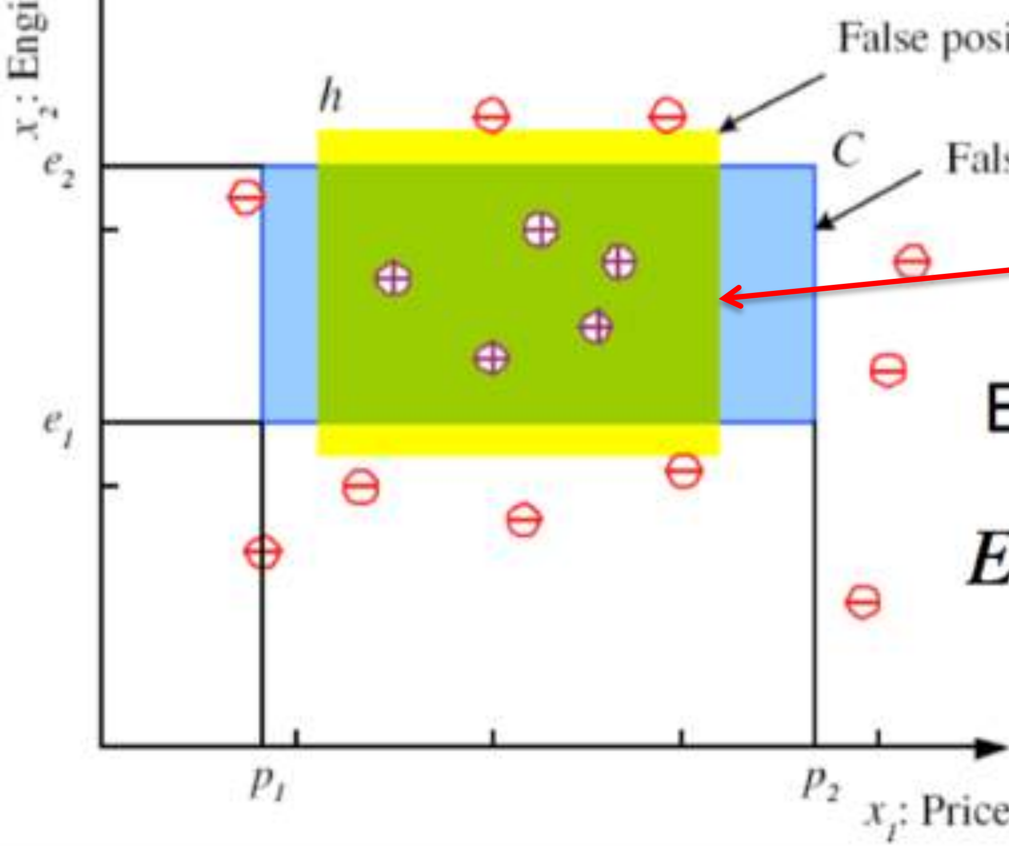
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

How many hypotheses can you propose?



Suppose this is f
(the real function)
=> you can't see it ☺

$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ classifies } \mathbf{x} \text{ as positive} \\ 0 \text{ if } h \text{ classifies } \mathbf{x} \text{ as negative} \end{cases}$$



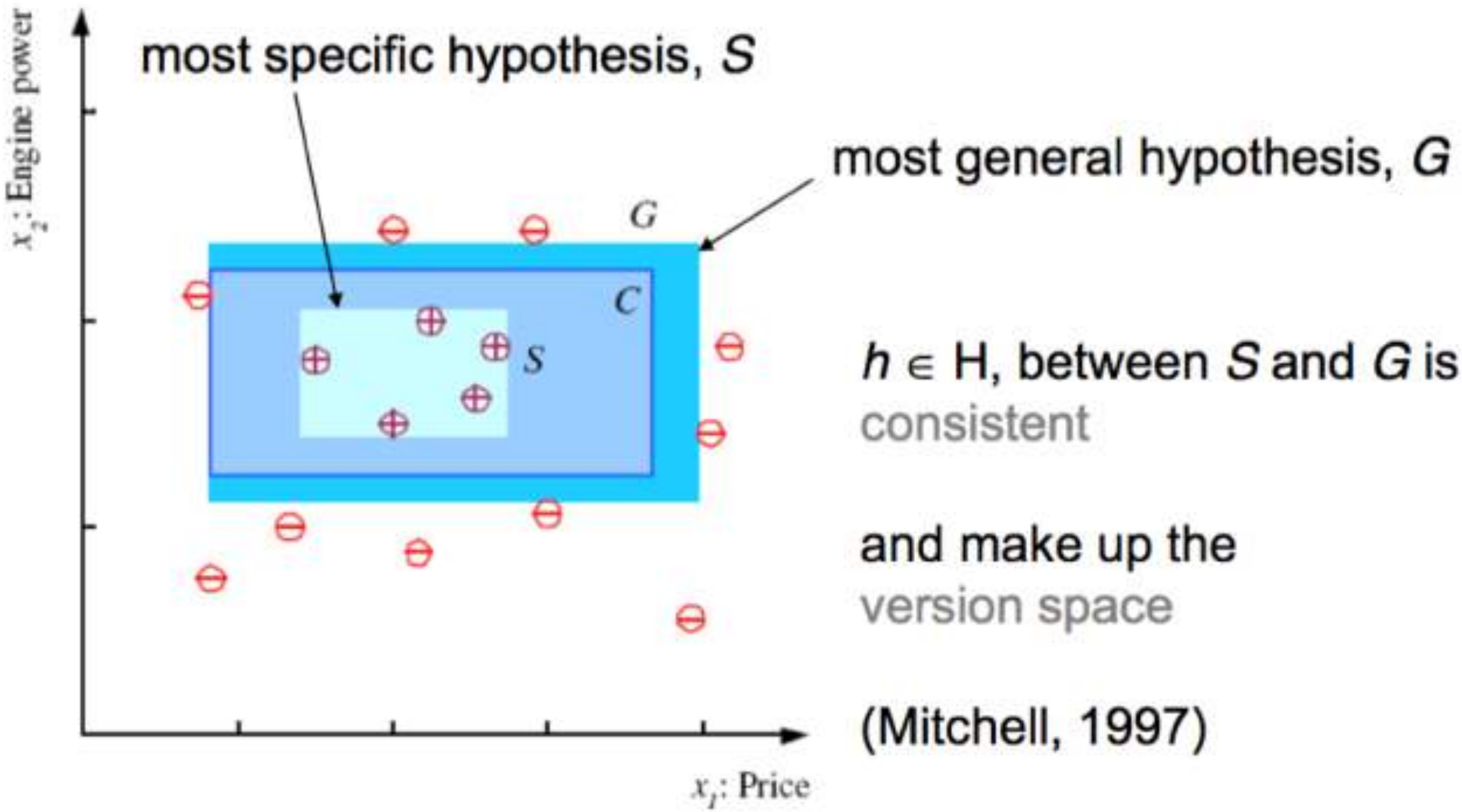
You propose this hypothesis h

Error of h on H

$$E(h \mid \mathsf{X}) = \sum_{t=1}^{N} 1\left(h\left(\mathbf{x}^t\right) \neq y^t\right)$$

Can you compute E?

S, G, and Version Space



Concept Learning

- Discrete attributes are more common so we will get back to those.
- Let's see another example
- Technically it's called learning a concept or Concept Learning

Concept Learning

You want to learn conditions in which a person (say John) enjoys playing tennis. <- This is the concept

Training data:

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

6 attributes, boolean classification

Training Examples for EnjoySport

	Sky	Temp	Humid	Wind	Water	Forecst		EnjoySpt
c(	Sunny	Warm	Normal	Strong	Warm	Same	)=1	Yes
c(	Sunny	Warm	High	Strong	Warm	Same	)=1	Yes
c(	Rainy	Cold	High	Strong	Warm	Change	)=0	No
c(	Sunny	Warm	High	Strong	Cool	Change	)=1	Yes

- Negative and positive learning examples
- Concept learning:

  c is the target concept

  - Deriving a Boolean function from training examples
    - Many "hypothetical" boolean functions
      - Hypotheses; find h such that h = c.
  - Other more complex examples:
    - Non-boolean functions
- Generate hypotheses for concept from TE's

Representing Hypotheses

- Task of finding appropriate set of hypotheses for concept given training data
- Represent hypothesis as Conjunction of constraints of the following form:
  - Values possible in any hypothesis
    - Specific value : Water = *Warm*
    - Don't-care value: Water = ?
    - No value allowed : Water = ∅ (This is an abstract concept, used for initialization)
      - i.e., no permissible value given values of other attributes
  - Use vector of such values as hypothesis:
    - ⟨ Sky AirTemp Humid Wind Water Forecast ⟩
      - Example: ⟨*Sunny* ? ? *Strong* ? *Same* ⟩
- Idea of *satisfaction of hypothesis* by some example
  - say "example satisfies hypothesis"
  - defined by a function *h(x):*

$$h(x) = 1 \text{ if } h \text{ is true on } x$$
$$= 0 \text{ otherwise}$$

- Want hypothesis that best fits examples:
  - Can reduce learning to search problem over space of hypotheses

Hypothesis representation

- A hypothesis:

  Sky AirTemp Humidity Wind Water Forecast

  < Sunny, ? , ? , Strong , ? , Same >

- *The most general hypothesis* – that every day is a positive example

  <?, ?, ?, ?, ?, ?>

- *The most specific hypothesis* – that no day is a positive example

  <0, 0, 0, 0, 0, 0>

- *EnjoySport concept learning task* requires learning the sets of days for which EnjoySport=yes, describing this set by a conjunction of constraints over the instance attributes.

Concept Learning

Given

- ***Instances X*** : set of all possible days, each described by the attributes
  - Sky – (values: Sunny, Cloudy, Rainy)
  - AirTemp – (values: Warm, Cold)
  - Humidity – (values: Normal, High)
  - Wind – (values: Strong, Weak)
  - Water – (values: Warm, Cold)
  - Forecast – (values: Same, Change)
- ***Target Concept (Function) c*** : EnjoySport : $X \rightarrow \{0,1\}$
- ***Hypotheses H*** : Each hypothesis is described by a conjunction of constraints on the attributes.
- ***Training Examples D*** : positive and negative examples of the target function

Determine

- A hypothesis h in H such that h(x) = c(x) for all x in D.

Instances and Hypotheses

- How many possible instances?
  Look at all combinations of attributes
  3 x 2 x 2 x 2 x 2 x 2 = 96
- What if you don't care about a specific value of attribute i.e. it can be denoted by ?
  Possible combinations (each increases by 1):
  4 x 3 x 3 x 3 x 3 x 3 = 972

How many labeling of instances?

- Let’s consider all instances (ignoring ?) – 96
- In how many ways can each instance be labeled?
  Each instance can be 0 (false) or 1 (true)
- So total number of labeling of conjunctions (instances)= 2^{96}

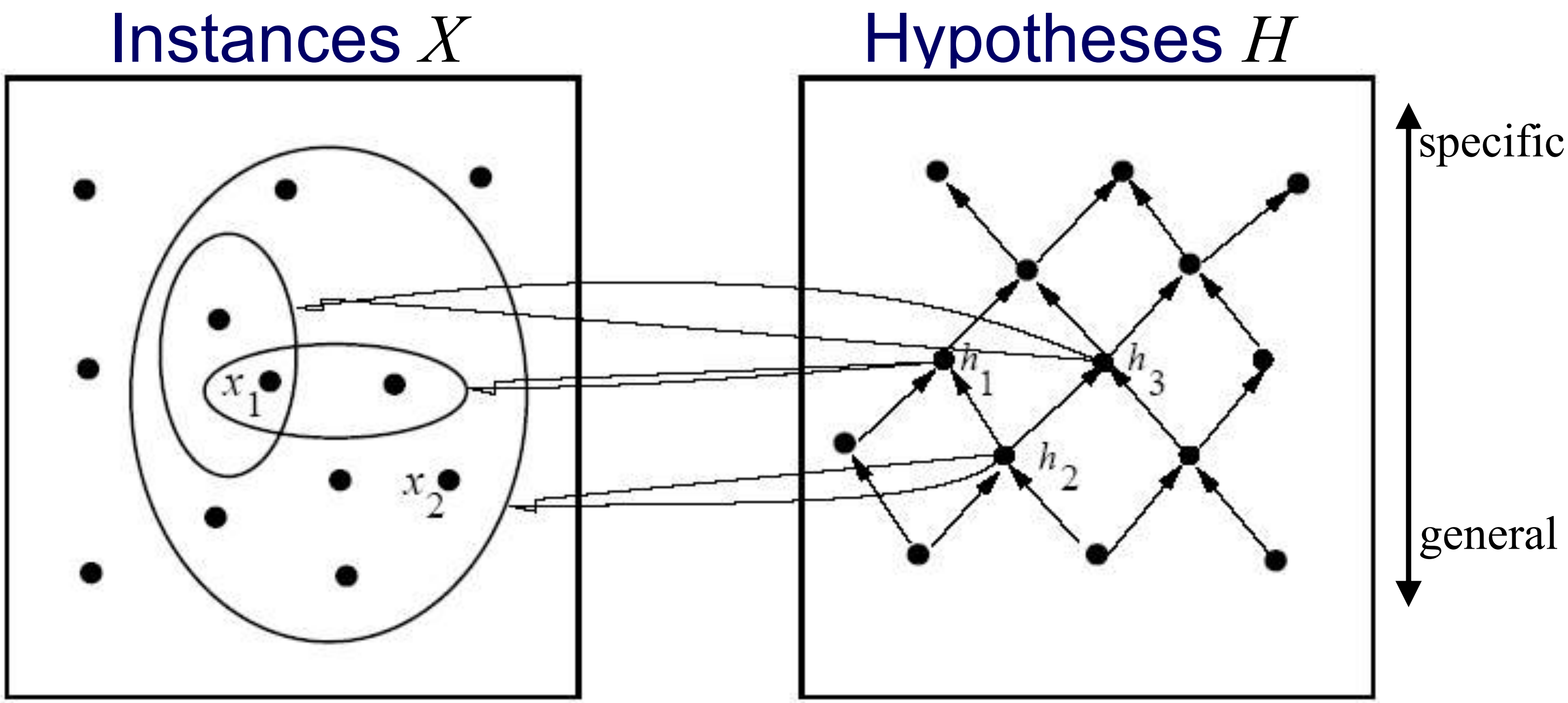
General to Specific Ordering

- Consider two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)
  h2 = (Sunny, ?, ?, ?, ?, ?)

- Now consider the sets of instances that are classified positive by hl and by h2.
  - Because h2 imposes fewer constraints on the instance, it classifies more instancesas positive.
  - In fact, any instance classified positive by hl will also be classified positive by h2.
  - Therefore, we say that h2 is *more general than* hl.

Ordering on Hypotheses



x_1=⟨*Sunny Warm High Strong Cool Same*⟩

x_2=⟨*Sunny Warm High Light Warm Same*⟩

h_1=⟨*Sunny ? ? Strong ? ?*⟩

h_2=⟨*Sunny ? ? ? ? ?*⟩

h_3=⟨*Sunny ? ? ? Cool ?*⟩

- h is more general than h'($h \geq_g h'$) if for each instance x, $h'(x) = 1 \rightarrow h(x) = 1$
- Which is the most general/most specific hypothesis?

Find-S Algorithm

Assumes

There is hypothesis h in H describing target function c

There are no errors in the TEs

Procedure

1. Initialize *h* to the most specific hypothesis in *H (what is this?)*
2. For each *positive* training instance *x*

   For each attribute constraint a_i in *h*

   If the constraint a_i in *h* is satisfied by *x*

   do nothing

   Else

   replace a_i in *h* by the next more general constraint that is satisfied by *x*
3. Output hypothesis *h*

Note

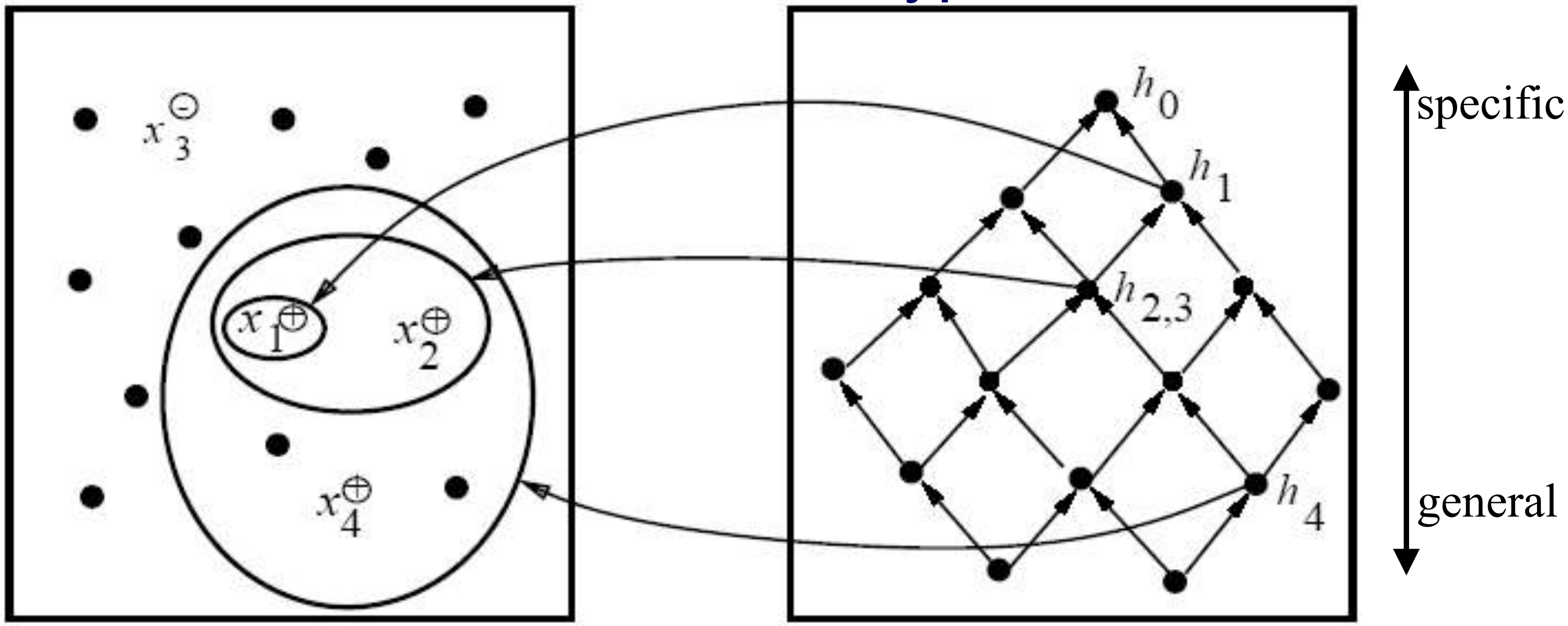
There is no change for a negative example, so they are ignored.

This follows from assumptions that there is h in H describing target function c *(ie.,for this h, h=c)* *and* that there are no errors in data. In particular, it follows that the hypothesis at any stage cannot be changed by neg example.

Assumption: Everything except the positive examples is negative

Example of Find-S

Instances X | **Hypotheses H**



$x_1 = \langle Sunny\ Warm\ Normal\ Strong\ Warm\ Same \rangle +$

$x_2 = \langle Sunny\ Warm\ High\ Strong\ Warm\ Same \rangle +$

$x_3 = \langle Rainy\ Cold\ High\ Strong\ Warm\ Change \rangle -$

$x_4 = \langle Sunny\ Warm\ High\ Strong\ Cool\ Change \rangle +$

$h_0 = \langle \varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing \rangle$

$h_1 = \langle Sunny\ Warm\ Normal\ Strong\ Warm\ Same \rangle$

$h_2 = \langle Sunny\ Warm\ ?\ Strong\ Warm\ Same \rangle$

$h_3 = \langle Sunny\ Warm\ ?\ Strong\ Warm\ Same \rangle$

$h_4 = \langle Sunny\ Warm\ ?\ Strong\ ?\ ? \rangle$

Example

Origin	Manufacturer	Color	Decade	Type	
Japan	Honda	Blue	1980	Economy	+
Japan	Toyota	Green	1970	Sports	-
Japan	Toyota	Blue	1990	Economy	+
USA	Chrysler	Red	1980	Economy	-
Japan	Honda	White	1980	Economy	+

Use find-S algorithm for the concept of Economy class

And this

Origin	Manufacturer	Color	Decade	Type	
Japan	Honda	Blue	1980	Economy	+
Japan	Toyota	Green	1970	Sports	-
Japan	Toyota	Blue	1990	Economy	+
USA	Chrysler	Red	1980	Economy	-
Japan	Honda	White	1980	Economy	+
Japan	Toyota	Green	1980	Economy	+
Japan	Honda	Red	1990	Economy	-

Use find-S algorithm for the concept of Economy class

Problems with Find-S

- Problems:
  - Throws away information!
    - Negative examples
  - Can't tell whether it has learned the concept
    - Depending on *H*, there might be several h's that fit TEs!
    - Picks a maximally specific *h* (why?)
  - Can't tell when training data is inconsistent
    - Since ignores negative TEs
- But
  - It is simple
  - Outcome is independent of order of examples
    - Why?
- What alternative overcomes these problems?
  - Keep *all* consistent hypotheses!
    - Candidate elimination algorithm

Consistent Hypotheses and Version Space

- A hypothesis h is consistent with a set of training examples D of target concept c if $h(x) = c(x)$ for each training example $\langle x, c(x)\rangle$ in D
  - Note that consistency is with respect to specific D.
- Notation:
  $Consistent\,(h, D) \equiv \forall\langle x, c(x)\rangle \in D :: h(x) = c(x)$
- The version space, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with D
- Notation:
  $VS_{H,D} = \{h \mid h \in H \wedge Consistent\,(h, D)\}$

List-Then-Eliminate Algorithm

1. *VersionSpace* $\leftarrow$ list of all hypotheses in H
2. For each training example $\langle x, c(x) \rangle$

   remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

- This is essentially a brute force procedure
- *Can we do better than this?*

Representing Version Spaces

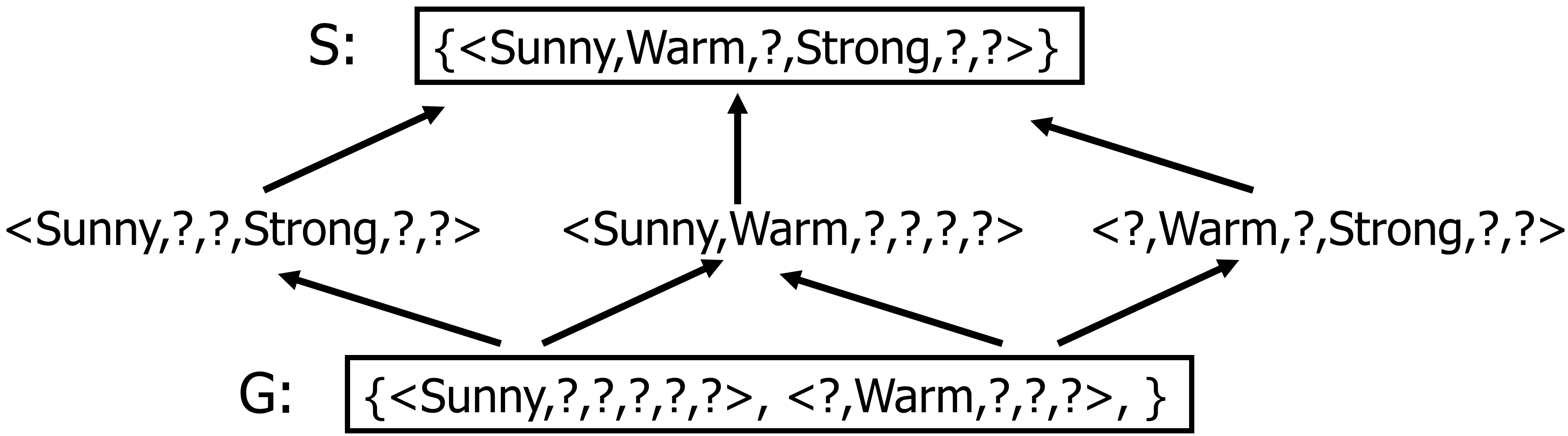
- The **general boundary**, G, of version space $VS_{H,D}$ is the set of maximally general members.
- The **specific boundary**, S, of version space $VS_{H,D}$ is the set of maximally specific members.
- Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H \mid (\exists\, s \in S)\ (\exists\, g \in G)\ (g \geq h \geq s)\}$$

where $x \geq y$ means x is more general or equal than y

Version Space is the set of all hypotheses that are more general than the s hypothesis and less general than the g hypothesis.

Example Version Space



x_1 = <Sunny Warm Normal Strong Warm Same> +
x_2 = <Sunny Warm High Strong Warm Same> +
x_3 = <Rainy Cold High Strong Warm Change> -
x_4 = <Sunny Warm High Strong Cool Change> +

Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H

S $\leftarrow$ maximally specific hypotheses in H

For each training example d=<x,c(x)>

If d is a positive example

Remove from G any hypothesis that is inconsistent with d

For each hypothesis s in S that is not consistent with d

- remove s from S.
- Add to S all minimal generalizations h of s such that
  - h consistent with d
  - Some member of G is more general than h
- Remove from S any hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm

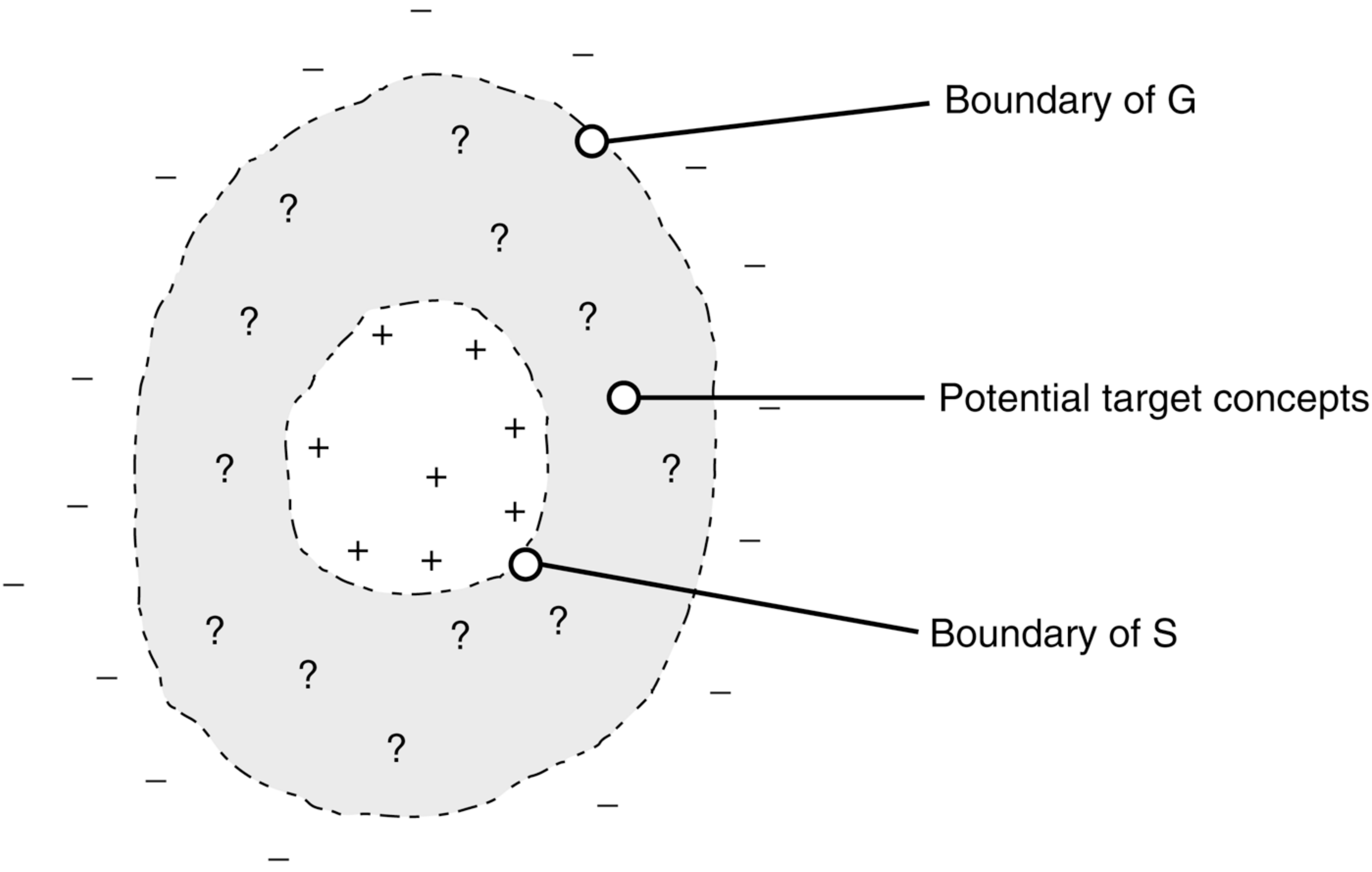
If d is a negative example

Remove from S any hypothesis that is inconsistent with d

For each hypothesis g in G that is not consistent with d

- remove g from G.
- Add to G all minimal specializations h of g such that
  - h consistent with d
  - Some member of S is more specific than h
- Remove from G any hypothesis that is less general than another hypothesis in G

Candidate Elimination Algorithm



Example Candidate Elimination

S: ~~{<∅, ∅, ∅, ∅, ∅, ∅ >}~~

G: {<?, ?, ?, ?, ?, ?>}

x_1 = <Sunny Warm Normal Strong Warm Same> +

S: ~~{< Sunny Warm Normal Strong Warm Same >}~~

G: {<?, ?, ?, ?, ?, ?>}

x_2 = <Sunny Warm High Strong Warm Same> +

S: {< Sunny Warm ? Strong Warm Same >}

G: {<?, ?, ?, ?, ?, ?>}

Example Candidate Elimination..

S: {< Sunny Warm ? Strong Warm Same >}

G: ~~{<?, ?, ?, ?, ?, ?>}~~

x_3 = <Rainy Cold High Strong Warm Change> -

S: ~~{< Sunny Warm ? Strong Warm Same >}~~

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?>, ~~<?,?,?,?,?,Same>~~}

x_4 = <Sunny Warm High Strong Cool Change> +

S: {< Sunny Warm ? Strong ? ? >}

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?> }

Example Candidate Elimination..

S: {<Sunny,Warm,?,Strong,?,?>}

<Sunny,?,?,Strong,?,?> <Sunny,Warm,?,?,?,?> <?,Warm,?,Strong,?,?>

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?>, }

x_1 = <Sunny Warm Normal Strong Warm Same> +
x_2 = <Sunny Warm High Strong Warm Same> +
x_3 = <Rainy Cold High Strong Warm Change> -
x_4 = <Sunny Warm High Strong Cool Change> +

Remarks on VS & CE

- Converges to correct h?
  - If no errors, H rich enough: yes
  - exact: G = S and both singletons
- Effect of errors (noise)
  - example 2 negative -> CE removes the correct target from VS!
  - detection: VS gets empty
- Similarly, VS gets empty when c can not be represented (e.g. disjunctions)

Classification of New Data

S: {<Sunny,Warm,?,Strong,?,?>}

<Sunny,?,?,Strong,?,?> <Sunny,Warm,?,?,?,?> <?,Warm,?,Strong,?,?>

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, }

x_5 = <Sunny Warm Normal Strong Cool Change> + 6/0

x_6 = <Rainy Cold Normal Light Warm Same> - 0/6

x_7 = <Sunny Warm Normal Light Warm Same> ? 3/3

x_8 = <Sunny Cold Normal Strong Warm Same> ? 2/4

How to use partially learned concepts?

- Classification with ambiguous VS
  - h(x) = 0/1 for every h in VS: ok
  - enough to check with G (0) & S (1)
  - 3rd example: 50% support for both
  - 4th: 66% support for 0
  - majority vote + confidence?
  - Ok if all h are equally likely

Inductive Bias

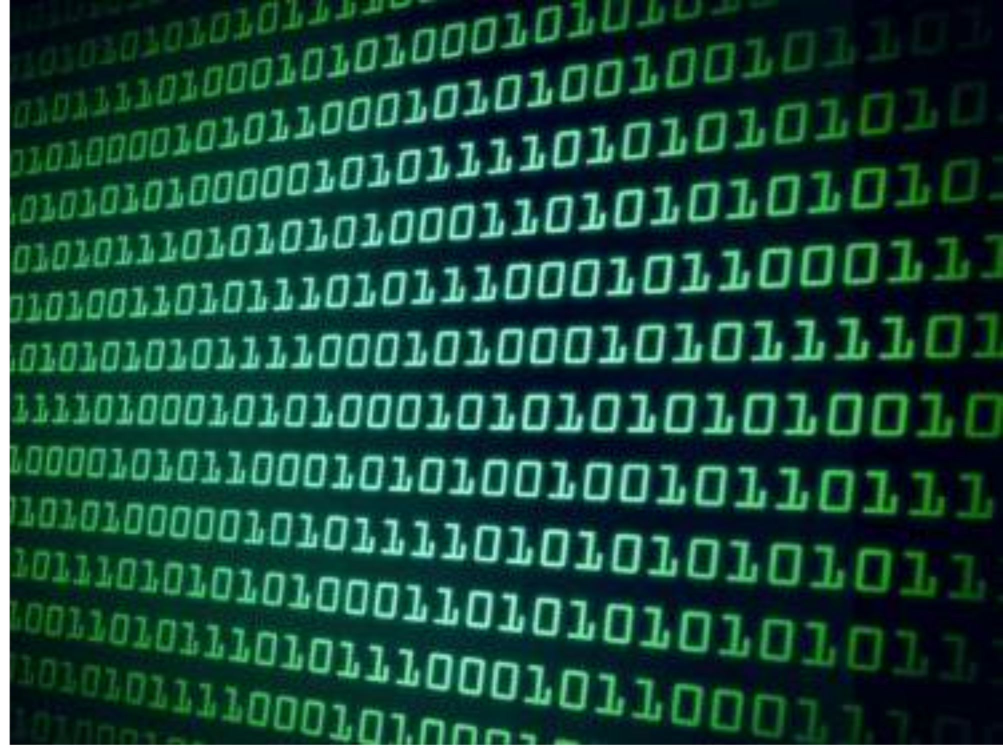
- Have we learned every possible representation of the concept?
- No, we have only learned concepts that can be represented as conjunctions:
  h: $(x_1 = a_1) \wedge (x_2 = a_1) \ldots \wedge (x_n = a_n)$
- We can not even represent simple disjunctions. e.g. We can not represent:
  (Sky=Sunny) v (Sky=Cloudy)

Inductive Bias

- For 96 possible instances, there are 2^{96} distinct possible concepts.
- We had an inductive bias (form of hypothesis) in the form of conjunctions.
- Can a learner be free of any bias?

Futility of Bias-Free Learning

- If
  - learner makes no a priori assumptions on the target concept
- Then
  - it has no rational basis for classifying any unseen instances
- Inductive bias = prior assumptions
- **Inductive bias is the preference for a hypothesis space H and a search mechanism over H.**



A bit of Boolean algebra

Conjunctions & Disjunctions

Suppose you want to learn following function:

f: X -> Y where X = $\{0, 1\}^n$ and Y = {0, 1}
i.e. there are n Boolean attributes and output (class) is also a Boolean.

- What type of hypotheses can you propose?
- How many hypotheses can you propose?

Conjunctions

Assume hypotheses are in form of monotone conjunctions i.e. there are no negations.

For example:

$x1 \wedge x2 \wedge \ldots \wedge xn$

A negation would be something like:

$x1 \wedge \neg x2 \wedge \ldots \wedge \neg xn$

Monotone Conjunctions

Example: An animal can be described by 5 attributes

x1: is a bird

x2: is a mammal

x3: lays eggs

x4: able to fly

x5: lives in North America

Conjunction could be:

$x2 \wedge x3$

Conjunctions

How to learn:

Simple:

- Take bitwise AND of all positive examples
- Construct monotone conjunction of these attributes
- Check over negative examples
    - If you get (+) for any negative example, the concept is not learnable.

Example

example	label
0 1 1 0 1	+
1 1 0 1 1	+
1 1 0 0 1	+
0 0 1 0 1	−
1 1 0 0 0	−

AND of positive examples leads to:
x2 ^ x5

Consistent concept

Conjunctive Normal Form

Simply stated, it is a conjunction of disjunctions

- $\neg A \wedge (B \vee C)$
- $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$
- $A \vee B$
- $A \wedge B$

k-CNF is a CNF such that each clause
has at most size k

E.g., $x_4(x_1 \vee x_2)(x_2 \vee \bar{x}_3)$ is a 2-CNF

Disjunctive Normal Form (DNF)

It is a disjunction of conjunctions:

i.e.

$(x1 \wedge x2) \vee (x1 \wedge \neg x3)$

Very easy to learn: For each positive instance, create a conjunction and then create disjunction of all these.

Guitar	Fast beat	Male singer	Acoustic	New	**Liked**
1	0	0	1	1	**1**
1	1	1	0	0	**0**
0	1	1	0	1	**0**
1	0	1	1	0	**1**
1	0	0	0	1	**0**

Monotone Conjunctions: x1 ^ x4

Example of DNF: (x1 ^ ¬x2 ^ ¬x3 ^ x4 ^x5) V (x1 ^¬ x2 ^ x3 ^ x4 ^¬x5)

This just memorizes the positive examples

k-DNF are useful in decision trees, which we will study next

Hypotheses Spaces

If you have 4 Boolean attributes and 1 Boolean output, how many instances and how many labeling (i.e. hypotheses) can you have?

x1	x2	x3	x4	y
0	0	0	0	..
…				
1	1	1	1	…

Number of instances = $|X| = 2^4$

Each instance has 2 labeling choice, so labeling possible

$= 2^{|X|}$

$= 2^{2^{\wedge}4}$

Question for the smart student: Can you make a decision tree to represent each instance and labeling?