Linear Regression

Simple linear regression lives up to its name: it is a very straightforward simple linear approach for predicting a quantitative response Y on the basis of a si gle predictor variable X. It assumes that t here is approximately a linear relationship between X and Y . Mathematically, we can write this linear relationship as

$Y \approx \beta0 + \beta1X$.

For example, X may represent TV advertising and Y may represent sales. Then we can regress sales onto TV by fitting the model sales ≈ β0 + β1 × TV.

, β0 and β1 are two unknown constants that represent the intercept and slope terms in the linear model. Together, β0 and β1 are intercept slope known as the model coefficients or parameters. Once we have used our coefficient parameter training data to produce estimates βˆ0 and βˆ1 for the model coefficients, we can predict future sales on the basis of a particular value of TV advertising by computing yˆ = βˆ0 + βˆ1x,

How to Find the Regression Equation

How to Find the Regression Equation

In the table below, the xi column shows scores on the aptitude test. Similarly, the

yi column shows statistics grades. The last two columns show deviations scores - the difference between the student's score and the average score on each test. The last two rows show sums and mean scores that we will use to conduct the regression analysis.

Student	xi	yi	(xi-x)	(yi-y)
1	95	85	17	8
2	85	95	7	18
3	80	70	2	-7
4	70	65	-8	-12
5	60	70	-18	-7
Sum	390	385		
Mean	78	77		

And for each student, we also need to compute the squares of the deviation scores (the last two columns in the table below).

Student	xi	yi	$(xi-x)^2$	$(yi-y)^2$
1	95	85	289	64

2	85	95	49	324
3	80	70	4	49
4	70	65	64	144
5	60	70	324	49
Sum	390	385	730	630
Mean	78	77		

And finally, for each student, we need to compute the product of the deviation scores.

Student	**xi**	**yi**	**(xi-x)(yi-y)**
1	95	85	136
2	85	95	126
3	80	70	-14
4	70	65	96
5	60	70	126
Sum	390	385	470
Mean	78	77	

The regression equation is a linear equation of the form: ŷ = b0 + b1x . To conduct a regression analysis, we need to solve for b0 and b1. Computations are shown below. Notice that all of our inputs for the regression analysis come from the above three tables.

First, we solve for the regression coefficient (b1):

$b1 = \Sigma [(xi - x)(yi - y)] / \Sigma [(xi - x)^2]$

b1 = 470/730

b1 = 0.644

Once we know the value of the regression coefficient (b1), we can solve for the regression slope (b0):

b0 = y - b1 * x

$b0 = 77 - (0.644)(78)$

$b0 = 26.768$

Therefore, the regression equation is: $\hat{y} = 26.768 + 0.644x$.

Assessing the Accuracy of the Model

How to Find the Coefficient of Determination

Whenever you use a regression equation, you should ask how well the equation fits the data. One way to assess fit is to check the coefficient of determination, which can be computed from the following formula.

$R^2 = \{ (1 / N) * \Sigma [(xi - x) * (yi - y)] / (\sigma x * \sigma y) \}^2$

where N is the number of observations used to fit the model, Σ is the summation symbol, xi is the x value for observation i, x is the mean x value, yi is the y value for observation i, y is the mean y value, σx is the standard deviation of x, and σy is the standard deviation of y.

Computations for the sample problem of this lesson are shown below. We begin by computing the standard deviation of x (σx):

$\sigma x = sqrt [\Sigma (xi - x)^2 / N]$

$\sigma x = sqrt(730/5) = sqrt(146) = 12.083$ Next, we find the standard deviation of y, (σy):

$\sigma y = sqrt [\Sigma (yi - y)^2 / N]$

$\sigma y = sqrt(630/5) = sqrt(126) = 11.225$

And finally, we compute the coefficient of determination (R2):

$R^2 = \{ (1 / N) * \Sigma [(xi - x) * (yi - y)] / (\sigma x * \sigma y) \}^2$ R2 = [(1/5) * 470 / (12.083 * 11.225)]2

R2 = (94 / 135.632)2 = (0.693)2 = 0.48

It gives you an idea of how many data points fall within the results of the line formed by the regression equation. The higher the coefficient, the higher percentage of points the line passes through when the data points and line are plotted. If the coefficient is 0.80, then 80% of the points should fall within the regression line.

THE SUM OF THE SQUARED ERROR (SSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Calculate the regression coefficient and obtain the lines of regression for the following data

X	1	2	3	4	5	6	7
Y	9	8	10	12	11	13	14

Y= 0.929X + 7.284

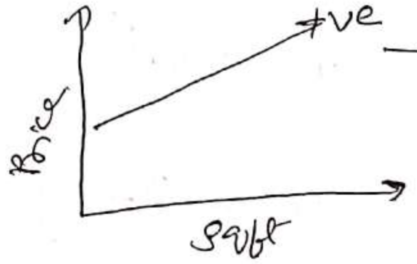
Regression :- which investigates the relationship b/w the independent variable and dependent variable. ≃ Predictive modelling

→ Graphing a line which fits the shape of data.

Eg :- Relationship between the age and income

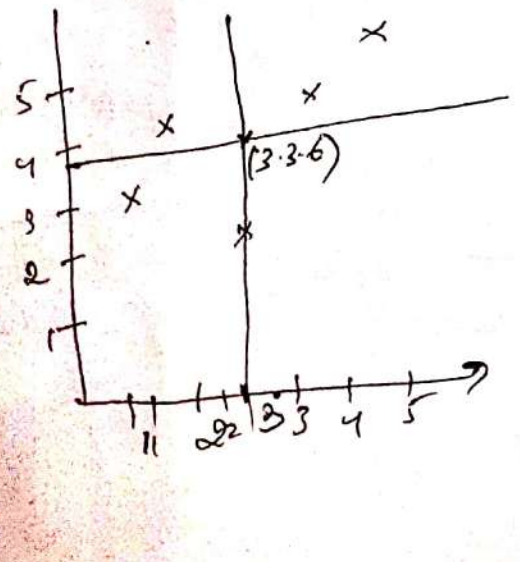
Regression :- how the dependent variable changes with the independent variable.

Eg :- what is the price of bitcoin in next 6 months.



→ $y = mx + c$

x	y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(x-\bar{x})(y-\bar{y})$
1	3	−2	−0.6	4	1.2
2	4	−1	0.4	1	−0.4
3	2	0	−1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.24	4	2.8
mean $\bar{x}$ 3	$3.6 = \bar{y}$			$\Sigma = 10$	$\Sigma = 4$



$m = \dfrac{\Sigma(x-\bar{x})(y-\bar{y})}{(x-\bar{x})^2} = \dfrac{4}{10}$

$m = 0.4$

$c = \bar{y} - m\bar{x}$

$c = 3.6 - 0.4(3) = 2.4$

$y = 0.4x + 2.4$

$= 0.4($

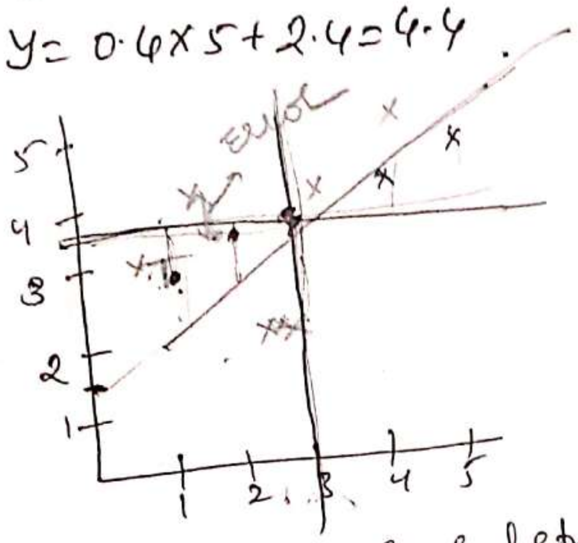
$y = 0.4 \times 1 + 2.4 = 2.8$

$y = 0.4 \times 2 + 2.4 = 3.2$

$y = 0.4 \times 3 + 2.4 = 3.6$

$y = 0.4 \times 4 + 2.4 = 4.0$

$y = 0.4 \times 5 + 2.4 = 4.4$



* Reduce the Error between the Predicted Value and actual Value.
* Line with least Error is the line of regression line and it is the best fit line.

[1
(

Scanned with CamScanner

CONCEPT LEARNING

- Learning involves acquiring general concepts from specific training examples. Example: People continually learn general concepts or categories such as "bird," "car," "situations in which I should study more in order to pass the exam," etc.
- Each such concept can be viewed as describing some subset of objects or events defined over a larger set
- Alternatively, each concept can be thought of as a Boolean-valued function defined over this larger set. (Example: A function defined over all animals, whose value is true for birds and false for other animals).

Definition: Concept learning **-** Inferring a Boolean-valued function from training examples of its input and output

A CONCEPT LEARNING TASK

Consider the example task of learning the target concept "Days on which ***Aldo*** enjoys his favorite water sport"

Example	**Sky**	**AirTemp**	**Humidity**	**Wind**	**Water**	**Forecast**	**EnjoySport**
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Table: Positive and negative training examples for the target concept ***EnjoySport.***

The task is to learn to predict the value of ***EnjoySport*** for an arbitrary day, based on the values of its other attributes?

What hypothesis representation is provided to the learner?

- Let's consider a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.
- Let each hypothesis be a vector of six constraints, specifying the values of the six attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*.

For each attribute, the hypothesis will either

- Indicate by a "?' that any value is acceptable for this attribute,
- Specify a single required value (e.g., Warm) for the attribute, or
- Indicate by a "Φ" that no value is acceptable

If some instance ***x*** satisfies all the constraints of hypothesis ***h***, then ***h*** classifies ***x*** as a positive example (***h(x) = 1***).

The hypothesis that ***PERSON*** enjoys his favorite sport only on cold days with high humidity is represented by the expression

(?, Cold, High, ?, ?, ?)

The most general hypothesis-that every day is a positive example-is represented by

(?, ?, ?, ?, ?, ?)

The most specific possible hypothesis-that no day is a positive example-is represented by

(Φ, Φ, Φ, Φ, Φ, Φ)

Notation

- The set of items over which the concept is defined is called the ***set of instances***, which is denoted by X.

Example: X is the set of all possible days, each represented by the attributes: Sky, AirTemp, Humidity, Wind, Water, and Forecast

- The concept or function to be learned is called the ***target concept***, which is denoted by c.

c can be any Boolean valued function defined over the instances X

$$c: X \rightarrow \{O, 1\}$$

Example: The target concept corresponds to the value of the attribute ***EnjoySport*** (i.e., c(x) = 1 if ***EnjoySport*** = Yes, and c(x) = 0 if ***EnjoySport*** = No).

- Instances for which c(x) = 1 are called ***positive examples***, or members of the target concept.
- Instances for which c(x) = 0 are called ***negative examples***, or non-members of the target concept.
- The ordered pair (x, c(x)) to describe the training example consisting of the instance x and its target ***concept value c(x).***
- ***D*** to denote the set of available training examples

- The symbol ***H*** to denote the set of all possible hypotheses that the learner may consider regarding the identity of the target concept. Each hypothesis ***h*** in ***H*** represents a Boolean-valued function defined over ***X***

$$h: X \rightarrow \{O, 1\}$$

The goal of the learner is to find a hypothesis h such that h(x) = c(x) for all x in X.

- Given:
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values Sunny, Cloudy, and Rainy),
    - *AirTemp* (with values Warm and Cold),
    - *Humidity* (with values Normal and High),
    - *Wind* (with values Strong and Weak),
    - *Water* (with values Warm and Cool),
    - *Forecast* (with values Same and Change).

  - Hypotheses *H*: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Φ" (no value is acceptable), or a specific value.

  - Target concept *c*: ***EnjoySport*** : X → {0, l}
  - 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 X.

Table: The ***EnjoySport*** concept learning task.

<u>The inductive learning hypothesis</u>

Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

CONCEPT LEARNING AS SEARCH

- Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- The goal of this search is to find the hypothesis that best fits the training examples.

Example:
Consider the instances X and hypotheses H in the *EnjoySport* learning task. The attribute Sky has three possible values, and *AirTemp*, *Humidity*, *Wind*, *Water*, *Forecast* each have two possible values, the instance space X contains exactly

3.2.2.2.2.2 = 96 distinct instances

5.4.4.4.4.4 = 5120 syntactically distinct hypotheses within H.

Every hypothesis containing one or more "Φ" symbols represents the empty set of instances; that is, it classifies every instance as negative.

1 + (4.3.3.3.3.3) = 973. Semantically distinct hypotheses

General-to-Specific Ordering of Hypotheses

Consider the two hypotheses

h_1 = (Sunny, ?, ?, Strong, ?, ?)

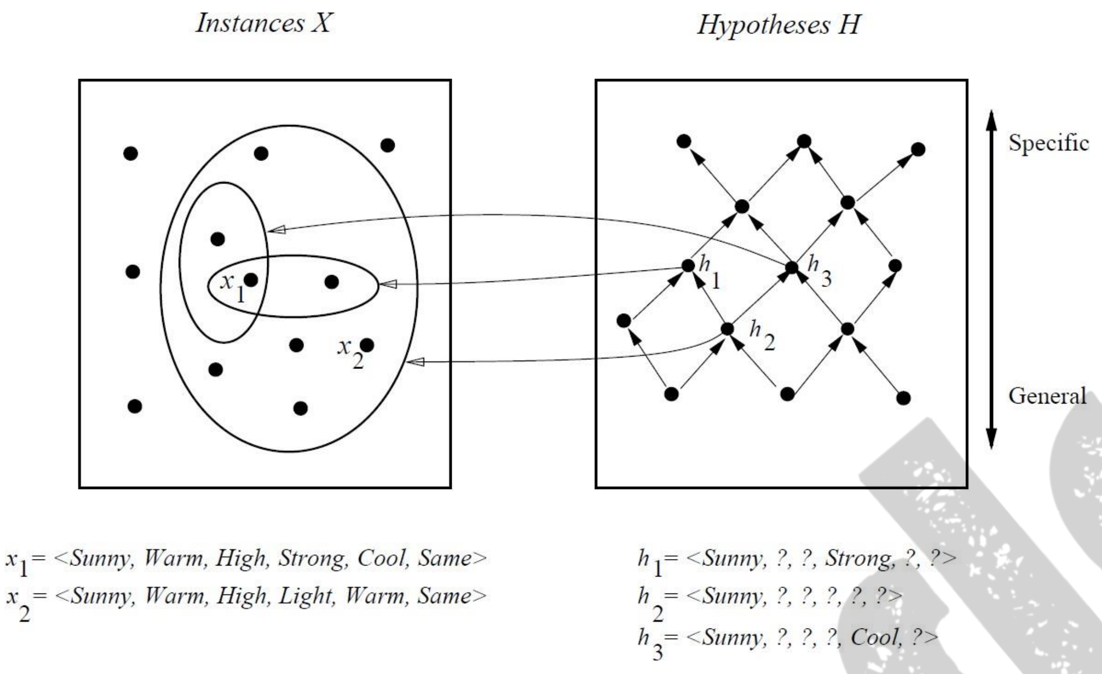
h_2 = (Sunny, ?, ?, ?, ?, ?)

- Consider the sets of instances that are classified positive by h_l and by h_2.
- h_2 imposes fewer constraints on the instance, it classifies more instances as positive. So, any instance classified positive by h_l will also be classified positive by h_2. Therefore, h_2 is more general than h_l.

Given hypotheses h_j and h_k, h_j is more-general-than or- equal do h_k if and only if any instance that satisfies h_k also satisfies h_i

Definition: Let h_j and h_k be Boolean-valued functions defined over X. Then h_j is ***more general-than-or-equal-to*** h_k (written $h_j \geq h_k$) if and only if

$$(\forall\ x \in X\ )\ [(h_k\ (x) = 1) \rightarrow (h_j\ (x) = 1)]$$


Instances X
Hypotheses H
Specific
General
x_1
x_2
h_1
h_2
h_3
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, ?>

- In the figure, the box on the left represents the set X of all instances, the box on the right the set H of all hypotheses.
- Each hypothesis corresponds to some subset of X-the subset of instances that it classifies positive.
- The arrows connecting hypotheses represent the more - general -than relation, with the arrow pointing toward the less general hypothesis.
- Note the subset of instances characterized by h_2 subsumes the subset characterized by h_l , hence h_2 is more - general– than h_1

FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

FIND-S Algorithm

```
1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
      For each attribute constraint $a_i$ in h
         If the constraint $a_i$ is satisfied by x
                  Then do nothing
         Else replace $a_i$ in h by the next more general constraint that is satisfied by x
3. Output hypothesis h
```

To illustrate this algorithm, assume the learner is given the sequence of training examples from the ***EnjoySport*** task

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- The first step of FIND-S is to initialize h to the most specific hypothesis in H

  h - (Ø, Ø, Ø, Ø, Ø, Ø)

- Consider the first training example

  x_1 = <Sunny Warm Normal Strong Warm Same>, +

  Observing the first training example, it is clear that hypothesis *h* is too specific. None of the "Ø" constraints in h are satisfied by this example, so each is replaced by the next *more general constraint* that fits the example

  h_1 = <Sunny Warm Normal Strong Warm Same>

- Consider the second training example

  x_2 = <Sunny, Warm, High, Strong, Warm, Same>, +

  The second training example forces the algorithm to further generalize h, this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example

  h_2 = <Sunny Warm ? Strong Warm Same>

- Consider the third training example

  x3 = <Rainy, Cold, High, Strong, Warm, Change>, -

  Upon encountering the third training the algorithm makes no change to h. The FIND-S algorithm simply ignores every negative example.

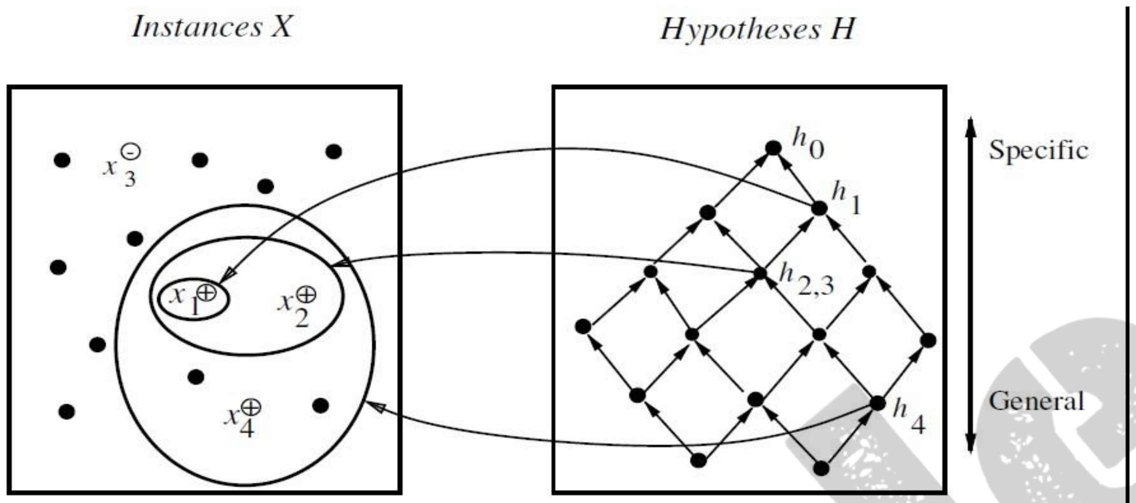
  h_3 = < Sunny Warm ? Strong Warm Same>

- Consider the fourth training example

  x_4 = <Sunny Warm High Strong Cool Change>, +

  The fourth example leads to a further generalization of h

  h_4 = < Sunny Warm ? Strong ? ? >



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>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
h_1 = <Sunny Warm Normal Strong Warm Sam
h_2 = <Sunny Warm ? Strong Warm Same>
h_3 = <Sunny Warm ? Strong Warm Same>
h_4 = <Sunny Warm ? Strong ? ? >

<u>The key property of the FIND-S algorithm</u>

- FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples
- FIND-S algorithm's final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H, and provided the training examples are correct.

<u>Unanswered by FIND-S</u>

1. Has the learner converged to the correct target concept?
2. Why prefer the most specific hypothesis?
3. Are the training examples consistent?
4. What if there are several maximally specific consistent hypotheses?

VERSION SPACES AND THE CANDIDATE-ELIMINATION ALGORITHM

The key idea in the CANDIDATE-ELIMINATION algorithm is to output a description of the set of all ***hypotheses consistent with the training examples***

Representation

Definition: consistent- A hypothesis h is **consistent** with **a** set of training examples ***D*** if **and** only if ***h(x)*** = ***c(x)*** for each example ***(x, c(x))*** in ***D.***

$$Consistent\ (h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

Note difference between definitions of *consistent* and *satisfies*

- An example *x* is said to ***satisfy*** hypothesis ***h*** when ***h(x)*** = 1, regardless of whether *x* is a positive or negative example of the target concept.
- An example *x* is said to ***consistent*** with hypothesis ***h*** iff ***h(x)*** = ***c(x)***

Definition: version space- The **version space,** denoted $VS_{H,D}$ with respect to hypothesis space ***H*** and training examples D, is the subset of hypotheses from ***H*** consistent with the training examples in D

$$VS_{H,D} \equiv \{h \in H \mid Consistent\ (h, D)\}$$

The LIST-THEN-ELIMINATION algorithm

The LIST-THEN-ELIMINATE algorithm first initializes the version space to contain all hypotheses in H and then eliminates any hypothesis found inconsistent with any training example.

```
1. VersionSpace c a list containing every hypothesis in H
2. For each training example, (x, c(x))
        remove from VersionSpace any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in VersionSpace
```

The LIST-THEN-ELIMINATE Algorithm

- List-Then-Eliminate works in principle, so long as version space is finite.
- However, since it requires exhaustive enumeration of all hypotheses in practice it is not feasible.

A More Compact Representation for Version Spaces

The version space is represented by its most general and least general members. These members form general and specific boundary sets that delimit the version space within the partially ordered hypothesis space.

Definition: The **general boundary** G, with respect to hypothesis space ***H*** and training data *D,* is the set of maximally general members of *H* consistent with *D*

$$G \equiv \{g \in H \mid Consistent\ (g, D) \wedge (\neg\exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

Definition: The **specific boundary** S, with respect to hypothesis space *H* and training data *D*, is the set of minimally general (i.e., maximally specific) members of *H* consistent with *D.*

$$S \equiv \{s \in H \mid Consistent\ (s, D) \wedge (\neg\exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

Theorem: Version Space representation theorem

Theorem: Let X be an arbitrary set of instances and Let H be a set of Boolean-valued hypotheses defined over X. Let c: X →{O, 1} be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples {(x, c(x)). For all X, H, c, and D such that S and G are well defined,

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

To Prove:

1. Every h satisfying the right hand side of the above expression is in ***VS*** $_{H,\,D}$

2. Every member of ***VS*** $_{H,\,D}$ satisfies the right-hand side of the expression

Sketch of proof:

1. let g, h, s be arbitrary members of G, H, S respectively with $g \geq_g h \geq_g s$
   - By the definition of ***S,*** **s** must be satisfied by all positive examples in D. Because $h \geq_g s$, h must also be satisfied by all positive examples in D.
   - By the definition of ***G,*** g cannot be satisfied by any negative example in D, and because $g \geq_g h$ h cannot be satisfied by any negative example in D. Because h is satisfied by all positive examples in D and by no negative examples in D, h is consistent with D, and therefore h is a member of $VS_{H,D}$.
2. It can be proven by assuming some h in $VS_{H,D}$,that does not satisfy the right-hand side of the expression, then showing that this leads to an inconsistency

CANDIDATE-ELIMINATION Learning Algorithm

The CANDIDATE-ELIMINTION algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

```
Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do
  • If d is a positive example
      • Remove from G any hypothesis 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 is consistent with d, and some member of G is more general than h
          • Remove from S any hypothesis that is more general than another hypothesis in S

  • If d is a negative example
      • Remove from S any hypothesis 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 is consistent with d, and some member of S is more specific than h
          • Remove from G any hypothesis that is less general than another hypothesis in G
```

CANDIDATE- ELIMINTION algorithm using version spaces

An Illustrative Example

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

CANDIDATE-ELIMINTION algorithm begins by initializing the version space to the set of all hypotheses in H;

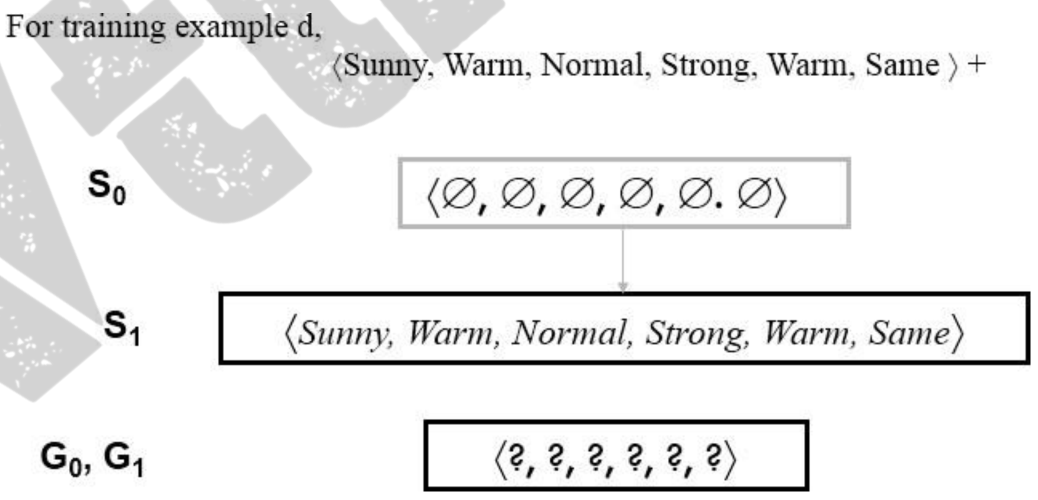
Initializing the G boundary set to contain the most general hypothesis in H

G_0 ⟨?, ?, ?, ?, ?, ?⟩

Initializing the S boundary set to contain the most specific (least general) hypothesis

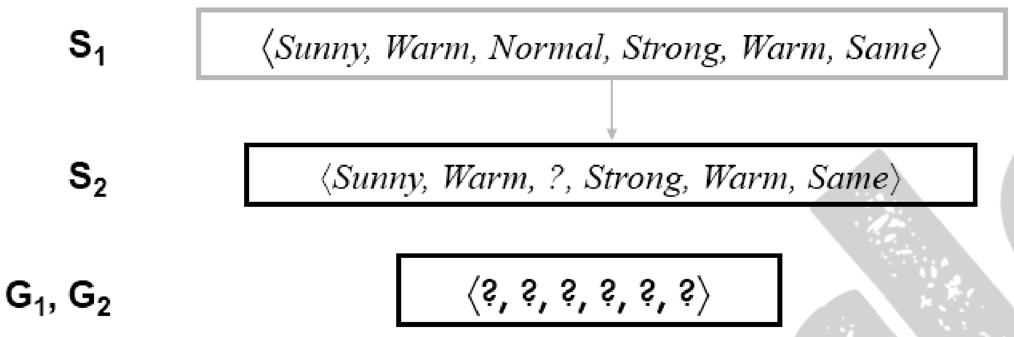
S_0 ⟨∅, ∅, ∅, ∅, ∅, ∅⟩

- <u>When the first training example is presented</u>, the CANDIDATE-ELIMINTION algorithm checks the S boundary and finds that it is overly specific and it fails to cover the positive example.
- The boundary is therefore revised by moving it to the least more general hypothesis that covers this new example
- No update of the G boundary is needed in response to this training example because G_o correctly covers this example


For training example d,
⟨Sunny, Warm, Normal, Strong, Warm, Same⟩ +
S0
⟨∅, ∅, ∅, ∅, ∅. ∅⟩
S1
⟨Sunny, Warm, Normal, Strong, Warm, Same⟩
G0, G1
⟨?, ?, ?, ?, ?, ?⟩

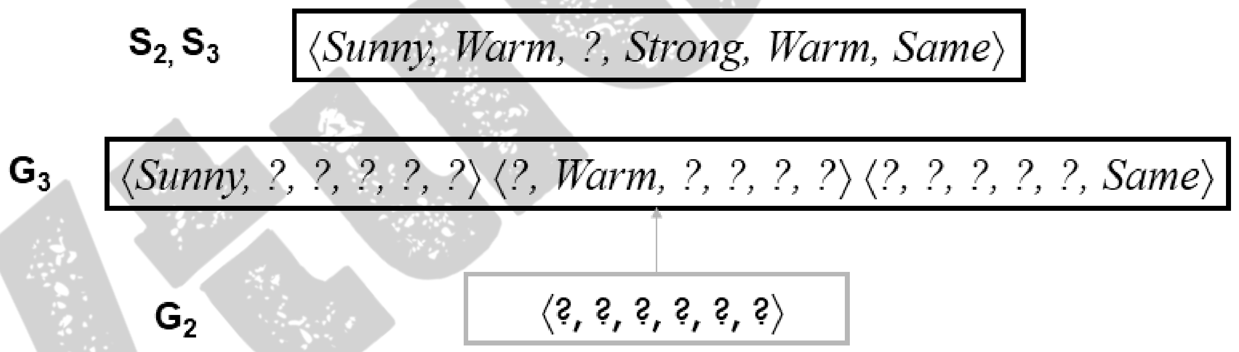
- When the second training example is observed, it has a similar effect of generalizing S further to S_2, leaving G again unchanged i.e., $G_2 = G_1 = G_0$

For training example d,
⟨Sunny, Warm, High, Strong, Warm, Same⟩ +


S_1
⟨Sunny, Warm, Normal, Strong, Warm, Same⟩
S_2
⟨Sunny, Warm, ?, Strong, Warm, Same⟩
G_1, G_2
⟨?, ?, ?, ?, ?, ?⟩

- Consider the third training example. This negative example reveals that the G boundary of the version space is overly general, that is, the hypothesis in G incorrectly predicts that this new example is a positive example.
- The hypothesis in the G boundary must therefore be specialized until it correctly classifies this new negative example

For training example d,
⟨Rainy, Cold, High, Strong, Warm, Change ⟩ –

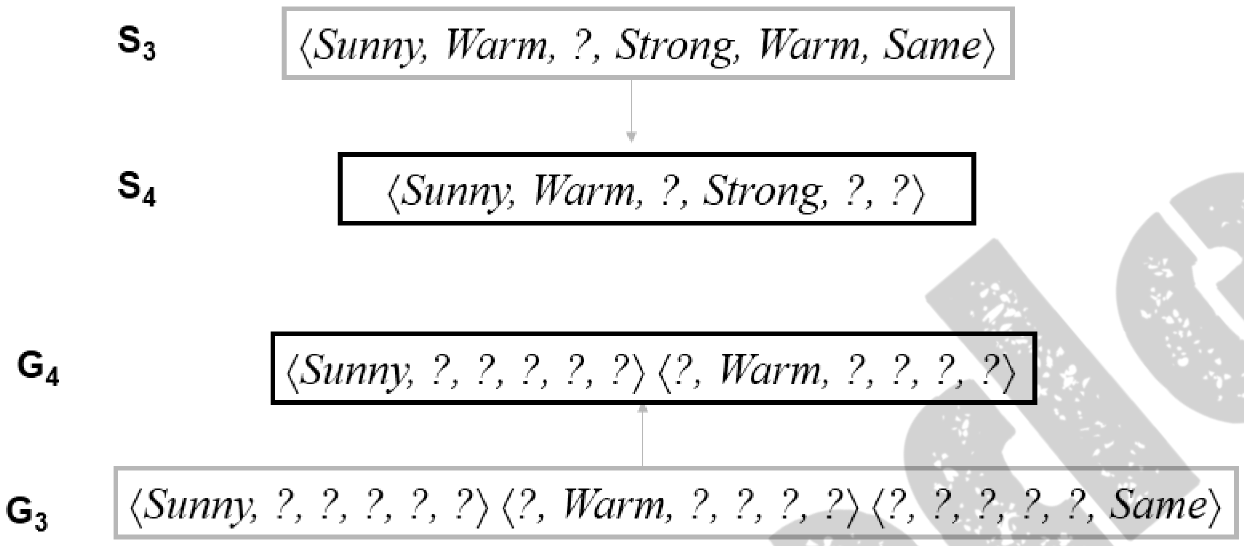

S_2, S_3
⟨Sunny, Warm, ?, Strong, Warm, Same⟩
G_3
⟨Sunny, ?, ?, ?, ?, ?⟩ ⟨?, Warm, ?, ?, ?, ?⟩ ⟨?, ?, ?, ?, ?, Same⟩
G_2
⟨?, ?, ?, ?, ?, ?⟩

Given that there are six attributes that could be specified to specialize G_2, why are there only three new hypotheses in G_3?

For example, the hypothesis h = (?, ?, Normal, ?, ?, ?) is a minimal specialization of G_2 that correctly labels the new example as a negative example, but it is not included in G_3. The reason this hypothesis is excluded is that it is inconsistent with the previously encountered positive examples

- <u>Consider the fourth training example</u>.

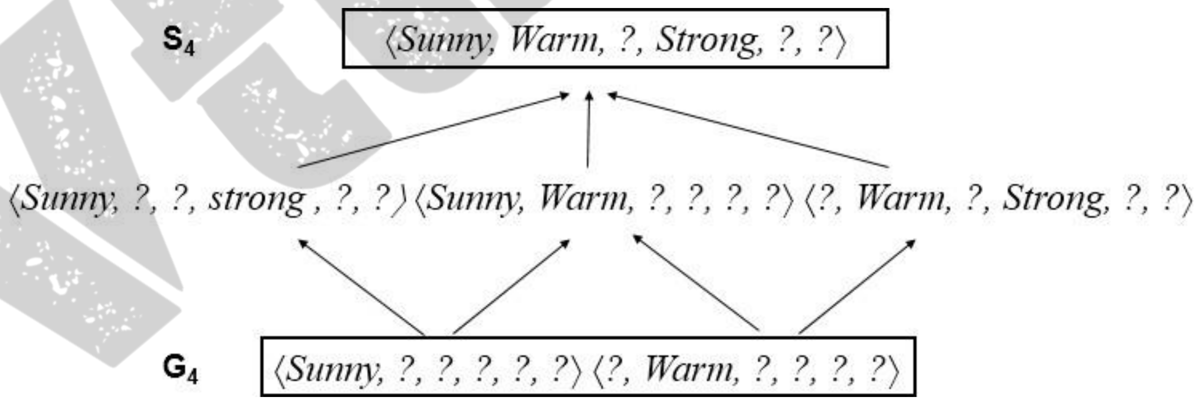
For training example d,
⟨Sunny, Warm, High, Strong, Cool Change ⟩ +



S_3
⟨Sunny, Warm, ?, Strong, Warm, Same⟩
S_4
⟨Sunny, Warm, ?, Strong, ?, ?⟩
G_4
⟨Sunny, ?, ?, ?, ?, ?⟩ ⟨?, Warm, ?, ?, ?, ?⟩
G_3
⟨Sunny, ?, ?, ?, ?, ?⟩ ⟨?, Warm, ?, ?, ?, ?⟩ ⟨?, ?, ?, ?, ?, Same⟩

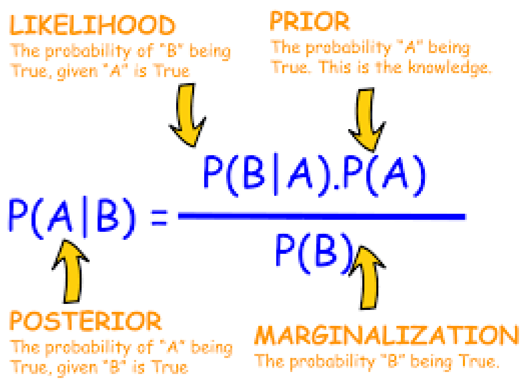
- This positive example further generalizes the S boundary of the version space. It also results in removing one member of the G boundary, because this member fails to cover the new positive example

After processing these four examples, the boundary sets S_4 and G_4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples.



S_4
⟨Sunny, Warm, ?, Strong, ?, ?⟩
⟨Sunny, ?, ?, strong , ?, ?⟩ ⟨Sunny, Warm, ?, ?, ?, ?⟩ ⟨?, Warm, ?, Strong, ?, ?⟩
G_4
⟨Sunny, ?, ?, ?, ?, ?⟩ ⟨?, Warm, ?, ?, ?, ?⟩

Bayes' Theorem states that the conditional probability of an event, based on the occurrence of another event, is equal to the likelihood of the second event given the first event multiplied by the probability of the first event.


LIKELIHOOD
The probability of "B" being True, given "A" is True
PRIOR
The probability "A" being True. This is the knowledge.
P(A|B) = P(B|A).P(A) / P(B)
POSTERIOR
The probability of "A" being True, given "B" is True
MARGINALIZATION
The probability "B" being True.

P(Cancer/Symptoms) = <u>P(Cancer) P(Symptons/Cancer)</u>

P(Symptons)

: Patient has liver disease." Past data tells you that 10% of patients entering your clinic have liver disease. Patient is an alcoholic." Five percent of the clinic's patients are alcoholics. You might also know that among those patients diagnosed with liver disease, 7% are alcoholics. What is the chances of having a liver disease if the patient is alcoholic?

- **A** could mean the event "Patient has liver disease." Past data tells you that 10% of patients entering your clinic have liver disease. P(A) = 0.10.
- **B** could mean the litmus test that "Patient is an alcoholic." Five percent of the clinic's patients are alcoholics. P(B) = 0.05.
- You might also know that among those patients diagnosed with liver disease, 7% are alcoholics. This is your **B|A:** the probability that a patient is alcoholic, given that they have liver disease, is 7%.

Bayes' theorem tells you:
P(A|B) = (0.07 * 0.1)/0.05 = 0.14

We will use Rain to mean rain during the day,

- (Rain) is Probability of Rain = 10%
- P(Cloud|Rain) is Probability of Cloud, given that Rain happens = 50%
- P(Cloud) is Probability of Cloud = 40%

P(Rain|Cloud) = *0.1 x 0.5/* **0.4** = .125

Parametric classification

Naive Bayes classifier

Let D be a set of tuples where each tuple ~~[illegible]~~ is an n-dimentional attribute vector given by $x:(x_1 \dots x_n)$. let there be m classes: $c_1, c_2, c_3 \dots c_m$.
Naive Bayes classifier predicts X belongs to class c_i if.

$$P(c_i/x) > P(c_j/x) \text{ for } 1 \leq j \leq m.$$

Maximum posteriori hypothesis is given by

$$P(c_i/x) = \frac{P(x/c_i) P(c_i)}{P(x)}$$

or

P(x) is a normalizing term which allows us to calculate the probability. we can despite when we are interested in the most probable hypothesis as it is constant and only used to normalize.

$$MAP(h) = \overset{arg}{\max}\left(P(x/c_i)\ P(c_i)\right)$$

↓

Maximum a posteriori hypothesis.

Naive Bayes is a classification algorithm for two class and multiclass classification problems.

List of probability.

1) class probabilities :- Probabilities of each class in the training dataset

2) conditional probabilities :- The conditional probabilities of each input value given each class value.

Scanned by CamScanner

Example of Naive Bayes classifier

TO carry out classification from a dataset using Naive Bayes classifier.

Dataset which gives the class of whether a person buys a computer or not

Sl.NO	Age	Income	Student	Credit	Buy
1.	<30	High	No	Fair	NO
2.	<30	High	No	Excellent	NO
3.	31-40	High	No	Fair	Yes
4.	>40	Medium	NO	Fair	Yes
5.	>40	low	Yes	~~Excellent~~ Fair	~~No~~ Yes
6	>40	low	Yes	Excellent	~~Yes~~ NO
7	31-40	low	Yes	Excellent	Yes
8	<30	Med.	NO	Fair	NO
9	<30	low	Yes	Fair	Yes
10	>40	Med	Yes	Fair	Yes
11	<30	med	Yes	Exc	Yes
12	31-40	med	NO	Exce	Yes
13	31-40	High	Yes	Fair	Yes
14	>40	Med	NO	Exc	NO

Two classes → buys computer - Yes
buys computer - NO

Tuple we wish to classify:

X = (age-youth, Income = medium, student = Yes, Creditrating = fair)

Scanned by CamScanner

Step1 :

we need to maximize $P(x/c_i)\,P(C_i)$ for $i=1,2$
The prior probability of each class $P(C_i)$ can be computed based on training tuples.

$P(\text{buys-computer} = \text{Yes}) = 9/14 = 0.643$

$P(\text{buys-computer} = \text{No}) = 5/14 = 0.357$

Step2:

TO compute $P(x/c_i)$ for $i=1,2$ we need to Compute Conditional probabilities.

$P(\text{age}=\text{Youth} \mid \text{buys-computer}=\text{Yes}) = \frac{2}{9} = 0.222$

$P(\text{age}=\text{Youth} \mid \text{buys-computer}=\text{No}) = 3/5 = 0.600$

P(~~age~~ In = med | buys-computer = Yes) = 4/9 = 0.444

P(~~age~~ In = med | buys-computer = NO) = 2/5 = 0.400

$P(\text{student}=\text{Yes} \mid \text{buys-computer}=\text{Yes}) = 6/9 = 0.667$

$P(\text{student}=\text{Yes} \mid \text{buys computer}=\text{No}) = 1/5 = 0.200$

$P(\text{Credit rating}=\text{fair} \mid \text{buyscomp}=\text{Yes}) = 6/9 = 0.667.$

$P(\text{credit-rating}=\text{fair} \mid \text{buyscomp}=\text{No}) = 2/5 = 0.400$

Step3:

$P(x \mid \text{buys-computer}=\text{Yes}) = P(\text{age}=\text{youth} \mid \text{buyscomp}=\text{Yes})$
$\times P(\text{income}=\text{med} \mid \text{buys comp}=\text{Yes})$
$\times P(\text{student}=\text{Yes} \mid \text{buyscom}=\text{Yes})$
$\times P(\text{cred}=\text{fair} \mid \text{buyscomp}=\text{Yes})$

= 0.222 X 0.444 X 0.667 X0.667 = ~~0.444~~ = 0.043

$P(x \mid \text{buyscomp}=\text{NO}) = 0.600 \times 0.400 \times 0.200 \times .400$
$= 0.019$

To find class C_i that maximizes $P(x/C_i)$ we compute

$P(x/\text{buys comput}=\text{Yes})\ P(\text{buyscomp}=\text{Yes}) = 0.044 \times 0.643 = 0.028$

$P(x/\text{buys comp}=\text{No})\ P(\text{buyscomp}=\text{No}) = 0.019 \times 0.357 = 0.007$

Naive Bayes classifier Predicts buyscomputer = Yes for tuple x.

~~Naive~~ Gaussian Naive Bayes

Gaussian Normal distribution → need to estimate the mean and standard deviation from training data

Above we calculated the probabilities for the input values for each class using frequency. we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution..

→ with probabilities of each class, we must also store the mean & standard deviation for each input variable for each class.

Calculating mean and standard deviation values of each input variable (x) for each class value

$$\underset{\mu_i}{\text{mean}(x)} = \frac{\text{sum}(x_i)}{n}.$$

$i \Rightarrow \underset{\text{class}_1}{\underset{J_1}{1}}, \underset{\text{class}_2}{\underset{J_1}{2}}$

$$\underset{\sigma_i}{\text{Standard deviation}} = \sqrt{\frac{\Sigma(x_i - m(x))^2}{N}}$$

Probability of new x values are calculated using

New PDF

$$\boxed{\text{Pdf}(x, m, \sigma) = f(x)} = \frac{1}{\sqrt{2\pi}\cdot\sigma_i} \times \exp\left[\frac{-(x-\mu_i)^2}{2\sigma_i^2}\right]$$

~~Pdf~~$(x, \text{mean}, \sigma) = $