

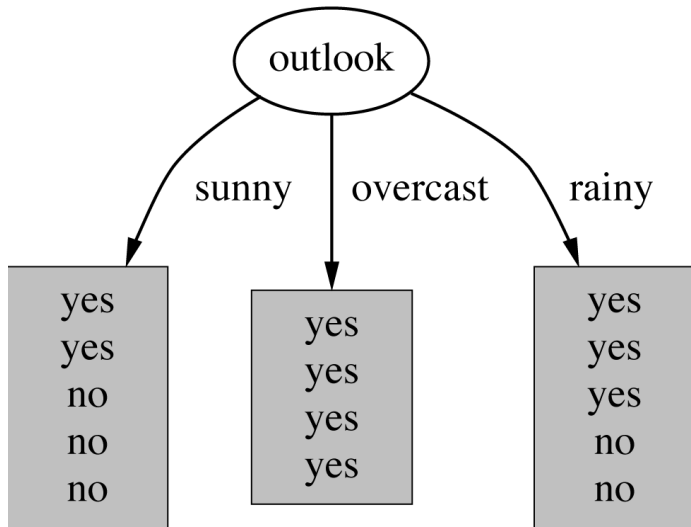
# Midterm Review

# Decision Trees

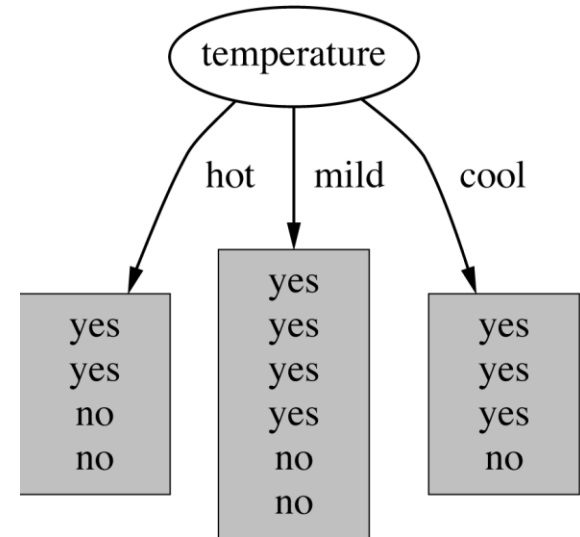
# Weather data

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

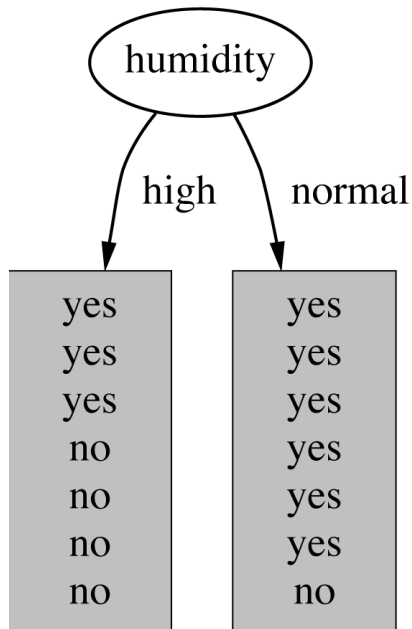
# Which attribute to select?



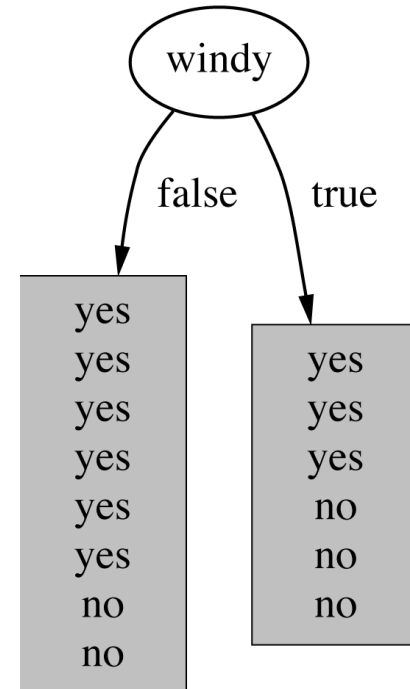
(a)



(b)



(c)



(d)

# Attribute “Outlook”

outlook=sunny

$$\text{entropy}(2/5, 3/5) = -2/5 * \log(2/5) - 3/5 * \log(3/5) = .971$$

outlook=overcast

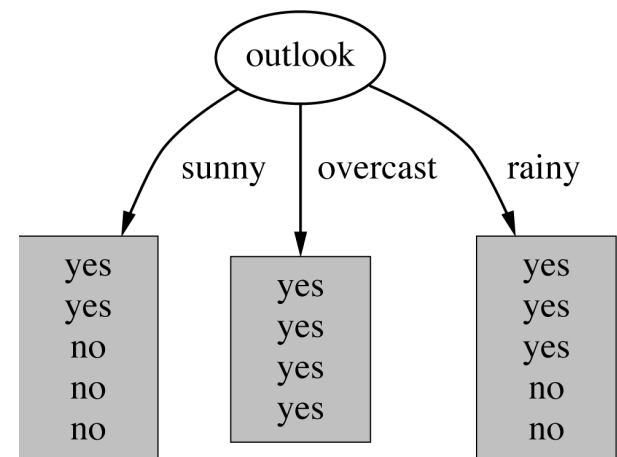
$$\text{entropy}(4/4, 0/4) = -1 * \log(1) - 0 * \log(0) = 0$$

outlook=rainy

$$\text{entropy}(3/5, 2/5) = -3/5 * \log(3/5) - 2/5 * \log(2/5) = .971$$

**Expected info:**

$$AE = .971 * (5/14) + 0 * (4/14) + .971 * (5/14) = \mathbf{.693}$$



# Attribute “Temperature”

temperature=hot

$$\text{entropy}(2/4, 2/4) = -2/4 * \log(2/4) - 2/4 * \log(2/4) = 1$$

temperature=mild

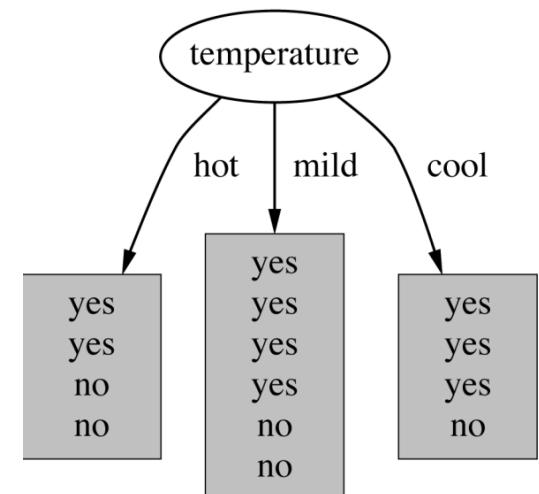
$$\text{entropy}(4/6, 2/6) = -4/6 * \log(1) - 2/6 * \log(2/6) = .528$$

temperature=cool

$$\text{entropy}(3/4, 1/4) = -3/4 * \log(3/4) - 1/4 * \log(1/4) = .811$$

**Expected info:**

$$AE = 1 * (4/14) + .528 * (6/14) + .811 * (4/14) = \mathbf{.744}$$



# Attribute “Humidity”

humidity=high

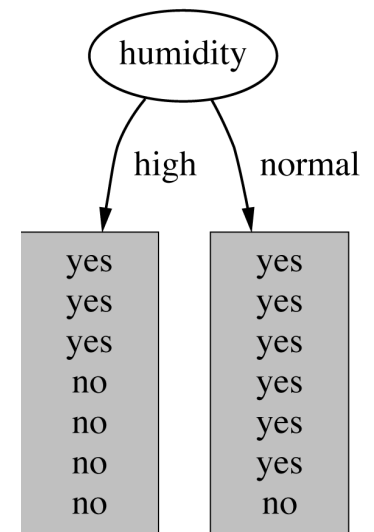
$$\text{entropy}(3/7, 4/7) = -3/7 * \log(3/7) - 4/7 * \log(4/7) = .985$$

humidity=normal

$$\text{entropy}(6/7, 1/7) = -6/7 * \log(6/7) - 1/7 * \log(1/7) = .592$$

**Expected info:**

$$AE = .985 * (7/14) + .592 * (7/14) = .788$$



# Attribute “Windy”

windy=false

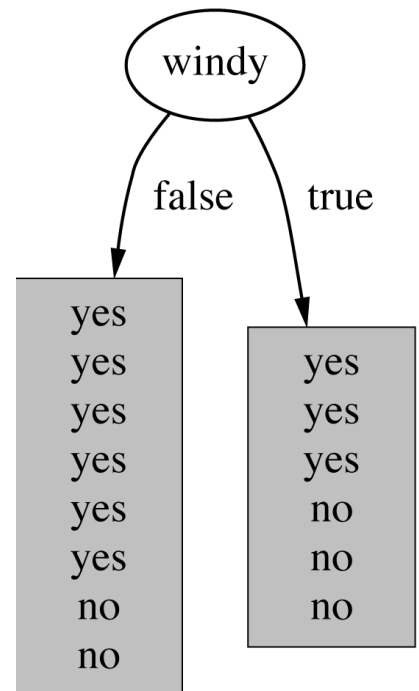
$$\text{entropy}(6/8, 2/8) = -6/8 * \log(6/8) - 2/8 * \log(2/8) = .811$$

humidity=true

$$\text{entropy}(3/6, 3/6) = -3/6 * \log(3/6) - 3/6 * \log(3/6) = 1$$

**Expected info:**

$$AE = .811 * (8/14) + 1 * (6/14) = .892$$





# And the winner is...

"Outlook"

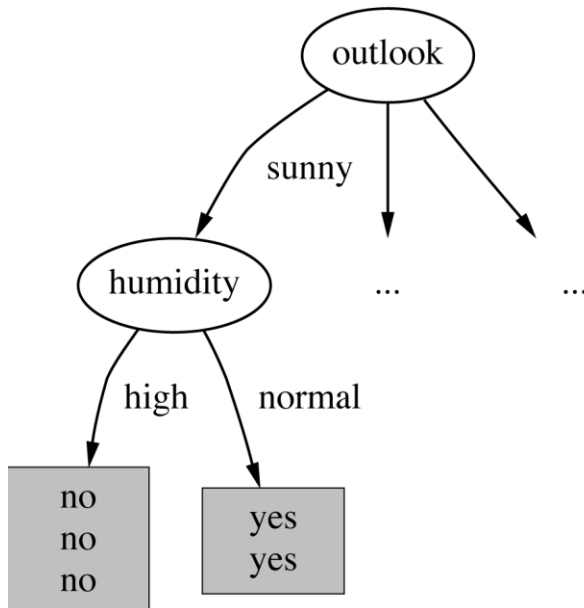
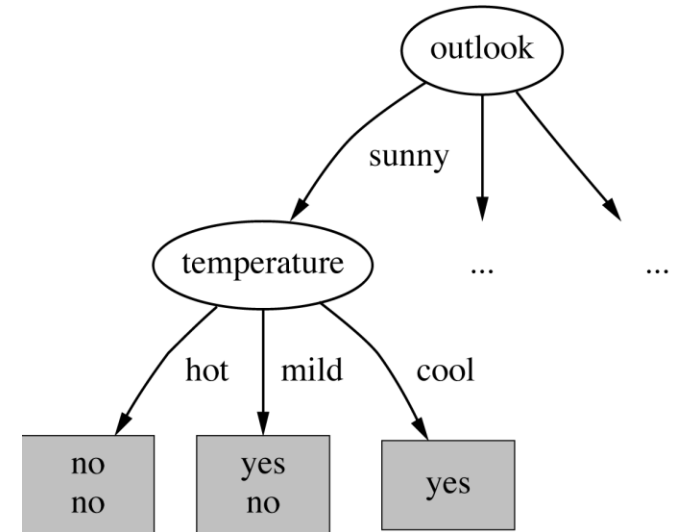
...So, the root will be "Outlook"



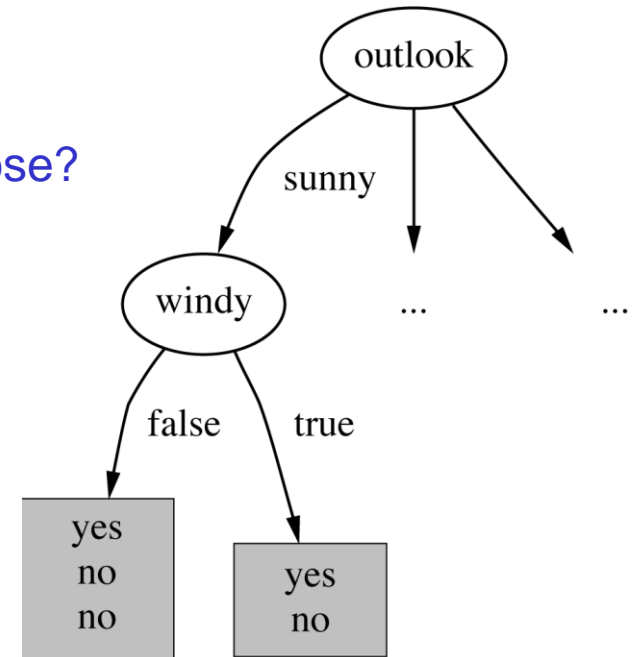
Outlook

# Continuing to split (for Outlook="Sunny")

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes



Which one to choose?



# SLIQ

- SLIQ is a decision tree classifier that can handle both **numerical** and **categorical** attributes
- Uses a **pre-sorting technique** in the **tree growing** phase
- Suitable for classification of **large disk-resident** datasets

# Some Data

rid	age	salary	marital	car
1	30	60	single	sports
2	25	20	single	mini
3	40	80	married	van
4	45	100	single	luxury
5	60	150	married	luxury
6	35	120	single	sports
7	50	70	married	van
8	55	90	single	sports
9	65	30	married	mini
10	70	200	single	luxury

# SLIQ - Attribute Lists

rid	age
1	30
2	25
3	40
4	45
5	60
6	35
7	50
8	55
9	65
10	70

rid	salary
1	60
2	20
3	80
4	100
5	150
6	120
7	70
8	90
9	30
10	200

rid	marital
1	single
2	single
3	married
4	single
5	married
6	single
7	married
8	single
9	married
10	single

These are projections on (rid, attribute).

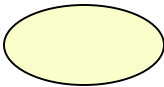
# SLIQ - Sort Numeric, Group Categorical

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	marital
3	married
5	married
7	married
9	married
1	single
2	single
4	single
6	single
8	single
10	single

# SLIQ - Class List

N1 

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

# SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

age ≤ 25 ?

	sports	mini	van	luxury
L				
R				

age ≤ 30 ?

	sports	mini	van	luxury
L				
R				

Evaluate each split,  
using Entropy or GINI.

...



# SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

age ≤ 25

	sports	mini	van	luxury
L	0	1	0	0
R	3	1	2	3

age ≤ 30

	sports	mini	van	luxury
L	1	1	0	0
R	2	1	2	3


Evaluate each split,  
using Entropy or GINI.

...

# SLIQ - Histograms

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

salary ≤ 20

	sports	mini	van	luxury
L	0	1	0	0
R	3	1	2	3

salary ≤ 30

	sports	mini	van	luxury
L	0	2	0	0
R	3	0	2	3

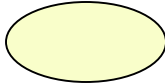
Evaluate each split,  
using Entropy or GINI.

...

# SLIQ - Histograms

rid	marital
3	married
5	married
7	married
9	married
1	single
2	single
4	single
6	single
8	single
10	single

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

	sports	mini	van	luxury
Married				
Single				

Evaluate each split,  
using Entropy or GINI.

# SLIQ - Histograms

rid	marital
3	married
5	married
7	married
9	married
1	single
2	single
4	single
6	single
8	single
10	single

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

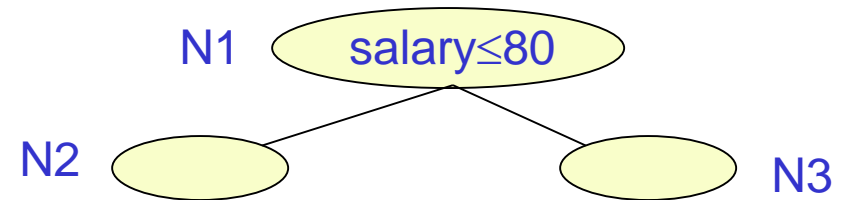
	sports	mini	van	luxury
Married	0	1	2	1
Single	3	1	0	2

Evaluate each split,  
using Entropy or GINI.

# SLIQ - Perform split(s)

Suppose  $\text{salary} \leq 80$   
has smallest entropy

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

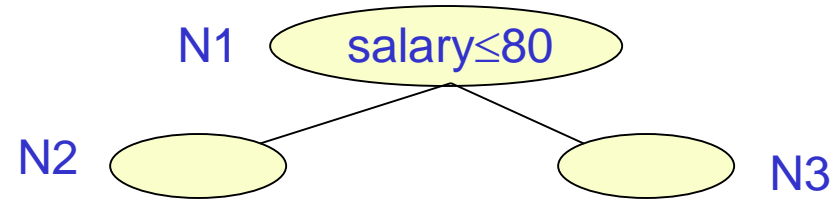


# SLIQ - Perform split(s)

Suppose  $\text{salary} \leq 80$   
has smallest entropy

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1



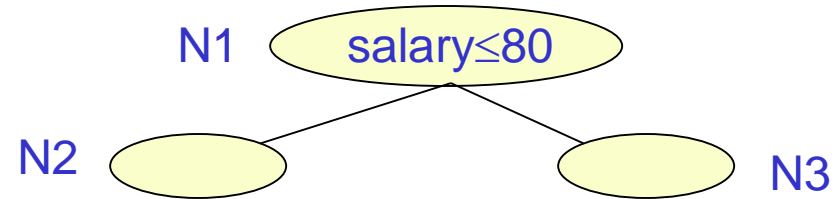
Read **salary** list again, update **class** list  
Use **rid** to jump from salary list to class list.

# SLIQ - Update Class List

Suppose  $\text{salary} \leq 80$   
has smallest entropy

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	car	LEAF
1	sports	N2
2	mini	N2
3	van	N2
4	luxury	N3
5	luxury	N3
6	sports	N3
7	van	N2
8	sports	N3
9	mini	N2
10	luxury	N3



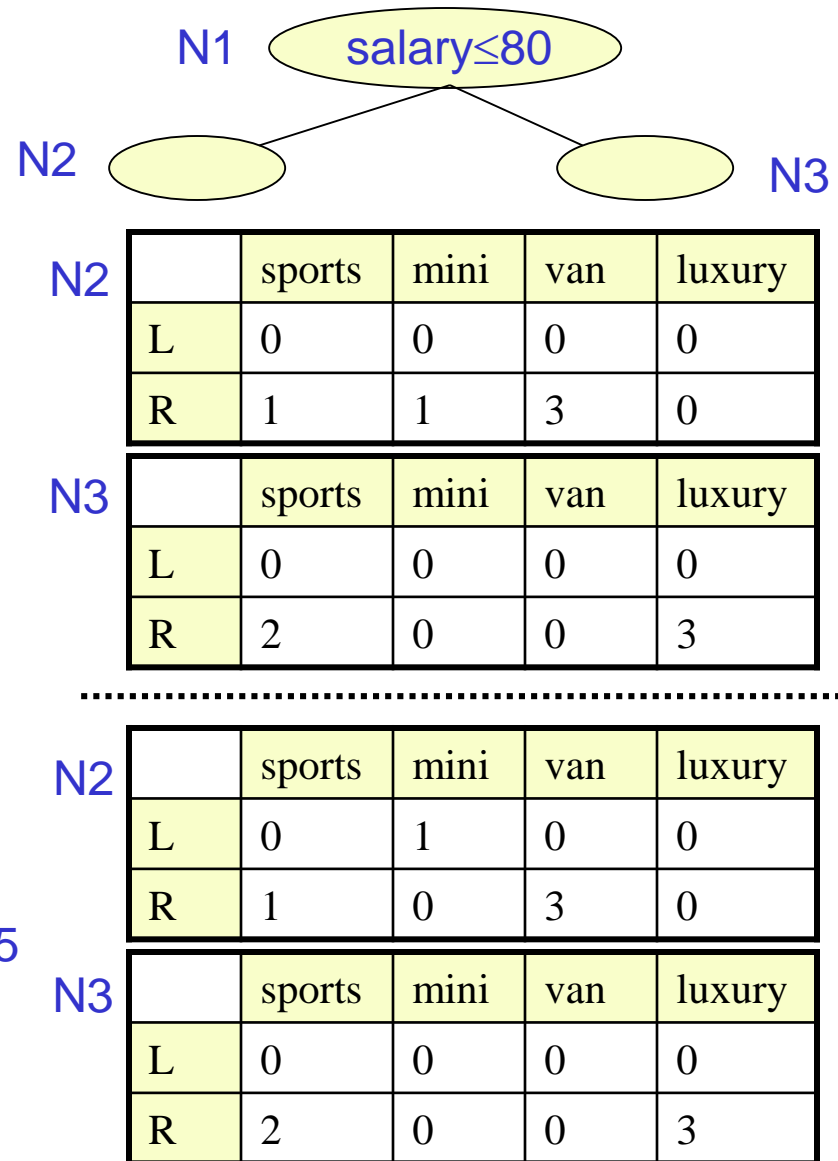
Read **salary list** again, update **class list**  
Use **rid** to jump from salary list to class list.

# SLIQ – Continuing to the next level

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N2
2	mini	N2
3	van	N3
4	luxury	N3
5	luxury	N3
6	sports	N3
7	van	N3
8	sports	N3
9	mini	N2
10	luxury	N3

Evaluate each split,  
using GINI or Entropy.



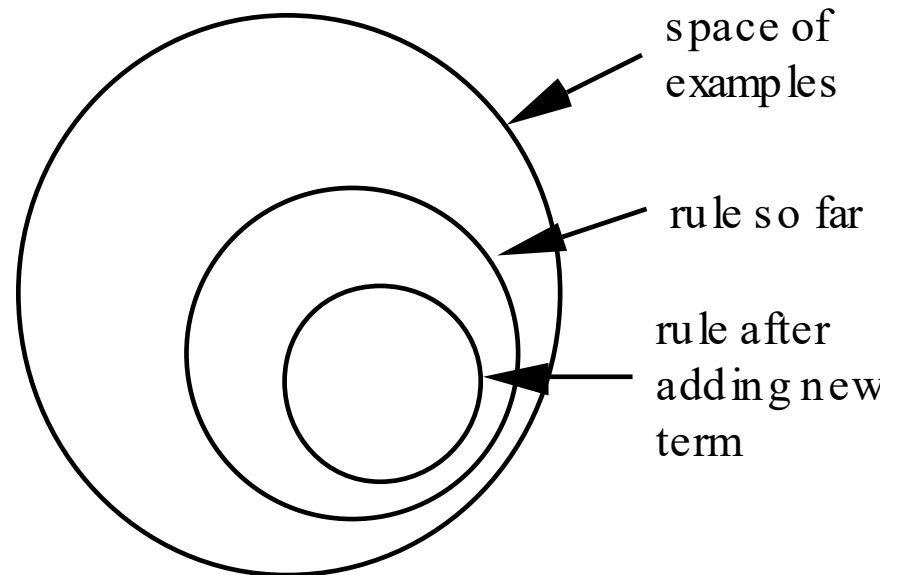
age ≤ 25



# Rule-Based Classifiers

# A simple covering algorithm

- Generates a rule by adding tests that maximize rule's accuracy
- Each new test (growing the rule) reduces rule's coverage, but it **increases the rule accuracy**.



# Example: contact lenses data

age	spectacle-prescrip	astigmatism	tear-prod-rate	contact-lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

# Example: contact lenses data

❖ Rule we seek:

```
If ?  
    then recommendation = hard
```

❖ Possible tests:

Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

# Modified rule and resulting data

## ❖ Rule with best test added:

```
If astigmatism = yes  
    then recommendation = hard
```

## ❖ Instances covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

The rule isn't very accurate, getting only 4 out of 12 that it covers. So, it needs further refinement. Repeat on covered instances.

# Naïve Bayes Classifier

# The weather data example

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

← *Evidence E*

$$\begin{aligned}
 &P(\text{Play}=\text{yes} \mid E) = \\
 &\quad P(\text{Outlook}=\text{Sunny} \mid \text{play}=\text{yes}) * \\
 &\quad P(\text{Temp}=\text{Cool} \mid \text{play}=\text{yes}) * \\
 &\quad P(\text{Humidity}=\text{High} \mid \text{play}=\text{yes}) * \\
 &\quad P(\text{Windy}=\text{True} \mid \text{play}=\text{yes}) * \\
 &\quad P(\text{play}=\text{yes}) / P(E) = \\
 = &\quad (2/9) * \\
 &\quad (3/9) * \\
 &\quad (3/9) * \\
 &\quad (3/9) * \\
 &\quad (9/14) / P(E) = 0.0053 / P(E)
 \end{aligned}$$

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

# The weather data example

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

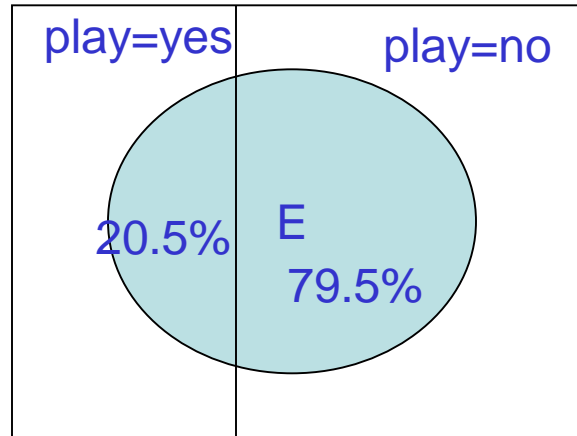
← *Evidence E*

$$\begin{aligned}
 &P(\text{Play=no} \mid E) = \\
 &\quad P(\text{Outlook=Sunny} \mid \text{play=no}) * \\
 &\quad P(\text{Temp=Cool} \mid \text{play=no}) * \\
 &\quad P(\text{Humidity=High} \mid \text{play=no}) * \\
 &\quad P(\text{Windy=True} \mid \text{play=no}) * \\
 &\quad P(\text{play=no}) / P(E) = \\
 &= (3/5) * \\
 &\quad (1/5) * \\
 &\quad (4/5) * \\
 &\quad (3/5) * \\
 &\quad (5/14) / P(E) = 0.0206 / P(E)
 \end{aligned}$$

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



# Normalization constant



$$P(\text{play=yes} \mid E) + P(\text{play=no} \mid E) = 1 \quad \text{i.e.}$$

$$0.0053 / P(E) + 0.0206 / P(E) = 1 \quad \text{i.e.}$$

$$P(E) = 0.0053 + 0.0206$$

So,

$$P(\text{play=yes} \mid E) = 0.0053 / (0.0053 + 0.0206) = \mathbf{20.5\%}$$

$$P(\text{play=no} \mid E) = 0.0206 / (0.0053 + 0.0206) = \mathbf{79.5\%}$$

# “zero-frequency problem” – Laplace

$$P(\text{play}=\text{yes} \mid E) =$$

$$P(\text{Outlook}=\text{Sunny} \mid \text{play}=\text{yes}) *$$

$$P(\text{Temp}=\text{Cool} \mid \text{play}=\text{yes}) *$$

$$P(\text{Humidity}=\text{High} \mid \text{play}=\text{yes}) *$$

$$P(\text{Windy}=\text{True} \mid \text{play}=\text{yes}) *$$

$$P(\text{play}=\text{yes}) / P(E) =$$

$$= (2/9) * (3/9) * (3/9) * (3/9) * (9/14) / P(E) =$$
$$0.0053 / P(E)$$

It will be instead:

Number of possible  
values for 'Outlook'

$$= ((2+1)/(9+3*1)) * ((3+1)/(9+3*1)) *$$
$$((3+1)/(9+2*1)) * ((3+1)/(9+2*1))$$
$$*((9+1)/(14+2*1)) / P(E) = 0.0069 / P(E)$$

We do Laplace always,  
not only when we see  
that a fraction is 0.

# Naïve Bayes for Text Classification

# Bernoulli Model

$P(c)$  : fraction of training documents that are of class  $c$ .

$$P(c) = \frac{N_c}{N}$$

$P(t|c)$  : fraction of documents of class  $c$  that contain term  $t$ .

$$P(t|c) = \frac{N_{c,t}}{N_c}$$

Avoid zero frequency by:

$$P(t|c) = \frac{N_{c,t} + 1}{N_c + 2}$$

Classify a new document  $d$  by computing for each  $c$

$$P(c | d) = \alpha * P(c) * \prod_{t \in d} P(t | c) * \prod_{t \notin d} (1 - P(t | c))$$

Produce as classification result:  $c_{map} = \operatorname{argmax}_c P(c|d)$

# Multinomial Model

$P(c)$  : fraction of training documents that are of class  $c$ .

$$P(c) = \frac{N_c}{N}$$

$P(t|c)$  : relative frequency of term  $t$  in documents of class  $c$ .

$$P(t|c) = \frac{T_{c,t}}{\sum_{t \in V} T_{c,t}}$$

$V$  : vocabulary

$T_{c,t}$  : number of occurrences of  $t$  in training documents of class  $c$ , including multiple occurrences of a term in a document.

Avoid zero frequency by:  $P(t|c) = \frac{T_{c,t}+1}{\sum_{t \in V} (T_{c,t}+1)} = \frac{T_{c,t}+1}{(\sum_{t \in V} T_{c,t}) + |V|}$

Classify a new document  $d$  by computing for each  $c$

$$P(c | d) = \alpha * P(c) * \prod_{t \in d} P(t | c)$$

Produce as classification result:  $c_{map} = \operatorname{argmax}_c P(c|d)$

# Bernoulli Model Example

	docID	words in document	in $c = \text{China?}$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

**Priors:**  $P(c) = 3/4$  and  $P(\bar{c}) = 1/4$

**Conditional probabilities:**

$$P(\text{Chinese}|c) = (3 + 1)/(3 + 2) = 4/5$$

$$P(\text{Japan}|c) = P(\text{Tokyo}|c) = (0 + 1)/(3 + 2) = 1/5$$

$$P(\text{Beijing}|c) = P(\text{Macao}|c) = P(\text{Shanghai}|c) = (1 + 1)/(3 + 2) = 2/5$$

$$P(\text{Chinese}|\bar{c}) = (1 + 1)/(1 + 2) = 2/3$$

$$P(\text{Japan}|\bar{c}) = P(\text{Tokyo}|\bar{c}) = (1 + 1)/(1 + 2) = 2/3$$

$$P(\text{Beijing}|\bar{c}) = P(\text{Macao}|\bar{c}) = P(\text{Shanghai}|\bar{c}) = (0 + 1)/(1 + 2) = 1/3$$

Denominators are  $(3 + 2)$  and  $(1 + 2)$  because there are three documents in  $c$  and one document in  $\bar{c}$  and because the constant we add is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

Then, we get:

$$P(c|d_5) \propto P(c) \cdot P(\text{Chinese}|c) \cdot P(\text{Japan}|c) \cdot P(\text{Tokyo}|c) \cdot (1 - P(\text{Beijing}|c)) \cdot (1 - P(\text{Shanghai}|c)) \cdot (1 - P(\text{Macao}|c)) = 3/4 \cdot 4/5 \cdot 1/5 \cdot 1/5 \cdot (1 - 2/5) \cdot (1 - 2/5) \cdot (1 - 2/5) \approx 0.005$$

$$P(\bar{c}|d_5) \propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1 - 1/3) \cdot (1 - 1/3) \cdot (1 - 1/3) \approx 0.022$$

Classifier assigns the test document to  $\bar{c} = \text{not-China}$ .

When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for  $\bar{c}$  ( $2/3 > 1/5$ ) and the conditional probabilities of Chinese for  $c$  and  $\bar{c}$  are not different enough ( $4/5$  vs.  $2/3$ ) to affect the classification decision.

# Multinomial Model Example

	docID	words in document	in $c = \textit{China}$ ?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

**Priors:**  $P(c) = 3/4$  and  $P(\bar{c}) = 1/4$

**Conditional probabilities:**

$$P(\textit{Chinese}|c) = (5 + 1)/(8 + 6) = 6/14 = 3/7$$

$$P(\textit{Tokyo}|\bar{c}) = P(\textit{Japan}|\bar{c}) = (0 + 1)/(3 + 6) = 1/9$$

$$P(\textit{Chinese}|\bar{c}) = (1 + 1)/(3 + 6) = 2/9$$

$$P(\textit{Tokyo}|c) = P(\textit{Japan}|c) = (1 + 1)/(8 + 6) = 2/14 = 1/7$$

Denominators are  $(8 + 6)$  and  $(3 + 6)$

because the lengths of  $\textit{text}_c$  and  $\textit{text}_{\bar{c}}$  are 8 and 3, respectively, and

because the constant we add is 6 (vocabulary consists of six terms).

**We then get:**

$$P(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/9 \cdot 1/9 \approx 0.0003.$$

$$P(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.$$

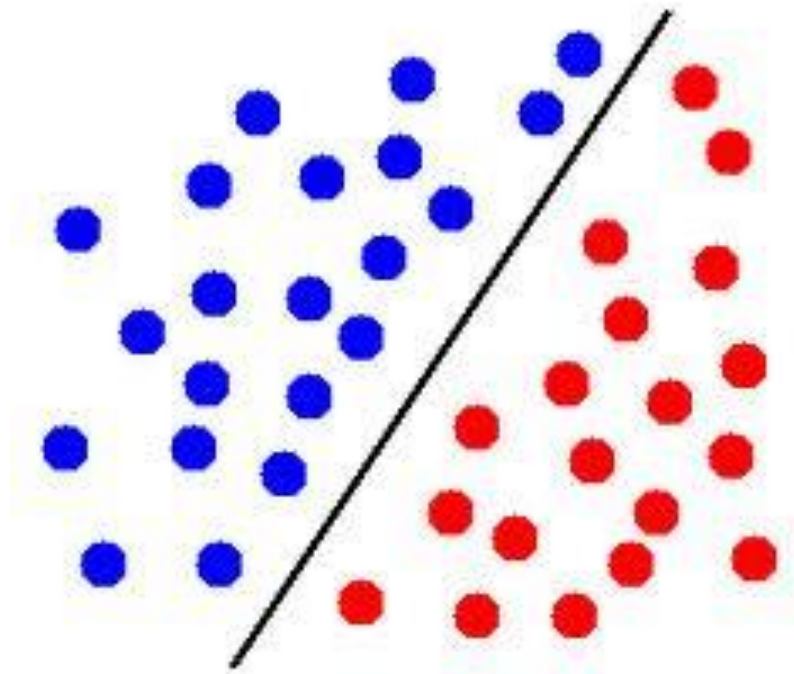
Thus, the classifier assigns the test document to  $c = \textit{China}$ .

The three occurrences of the positive indicator Chinese in  $d_5$  outweigh the occurrences of the two negative indicators Japan and Tokyo.

# Linear Models



# Simple Linear Boundary



# Perceptron: Linear Classifier

$$w_0 = b$$

$$\mathbf{x} = [1, x_1, \dots, x_m]$$

$$\mathbf{w} = [w_0, w_1, \dots, w_m]$$

$$h(\mathbf{x}, \mathbf{w}) = \text{sign}\left(\sum_{i=0}^m w_i x_i\right)$$

$$= \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

$$= \text{sign}(\mathbf{w}^T \mathbf{x})$$

Which outputs  $+1$  or  $-1$ .

Say:

$+1$  corresponds to blue, and

$-1$  to red, or vice versa.

# Perceptron: Learning Algorithm

Start with  
random  $\mathbf{w}$ 's

$$h(\mathbf{x}, \mathbf{w}) = \text{sign}(\mathbf{w}^T \mathbf{x})$$

Training tuples

$$\mathbf{x}^1, y^1$$

$$\mathbf{x}^2, y^2$$

...

$$\mathbf{x}^n, y^n$$

For each misclassified training tuple, e.g.

$$\text{sign}(\mathbf{w}^T \mathbf{x}^k) \neq y^k$$

Update  $\mathbf{w}$

$$\mathbf{w} = \mathbf{w} + \eta \cdot y^k \mathbf{x}^k$$

In the original algorithm,  $\mathbf{w}$  is updated and used immediately for the next training tuple.

In the Excel example, we wait until we process all the tuples before updating  $\mathbf{w}$ .

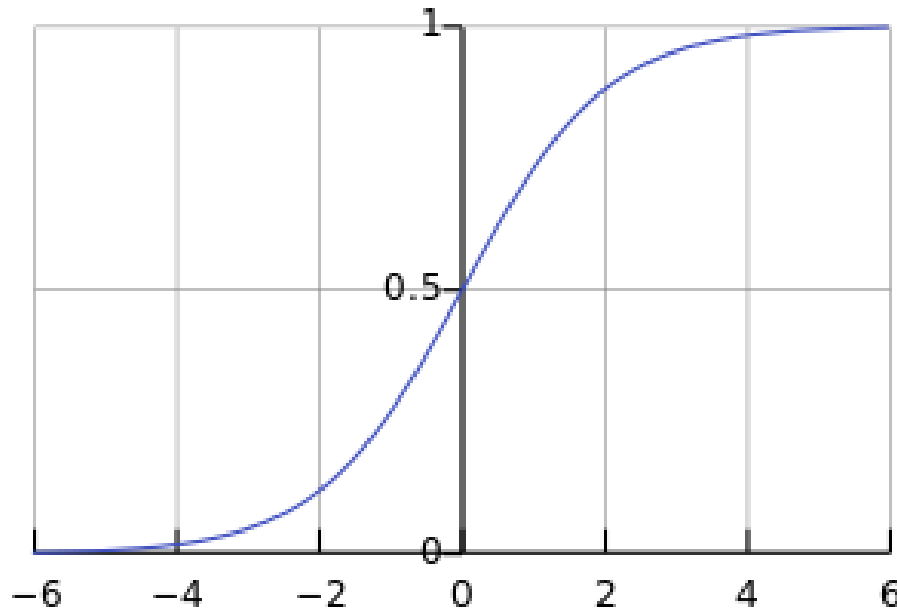
Well, why is this a good rule?

It can be shown that if the data is linearly separable, and we repeat this procedure many times, we will get a line that separates the training tuples.

# Logistic Regression

# Idea

- Similar to perceptron, but **sigmoid** function applied to linearity.



$$S(\mathbf{w}^T \mathbf{x}) = S(z) = \frac{1}{1 + e^{-z}}$$

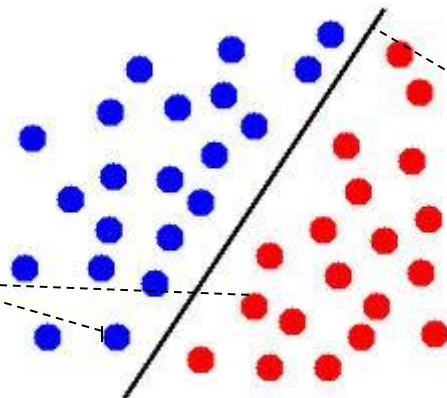
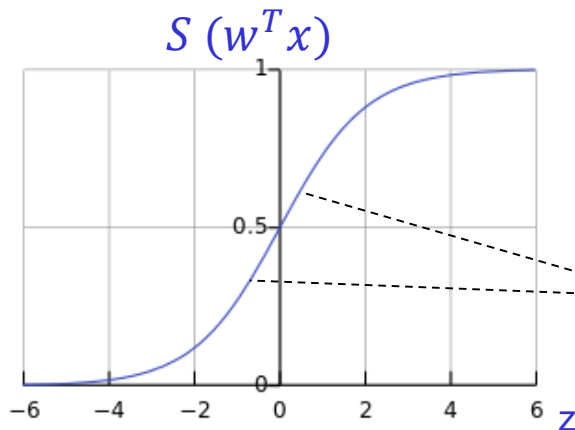
# Logistic Regression: Linear Classifier

- A new tuple comes:  $(\mathbf{x}, ?)$

$$S(\mathbf{w}^T \mathbf{x}) > .5 \quad \text{Typical threshold of 0.5}$$

If yes, predict  $y=1$

If not, predict  $y=-1$



Logistic Regression gives a linear separator, namely the hyperplane defined by  $\mathbf{w}$ . (in 2D this is just a line)

If  $S(\mathbf{w}^T \mathbf{x}) = .5$ ,  
then  $\mathbf{w}^T \mathbf{x} = 0$   
i.e.  $\mathbf{x}$  lies on the hyperplane.

If  $S(\mathbf{w}^T \mathbf{x}) > .5$ ,  
then  $\mathbf{w}^T \mathbf{x} > 0$   
i.e.  $\mathbf{x}$  is above the hyperplane.

If  $S(\mathbf{w}^T \mathbf{x}) \leq .5$ ,  
then  $\mathbf{w}^T \mathbf{x} \leq 0$   
i.e.  $\mathbf{x}$  is below the hyperplane.

Line defined by  $\mathbf{w}$ .

## Probability Interpretation

$$p(y = \pm 1 | \mathbf{x})$$

$$p(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$p(y = -1 | \mathbf{x}) = 1 - \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

- Combining the two we get:

$$p(y = \pm 1 | \mathbf{x}) = \frac{1}{1 + e^{-y\mathbf{w}^T \mathbf{x}}}$$

Why does this interpretation make sense?

The farther from the linear separator a point is, the surer we are for the classification produced.

$\mathbf{w}^T \mathbf{x}$  is a proxy for the distance from the linear separator defined by  $\mathbf{w}$ .

Suppose that for a point  $\mathbf{x}$ ,  $\mathbf{w}^T \mathbf{x}$  is a big positive number. Then, the distance is big.  $S(\mathbf{w}^T \mathbf{x})$  will be close to 1 and we predict class +1 with high certainty.

Suppose that for a point  $\mathbf{x}$ ,  $\mathbf{w}^T \mathbf{x}$  is a big negative number. Then, the distance is again big, but the point is on the other side of the linear separator.  $S(\mathbf{w}^T \mathbf{x})$  will be close to 0 and we predict class -1 with high certainty.

Suppose that for a point  $\mathbf{x}$ ,  $\mathbf{w}^T \mathbf{x}$  is a small positive or negative number. Then, the distance is small.  $S(\mathbf{w}^T \mathbf{x})$  will be close to 0.5 and we predict the class with low certainty.

# Maximum likelihood

- Find  $\mathbf{w}$  that gives the greatest likelihood (probability) of producing the given training data.
  - i.e. maximize **likelihood function**:

$$L(\mathbf{w}) = \prod_{k=1}^n p(y^k | \mathbf{x}^k)$$
$$= \prod_{k=1}^n \frac{1}{1 + e^{-y^k \mathbf{w}^T \mathbf{x}^k}}$$

Probability that the data-source will generate  $y^k$  given  $\mathbf{x}^k$ .  
Plain lang: Probability the training instances have the class they have.  
(Assuming the training instances are independent)

Same  $\mathbf{w}$  for all the training instances.

- Maximizing  $L(\mathbf{w})$  is the same as **maximizing**:

$$\ln L(\mathbf{w}) = \sum_{k=1}^n \ln \left( \frac{1}{1 + e^{-y^k \mathbf{w}^T \mathbf{x}^k}} \right)$$

We are not saying that  $\ln L(\mathbf{w})$  is the same as  $E(\mathbf{w})$ .

- Which is the same as **minimizing**:

$$E(\mathbf{w}) = \frac{1}{n} \sum_{k=1}^n \ln \left( 1 + e^{-y^k \mathbf{w}^T \mathbf{x}^k} \right)$$



# Gradient Descent Algorithm

Initialize  $\mathbf{w}=\mathbf{0}$

For  $t=0,1,2,\dots$  do

    Compute the gradient  $\nabla_E(\mathbf{w}) = -\frac{1}{n} \sum_{k=1}^n \frac{y^k \mathbf{x}^k}{1 + e^{y^k \mathbf{w}^T \mathbf{x}^k}}$

    Update the weights  $\mathbf{w} \leftarrow \mathbf{w} - \kappa \nabla_E(\mathbf{w})$

    Iterate with the next step until  $\mathbf{w}$  doesn't change too much  
    (or for a fixed number of iterations)

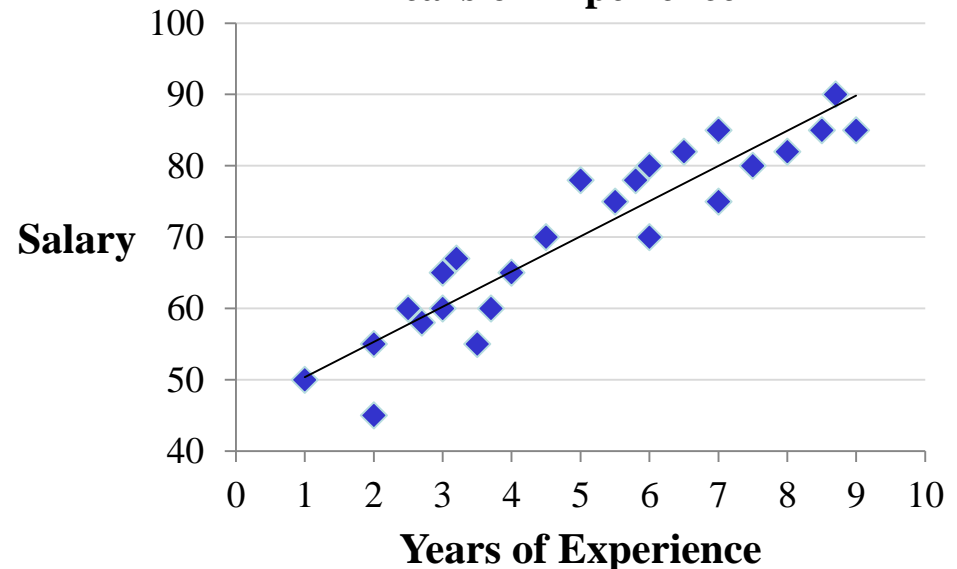
Return final  $\mathbf{w}$ .

# Linear Regression

# Another kind of prediction

YearsOfExperience ( $x_1$ )	Salary ( $y$ )
1	50
2	55
2	45
2.5	60
2.7	58
...	...

- Suppose we'd like to predict **salary** based on number of **years of experience**.
- Different prediction problem because class is a continuous attribute.
  - Called **Regression**
- **Linear Regression**: Build a prediction **line**.



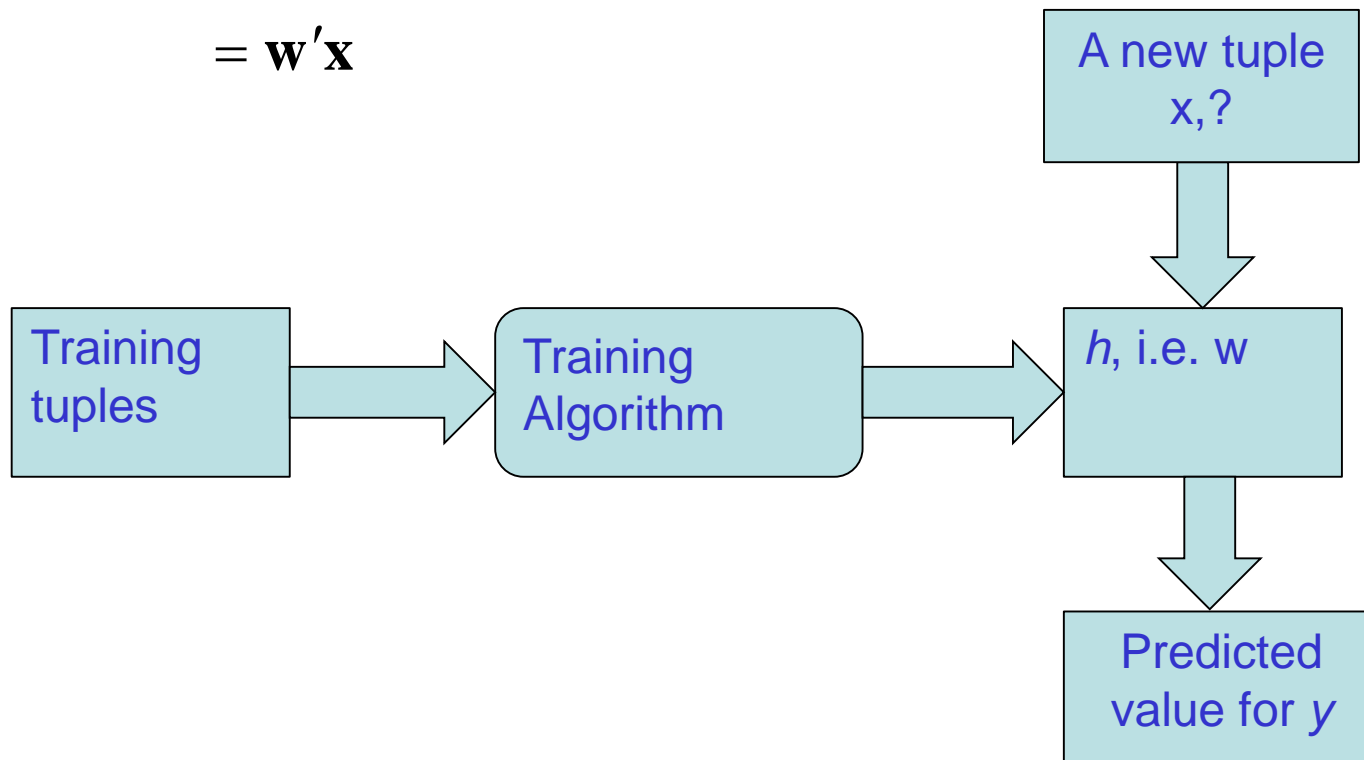
# Linear regression

$$h(\mathbf{w}, \mathbf{x})$$

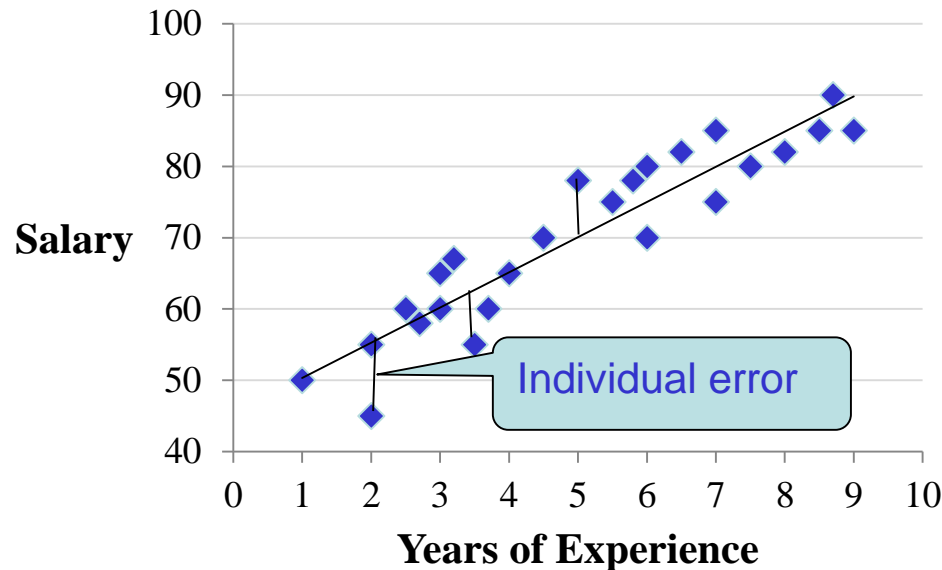
$$= \mathbf{w} \cdot \mathbf{x}$$

$$= \mathbf{w}'\mathbf{x}$$

$$x_0 = 1$$



# Cost/Error/Penalty Function



- **Goal:** find a line (hypothesis)  $h(\mathbf{w}, \mathbf{x})$  that for the training tuples gives numbers close to their  $y$ 's.

$n$  is the number of training tuples

We want to minimize  $E$  over possible  $\mathbf{w}$ 's.  
 $\mathbf{x}^k$  are fixed, they are the training tuples.

$$E(\mathbf{w}) = \frac{1}{2n} \sum_{k=1}^n (y^k - \mathbf{w}'\mathbf{x}^k)^2$$

Average squared error.  
 $\frac{1}{2}$  in the front is just to make the math easier.

# Iterative Method with Gradient

$$\nabla_E(\mathbf{w}) = -\frac{1}{n} \sum_{k=1}^n (y^k - \mathbf{w}'\mathbf{x}^k) \mathbf{x}^k$$

$$\mathbf{w} \leftarrow \mathbf{w} - \kappa \nabla_E(\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \kappa \frac{1}{n} \sum_{k=1}^n (y^k - \mathbf{w}'\mathbf{x}^k) \mathbf{x}^k$$

# Regression in Python

`X = np.array([[1,...], [1,...]])`      # n by m+1

`y = np.array([[...]])`      # 1 by n

`w = np.array([[...]])`      # 1 by m+1

`kappa = 0.1`

`E = (1/(2*n)) * ( np.sum((y-w@X.T)**2) )`

`w = w + kappa*( (1/n)*( np.sum( (y-w@X.T).T*X, axis=0, keepdims=True ) ) )`

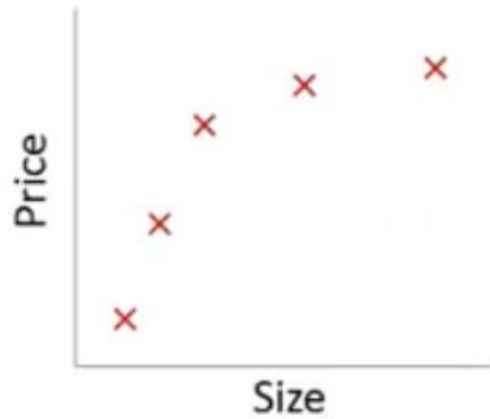
$$E(\mathbf{w}) = \frac{1}{2n} \sum_{k=1}^n (y^k - \mathbf{w}'\mathbf{x}^k)^2$$

$$\mathbf{w} \leftarrow \mathbf{w} + \kappa \frac{1}{n} \sum_{k=1}^n (y^k - \mathbf{w}'\mathbf{x}^k) \mathbf{x}^k$$

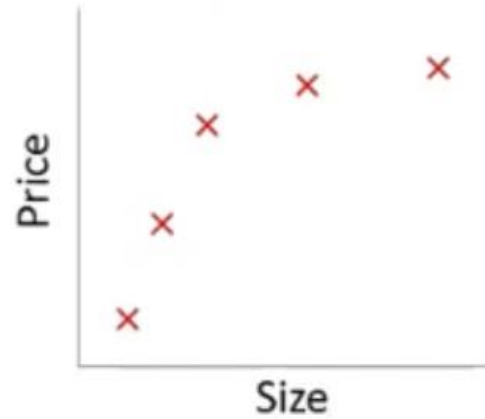
# Underfitting, Overfitting, and Regularization



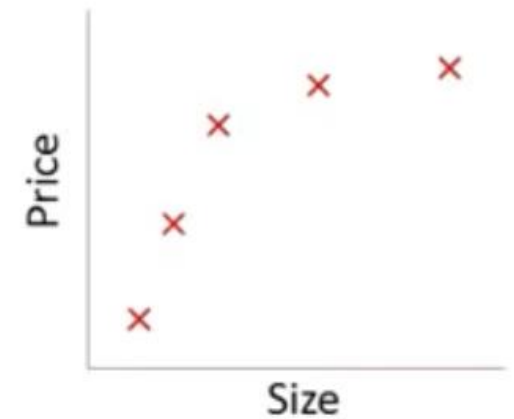
# Example: Regression



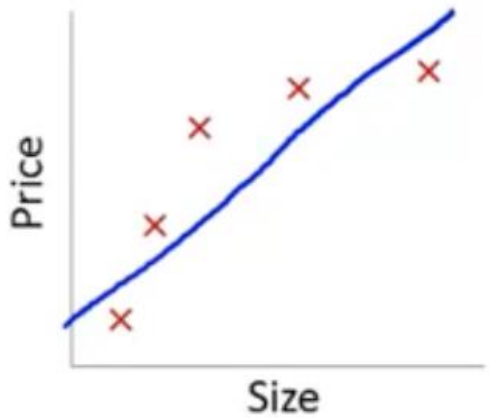
$$w_0 + w_1x$$



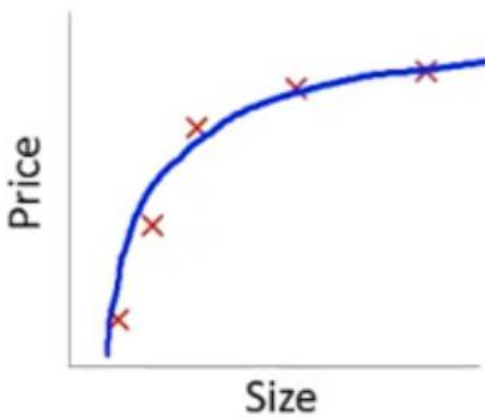
$$w_0 + w_1x + w_2x^2$$



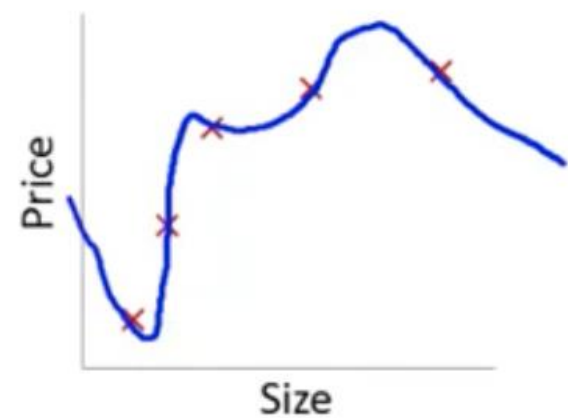
$$w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



"Underfit" "High bias"

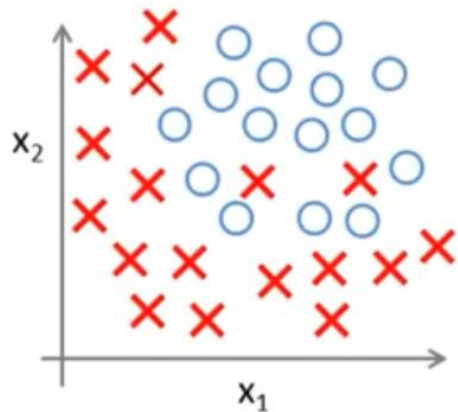


"Just right"

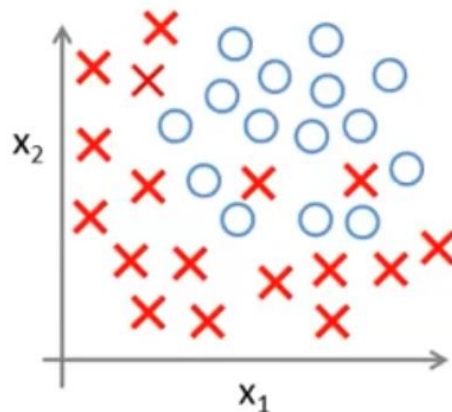


"Overfit" "High variance"

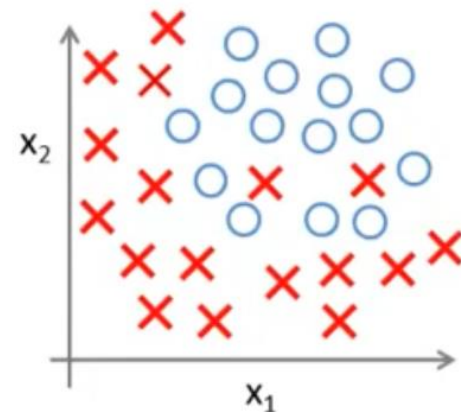
# Example: Logistic Regression



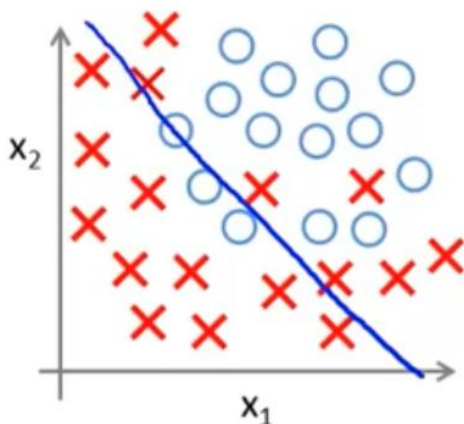
$$\sigma(w_0 + w_1x_1 + w_2x_2)$$



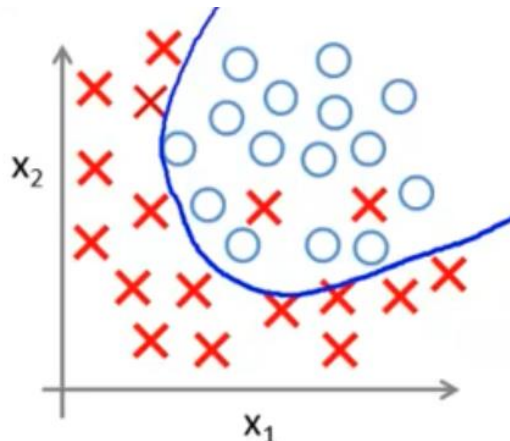
$$\sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2)$$



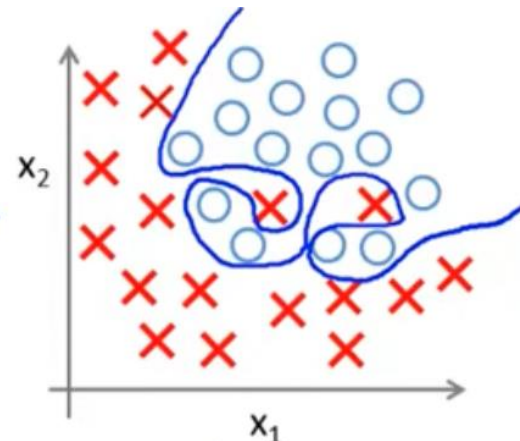
$$\sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + w_6x_1^3x_2 + \dots)$$



"Underfit" "High bias"

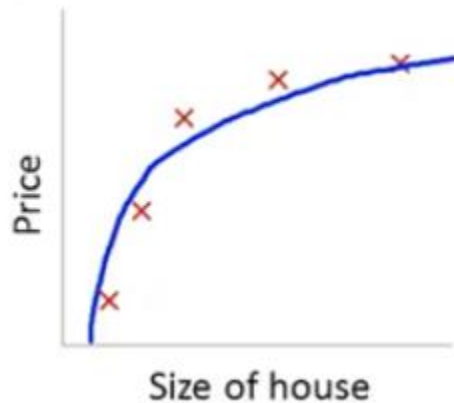


"Just right"

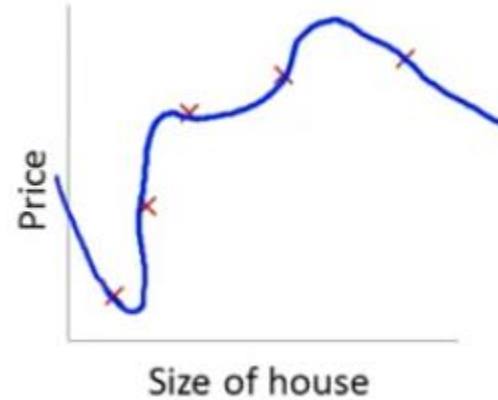


"Overfit" "High variance"

# Regularization - Intuition



$$w_0 + w_1x + w_2x^2$$

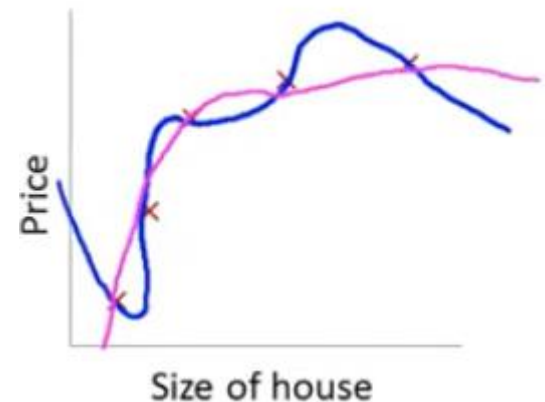


$$w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$

Suppose we penalize and make  $w_3, w_4$  small.

$$\min \frac{1}{2n} \sum_{k=1}^n (y^k - h_w(x^k))^2 + 1000w_3^2 + 1000w_4^2$$

We will end up with small  $w_3, w_4$



# Regularization

- Small values for parameters  $w_1, w_2, \dots, w_m$ 
  - Simpler hypothesis
  - Less prone to overfitting
- Housing:
  - Features:  $x_1, x_2, \dots, x_{100}$
  - Parameters:  $w_1, w_2, \dots, w_{100}$

$$E(\mathbf{w}) = \frac{1}{2n} \left[ \sum_{k=1}^n (y^k - h_{\mathbf{w}}(\mathbf{x}^k))^2 + \lambda \sum_{i=1}^m w_i^2 \right]$$

By convention, we don't include  $w_0$  in the second sum. However, we can include it, without causing much of a difference.

# Regularization

$$E(\mathbf{w}) = \frac{1}{2n} \left[ \sum_{k=1}^n (y^k - h_{\mathbf{w}}(\mathbf{x}^k))^2 + \lambda \sum_{i=1}^m w_i^2 \right]$$

- $\lambda$  is the regularization parameter controls trade off between two goals
  - 1) Want to fit the training set well
  - 2) Want to keep parameters small
- If  $\lambda$  is large we penalize ALL the parameters so all the parameters end up being close to 0
  - If this happens, it's like we got rid of all the terms in the hypothesis
    - This results in underfitting
- So,  $\lambda$  should be chosen carefully - not too big...
  - We look at some automatic ways to select  $\lambda$  later in the course



If  $\lambda$  is large, say  $10^{10}$

$$w_0 + \cancel{w_1 x} + \cancel{w_2 x^2} + \cancel{w_3 x^3} + \cancel{w_4 x^4}$$