Supervised Learning

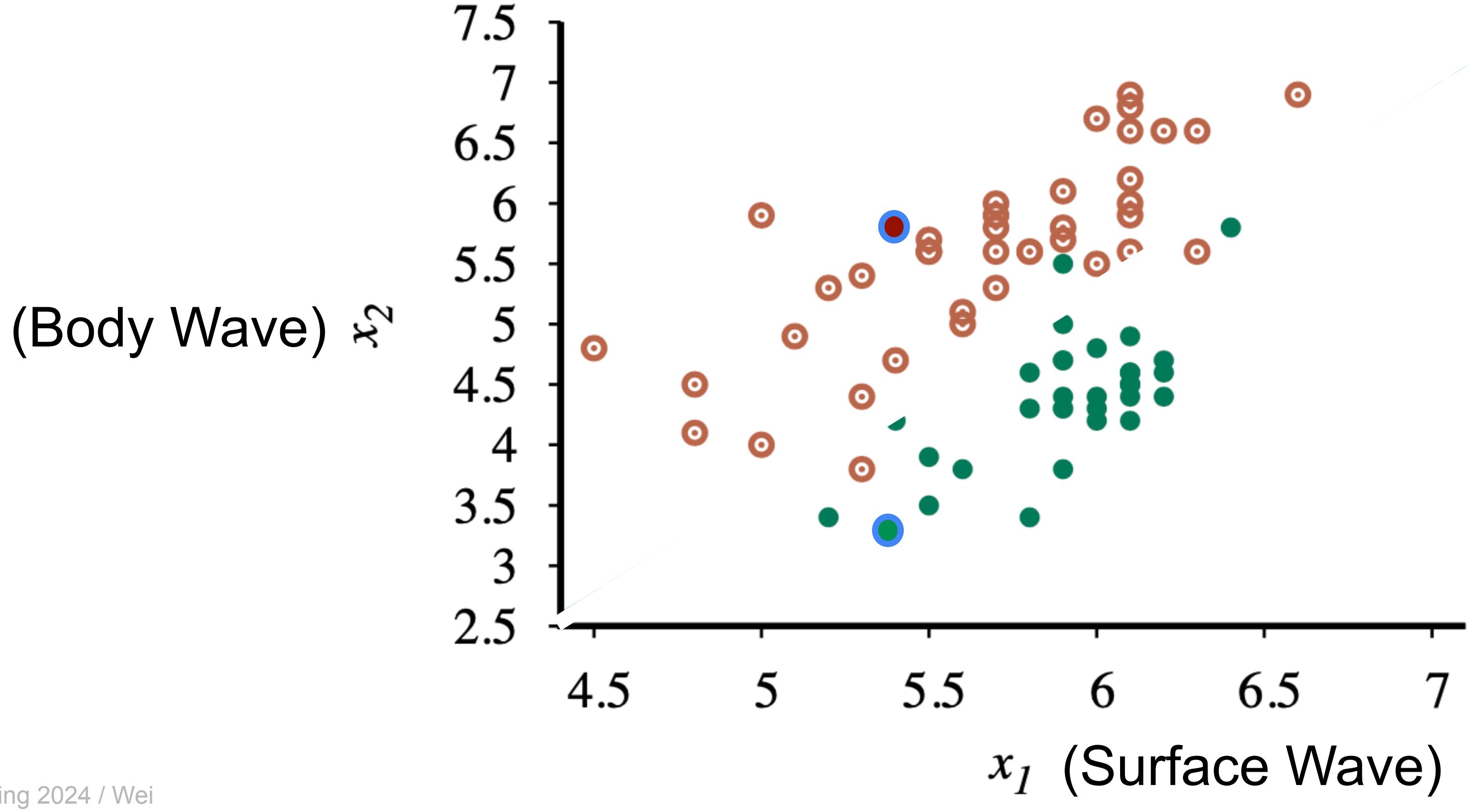
- Regression
- **Classification**

Classification

- Learning a predictor that has **discrete** outputs (labels)
  - Binary classification
  - Multi-class classification

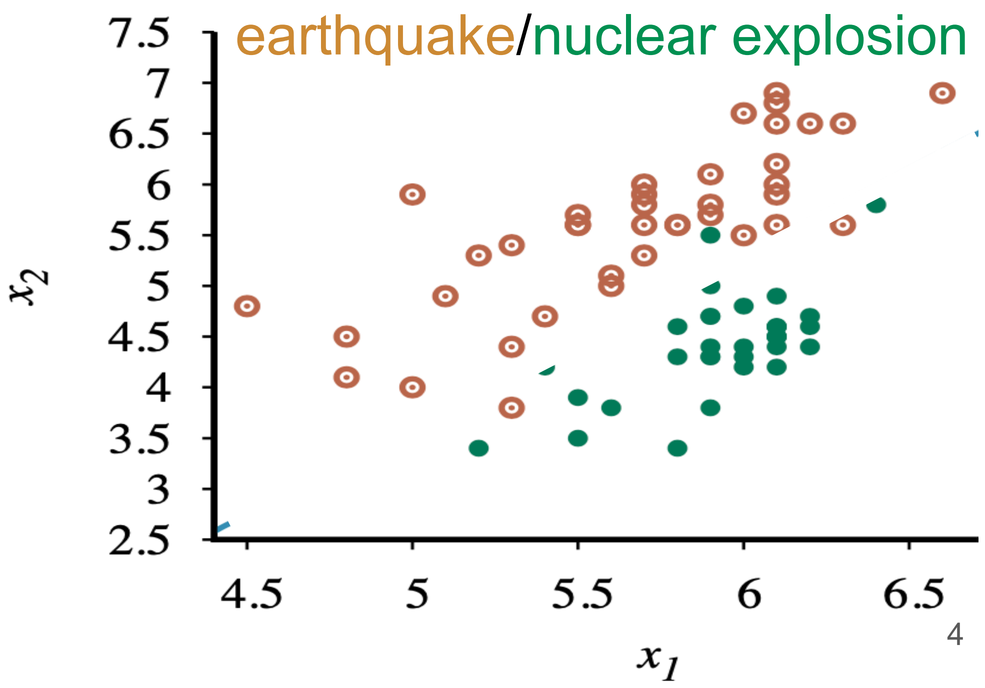
Example: Binary Classification

- Two classes: earthquake/nuclear explosion



Decision Boundary

- A **decision boundary** is a line (or a surface, in higher dimensions) that separates the two classes
- A **linear decision boundary** is called a **linear separator** and data that admit such a separator are called **linearly separable**



Decision Boundary

- Linear separator: $x_2 = w_1 x_1 + w_0$

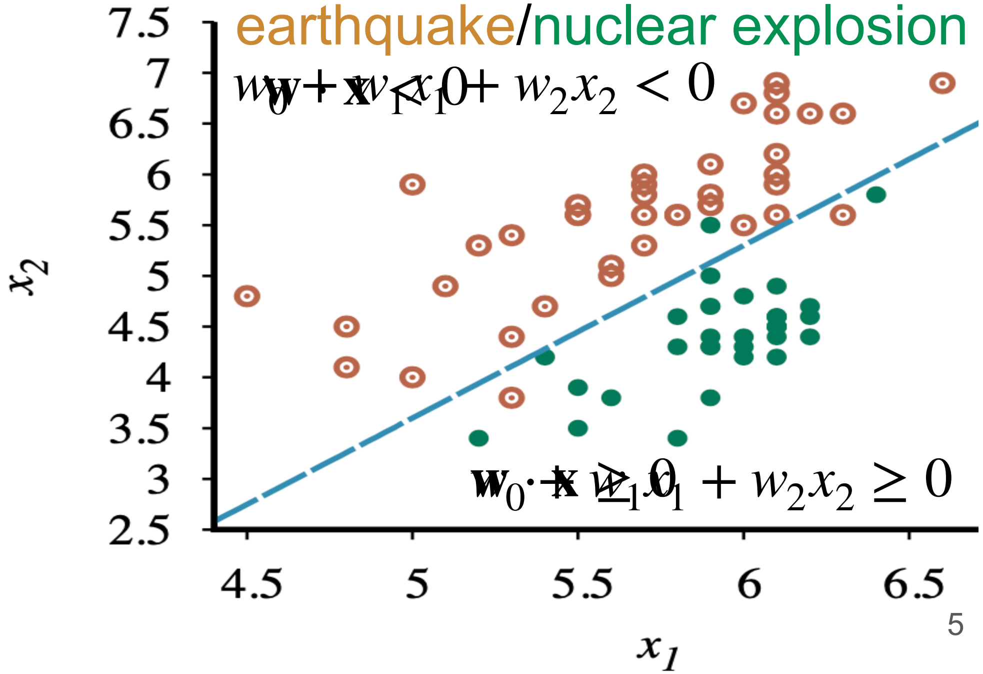
  $\Rightarrow w_0 + w_1 x_1 - x_2 = 0$

  $\Rightarrow w_0 + w_1 x_1 + w_2 x_2 = 0$

  $\Rightarrow \mathbf{w} \cdot \mathbf{x} = 0$

  Weight vector: $\mathbf{w} = < w_0, w_1, w_2 >$

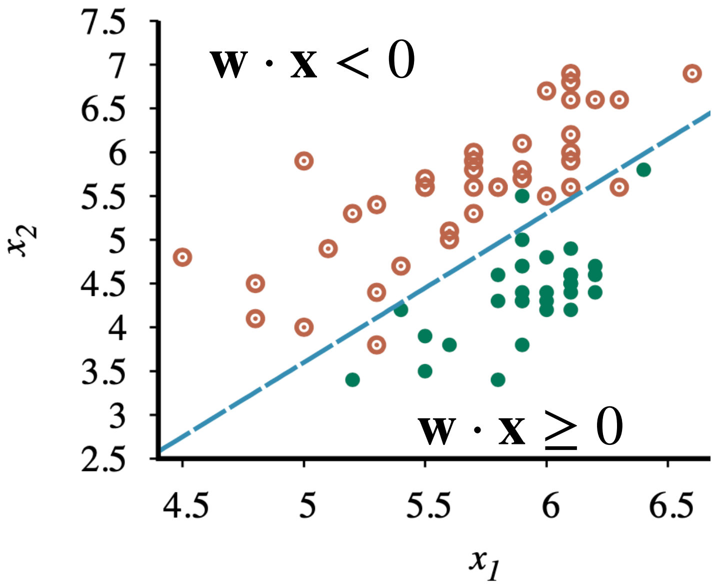
  Input vector: $\mathbf{x} = < 1, x1, x2 >$

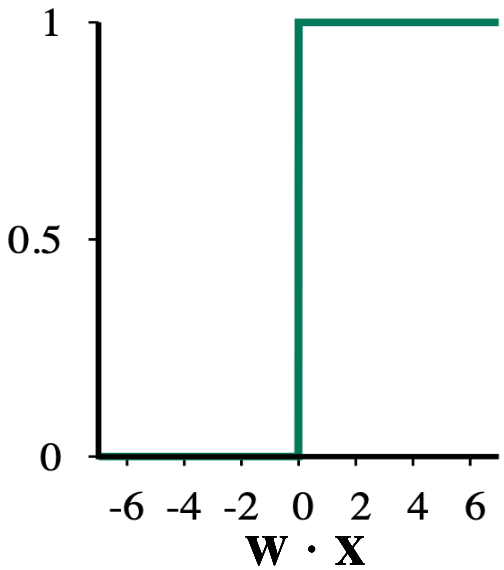


Linear Classifiers

- Given data points of two classes: earthquake/nuclear explosion learn a classification hypothesis h

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



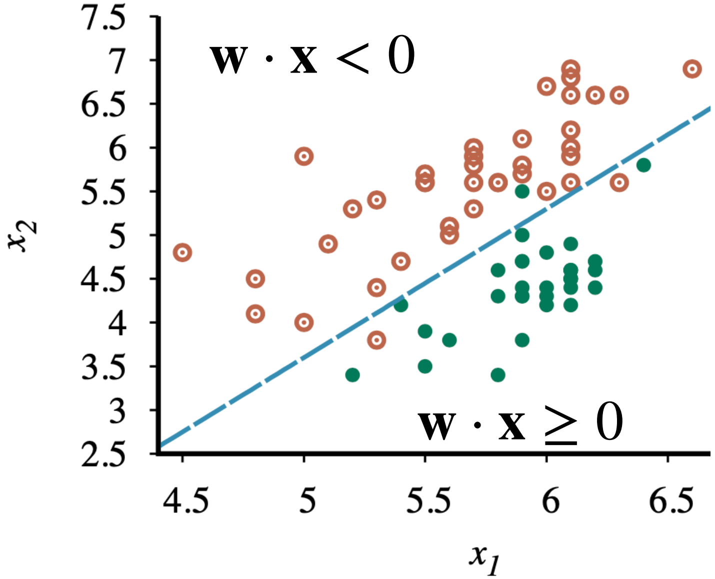


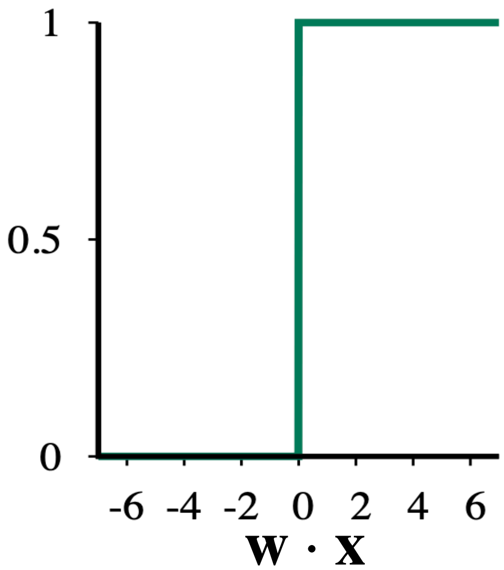
Linear Classifiers with a Hard Threshold

- Given data points of two classes: earthquake/nuclear explosion
  learn a classification hypothesis h

hard threshold

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$





Perceptron Learning Rule

- Given data point $(\mathbf{x}, y)$, update each weight according to

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

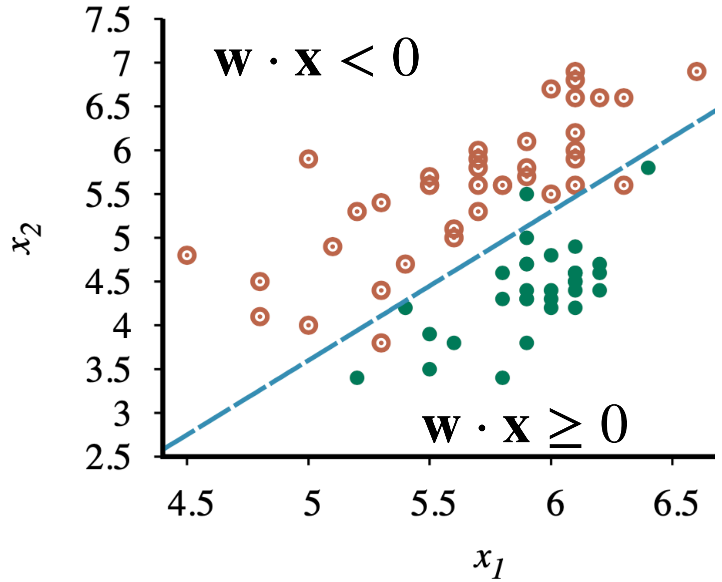
y	$h_{\mathbf{w}}(\mathbf{x})$	x_i	w_i

Perceptron Learning Rule

- Given data point $(\mathbf{x}, y)$, update each weight according to

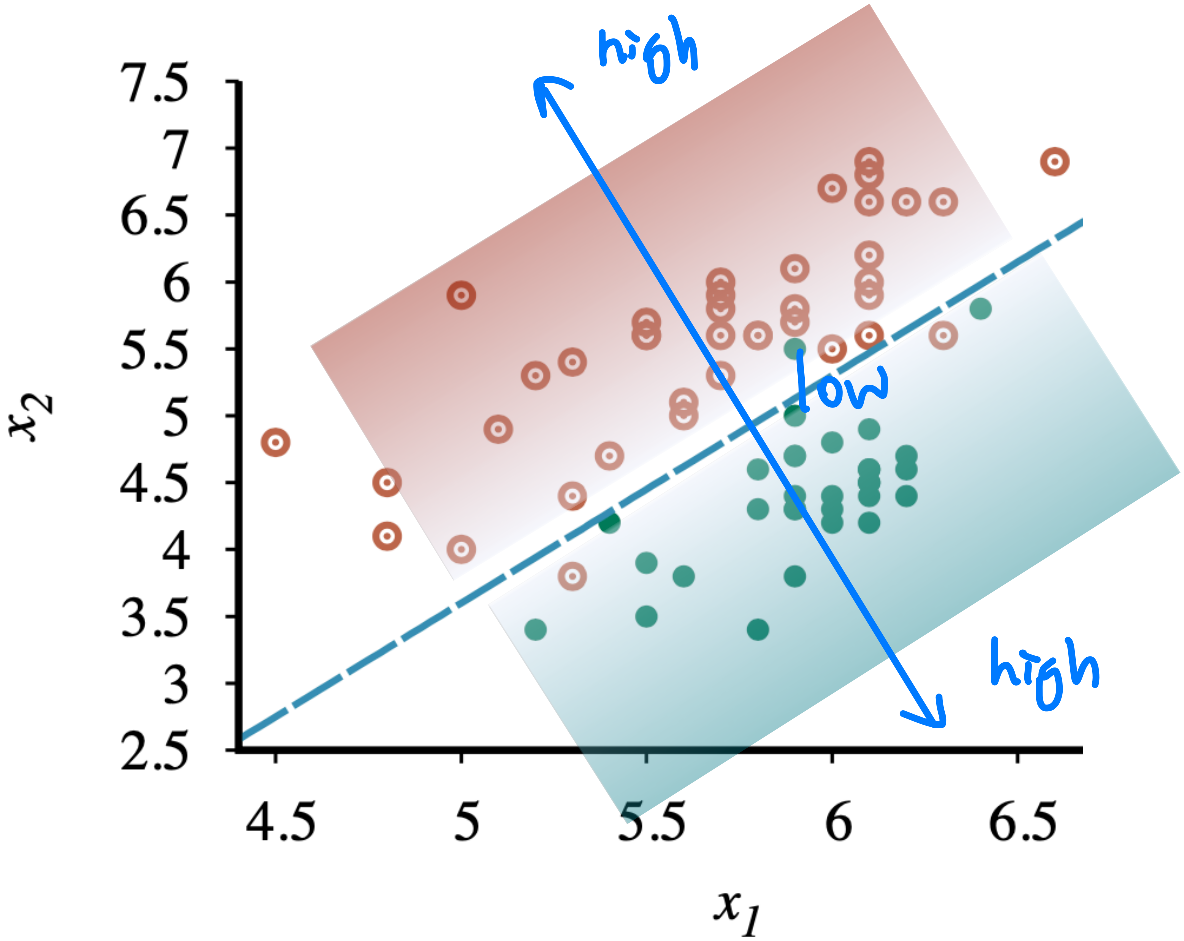
$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

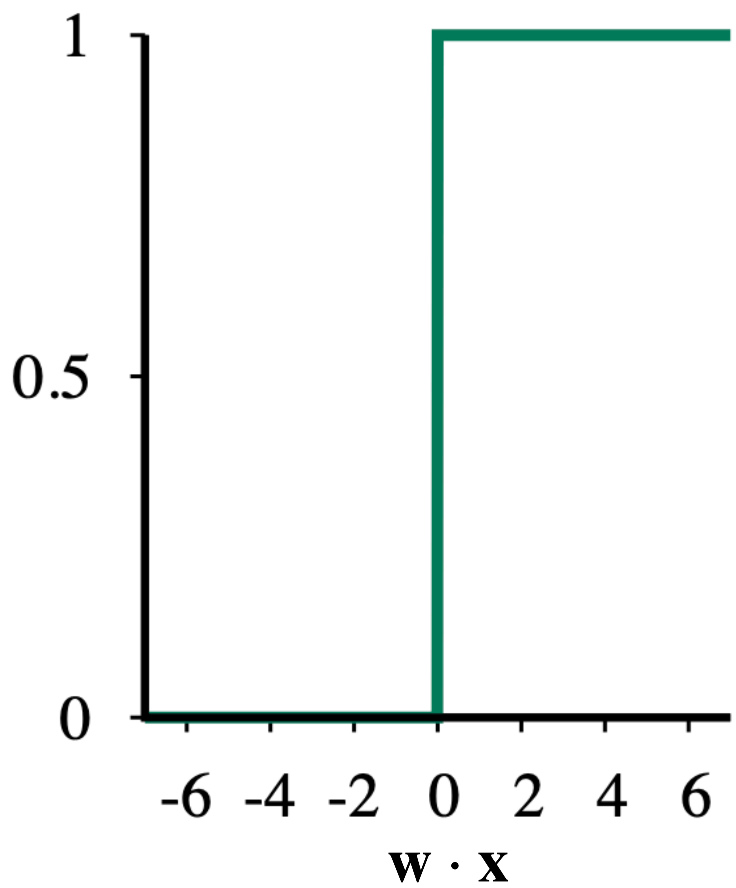
	y	$h_{\mathbf{w}}(\mathbf{x})$	x_i	w_i
$\mathbf{w} \cdot \mathbf{x} \uparrow$	1	0	+	$\uparrow$
	1	0	-	$\downarrow$
$\mathbf{w} \cdot \mathbf{x} \downarrow$	0	1	+	$\downarrow$
	0	1	-	$\uparrow$



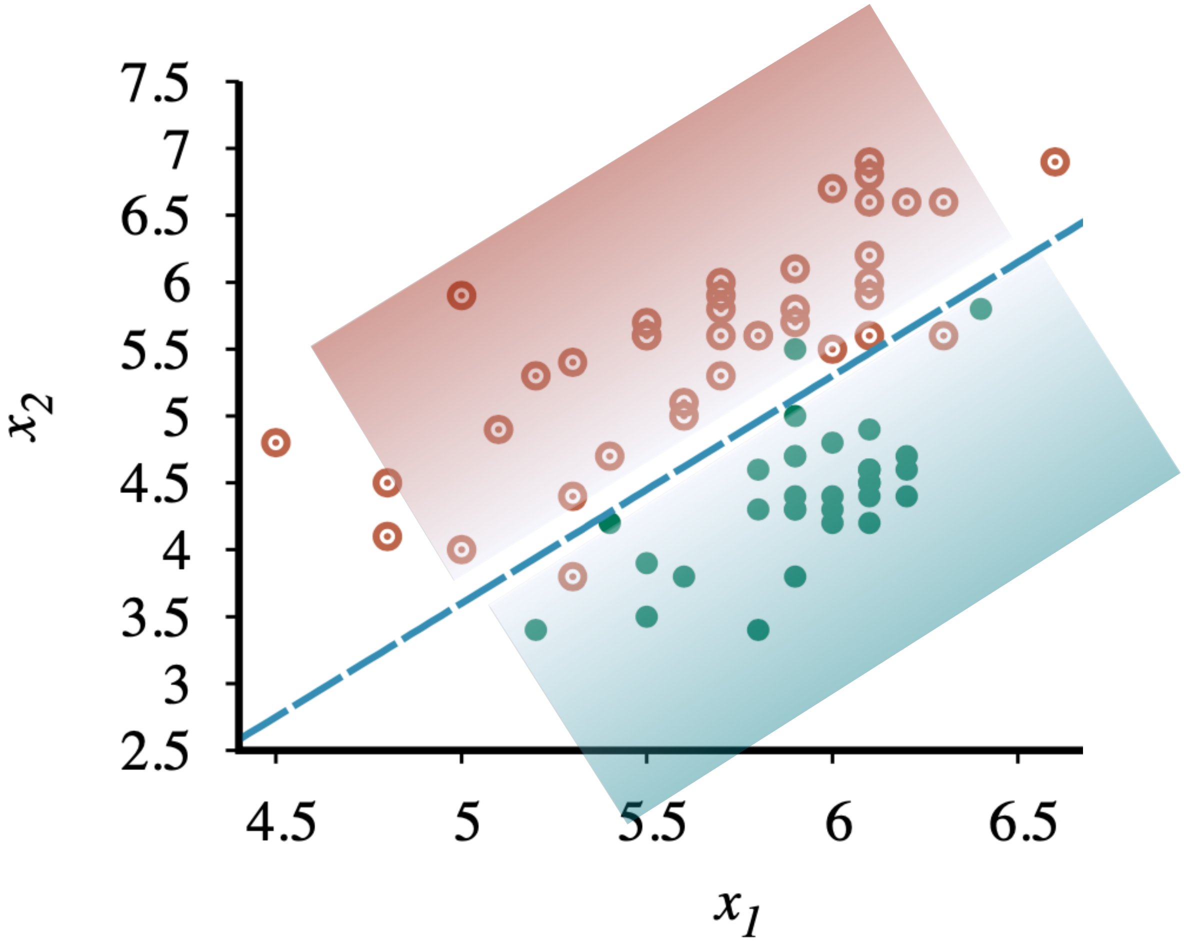
Issues

confidence

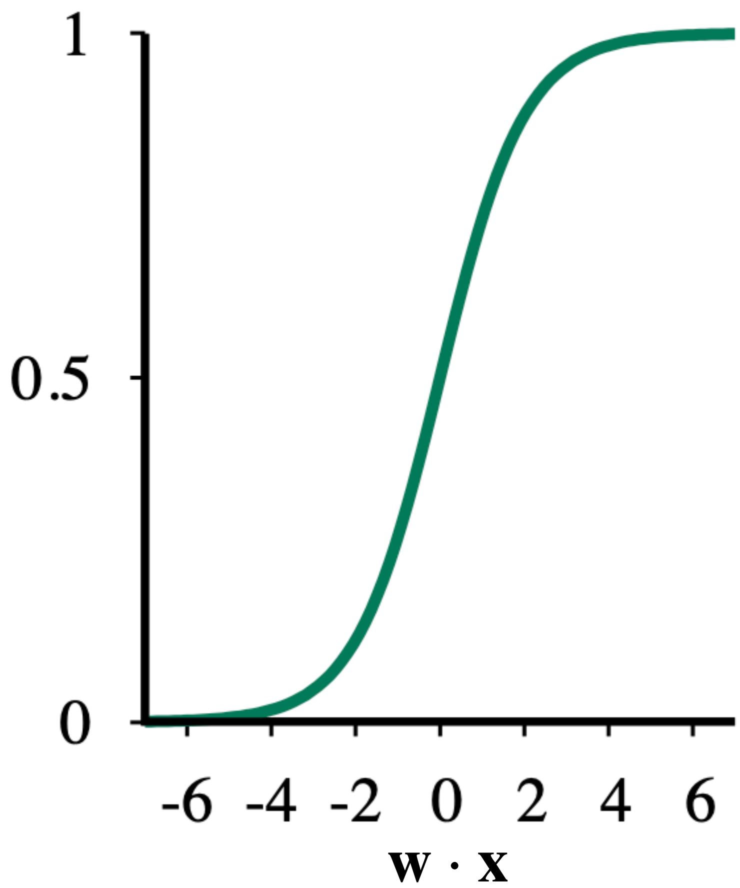




Issues



7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
x_1

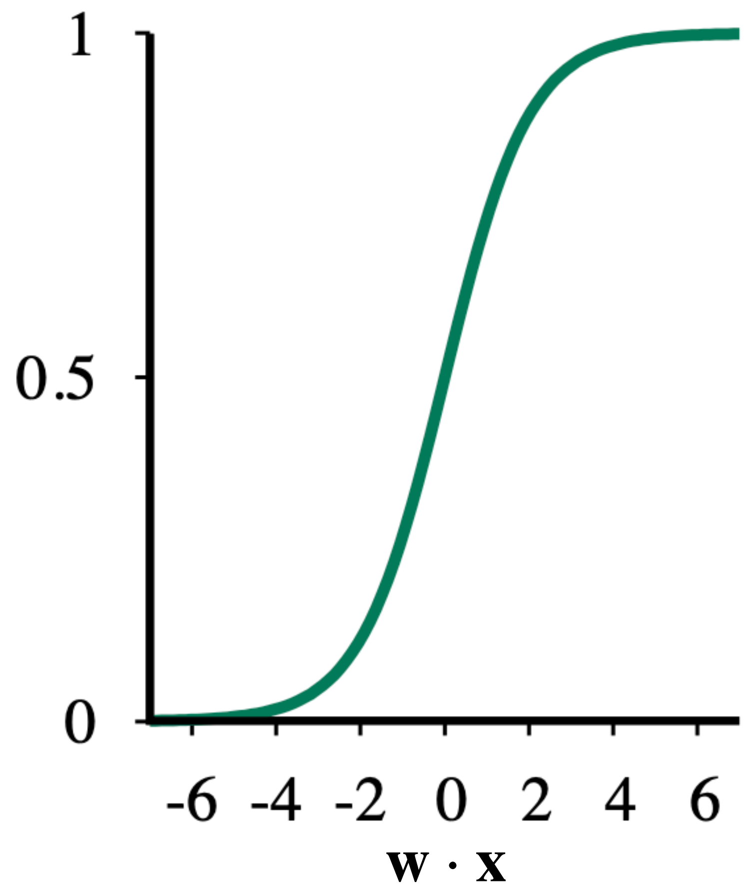


1
0.5
0
-6
-4
-2
0
2
4
6
$\mathbf{w} \cdot \mathbf{x}$

Linear Classifies with a Soft Threshold

- Logistic function (sigmoid function):

$$Logistic(z) = \frac{1}{1 + e^{-z}}$$



Linear Classifies with a Soft Threshold

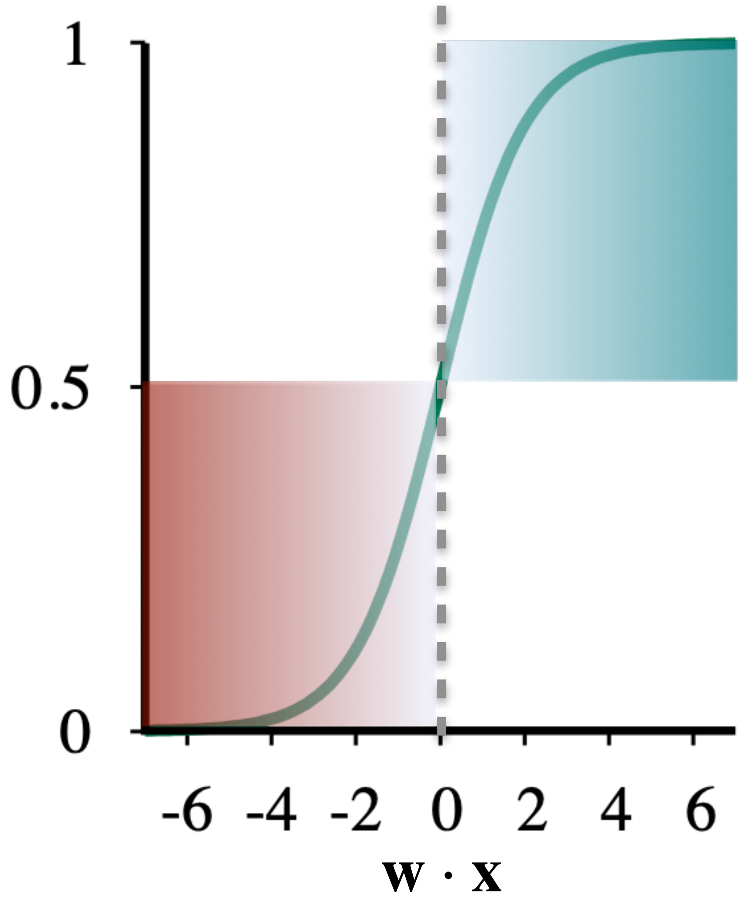
- Logistic function (sigmoid function):

$$Logistic(z) = \frac{1}{1 + e^{-z}}$$

- Let a hypothesis be

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

- w **(weights or coefficients)**:
  - A vector of learned parameters or weights that are optimized during training. Each element of this vector represents the importance of a particular feature in the dataset.
  - The weight values are adjusted during the training process to best fit the training data, minimizing the prediction error (typically via cross-entropy loss).
- x **(features)**:
  - A vector representing the features of a data point. Each element of this vector corresponds to a specific feature or characteristic used to predict the target variable.
  - For example, if you're predicting whether an email is spam or not based on words used, each feature could represent the frequency of a specific word in the email.

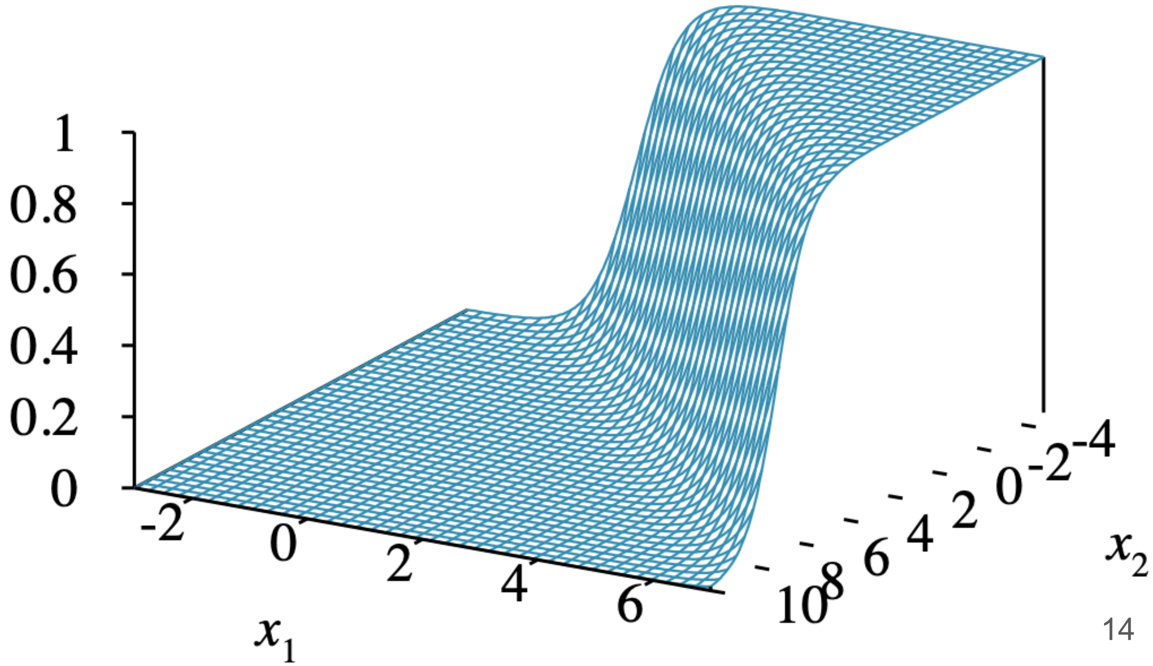


Logistic Regression

- The process to fit the weights of the model

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

  to minimize loss on a data set
  - L_2 loss



Logistic Regression

- Find the optimal value of **w** with this model
  - Gradient descent computation

$$\mathbf{w} \leftarrow \text{any point in the parameter space}$$

$$\textbf{while not } \text{converged} \textbf{ do}$$

$$\quad \textbf{for each } w_i \textbf{ in w do}$$

$$\qquad w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

$$h_{\mathbf{w}}(\mathbf{x}) = \boxed{g(\mathbf{w} \cdot \mathbf{x})} = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

Gradient Descent Computation

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(\mathbf{x}))^2$$ (L2 Loss)

$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i} (y - \boxed{h_{\mathbf{w}}(\mathbf{x})})$$ (Chain Rule)

$$h_{\mathbf{w}}(\mathbf{x}) = \boxed{g(\mathbf{w} \cdot \mathbf{x})} = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

Gradient Descent Computation

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(\mathbf{x}))^2$$

$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i} (y - \boxed{g(\mathbf{w} \cdot \mathbf{x})})$$

$(h_w(x) = g(w \cdot x))$

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

Gradient Descent Computation

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(\mathbf{x}))^2$$

$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i} (y - g(\mathbf{w} \cdot \mathbf{x}))$$

$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times \frac{\partial}{\partial w_i} \mathbf{w} \cdot \mathbf{x}$$ (Chain Rule)

Gradient Descent Computation

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

$$g(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - g(\mathbf{w} \cdot \mathbf{x}))$$

$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times \frac{\partial}{\partial w_i} \mathbf{w} \cdot \mathbf{x}$$

$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times x_i.$$

$$g'(z) = \frac{e^z \cdot (1 + e^z) - e^z \cdot e^z}{(1 + e^z)^2}$$

$$= \frac{e^z}{(1 + e^z)^2}$$

$$= g(z)(1 - g(z))$$

$$\boxed{g'(\mathbf{w} \cdot \mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})(1 - g(\mathbf{w} \cdot \mathbf{x}))}$$

$$h_{\mathbf{w}}(\mathbf{x}) = \boxed{g(\mathbf{w} \cdot \mathbf{x})} = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

Gradient Descent Computation

$$g(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

$$g'(z) = \frac{e^z \cdot (1 + e^z) - e^z \cdot e^z}{(1 + e^z)^2}$$

$$= \frac{e^z}{(1 + e^z)^2}$$

$$= g(z)(1 - g(z))$$

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - \boxed{g(\mathbf{w} \cdot \mathbf{x})})$$

$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times \frac{\partial}{\partial w_i} \mathbf{w} \cdot \mathbf{x}$$

$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times x_i .$$

$$\boxed{g'(\mathbf{w} \cdot \mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})(1 - g(\mathbf{w} \cdot \mathbf{x})) = h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))}$$

Logistic Regression

- Find the optimal value of **w** with this model
  - Gradient descent computation

$$\mathbf{w} \leftarrow \text{any point in the parameter space}$$

$$\textbf{while not } \text{converged } \textbf{do}$$

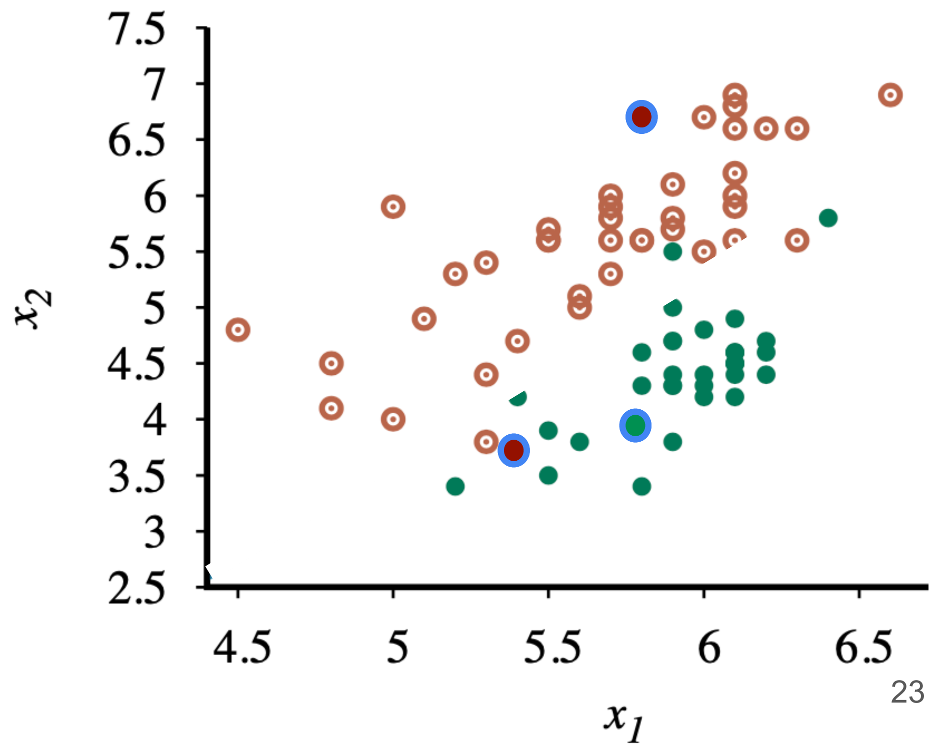
$$\quad \textbf{for each } w_i \textbf{ in } \mathbf{w} \textbf{ do}$$

$$\qquad w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss\,(\mathbf{w})$$

$$\boxed{\alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \,\times\, h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \,\times\, x_i}$$

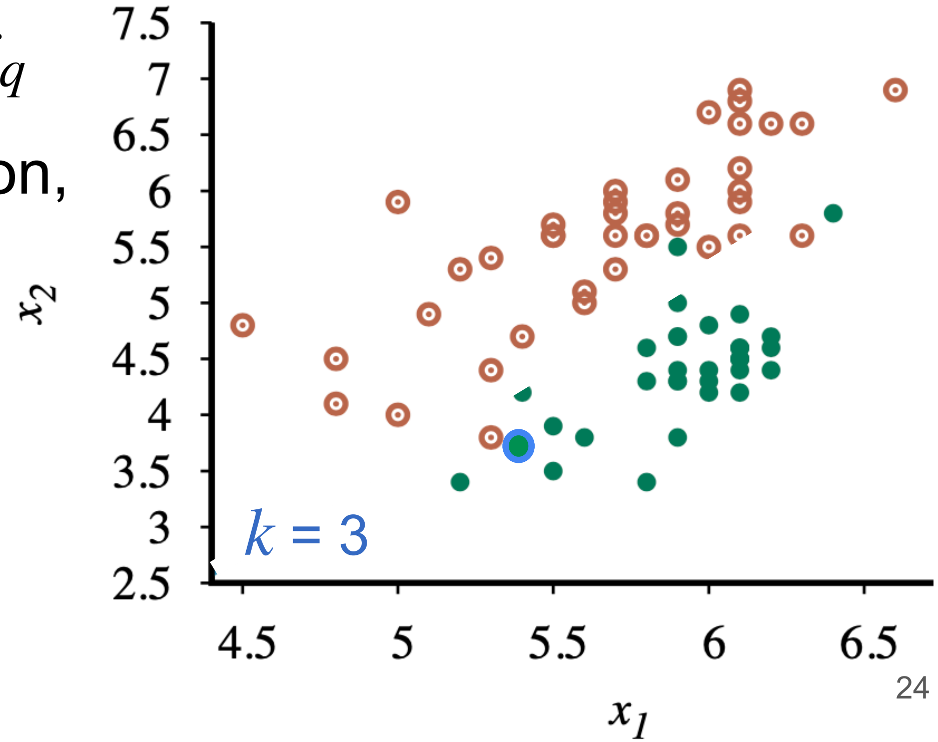
Nearest-Neighbor Models for Binary Classification

- **Nearest-neighbor algorithm**
  - Given a query x_q, the algorithm chooses the class of the nearest example to x_q



Nearest-Neighbor Models for Binary Classification

- ***k*-nearest-neighbors algorithm**
  - Given a query x_q, the algorithm chooses the most common class out of the k nearest examples to x_q
  - To avoid ties on binary classification, k is usually chosen to be an odd number



Distance Measurement

- Minkowski distance (L^p norm)

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = \left(\sum_i \left| x_{j,i} - x_{q,i} \right|^p \right)^{1/p}$$

- Examples
  - p = 2: Euclidean distance
  - p = 1: Manhattan distance

Issues for Distance Measurement

- The total distance will be affected by a change in units in any dimension
- Different scales

- Scaling approach
  - Normalization

Normalization

- **Z-score normalization (Standardization)**

$$x'_{j,i} = \frac{x_{j,i} - \mu_i}{\sigma_i}$$

  - μ_i is the mean of the values in each dimension
  - σ_i is the standard deviation of the values in each dimension

Normalization

- **Min-max normalization**

$$x'_{j,i} = \frac{x_{j,i} - min_i}{max_i - min_i}$$

  - max_i is the maximum of the values in each dimension
  - min_i is the minimum of the values in each dimension
  - The new value $x'_{j,i}$ is in [0,1]

Example: Weather Data

Input Output

Outlook	Temperature	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

Example: Weather Data

- The new data E:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

Q: **Play is Yes or No**?

Recap: Probabilities

- Bayes' Rule
  - For any two propositions a and b,

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)}$$

- Conditional independence of two variables X and Y, given a third variable Z, is

$$\mathbf{P}(X,Y \mid Z) = \mathbf{P}(X \mid Z)\mathbf{P}(Y \mid Z)$$

Probabilistic Model

- The new data E:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

Q: **Play is Yes or No**?

$$P(Play = yes \mid E) = ?$$

$$P(Play = no \mid E) = ?$$

Naïve Bayes Models

- Naïve Assumptions
  - Features/Attributes are equally important
  - Features/Attributes are conditionally independent

Naïve Bayes Models

- Naïve Assumptions
  - Features/Attributes are equally important
  - Features/Attributes are conditionally independent

$$\Rightarrow P(E \mid Play = yes) = P(E_1 \mid Play = yes)P(E_2 \mid Play = yes)\cdots P(E_n \mid Play = yes)$$

$$= \Pi_i(E_i \mid Play = yes)$$

- Naïve Bayes

$$P(Play = yes \mid E) = \frac{P(E \mid Play = yes)P(Play = yes)}{P(E)}$$

Evidence

$$= \frac{\Pi_i P(E_i \mid Play = yes)P(Play = yes)}{P(E)}$$

Example: Naïve Bayes Models

- The new data E:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$P(O = sunny \mid Play = yes)P(T = cool \mid Play = yes)P(H = high \mid Play = yes)P(W = true \mid Play = yes)$$

$$P(Play = yes \mid E) = \frac{\Pi_i P(E_i \mid Play = yes) P(Play = yes)}{P(E)}$$

Outlook			Temperature			Humidity			Windy			Play	
	Yes	***No***		***Yes***	***No***		***Yes***	***No***		***Yes***	***No***	***Yes***	***No***
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								

Outlook	Temperature	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

Outlook			Temperature			Humidity			Windy			Play	
	Yes	***No***		***Yes***	***No***		***Yes***	***No***		***Yes***	***No***	***Yes***	***No***
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$P(O = sunny \,|\, Play = yes)P(T = cool \,|\, Play = yes)P(H = high \,|\, Play = yes)P(W = true \,|\, Play = yes)$$

$$P(Play = yes \,|\, E) = \frac{\Pi_i P(E_i \,|\, Play = yes) P(Play = yes)}{P(E)}$$

$$= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{P(E)} = \frac{0.0053}{P(E)}$$

Outlook			Temperature			Humidity			Windy			Play	
	Yes	**No**		**Yes**	**No**		**Yes**	**No**		**Yes**	**No**	**Yes**	**No**
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$P(O = sunny \mid Play = no)P(T = cool \mid Play = no)P(H = high \mid Play = no)P(W = true \mid Play = no)$$

$$P(Play = no \mid E) = \frac{\Pi_i P(E_i \mid Play = no) P(Play = no)}{P(E)}$$

$$= \frac{\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14}}{P(E)} = \frac{0.0206}{P(E)}$$

Example: Naïve Bayes Models

- The new data E:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

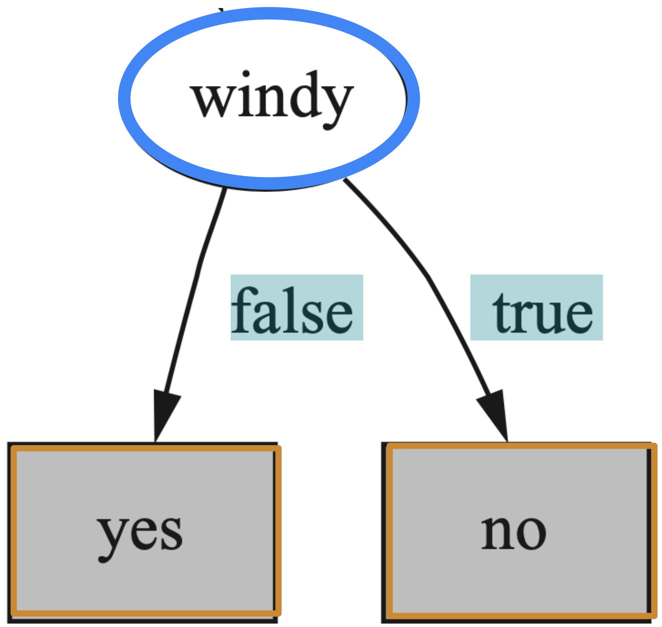
$$P(Play = yes \mid E) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%,$$

$$P(Play = no \mid E) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%.$$

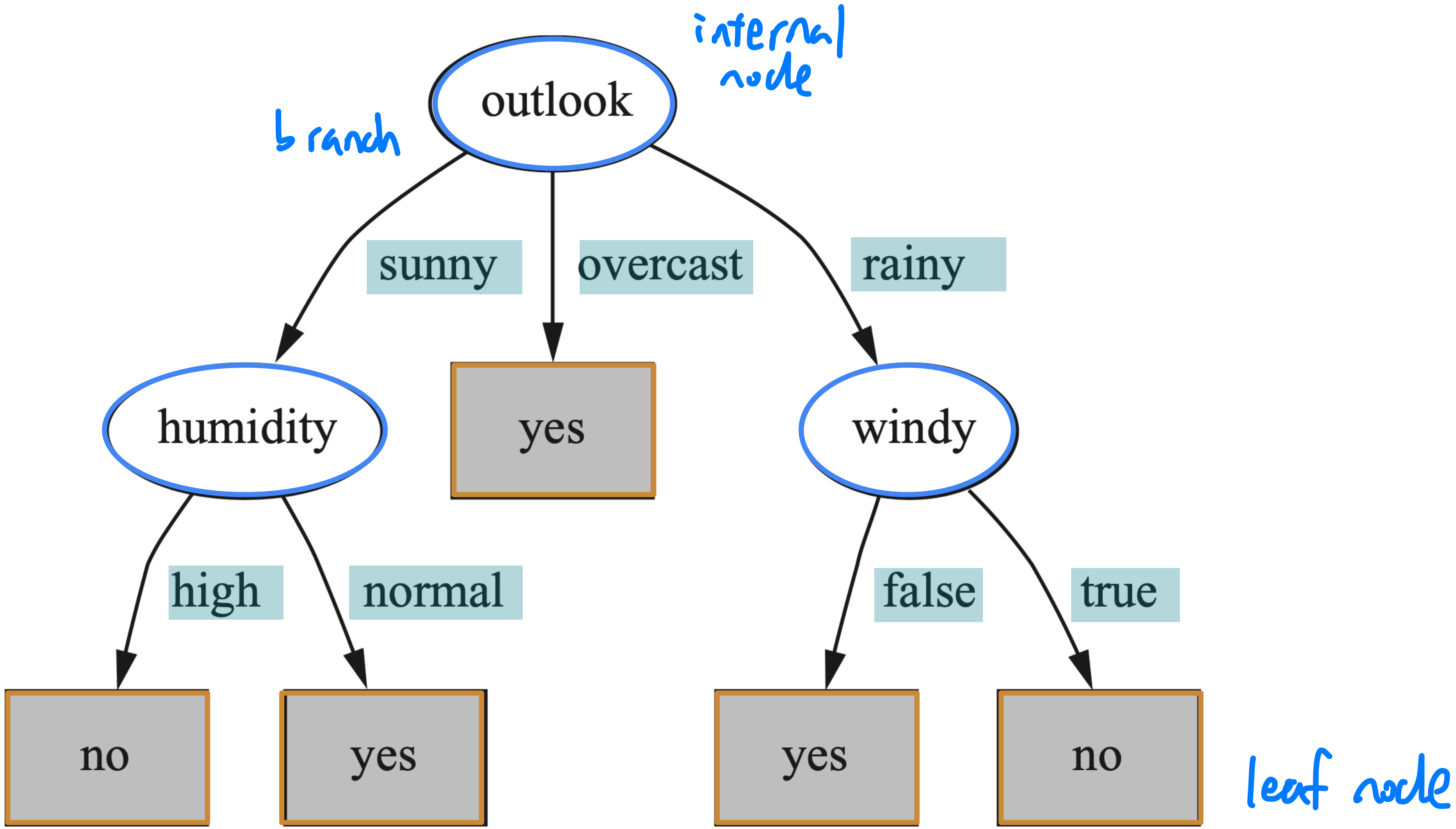
Play = no

Decision Trees

- A **decision tree** is a representation of a function that maps **a vector of attribute values** to **a single output value**, a “decision”
  - Each **internal node** in the tree corresponds to a test of the value of one of the input attributes
  - The **branches** from the node are labeled with the possible values of the attribute
  - The **leaf nodes** specify what value is to be returned by the function

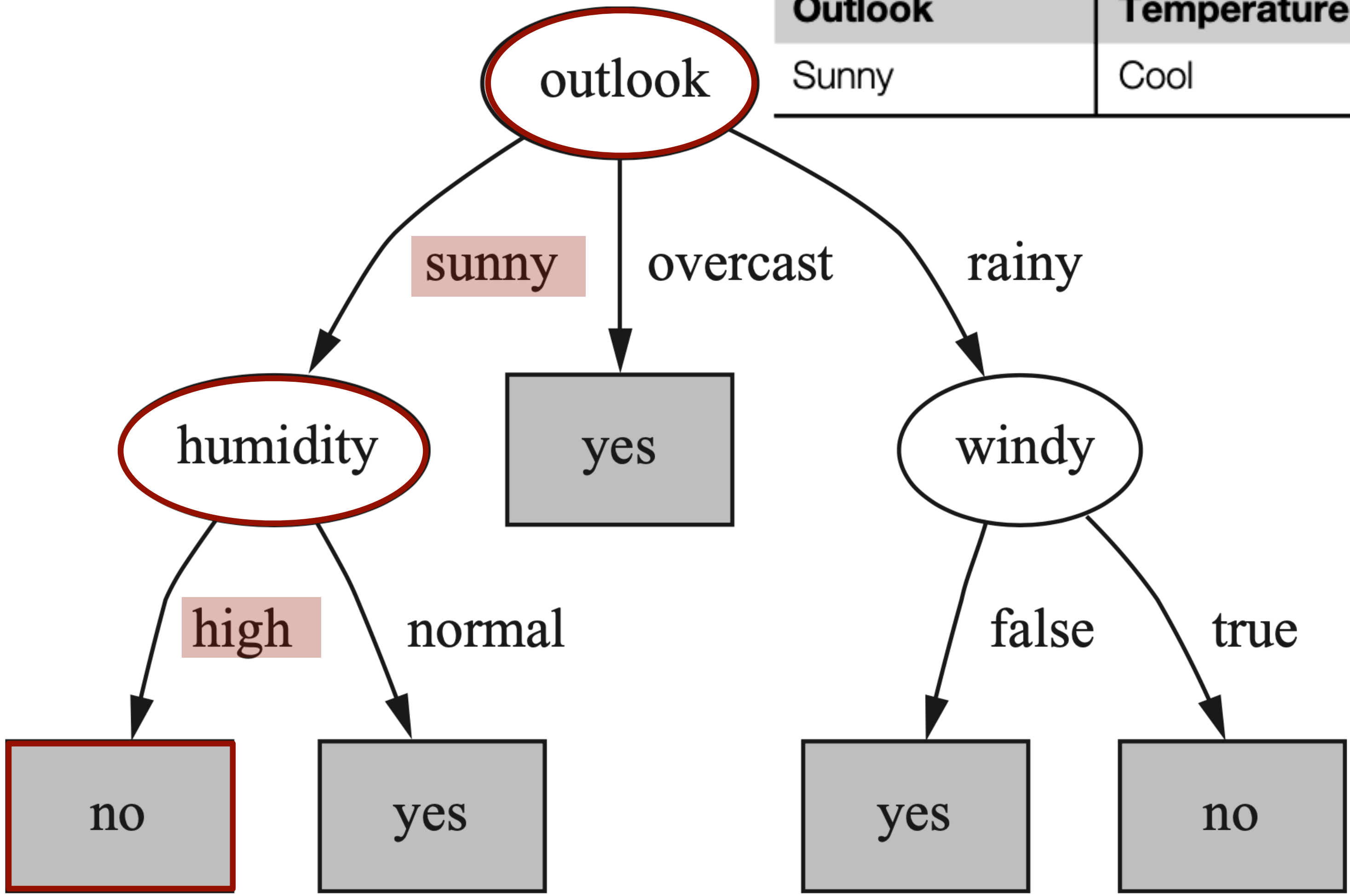


Example: Decision Trees



Example: Decision Trees

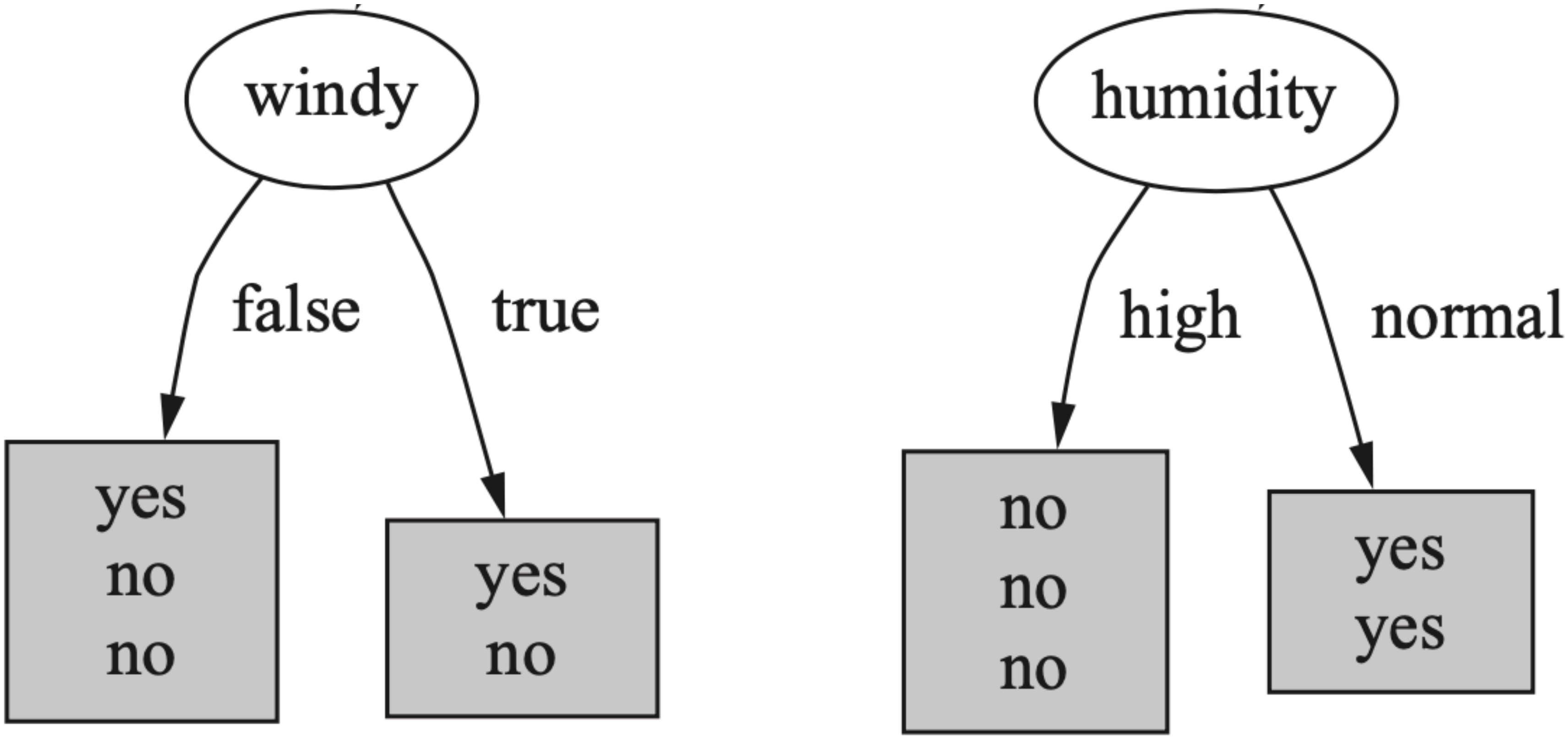
Outlook	Temperature	Humidity	Windy
Sunny	Cool	High	True



Learning Decision Trees from Examples

- Idea
  - Get to the correct classification with a small number of tests
  - Find a tree that is consistent with the examples and the tree is as small as possible
- A greedy divide-and-conquer strategy
  - Always test the **most important attribute** first
    - Makes the most difference to the classification of an example
  - Recursively solve the smaller subproblems that are defined by the possible results of the test

Example: Attribute Importance



Learning Decision Trees from Examples

- Four cases to consider for these recursive subproblems
  - **The remaining examples are all positive (or all negative):**
    Done! We can answer Yes or No
  - **Some positive and some negative examples:**
    Choose the best attribute to split them
  - **No examples left:**
    (i.e., no example has been observed for this combination of attribute values),
    Return *the most common output value from the set of examples that were used in constructing the node's parent*
  - **No attributes left, but both positive and negative examples:**
    (This can happen because there is an error or noise in the data.)
    Return the *most common output value of the remaining examples*

Decision Tree Learning Algorithm

function LEARN-DECISION-TREE(*examples*, *attributes*, *parent_examples*) **returns** a tree

```
if examples is empty then return PLURALITY-VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY-VALUE(examples)
else
    A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
    tree ← a new decision tree with root test A
    for each value v of A do
        exs ← {e : e ∈ examples and e.A = v}   # examples for the subset
        subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)   #recursion
        add a branch to tree with label (A = v) and subtree subtree
    return tree
```

PLURALITY-VALUE Select the most common output value among a set of examples, breaking ties randomly

Importance Measurement

- Entropy (Information Theory)
  - The average amount of information contained in each message received (that is measured in bits)
  - A measure of the uncertainty of a random variable
    - e.g., a coin that always comes up heads has no uncertainty and its entropy is defined as zero
- The entropy of a random variable V with values v_k having probability P(v_k) is defined as

$$\text{Entropy:} \quad H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

Fair Coin Flip

Four-sided Die

1 2 3 4

https://www.cmc.gov.tw/UpLoadFiles/ProductFiles/A0C38C819E564136BEE447BE8B2861E3.gif

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

Examples: Entropy

- The entropy of a fair coin flip is 1 bit:

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

- The entropy of a four-sided die is 2bit:

$$H(Die4) = -(0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25) = 2$$

Importance Measurement

- **The entropy of a Boolean random variable** that is true with probability q:

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

- If a training set contains p positive examples and n negative examples, then **the entropy of the output variable on the whole set** is

$$H(Output) = B\left(\frac{p}{p+n}\right)$$

Examples: Entropy

$$H(Output) = B\left(\frac{p}{p+n}\right)$$

$$B(q) = -(q\log_2 q + (1-q)\log_2(1-q))$$

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								

- H(Play) = $B(\frac{9}{9+5}) = -(\frac{9}{14}log_2\frac{9}{14} + \frac{5}{14}log_2\frac{5}{14}) = 0.940$ bits

Importance Measurement

- An attribute A with *d* distinct values divides the training set *E* into subsets $E_1, \cdots, E_d$
- Each subset E_k has p_k positive examples and n_k negative examples
- A randomly chosen example from the training set has the *k*th value for the attribute (i.e., is in E_k with probability $\frac{p_k + n_k}{p + n}$), so **the expected entropy remaining after testing attribute A** is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$B(q) = -(q \log_2 q + (1 - q) \log_2 (1 - q))$$

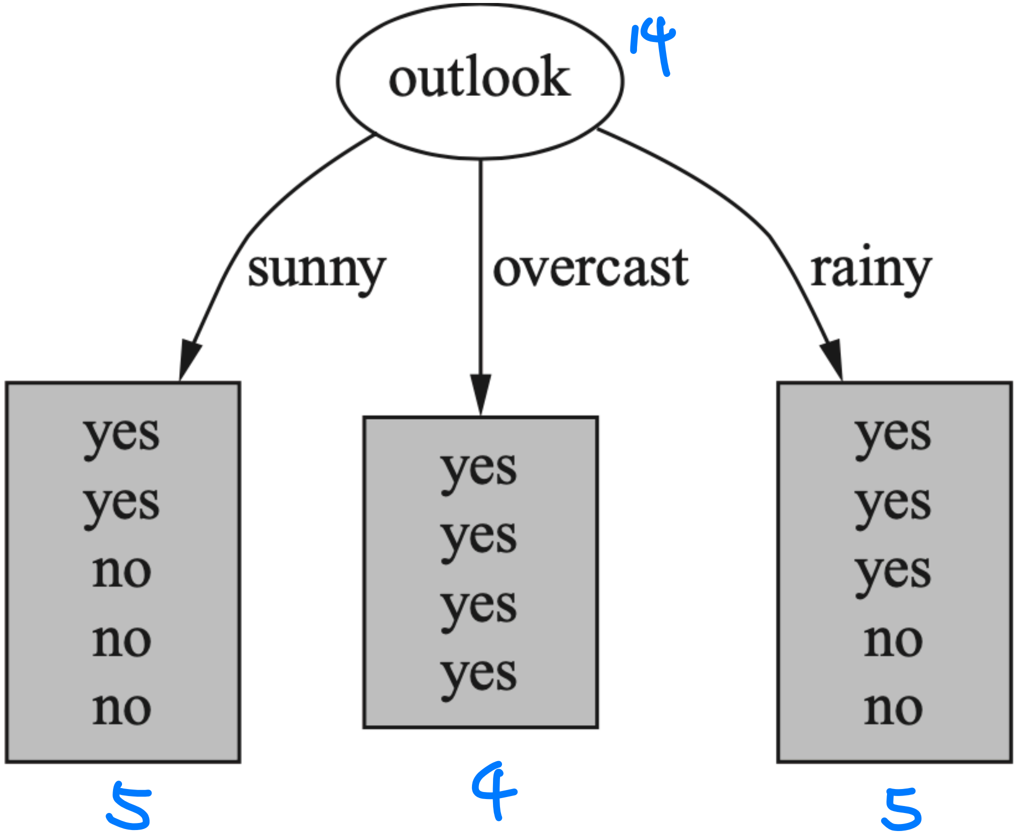
Examples: Entropy

- Remainder(outlook)

$$= \frac{5}{14}\mathbf{B}(\frac{2}{5}) + \frac{4}{14}\mathbf{B}(\frac{4}{4}) + \frac{5}{14}\mathbf{B}(\frac{3}{5})$$

$$= (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971$$

$$= 0.693 \text{ bits}$$



Importance Measurement

- Importance function
  - The **information gain** from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A).$$

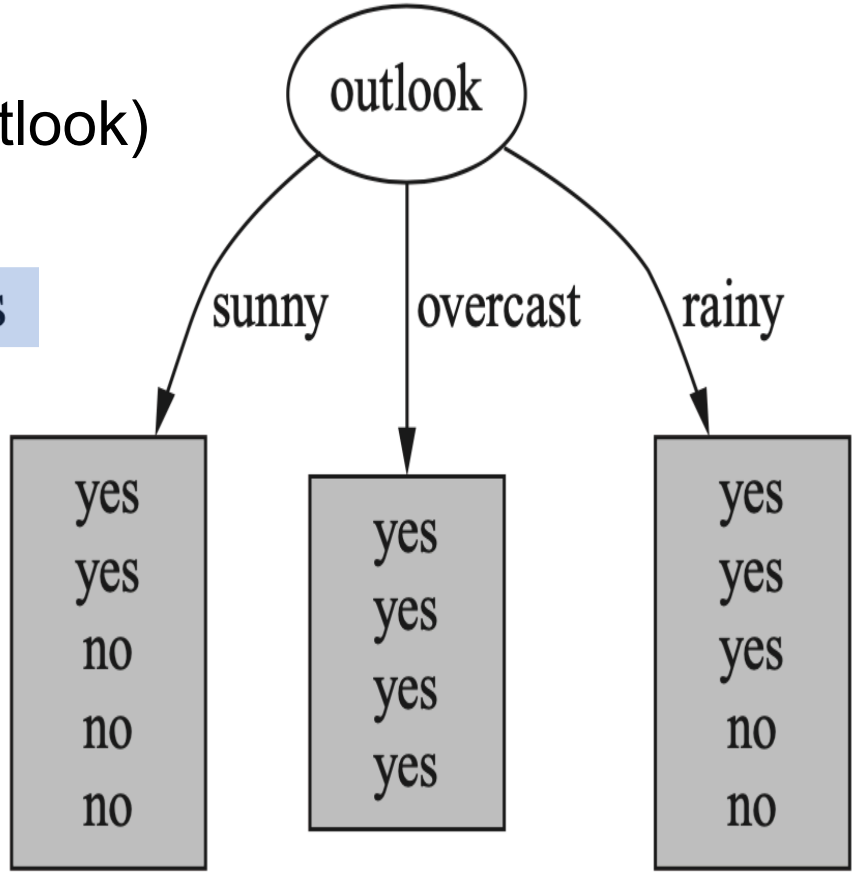
$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A).$$

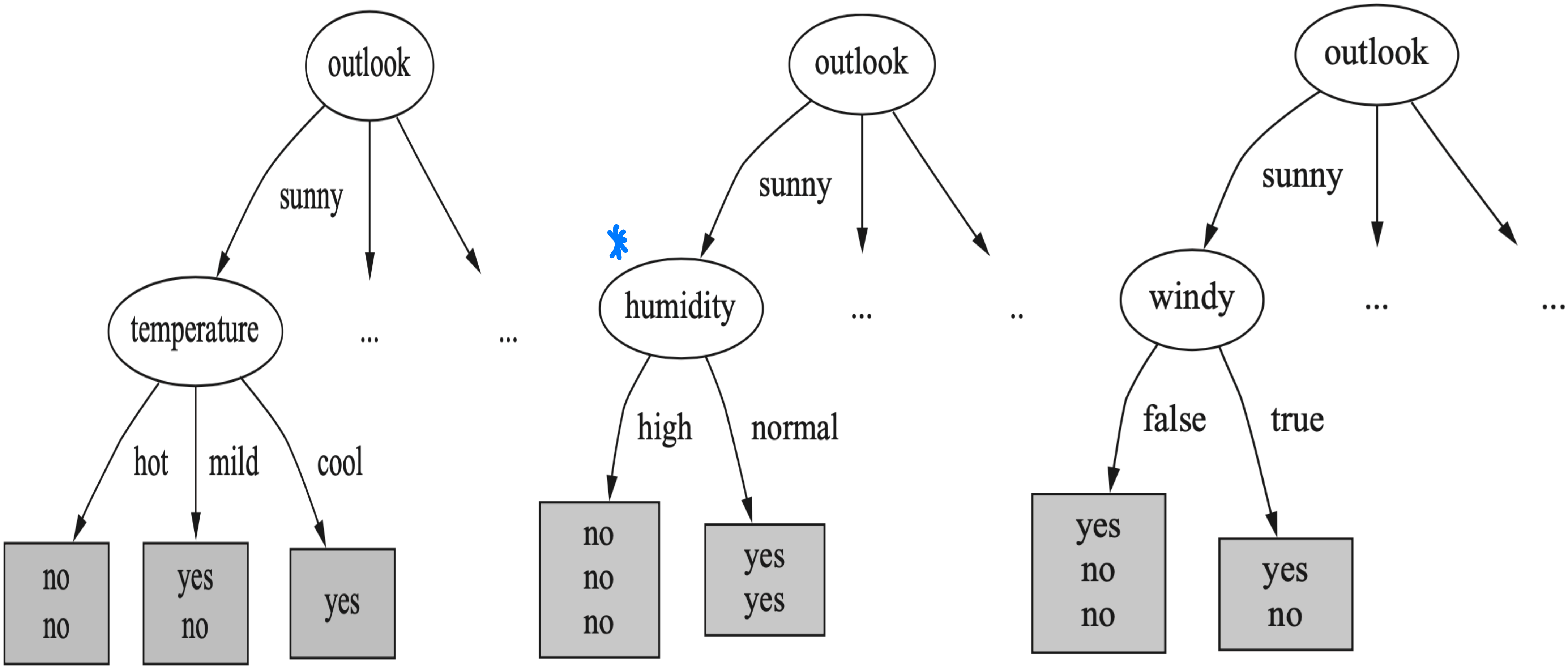
Example: Information Gain

- Gain(outlook) = B($\frac{9}{9+5}$) - Remainder(outlook)

  $= 0.940 - 0.693 \;\; = 0.247$ bits

- Gain(temperature) = 0.029 bits
- Gain(humidity) = 0.152 bits
- Gain(windy) = 0.048 bits





$$\text{gain}(\textit{temperature}) = 0.571 \text{ bits}$$
$$\text{gain}(\textit{humidity}) = 0.971 \text{ bits}$$
$$\text{gain}(\textit{windy}) = 0.020 \text{ bits}$$

Performance Metrics for Classification Models

- **Confusion matrix**
  - Example: two-class prediction

		Predicted Class	
		yes	*no*
Actual Class	*yes*	true positive (TP)	false negative(FN)
	no	false positive (FP)	true negative (TN)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

(Sensitivity)

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$