CS5841/EE5841 Machine Learning

Lecture 4: Classification Part 1

Evan Lucas

Overview

- Course updates
- Classification part 2
  - Gradient descent/ascent methods
  - Naive Bayes
  - SVM
- Classification part 3
  - Decision Trees
  - Ensemble methods

Class updates

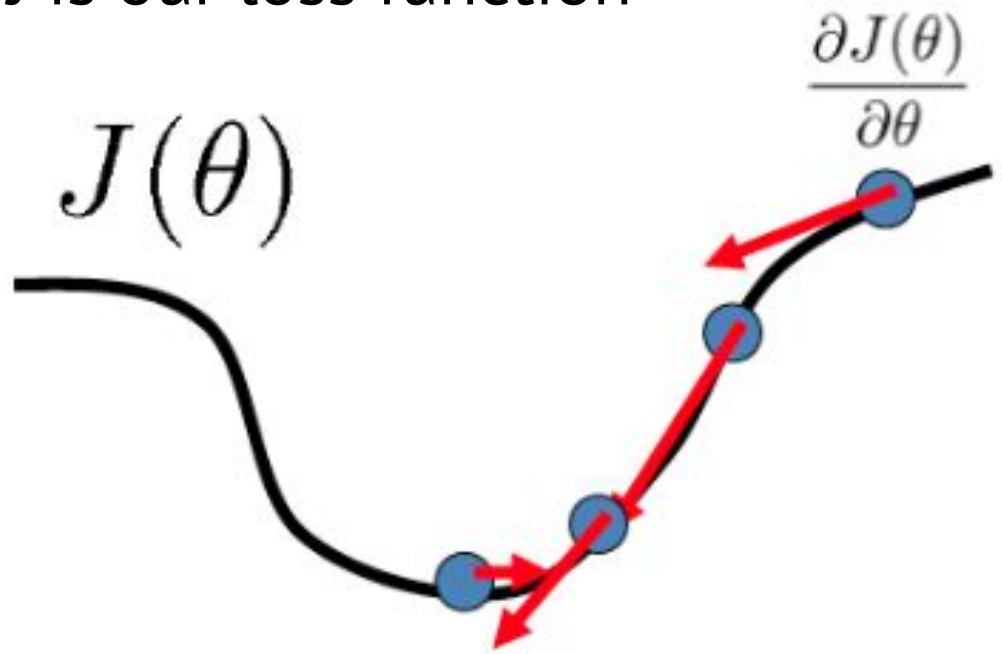
- HW1 hints sent out
- HW2 released

Gradient descent

- Iteratively update weights
- How do we determine step size?
  - Should step size be constant?

$$\theta \leftarrow \theta - \alpha \frac{d}{d\theta} J(\theta)$$

J is our loss function



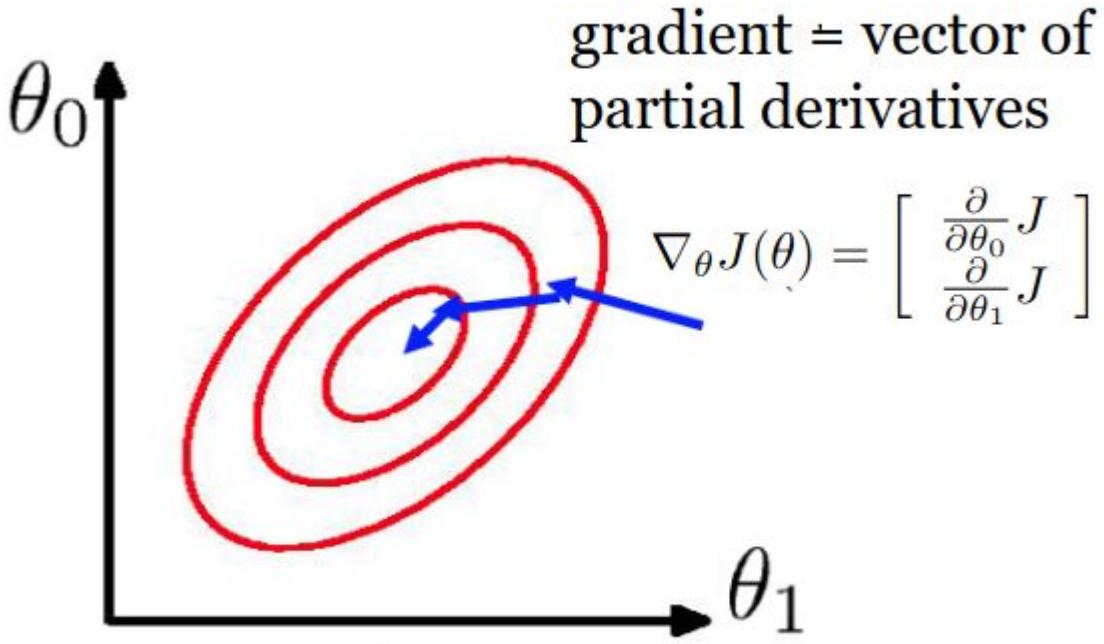
Gradient ascent?

- Same thing, just flip the sign
  - Sometimes researchers flip signs by accident…
- Loss - we want to minimize
- Likelihood - maximize

Extending gradient descent to 2D

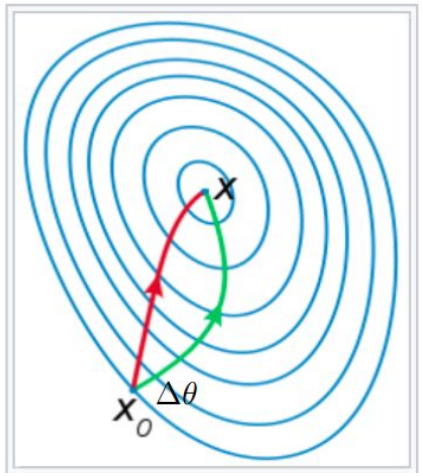
The pattern continues for higher dimensions

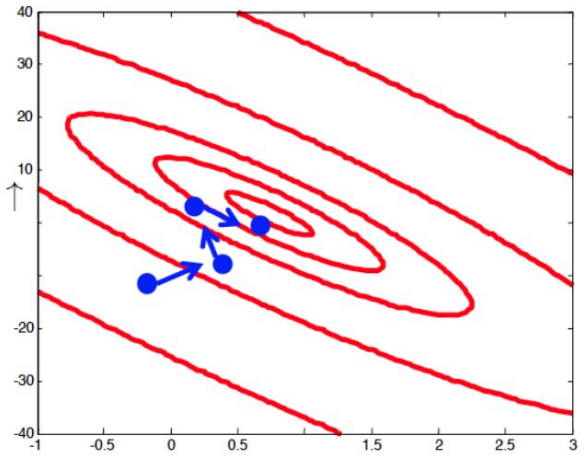
$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$$



Gradient Descent vs. Stochastic Gradient Descent

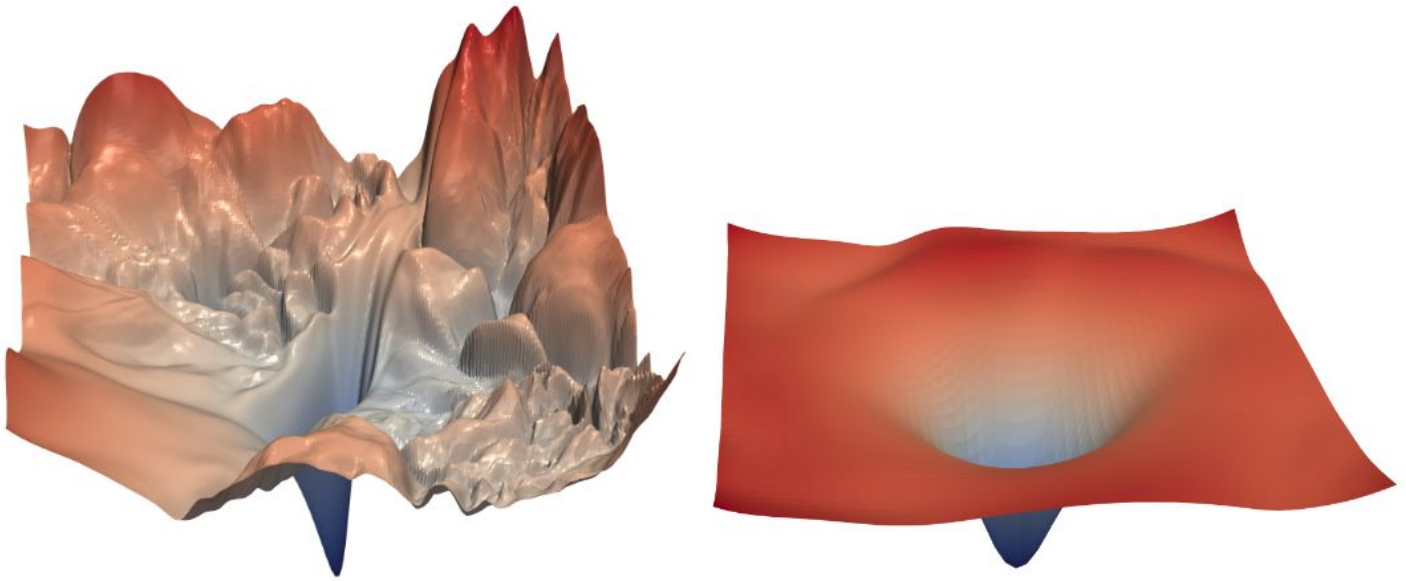
- SGD takes a 'noisier' path due to less information being contained in each example
- Good optimizers take second order derivative into account for a faster (red in left plot)





Loss Landscapes

- Many dimensions - but we can project into 3 to visualize



(a) without skip connections

(b) with skip connections

https://arxiv.org/pdf/1712.09913.pdf

More loss landscapes with learning trajectories

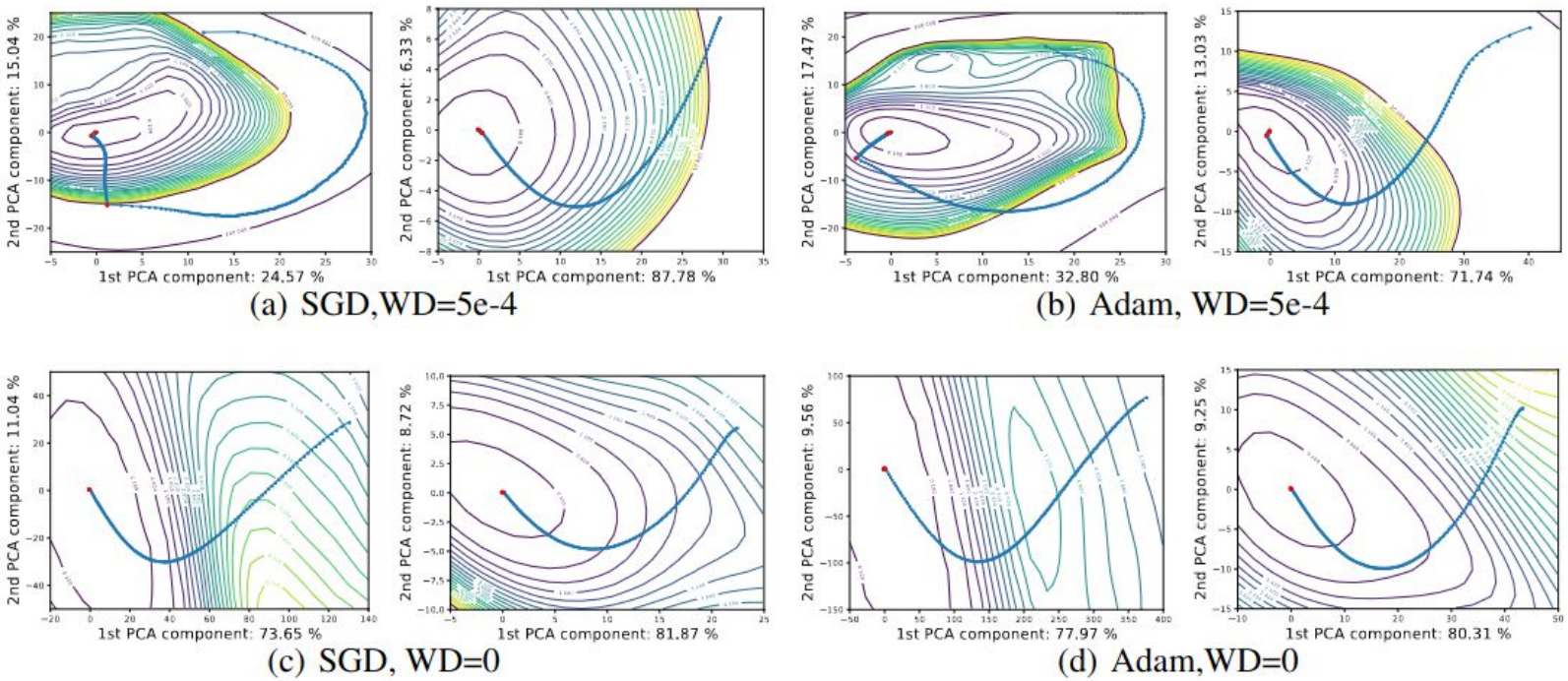
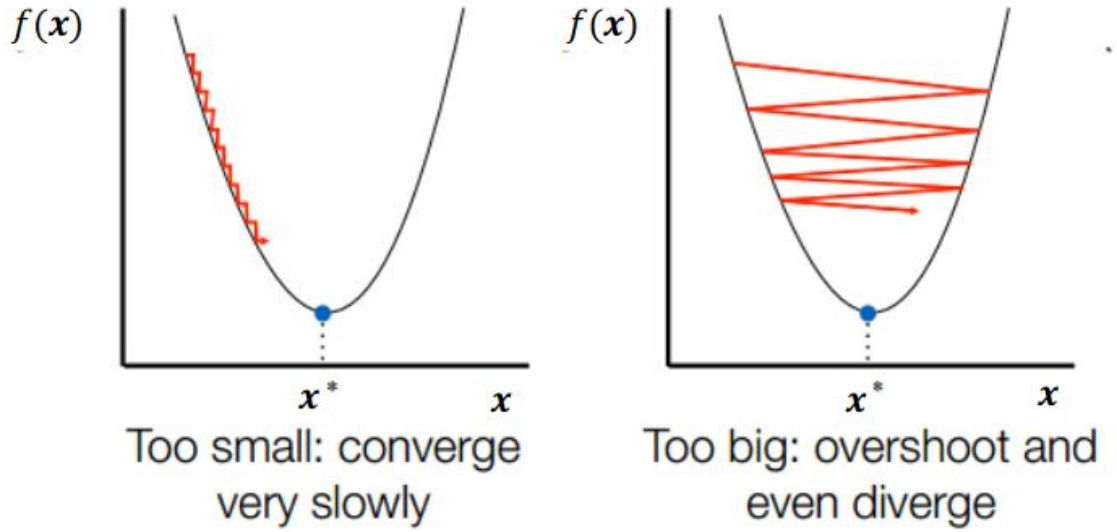


Figure 9: Projected learning trajectories use normalized PCA directions for VGG-9. The left plot in each subfigure uses batch size 128, and the right one uses batch size 8192.

https://arxiv.org/pdf/1712.09913.pdf

Choosing step size

- We usually decay step size to ensure convergence
  - Paper on this here: https://arxiv.org/abs/2102.09393



Reading for Naive Baye

Bishop 4.2

ESL Ch 4

Generative and Discriminative models

Discriminative Models

- Model P(y|x)
- Pros
  - Usually good performance
- Cons
  - Slow convergence
  - Expensive computation
  - Sensitive to noise data

Generative Models

- Model P(x|y)
- Pros
  - Usually fast converge
  - Cheap computation
  - Robust to noise data
- Cons
  - Usually performs worse

Generative Models

- Given a class, what is the distribution of features?
- Examples:
  - Naive Bayes
  - Linear Discriminant Analysis
- Model the joint probability distribution

Probability Terminology

Name	What it is	Common Symbols	What it means
Sample Space	Set	Ω, S	"Possible outcomes."
Event Space	Collection of subsets	$\mathcal{F}, E$	"The things that have probabilities.."
Probability Measure	Measure	P, π	Assigns probabilities to events.
Probability Space	A triple	$(\Omega, \mathcal{F}, P)$	

Probability Terminology

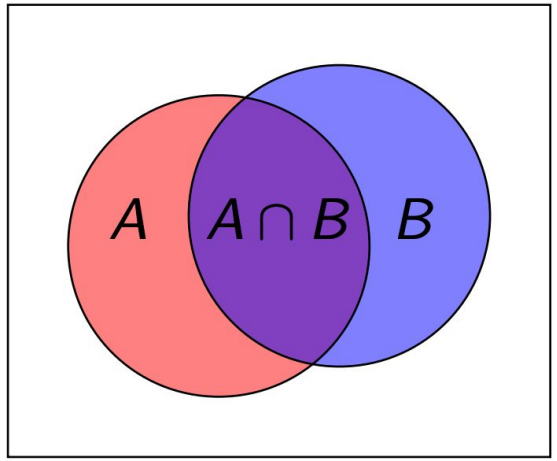
Name	What it is	Common Symbols	What it means
Sample Space	Set	Ω, S	"Possible outcomes."
Event Space	Collection of subsets	$\mathcal{F}, E$	"The things that have probabilities.."
Probability Measure	Measure	P, π	Assigns probabilities to events.
Probability Space	A triple	$(\Omega, \mathcal{F}, P)$	

- rolling a fair die

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$
$$\mathcal{F} = 2^{\Omega} = \{\{1\}, \{2\} \ldots \{1, 2\} \ldots \{1, 2, 3\} \ldots \{1, 2, 3, 4, 5, 6\}, \{\}\}$$

$P(\{1\}) = P(\{2\}) = \ldots = \frac{1}{6}$ (i.e., a fair die)
$P(\{1, 3, 5\}) = \frac{1}{2}$ (i.e., half chance of odd result)
$P(\{1, 2, 3, 4, 5, 6\}) = 1$ (i.e., result is "almost surely" one of the faces).

Conditional Probabilities



For events $A, B \in \mathcal{F}$ with $P(B) > 0$, we may write the **conditional probability of A given B**:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Interpretation: the outcome is definitely in B, so treat B as the entire sample space and find the probability that the outcome is also in A.

This rapidly leads to: $P(A|B)P(B) = P(A \cap B)$ aka the "chain rule for probabilities." (why?)

When $A_1, A_2 \ldots$ are a partition of Ω:

$$P(B) = \sum_{i=1}^{\infty} P(B \cap A_i) = \sum_{i=1}^{\infty} P(B|A_i)P(A_i)$$

This is also referred to as the "law of total probability."

Conditional Probabilities - Example

Suppose we throw a fair die:
$\Omega = \{1, 2, 3, 4, 5, 6\},\ \mathcal{F} = 2^{\Omega},\ P(\{i\}) = \frac{1}{6},\ i = 1 \ldots 6$
$A = \{1, 2, 3, 4\}$ i.e., "result is less than 5,"
$B = \{1, 3, 5\}$ i.e., "result is odd."

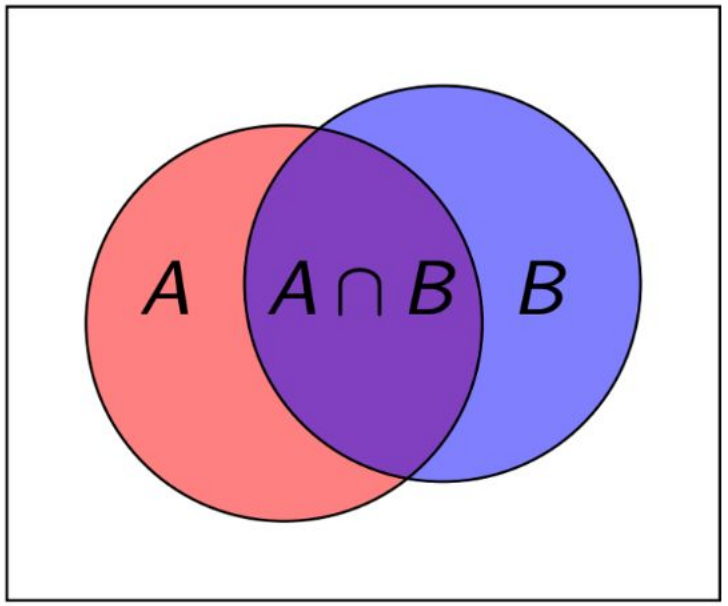
$$P(A) = \frac{2}{3}$$
$$P(B) = \frac{1}{2}$$

Conditional Probabilities - Example

Suppose we throw a fair die:
$\Omega = \{1, 2, 3, 4, 5, 6\},\ \mathcal{F} = 2^{\Omega},\ P(\{i\}) = \frac{1}{6},\ i = 1 \ldots 6$
$A = \{1, 2, 3, 4\}$ i.e., "result is less than 5,"
$B = \{1, 3, 5\}$ i.e., "result is odd."

$$
\begin{aligned}
P(A) &= \frac{2}{3} \\
P(B) &= \frac{1}{2} \\
P(A|B) &= \frac{P(A \cap B)}{P(B)} \qquad\qquad P(B|A) \\
&= \frac{P(\{1, 3\})}{P(B)} \\
&= \frac{2}{3}
\end{aligned}
$$

Bayes rule



Thomas Bayes

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

$$= \frac{P(B|A)P(A)}{P(B)}$$

Corollary: The chain rule

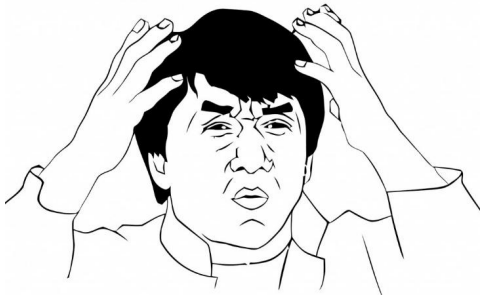
$$P(A,B) = P(A|B)P(B) = \mathrm{P(B)P(A|B)}$$

Other forms of Bayes rule

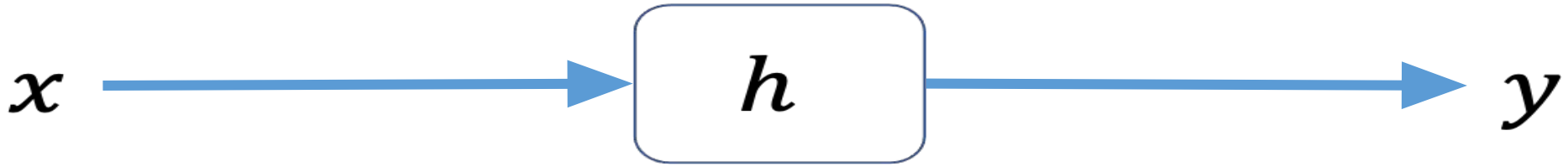
- $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B,X) = \frac{P(B|A,X)P(A,X)}{P(B,X)}$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\sim A)P(\sim A)}$$



Why we are learning this?



Hypothesis

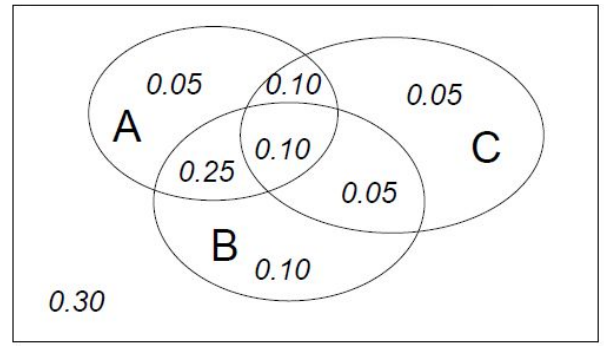
Learn $P(Y|X)$

Joint distribution

- Making a joint distribution of M variables

1. Make a truth table listing all combinations
2. For each combination of values, say how probable it is
3. Probability must sum to 1

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



Using joint distribution

- Can ask for any logical expression involving these variables

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

- $P(E) = \sum_{\text{rows matching E}} P(\text{row})$
- $P(E_1|E_2) = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$
- Given the joint distribution, we can make inferences
  - E.g., P(Male|Poor)? Or P(Wealth | Gender, Hours)?

The solution to learn $P(Y|X)$?

- Main problem: learning $P(Y|X)$ may require more data than we have

- Say, learning a joint distribution with 100 boolean attributes

- # of rows in this table?

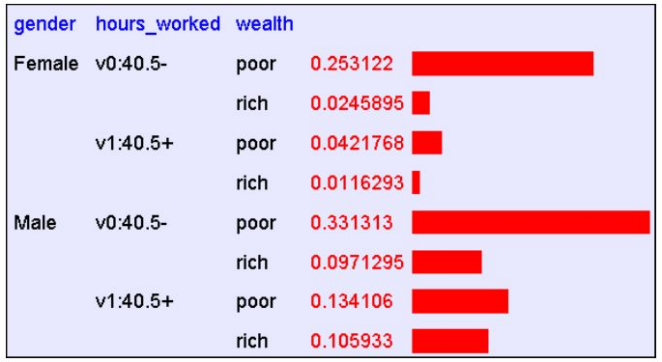
$$2^{100} \geq 10^{30}$$

- # of people on earth?

$$10^{9}$$

How many parameters must we estimate?

- Suppose $X = [X_1, \cdots, X_n]$, where X_i and Y are Boolean random variables



To estimate $P(Y|X_1, \cdots, X_n)$

P(wealth | gender, hours_worked)

Gender	HrsWorked	P(rich | G,HW)	P(poor | G,HW)
F	<40.5	.09	.91
F	>40.5	.21	.79
M	<40.5	.23	.77
M	>40.5	.38	.62

When $n = 2$ (Gender, Hours-worked)?

4

When $n = 30$?

2^30

Can we reduce parameters using Bayes rule?

- $$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$
- Assume, X and Y are binary random variable
- How many parameters do we need to estimate $P(Y)$?

$$1$$

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

- How many parameters for $P(X_1, \cdots, X_n|Y)$?

$$(2^n - 1)\times 2$$

Bonus point

Naïve Bayes

- Assumption:

$$P(X_1, \cdots, X_n|Y) = \prod_{j=1}^{n} P(X_j|Y)$$

- i.e., X_i and X_j are <u>conditionally independent</u> given Y for $i \neq j$

Conditional independence

- **Definition**: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

$$(\forall i, j, k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain, Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Independent vs Conditional independent

- Two variables X, Y are **independent** if
  - $\forall i, j: P(X = x_i, Y = y_j) = P(X = x_i)P(Y = y_j)$
  - $P(XY) = P(X)P(Y)$

- Two variables X, Y are **conditional independent** give Z if

$$\forall i, j, k: P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Applying conditional independence

- Naïve Bayes assumes X_i are conditionally independent given Y
- e.g., $P(X_1|X_2, Y) = P(X_1|Y)$
- $P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$
  $= P(X_1|Y)P(X_2|Y)$

Chain rule

Conditional Independence

General form: $P(X_1, \cdots, X_n|Y) = \prod_{j=1}^{n} P(X_j|Y)$

How many parameters to describe $P(X_1, \cdots, X_n|Y)$? $P(Y)$?

- Without conditional indep assumption? $(2^n-1) * 2 + 1$
- With conditional indep assumption? **2n+1**

Bonus point

Naïve Bayes classifier

- Bayes rule:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) P(X_1, \cdots, X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1, \cdots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i's:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) \Pi_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \Pi_i P(X_i | Y = y_j)}$$

- Pick the most probable Y

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Naïve Bayes algorithm – discrete X_i

- For each value y_k
  - Estimate $\pi_k = P(Y = y_k)$
  - For each value x_{ij} of each attribute X_i
    - Estimate $\theta_{ijk} = P(X_i = x_{ij} | Y = y_k)$

$$\frac{\#\ examples\ for\ which\ Y = y_k}{\#\ examples}$$

$$\frac{\#\ examples\ for\ which\ X_i = x_{ij}\ and\ Y = y_k}{\#\ examples\ for\ which\ Y = y_k}$$

- Classify X^{test}

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, P(Y = y_k) \Pi_i P\left(X_i^{test} | Y = y_k\right)$$

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, \pi_k\, \Pi_i \theta_{ijk}$$

Naïve Bayes algorithm - Example

The weather data, with counts and probabilities												
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								

A new day				
outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}} P(Y = y_k) \Pi_i P(X_i^{\text{test}} | Y = y_k)$$

Naïve Bayes algorithm - Example

The weather data, with counts and probabilities

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								

A new day

outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

- Likelihood of yes
  - $\frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = 0.0053$
- Likelihood of no
  - $\frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0.0206$

the prediction is No

Issues with Naïve Bayes - 1

- Often the X_i are not really conditionally independent

- Naïve Bayes often works pretty well anyway
  - Often the right classification, even when not the right probability [Domingos & Pazzani, 1996])

Issues with Naïve Bayes - 2

- $P(Y = y_k | X_1, \cdots, X_n) \propto P(Y = y_k) \Pi_i P(X_i | Y = y_k)$

What if $P(X_i | Y = y_k) = 0$?

- What can we do to address this?
  - Laplace Smoothing

Laplace Smoothing

- Adds arbitrary low probabilities in such cases so that the probability computation does not become zero.
- Basic Idea: Pretend that you saw every feature-class outcome pair λ extra times.

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k)$$
$$= \frac{\#D\{X_i = x_{ij}, Y = y_k\} + \lambda}{\#D\{Y = y_k\} + \lambda|values(X_i)|}$$

λ: a small number (e.g., 1)

What if we have continuous $\mathbf{X_i}$

- **Gaussian Naïve Bayes (GNB): assume**

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left(-\frac{(x - \mu_{ik})^2}{2\sigma_{ik}^2}\right)$$

- **Additional assumption on σ_{ik}:**
  - Is independent of Y (σ_i)
  - Is independent of X_i (σ_k)
  - Is independent of X_i and Y (σ_k)

Naïve Bayes algorithm – continuous X_i

- For each value y_k

  Estimate $\pi_k = P(Y = y_k)$

  For each attribute X_i estimate

  Class conditional mean μ_{ik}, variance σ_{ik}

- Classify X^{test}

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, P(Y = y_k)\Pi_i P\left(X_i^{test} \middle| Y = y_k\right)$$

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, \pi_k\, \Pi_i \quad Normal(X_i^{test}, \mu_{ik}, \sigma_{ik})$$

Summary

- Probability basics
- Naive Bayes

$$P(Y = y_k | X_1, \cdots, X_n) \propto P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

- Advantages
  - Fast to train (just one scan of database) and classify
  - Not sensitive to irrelevant features
  - Handle discrete data well

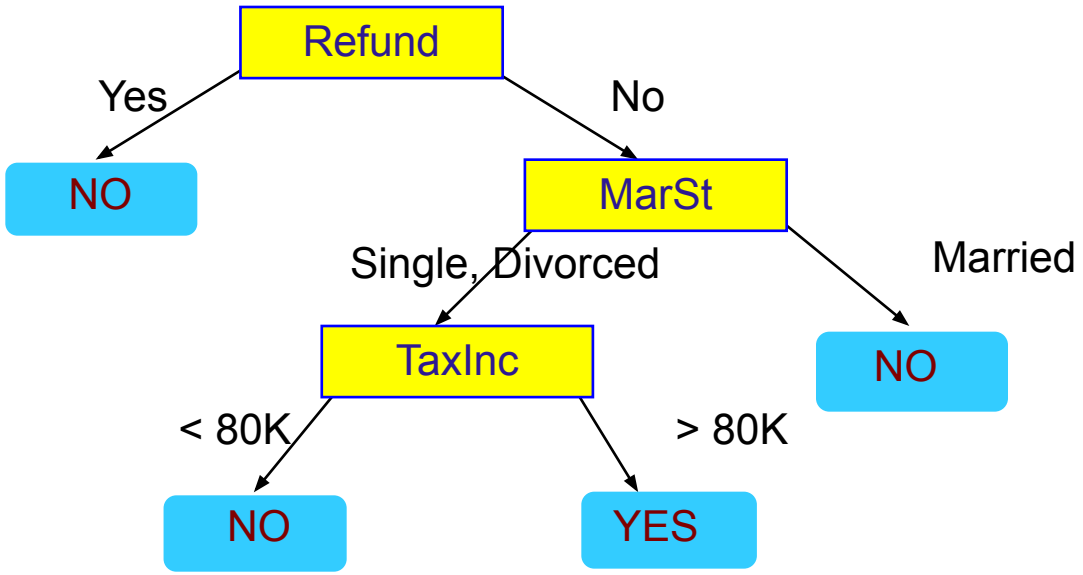
- Disadvantage
  - Assume conditional independence of features - Losing the accuracy

Decision Trees

Example of a Decision Tree

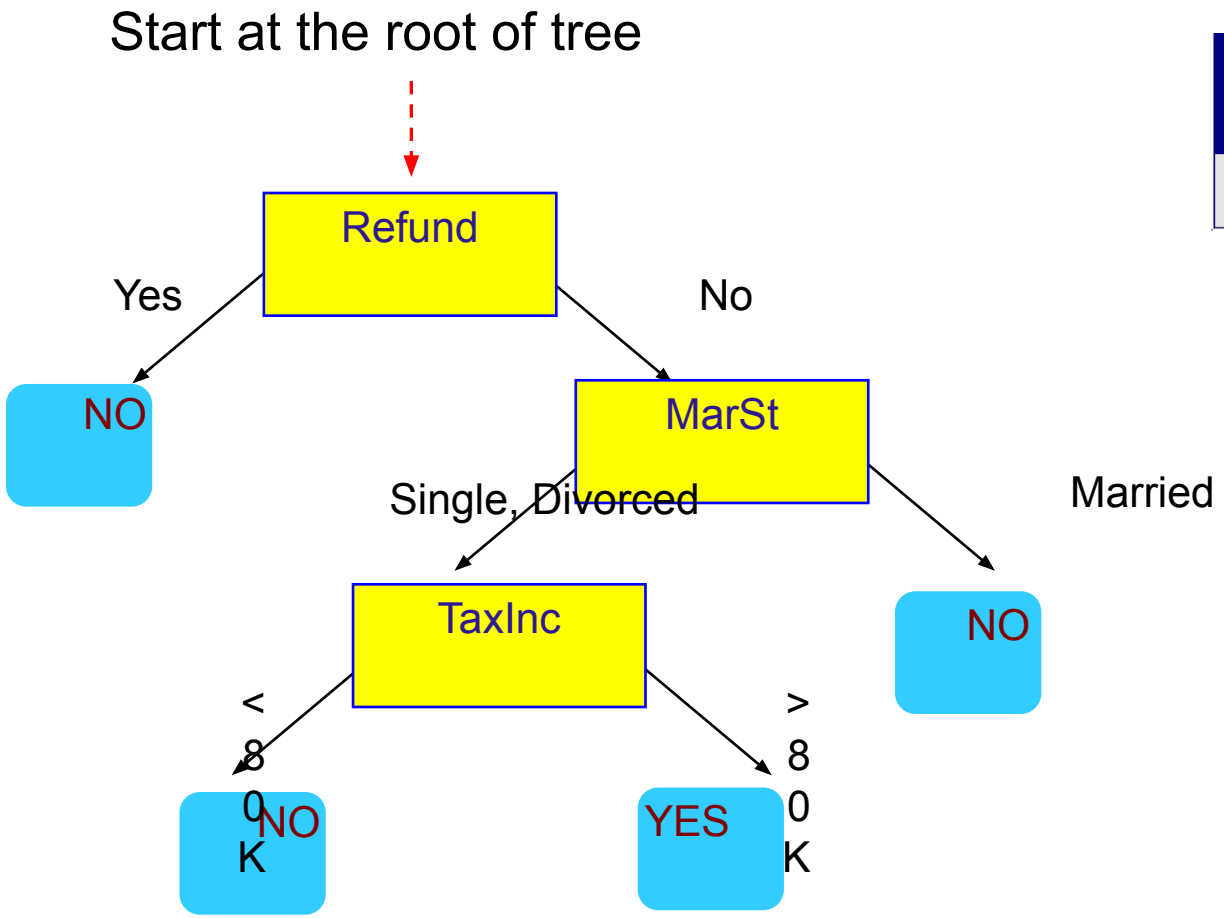
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Apply Model to Test Data



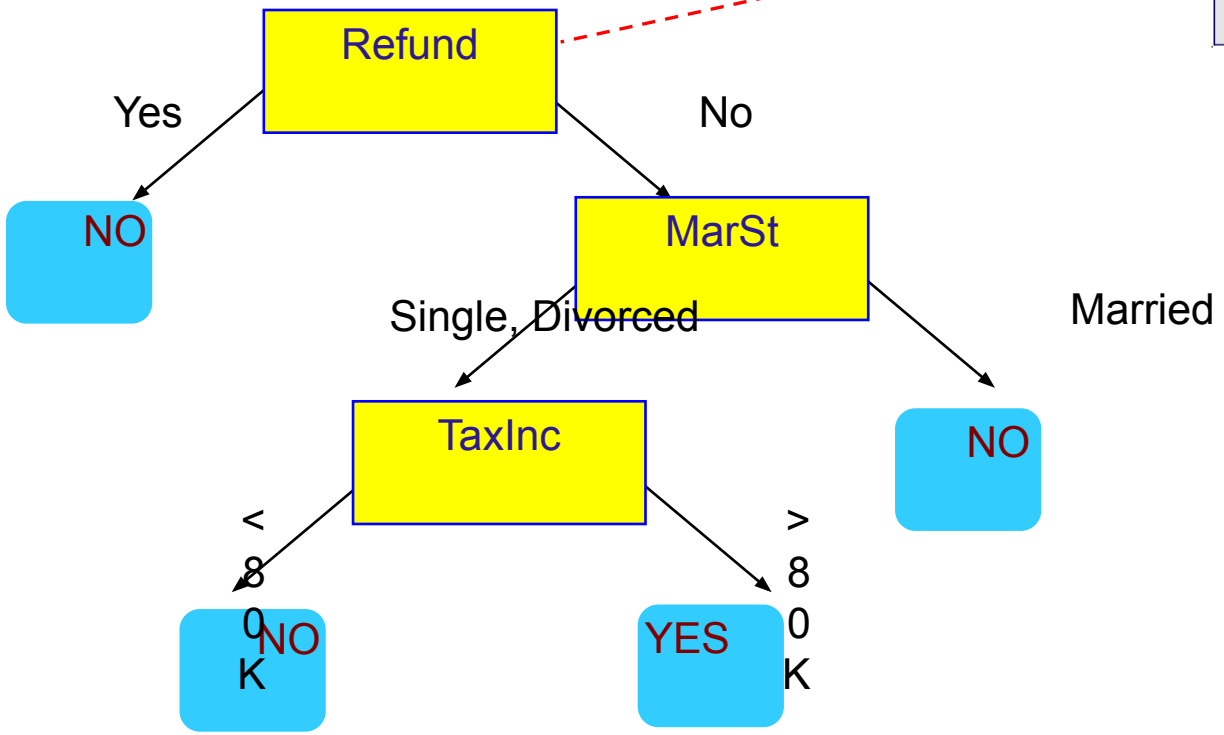
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

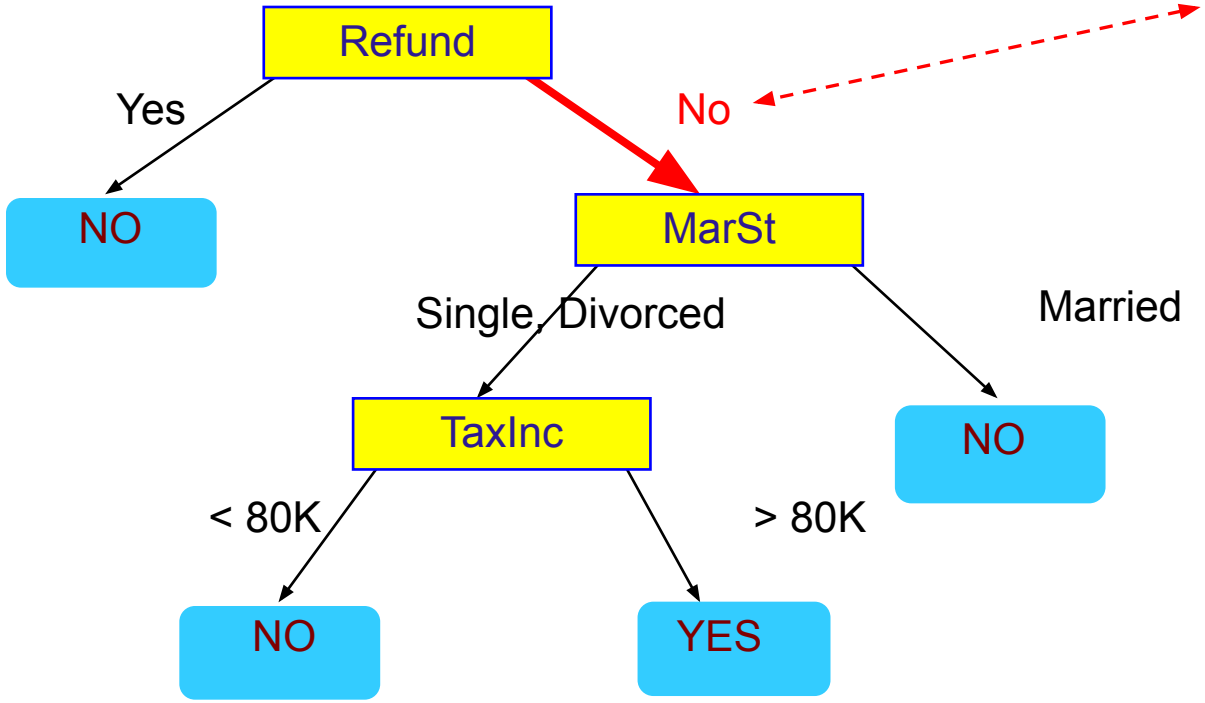
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

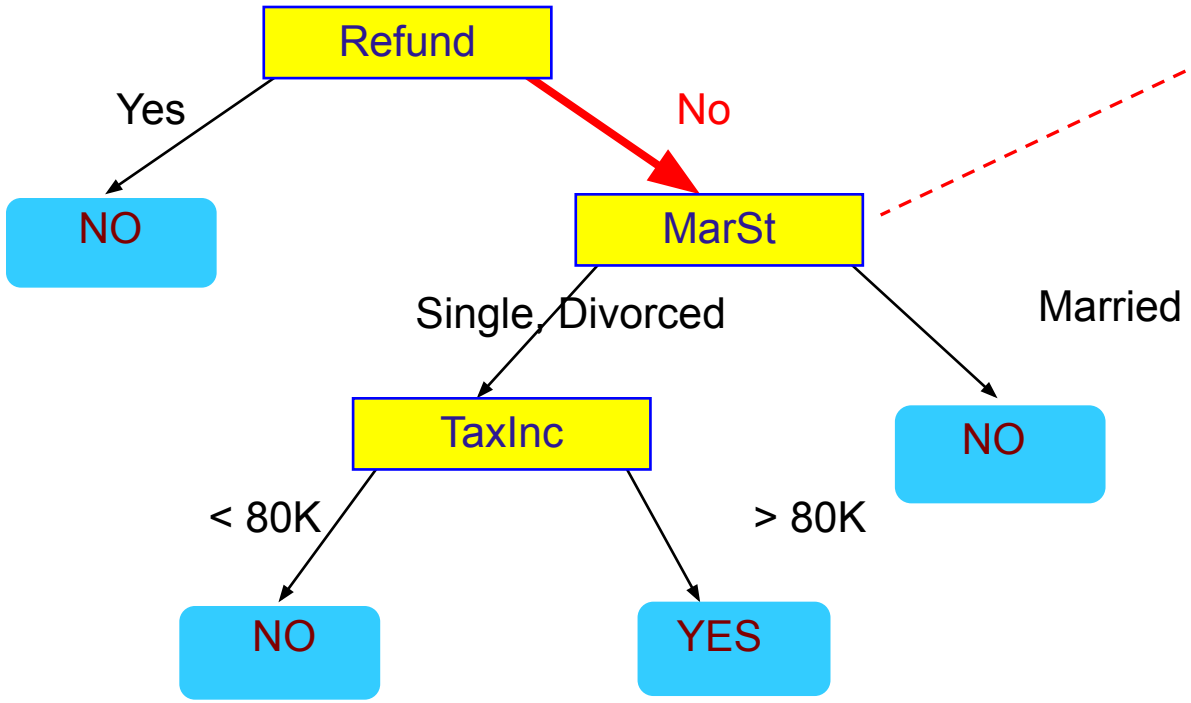
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

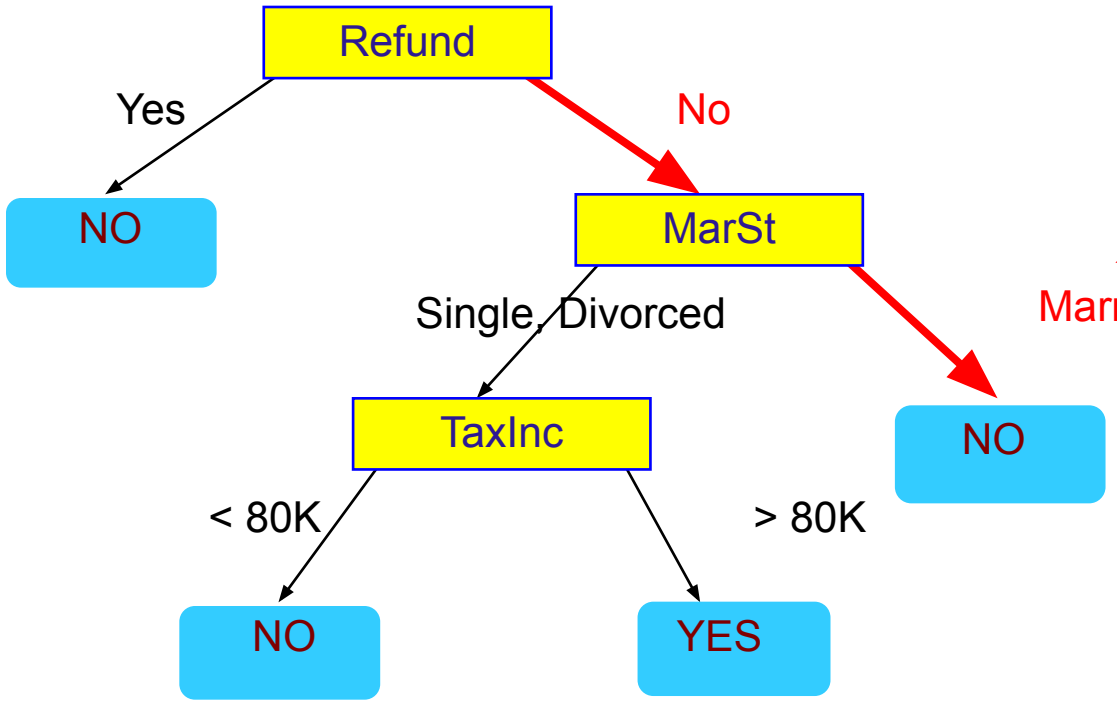
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

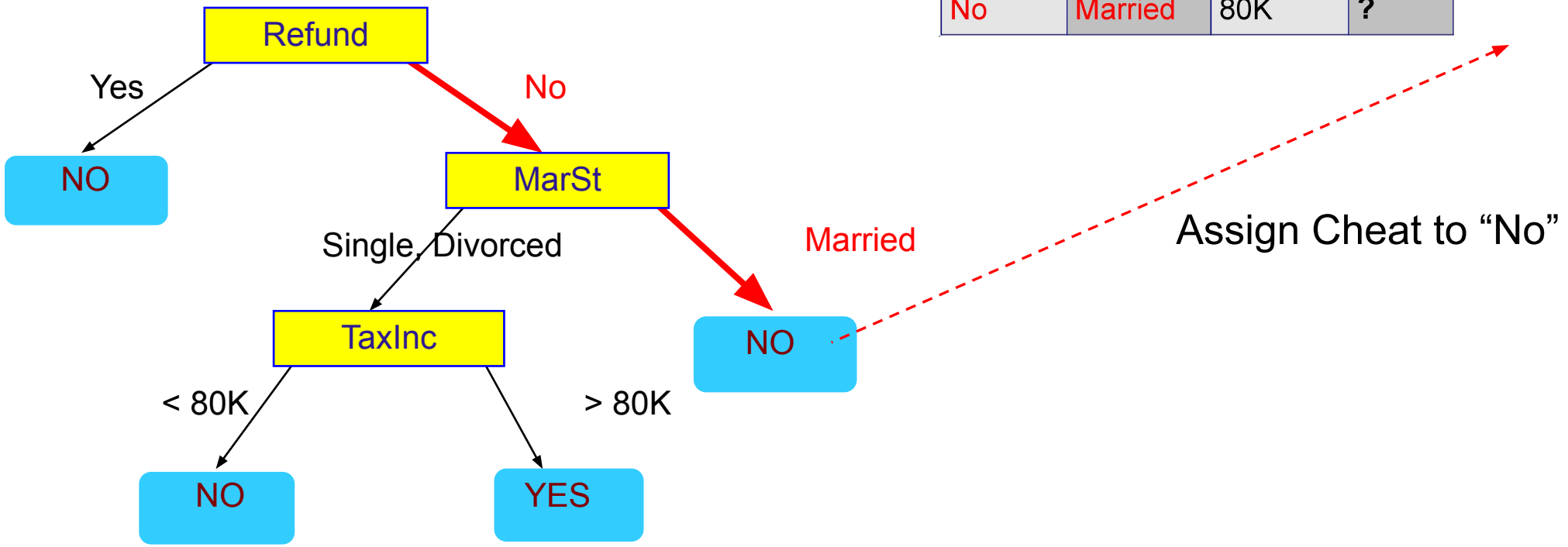
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decision Tree Representation

- Each internal node is a test:
  - Theoretically, a node can test multiple attributes
  - In most systems, a node tests exactly one attribute
- Each branch corresponds to test results
  - A branch corresponds to an attribute value or a range of attribute values
- Each leaf node assigns
  - a class: decision tree
  - a real value: regression tree

Decision Tree Learning: ID3

- Function ID3(*TrainingSet*, *Attributes*)
  - If all elements in *TrainingSet* are in same class, then return leaf node labeled with th
  - Else if *Attributes* is empty, then return leaf node labeled with majority class in *Traini*
  - Else if *TrainingSet* is empty, then return leaf node labeled with default majority class
  - Else
    - Select and remove *A* from *Attributes*
    - Make *A* the root of the current tree
    - For each value *V* of *A*
      - Create a branch of the current tree labeled by *V*
      - *Partition_V* ← Elements of *TrainingSet* with value *V* for *A*
      - Induce-Tree(*Partition_V*, *Attributes*)
      - Attach result to branch *V*

Illustrative Training Set

Risk Assessment for Loan Applications

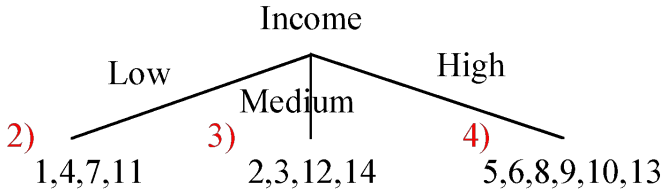
Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**

ID3 Example (I)

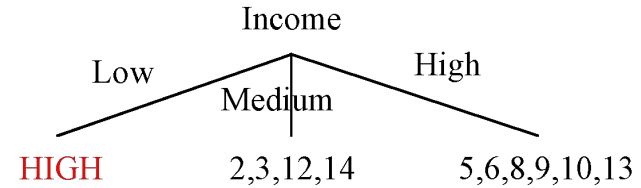
Risk Assessment for Loan Applications

Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	HIGH
2	Unknown	High	None	Medium	HIGH
3	Unknown	Low	None	Medium	MODERATE
4	Unknown	Low	None	Low	HIGH
5	Unknown	Low	None	High	LOW
6	Unknown	Low	Adequate	High	LOW
7	Bad	Low	None	Low	HIGH
8	Bad	Low	Adequate	High	MODERATE
9	Good	Low	None	High	LOW
10	Good	High	Adequate	High	LOW
11	Good	High	None	Low	HIGH
12	Good	High	None	Medium	MODERATE
13	Good	High	None	High	LOW
14	Bad	High	None	Medium	HIGH

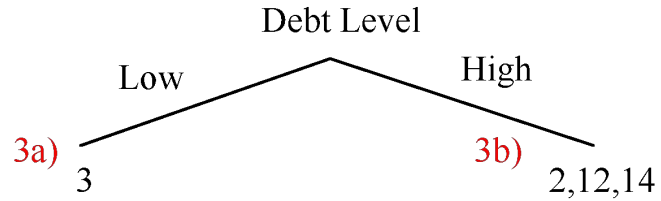
1) Choose Income as root of tree.



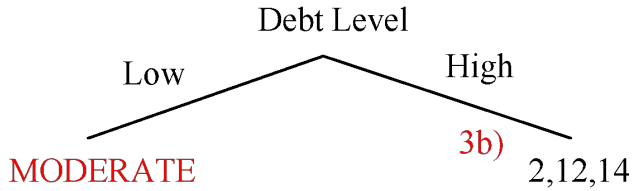
2) All examples are in the same class, HIGH.
Return Leaf Node.



3) Choose Debt Level as root of subtree.



3a) All examples are in the same class, MODERATE.
Return Leaf node.

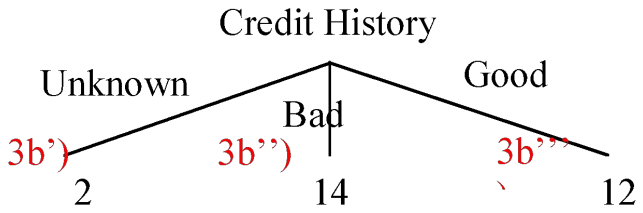


ID3 Example (II)

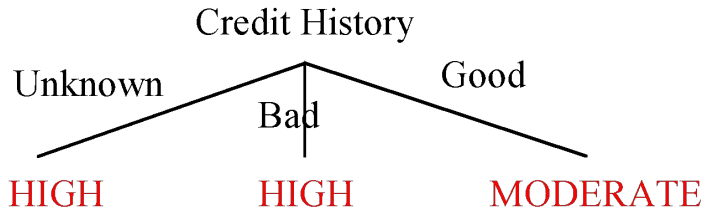
Risk Assessment for Loan Applications

Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	HIGH
2	Unknown	High	None	Medium	HIGH
3	Unknown	Low	None	Medium	MODERATE
4	Unknown	Low	None	Low	HIGH
5	Unknown	Low	None	High	LOW
6	Unknown	Low	Adequate	High	LOW
7	Bad	Low	None	Low	HIGH
8	Bad	Low	Adequate	High	MODERATE
9	Good	Low	None	High	LOW
10	Good	High	Adequate	High	LOW
11	Good	High	None	Low	HIGH
12	Good	High	None	Medium	MODERATE
13	Good	High	None	High	LOW
14	Bad	High	None	Medium	HIGH

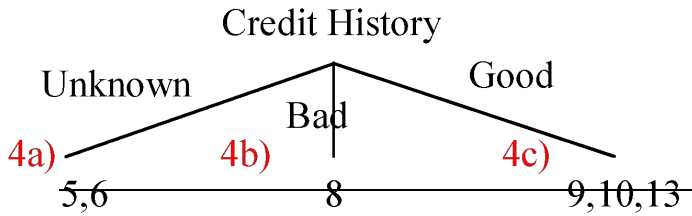
3b) Choose Credit History as root of subtree.



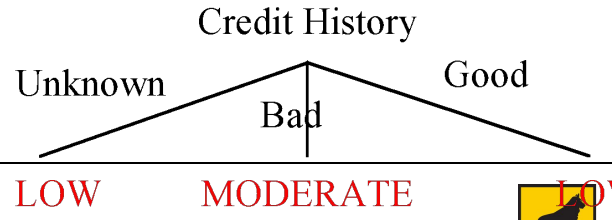
3b'-3b''') All examples are in the same class.
Return Leaf nodes.



4) Choose Credit History as root of subtree.

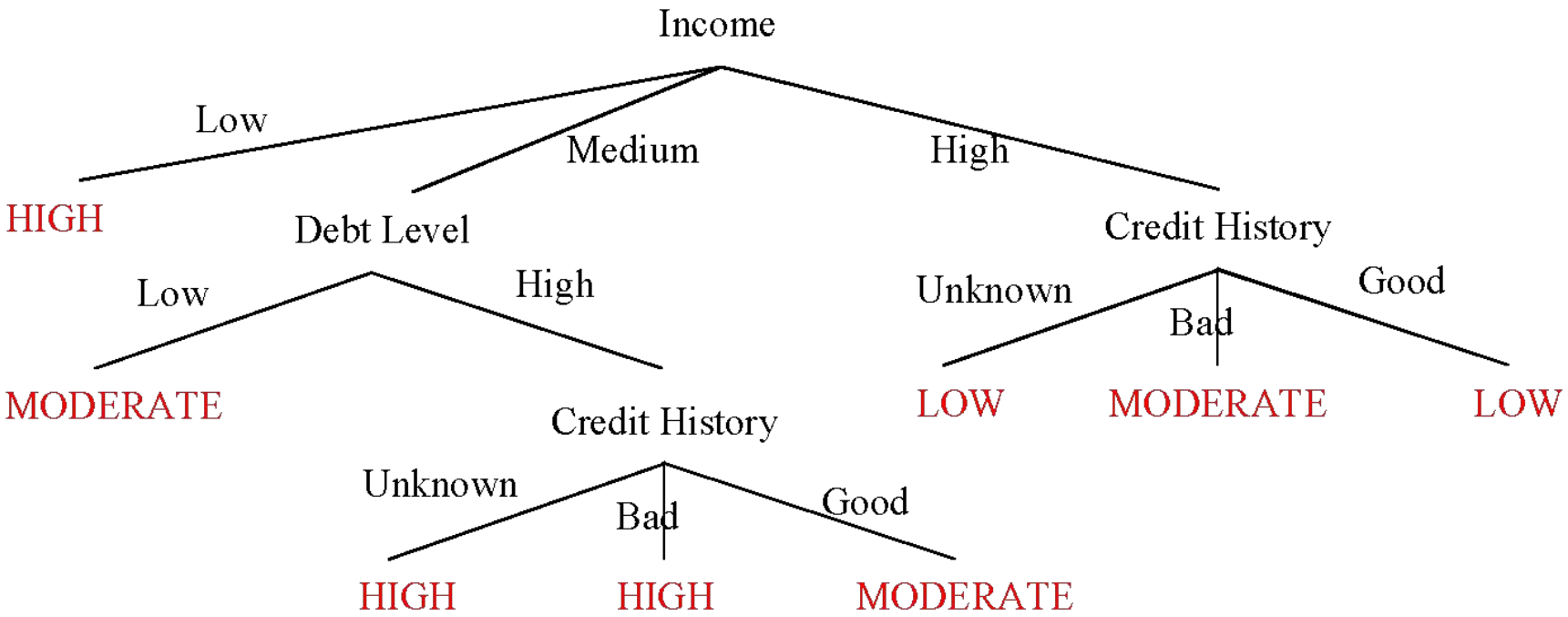


4a-4c) All examples are in the same class.
Return Leaf nodes.



ID3 Example (III)

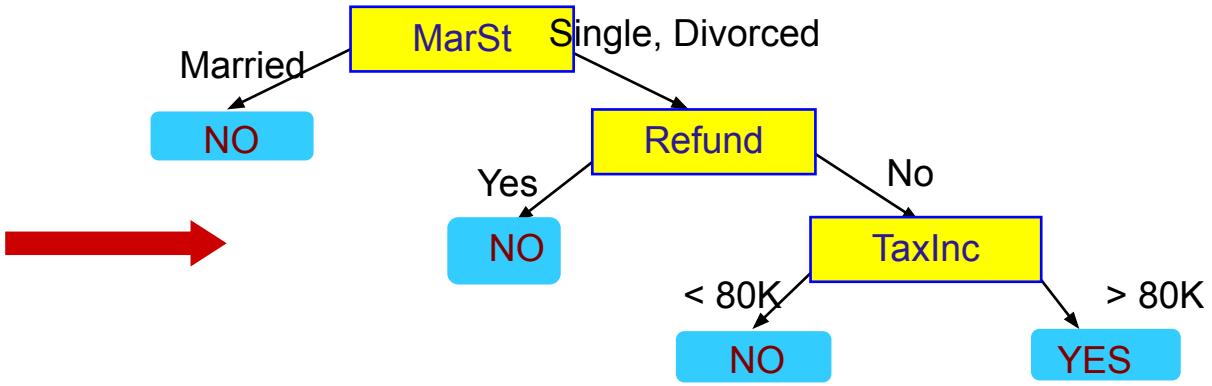
Attach subtrees at appropriate places.



Non-Uniqueness

- Decision trees are not unique:
  - Given a set of training instances *T*, there generally exists a number of decision trees that are consistent with (or fit) *T*

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



ID3's Question

- Given a training set, which of all of the decision trees consistent with that training set should we pick?

- More precisely:
  - Given a training set, which of all of the decision trees consistent with that training set has the greatest likelihood of correctly classifying unseen instances of the population?

ID3's Question

- ID3 (and family) prefers simpler decision trees

- Occam's Razor Principle
  - Always accept the simplest answer that fits the data, avoid unnecessary constraints
  - Simpler trees are more general

ID3's Question

- Since ID3 builds a decision tree by recursively selecting attributes and splitting the training data based on the values of these attributes

- Practically
  - Given a training set, how do we select attributes so that the resulting tree is as small as possible, i.e. contains as few attributes as possible?

Not All Attributes Are Created Equal

- Each attribute of an instance may be thought of as contributing a certain amount of information to its classification
  - Think 20-Questions:
    - What are good questions?
    - Ones whose answers maximize information gained
  - For example, determine shape of an object:
    - Num. sides contributes a certain amount of information
    - Color contributes a different amount of information
- ID3 measures information gained by making each attribute the root of the current subtree, and subsequently chooses the attribute that produces the greatest information gain

Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(where $p(j \mid t)$ is the relative frequency of class j at node t)

- Minimum (0.0) when all records belong to one class, implying most information
- Maximum (log C) when records are equally distributed among all classes, implying least information
- Intuitively, the smaller the entropy the purer the partition

Examples of Computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

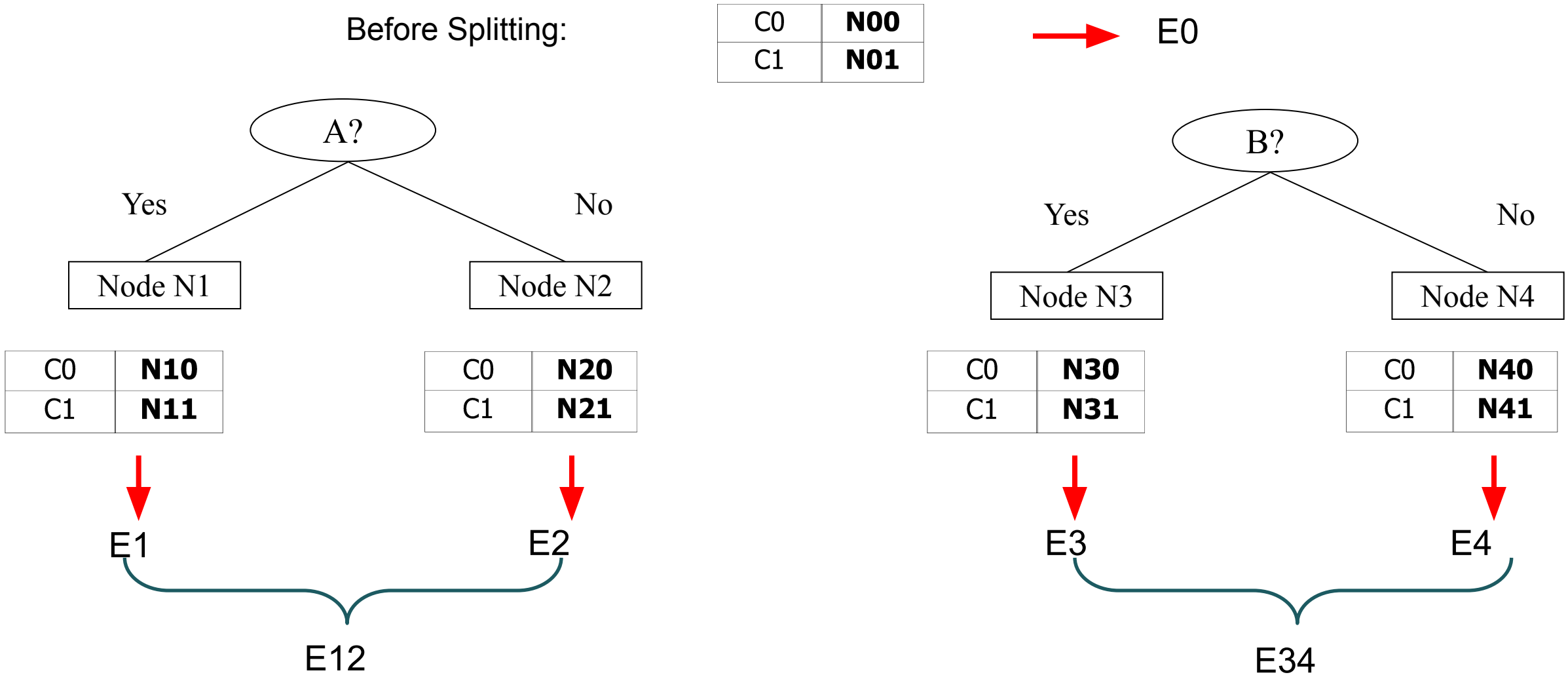
Information Gain

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

(where parent Node p is split into k partitions, and n_i is number of records in partition i)

- Measures how much entropy is reduced because of the split maximize
- ID3 chooses to split on the attribute that results in the largest reduction, i.e., (maximizes GAIN)

Computing Gain



Gain = E0 – E12 vs. E0 – E34

Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

(where parent Node, p is split into k partitions, and n_i is number of records in partition i)

- Designed to overcome the disadvantage of GAIN when attributes with a large number of values
- Adjusts GAIN by the entropy of the partitioning (SplitINFO)
- Higher entropy partitioning (large number of small partitions) is penalized
- Used by C4.5 (an extension of ID3)

Other Splitting Criterion: GINI Index

- GINI Index for a given node t :

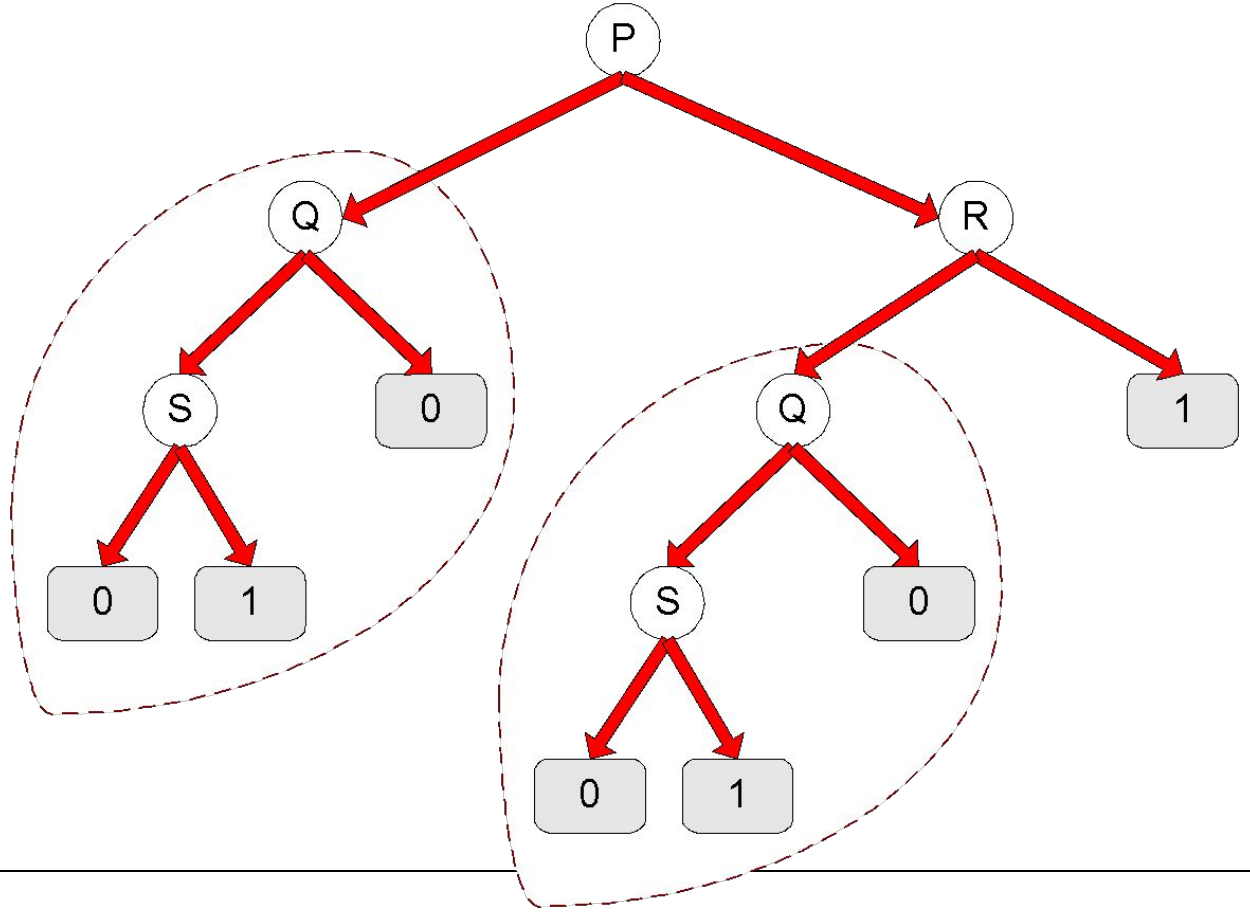
$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

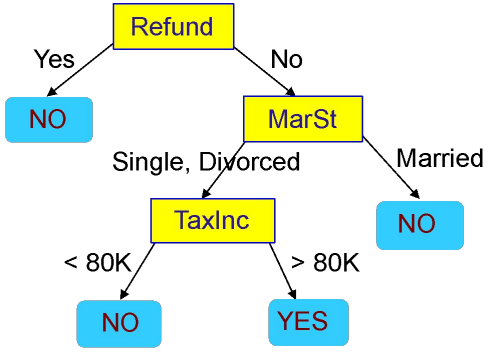
- Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information
- Minimize GINI split value
- Used by CART, SLIQ, SPRINT

Subtree Replication

- Same subtree appears in multiple branches



Summary



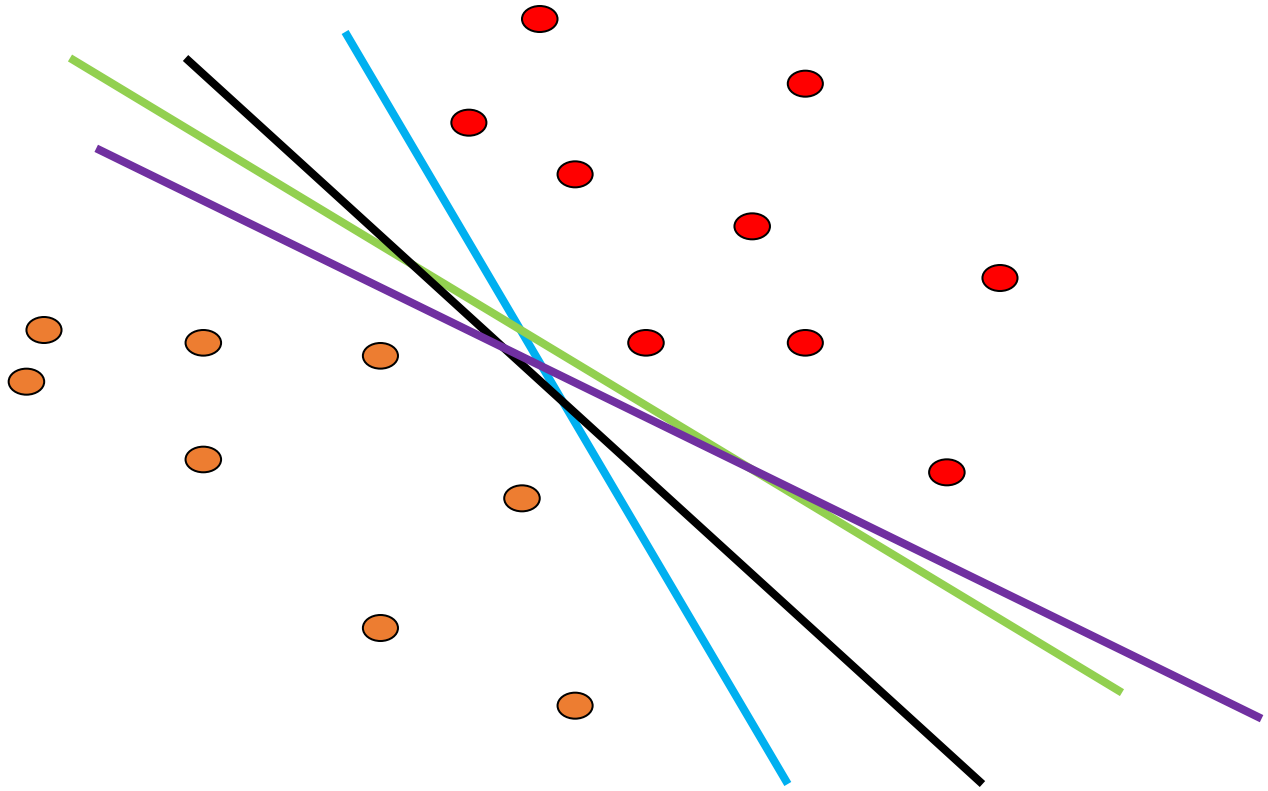
Model: Decision Tree

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
    - Used in Explainable AI (XAI)
- Disadvantages:
  - Redundancy
  - Need data to fit in memory
  - Need to retrain with new data

Support Vector Machines (SVM)

Linear classifiers

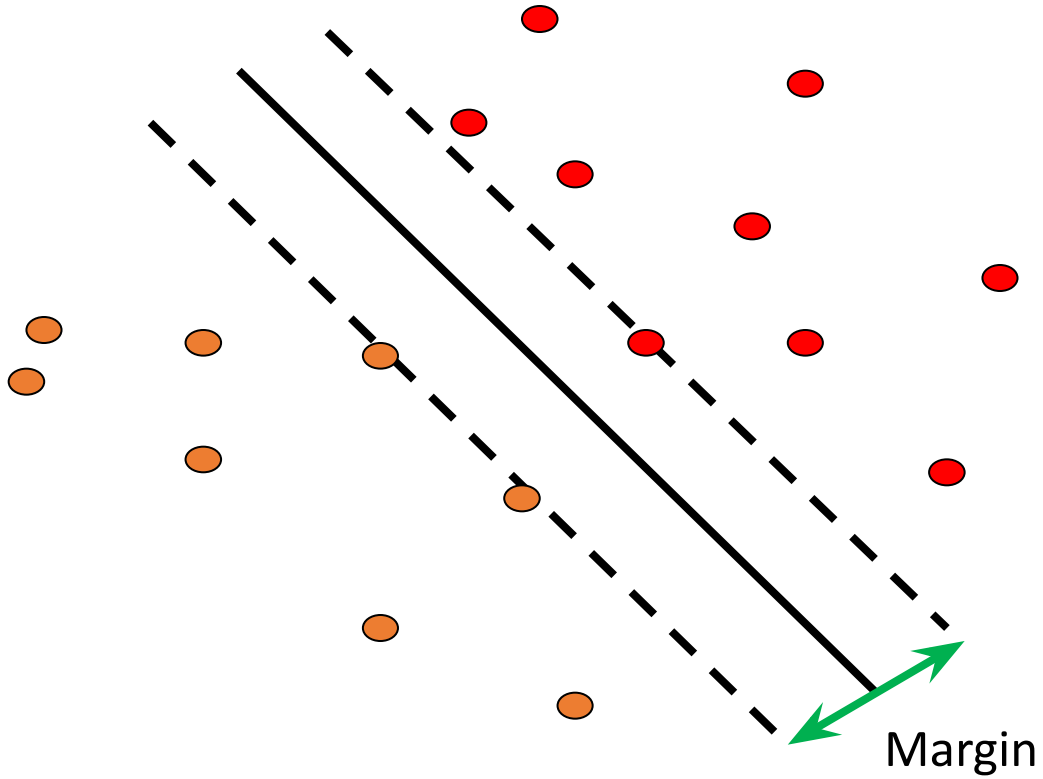
- Find linear function to separate positive and negative examples



$$\mathbf{x}_i \text{ positive:} \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$
$$\mathbf{x}_i \text{ negative:} \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

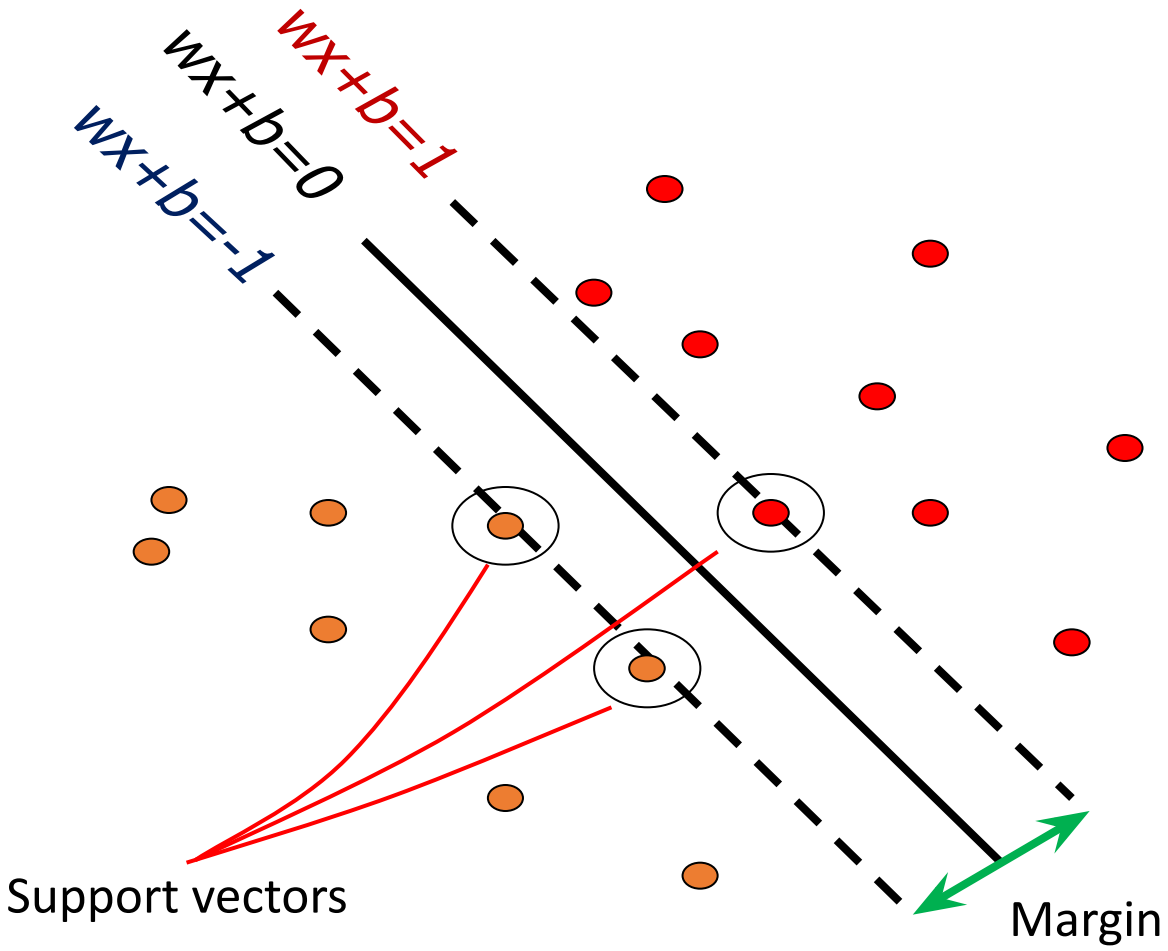
Which line is best?

Support vector machine



- Maximize the *margin* between the positive and negative training examples

Support vector machine

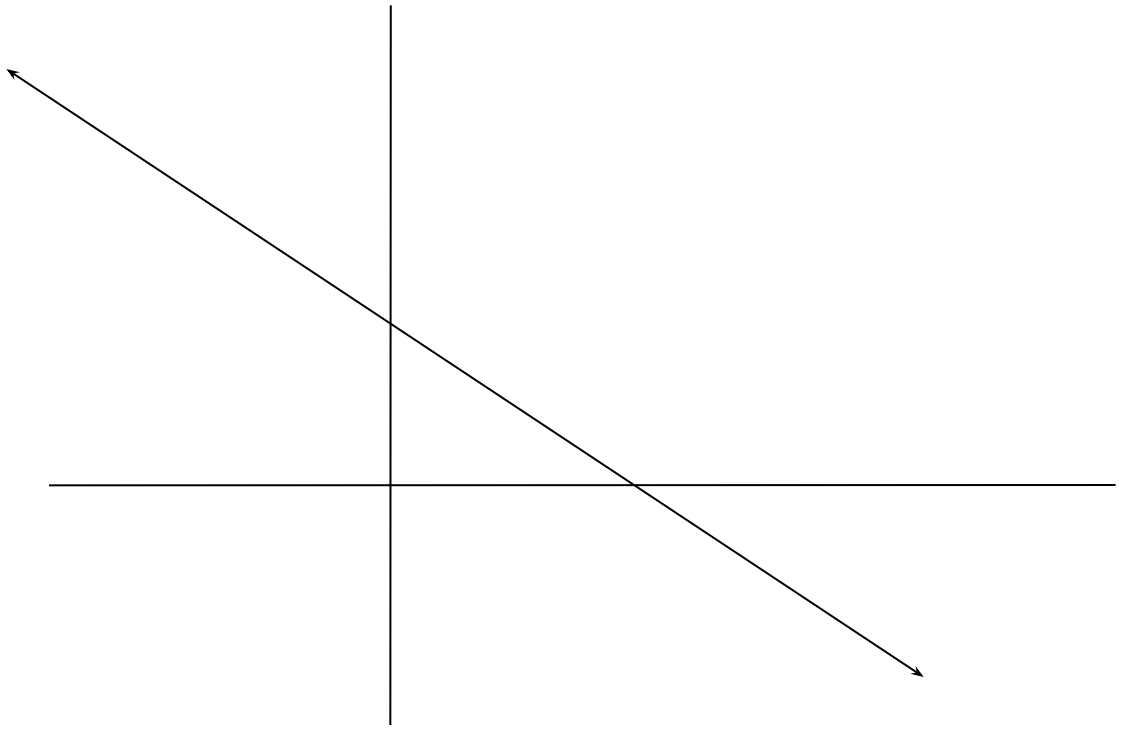


- Want line that maximizes the margin.

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

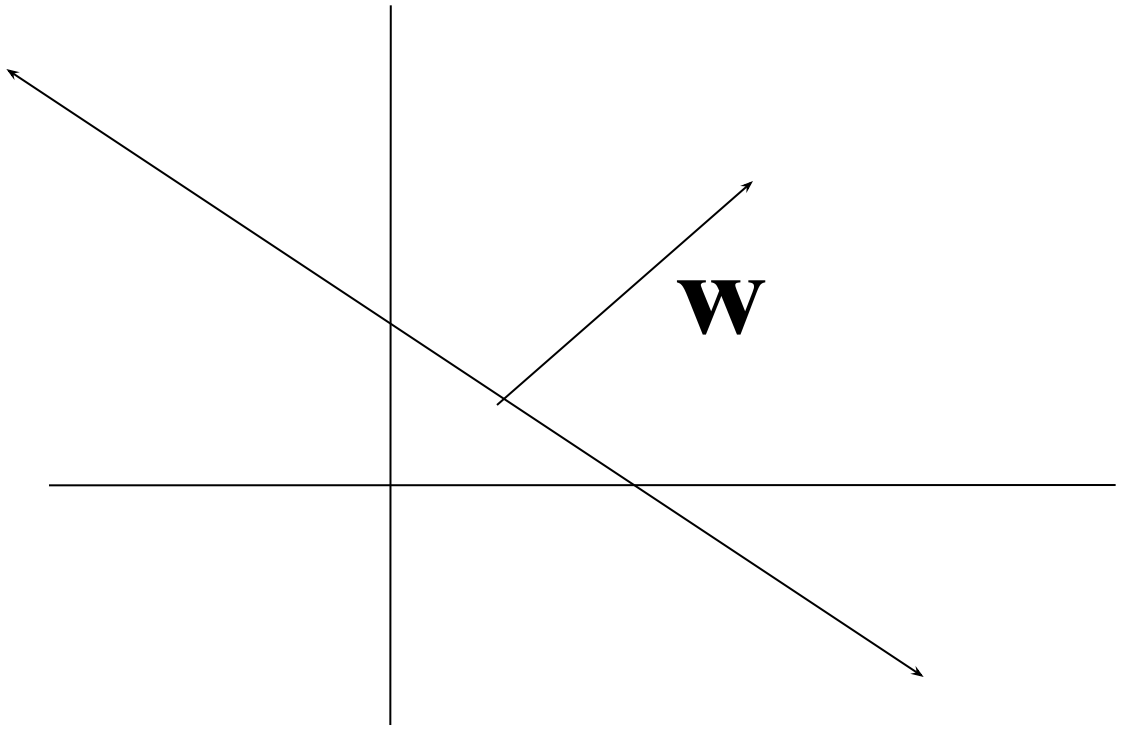
Aside: Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Aside: Distance from line



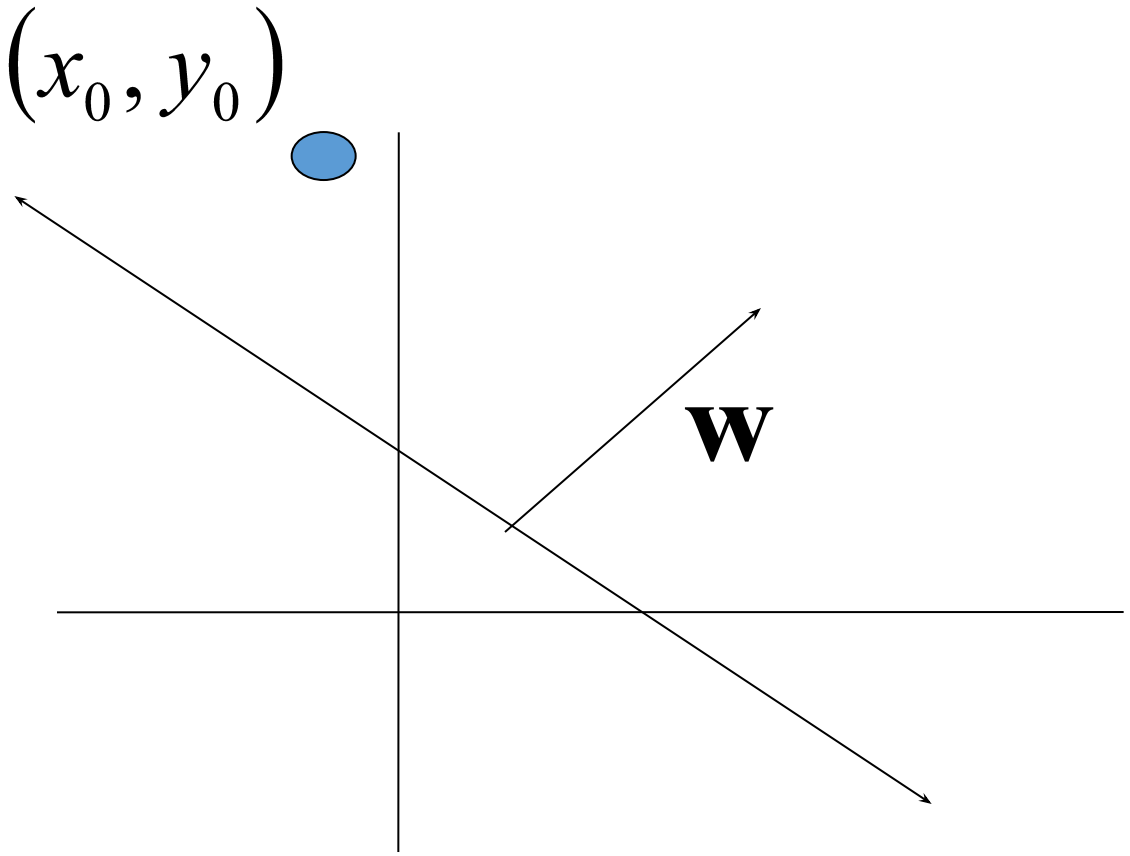
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Aside: Distance from line



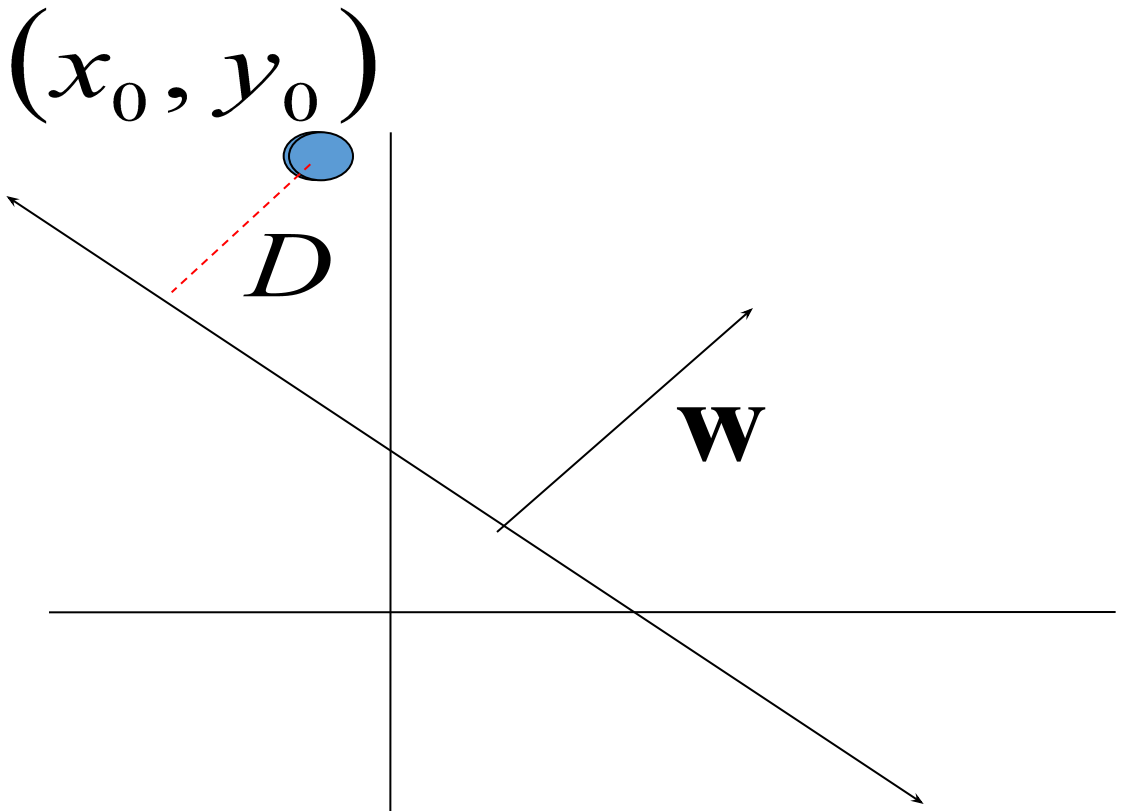
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Aside: Distance from line



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

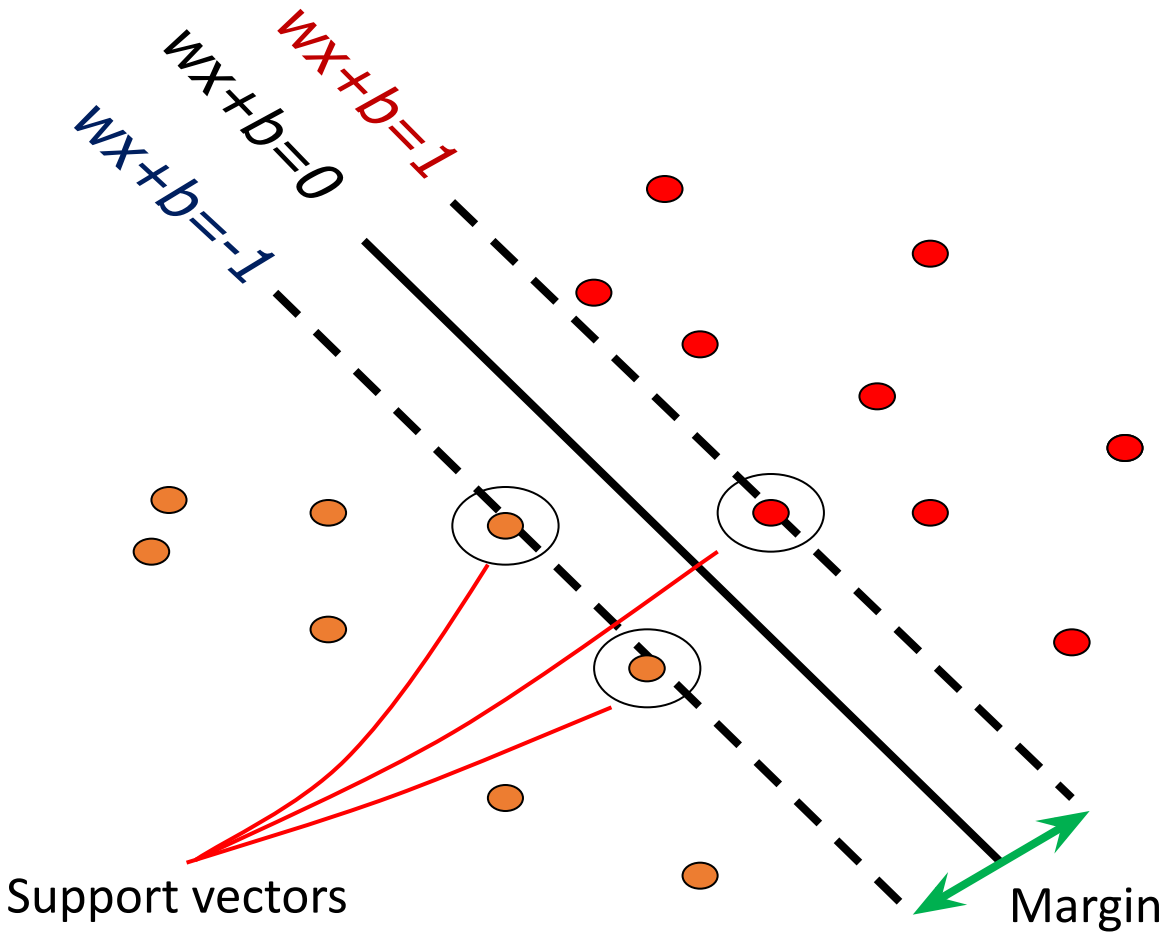
$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

distance from point to line

Support vector machine



- Want line that maximizes the margin.

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

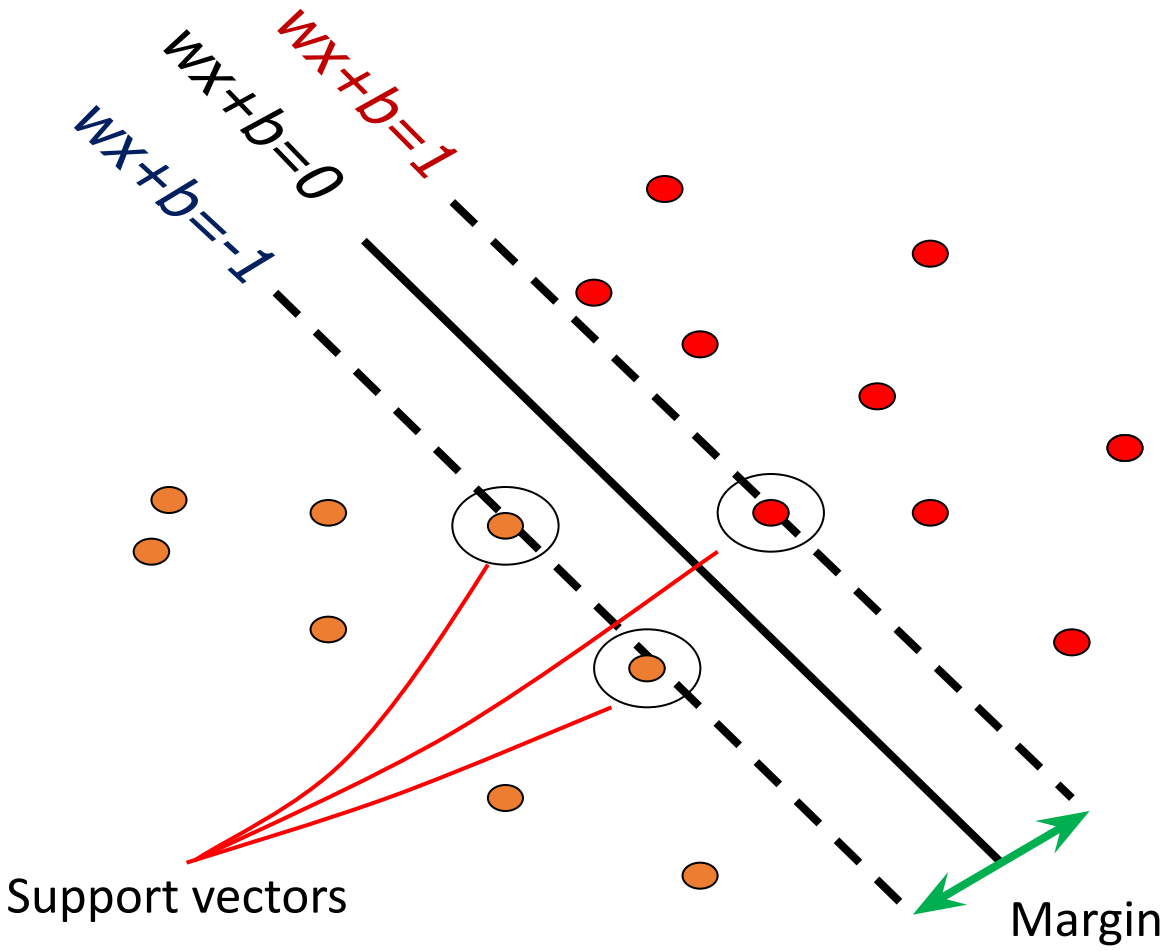
Distance between point and line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors:

$$\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|}$$

$$M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

Support vector machine



- Want line that maximizes the margin.

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

Finding the maximum margin line

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Quadratic optimization problem*:

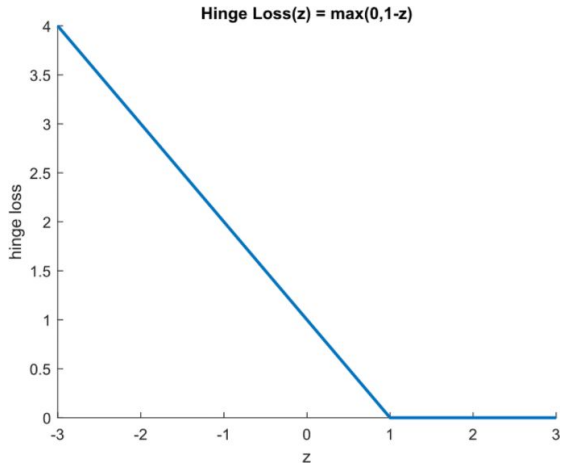
$$\text{Minimize} \quad \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}$$

$$\text{Subject to } y_i(x_i \boldsymbol{w} + b) \geq 1$$

Note sign trick.

One constraint for each training point.

Hinge loss (unconstrained objective)



- Let $\mathrm{h}(z) = \max(0, 1 - z)$
- Then we can define **Hinge loss**:

$$\mathrm{h}(y_i(x_i\boldsymbol{w} + b)) = \max(0, 1 - y_i(x_i w + b))$$

- SVM objective function:

$$\text{Minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\text{Subject to } y_i(x_i\boldsymbol{w} + b) \geq 1$$

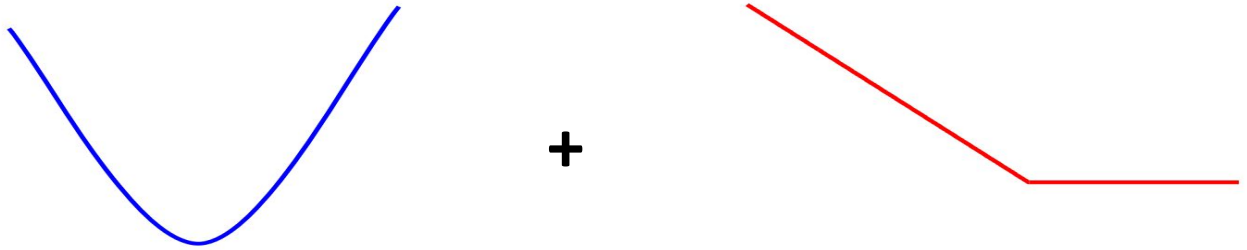
$$\text{Minimize } \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \mathrm{C}\sum_i \mathrm{h}(y_i(x_i\boldsymbol{w} + b))$$

SVM Optimization

- SVM objective function:

$$\text{Minimize } \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \mathrm{C}\sum_i \mathrm{h}(y_i(x_i\boldsymbol{w} + b))$$

$$\mathrm{h}(z) = \max(0, 1 - z)$$



- Convex function → (Stochastic) Gradient Descent

SVM Optimization

- SVM objective function:

$$\text{Minimize } \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \mathrm{C}\sum_i \mathrm{h}(y_i(x_i\boldsymbol{w} + b))$$

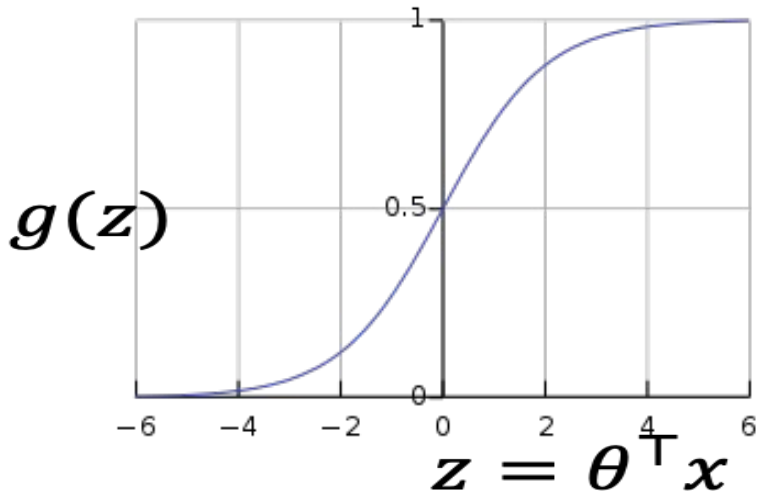
$$\mathrm{h}(z) = \max(0, 1 - z)$$

- $C\left(=\frac{1}{\lambda}\right)$
  - Large C: Lower bias, high variance.
  - Small C: Higher bias, low variance.

Optimization View - SVM

Logistic regression

$$h_\theta(x) = g(\theta^\top x)$$
$$g(z) = \frac{1}{1 + e^{-z}}$$



Suppose predict "y = 1" if $h_\theta(x) \geq 0.5$

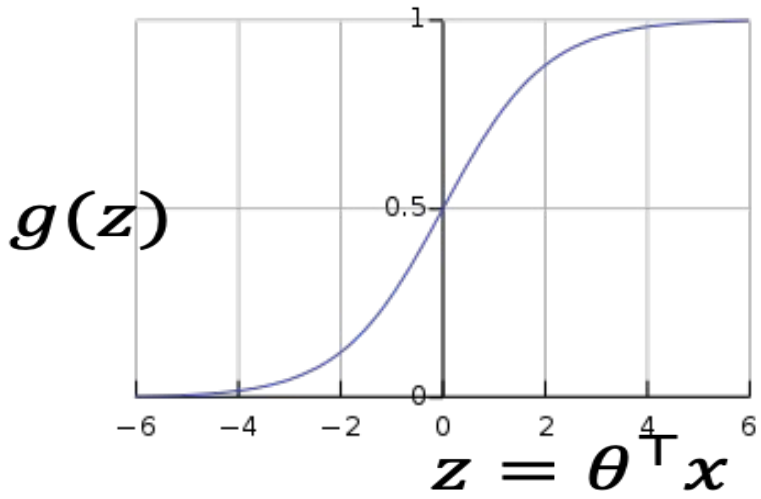
$$z = \theta^\top x \geq 0$$

predict "y = 0" if $h_\theta(x) < 0.5$

$$z = \theta^\top x < 0$$

Alternative view of logistic regression

$$h_\theta(x) = g(\theta^\top x)$$
$$g(z) = \frac{1}{1 + e^{-z}}$$



If "y = 1", we want $h_\theta(x) \approx 1$

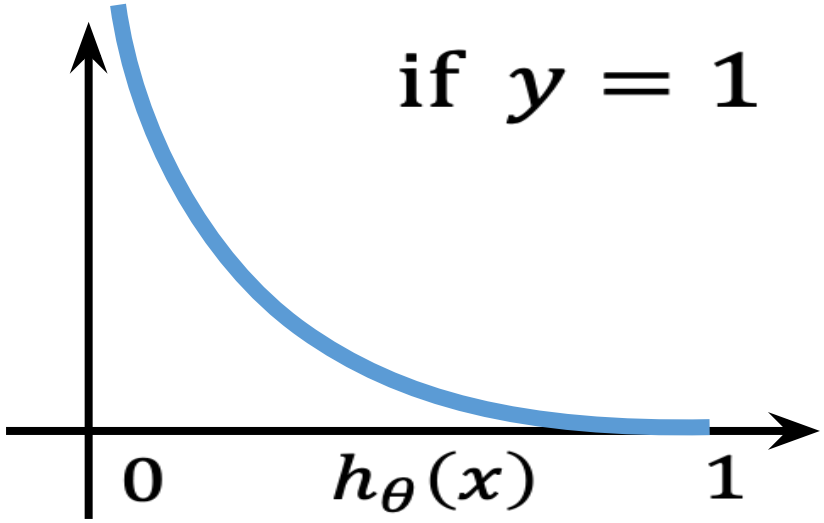
$$z = \theta^\top x \gg 0$$

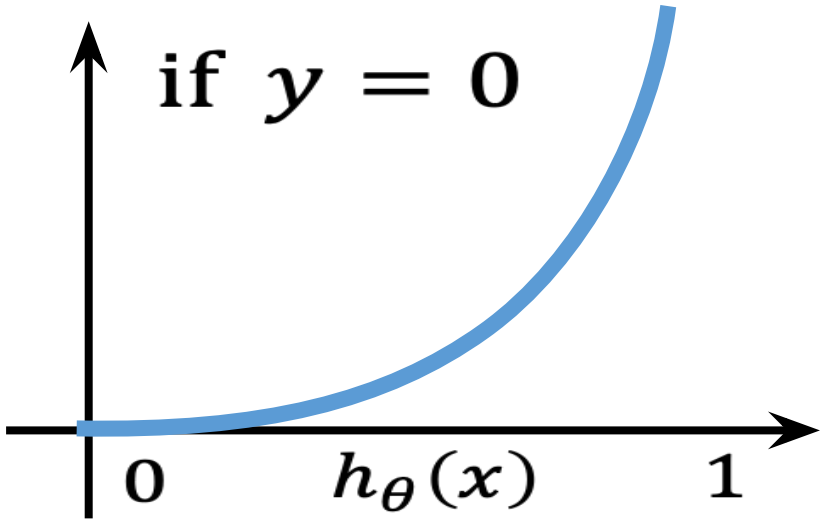
If "y = 0", we want $h_\theta(x) \approx 0$

$$z = \theta^\top x \ll 0$$

Alternative view of logistic regression

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

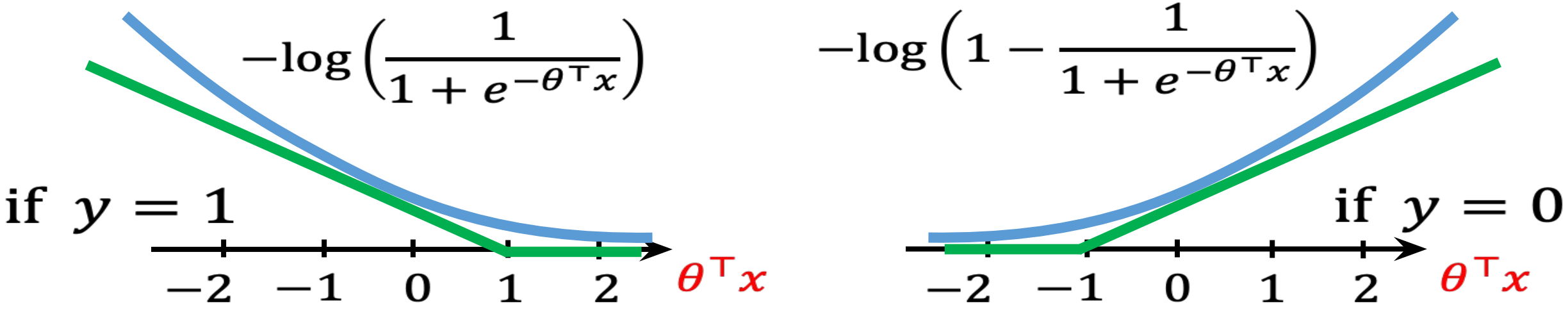




Alternative view of logistic regression

- $\text{Cost}(h_\theta(x), y) = -y\ \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$

$$= y\left(-\log\left(\frac{1}{1+e^{-\theta^\top x}}\right)\right) + (1-y)\left(-\log\left(1-\frac{1}{1+e^{-\theta^\top x}}\right)\right)$$

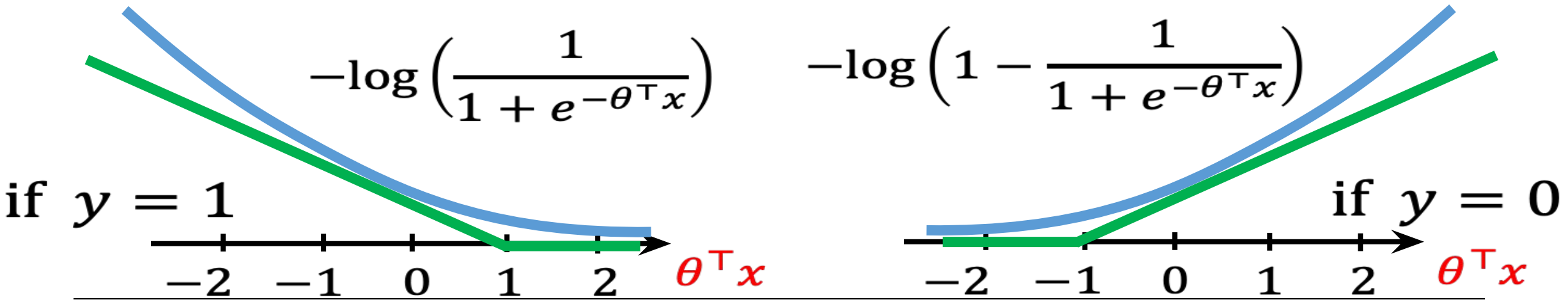


- Logistic regression (**logistic loss**)

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log\left(h_\theta(x^{(i)})\right)\right) + (1-y^{(i)})\left(-\log\left(1-h_\theta(x^{(i)})\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Support vector machine (**hinge loss**)

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1(\theta^\top x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^\top x^{(i)})\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$



Optimization objective for SVM

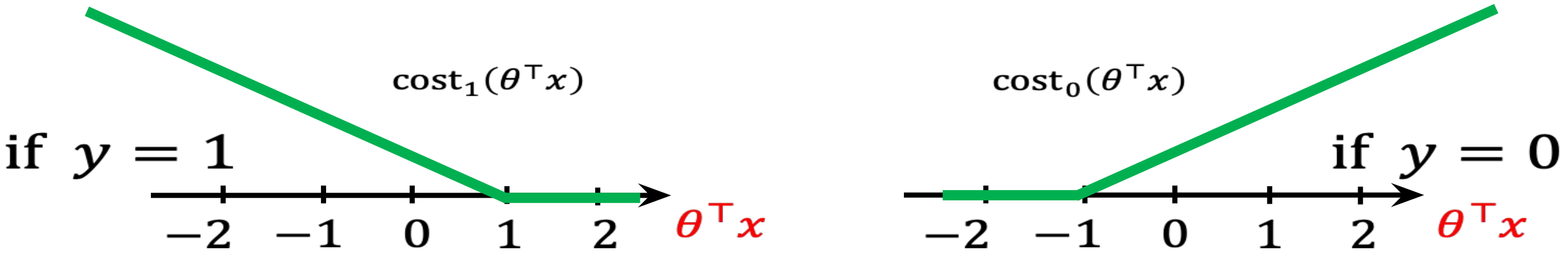
- $$\min_{\theta} \left[\sum_{i=1}^{m} y^{(i)} \ \text{cost}_1(\theta^\top x^{(i)}) + (1 - y^{(i)}) \ \text{cost}_0(\theta^\top x^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

Multiply $C = \frac{1}{\lambda}$

$$\min_{\theta} C \left[\sum_{i=1}^{m} y^{(i)} \ \text{cost}_1(\theta^\top x^{(i)}) + (1 - y^{(i)}) \ \text{cost}_0(\theta^\top x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

Support vector machine

$$\min_{\theta} C\left[\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1(\theta^\top x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^\top x^{(i)})\right] + \sum_{j=1}^{n}\theta_j^2$$



If "y = 1", we want $\theta^\top x \geq 1$ (not just ≥ 0)

If "y = 0", we want $\theta^\top x \leq -1$ (not just < 0)

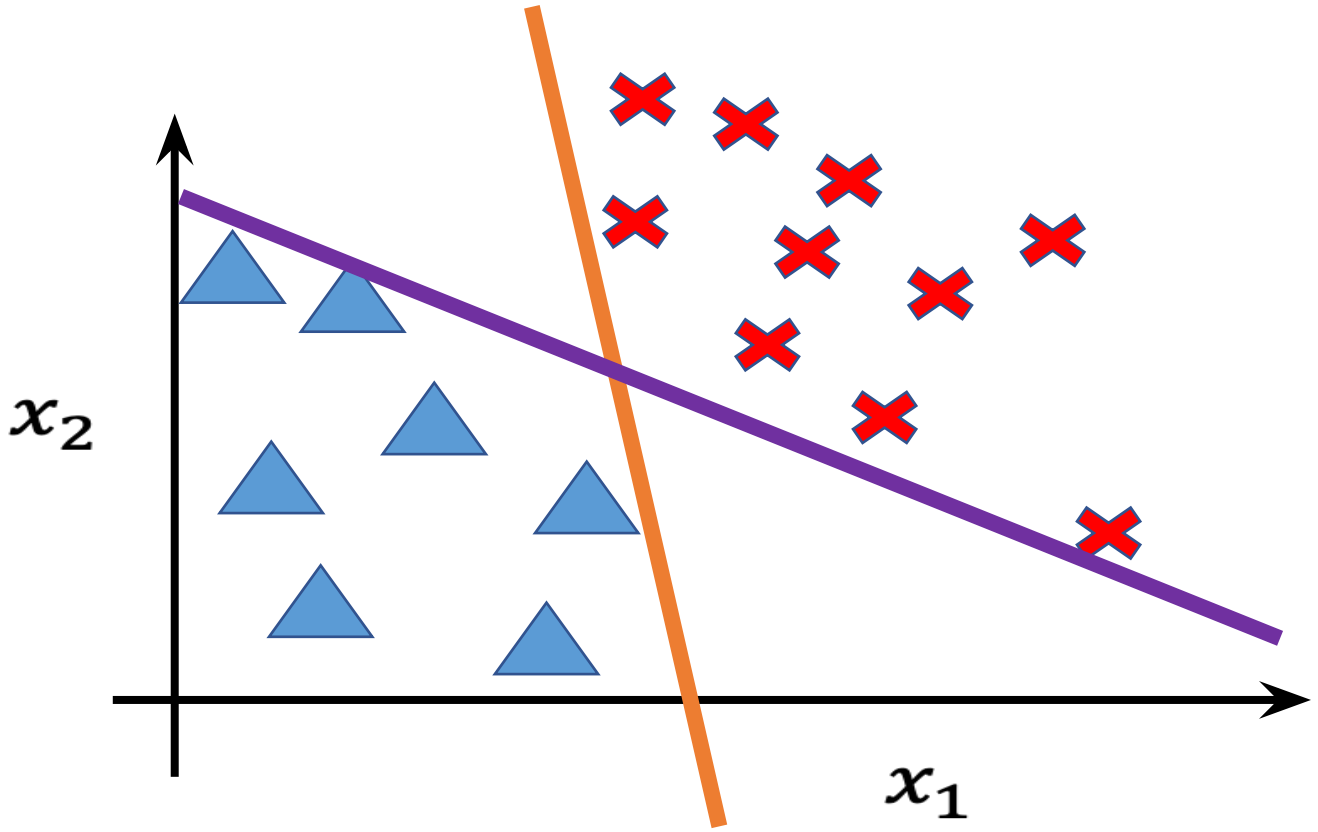
Hypothesis of SVM

- $$\min_{\theta} C\left[\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1(\theta^\top x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^\top x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

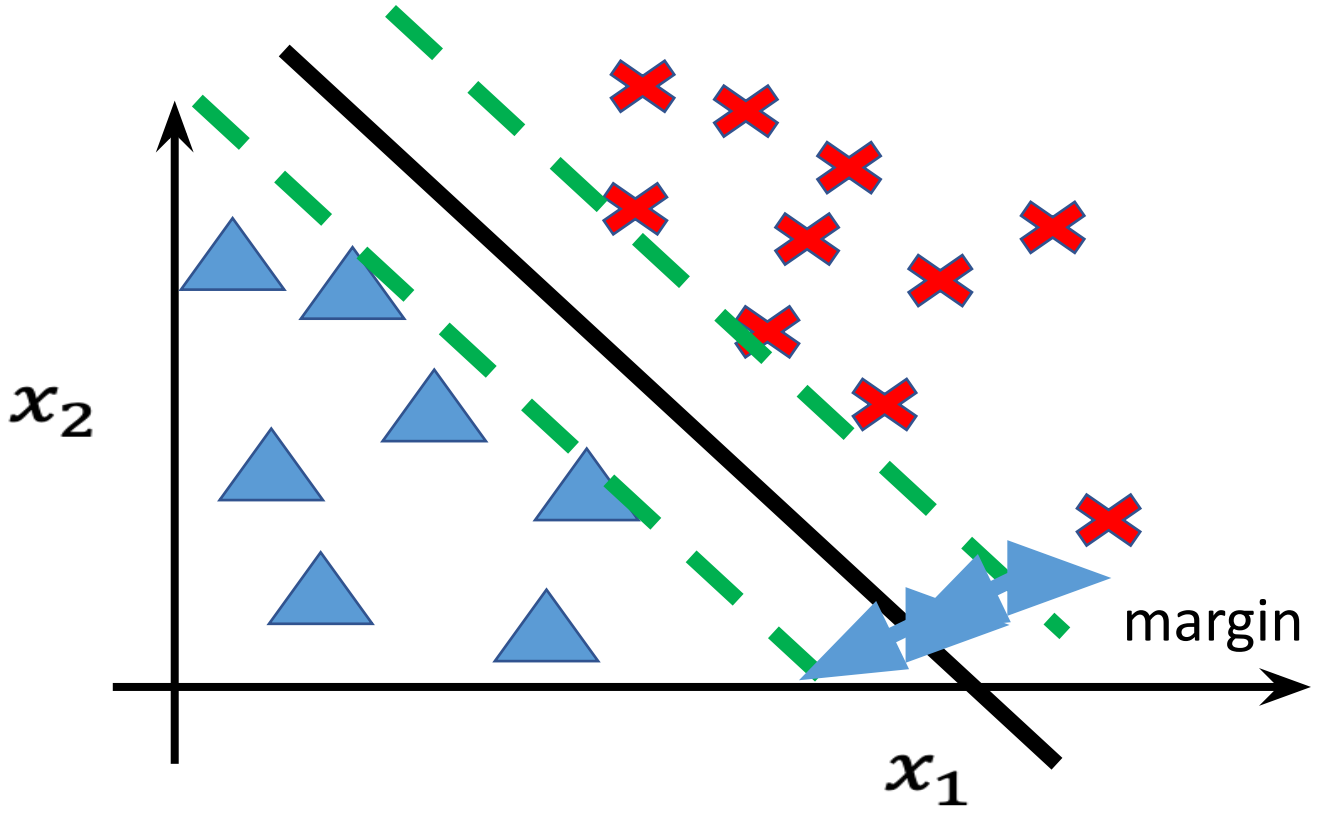
- Hypothesis

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^\top x \geq 0 \\ 0 & \text{if } \theta^\top x < 0 \end{cases}$$

SVM decision boundary: Linearly separable case



SVM decision boundary: Linearly separable case



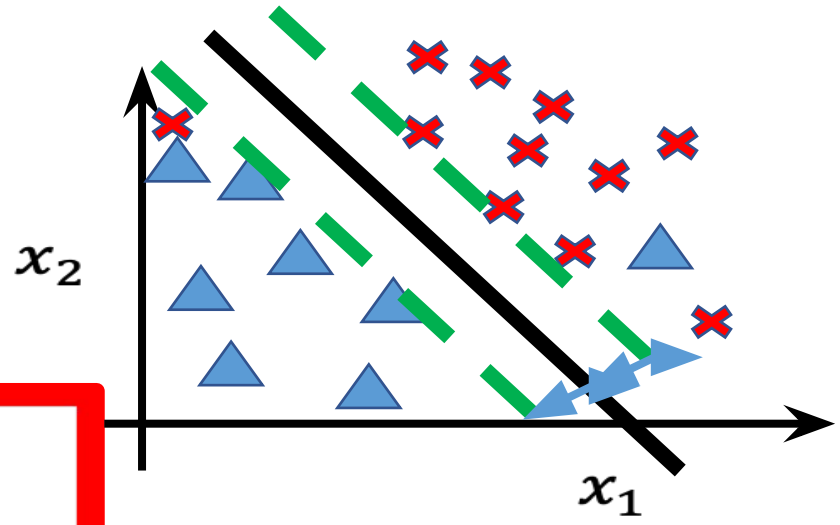
Data not linearly separable?

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n} \theta_j^2 + C(\#\text{misclassification})$$

$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$



NP-hard ☹

Convex relaxation

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 + C(\#\text{misclassification})$$

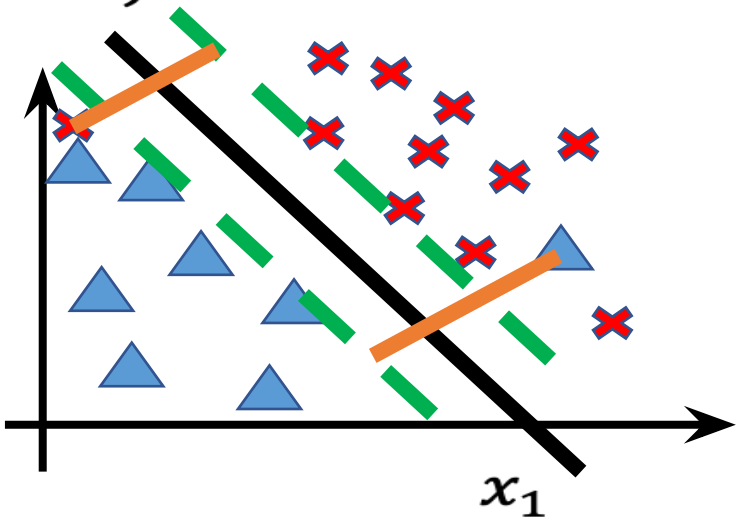
$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

NP-hard ☹

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 + C\sum_i \xi^{(i)}$$

$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 - \xi^{(i)} \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 + \xi^{(i)} \quad \text{if} \quad y^{(i)} = 0$$
$$\xi^{(i)} \geq 0 \quad \forall\, i$$

$\xi^{(i)}$: slack variables



Hard-margin SVM formulation

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

Soft-margin SVM formulation

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n} \theta_j^2 + C\sum_i \xi^{(i)}$$

$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 - \xi^{(i)} \quad \text{if} \quad y^{(i)} = 1$$
$$\theta^\top x^{(i)} \leq -1 + \xi^{(i)} \quad \text{if} \quad y^{(i)} = 0$$
$$\xi^{(i)} \geq 0 \quad \forall\, i$$

Finding the maximum margin line

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Quadratic optimization problem*:

$$\text{Minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

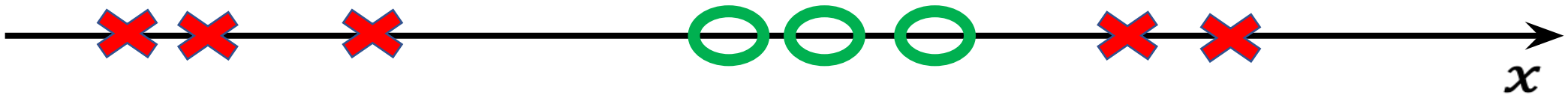
$$\text{Subject to } y_i(x_i\boldsymbol{w} + b) \geq 1$$

$$\text{Minimize } \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \mathrm{C}\sum_i \mathrm{h}(y_i(x_i\boldsymbol{w} + b))$$

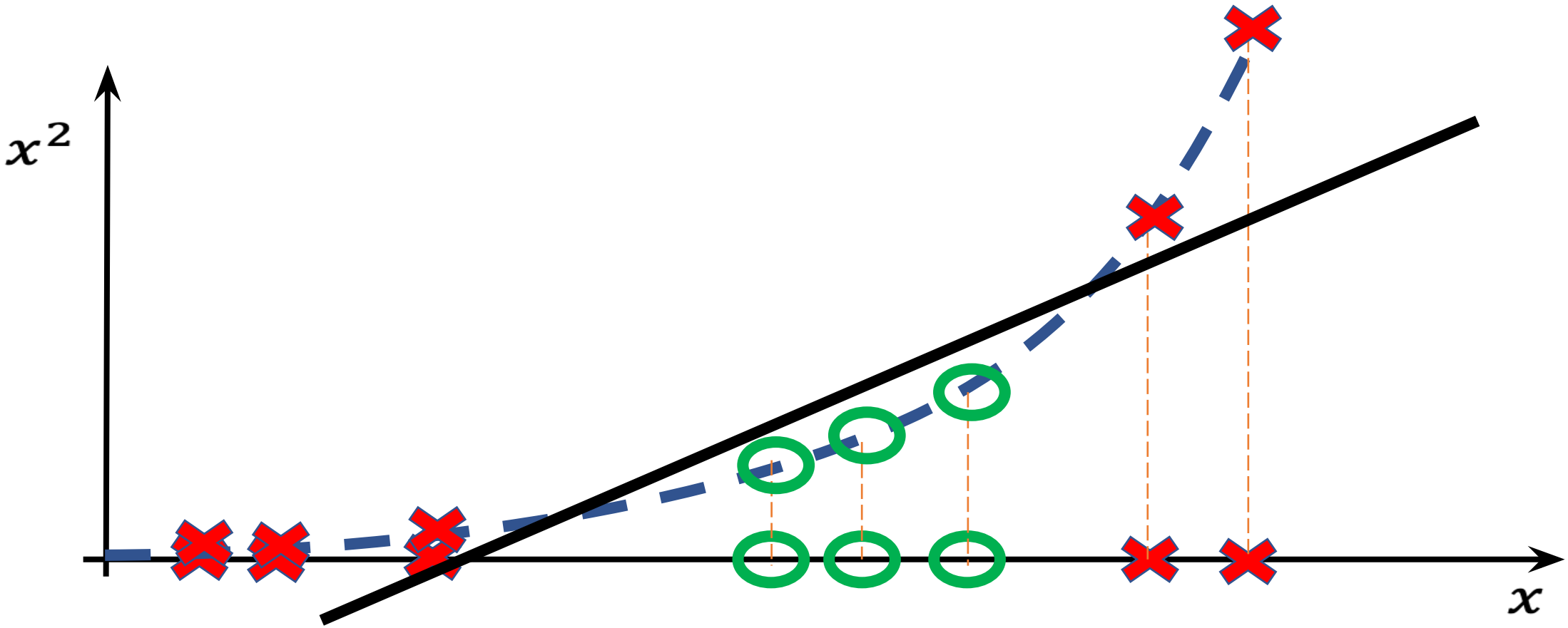
Non-linear classification

- How do we separate the two classes using a hyperplane?

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^\top x \geq 0 \\ 0 & \text{if } \theta^\top x < 0 \end{cases}$$

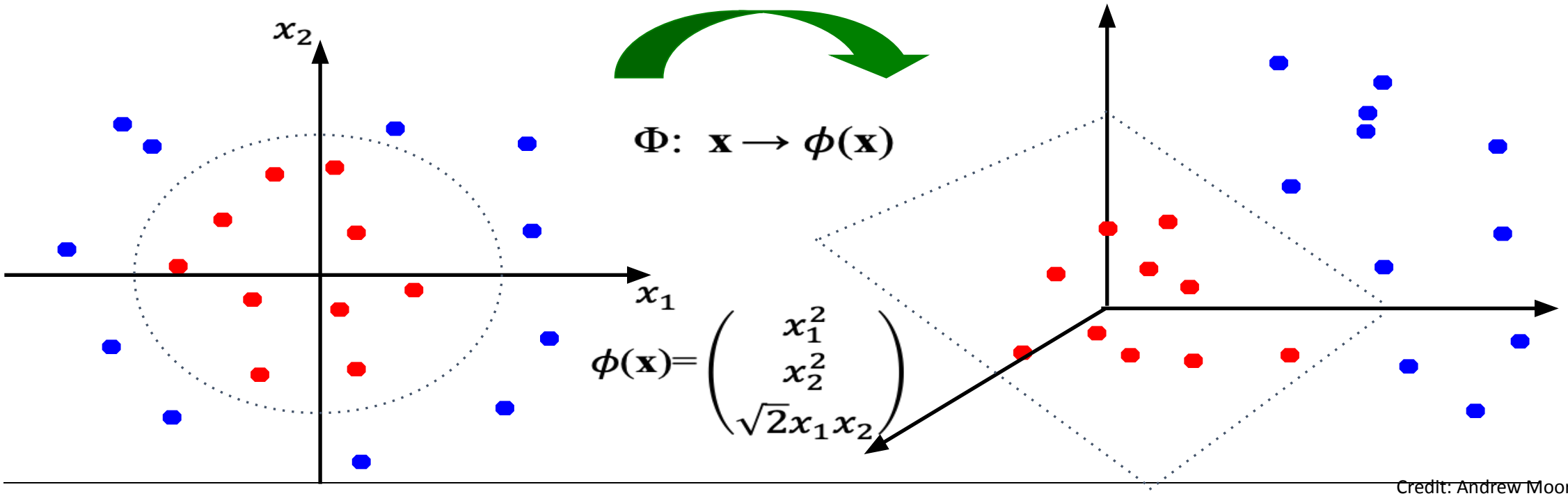


Non-linear classification



Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Credit: Andrew Moore

Kernel

- $K(\cdot,\cdot)$ a legal definition of inner product:

$$\exists \phi: X \rightarrow R^N$$
$$\text{s.t. } K(x,z) = \phi(x)^\top \phi(z)$$

Why Kernels matter?

- Many algorithms interact with data only via **dot-products**
- Replace $x^\top z$ with $K(x, z) = \phi(x)^\top \phi(z)$
- Act *implicitly* as if data was in the higher-dimensional ϕ-space

Example

$K(x, z) = (x^\top z)^2$ corresponds to

$(x_1, x_2) \rightarrow \phi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$

$$\phi(x)^\top \phi(z) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)\left(z_1^2, z_2^2, \sqrt{2}z_1z_2\right)^\top$$

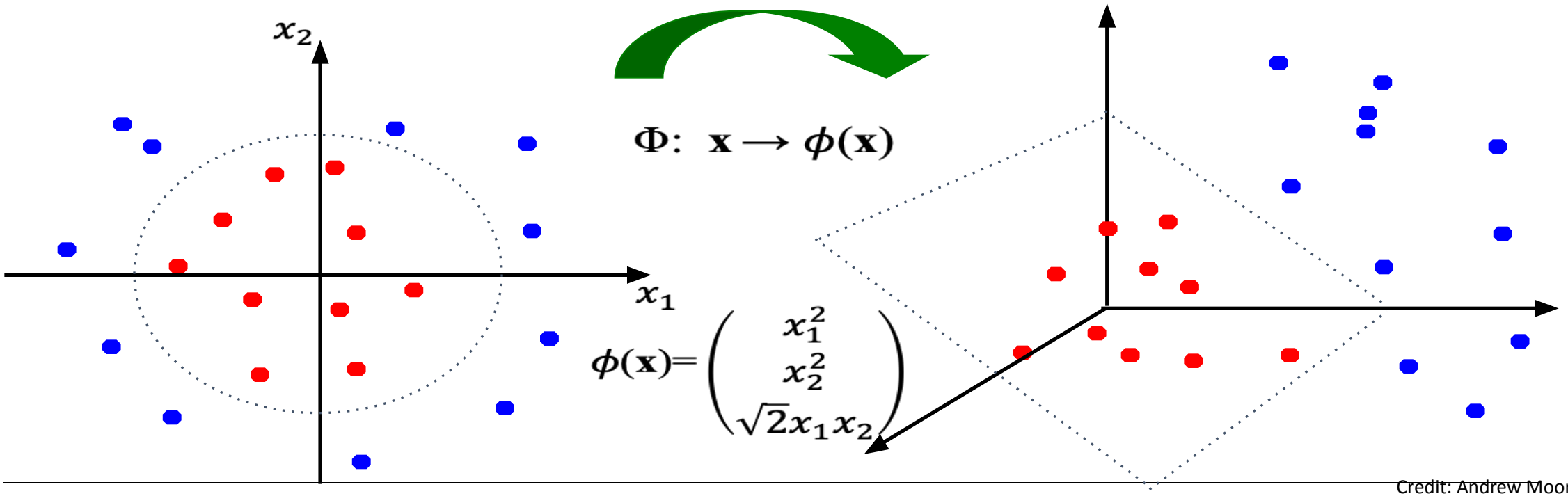
$$= x_1^2z_1^2 + x_2^2z_2^2 + 2\mathrm{x}_1\mathrm{x}_2\mathrm{z}_1\mathrm{z}_2 = (\mathrm{x}_1\mathrm{z}_1 + \mathrm{x}_2\mathrm{z}_2)^2$$

$$= (x^\top z)^2 = K(x, z)$$

Example

- $K(x, z) = (x^T z)^2$ corresponds to

$$(x_1, x_2) \rightarrow \phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$



Credit: Andrew Moore

Constructing new kernels

- Positive scaling

$$K(x, z) = cK_1(x, z), c > 0$$

- Exponentiation

$$K(x, z) = \exp(K_1(x, z))$$

- Addition

$$K(x, z) = K_1(x, z) + K_2(x, z)$$

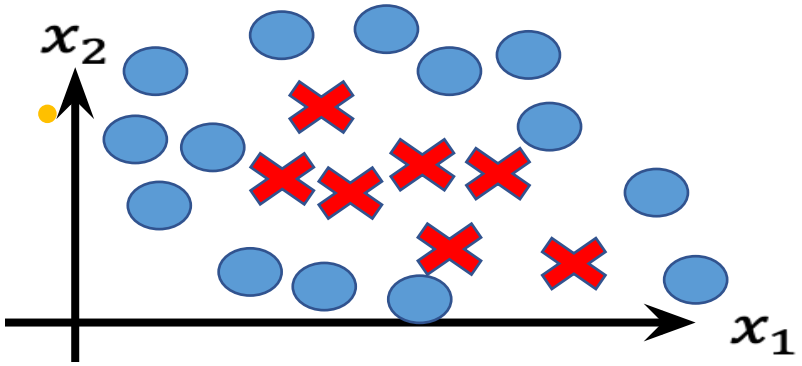
- Multiplication with function

$$K(x, z) = f(x)K_1(x, z)f(z)$$

- Multiplication

$$K(x, z) = K_1(x, z)K_2(x, z)$$

Non-linear decision boundary



Predict $y = 1$ if

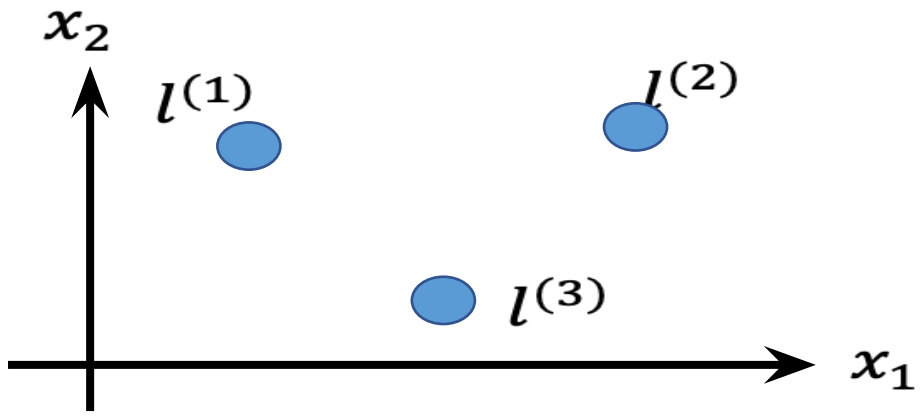
$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$$

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, \cdots$$

Is there a different/better choice of the features $f_1, f_2, f_3, \cdots$?

Kernel



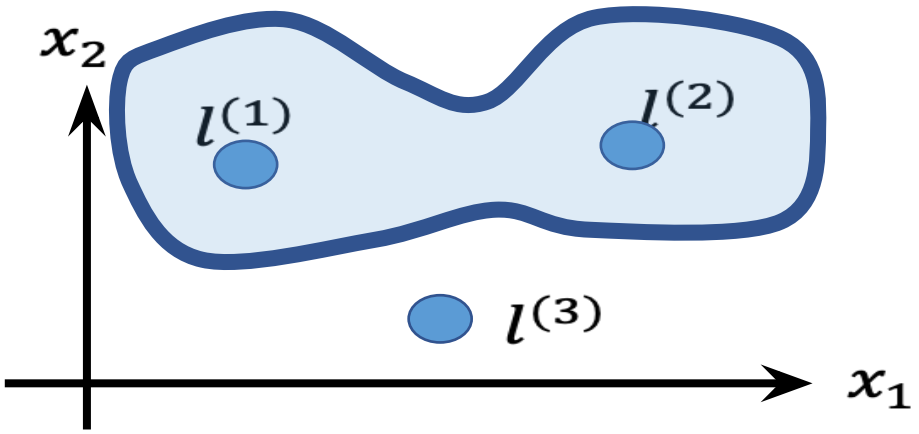
Give x, compute new features depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$f_3 = \text{similarity}(x, l^{(3)})$$

Gaussian kernel

$$\text{similarity}(x, l^{(i)}) = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

Kernel



Predict $y = 1$ if

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

Ex: $\theta_0 = -0.5, \theta_1 = 1, \theta_2 = 1, \theta_3 = 0$

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$f_3 = \text{similarity}(x, l^{(3)})$$

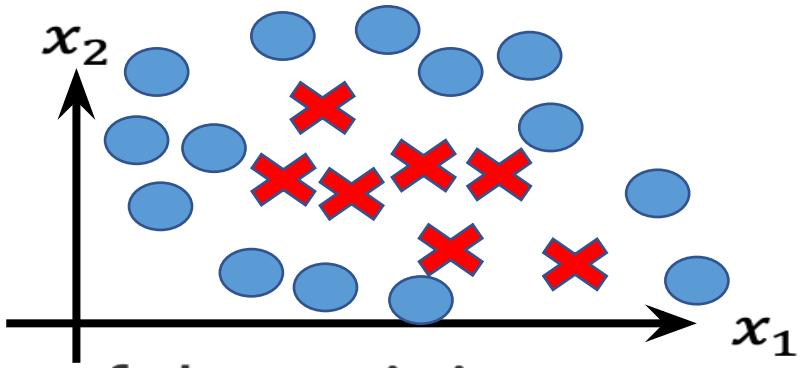
Choosing the landmarks

- Given x

$$f_i = \text{similarity}(x, l^{(i)}) = \exp(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2})$$

- Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

- Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \cdots$?



$l^{(1)}$
$l^{(2)}$
$l^{(3)}$
$\vdots$
$l^{(m)}$

- Choose landmarks at the point of the training examples

SVM with kernels

- Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
- Choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, l^{(3)} = x^{(3)}, \cdots, l^{(m)} = x^{(m)}$
- Given example x:
  - $f_1 = \text{similarity}(x, l^{(1)})$
  - $f_2 = \text{similarity}(x, l^{(2)})$
  - $\cdots$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$$

- For training example $(x^{(i)}, y^{(i)})$:
  - $x^{(i)} \rightarrow f^{(i)}$

SVM with kernels

- Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$
  - Predict $y = 1$ if $\theta^\top f \geq 0$
- **Training (original)**

$$\min_\theta C\left[\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1\left(\theta^\top x^{(i)}\right) + \left(1 - y^{(i)}\right)\text{cost}_0\left(\theta^\top x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

- **Training (with kernel)**

$$\min_\theta C\left[\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1\left(\theta^\top f^{(i)}\right) + \left(1 - y^{(i)}\right)\text{cost}_0\left(\theta^\top f^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{m}\theta_j^2$$

Feature scaling in SVM

- Feature scaling is important when using SVM especially Gaussian Kernels
  - if the ranges vary a lot then the similarity feature would be dominated by features with large values.

Multi-class problems

- Instead of just two classes, we now have C classes
  - E.g. predict which movie genre a viewer likes best
  - Possible answers: action, drama, indie, thriller, etc.
- Two approaches:
  - One-vs-rest
  - One-vs-one

Multi-class problems

- One-vs-rest
- Train K classifiers
  - In each, pos = data from class *i*, neg = data from classes other than *i*
  - The class with the most confident prediction wins
  - Example:
    - You have 4 classes, train 4 classifiers
    - 1 vs rest: score 3.5
    - 2 vs rest: score 6.2
    - 3 vs rest: score 1.4
    - 4 vs rest: score 5.5
    - Final prediction: class 2

Multi-class problems

- One-vs-one (a.k.a. all-vs-all)
  - Train C(C-1)/2 binary classifiers (all pairs of classes)
  - They all vote for the label
  - Example:
    - You have 4 classes, then train 6 classifiers
    - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
    - Votes: 1, 1, 4, 2, 4, 4
    - Final prediction is class 4

Summary

- Pros
  - Kernel-based framework is very powerful, flexible
  - Often a sparse set of support vectors – compact at test time
  - Work very well in practice, even with very small training sample sizes
  - Solution can be formulated as a quadratic program

- Cons
  - Can be tricky to select best kernel function for a problem
  - Computation, memory
    - At training time, must compute kernel values for all example pairs
    - Learning can take a very long time for large-scale problems

Credit: Lana Lazebnik

Questions + Comments?