

# Machine Learning (CS 567) Lecture 3

**Fall 2008**

**Time: T-Th 5:00pm - 6:20pm**

**Location: GFS 118**

**Instructor: Sofus A. Macskassy ([macskass@usc.edu](mailto:macskass@usc.edu))**

**Office: SAL 216**

**Office hours: by appointment**

**Teaching assistant: Cheol Han ([cheolhan@usc.edu](mailto:cheolhan@usc.edu))**

**Office: TBA**

**Office hours: TBA**

**Class web page:**

<http://www-scf.usc.edu/~csci567/index.html>

# Administrative

- Email me to get on the mailing list
  - To: [macskass@usc.edu](mailto:macskass@usc.edu)
  - Subject: "csci567 student"
- Please stop me if I talk too fast
- Please stop me if you have any questions
- *Any* feedback is better than *no* feedback
  - Sooner is better
  - End of the semester is too late to make it easier on you

# Administrative

- Weekly quizzes will start next lecture
  - Last 10 minutes of class
  - Example quizz at end of today's class

# Questions from Last Class

- Single-person Projects
- Need for primary book
- Supervised Learning Example  $P(y|x) = 0.1$

# Example: Making the tumor decision

- Suppose our tumor recurrence detector predicts:

$$P(y = \text{"recurrence"} \mid \mathbf{x}) = 0.1$$

What is the optimal classification decision  $\hat{y}$ ?

- Expected loss of  $\hat{y} = \text{"recurrence"}$  is:

$$0 * 0.1 + 1 * 0.9 = 0.9$$

- Expected loss of  $\hat{y} = \text{"non recurrence"}$  is:

$$100 * 0.1 + 0 * 0.9 = 10$$

- Therefore, the optimal prediction is "recurrence"

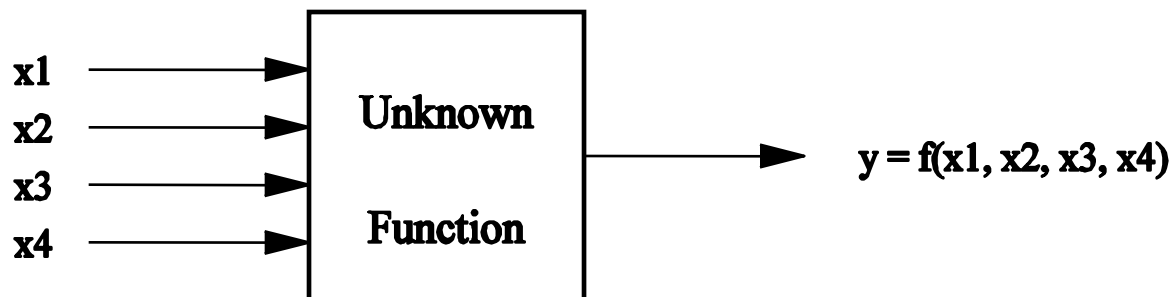
Loss function  $L(\hat{y}, y)$ :

predicted label $\hat{y}$	true label $y$	
	recurrence	non recurrence
recurrence	0	1
non recurrence	100	0
$P(y \mid \mathbf{x})$	0.1	0.9

# Lecture 3 Outline

- A Framework for hypothesis spaces
- A Framework for algorithms
- Linear Threshold Units
- Perceptron

# Fundamental Problem of Machine Learning: It is ill-posed



Example	$x_1$	$x_2$	$x_3$	$x_4$	$y$
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

# Learning Appears Impossible

- There are  $2^{16} = 65536$  possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 7 examples, we still have  $2^9$  possibilities.

$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

# Illustration: Simple Conjunctive Rules

- There are only 16 simple conjunctions (no negation)
- However, no simple rule explains the data. The same is true for simple clauses

Example	$x_1$	$x_2$	$x_3$	$x_4$	$y$
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Rule	Counterexample
$\text{true} \Leftrightarrow y$	1
$x_1 \Leftrightarrow y$	3
$x_2 \Leftrightarrow y$	2
$x_3 \Leftrightarrow y$	1
$x_4 \Leftrightarrow y$	7
$x_1 \wedge x_2 \Leftrightarrow y$	3
$x_1 \wedge x_3 \Leftrightarrow y$	3
$x_1 \wedge x_4 \Leftrightarrow y$	3
$x_2 \wedge x_3 \Leftrightarrow y$	3
$x_2 \wedge x_4 \Leftrightarrow y$	3
$x_3 \wedge x_4 \Leftrightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Leftrightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Leftrightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Leftrightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Leftrightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Leftrightarrow y$	3

# A larger hypothesis space: *m*-of-*n* rules

- At least  $m$  of the  $n$  variables must be true
- There are 32 possible rules
- Only one rule is consistent!

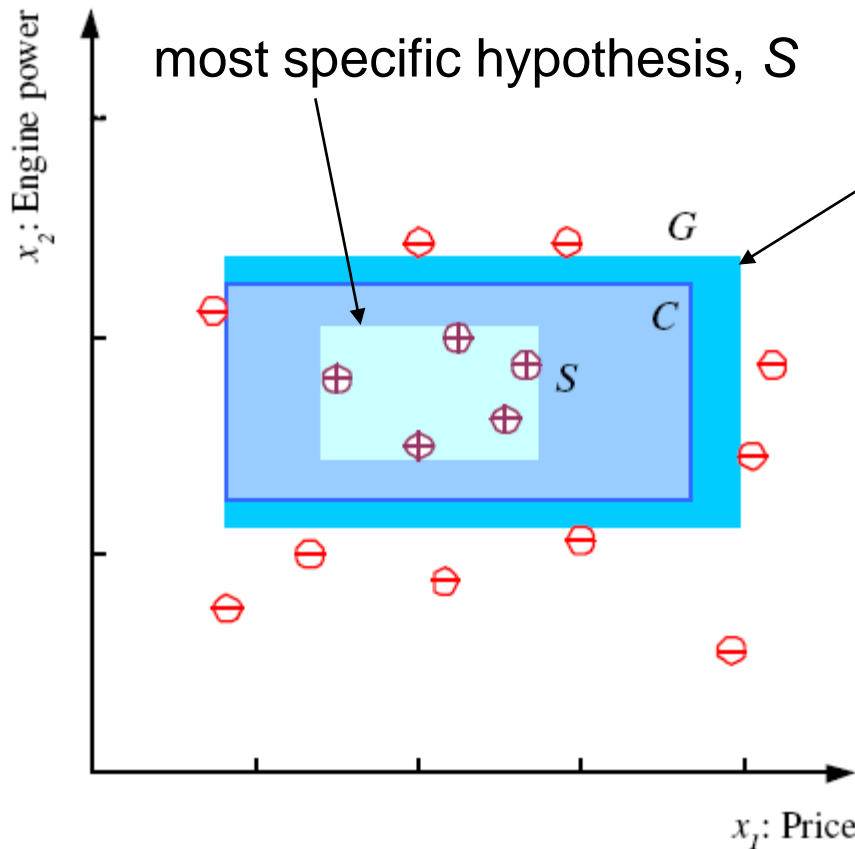
Example	$x_1$	$x_2$	$x_3$	$x_4$	$y$
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

variables	Counterexample			
	1-of	2-of	3-of	4-of
$\{x_1\}$	3	—	—	—
$\{x_2\}$	2	—	—	—
$\{x_3\}$	1	—	—	—
$\{x_4\}$	7	—	—	—
$\{x_1, x_2\}$	3	3	—	—
$\{x_1, x_3\}$	4	3	—	—
$\{x_1, x_4\}$	6	3	—	—
$\{x_2, x_3\}$	2	3	—	—
$\{x_2, x_4\}$	2	3	—	—
$\{x_3, x_4\}$	4	4	—	—
$\{x_1, x_2, x_3\}$	1	3	3	—
$\{x_1, x_2, x_4\}$	2	3	3	—
$\{x_1, x_3, x_4\}$	1	***	3	—
$\{x_2, x_3, x_4\}$	1	5	3	—
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

# Two Strategies for Machine Learning

- Develop Languages for Expressing Prior Knowledge
  - Rule grammars, stochastic models, Bayesian networks
  - (Corresponds to the Prior Knowledge view)
- Develop Flexible Hypothesis Spaces
  - Nested collections of hypotheses: decision trees, neural networks, cases, SVMs
  - (Corresponds to the Guessing view)
- In either case we must develop algorithms for finding an hypothesis that fits the data

# S, G, and the Version Space



most general hypothesis, G

$h \in H$ , between S and G is consistent

and make up the version space

(Mitchell, 1997)

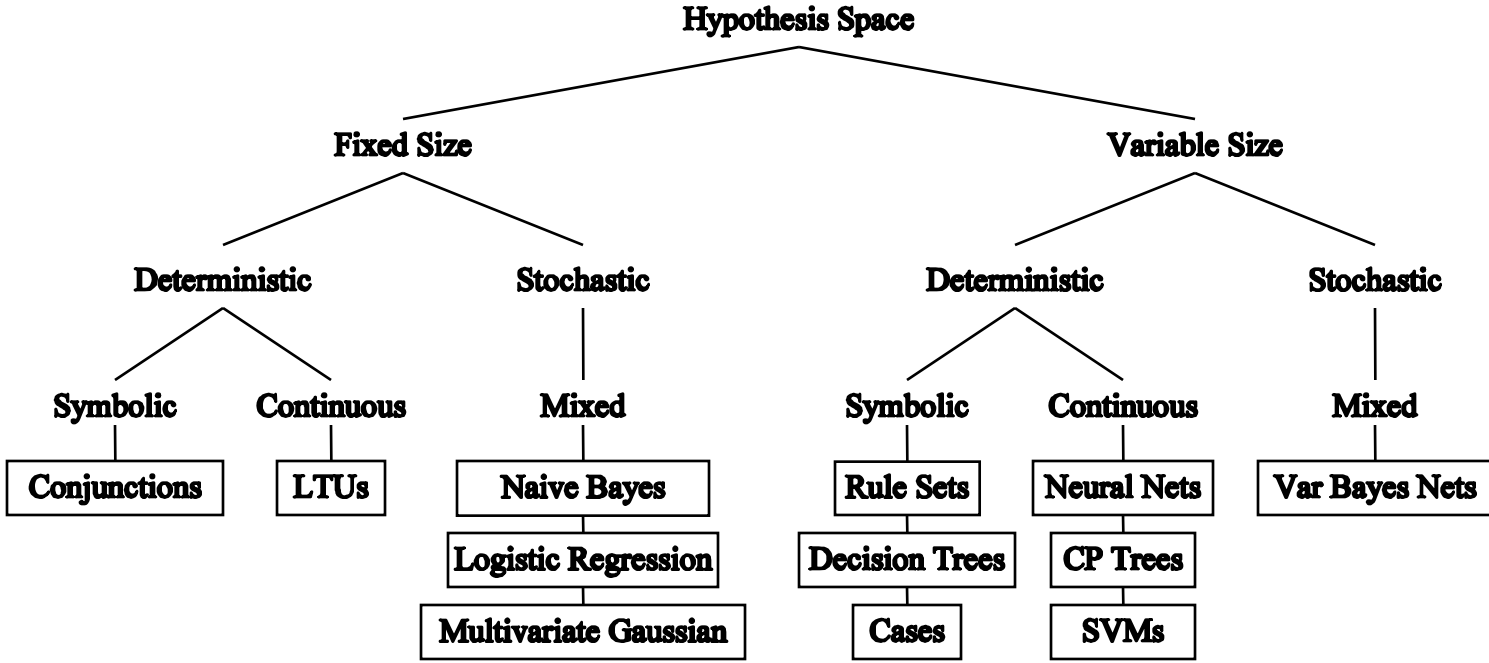
# Key Issues in Machine Learning

- What are good hypothesis spaces?
  - which spaces have been useful in practical applications?
- What algorithms can work with these spaces?
  - Are there general design principles for learning algorithms?
- How can we optimize accuracy on future data points?
  - This is related to the problem of “overfitting”
- How can we have confidence in the results? (the statistical question)
  - How much training data is required to find an accurate hypotheses?
- Are some learning problems computationally intractable? (the computational question)
- How can we formulate application problems as machine learning problems? (the engineering question)

# A framework for hypothesis spaces

- Size: Does the hypothesis space have a fixed size or a variable size?
  - fixed-sized spaces are easier to understand, but variable-sized spaces are generally more useful. Variable-sized spaces introduce the problem of overfitting
- Stochasticity. Is the hypothesis a classifier, a conditional distribution, or a joint distribution?
  - This affects how we evaluate hypotheses. For a deterministic hypothesis, a training example is either *consistent* (correctly predicted) or *inconsistent* (incorrectly predicted). For a stochastic hypothesis, a training example is *more likely* or *less likely*.
- Parameterization. Is each hypothesis described by a set of symbolic (discrete) choices or is it described by a set of continuous parameters? If both are required, we say the space has a mixed parameterization.
  - Discrete parameters must be found by combinatorial search methods; continuous parameters can be found by numerical search methods

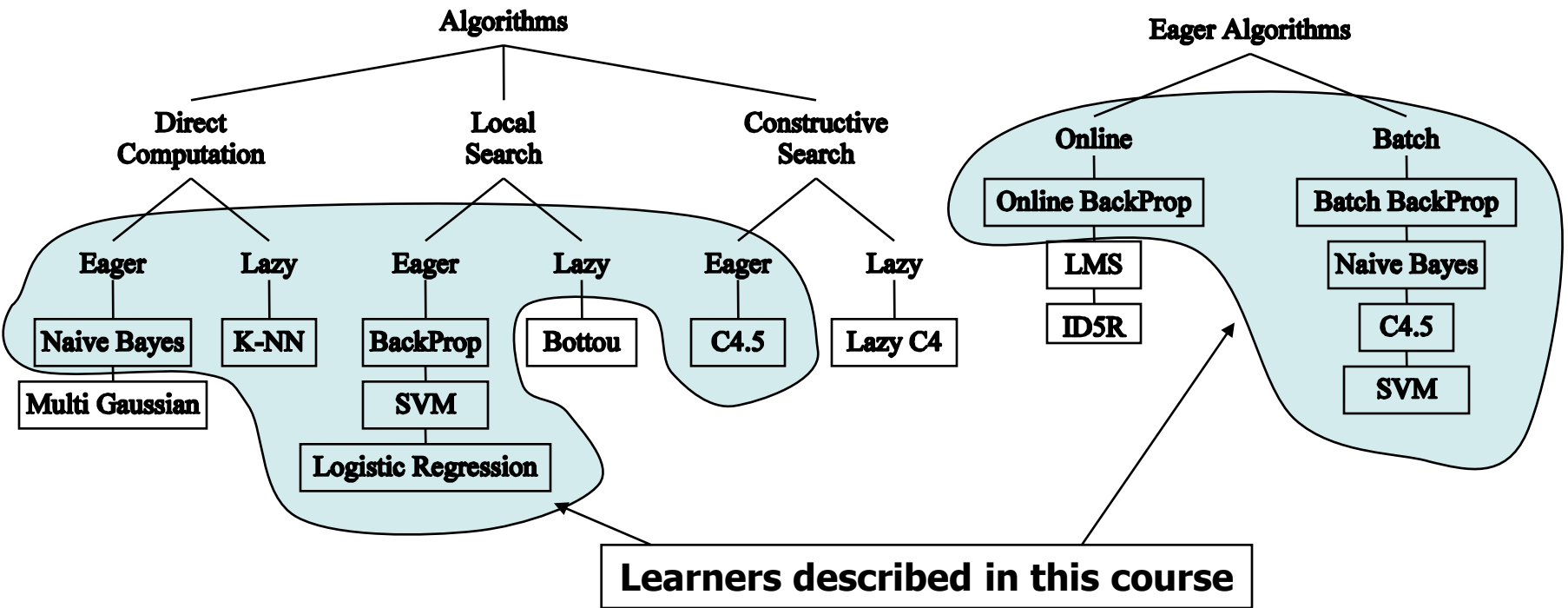
# A Framework for Hypothesis Spaces (2)



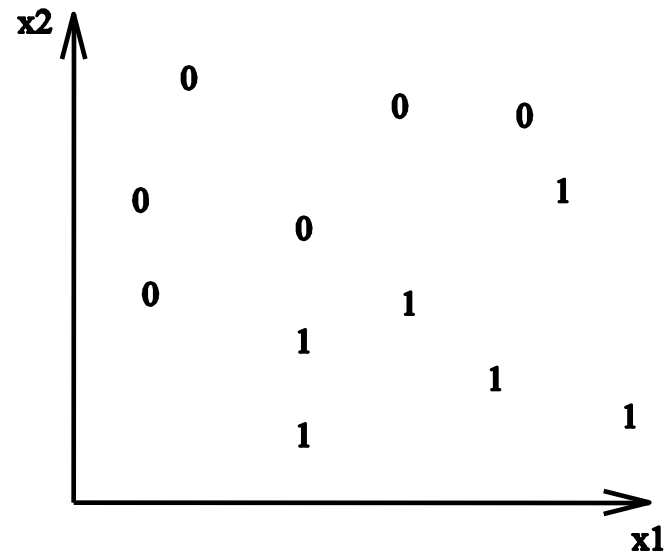
# A Framework for Learning Algorithms

- Search Procedure
  - Direct Computation: solve for the hypothesis directly
  - Local Search: start with an initial hypothesis, make small improvements until a local maximum
  - Constructive Search: start with an empty hypothesis, gradually add structure to it until a local optimum
- Timing
  - Eager: analyze training data and construct an explicit hypothesis
  - Lazy: store the training data and wait until a test data point is presented, then construct an ad hoc hypothesis to classify that one data point
- Online vs. Batch (for eager algorithms)
  - Online: analyze each training example as it is presented
  - Batch: collect examples, analyze them in a batch, output an hypothesis

# A Framework for Learning Algorithms (2)

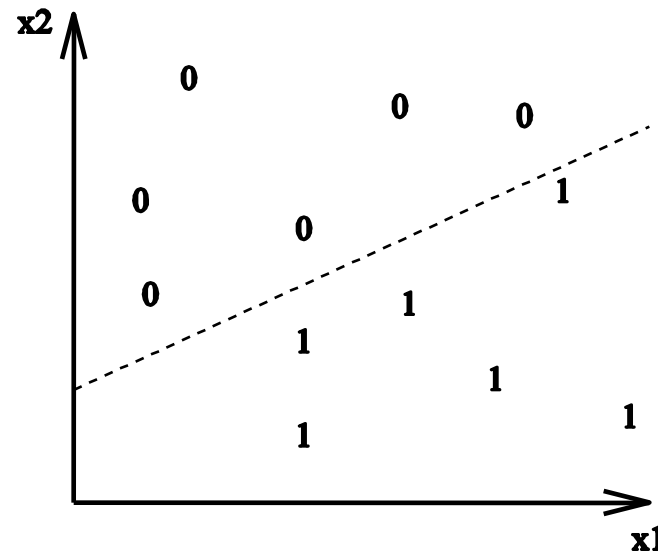


# Choosing a target function



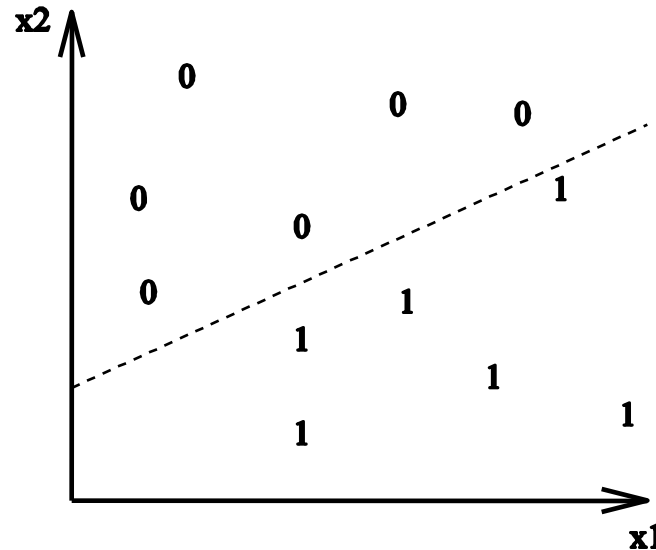
# Choosing a target function

- A linear function will do the trick



# Linear Threshold Units

- A classifier can be viewed as partitioning the input space or feature space  $X$  into decision regions



- A linear threshold unit always produces a linear decision boundary. A set of points that can be separated by a linear decision boundary is said to be linearly separable.

# Linear Threshold Units

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } w_1x_1 + \dots + w_nx_n \geq w_0 \\ -1 & \text{otherwise} \end{cases}$$

- We assume that each feature  $x_j$  and each weight  $w_j$  is a real number
- We will study three different algorithms for learning linear threshold units:
  - Perceptron: classifier
  - Logistic Regression: conditional distribution
  - Linear Discriminant Analysis: joint distribution

# What can be represented by an LTU:

- Conjunctions

$$x_1 \wedge x_2 \wedge x_4 \Leftrightarrow y$$

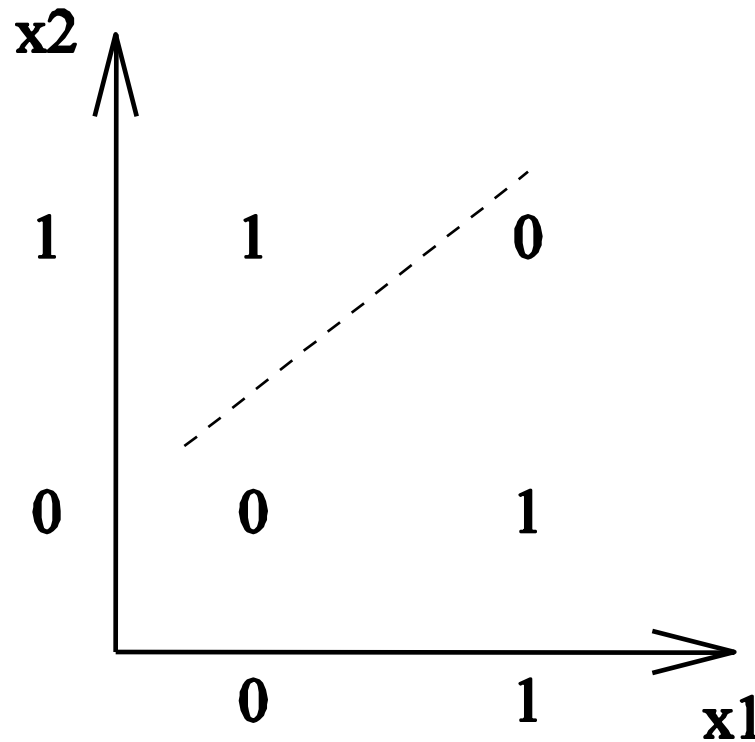
$$1 \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3 + 1 \cdot x_4 \geq 3$$

- At least  $m$ -of- $n$

$$\text{at-least-2-of}\{x_1, x_3, x_4\} \Leftrightarrow y$$

$$1 \cdot x_1 + 0 \cdot x_2 + 1 \cdot x_3 + 1 \cdot x_4 \geq 2$$

# Exclusive-OR is Not Linearly Separable



# Things that cannot be represented:

- Non-trivial disjunctions:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \Leftrightarrow y$$

$1 \cdot x_1 + 1 \cdot x_2 + 1 \cdot x_3 + 1 \cdot x_4 \geq 2$  predicts

$$f(\langle 0110 \rangle) = 1.$$

- Exclusive-OR:

$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \Leftrightarrow y$$

# A canonical representation

- Given a training example of the form  
 $(\langle x_1, x_2, x_3, x_4 \rangle, y)$
- transform it to  
 $(\langle 1, x_1, x_2, x_3, x_4 \rangle, y)$
- The parameter vector will then be  
 $\mathbf{w} = \langle w_0, w_1, w_2, w_3, w_4 \rangle.$
- We will call the *unthresholded* hypothesis  $u(\mathbf{x}, \mathbf{w})$   
 $u(\mathbf{x}, \mathbf{w}) = \mathbf{w} \cdot \mathbf{x}$
- Each hypothesis can be written  
 $h(\mathbf{x}) = \text{sgn}(u(\mathbf{x}, \mathbf{w}))$
- Our goal is to find  $\mathbf{w}$ .

# A canonical representation

- Given a training example of the form

$$(\langle x_1, x_2, x_3, x_4 \rangle, y)$$

- transform it to 
$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } w_1x_1 + \dots + w_nx_n \geq w_0 \\ -1 & \text{otherwise} \end{cases}$$

$$(\langle 1, x_1, x_2, x_3, x_4 \rangle, y)$$

- The parameter vector will then be

$$\mathbf{w} = \langle w_0, w_1, w_2, w_3, w_4 \rangle.$$

- We will call the *unthresholded* hypothesis  $u(\mathbf{x}, \mathbf{w})$

$$u(\mathbf{x}, \mathbf{w}) = \mathbf{w} \cdot \mathbf{x}$$

- Each hypothesis can be written

$$h(\mathbf{x}) = \text{sgn}(u(\mathbf{x}, \mathbf{w}))$$

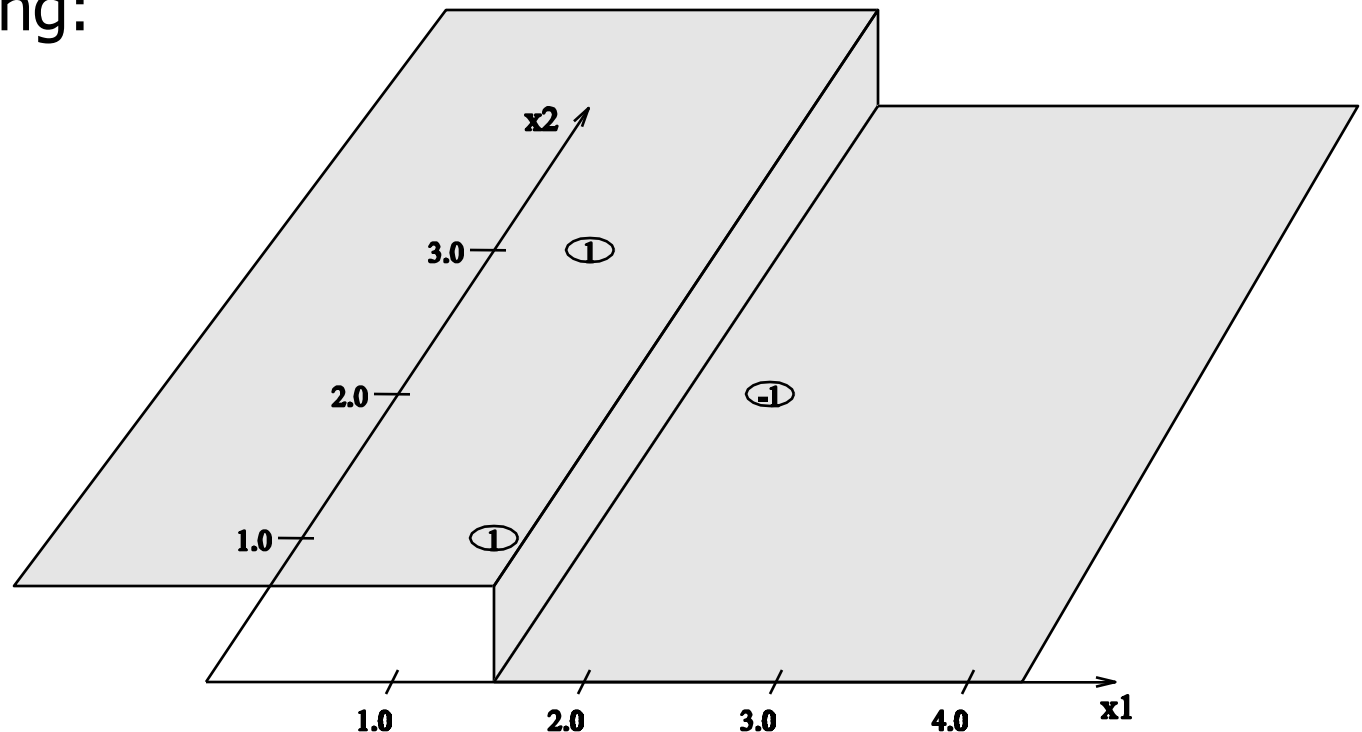
- Our goal is to find  $\mathbf{w}$ .

# The LTU Hypothesis Space

- Fixed size: There are  $O(2^{n^2})$  distinct linear threshold units over  $n$  boolean features
- Deterministic
- Continuous parameters

# Geometrical View

- Consider three training examples:  $(\langle 1.0, 1.0 \rangle, +1)$   
 $(\langle 0.5, 3.0 \rangle, +1)$   
 $(\langle 2.0, 2.0 \rangle, -1)$
- We want a classifier that looks like the following:



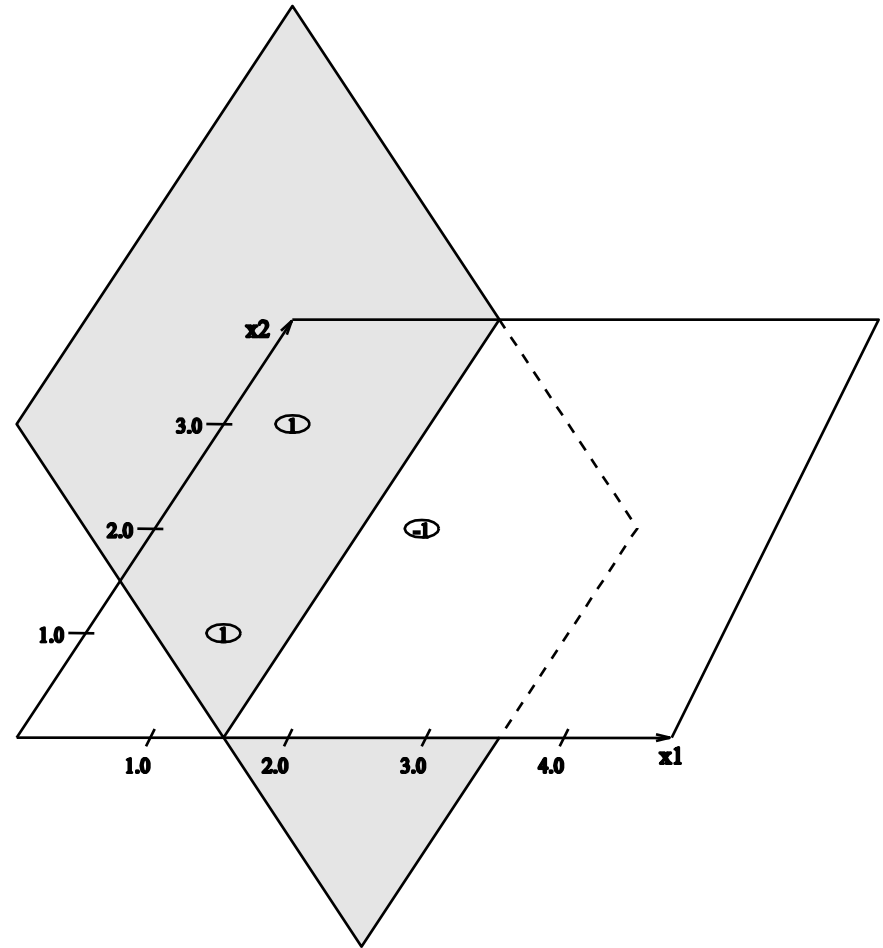
# The Unthresholded Discriminant Function is a Hyperplane

- The equation

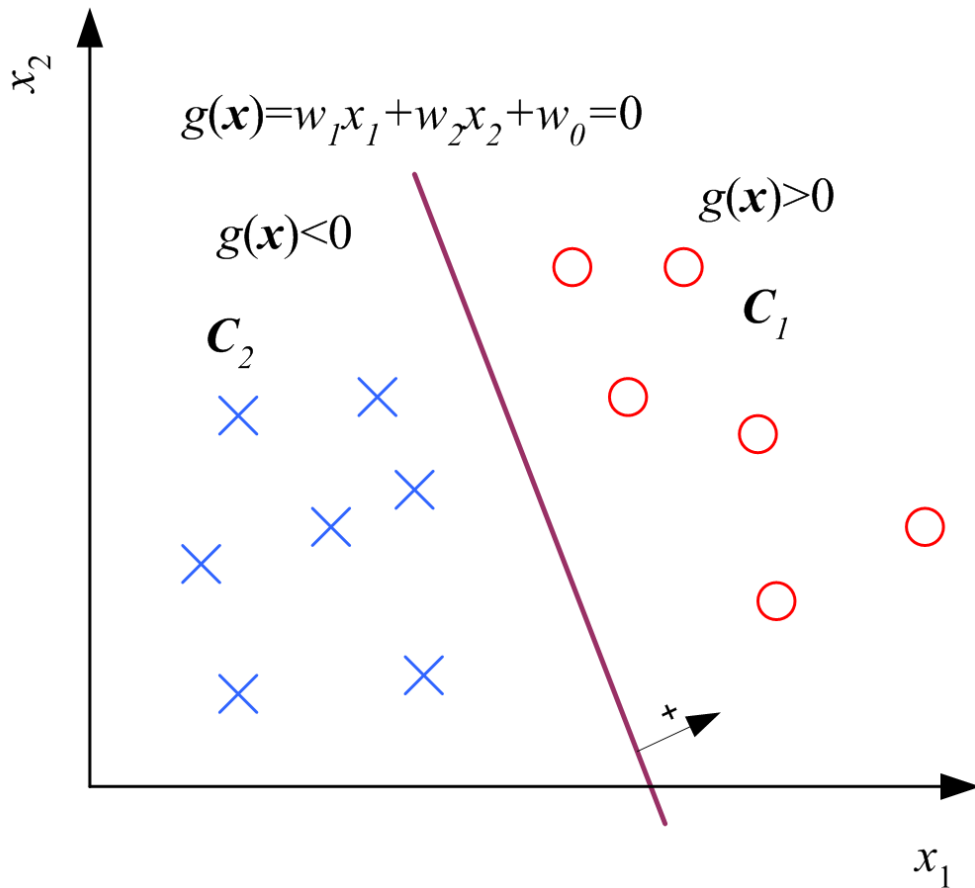
$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$$

is a plane

$$\hat{y} = \begin{cases} +1 & \text{if } g(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



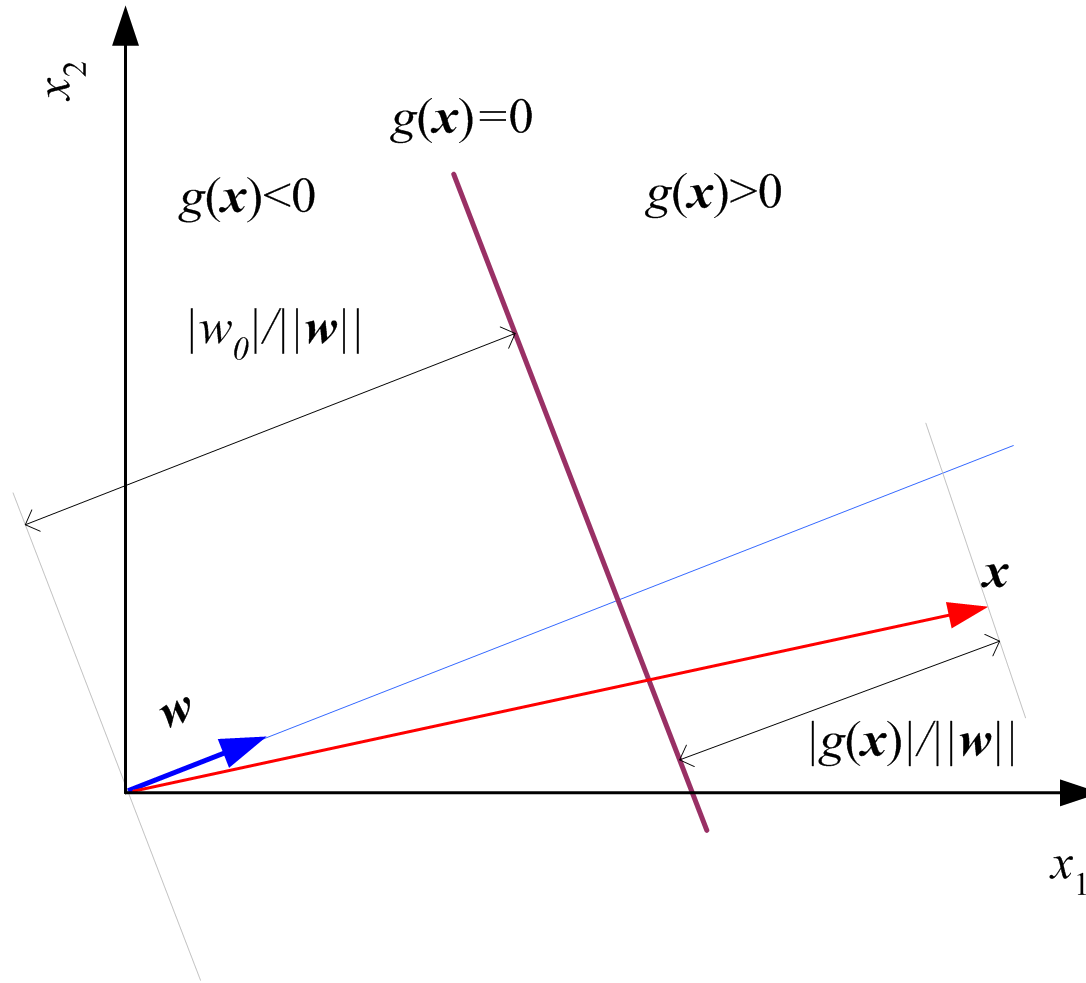
# Alternatively...



$$\begin{aligned} g(\mathbf{x}) &= g_1(\mathbf{x}) - g_2(\mathbf{x}) \\ &= (\mathbf{w}_1^T \mathbf{x} + w_{10}) - (\mathbf{w}_2^T \mathbf{x} + w_{20}) \\ &= (\mathbf{w}_1 - \mathbf{w}_2)^T \mathbf{x} + (w_{10} - w_{20}) \\ &= \mathbf{w}^T \mathbf{x} + w_0 \end{aligned}$$

choose  $\begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$

# Geometry



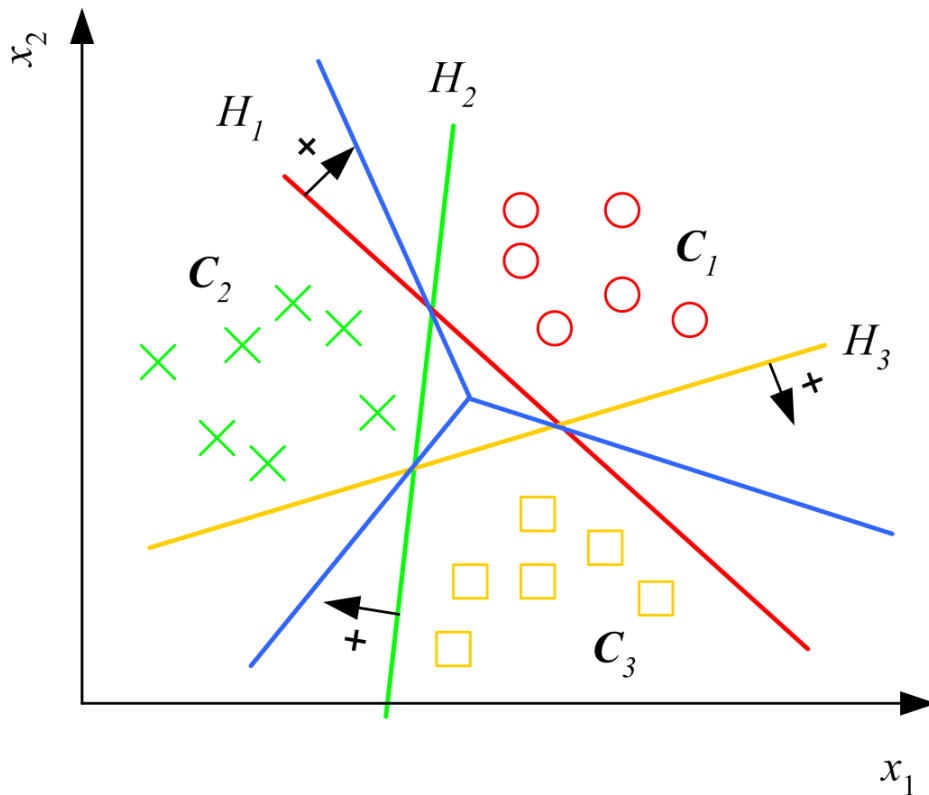
# Multiple Classes

$$g_i(\mathbf{x} | \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

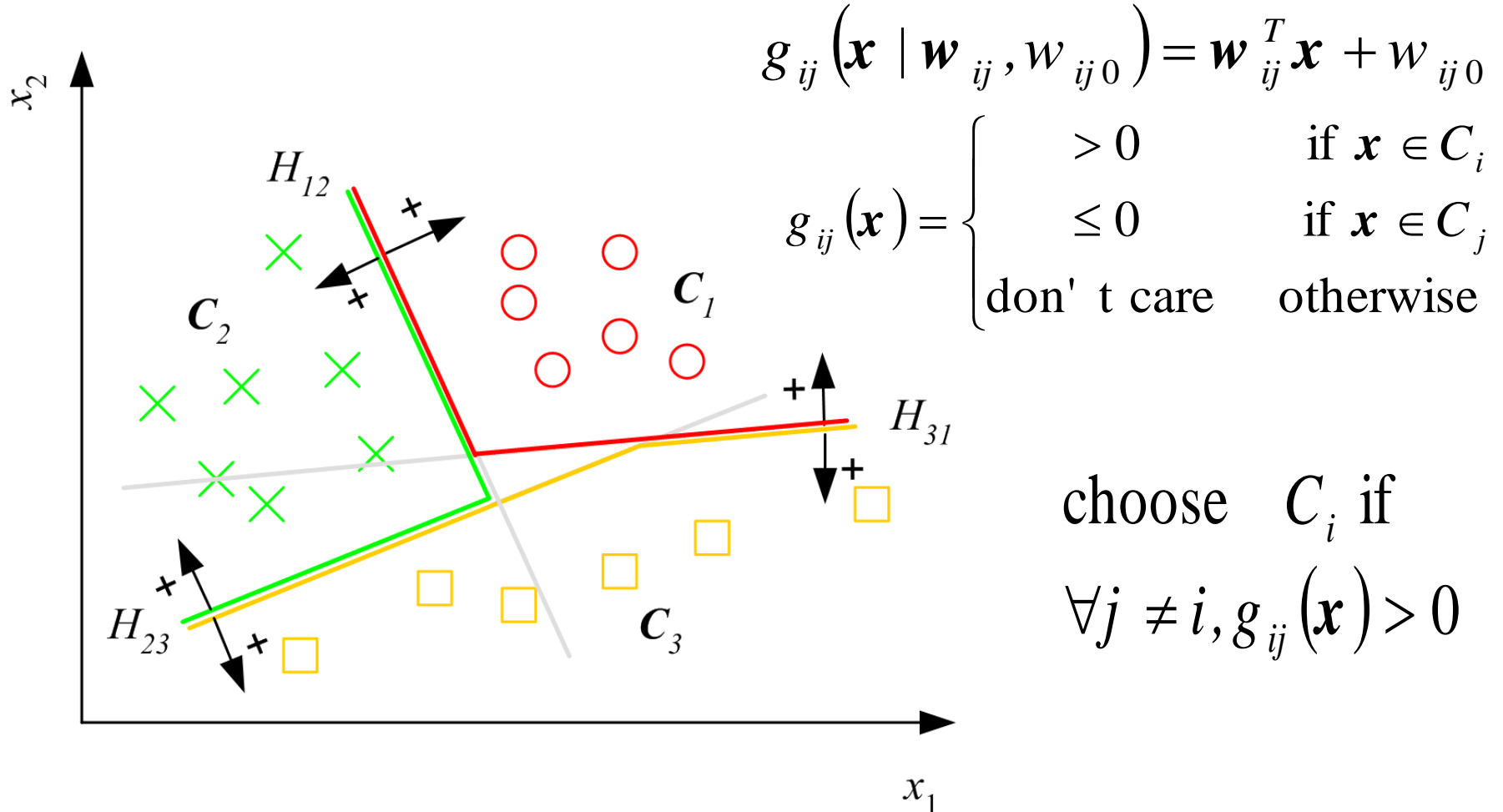
Choose  $C_i$  if

$$g_i(\mathbf{x}) = \max_{j=1}^K g_j(\mathbf{x})$$

Classes are linearly separable



# Pairwise Separation



# Machine Learning and Optimization

- When learning a classifier, the natural way to formulate the learning problem is the following:
  - Given:
    - A set of  $N$  training examples  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
    - A loss function  $L$
  - Find:
    - The weight vector  $\mathbf{w}$  that minimizes the expected loss on the training data

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(\text{sgn}(\mathbf{w} \cdot \mathbf{x}_i), y_i).$$

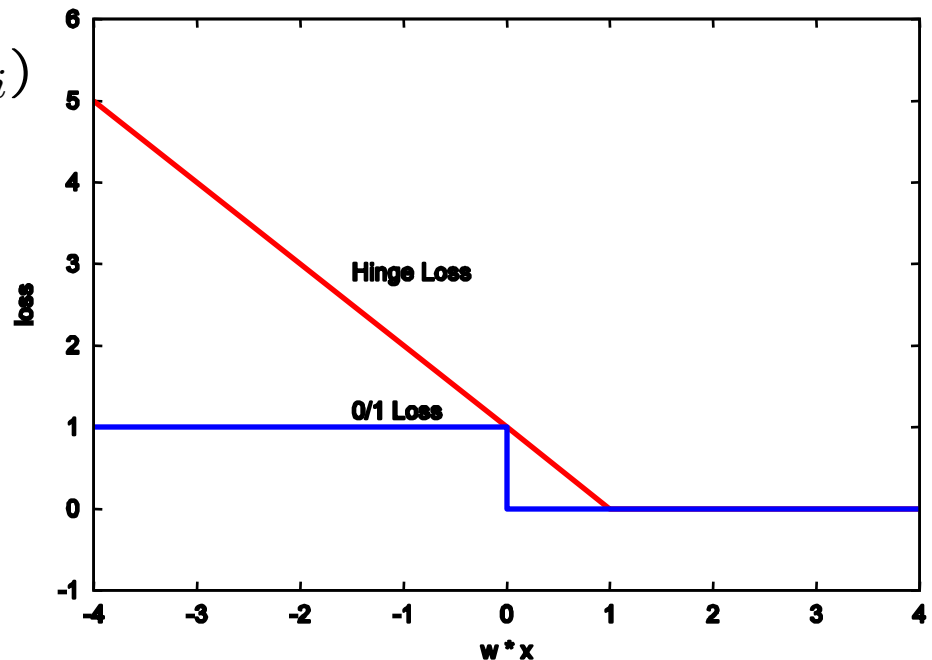
- In general, machine learning algorithms apply some optimization algorithm to find a good hypothesis. In this case,  $J$  is piecewise constant, which makes this a difficult problem

# Approximating the expected loss by a smooth function

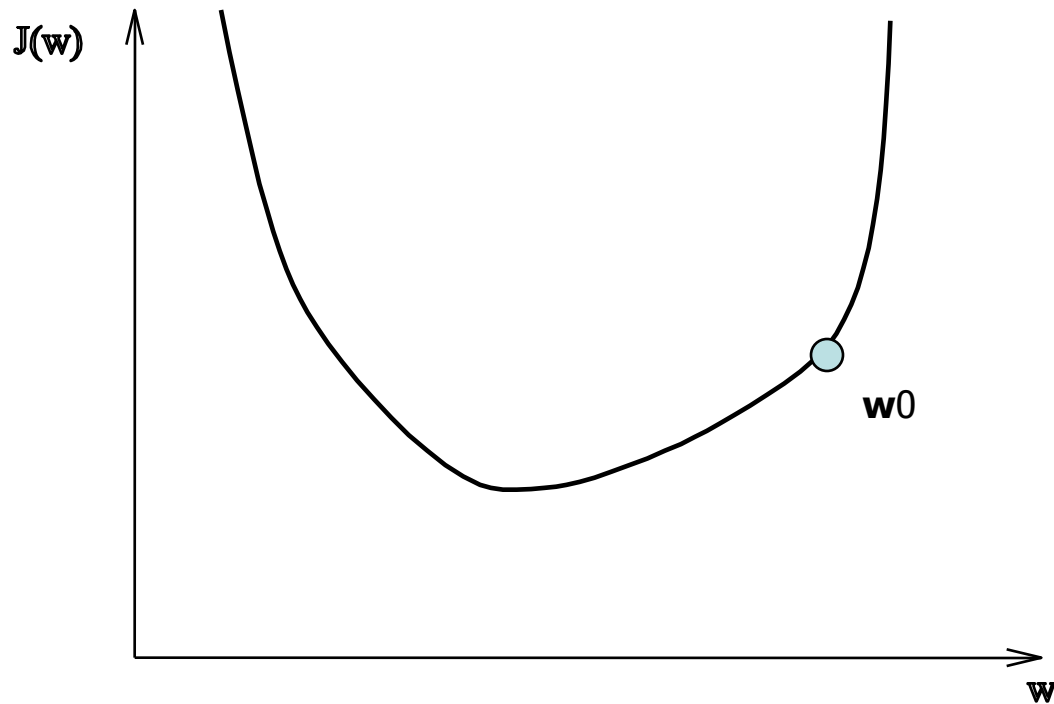
- Simplify the optimization problem by replacing the original objective function by a smooth, differentiable function. For example, consider the *hinge loss*:

$$\tilde{J}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \max(0, 1 - y_i \mathbf{w} \cdot \mathbf{x}_i)$$

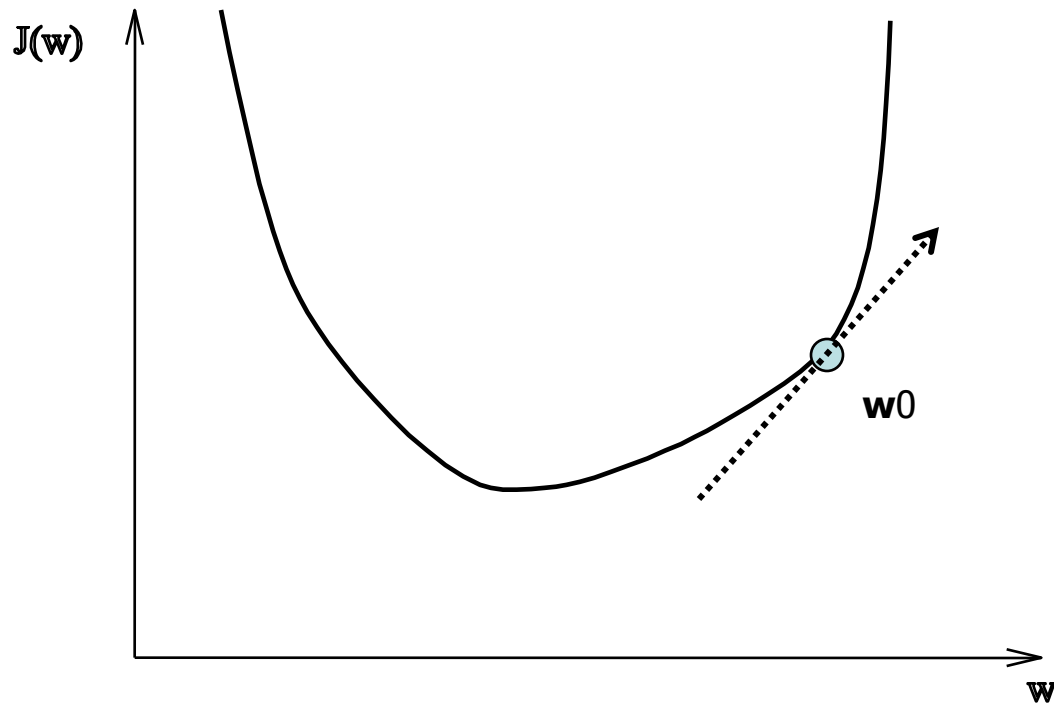
When  $y = 1$



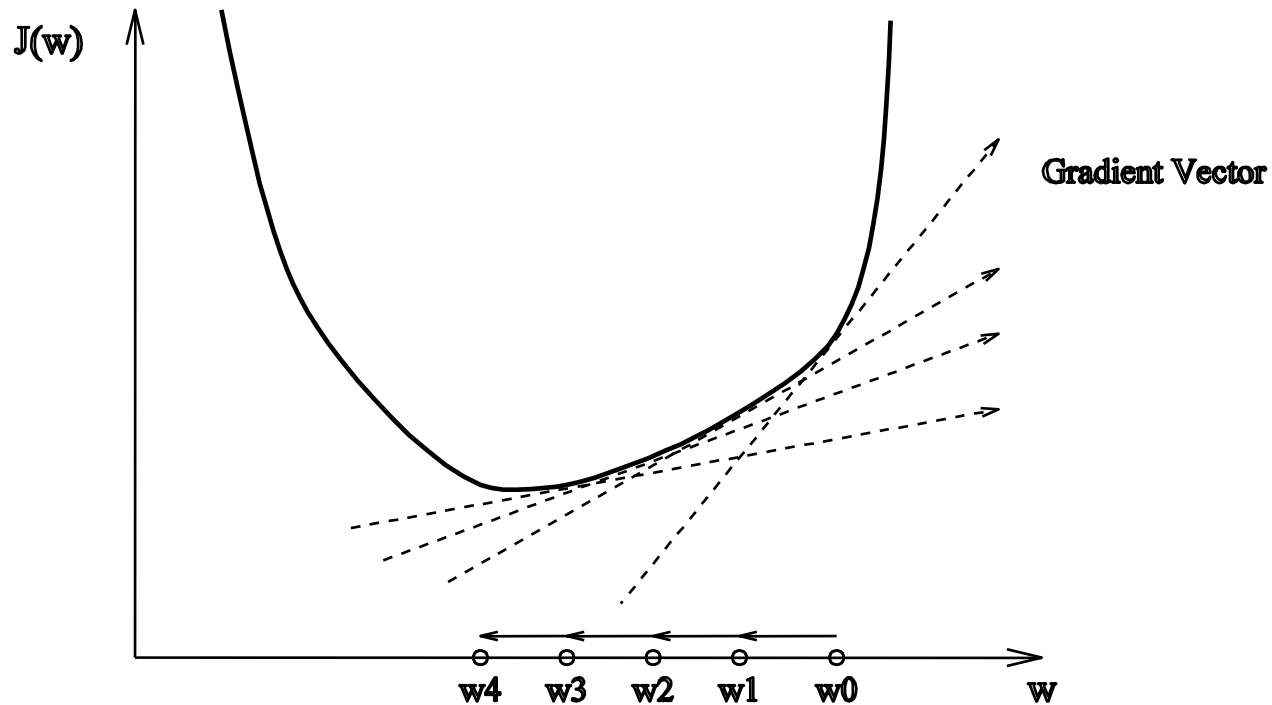
# Optimizing... what?



# Optimizing... what?



# Minimizing $\tilde{J}$ by Gradient Descent Search



- Start with weight vector  $\mathbf{w}_0$
- Compute gradient  $\nabla \tilde{J}(\mathbf{w}_0) = \left( \frac{\partial \tilde{J}(\mathbf{w}_0)}{\partial w_0}, \frac{\partial \tilde{J}(\mathbf{w}_0)}{\partial w_1}, \dots, \frac{\partial \tilde{J}(\mathbf{w}_0)}{\partial w_n} \right)$
- Compute  $\mathbf{w}_1 = \mathbf{w}_0 - \eta \nabla \tilde{J}(\mathbf{w}_0)$   
where  $\eta$  is a “step size” parameter
- Repeat until convergence

# EXAMPLE QUIZ