

# Math Foundations for ML

Computational

10-606



Geoff Gordon

# About us

- Me: Geoff Gordon
- TAs:
  - ▶ Aditya Paul
  - ▶ Aishwarya Jadhav
  - ▶ Xiaoyu Xu

# Notes and reminders

- Location: CUC McKenna (when in person)
- Most weeks: two lectures, one “other”
  - ▶ This week: virtual-only lectures today and Wednesday
  - ▶ Lab 0 (optional) on Friday: Python review
- <https://www.cs.cmu.edu/~ggordon/10606s22/syllabus-and-lecture-outline.html>
- Ask questions, participate, help one another learn! <https://xkcd.com/1053/>

# What is ML?

## Speech Recognition

### 1. Learning to recognize spoken words

THEN	NOW
<p>"...the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal...neural network methods...hidden Markov models..."</p>	

(Mitchell, 1997)

Source: <https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults>

## Robotics

### 2. Learning to drive an autonomous vehicle

THEN	NOW
<p>"...the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars..."</p>	

(Mitchell, 1997)

waymo.com

## Games / Reasoning

### 3. Learning to beat the masters at board games

THEN	NOW
<p>"...the world's top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself..."</p>	

(Mitchell, 1997)

## Computer Vision

### 4. Learning to recognize images

THEN	NOW
<p>"...The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors..."</p>	

(LeCun et al., 1995)

Images from <https://blog.openai.com/generative-models/>

## Learning Theory

### 5. In what cases and how well can we learn?

**Sample Complexity Results**

**Definition 0.4.** The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about...

	Realizable	Agnostic
Finite [N]	$N \geq \frac{1}{\epsilon} \ln \frac{1}{\delta} \left( \frac{1}{\epsilon} \ln \frac{1}{\delta} + \ln \frac{1}{\delta} \right)$ to find examples sufficient so that with probability $1 - \delta$ for all $s \in S$ we have that $ R(s) - R^*(s)  \leq \epsilon$	$N \geq \frac{1}{\epsilon^2} \ln \frac{1}{\delta} \left( \frac{1}{\epsilon} \ln \frac{1}{\delta} + \ln \frac{1}{\delta} \right)$ to find examples sufficient so that with probability $1 - \delta$ for all $s \in S$ we have that $ R(s) - R^*(s)  \leq \epsilon$
Infinite [N]	$N = \frac{1}{\epsilon^2} \ln \frac{1}{\delta} \left( \frac{1}{\epsilon} \ln \frac{1}{\delta} + \ln \frac{1}{\delta} \right)$ to find examples sufficient so that with probability $1 - \delta$ for all $s \in S$ we have that $ R(s) - R^*(s)  \leq \epsilon$	$N = \frac{1}{\epsilon^2} \ln \frac{1}{\delta} \left( \frac{1}{\epsilon} \ln \frac{1}{\delta} + \ln \frac{1}{\delta} \right)$ to find examples sufficient so that with probability $1 - \delta$ for all $s \in S$ we have that $ R(s) - R^*(s)  \leq \epsilon$

**Two types of Error**

- ① Test Error (aka expected risk) (aka Statistical Error)
- ② Train Error (aka empirical risk)

$R(s) = \sum_{x \in S} p(x) \ell(x, h(x))$  ← only unknown

$\hat{R}(s) = \sum_{x \in S} \frac{1}{N} \ell(x, h(x))$  ← known, computable

**PK Learning**

Q: Can we bound  $R(s)$  in terms of  $\hat{R}(s)$ ?

A: Yes!

PK tells us: Probably Approximately Correct (PAC) theory yields hypothesis  $h$ , which is approximately correct  $R(h) \approx 0$  with high probability  $P(R(h) \approx 0) \approx 1$

Def: PAC Criterion  $P(R(h) - R^*(h) \leq \epsilon) \geq 1 - \delta$

1. How many examples do we need to learn?
2. How do we quantify our ability to generalize to unseen data?
3. Which algorithms are better suited to specific learning settings?



- Dual problem (derivation):

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_{j=1}^n \alpha_j [(\mathbf{w} \cdot \mathbf{x}_j + b) y_j - 1]$$

$$\alpha_j \geq 0, \forall j$$

$\alpha$  - weights on training pts (n-dim problem)

## Dual SVM – linearly separable case

- Dual problem (derivation):

$$\max_{\alpha_j \geq 0} d(\alpha)$$

$$\max_{\alpha} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_j \alpha_j [(\mathbf{w} \cdot \mathbf{x}_j + b) y_j - 1]$$

$$\alpha_j \geq 0, \forall j$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_j \alpha_j \mathbf{x}_j y_j$$

$$\rightarrow \frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\rightarrow \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_j \alpha_j y_j = 0$$

$$\frac{\partial L}{\partial b} = \sum_j \alpha_j y_j$$

# Why this course?

## Dual SVM – linearly separable case

- Dual problem:

$$\sum_j \alpha_j b y_j = b \sum_j \alpha_j y_j = 0$$

$$\max_{\alpha} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_j \alpha_j [(\mathbf{w} \cdot \mathbf{x}_j + b) y_j - 1]$$

$$\alpha_j \geq 0, \forall j$$

$$\Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j \quad \Rightarrow \sum_j \alpha_j y_j = 0$$

$$\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_j \alpha_j \left( \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}_j \right) y_j + \sum_j \alpha_j$$

$$\sum_j \alpha_j - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j =: d(\alpha)$$

# This course is a bit odd

- Most courses:
  - ▶ teach one semester's worth of material in one semester
  - ▶ go over enough examples of every topic that you can learn it from scratch with no trouble
- This course:
  - ▶ several semester's worth of material in one!
  - ▶ assumption: you've seen at least some of it before, need to work on rest
  - ▶ not enough examples on any one topic to learn from scratch: you must ask questions and seek out your own material
- Benefit: we cover a lot of ground, help build a strong base for ML courses

# Formal systems

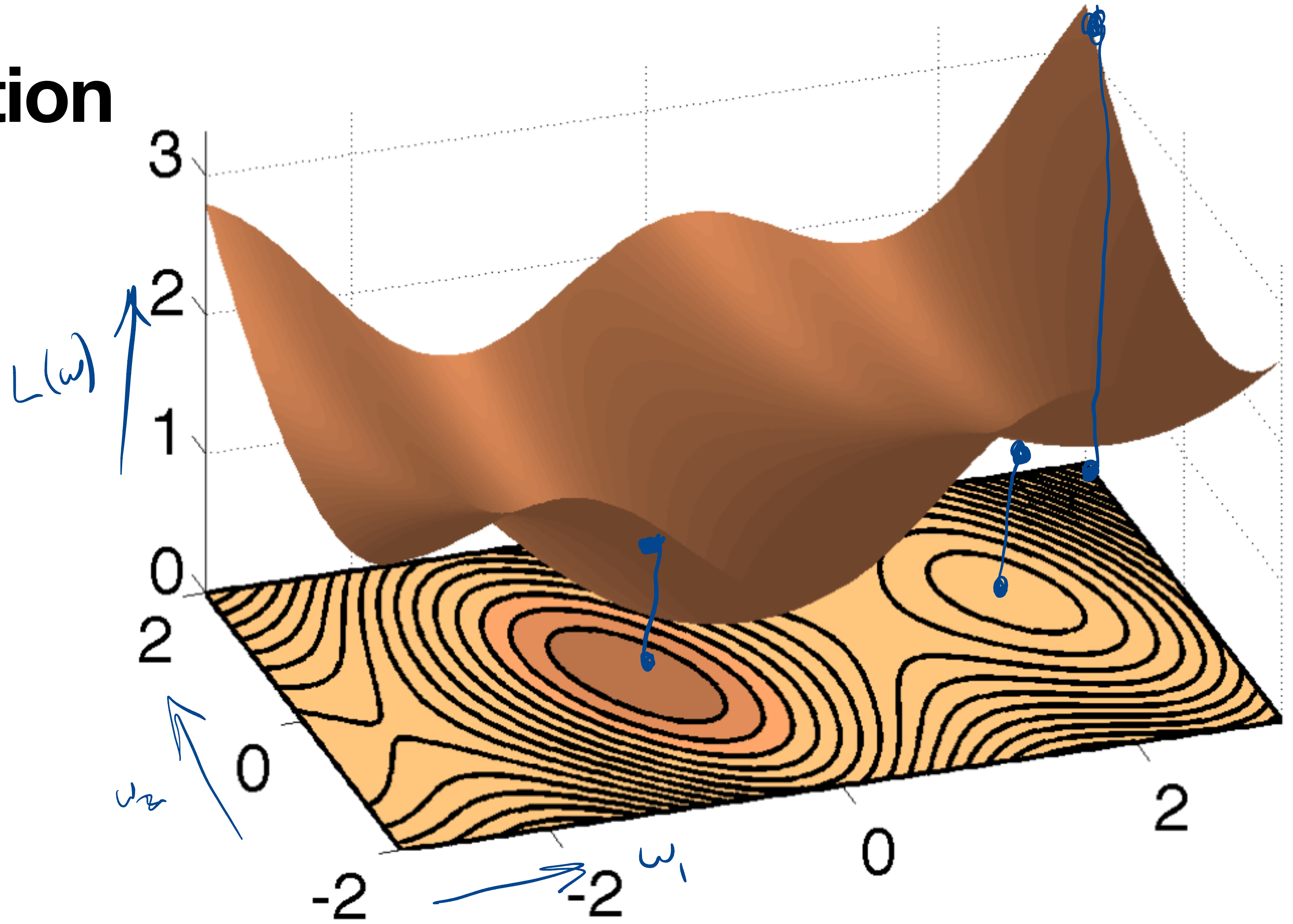
Objects  $\in$  Data type

↳ API

# Optimization

$$\min_w L(w)$$

$$w \in \mathbb{R}^2$$





# Optimization from data

$$P(x_i | w)$$

$$X \quad x_1, x_2 \dots x_T$$

$$P(w | X) \propto P(x | w) \frac{P(w)}{P(x)}$$

$$\hookrightarrow P(x_1 | w) P(x_2 | w) \dots P(x_T | w)$$

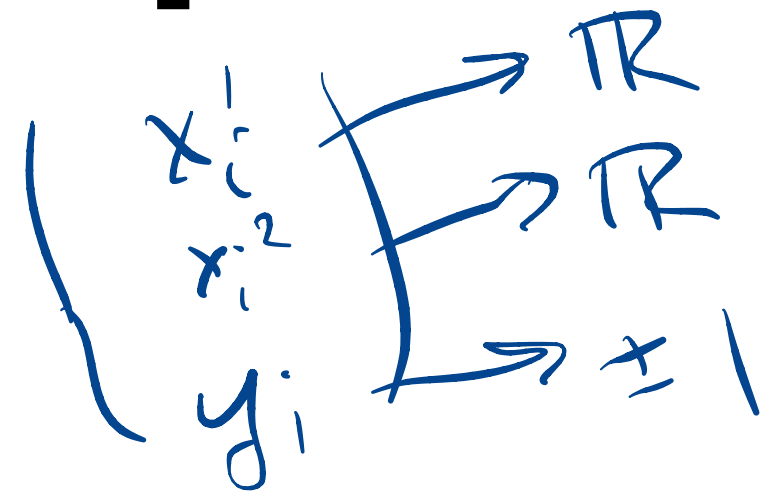
$$\begin{array}{c} L(w) \\ \downarrow \\ \max_w P(w | X) \end{array}$$

$$P(w | X, Y) \propto P(Y | X, w) \frac{P(w)}{P(X)}$$

$$P(y_i | x_i, w)$$

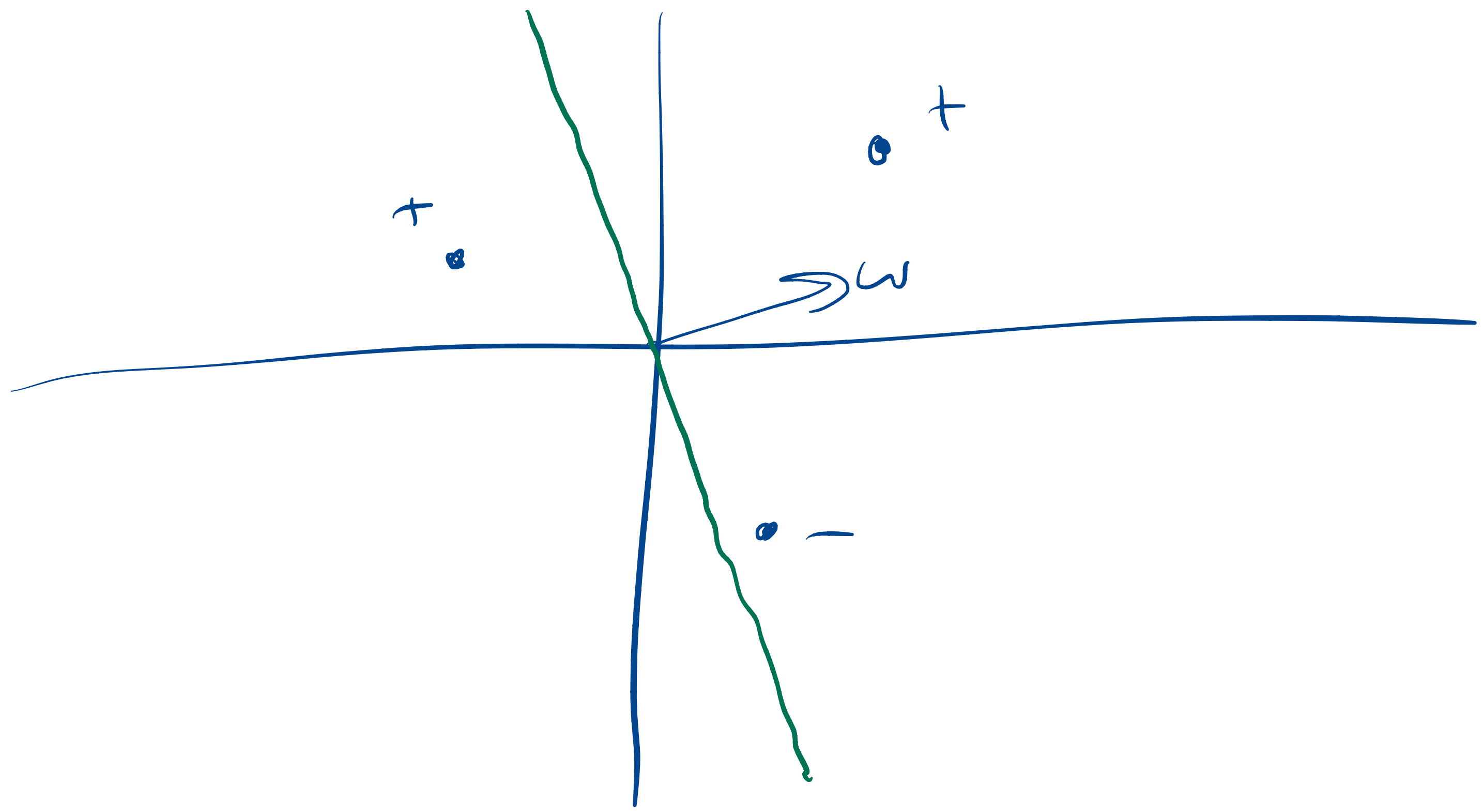
$$\begin{array}{c} P(y_1 | x_1, w) P(y_2 | x_2, w) \\ \dots P(y_T | x_T, w) \end{array} \frac{P(w)}{P(Y)}$$

# Example: classification



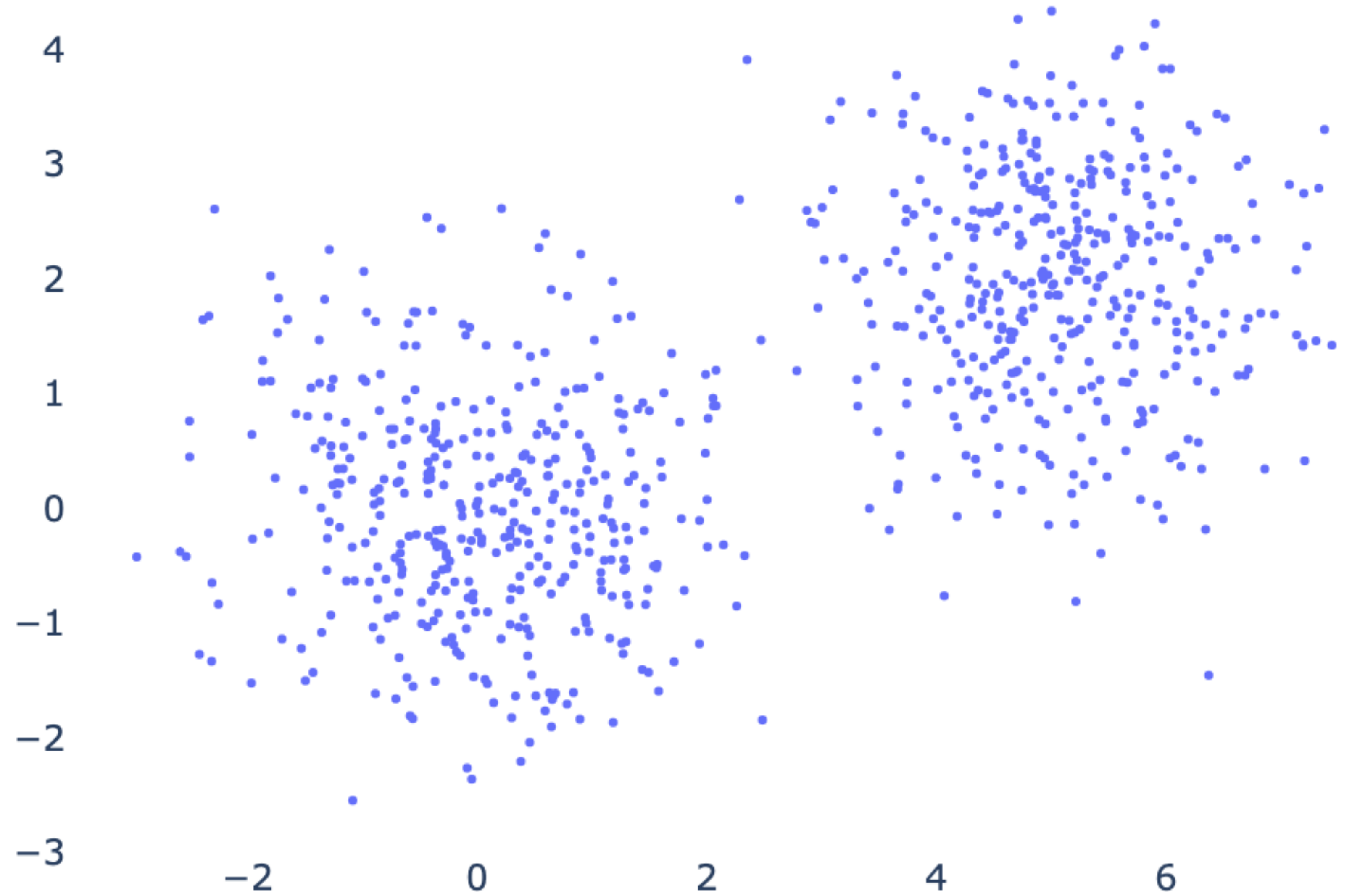
$$L(\omega) = \prod_i P(y_i | x_i, \omega)$$

Classifier:  
input  $x \in \mathbb{R}^k$   
output  $y \in \{\pm 1\}$



# Example: density estimation

$$p(\omega) \propto \prod_{i=1}^n p(x_i | \omega)$$

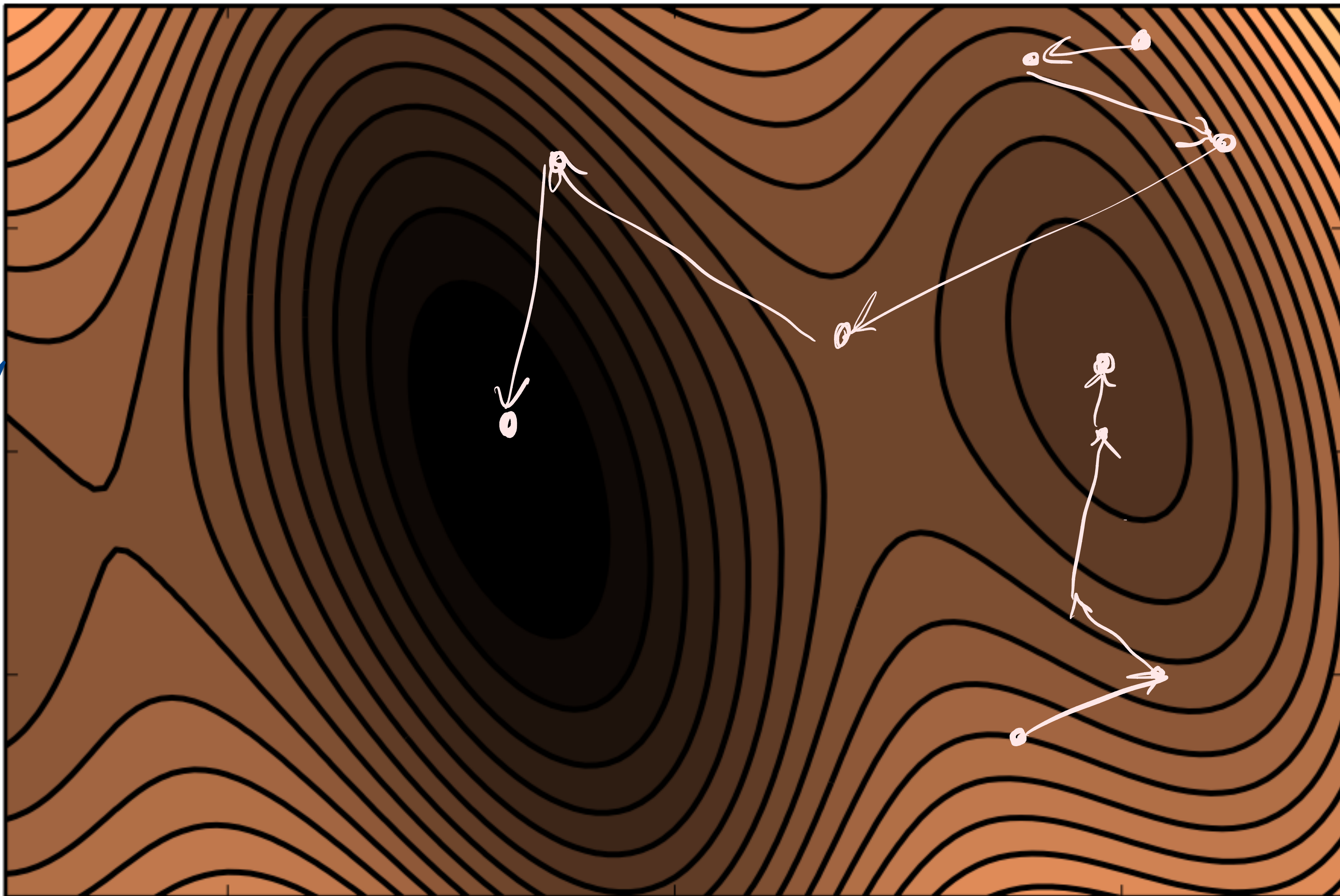




Optimizer:  
tries to  
follow  
steps  
of decreasing  
 $L(w)$

$w_2$   
↑

→  $w_1$





# Gradient descent

start at  $w^{(0)}$

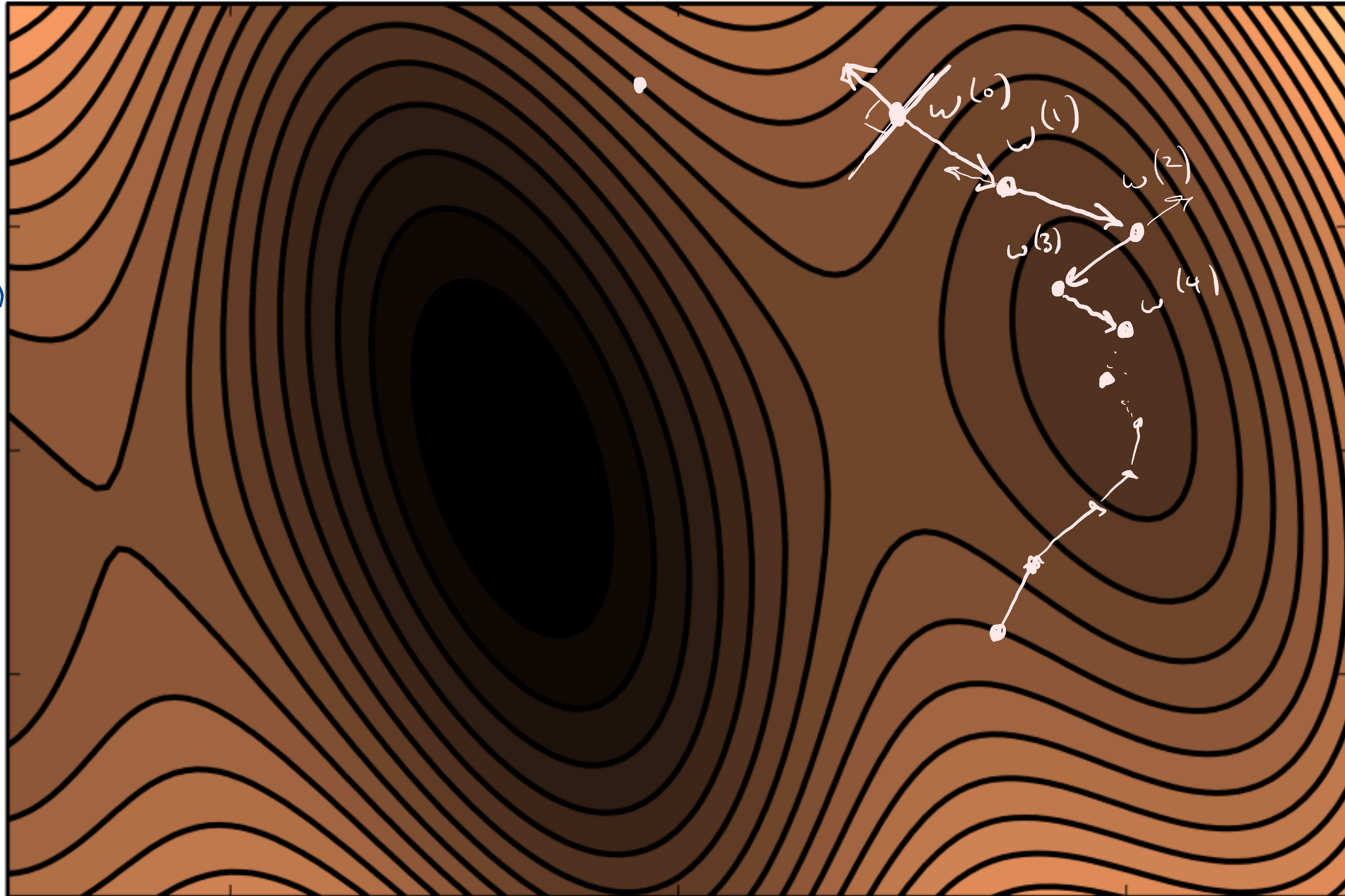
repeat  $i = 1, 2, \dots$

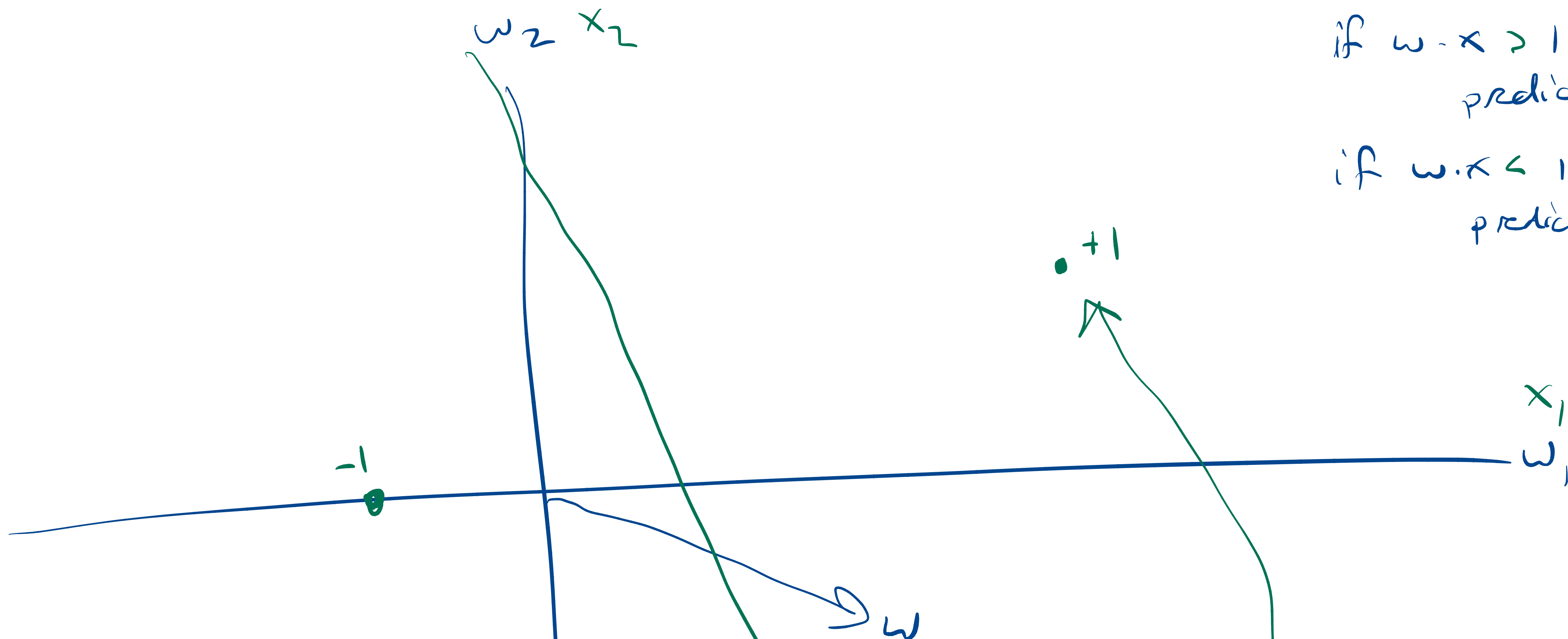
$$w^{(i)} = w^{(i-1)} - \eta g^{(i)}$$

$$g^{(i)} = \nabla L(w^{(i-1)})$$

$$\eta \in \mathbb{R}$$

$$\eta > 0$$





if  $w \cdot x > 1$   
 predict +1  
 if  $w \cdot x < 1$   
 predict -1

decision boundary:  
 $\{x \mid w \cdot x = 1\}$

if  $w$  is over here  
 we get this example  
 right  
 $\{w \mid w \cdot x > 1\}$



**Example**  
go to [repl.it](http://repl.it)

