

# Lecture 11: Support Vector Machines

Winter 2018

Kai-Wei Chang

CS @ UCLA

[kw+cm146@kwchang.net](mailto:kw+cm146@kwchang.net)

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

# Announcement

- ❖ Hw3 will be released tonight
- ❖ Solution and grading rubrics will be released tonight
- ❖ Please do a survey in CCLE to help to adjust this course

# Recap: The Perceptron Algorithm [Rosenblatt 1958]

Given a training set  $\mathcal{D} = \{(x, y)\}$

1. Initialize  $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For  $(x, y)$  in  $\mathcal{D}$ :
3.     if  $y(w^\top x) \leq 0$
4.          $w \leftarrow w + yx$
- 5.
6. Return  $w$

Assume  $y \in \{1, -1\}$

Prediction:  $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

# Recap: The Marginal Perceptron Algorithm

Given a training set  $\mathcal{D} = \{(x, y)\}$

1. Initialize  $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For  $(x, y)$  in  $\mathcal{D}$ :
3.     if  $y(w^\top x) \leq \gamma$
4.          $w \leftarrow w + yx$
- 5.
6. Return  $w$

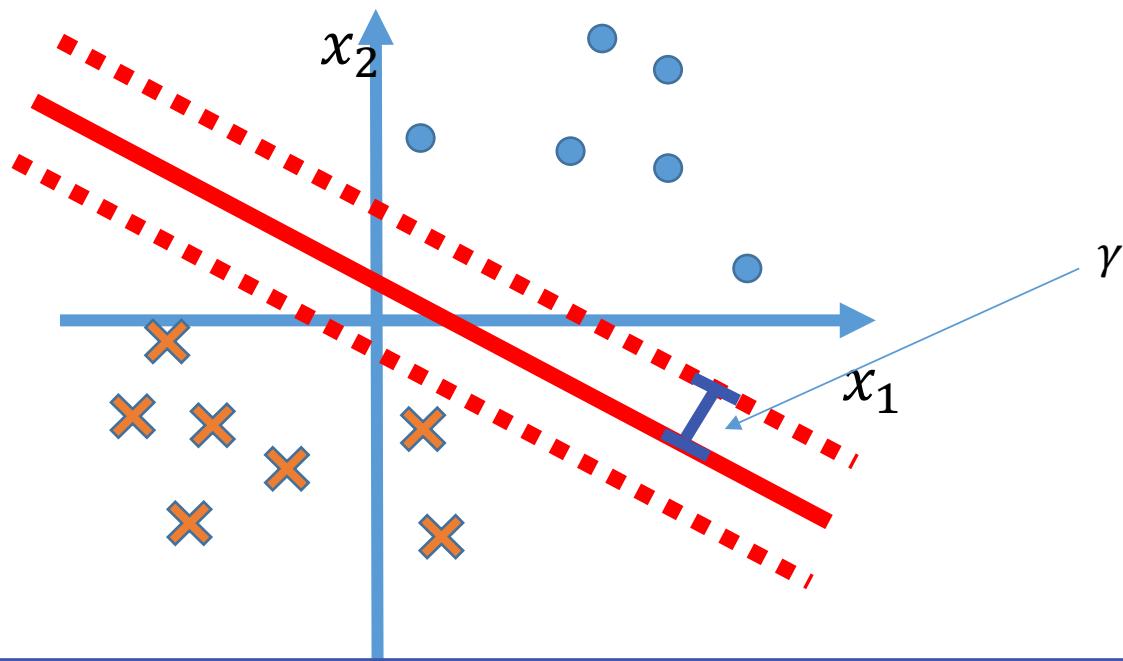
Assume  $y \in \{1, -1\}$

$\gamma \geq 0$  is a hyper-parameter

Prediction:  $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

# Marginal Perceptron



Is there a way to find out the best  $\gamma$  automatically?

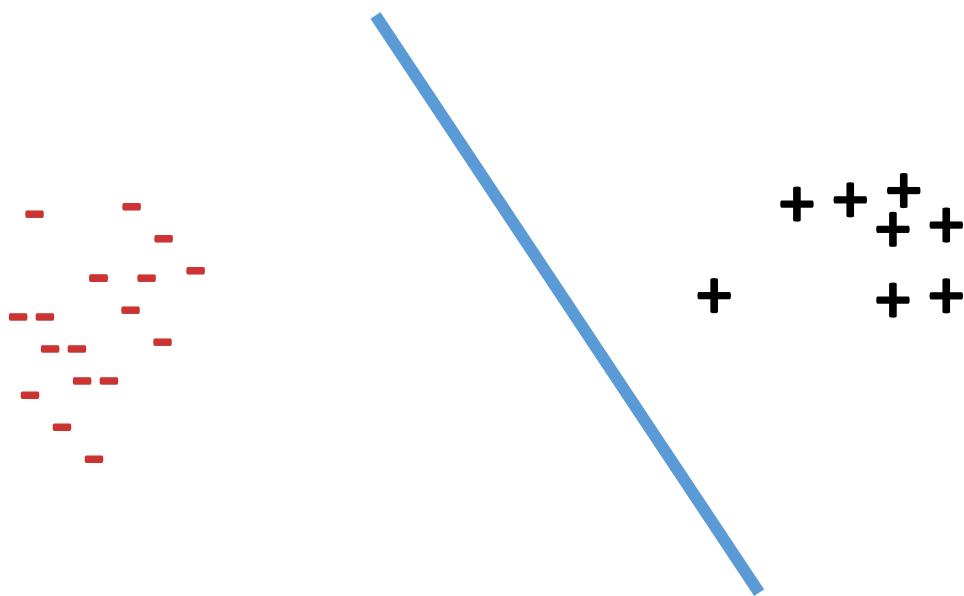
# This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels



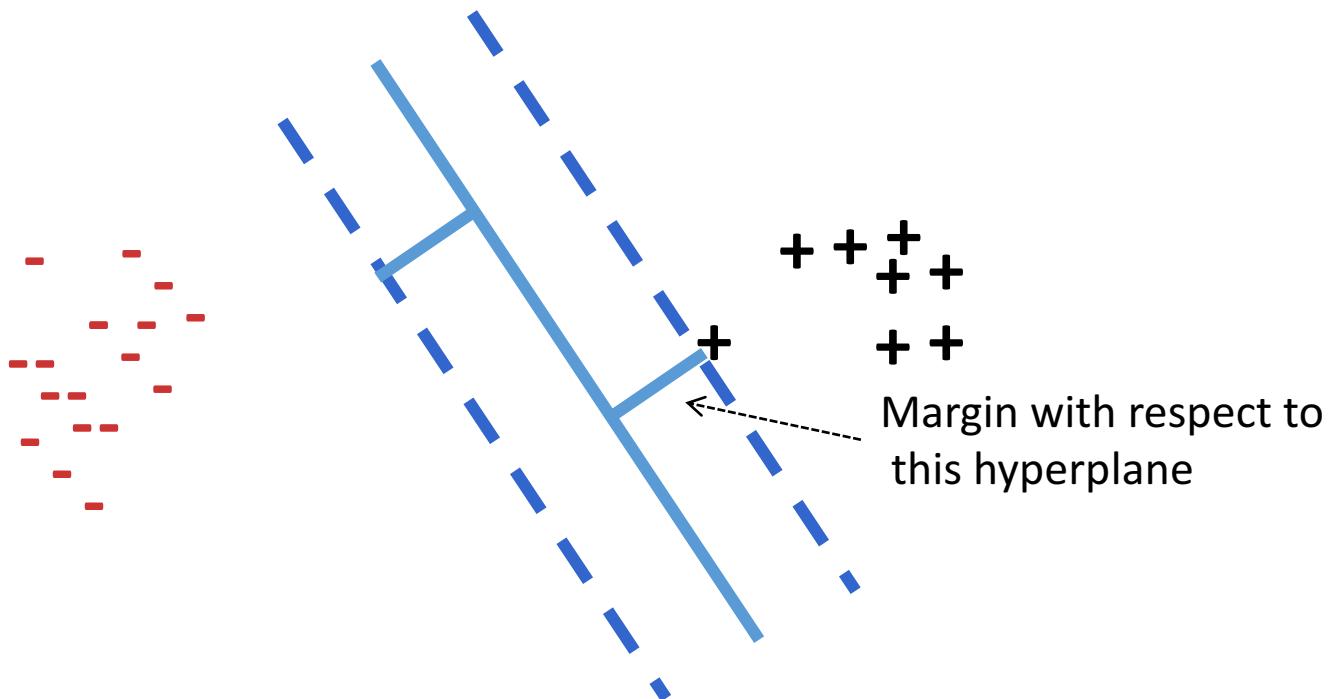
# Recall: Margin

The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

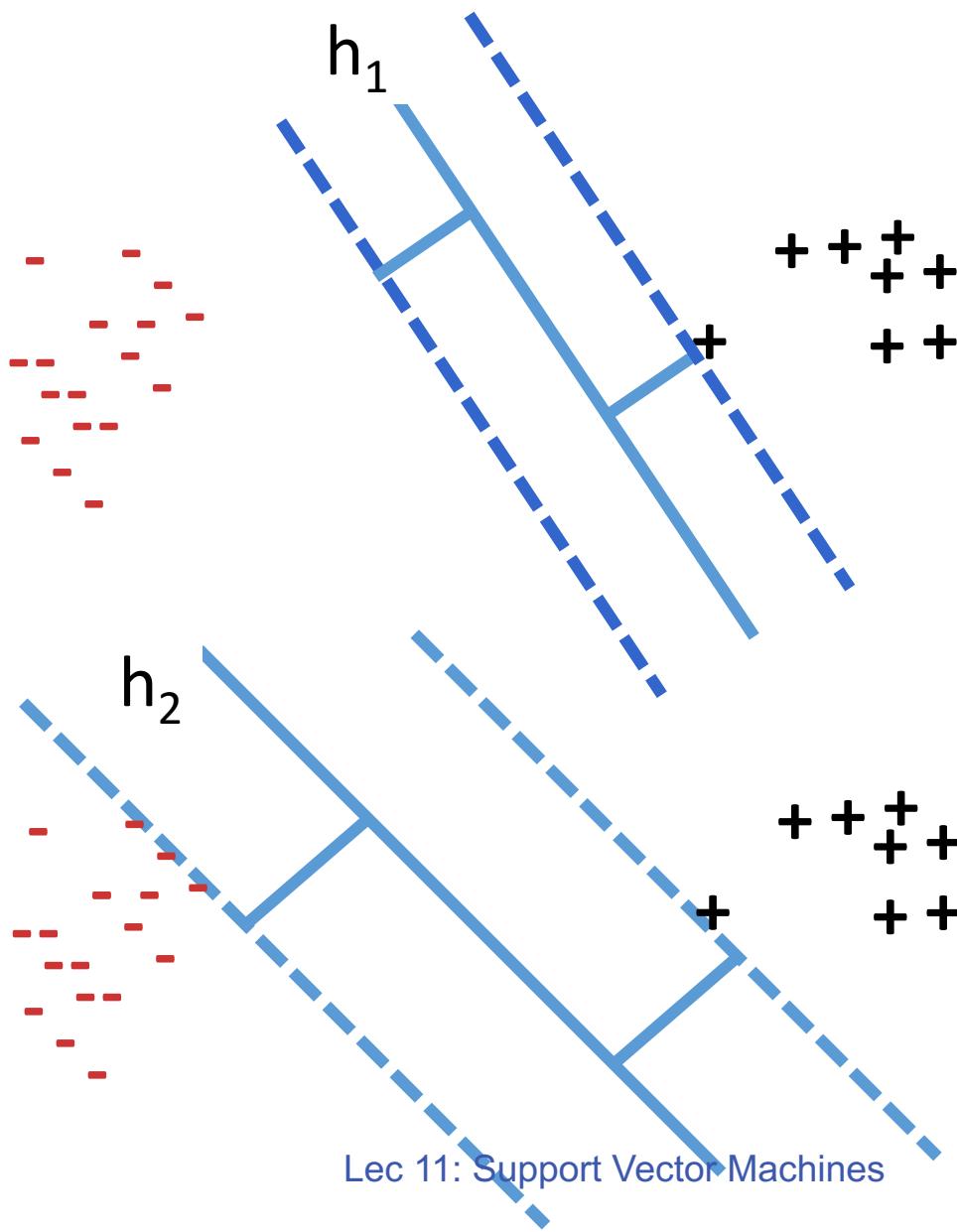


# Recall: Margin

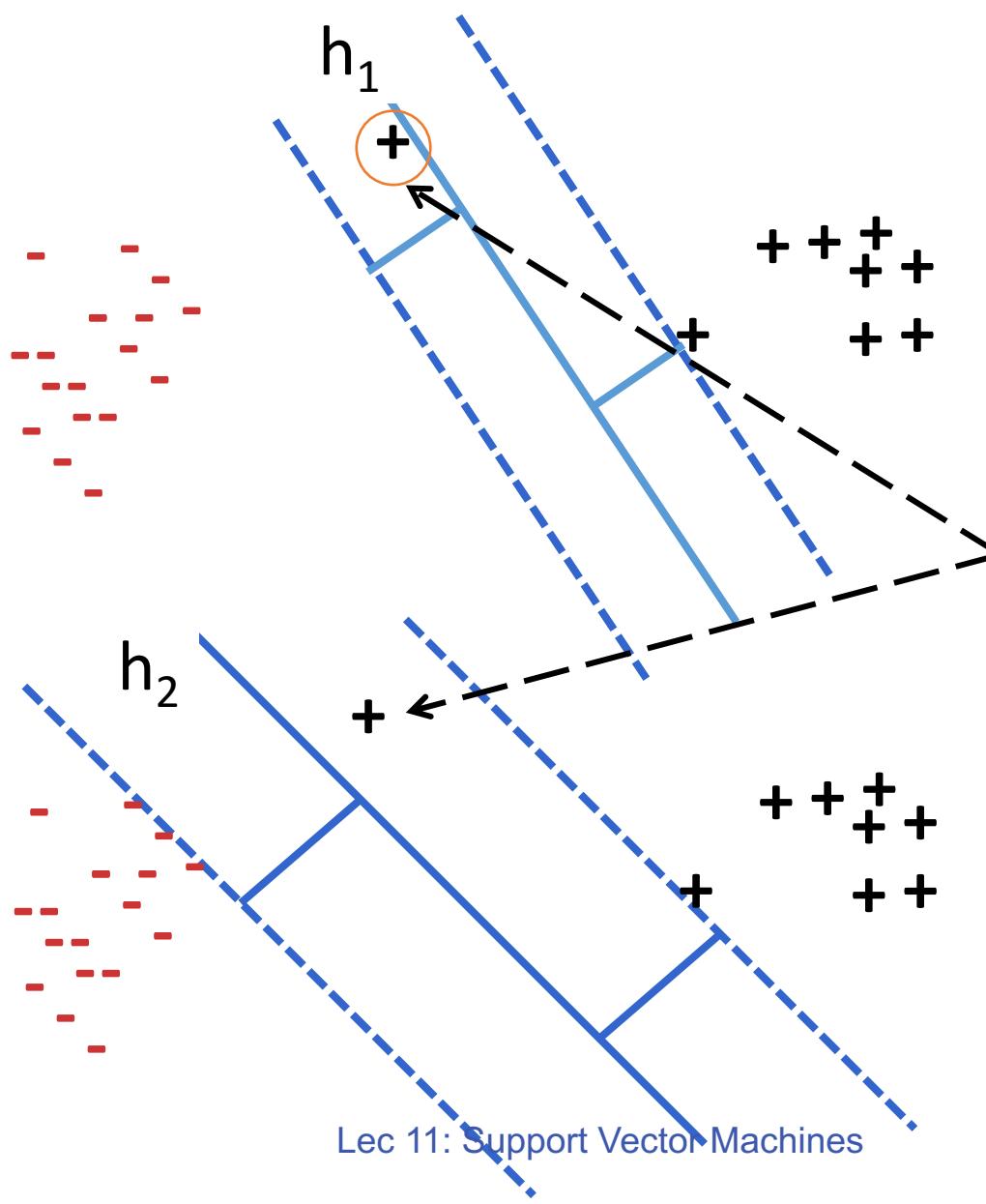
The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



# Which line is a better choice? Why?



# Which line is a better choice? Why?



A new example, not from the training set might be misclassified if the margin is smaller

# Advanced topic: (not included in the exam) Data dependent VC dimension

## Theorem (Vapnik):

- ❖ Let  $H$  be the set of linear classifiers that separate the training set by a margin at least  $\gamma$
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖  $R$  is the radius of the smallest sphere containing the data

# Advanced topic: (not in the exam) Data dependent VC dimension

## Theorem (Vapnik):

- ❖ Let  $H$  be the set of linear classifiers that separate the training set by a margin at least  $\gamma$
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖  $R$  is the radius of the smallest sphere containing the data

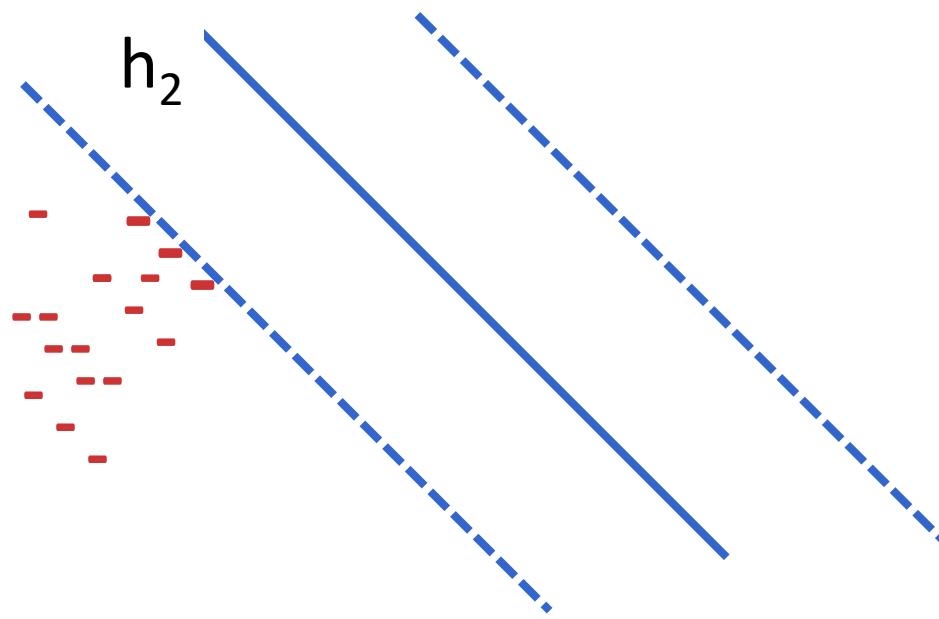
Larger margin  $\rightarrow$  Lower VC dimension

Lower VC dimension  $\rightarrow$  Better generalization bound

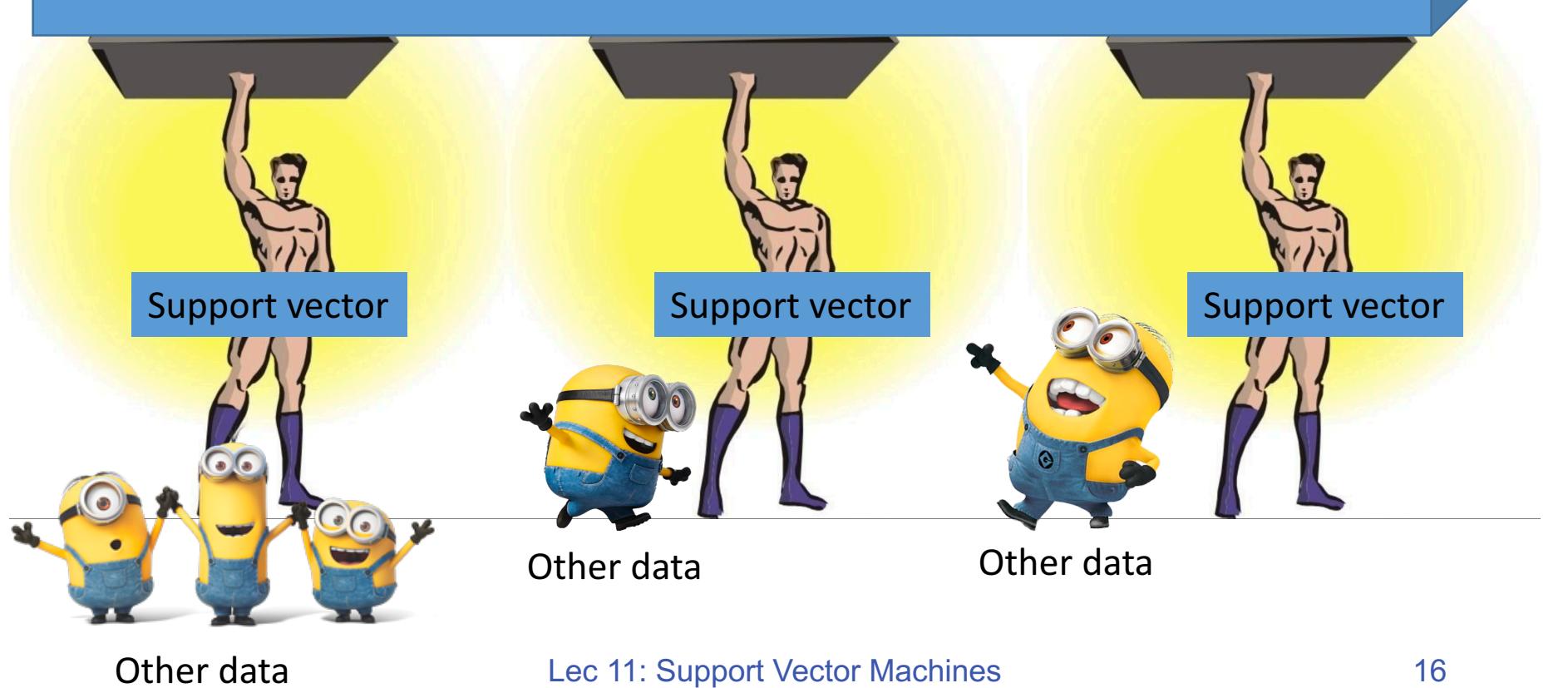
# Learning strategy

Find the linear separator that maximizes the margin

# Why it called support vector machines?



# Decision Boundary

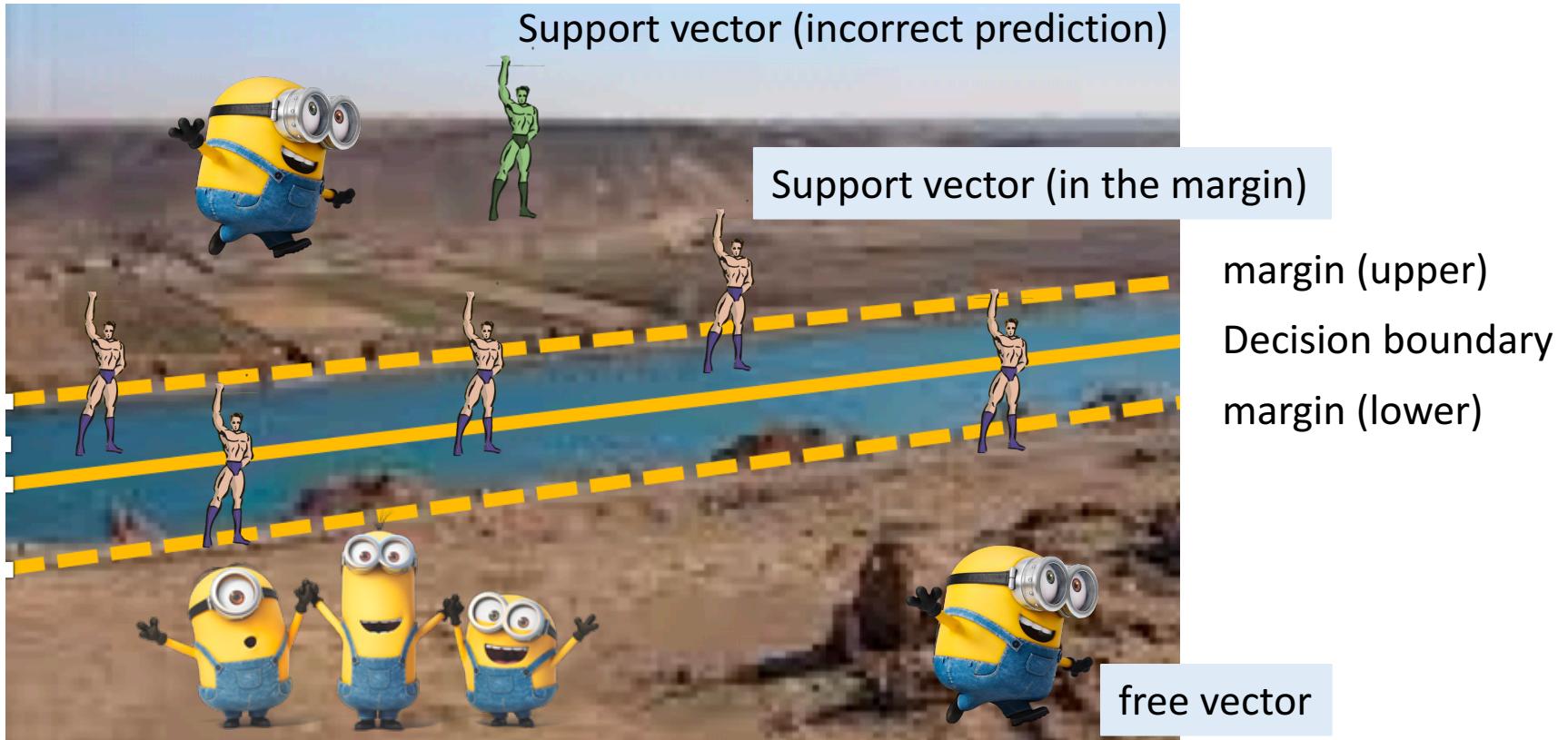


# Why it called support vector machines?



margin (upper)  
Decision boundary  
margin (lower)

# Why it called support vector machines?

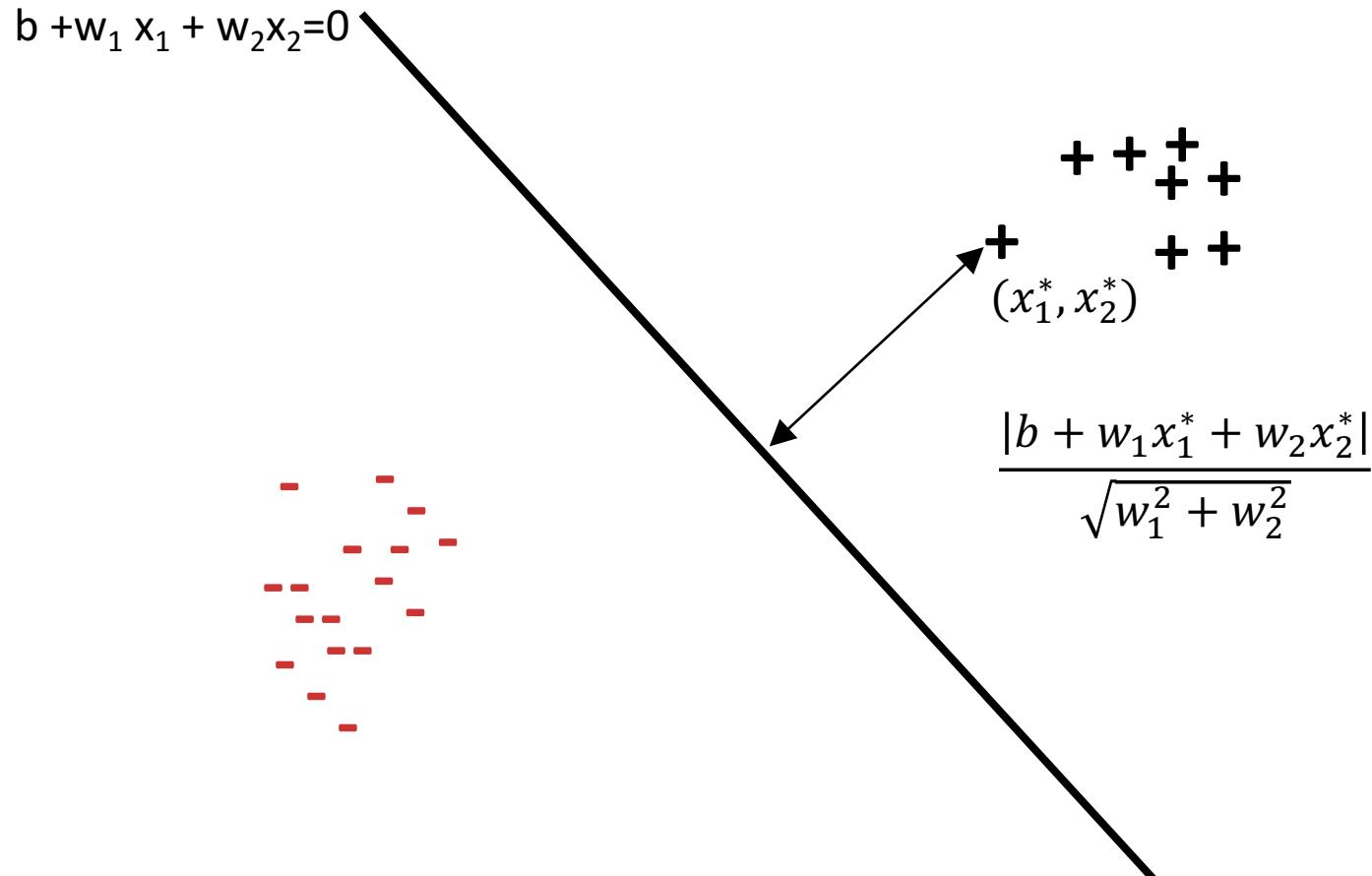


# This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

# Recall: The geometry of a linear classifier

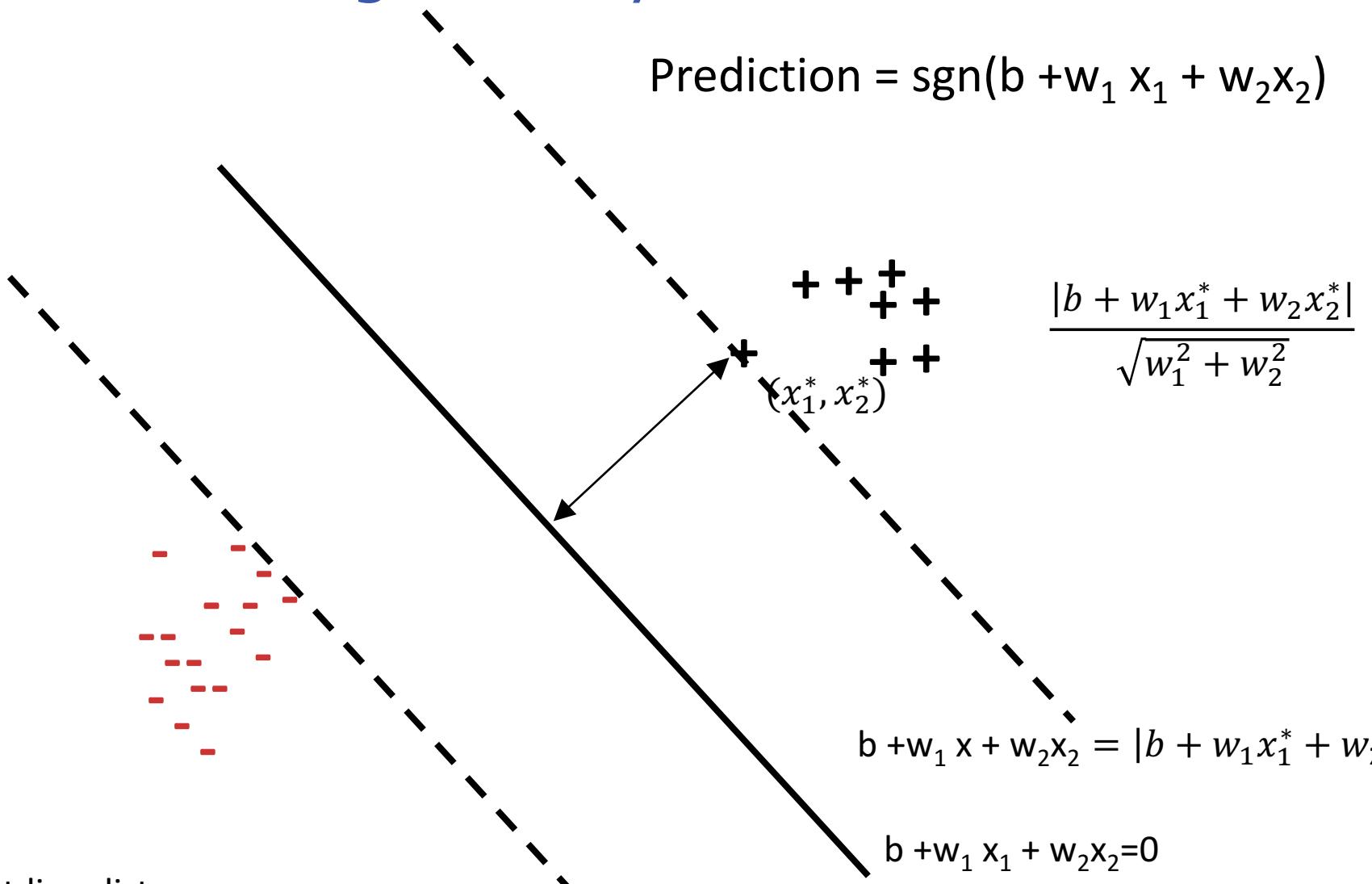
$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html> Lec 11: Support Vector Machines

# Recall: The geometry of a linear classifier



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

# Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

# Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want  $\max_{\mathbf{w}} \gamma$

Some people call this the *geometric margin*

The numerator alone is called the *functional margin*

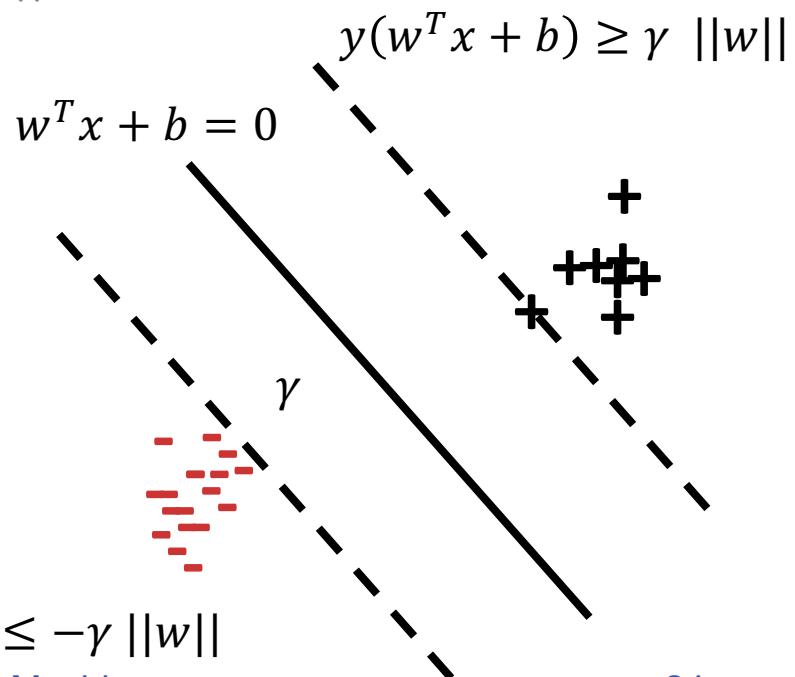
# Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

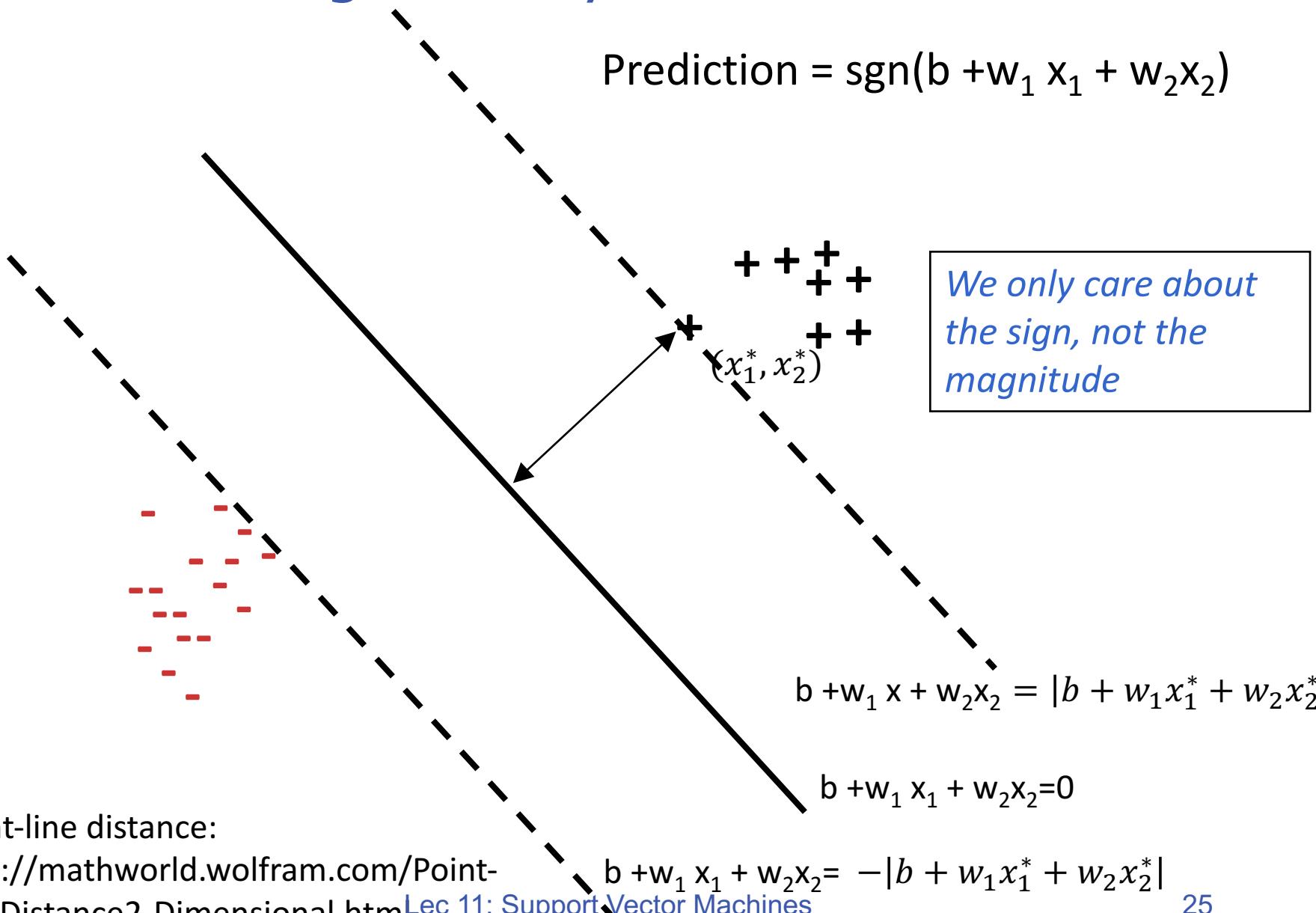
$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

$$s.t. \forall i, y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma \|\mathbf{w}\|$$

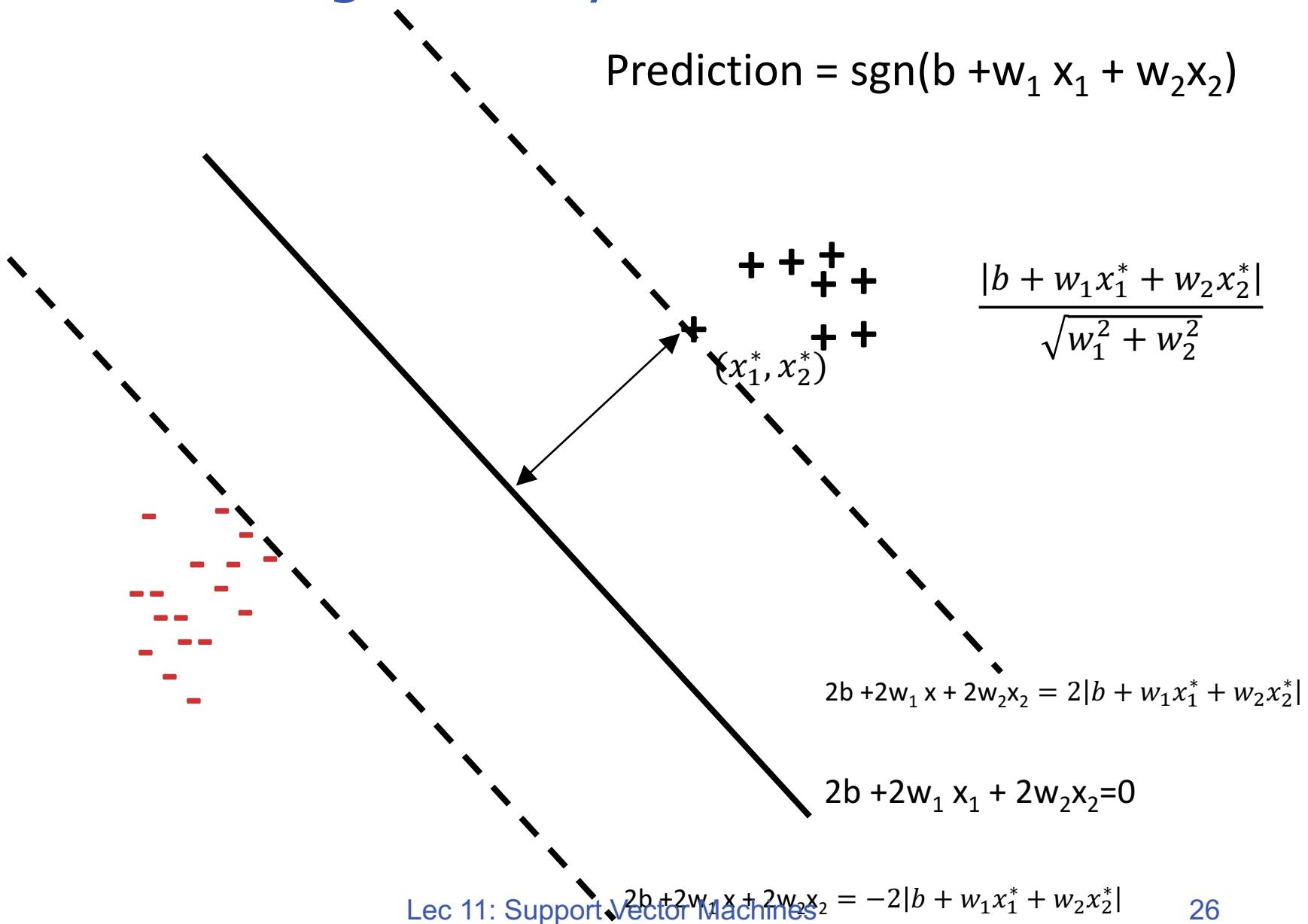
$$\max_{\gamma} \gamma$$



# Recall: The geometry of a linear classifier

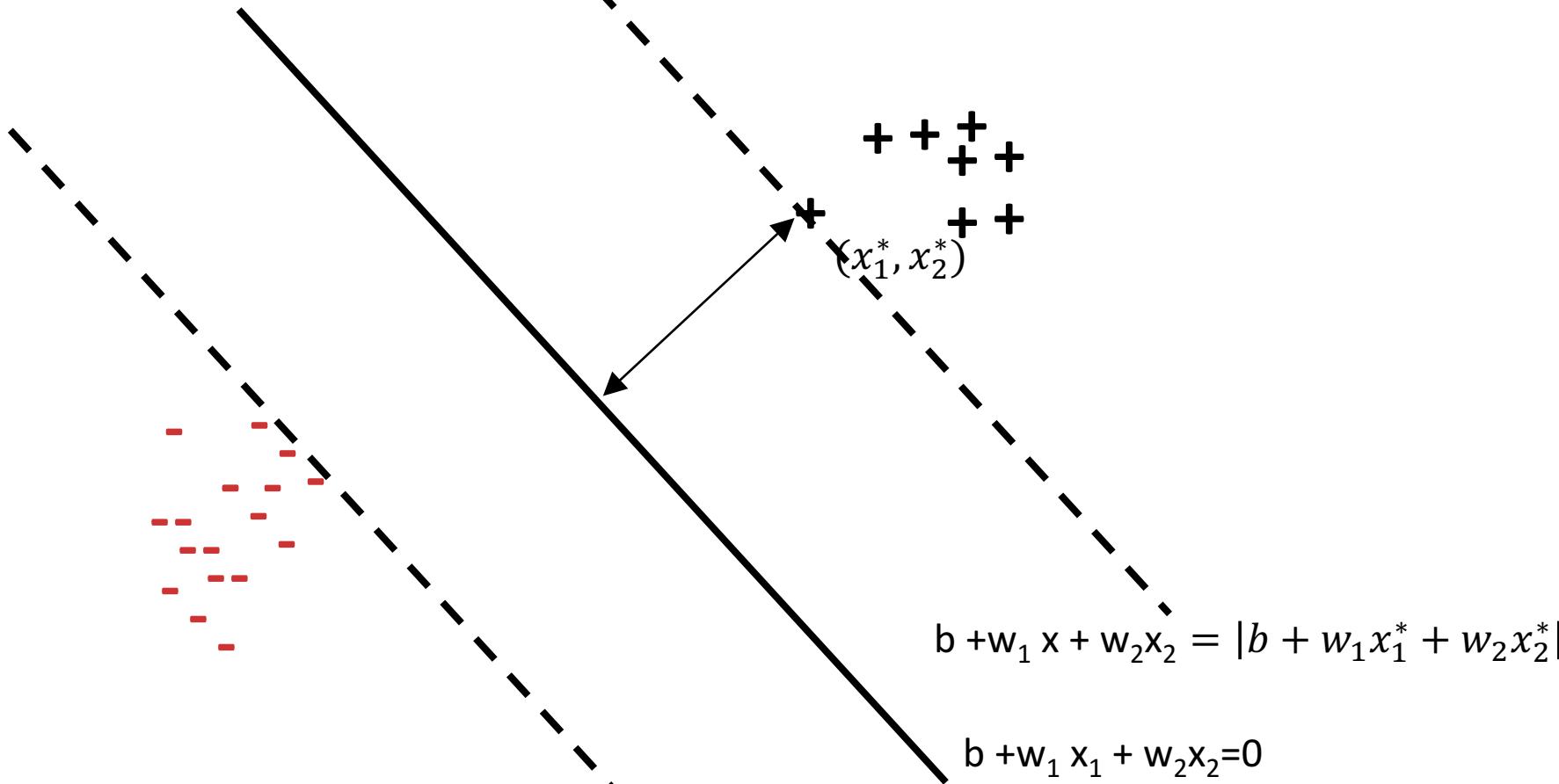


# Recall: The geometry of a linear classifier



# Recall: The geometry of a linear classifier

We have the freedom to scale up/down  $w$  and  $b$  so that we can make  $b + w_1x_1^* + w_2x_2^*=1$ .



Point-line distance:

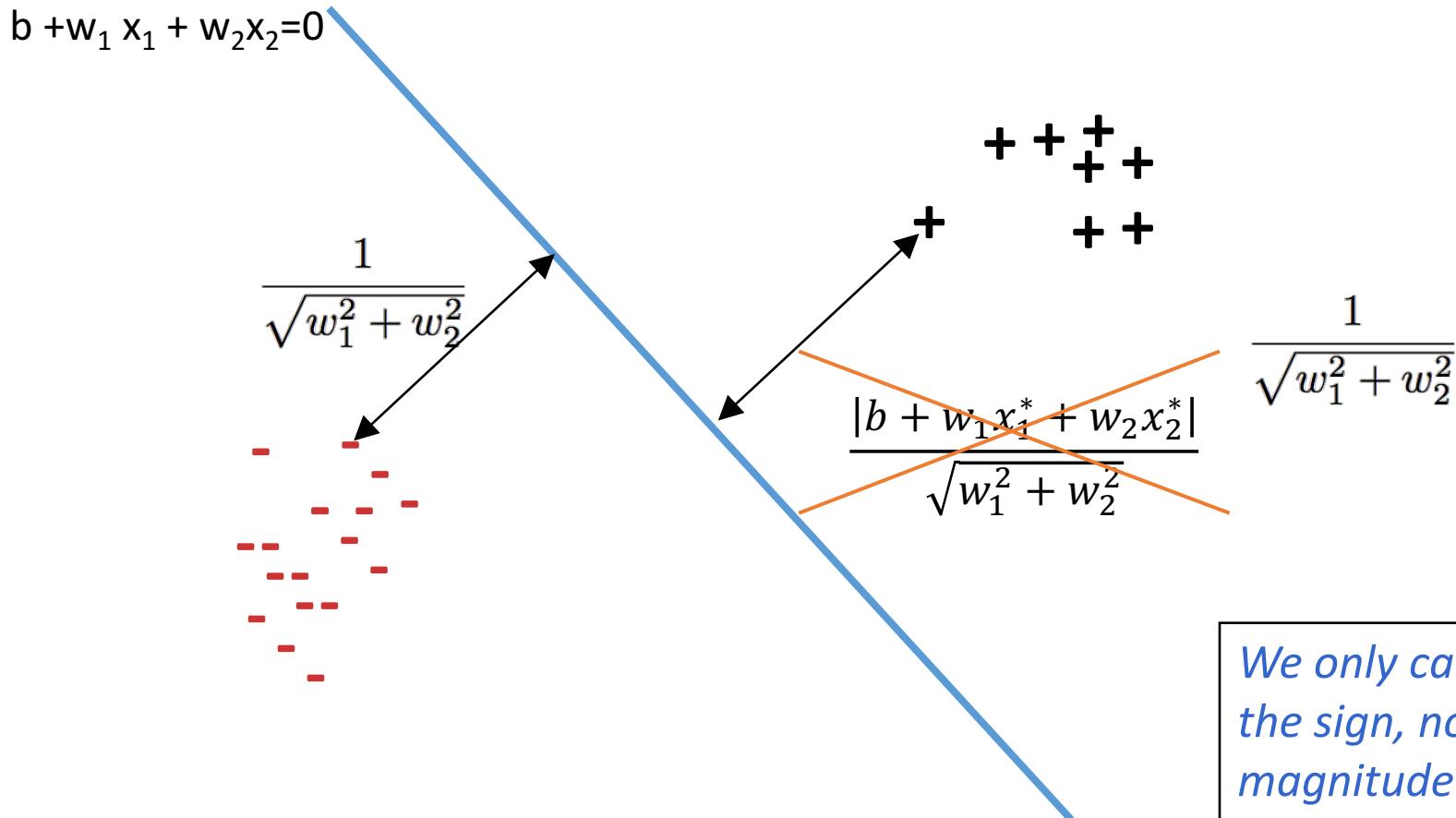
<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

$$b + w_1 x_1 + w_2 x_2 = -|b + w_1 x_1^* + w_2 x_2^*|$$

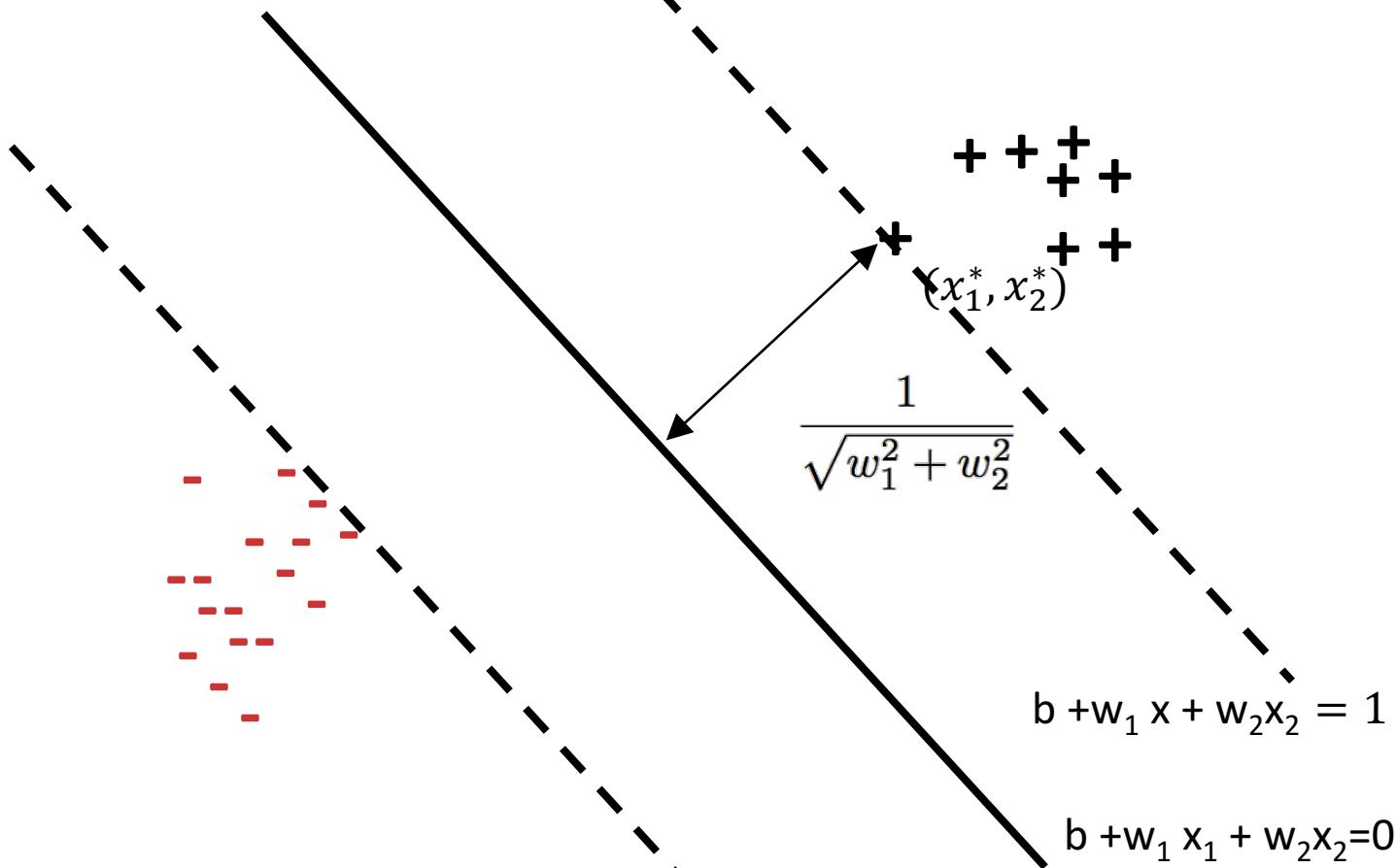
# Recall: The geometry of a linear classifier

$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



# Recall: The geometry of a linear classifier

We have the freedom to scale up/down  $w$  and  $b$  so that we can make  $b + w_1x_1^* + w_2x_2^* = 1$ .



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

# Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want  $\max_{\mathbf{w}} \gamma$
- ❖ We only care about the sign of  $\mathbf{w}$  and  $b$  in the end and not the magnitude
  - ❖ Set the absolute score (functional margin) of the closest point to be 1 and allow  $\mathbf{w}$  to adjust itself

$\max_{\mathbf{w}} \gamma$  is equivalent to  $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$  in this setting

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

# Max-margin classifiers

## ❖ Learning problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

Mimimizing gives us  $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

# Max-margin classifiers

- ❖ Learning problem:

$$\begin{aligned} & \min_{w,b} \frac{1}{2} w^T w \\ s.t. \quad & \forall i, \quad y_i (w^T x_i + b) \geq 1 \end{aligned}$$

Mimimizing gives us  $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

This condition is true for every example, specifically, for the example closest to the separator

- ❖ This is called the “hard” Support Vector Machine

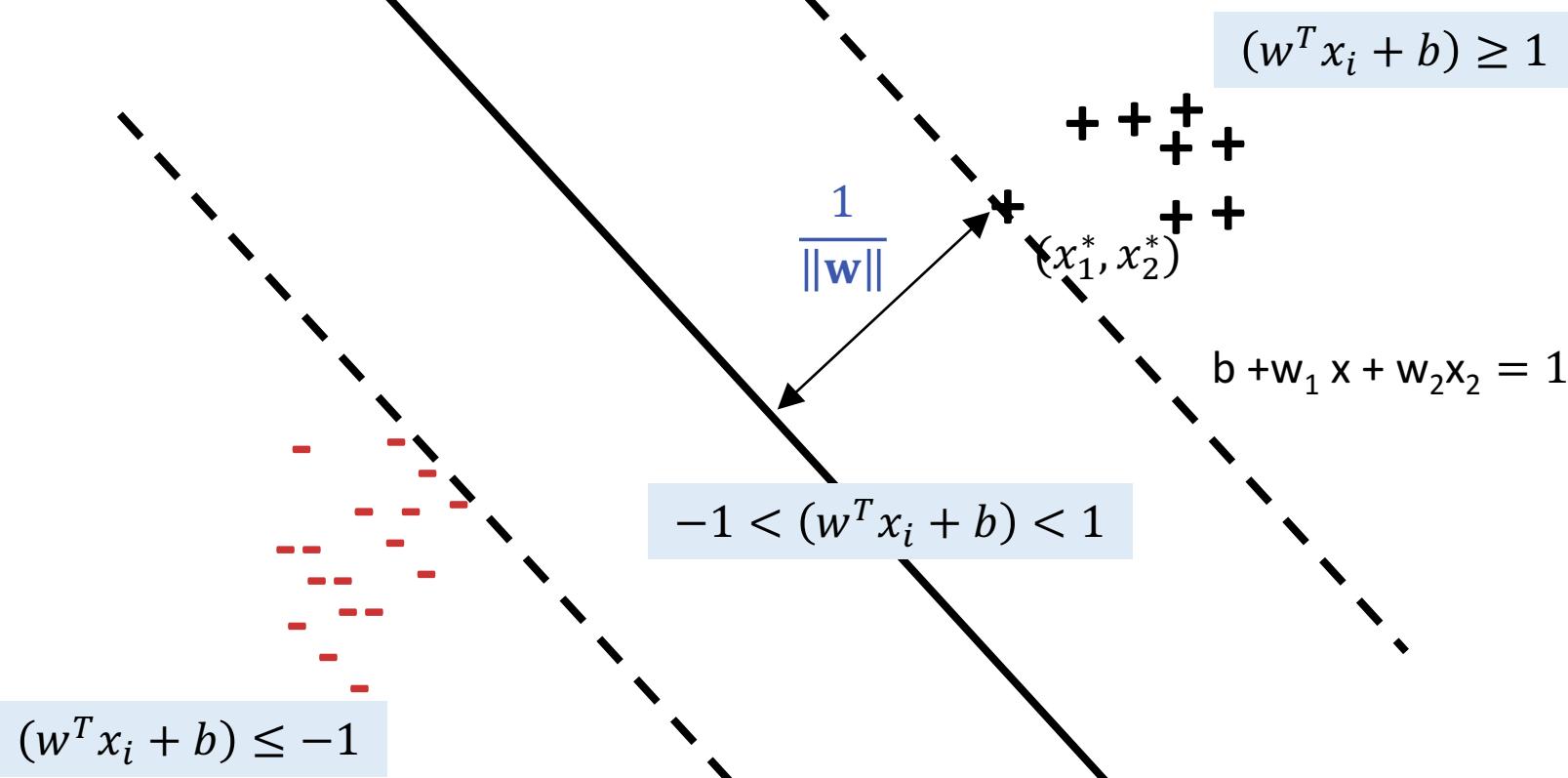
We will look at how to solve this optimization problem later

# Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$b + w_1 x_1 + w_2 x_2 = 0$$



Point-line distance:

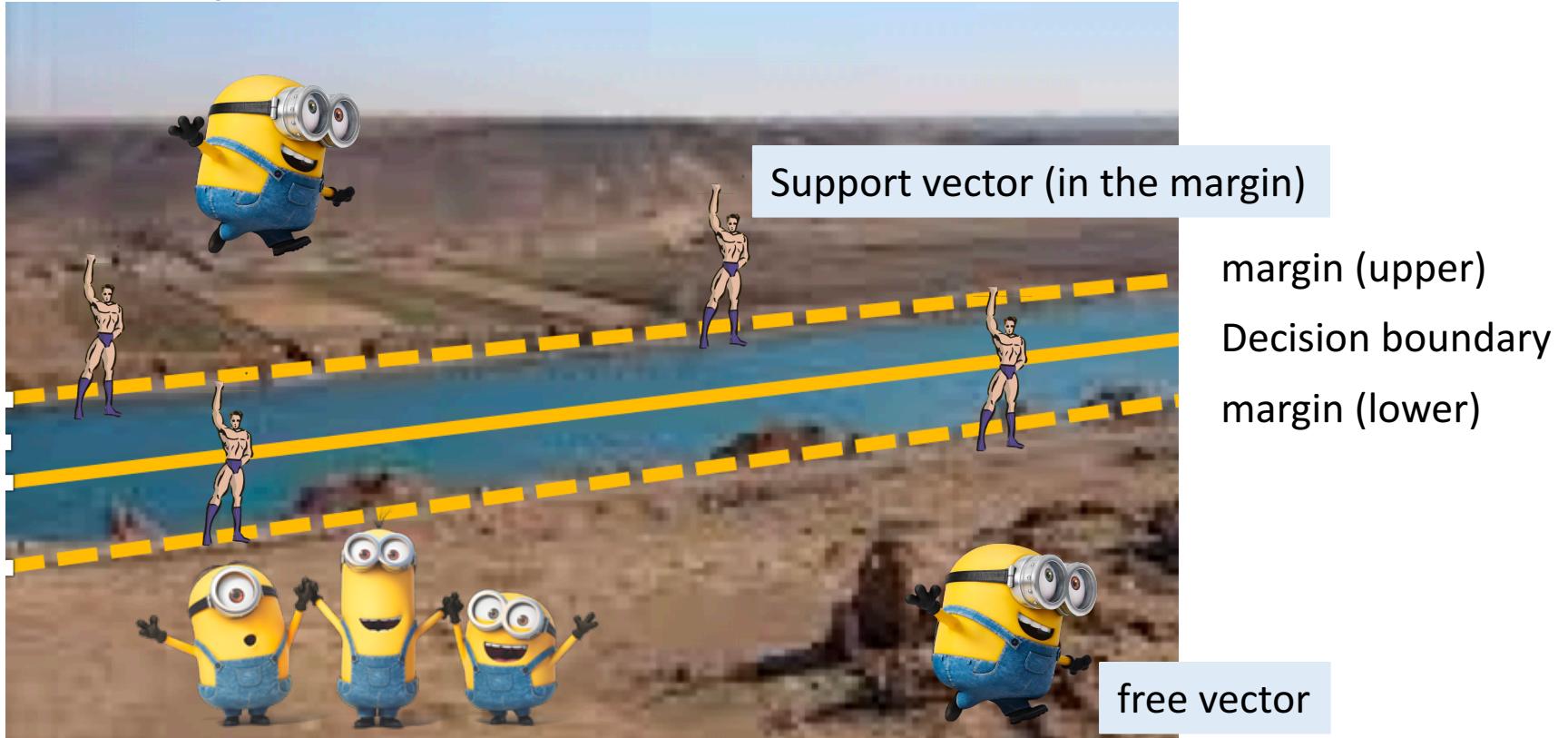
<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

$b + w_1 x_1 + w_2 x_2 = -1$

# Hard support vector machines?

No training error can be made. All support vectors are on the boundary



# What if the data is not separable?

Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an functional margin of at least 1

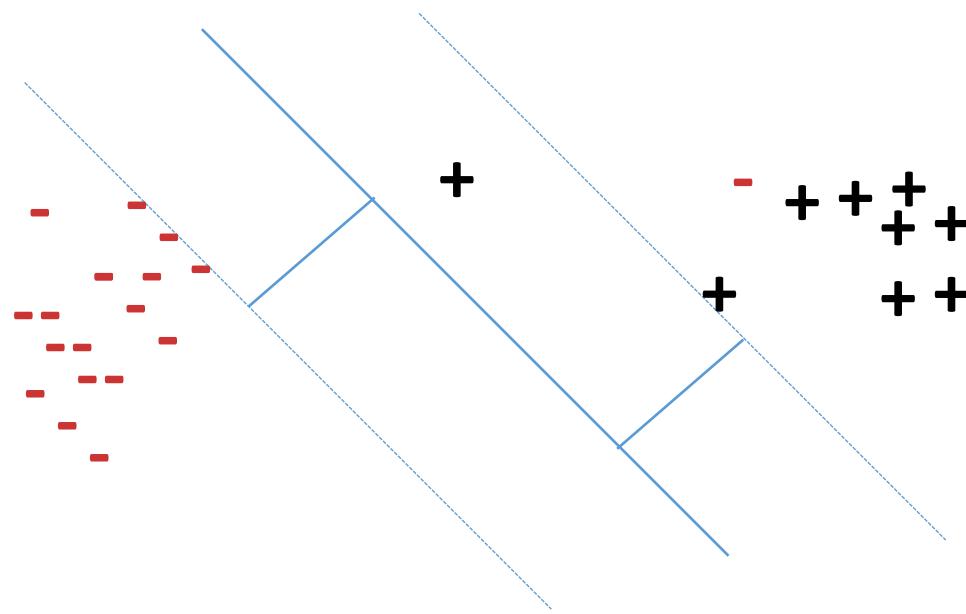
- ❖ This is a constrained optimization problem
- ❖ If the data is not separable, there is no  $w$  that will classify the data
- ❖ Infeasible problem, no solution!



If you made an mistake in your midterm, got 0 point!

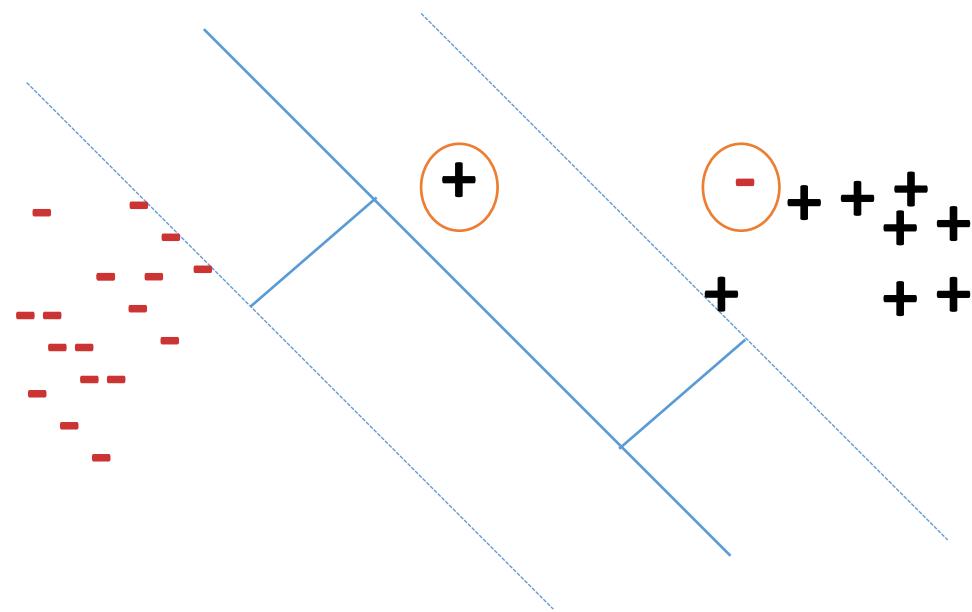
# Dealing with non-separable data

**Key idea:** Allow some examples to “break into the margin” or “make mistake”



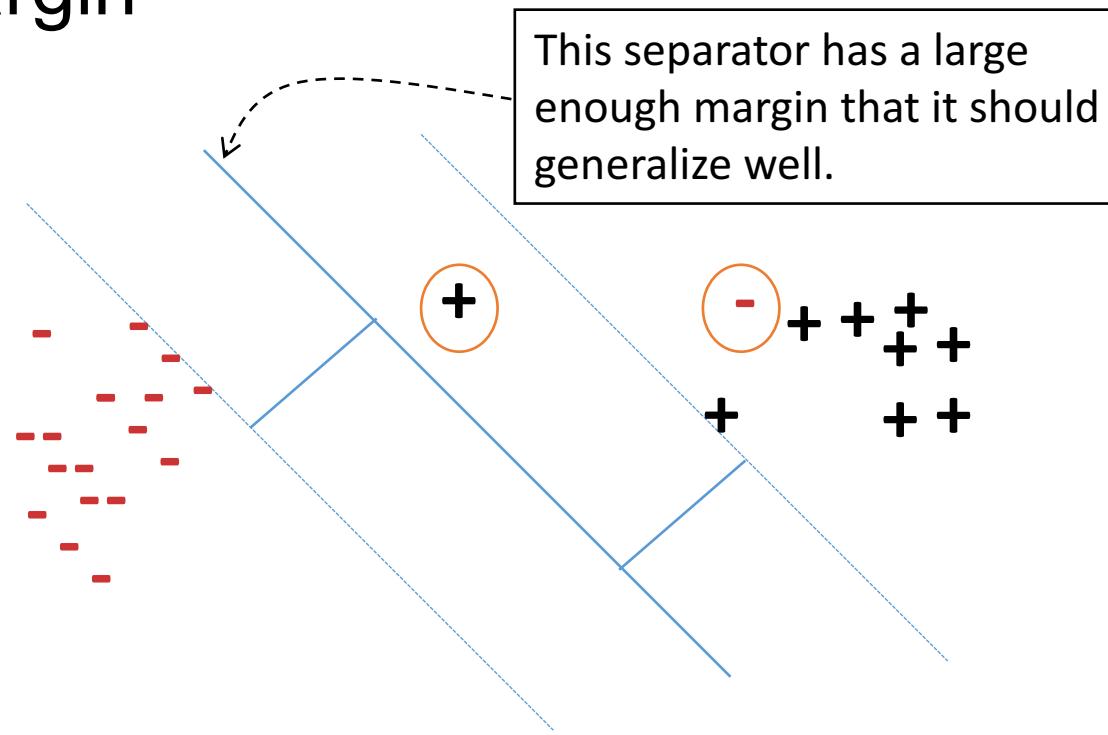
# Dealing with non-separable data

**Key idea:** Allow some examples to “break into the margin”



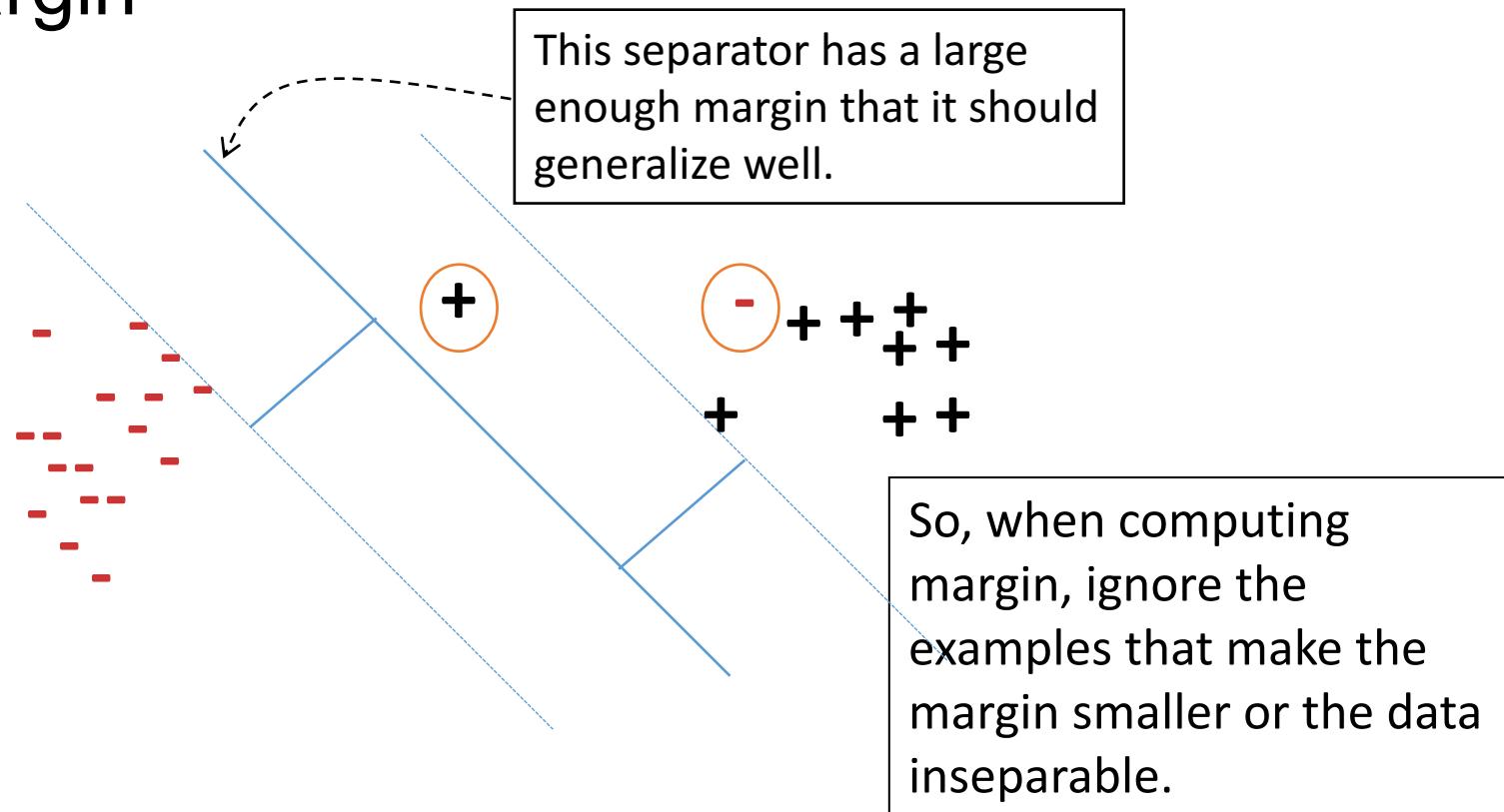
# Dealing with non-separable data

**Key idea:** Allow some examples to “break into the margin”



# Dealing with non-separable data

**Key idea:** Allow some examples to “break into the margin”



# Soft SVM

## ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an  
functional margin of at least 1

# Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable*  $\xi_i$  per example

- ❖ And require  $y_i(w^T x_i + b) \geq 1 - \xi_i$  and  $\xi_i \geq 0$

# Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable*  $\xi_i$  per example

- ❖ And require  $y_i(w^T x_i + b) \geq 1 - \xi_i$  and  $\xi_i \geq 0$

**Intuition:** The slack variable allows examples to “break” into the margin

If the slack value is zero, then the example is either on or outside the margin

# Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ New optimization problem for learning

$$\min_{w,b, \xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

C is the hyper-parameter

# Soft SVM

$$\min_{w, b, \xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$\begin{aligned} s.t. \quad & \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

# Soft SVM

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

Maximize margin  
Tradeoff between the two terms  
Minimize total slack (i.e allow as few examples as possible to violate the margin)

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

Equivalently, we can eliminate the slack variables to rewrite this:

# Soft SVM

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

Maximize margin  
Tradeoff between the two terms  
Minimize total slack (i.e allow as few examples as possible to violate the margin)

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

Equivalently, we can eliminate the slack variables to rewrite this:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

# Maximizing margin and minimizing loss

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Maximize margin

Penalty for the prediction

# Maximizing margin and minimizing loss

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

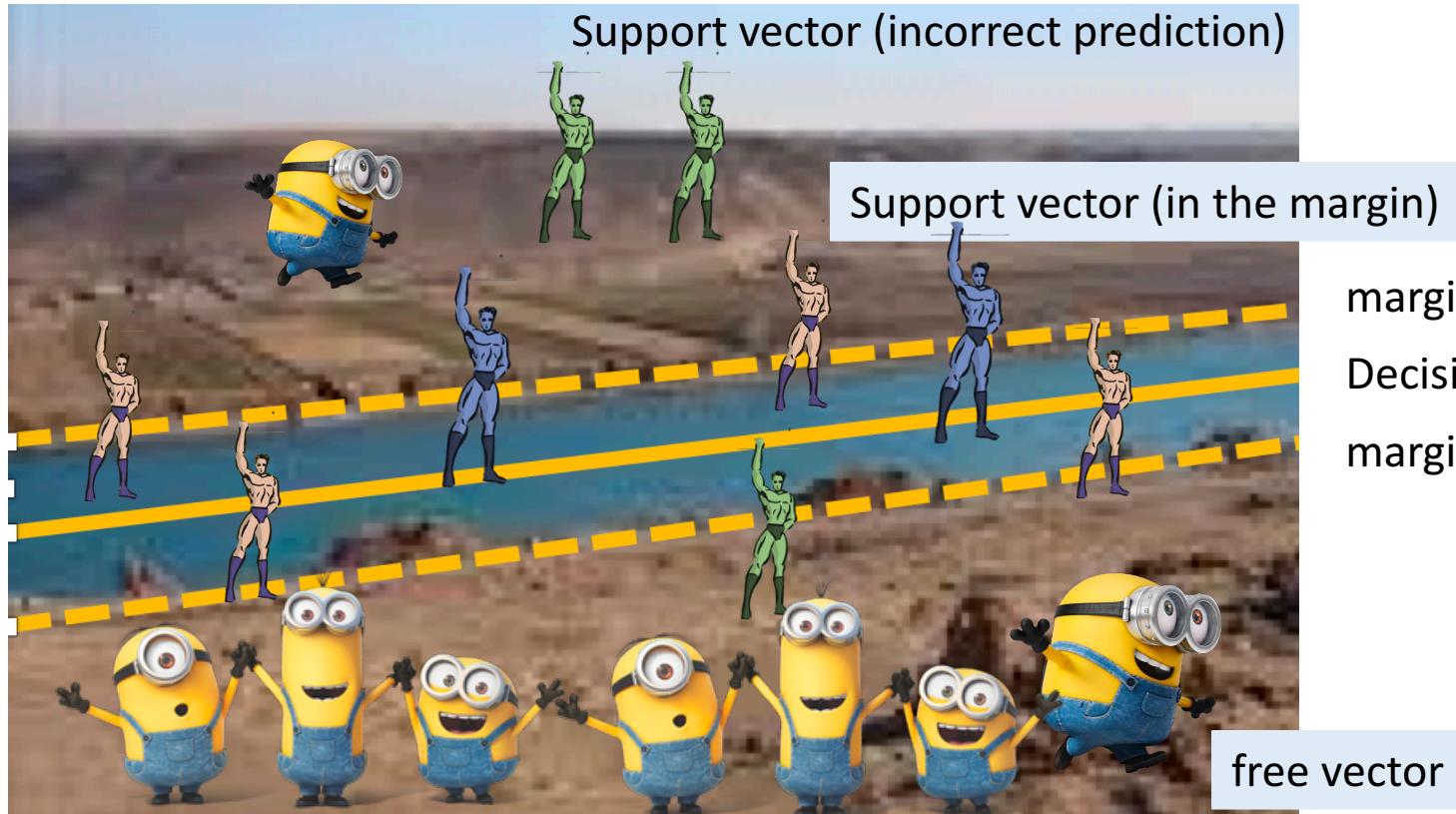
Maximize margin                      Penalty for the prediction

We can consider three cases

- ❖ Example is **correctly** classified and is outside the margin:  
penalty = 0
- ❖ Example is **incorrectly** classified:  
penalty =  $1 - y_i(w^T x_i + b)$
- ❖ Example is **correctly** classified but **within the margin**:  
penalty =  $1 - y_i(w^T x_i + b)$

This is the **hinge loss** function

# Why it called support vector machines?



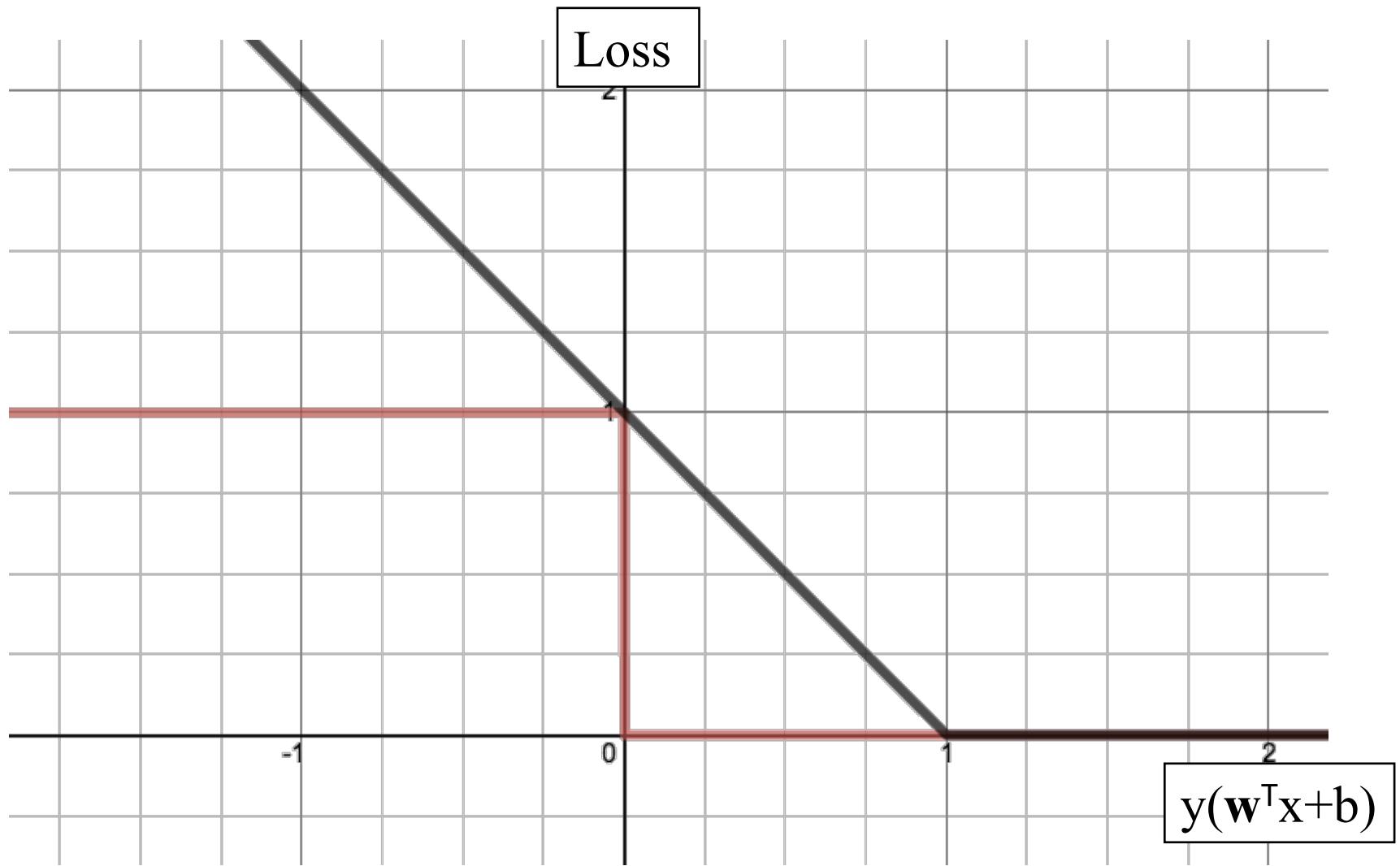
margin (upper)

Decision boundary

margin (lower)

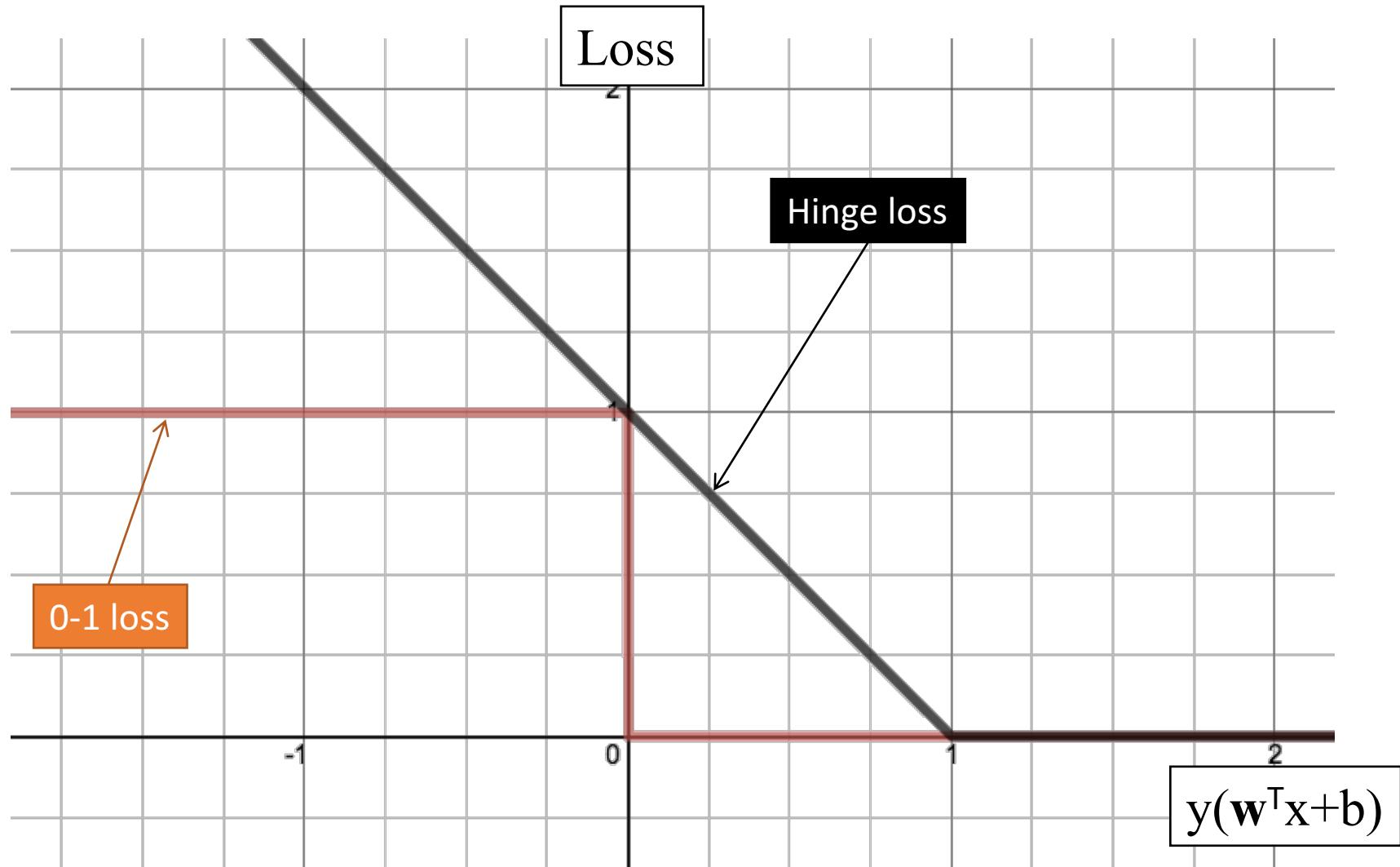
free vector

# The Hinge Loss



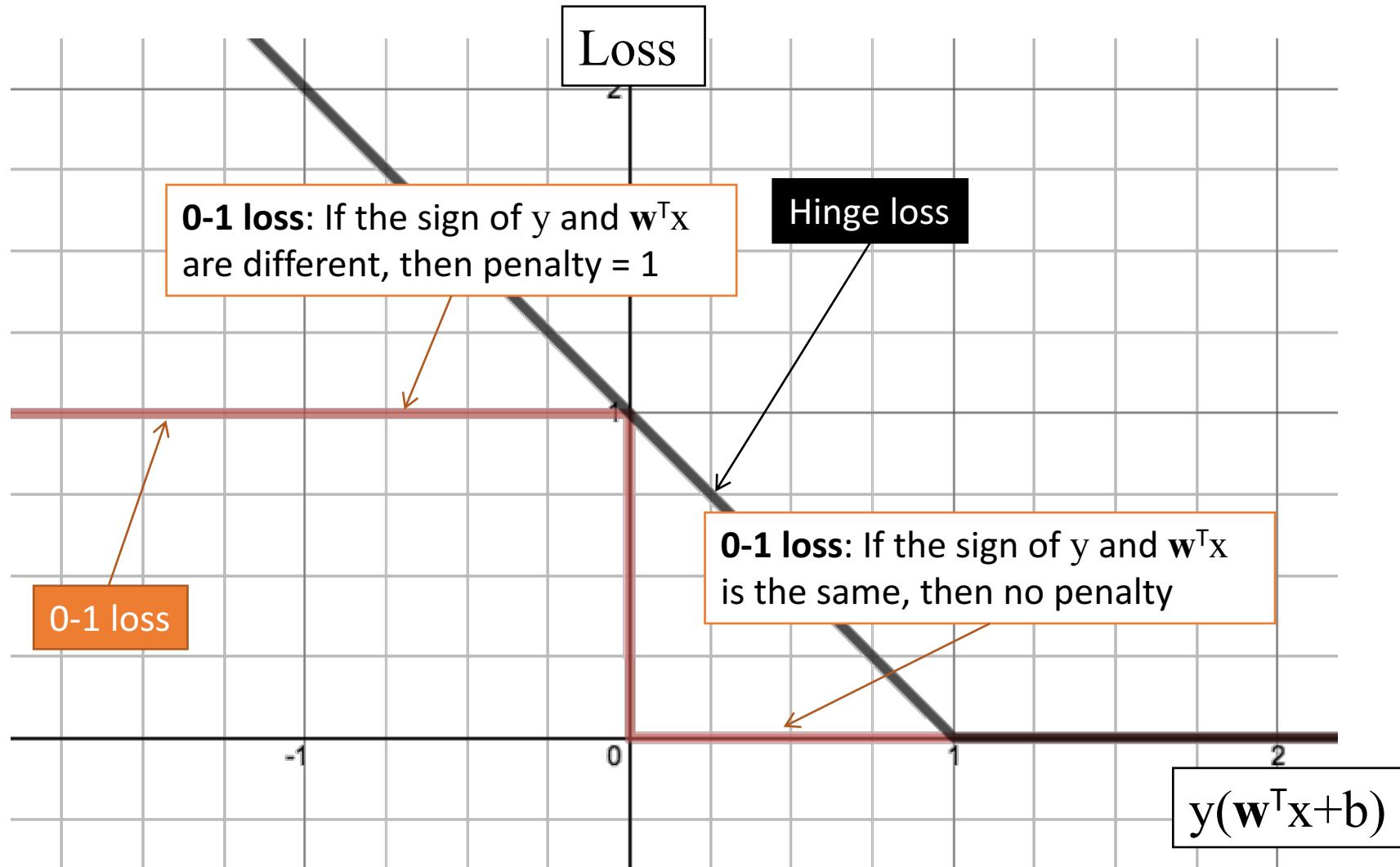
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

# The Hinge Loss



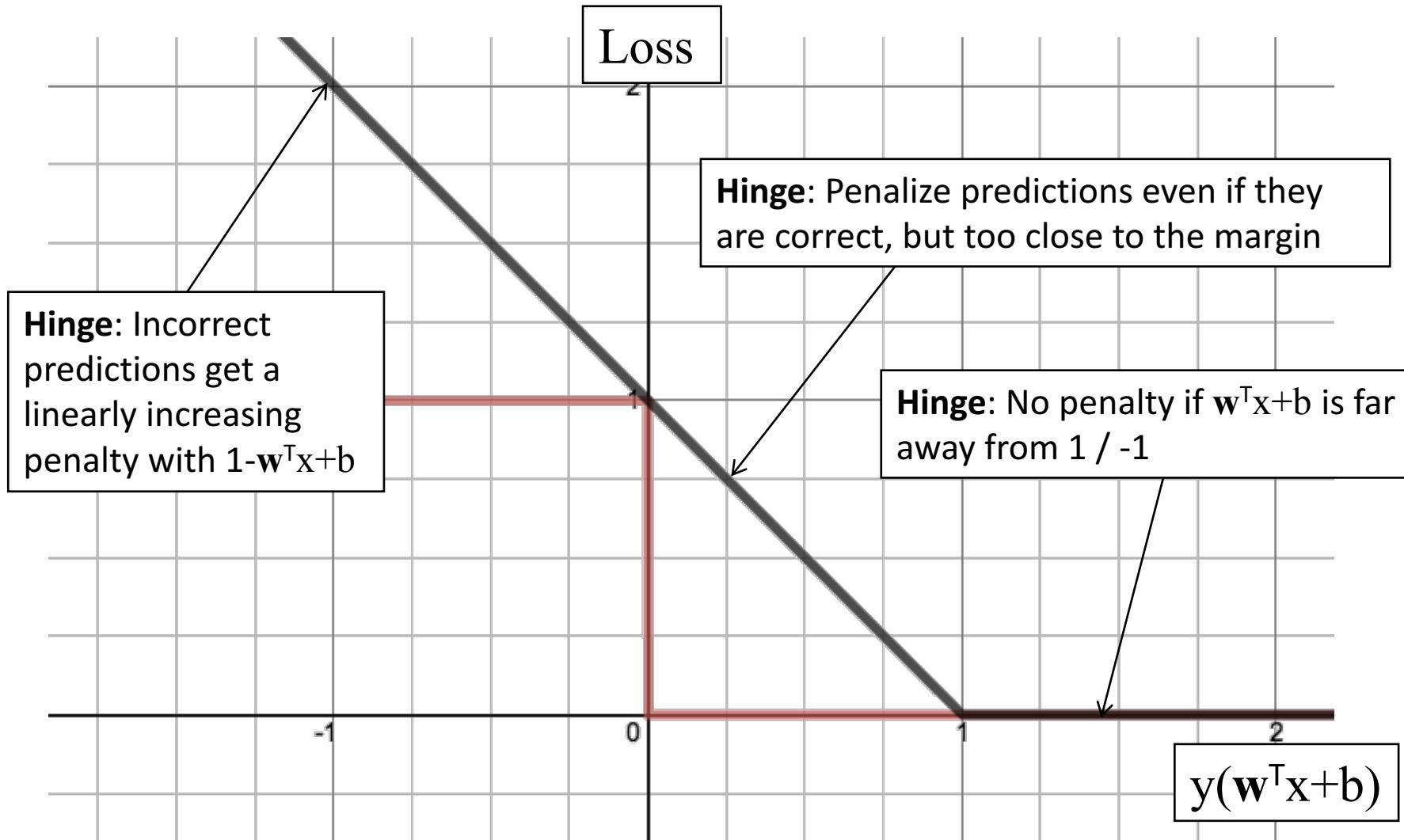
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

# The Hinge Loss



$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

# The Hinge Loss



# General learning principle

## Risk minimization

Define the notion of “loss” over the training data as a function of a hypothesis

Learning = find the hypothesis that has lowest loss on the training data

# General learning principle

## Regularized risk minimization

Define a regularization function  
that penalizes over-complex  
hypothesis.

Capacity control gives better  
generalization

Define the notion of “loss”  
over the training data as a  
function of a hypothesis

Learning =  
find the hypothesis that has lowest  
[Regularizer + loss on the training data]

# SVM objective function

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

## Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

## Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

# SVM objective function

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

## Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

## Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A **hyper-parameter** that controls the tradeoff between a large margin and a small hinge-loss

# This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

# Outline: Training SVM by optimization

1. Check convexity
2. Stochastic gradient descent
3. Sub-derivatives of the hinge loss
4. Stochastic sub-gradient descent for SVM
5. Comparison to perceptron

# Solving the SVM optimization problem

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

This function is **convex** in  $w$

# Solving the SVM optimization problem

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

This function is convex in  $w$

- ❖ This is a quadratic optimization problem because the objective is quadratic
- ❖ Older methods: Used techniques from Quadratic Programming
  - ❖ Very slow
- ❖ No constraints, can use *gradient descent*
  - ❖ Still very slow!

# Outline: Training SVM by optimization

1. Check convexity
2. Stochastic gradient descent
3. Sub-derivatives of the hinge loss
4. Stochastic sub-gradient descent for SVM
5. Comparison to perceptron

# Stochastic gradient Descent

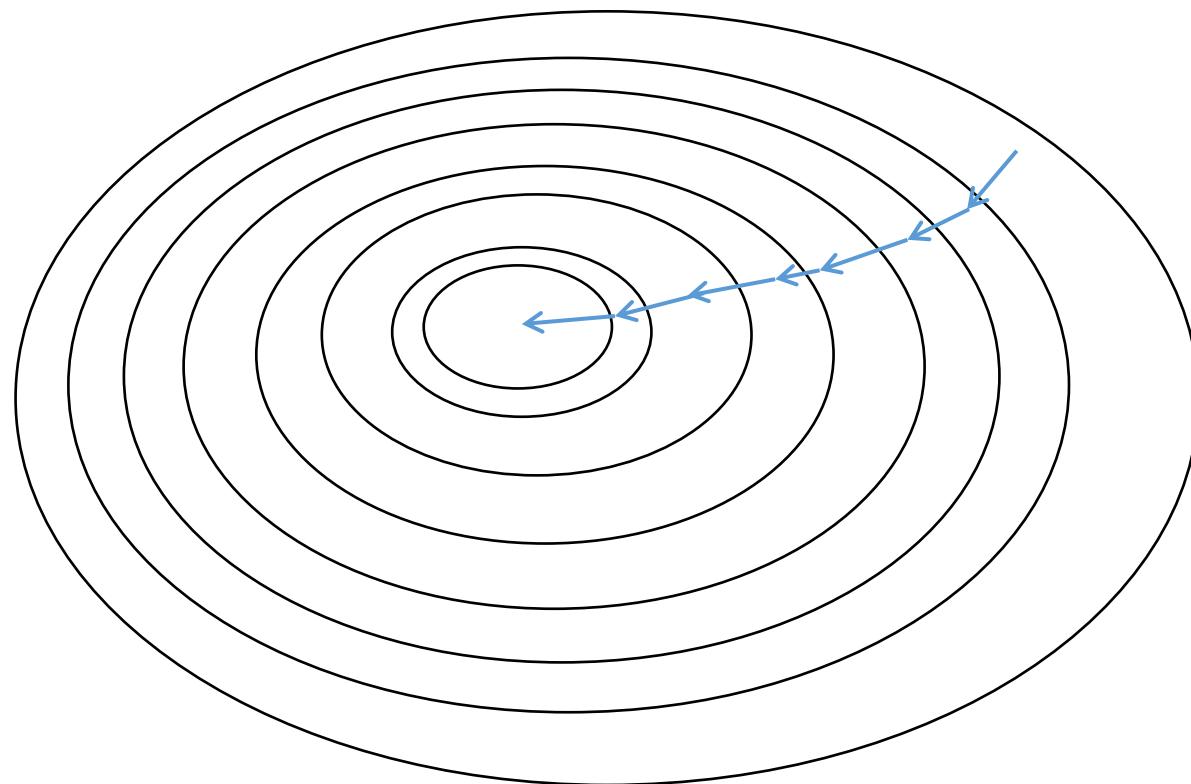
Given a training set  $\mathcal{D} = \{(x, y)\}$

1. Initialize  $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch 1 ...  $T$ :
3.     For  $(x, y)$  in  $\mathcal{D}$ :
4.         Update  $w \leftarrow w - \eta \nabla_w f(x, y)$
5. Return  $w$

$$\min \sum_{(x,y) \in D} f(x, y)$$

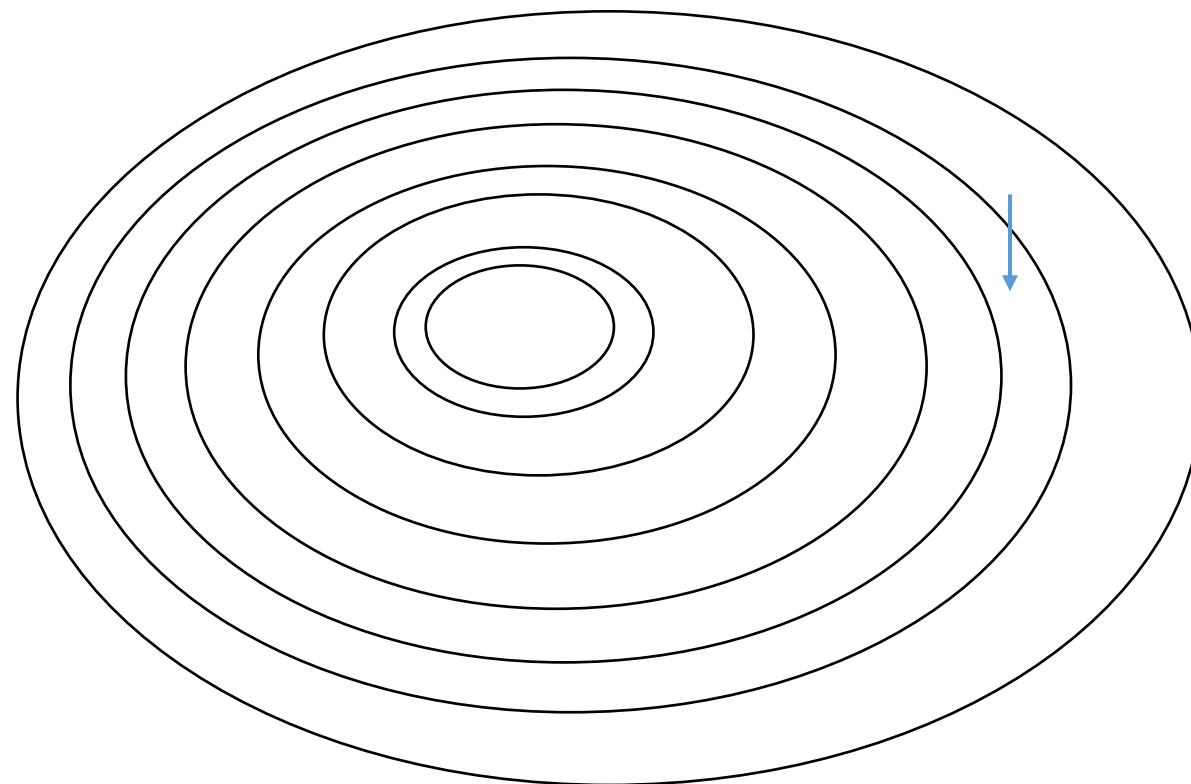
We will see more example later in this lecture

# Gradient Descent vs SGD



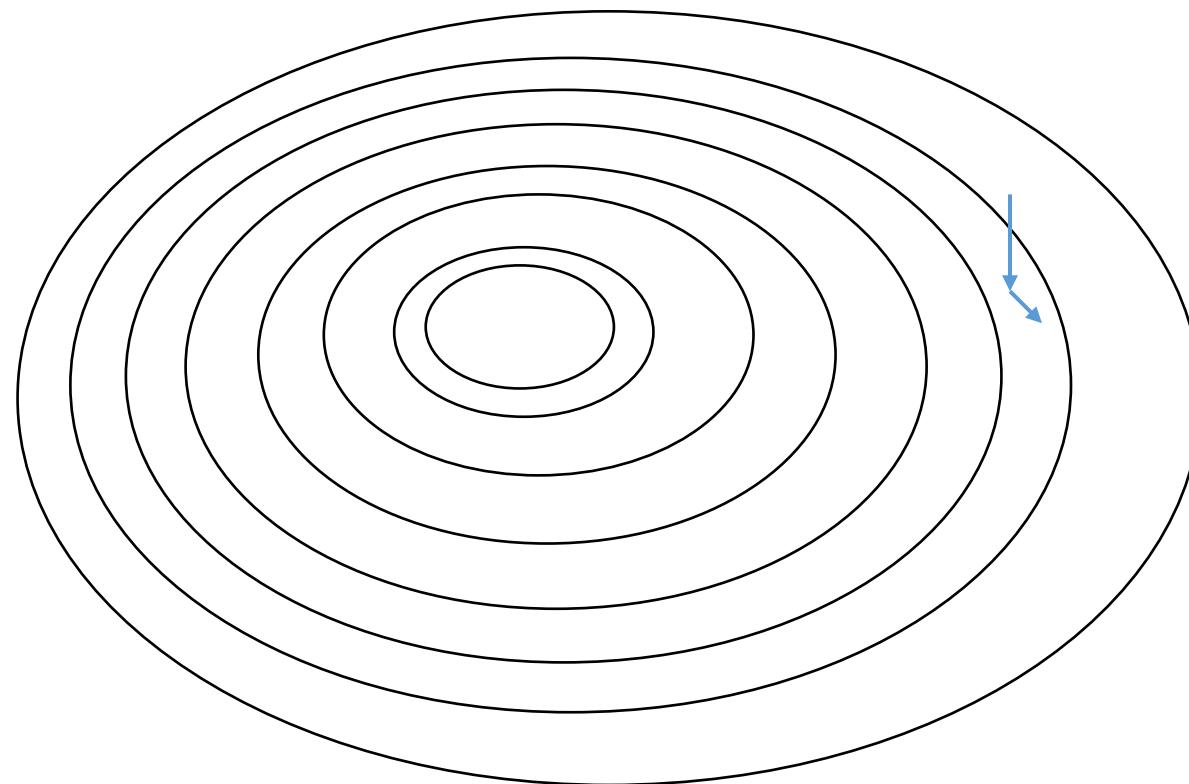
Gradient descent

# Gradient Descent vs SGD



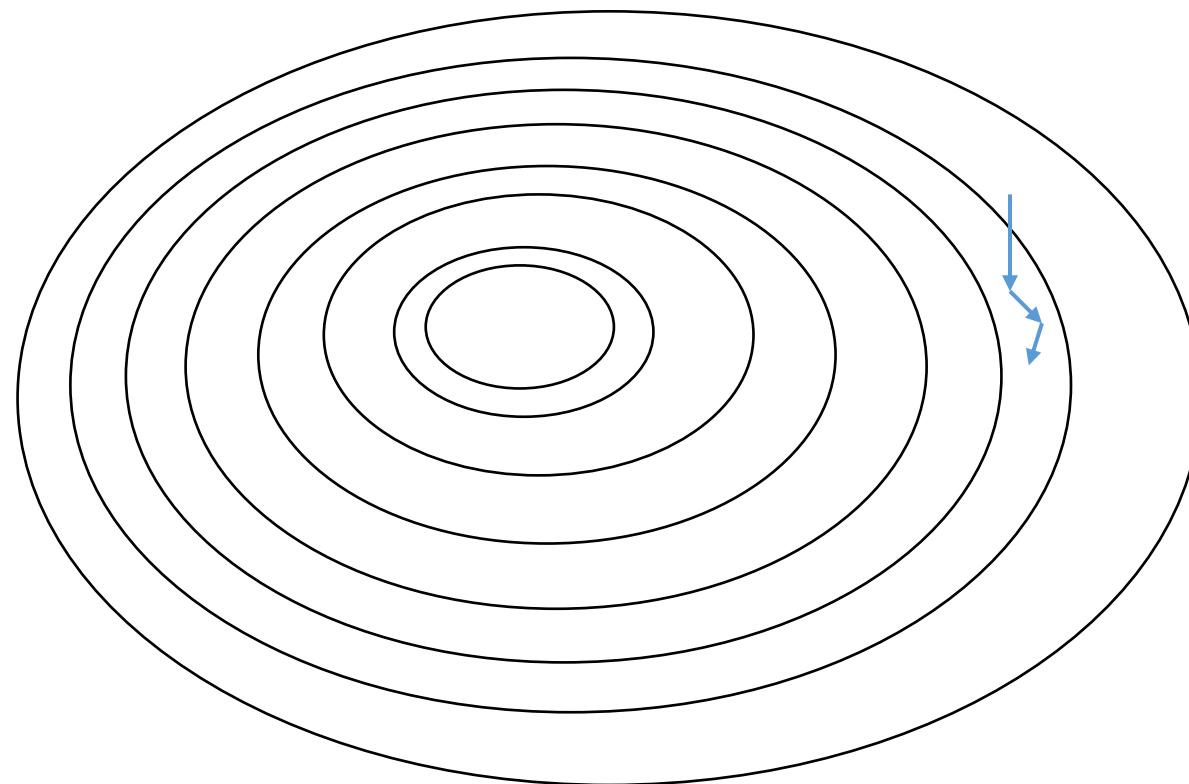
Stochastic Gradient descent

# Gradient Descent vs SGD



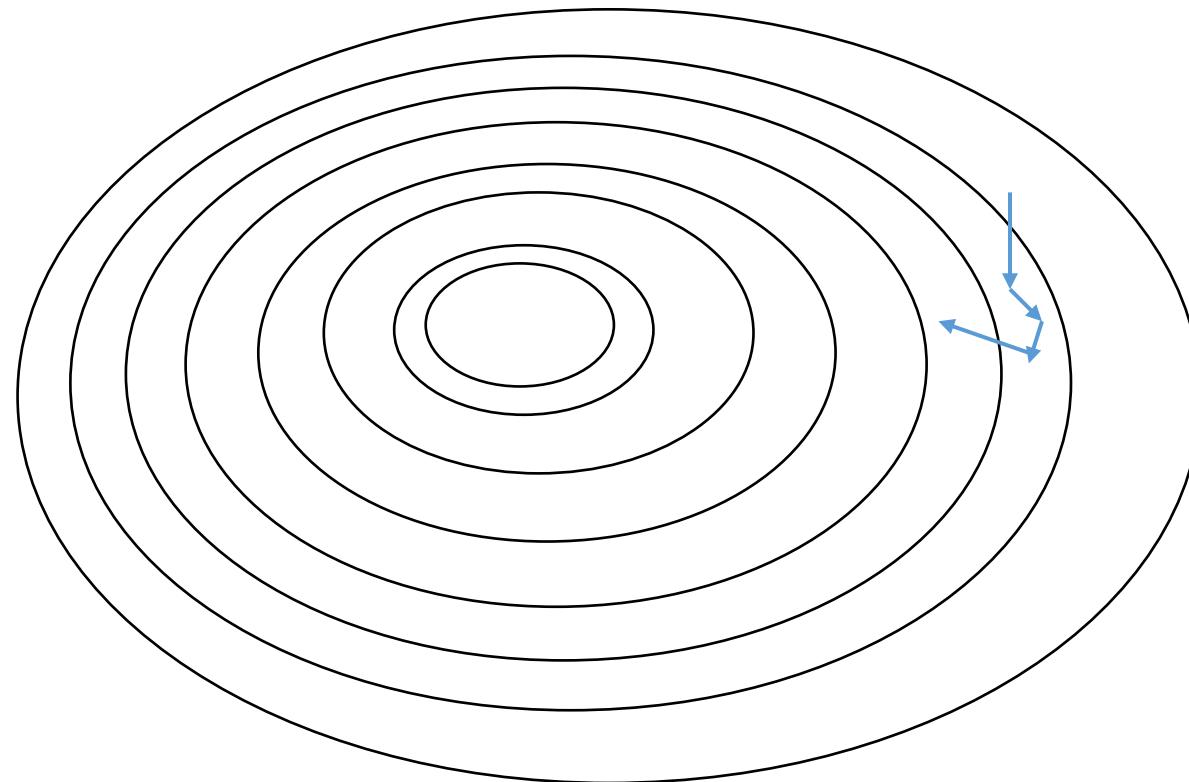
Stochastic Gradient descent

# Gradient Descent vs SGD



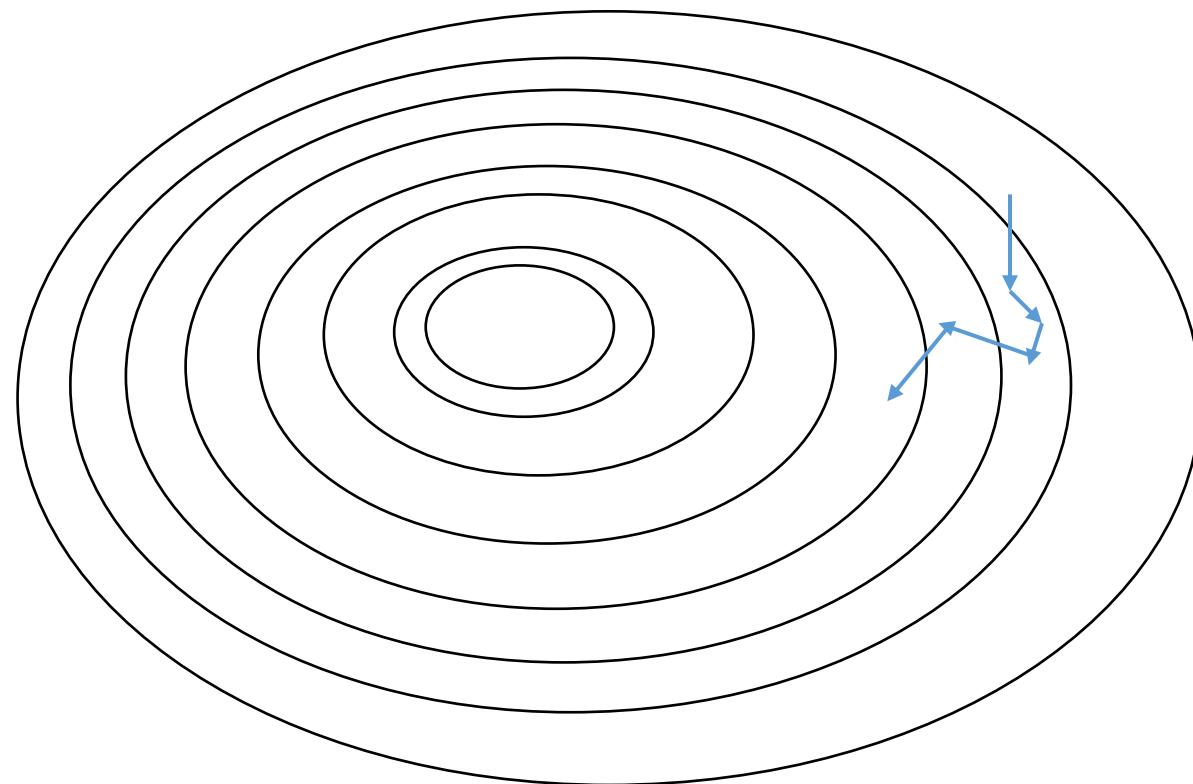
Stochastic Gradient descent

# Gradient Descent vs SGD



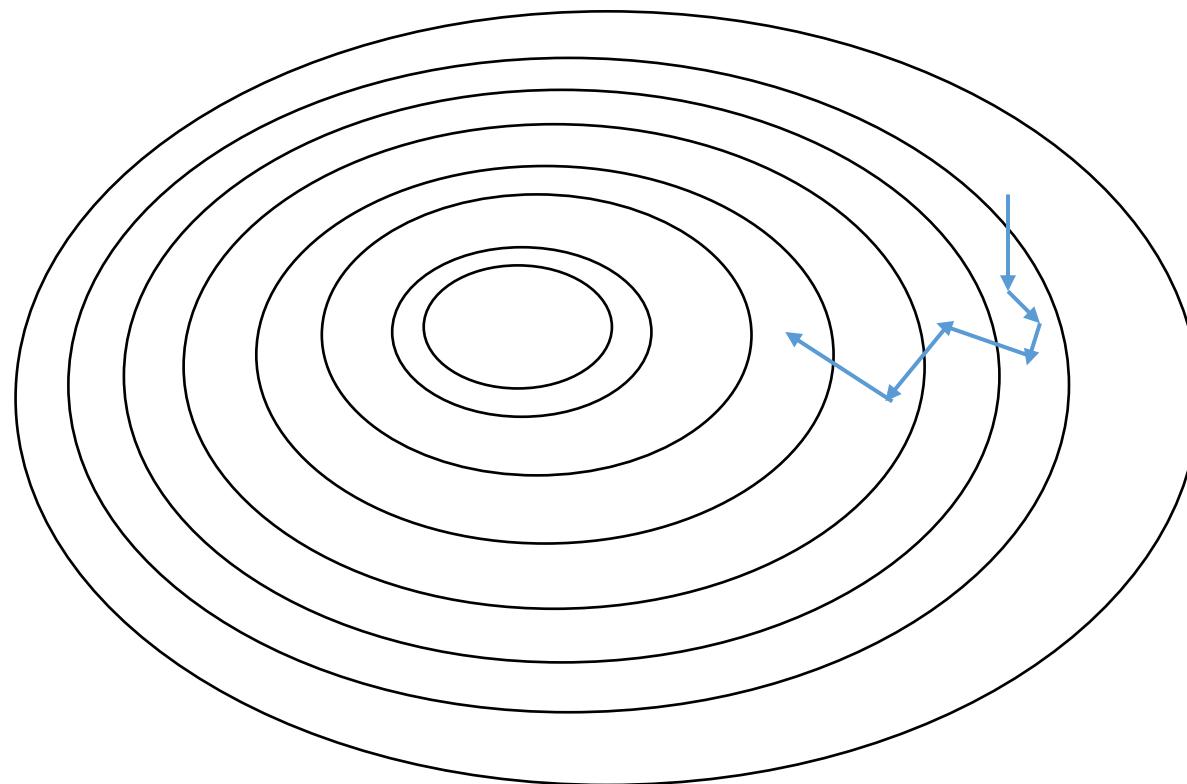
Stochastic Gradient descent

# Gradient Descent vs SGD



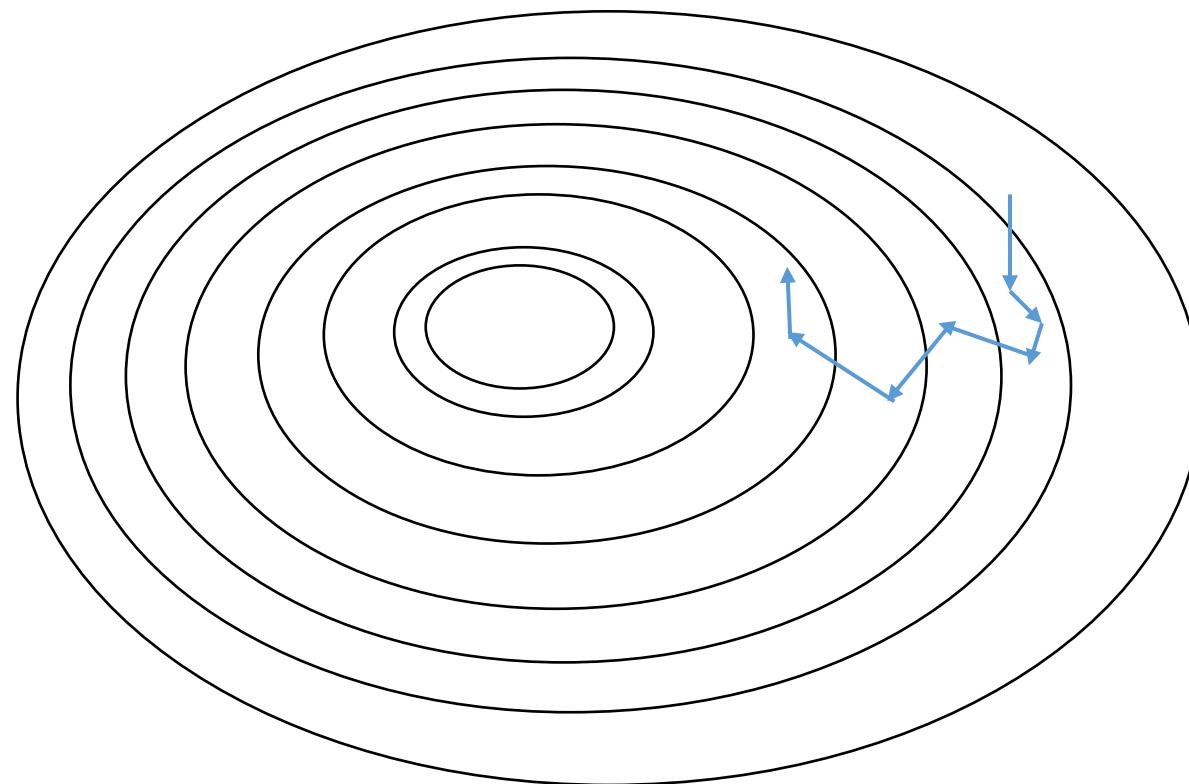
Stochastic Gradient descent

# Gradient Descent vs SGD



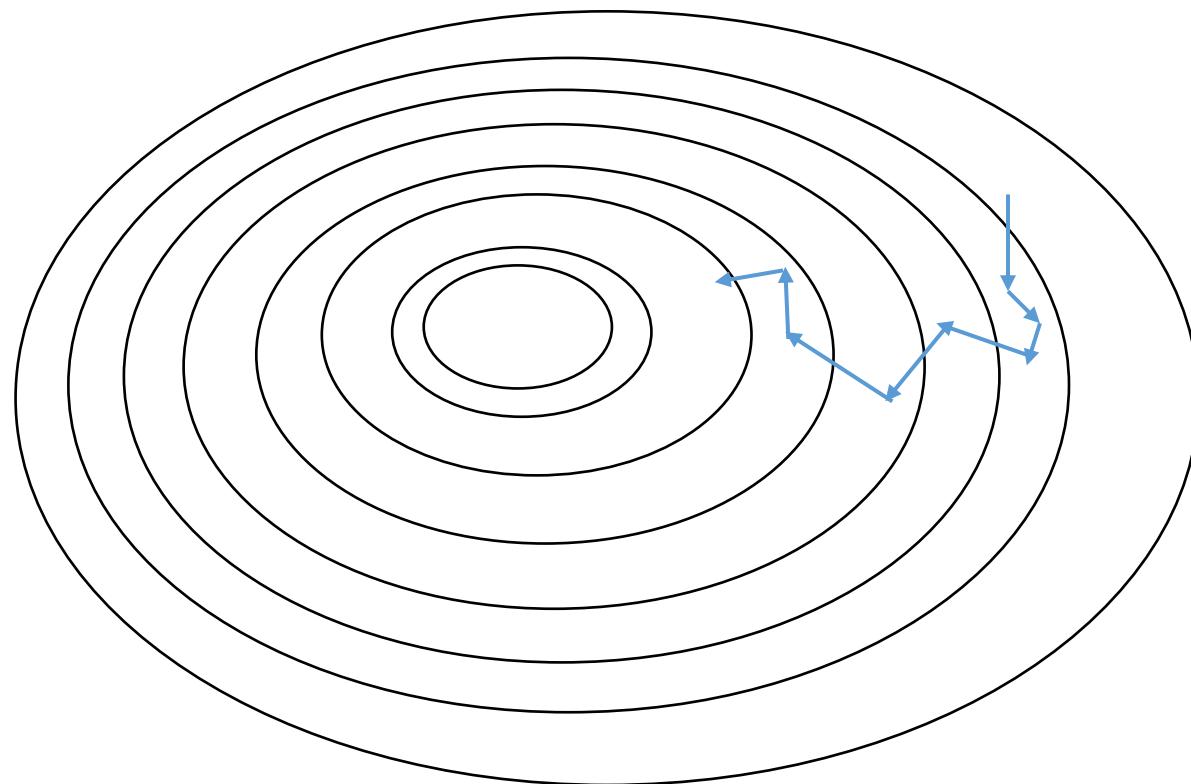
Stochastic Gradient descent

# Gradient Descent vs SGD



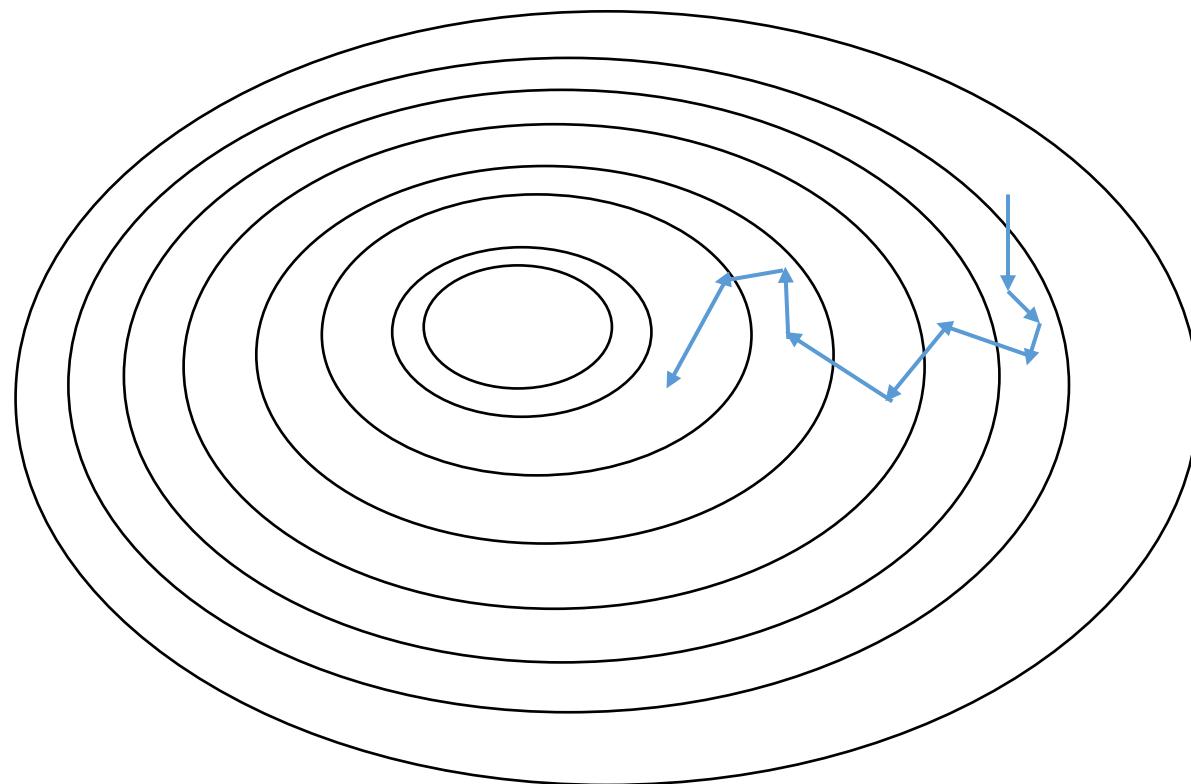
Stochastic Gradient descent

# Gradient Descent vs SGD



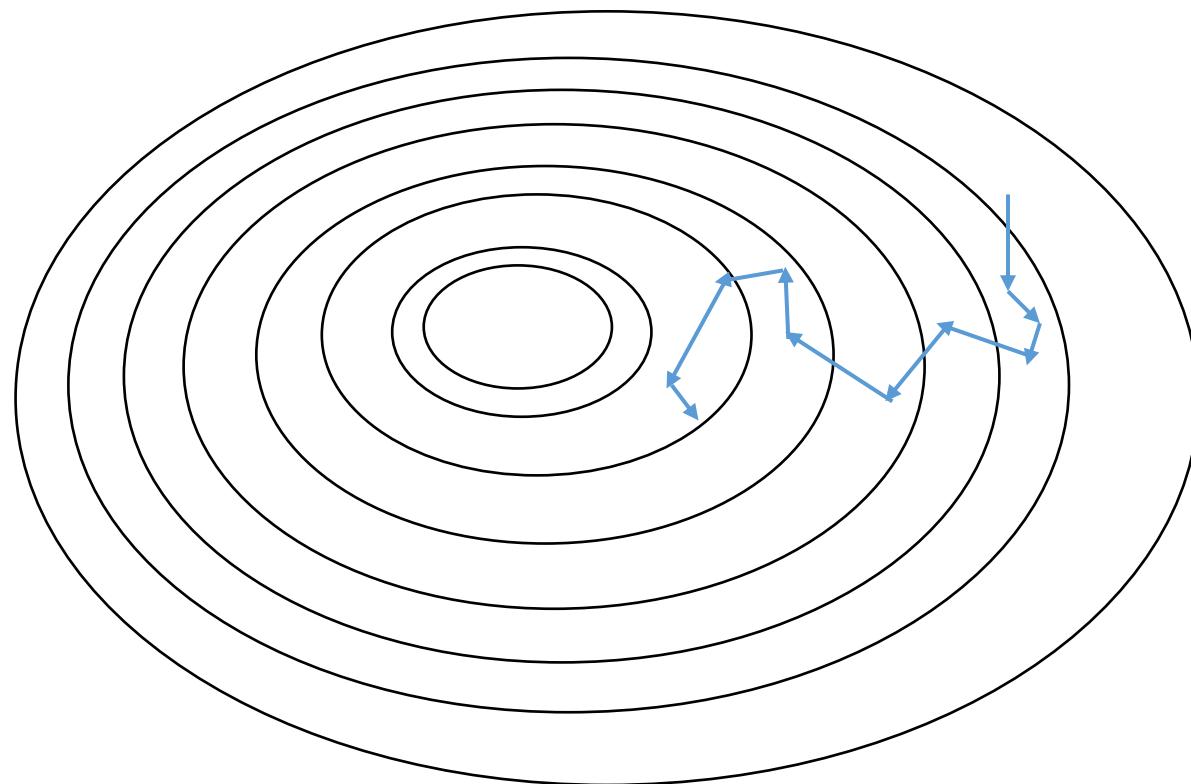
Stochastic Gradient descent

# Gradient Descent vs SGD



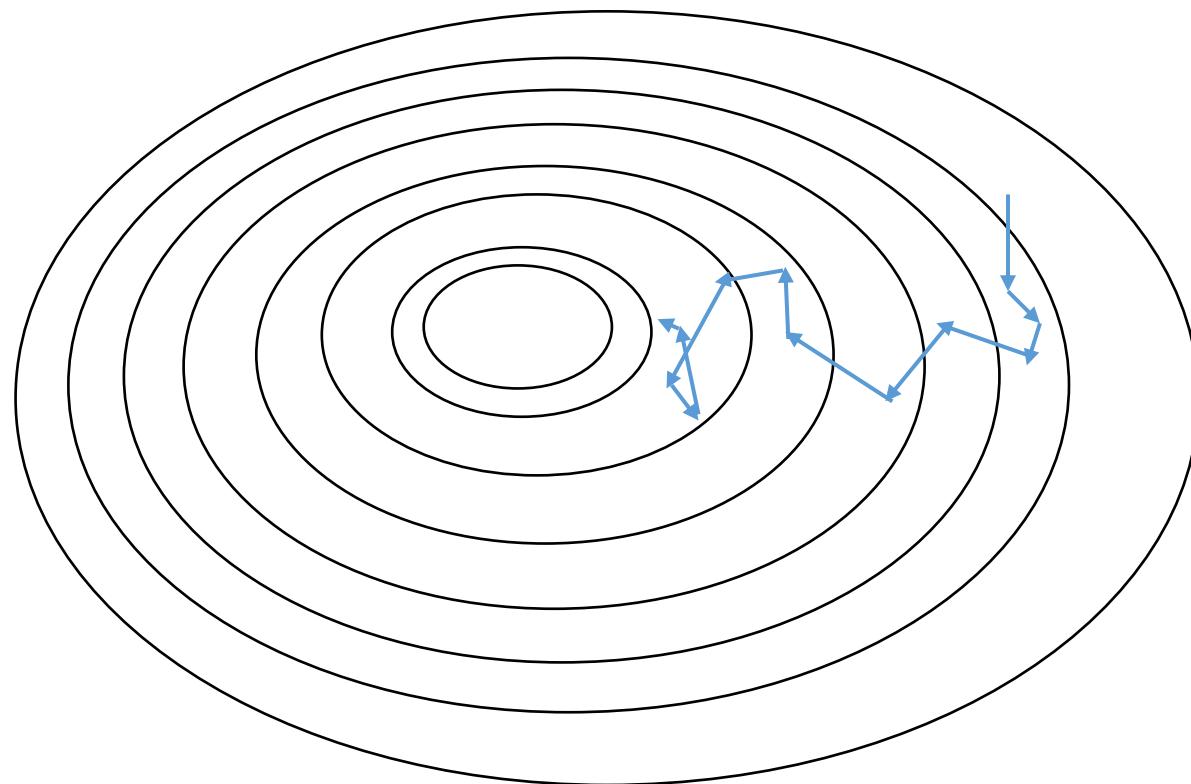
Stochastic Gradient descent

# Gradient Descent vs SGD



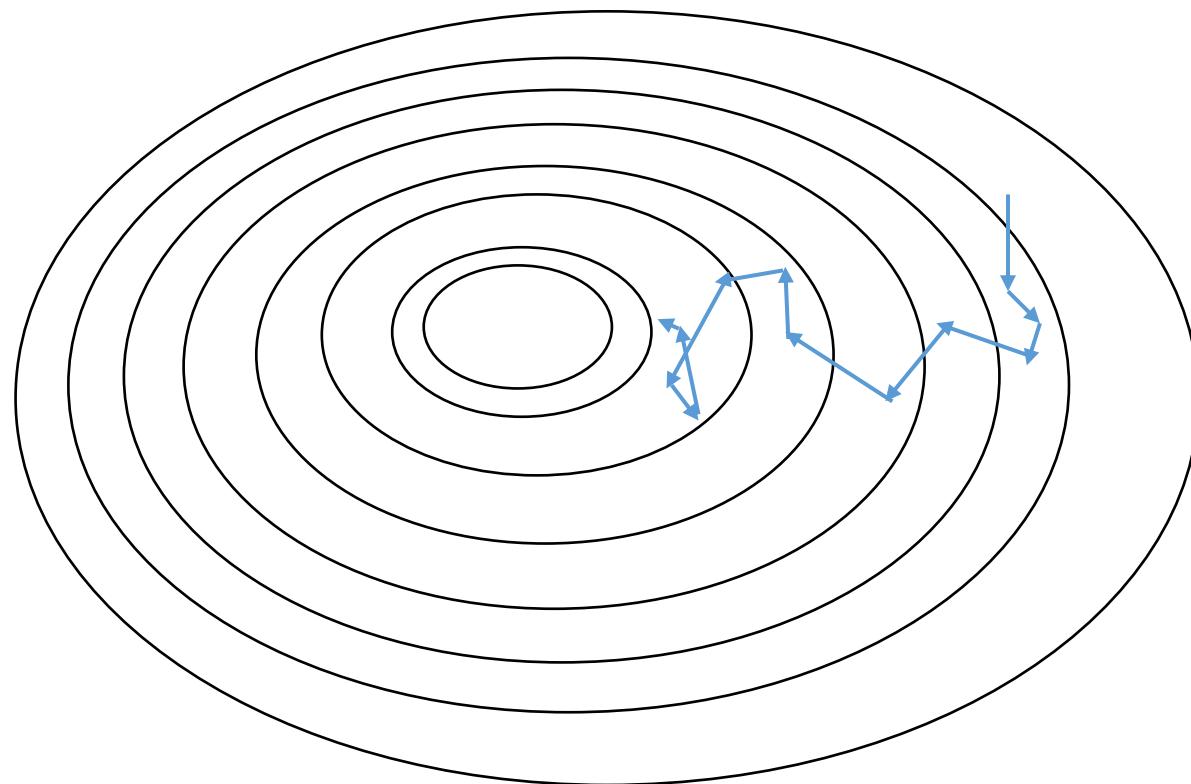
Stochastic Gradient descent

# Gradient Descent vs SGD



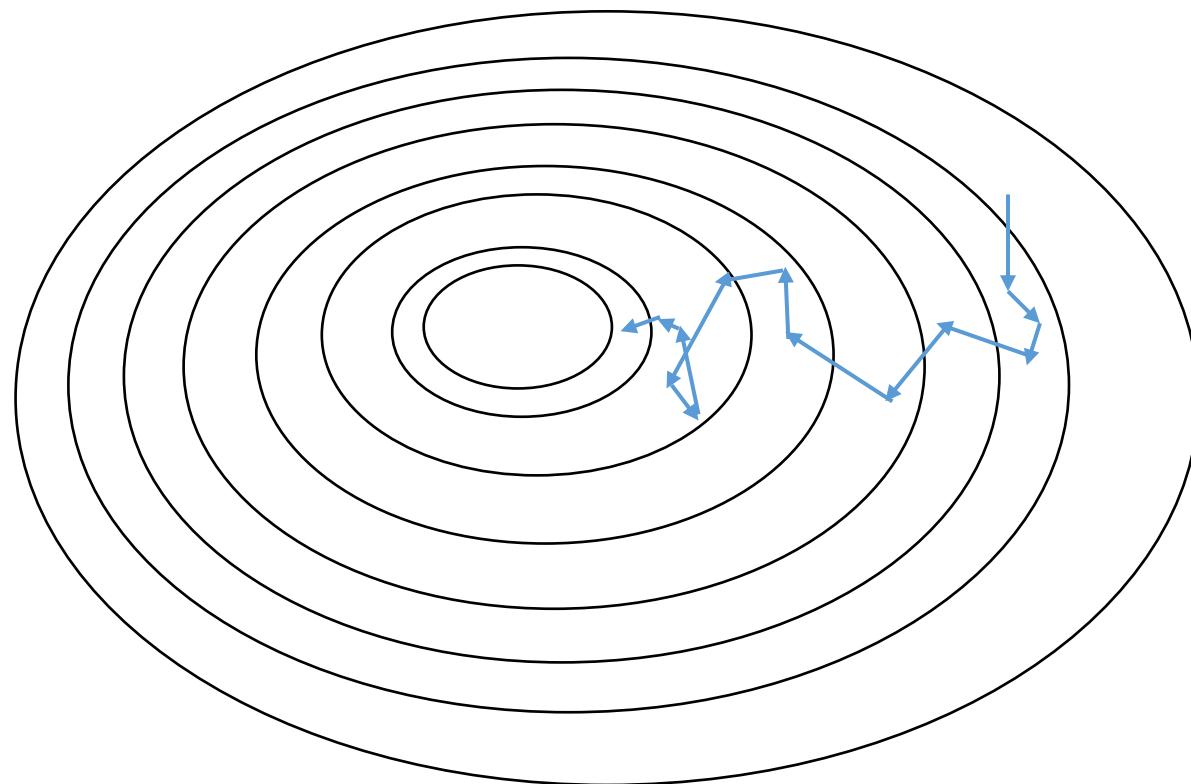
Stochastic Gradient descent

# Gradient Descent vs SGD



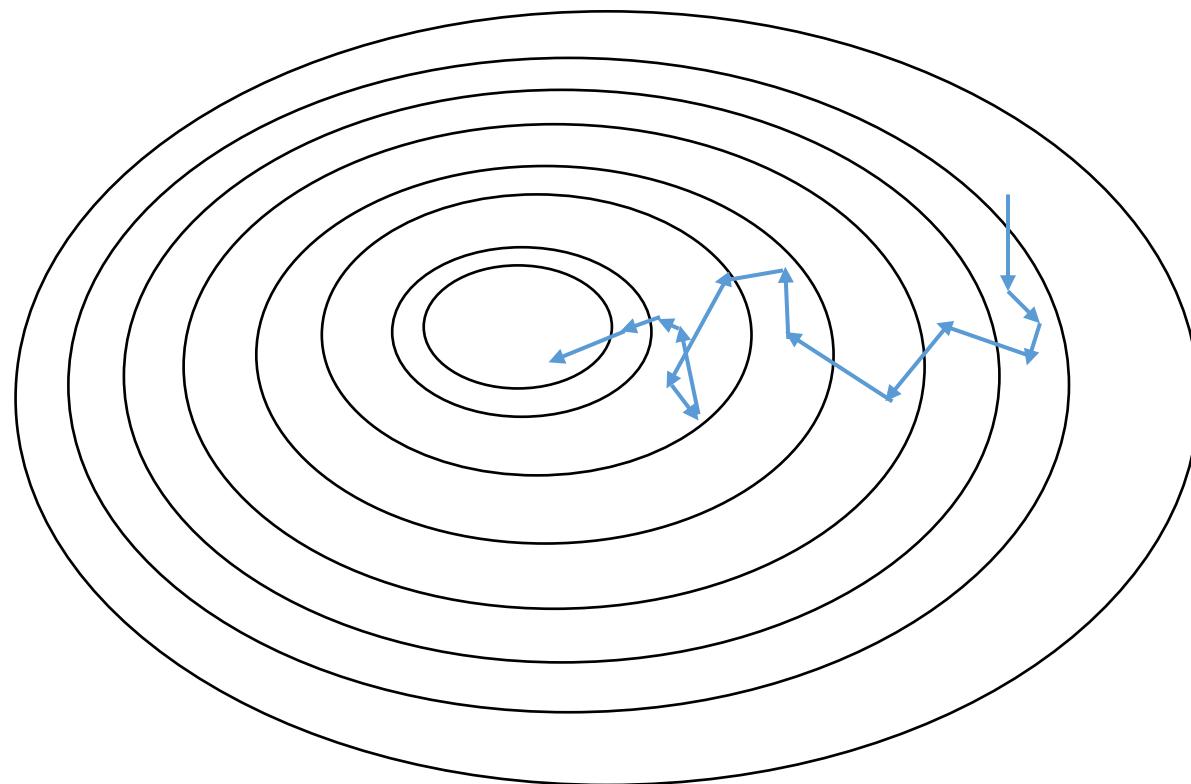
Stochastic Gradient descent

# Gradient Descent vs SGD



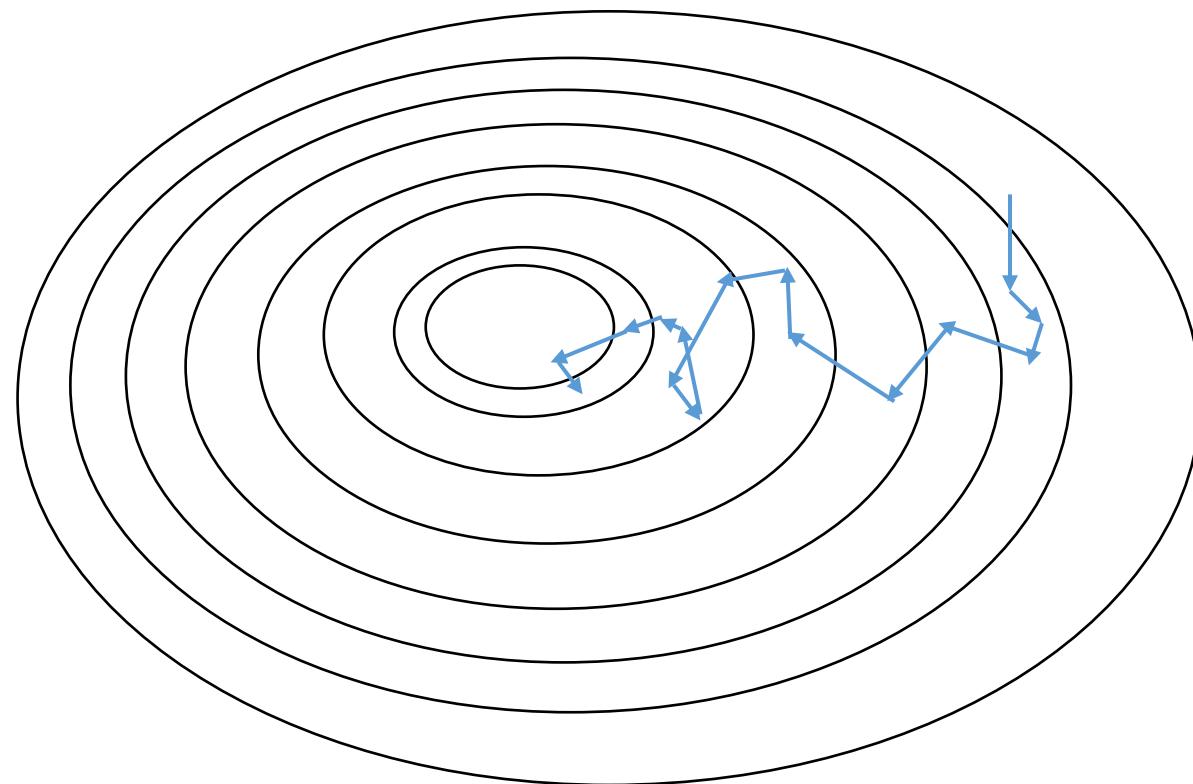
Stochastic Gradient descent

# Gradient Descent vs SGD



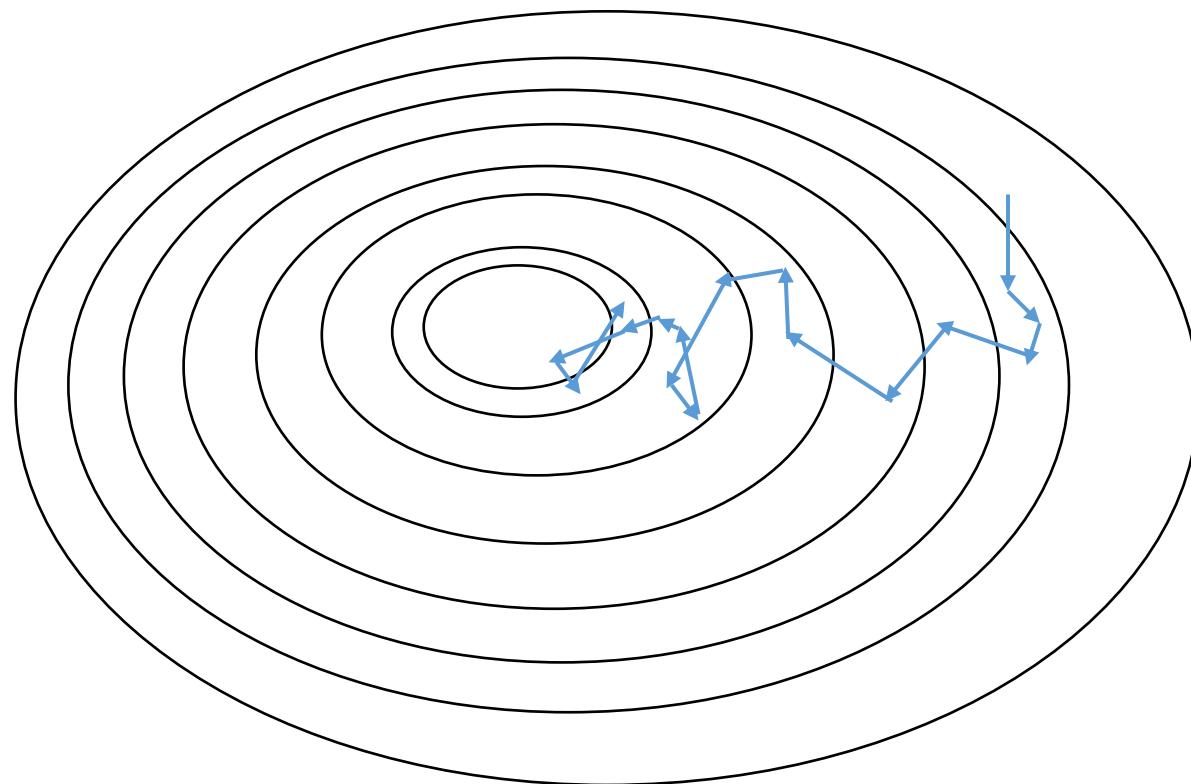
Stochastic Gradient descent

# Gradient Descent vs SGD



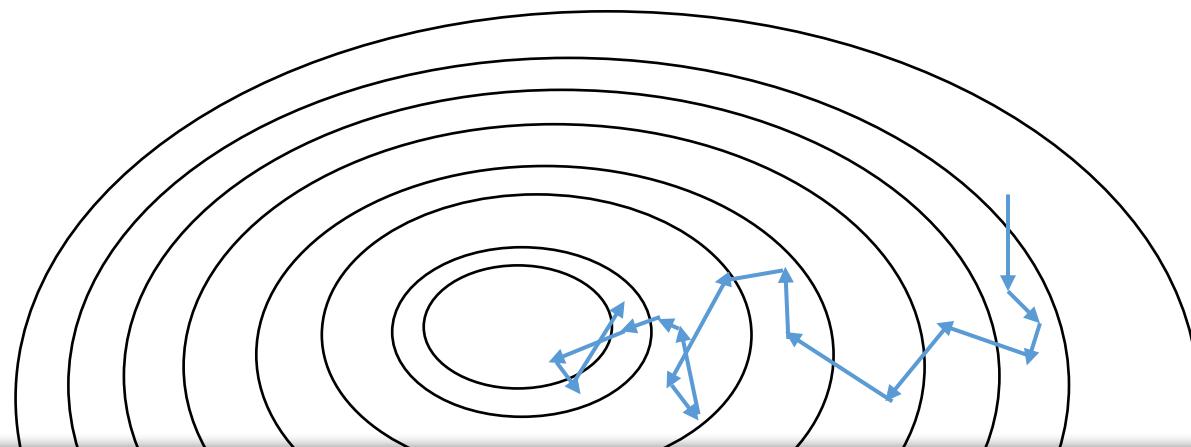
Stochastic Gradient descent

# Gradient Descent vs SGD



Stochastic Gradient descent

# Gradient Descent vs SGD



Many more updates than gradient descent, but each individual update is less computationally expensive

Stochastic Gradient descent

# Outline: Training SVM by optimization

1. Check convexity
2. Stochastic gradient descent
3. Sub-derivatives of the hinge loss
4. Stochastic sub-gradient descent for SVM
5. Comparison to perceptron

# Hinge loss is not differentiable!

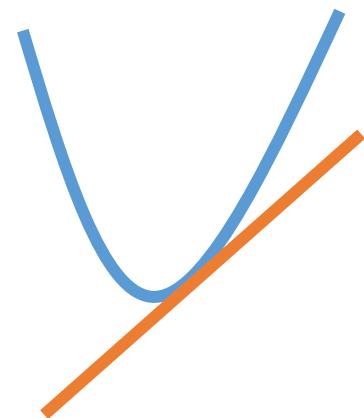
**What is the derivative of the hinge loss with respect to w?**

$$\frac{1}{2} w^T w + C \max(0, 1 - y_i(w^T x_i + b))$$

# Detour: Sub-gradients

Generalization of gradients to non-differentiable functions

(Remember that every tangent lies below the function for convex functions)



Informally, a sub-tangent at a point is any line lies below the function at the point.

A sub-gradient is the slope of that line

# Advanced topic [not in exam] Sub-gradients

Formally,  $g$  is a subgradient to  $f$  at  $x$  if

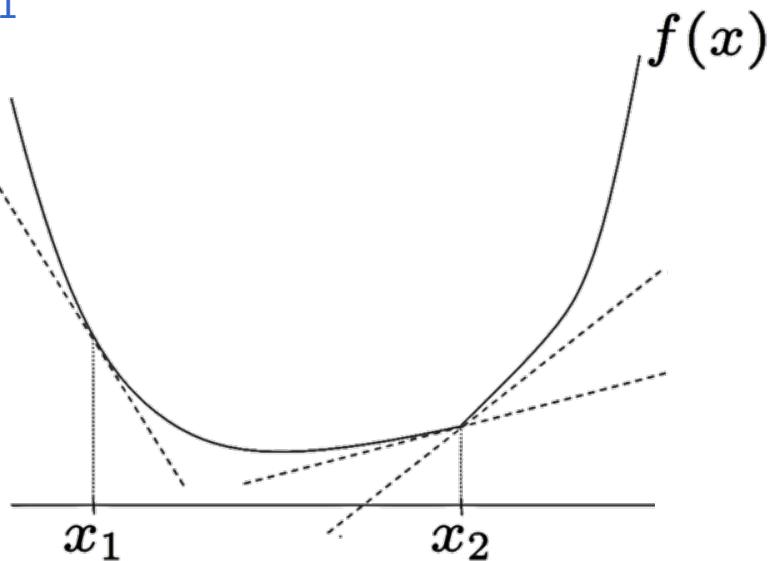
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

$f$  is differentiable at  $x_1$

Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

$g_1$  is a gradient at  $x_1$



# Sub-gradients

Formally,  $g$  is a subgradient to  $f$  at  $x$  if

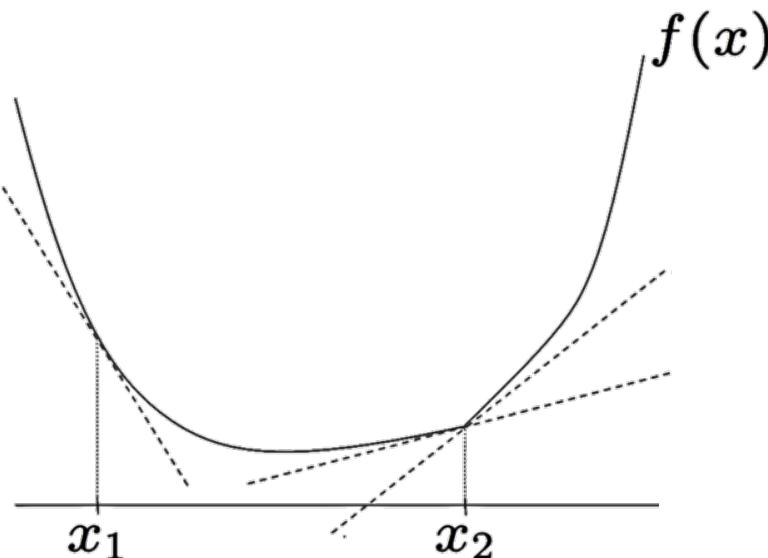
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

$f$  is differentiable at  $x_1$

Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

$g_1$  is a gradient at  $x_1$



# Sub-gradients

Formally,  $g$  is a subgradient to  $f$  at  $x$  if

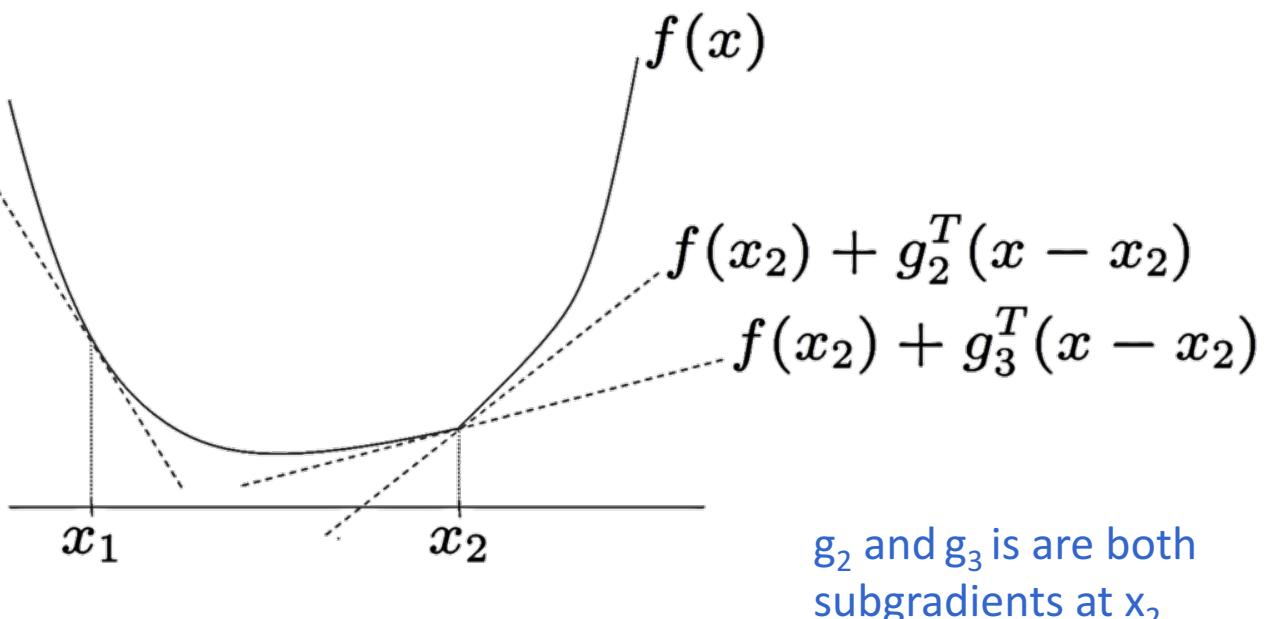
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

$f$  is differentiable at  $x_1$

Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

$g_1$  is a gradient at  $x_1$



# Sub-gradient of the SVM objective

$$J^t(w) = \frac{1}{2} w^T w + C \max(0, 1 - y_i(w^T x_i + b))$$

**General strategy:** First solve the max and compute the gradient for each case

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max(0, 1 - y_i(w^T x_i + b)) = 0 \\ \mathbf{w} - Cy_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

# Outline: Training SVM by optimization

1. Check convexity
2. Stochastic gradient descent
3. Sub-derivatives of the hinge loss
4. Stochastic sub-gradient descent for SVM
5. Comparison to perceptron

# Stochastic gradient Descent

Given a training set  $\mathcal{D} = \{(x, y)\}$

Initialize  $w \leftarrow 0 \in \mathbb{R}^n$

For epoch 1 ...  $T$ :

For  $(x, y)$  in  $\mathcal{D}$ :

if  $w^T x \geq 1$

$w \leftarrow w - \eta w$

else

$w \leftarrow w - \eta C y x$

Update  $w \leftarrow w - \eta \nabla_w f(x, y)$

Return  $w$

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max(0, 1 - y_i(w^T x_i + b)) = 0 \\ \mathbf{w} - C y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

# Outline: Training SVM by optimization

1. Check convexity
2. Stochastic gradient descent
3. Sub-derivatives of the hinge loss
4. Stochastic sub-gradient descent for SVM
5. Comparison to perceptron

# Stochastic gradient Descent

Given a training set  $\mathcal{D} = \{(x, y)\}$

Initialize  $w \leftarrow 0 \in \mathbb{R}^n$

For epoch 1 ...  $T$ :

    For  $(x, y)$  in  $\mathcal{D}$ :

        if  $y(w^T x + b) \geq 1$

$w \leftarrow w - \eta w$

        else

$w \leftarrow w - \eta(w - C y x)$

Update  $w \leftarrow w - \eta \nabla_w f(x, y)$

Return  $w$

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max(0, 1 - y_i(w^T x_i + b)) = 0 \\ \mathbf{w} - C y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

# Recap: The Perceptron Algorithm

Given a training set  $\mathcal{D} = \{(x, y)\}$

1. Initialize  $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For  $(x, y)$  in  $\mathcal{D}$ :
3.     if  $y(w^\top x + b) \leq 0$
4.          $w \leftarrow w + yx$
- 5.
6. Return  $w$

SVM:

If  $y(w^\top x + b) < 1$   
 $w \leftarrow w - \eta(w - Cyx)$   
else:  $w \leftarrow w - \eta w$

Prediction:  $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

# Perceptron vs. SVM

- ❖ Perceptron: Stochastic sub-gradient descent for a different loss
  - ❖ No regularization though

$$L_{\text{Perceptron}}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T \mathbf{x})$$

- ❖ SVM optimizes the hinge loss
  - ❖ With regularization

$$L_{\text{Hinge}}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

# This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

# SVM: Primals and duals

## The SVM objective

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$\begin{aligned} s.t. \quad & \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

This is called the *primal form* of the objective

This can be converted to its *dual form*, which will let us prove a very useful property

# SVM: Primals and duals

## The SVM objective

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

This is called the *primal form* of the objective

This can be converted to its *dual form*, which will let us prove a very useful property

Another optimization problem

Has the property that  
max Dual = min Primal

# Support vector machines

Let  $\mathbf{w}$  be the minimizer of the SVM problem for some dataset with  $m$  examples:  $\{(\mathbf{x}_i, y_i)\}$

Then, for  $i = 1 \dots m$ , there exist  $\alpha_i \geq 0$  such that the optimum  $\mathbf{w}$  can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

# Support vector machines

Let  $\mathbf{w}$  be the minimizer of the SVM problem for some dataset with  $m$  examples:  $\{(\mathbf{x}_i, y_i)\}$

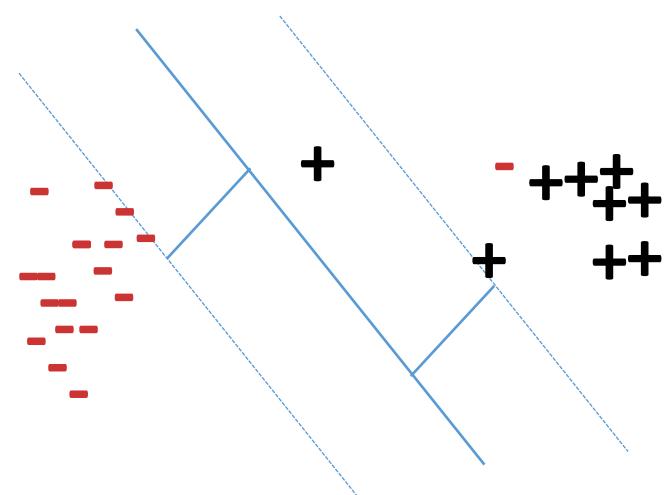
Then, for  $i = 1 \dots m$ , there exist  $\alpha_i \geq 0$  such that the optimum  $\mathbf{w}$  can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\alpha_i = 0 \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

All points outside the margin



# Support vector machines

Let  $\mathbf{w}$  be the minimizer of the SVM problem for some dataset with  $m$  examples:  $\{(\mathbf{x}_i, y_i)\}$

Then, for  $i = 1 \dots m$ , there exist  $\alpha_i \geq 0$  such that the optimum  $\mathbf{w}$  can be written as

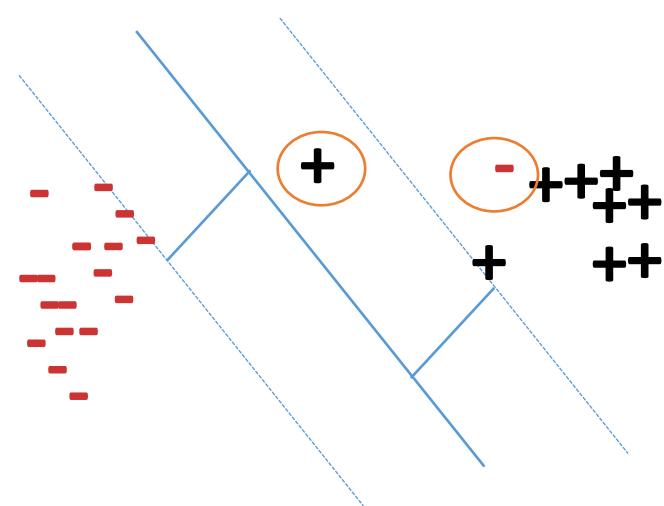
$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\alpha_i = 0 \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

$$\alpha_i = C \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \leq 1$$

All points on the wrong side of the margin



# Support vector machines

Let  $\mathbf{w}$  be the minimizer of the SVM problem for some dataset with  $m$  examples:  $\{(\mathbf{x}_i, y_i)\}$

Then, for  $i = 1 \dots m$ , there exist  $\alpha_i \geq 0$  such that the optimum  $\mathbf{w}$  can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

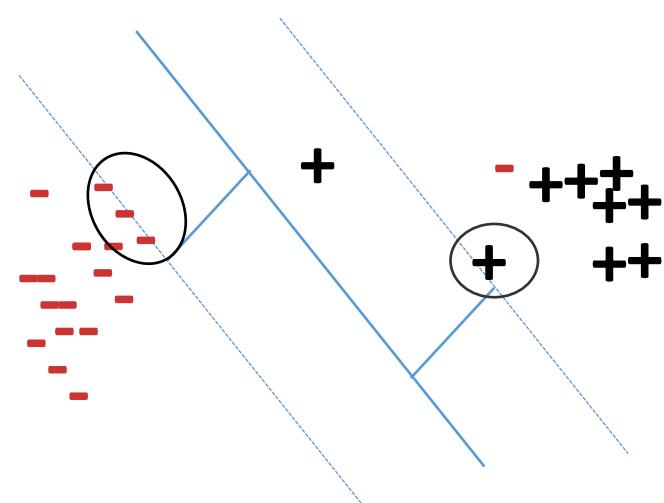
Furthermore,

$$\alpha_i = 0 \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

$$\alpha_i = C \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \leq 1$$

$$0 \leq \alpha_i \leq C \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i = 1$$

All points on the margin



# Support vectors

The weight vector is completely defined by training examples whose  $\alpha_i$ s are not zero

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

These examples are called the *support vectors*