Lecture 13: Support Vector Machines
Winter 2018

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, 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.

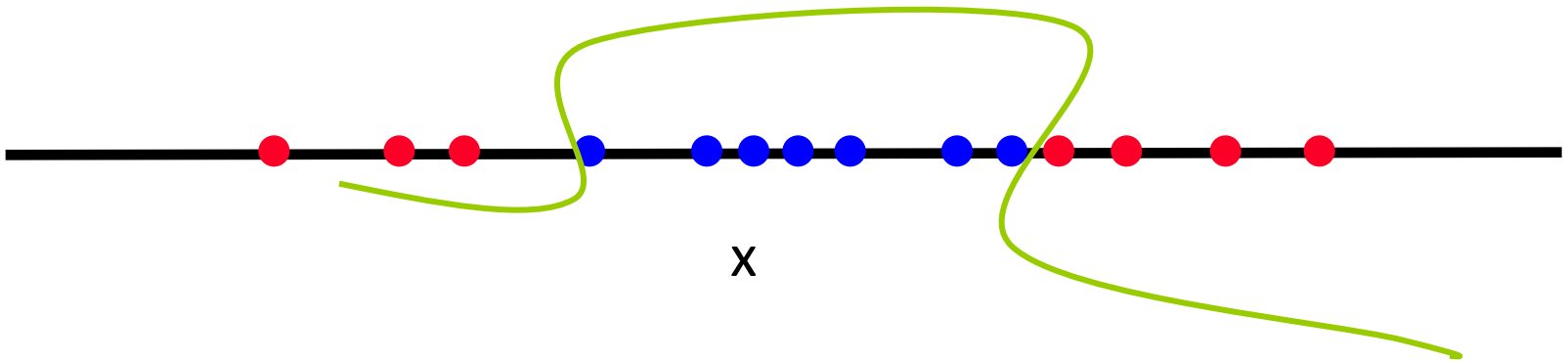
Announcement

- Hw3 will be released tonight

Kernel and Kernel methods

Functions Can be Made Linear

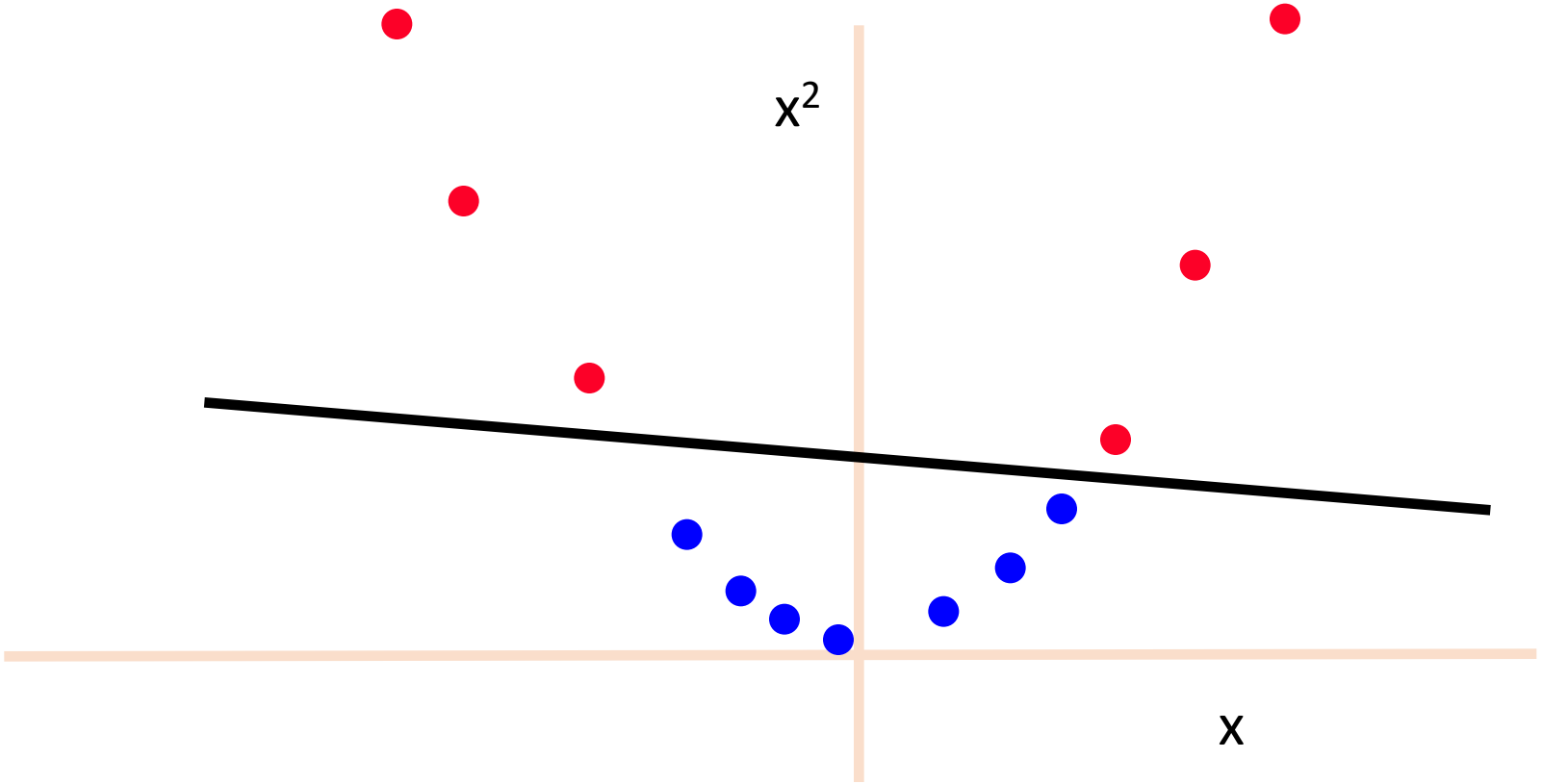
- Data are not linearly separable in one dimension
- Not separable if you insist on using a specific class of functions



Can we do some mapping to make it linear spreadable?

Blown Up Feature Space

- Data are separable in <x, x^2> space



The Perceptron Algorithm [Rosenblatt 1958]

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

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

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

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

The Perceptron Algorithm [Rosenblatt 1958]

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

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^{2n}$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y\, \boldsymbol{w}^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$
5.

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

What if our mapping function is more complex?
E.g., mapping data to infinite # dimensions

Ans: it's okay if we can compute $\phi(x_i)^T \phi(x_j)$

Dual Representation

if $y(\boldsymbol{w}^\top \boldsymbol{x}) \leq 0$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$

- Let w be an initial weight vector for perceptron. Let $(x_1,+)$, $(x_2,+)$, $(x_3,-)$, $(x_4,-)$ be examples and assume mistakes are made on x_1, x_2 and x_4.
- What is the resulting weight vector?

  w = w + x_1 + x_2 - x_4

- In general, the weight vector w can be written as a linear combination of examples:

$$w = \sum_{1..m} \alpha_i \, y_i x_i$$

- Where α_i is the number of mistakes made on x_i.

Predicting with linear classifiers

- Prediction = $sgn(\mathbf{w}^T\mathbf{x})$ and $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$
- That is, we just showed that

$$\mathbf{w}^T\mathbf{x} = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

  - We only need to compute dot products between training examples and the new example **x**
- This is true even if we map examples to a high dimensional space

$$\mathbf{w}^T\phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Many learning algorithm require to compute inner products

- Perceptron:

$$y(\boldsymbol{w}^{\top}\boldsymbol{x}) \leq \mathbf{0}$$

- K-NN:

$$similarity\left(\boldsymbol{x}, \boldsymbol{x}^{\boldsymbol{neighbor}}\right) = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{neighbor}}$$

$$dist\left(\boldsymbol{x}, \boldsymbol{x}^{\boldsymbol{neighbor}}\right) = ||\boldsymbol{x} - \boldsymbol{x}^{\boldsymbol{neighbor}}||^{2}$$

$$dist\left(\boldsymbol{x}, \boldsymbol{x}^{\boldsymbol{neighbor}}\right) = \left|\left|\boldsymbol{x}\right|\right|^{2} + \left|\left|\boldsymbol{x}^{\boldsymbol{neighbor}}\right|\right|^{2} - 2\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{neighbor}}$$

Is there a smarter way to compute the inner product?

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$sgn(\mathbf{w}^T \phi(\mathbf{x})) = sgn\left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})\right)$$

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$\left(\sum_i \alpha_i y_i K(\quad) \right)$$

If we can compute the value of K without explicitly writing the blown up representation, then we will have a computational advantage.

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Example

- Assume the mapping:

$$\phi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_D \\ {x_1}^2 \\ x_1x_2 \\ \vdots \\ x_1x_D \\ x_2x_1 \\ {x_2}^2 \\ \vdots \\ x_2x_D \\ \vdots \\ x_Dx_1 \\ \vdots \\ {x_D}^2 \end{pmatrix}$$

Computing $\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})$ is $\mathrm{O}(D^2)$.

Inner product can be computed efficently

Many learning algorithms can be rewritten to depend on the instances $\boldsymbol{x}_i, \boldsymbol{x}_j$ only through inner products $\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$.

Why is this helpful ?

$$\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^2$$

Inner product can be computed in $\mathrm{O}(D)$.

Example: polynomial kernel

Let us examine more closely the inner products $\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$ for a pair of data points $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$.

Polynomial-based nonlinear basis functions consider the following $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{\phi} : \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) &= x_{m1}^2x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2 \end{aligned}$$

Namely, the inner product can be computed by a function $(\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2$ defined in terms of the original features, *without computing* $\boldsymbol{\phi}(\cdot)$.

The Kernel Trick

Suppose we wish to compute

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Here ϕ maps **x** and **z** to a high dimensional space

The Kernel Trick: Save time/space by computing the value of K(**x**, **z**) by performing operations in the original space (without a feature transformation!)

Kernel function

Kernel functions

Definition: a kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

Examples we have seen

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$$
$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / 2\sigma^2}$$

Example that is not a kernel

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2$$

Exercise

- Showing $\left(4 + 9x_i^T x_j\right)^2$ is a valid kernel.

Kernel Matrix

Without specifying $\phi(\cdot)$, the kernel matrix

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is exactly the same as

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$$

$$= \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \end{pmatrix}$$

Conditions for being a positive semidefinite kernel function

Mercer theorem (loosely), a bivariate function $k(\cdot, \cdot)$ is a kernel function, if and only if, for *any N and any $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., and $\boldsymbol{x}_N$*, the matrix

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is positive semidefinite.

Support Vector Machine

Recap: The Perceptron Algorithm [Rosenblatt 1958]

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

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

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

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

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 $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

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

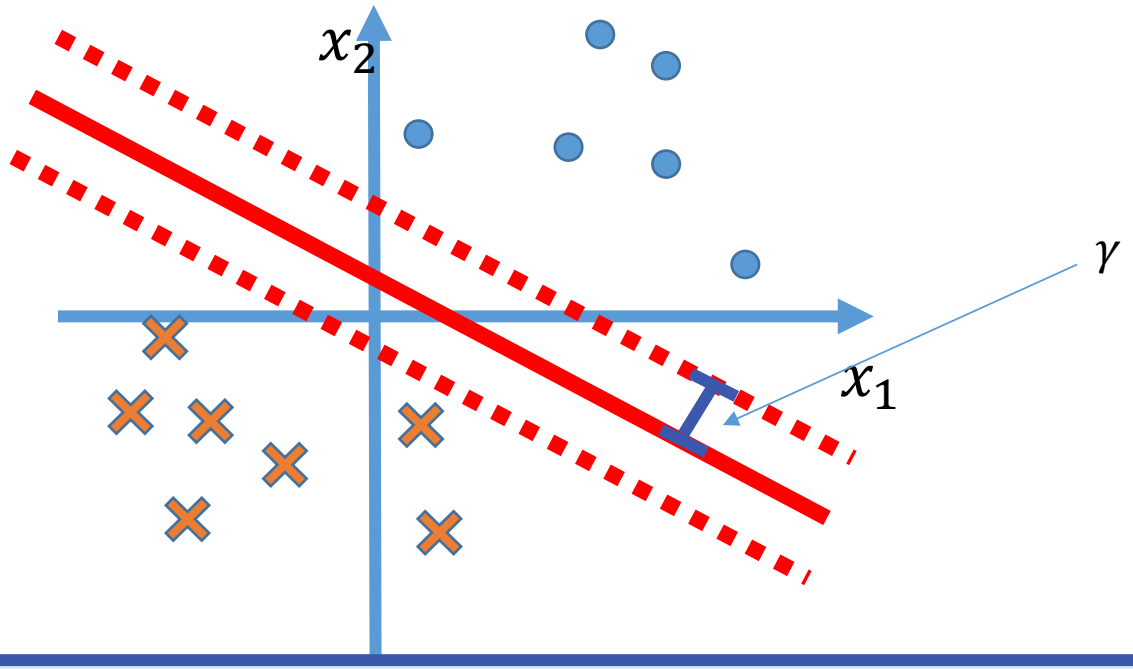
Assume $y \in \{1, -1\}$

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

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

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 γ automatically?

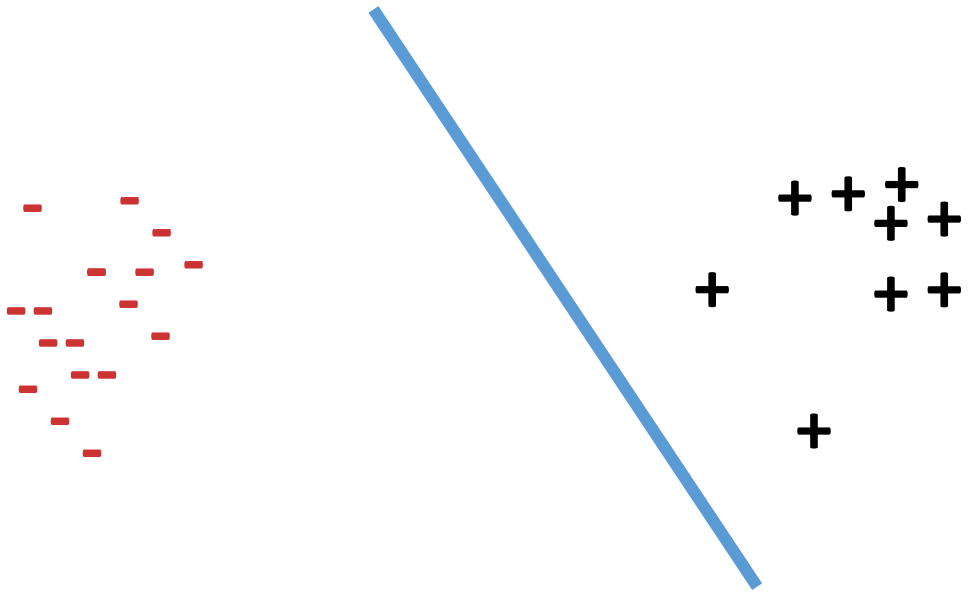
This lecture: Support vector machines

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


SUPPORT VECTOR MACHINES
REPEA
POWE
LAWS
END
DUALITY
GAP
Map Reduce
Map Reuse
Map Recycle
Green Data Processing
BAYESIANS AGAINST DISCRIMINATION
FREE VARIABLES!
BAN GENETIC ALGORITHMS

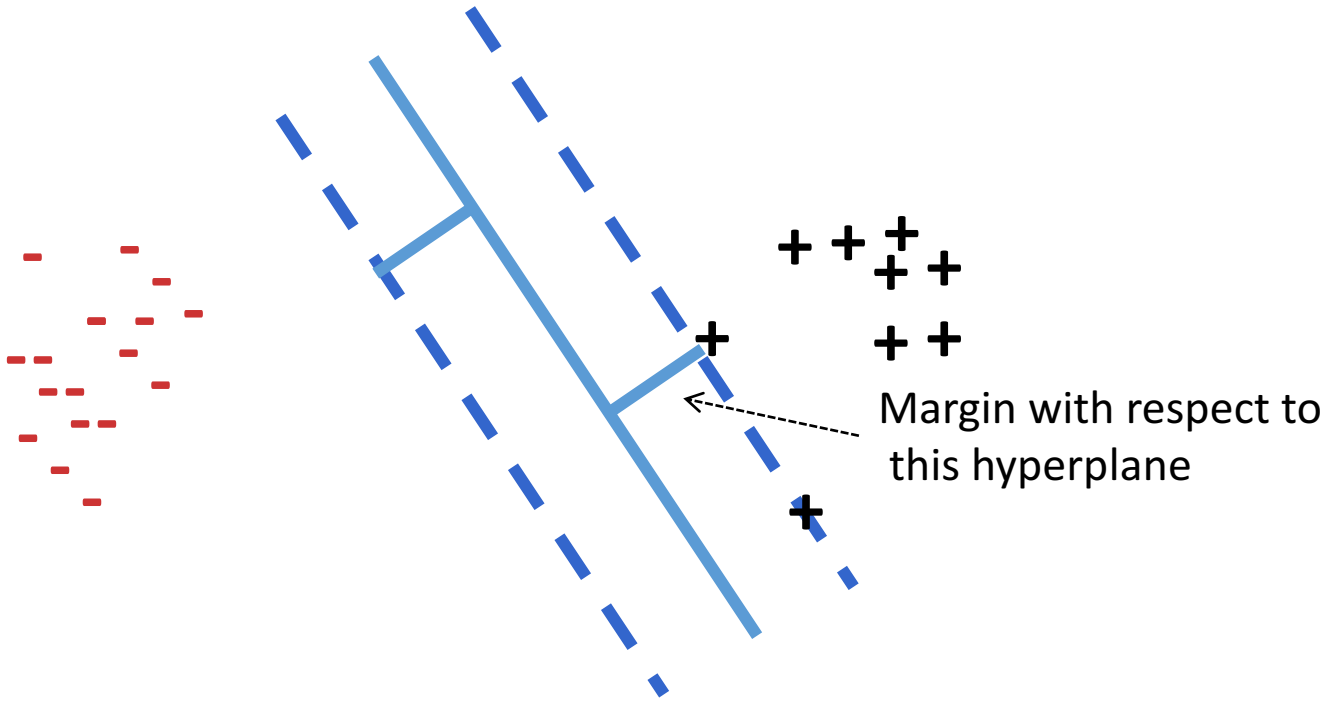
Recall: Margin

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

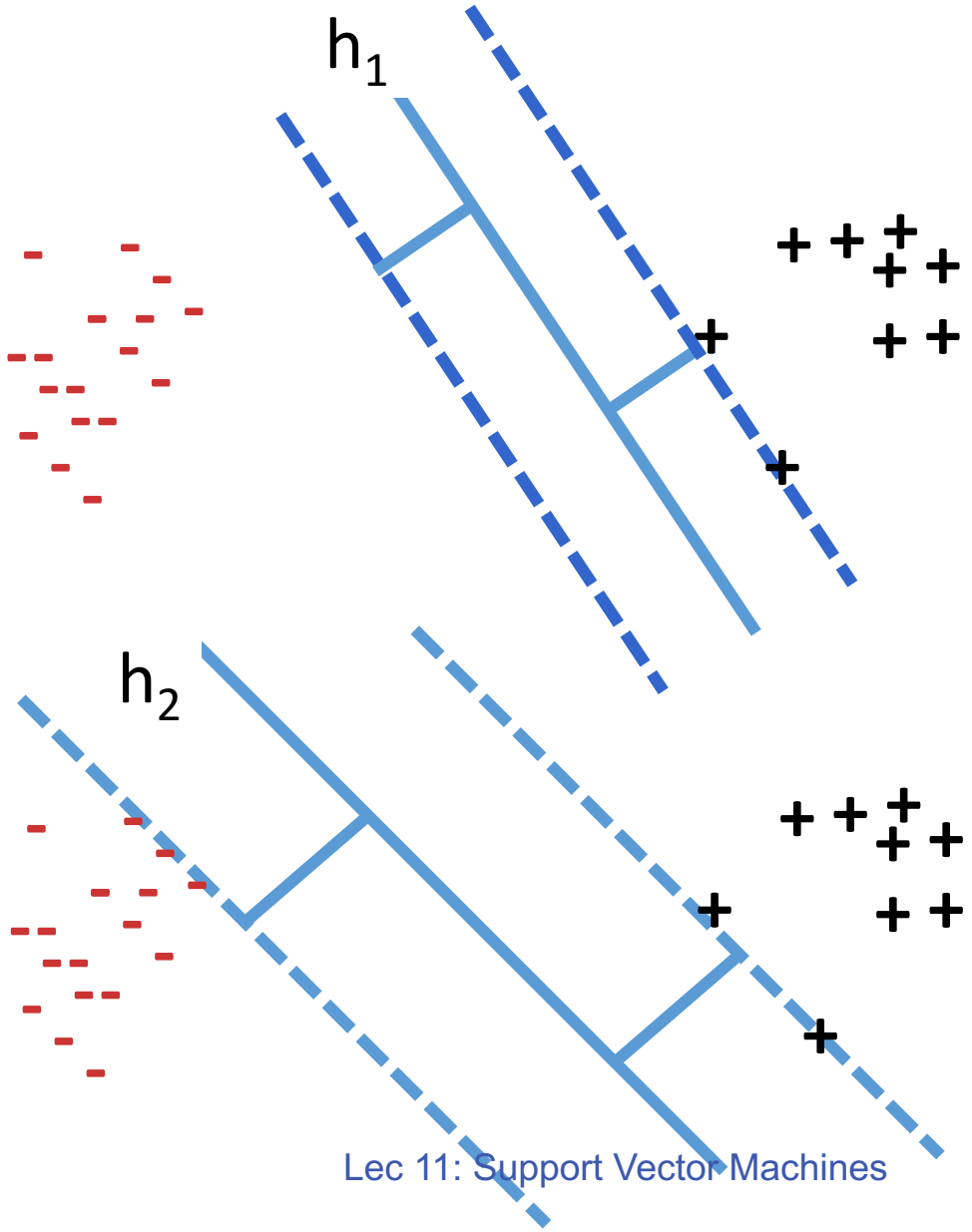


Recall: Margin

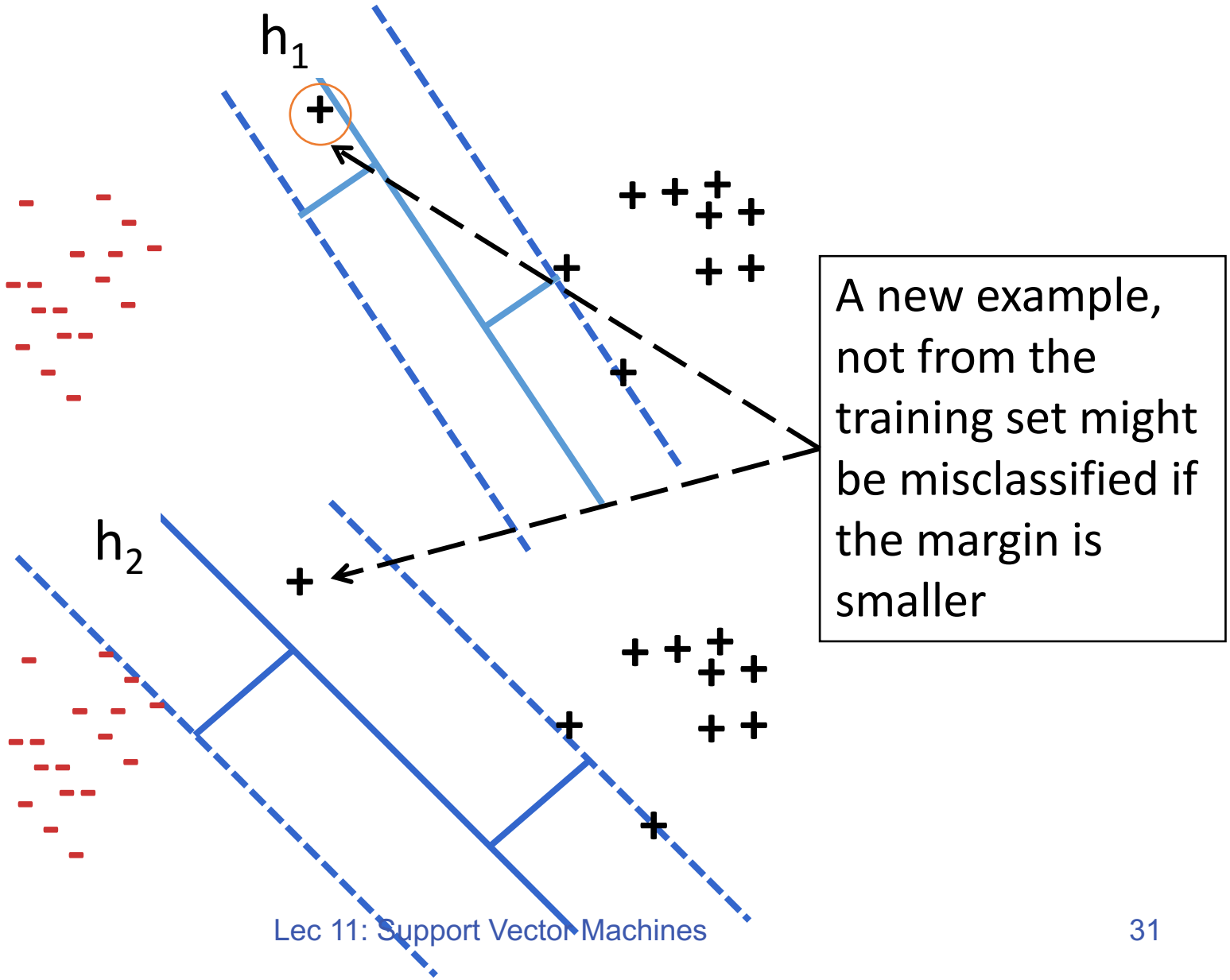
The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Which line is a better choice? Why?



Which line is a better choice? Why?



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 γ
- 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 γ
- 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

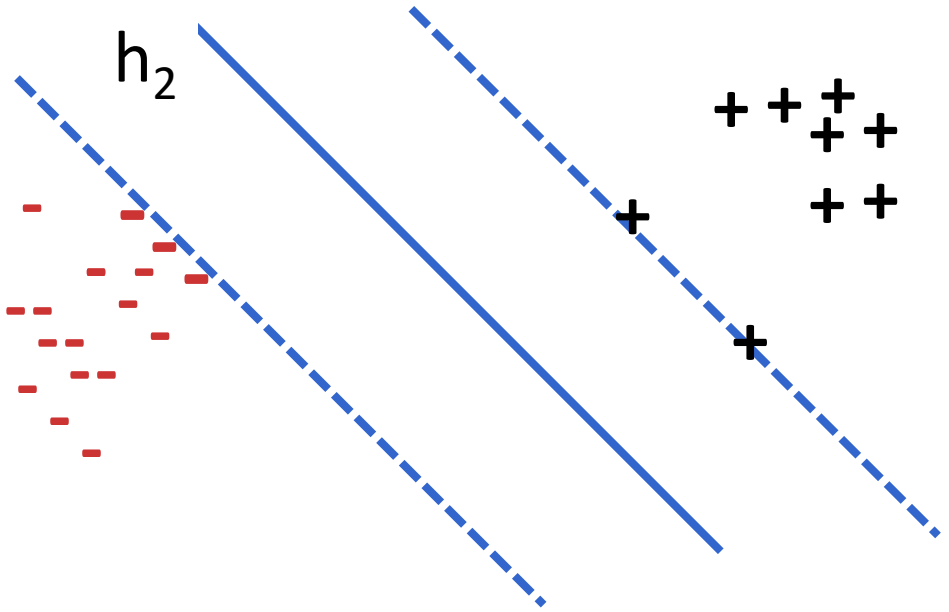
Larger margin → Lower VC dimension

Lower VC dimension → Better generalization bound

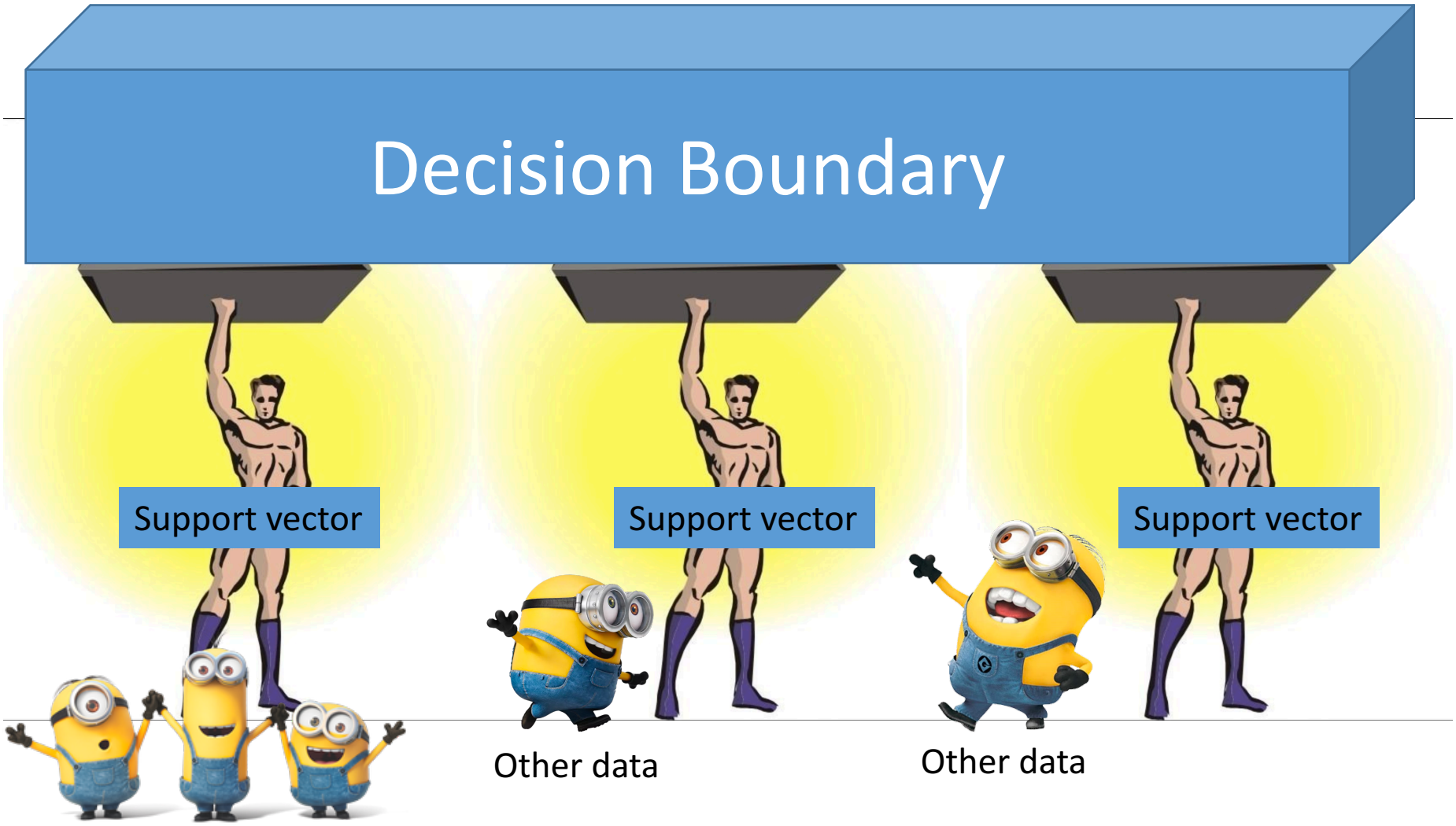
Learning strategy

Find the linear separator that maximizes the margin

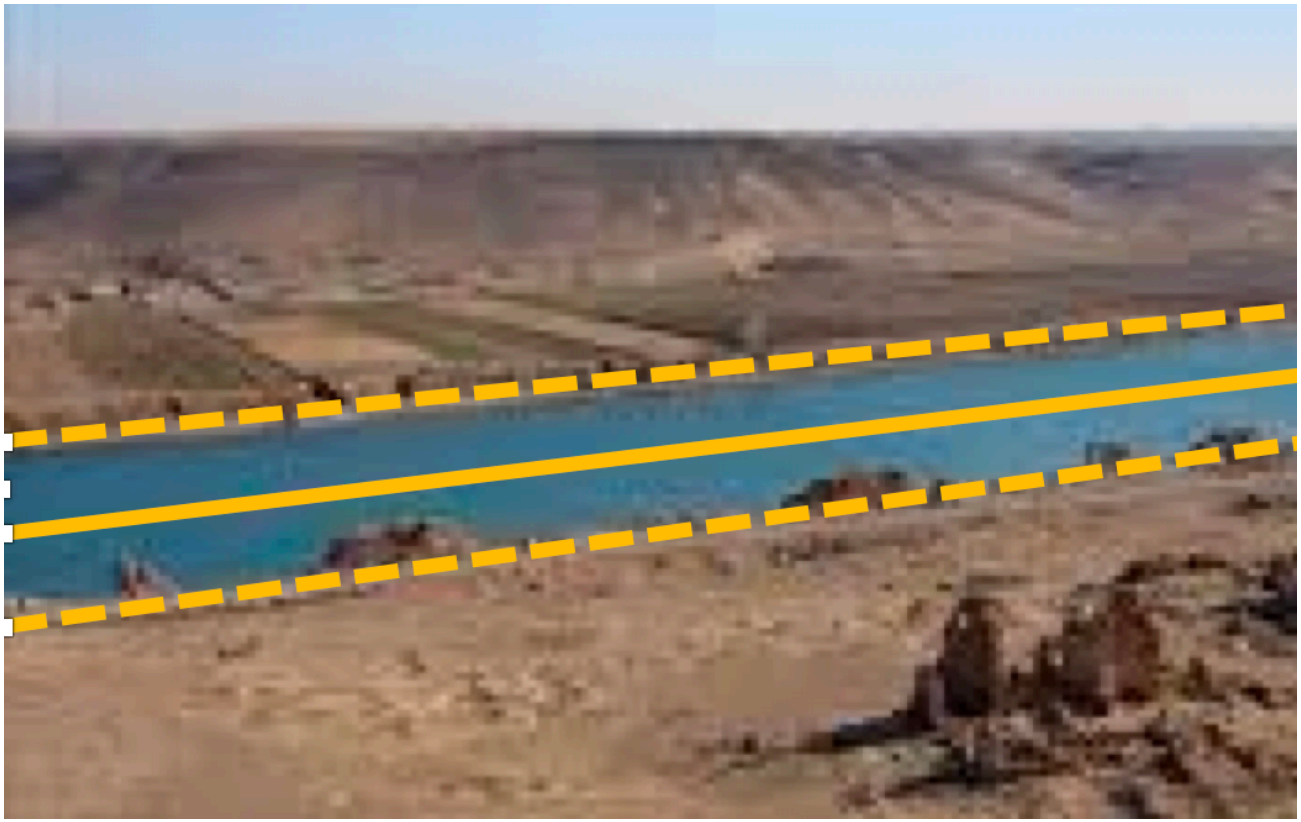
Why it called support vector machines?



Decision Boundary



Why it called support vector machines?

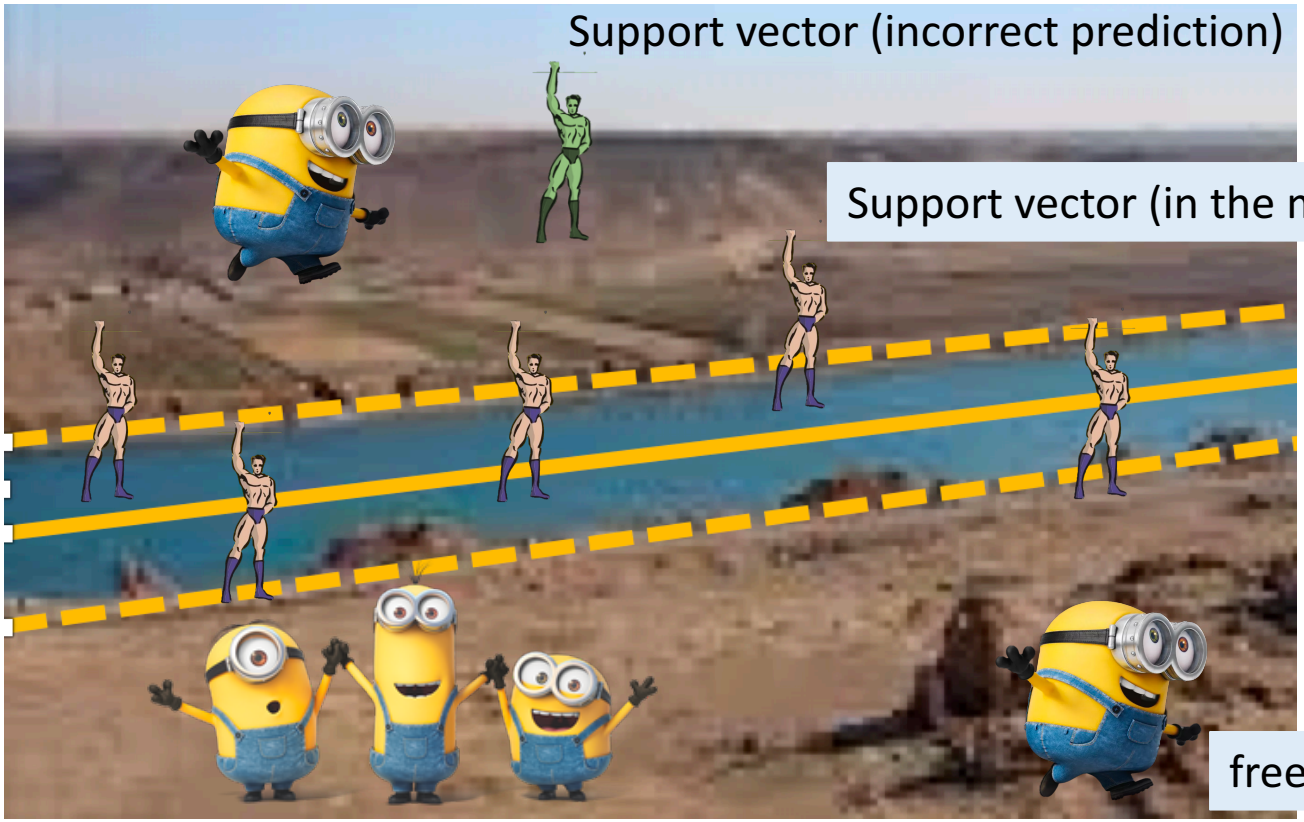


margin (upper)

Decision boundary

margin (lower)

Why it called support vector machines?

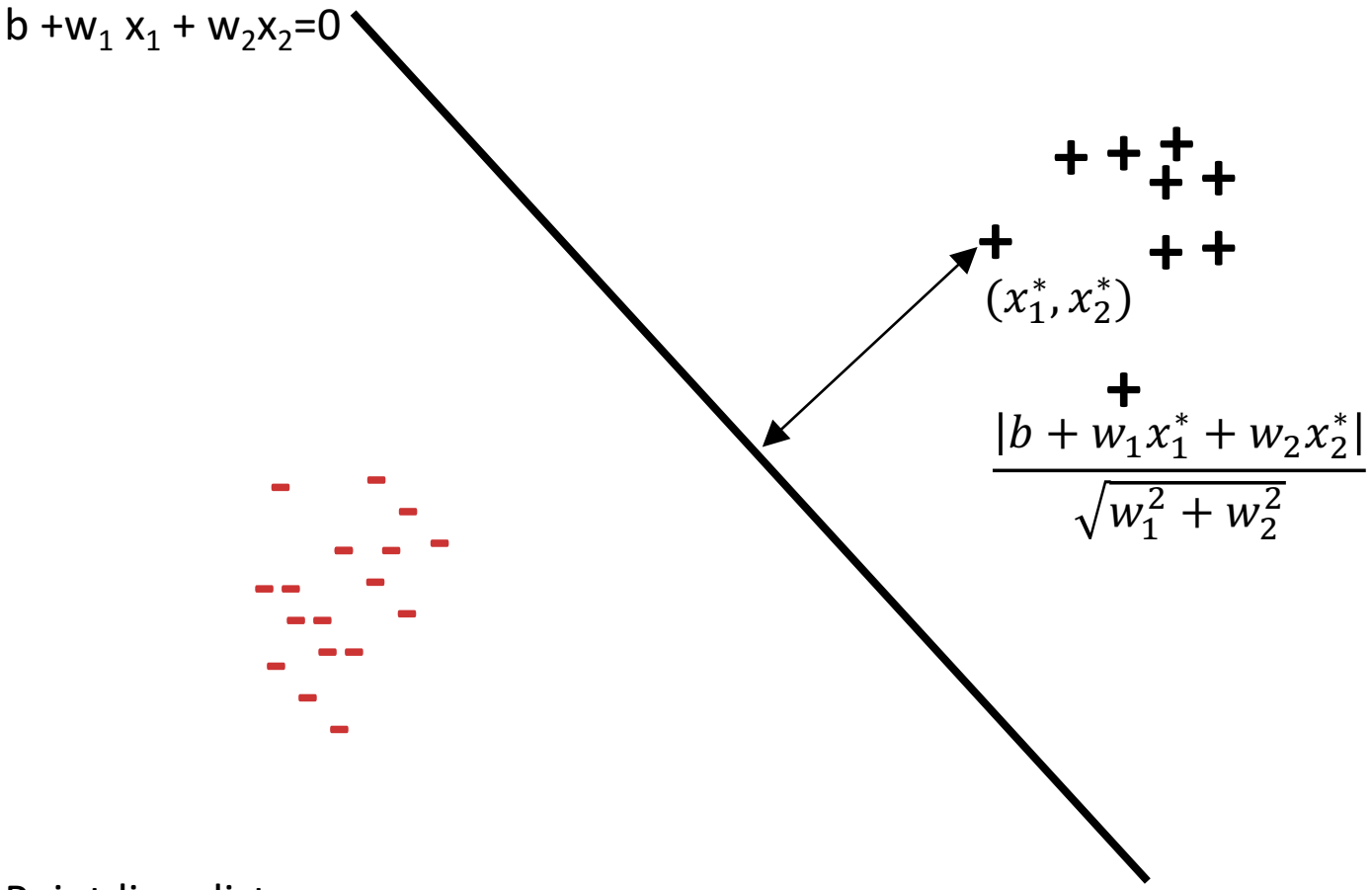


This lecture: Support vector machines

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

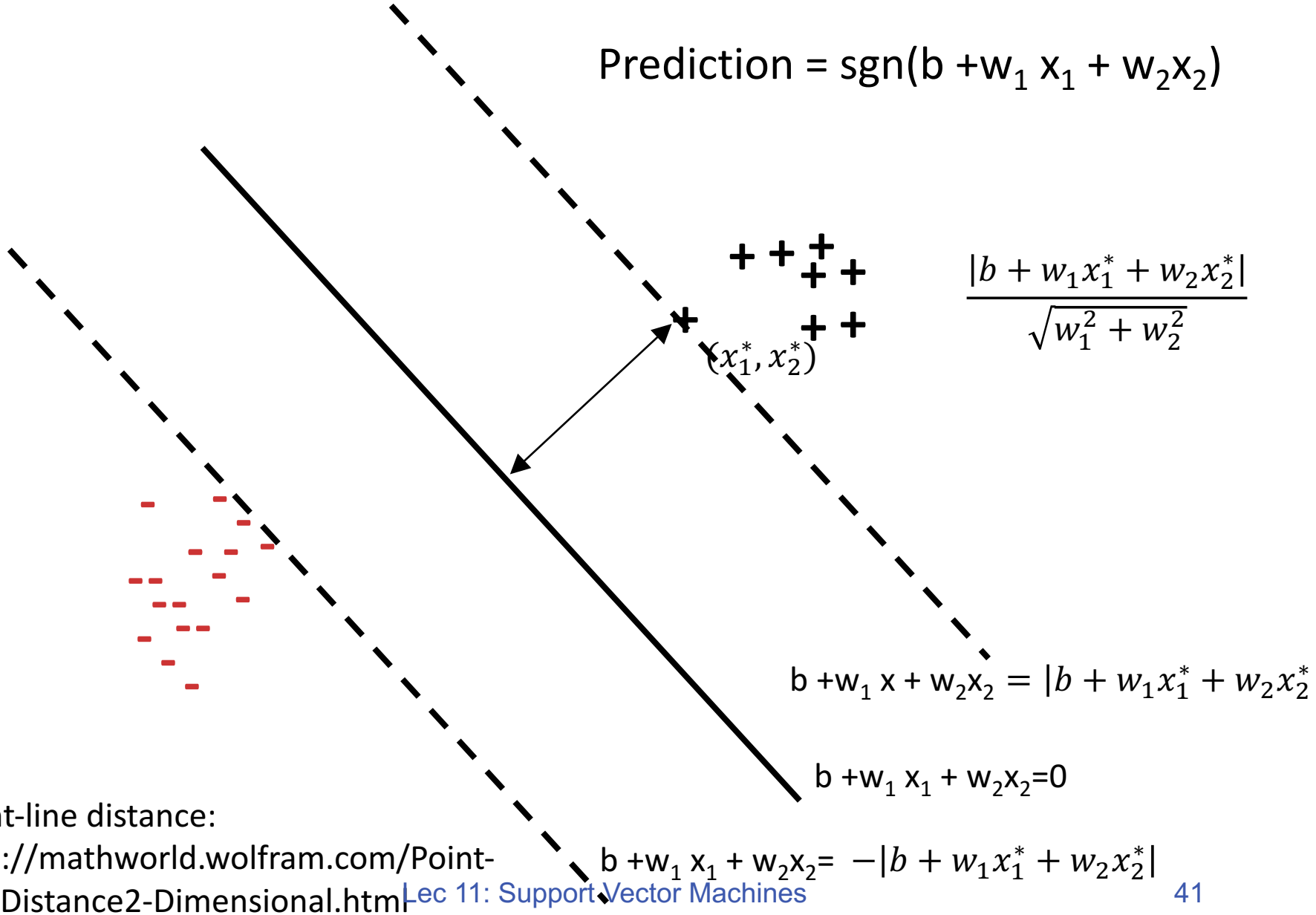
Recall: The geometry of a linear classifier

Prediction = sgn(b +w_1 x_1 + w_2x_2)



Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Recall: The geometry of a linear classifier



Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Maximizing margin

- Margin = distance of the closest point from the hyperplane

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

Maximizing margin

- Margin = distance of the closest point from the hyperplane

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

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

Some people call this the *geometric margin*

The numerator alone is called the *functional margin*

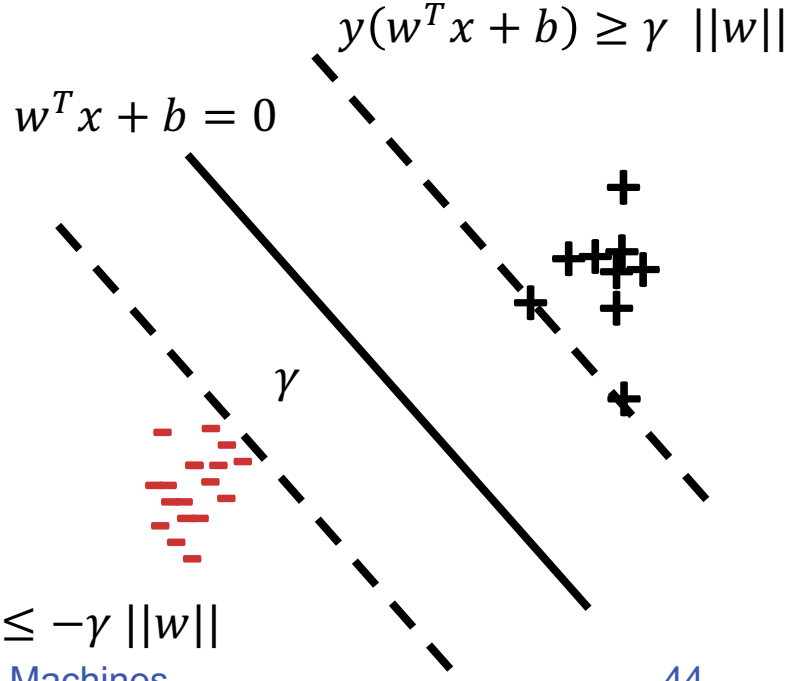
Maximizing margin

- Margin = distance of the closest point from the hyperplane

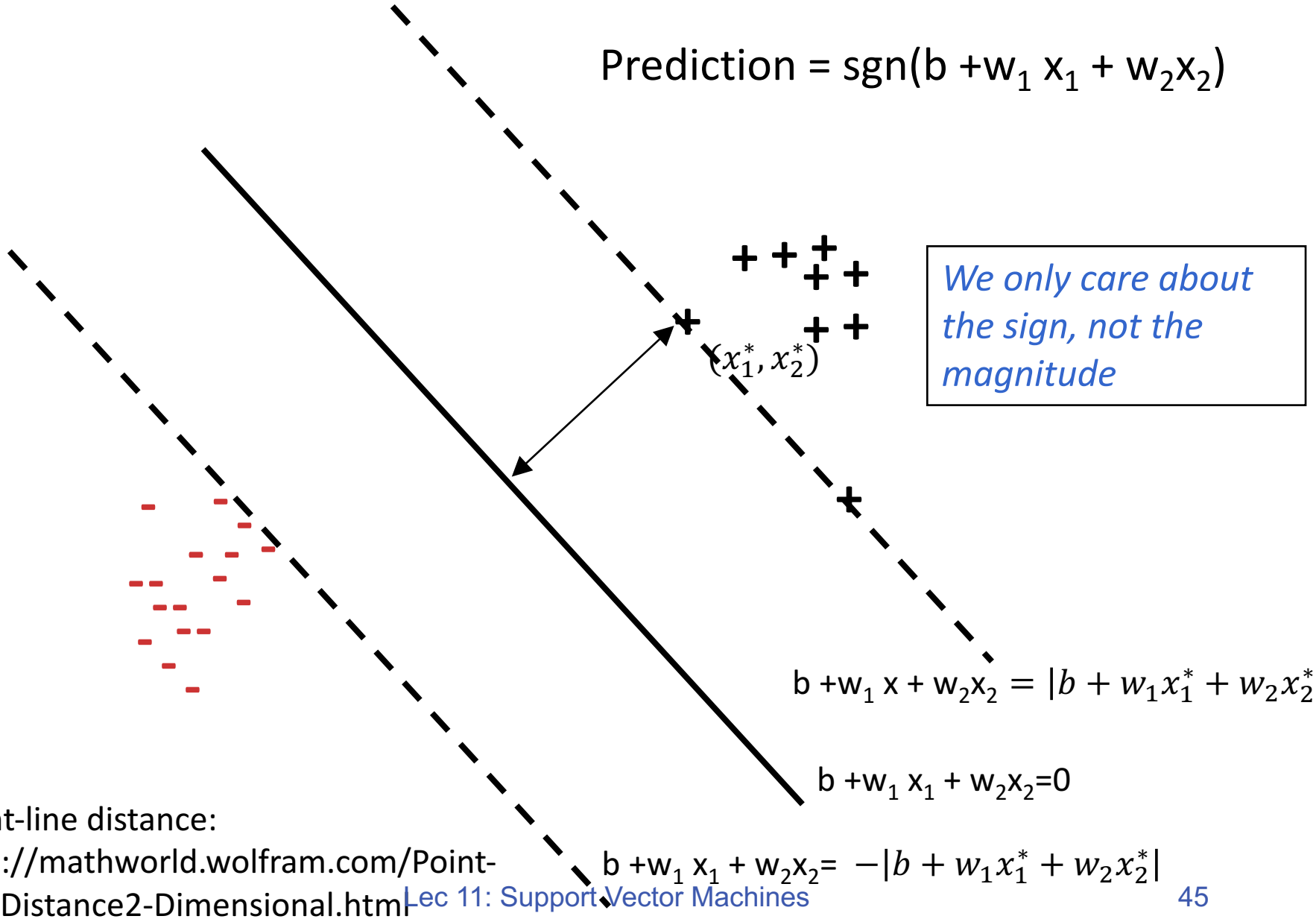
$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right)}{||\mathbf{w}||}$$

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

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

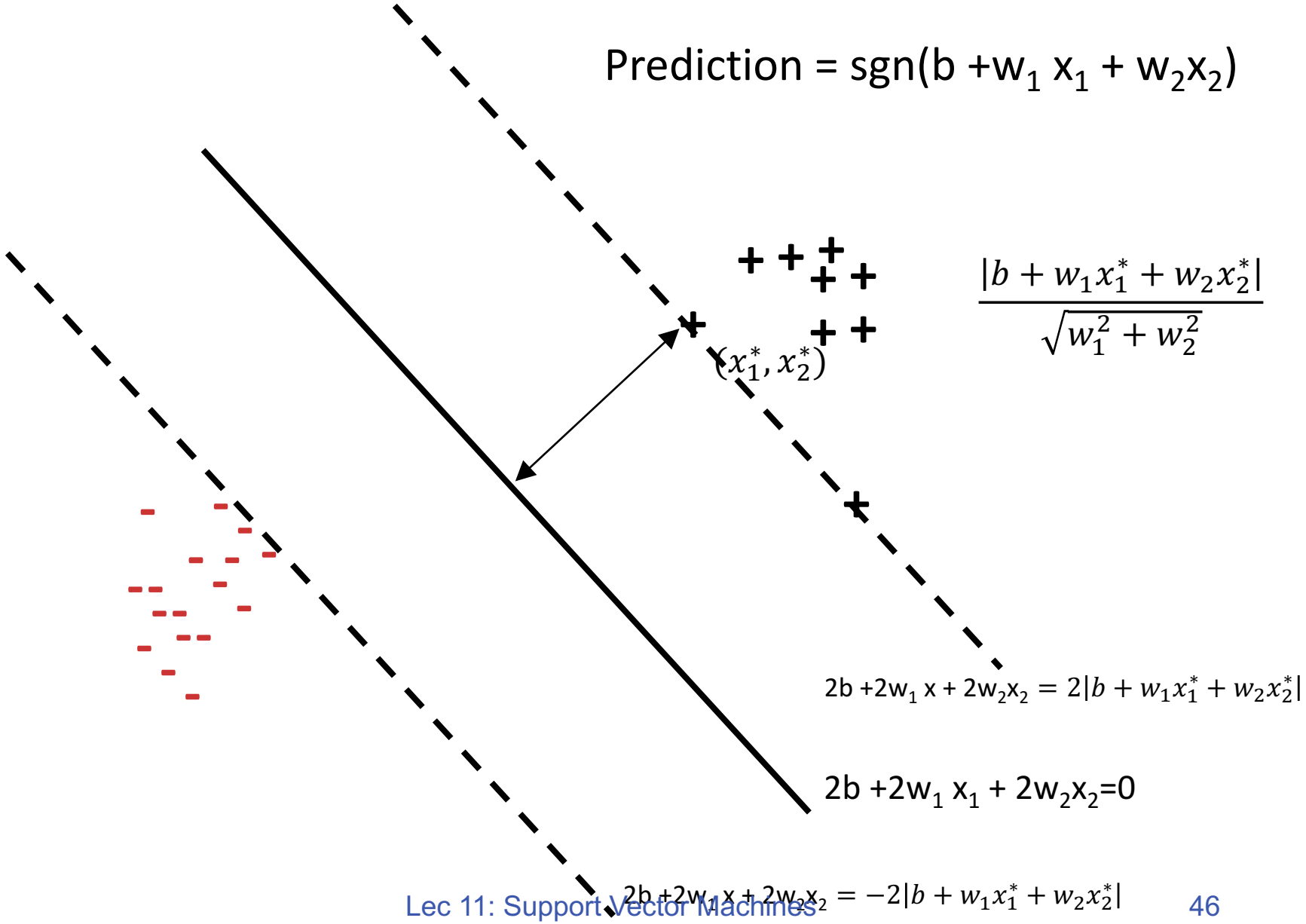


Recall: The geometry of a linear classifier



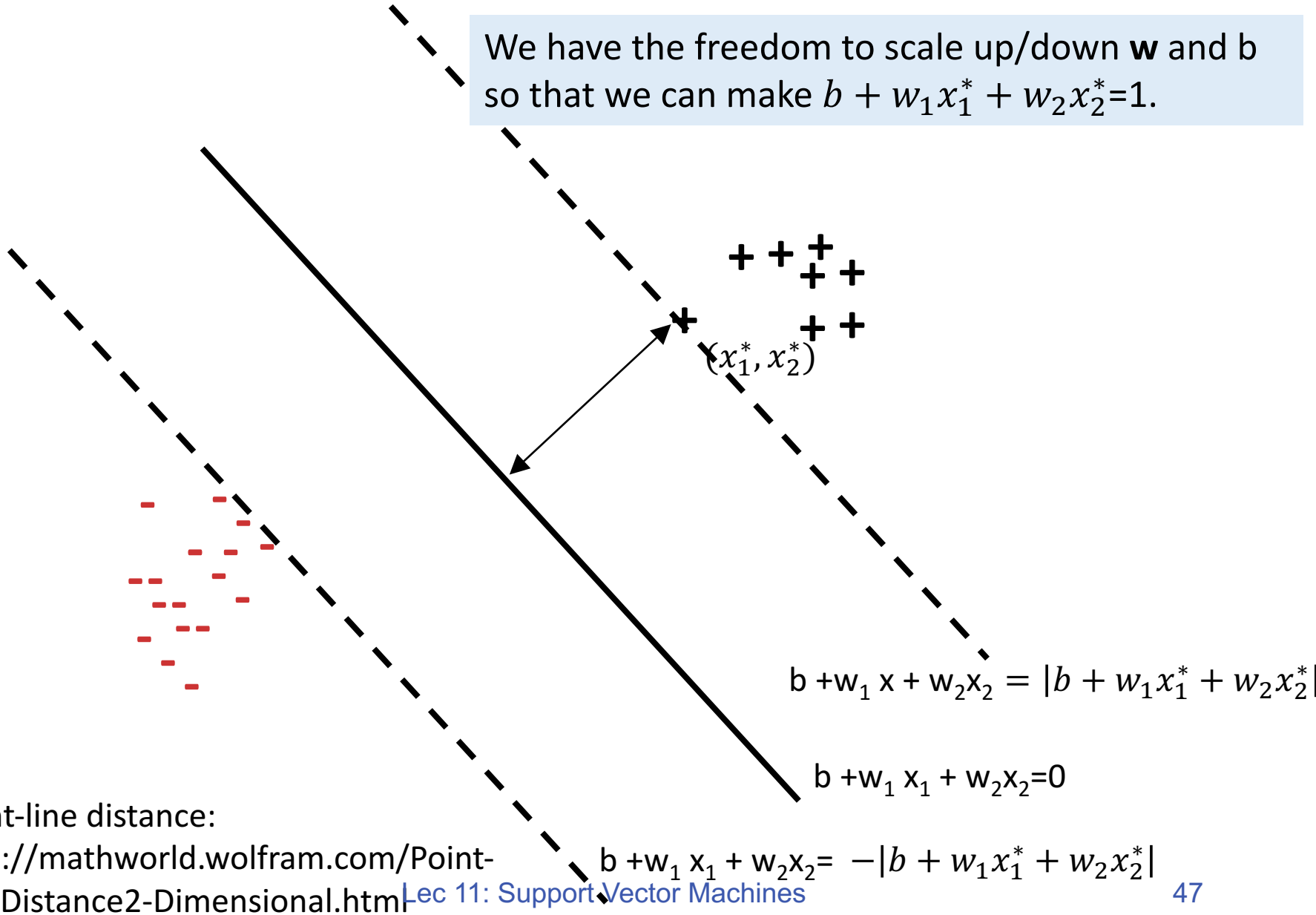
Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Recall: The geometry of a linear classifier



Recall: The geometry of a linear classifier

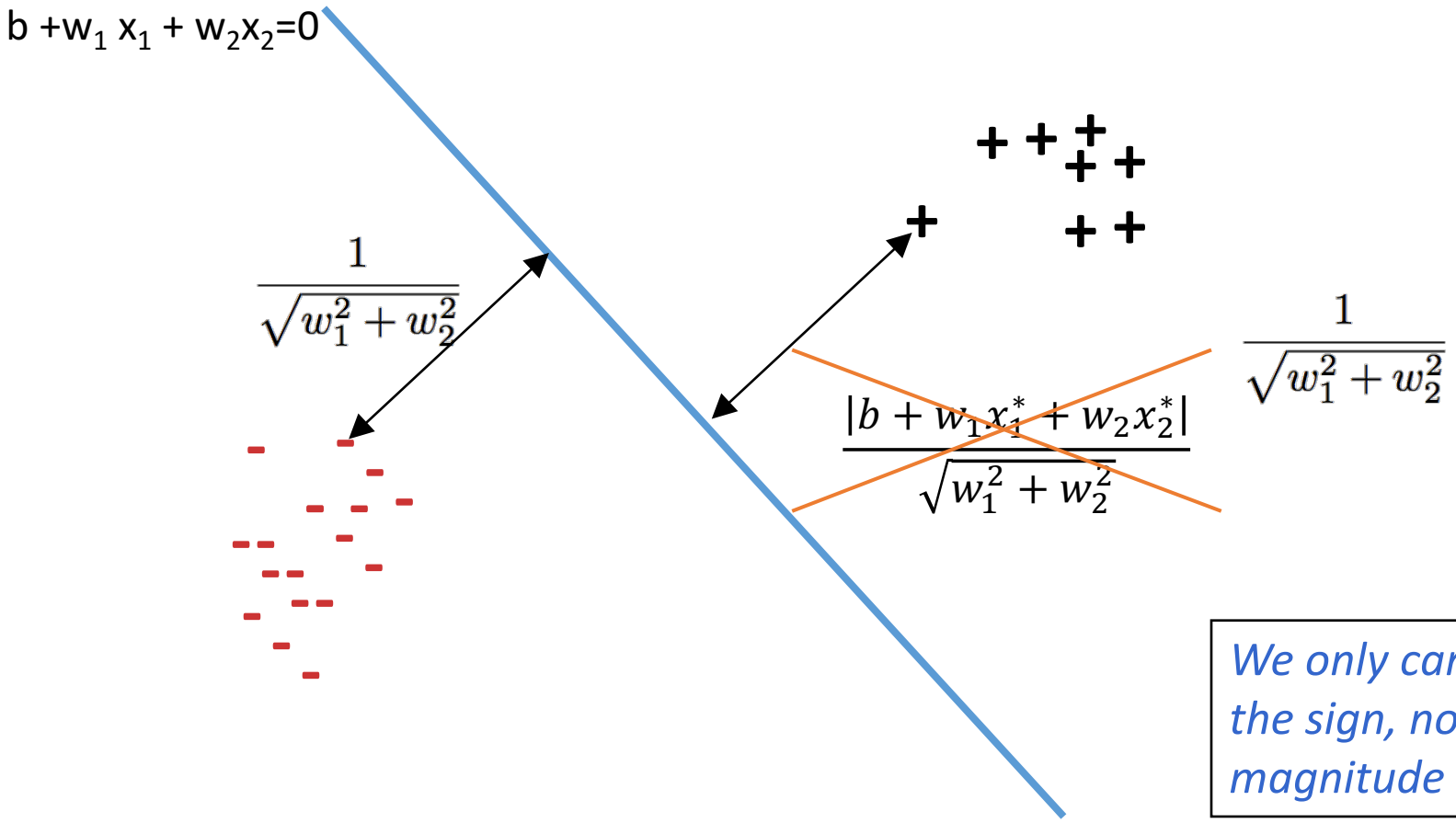
We have the freedom to scale up/down **w** and b so that we can make $b + w_1 x_1^* + w_2 x_2^*$=1.



Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Recall: The geometry of a linear classifier

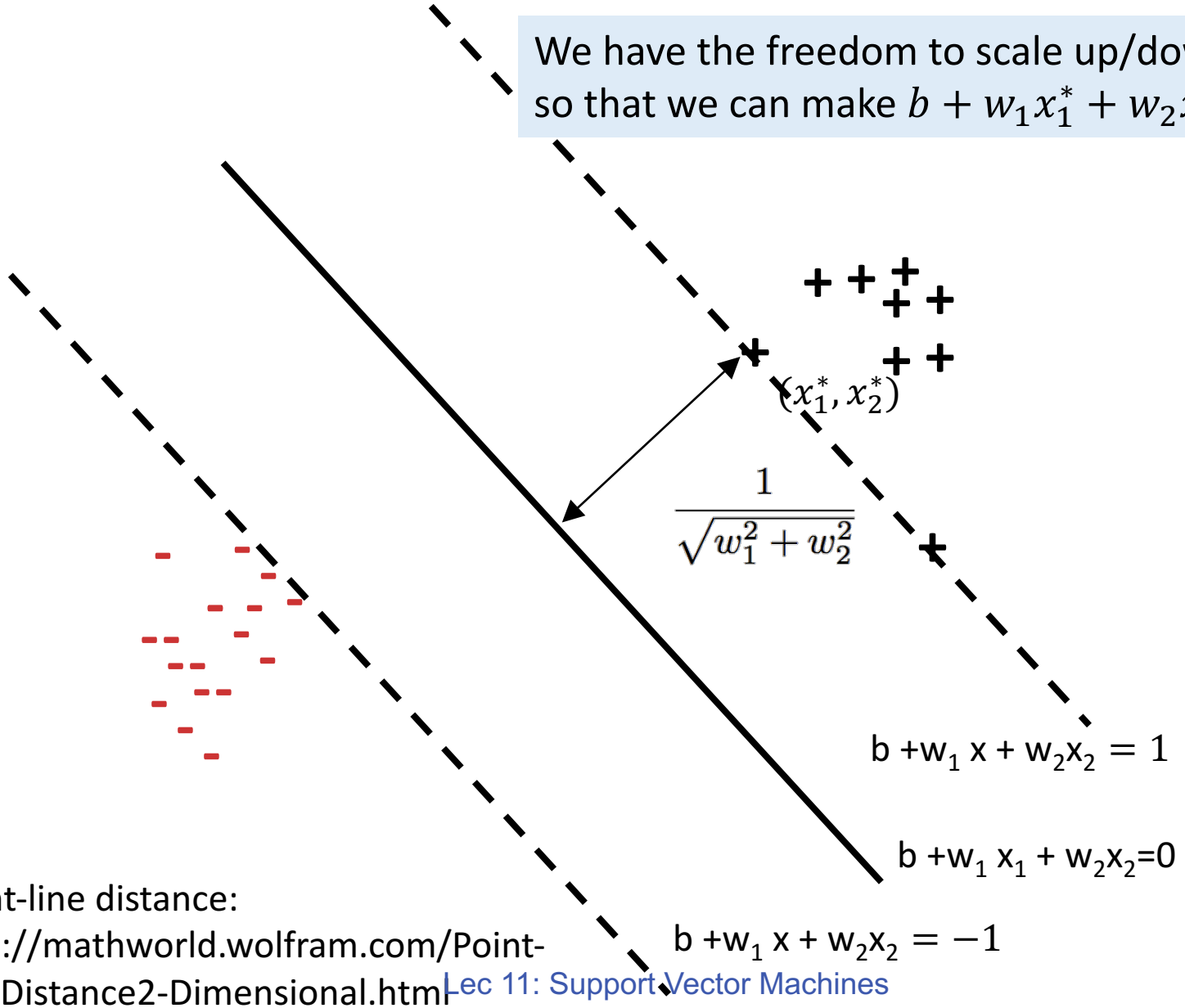
Prediction = sgn(b +w_1 x_1 + w_2x_2)



We only care about the sign, not the magnitude

Recall: The geometry of a linear classifier

We have the freedom to scale up/down **w** and b so that we can make $b + w_1 x_1^* + w_2 x_2^* = 1$.



Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Maximizing margin

- Margin = distance of the closest point from the hyperplane

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

- We want $\max_{\mathbf{w}} \gamma$
- We only care about the sign of **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 **w** to adjust itself

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

Max-margin classifiers

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

- Learning problem:

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

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

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

Max-margin classifiers

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

- Learning problem:

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

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

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

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

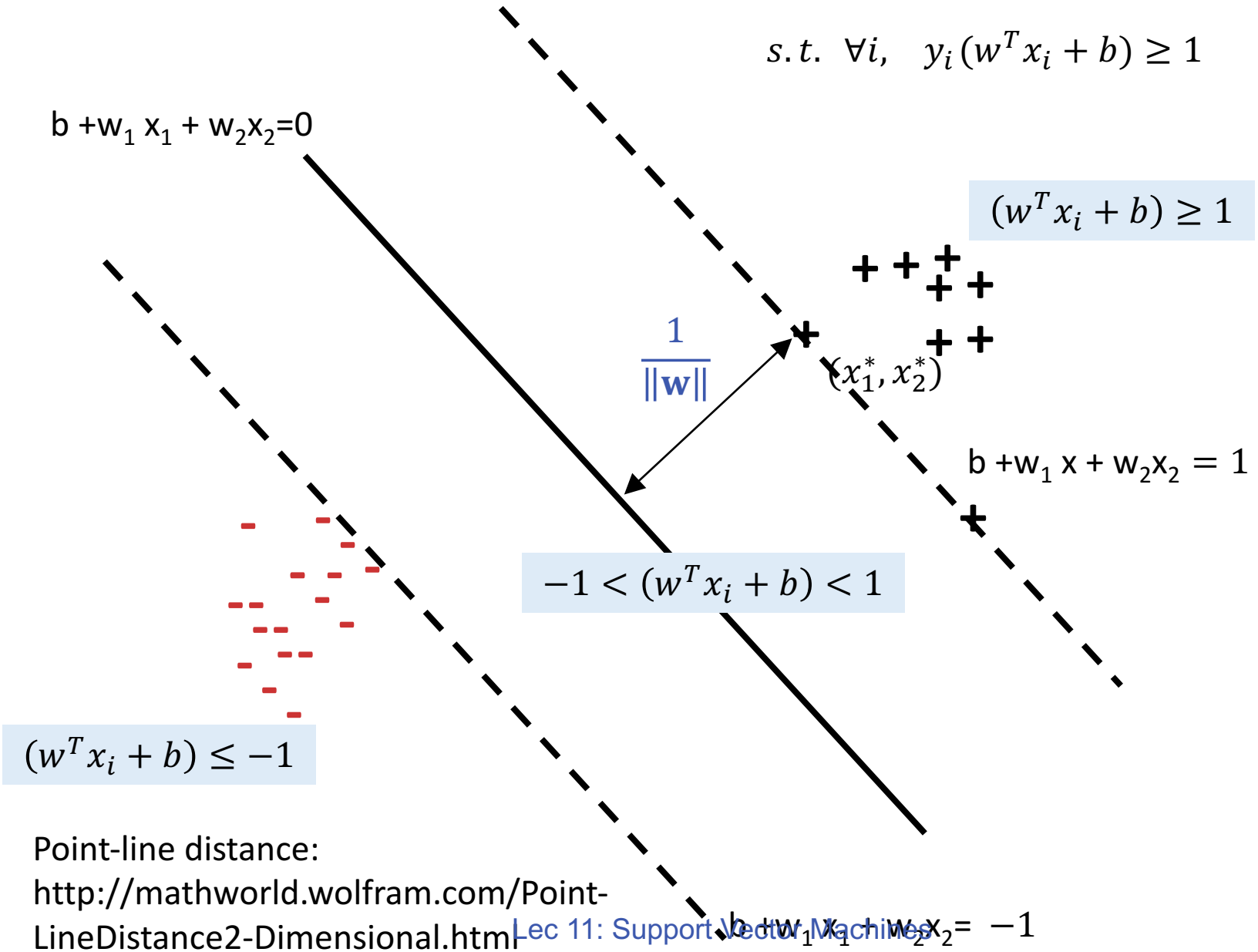
- This is called the "hard" Support Vector Machine

We will look at how to solve this optimization problem later

Hard SVM

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

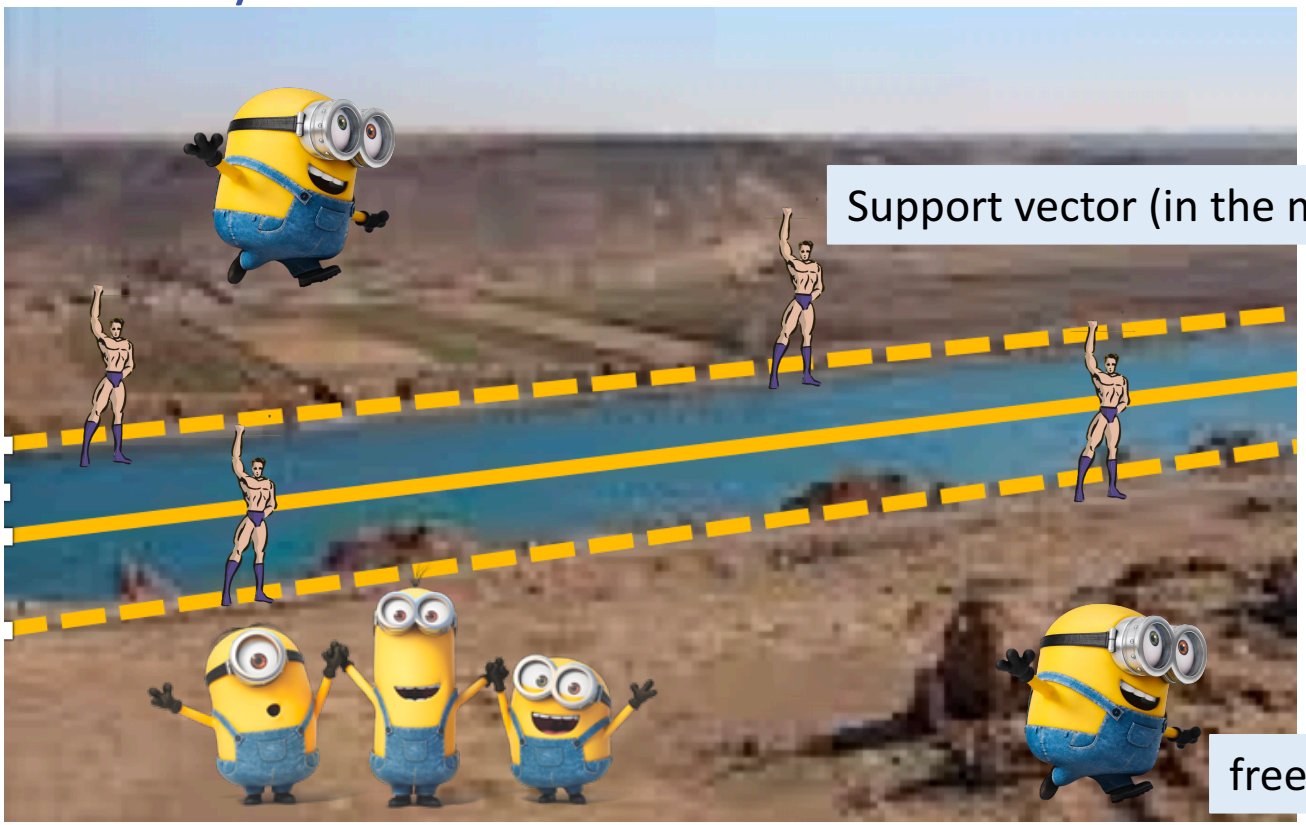
$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$



Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

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

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

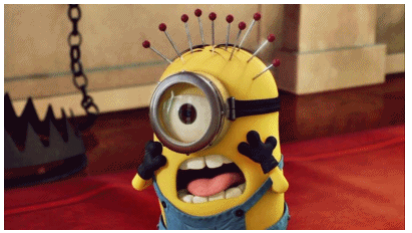
Maximize margin

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

Every example has an functional margin of at least 1

- This is a constrained optimization problem

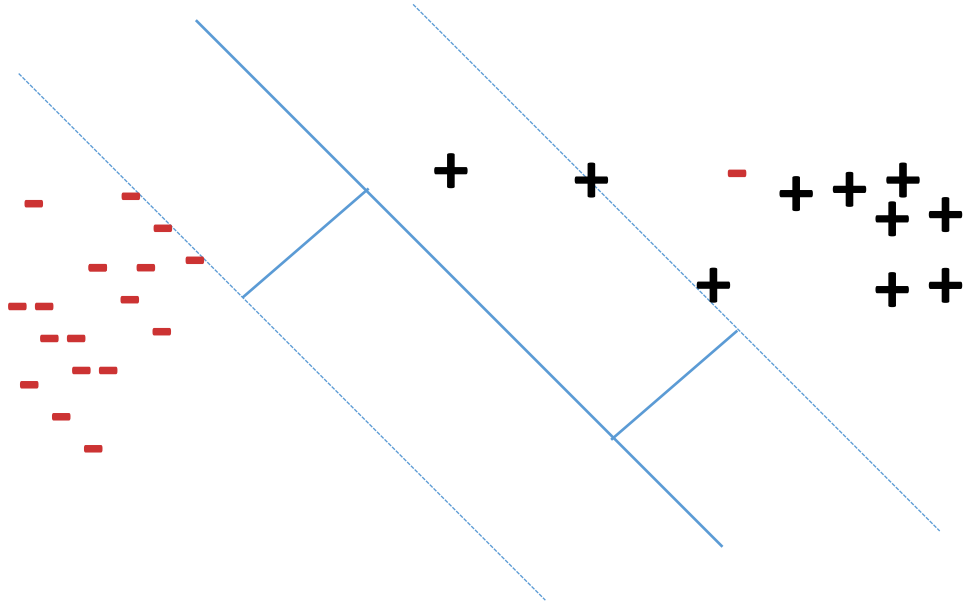
- If the data is not separable, there is no **w** that will classify the data
- Infeasible problem, no solution!



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

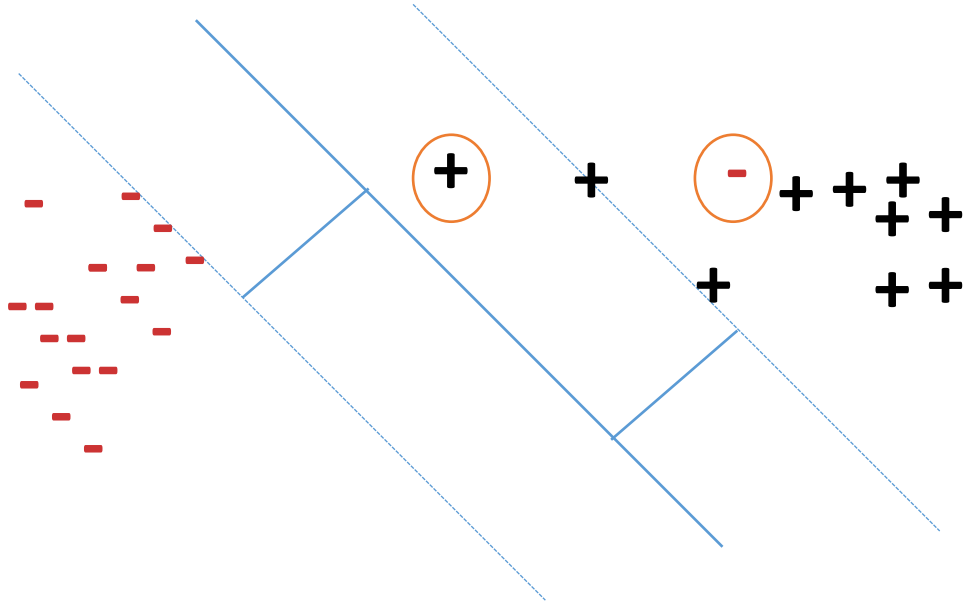
Dealing with non-separable data

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



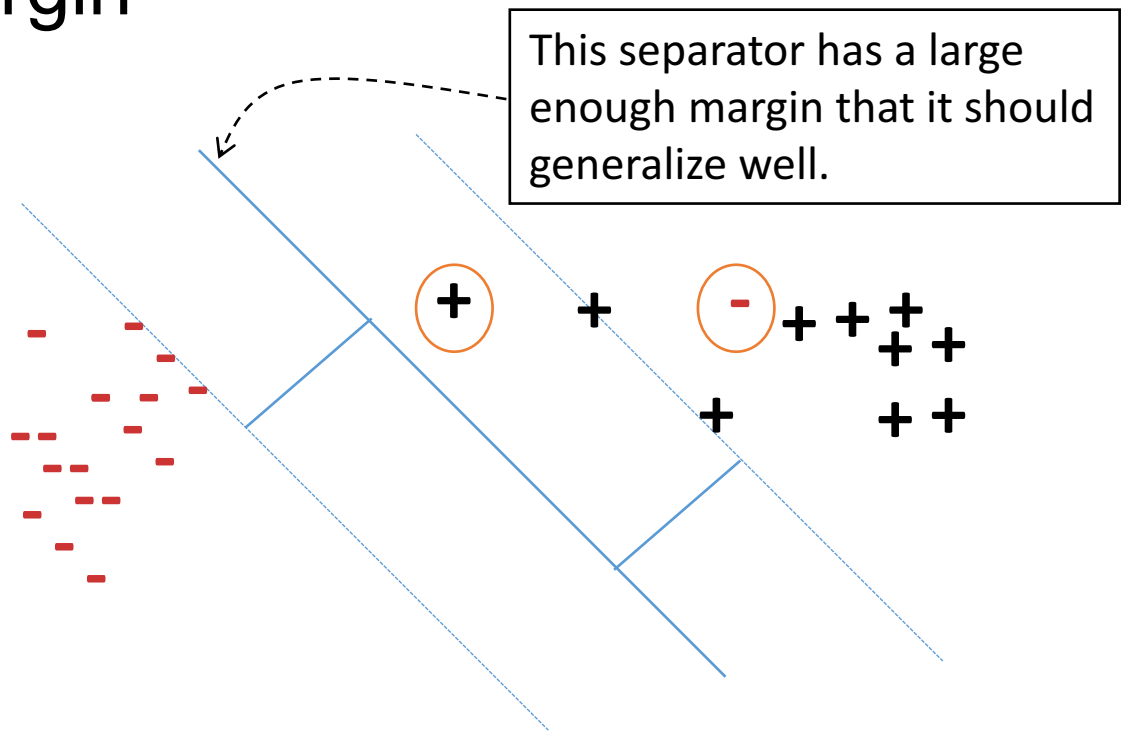
Dealing with non-separable data

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



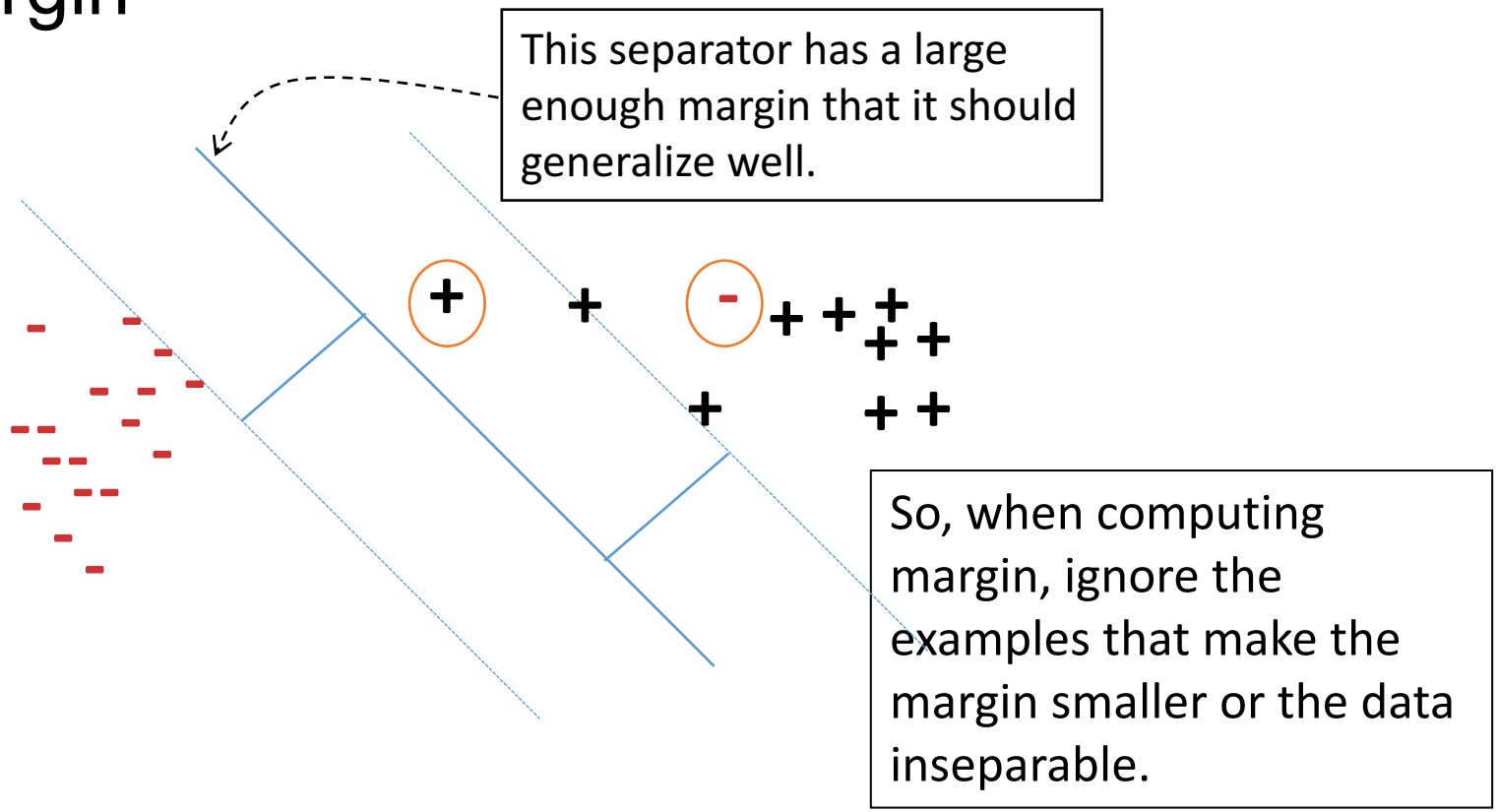
Dealing with non-separable data

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

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

Every example has an functional margin of at least 1

Soft SVM

- Hard SVM:

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

Maximize margin

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

Every example has an functional margin of at least 1

- Introduce one *slack variable* ξ_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:

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

Maximize margin

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

Every example has an functional margin of at least 1

- Introduce one *slack variable* ξ_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:

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

Maximize margin

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

Every example has an functional margin of at least 1

- New optimization problem for learning

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

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

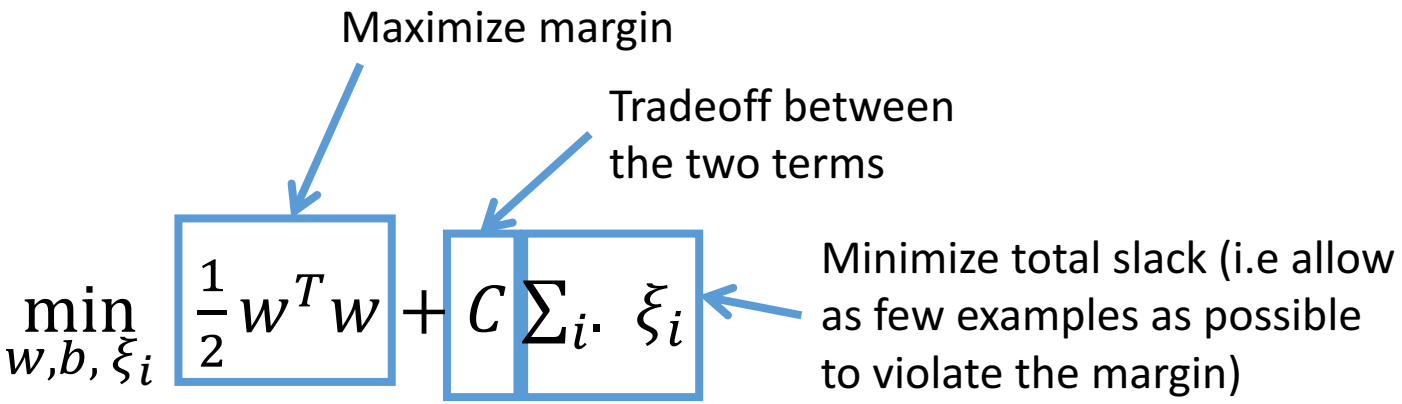
C is the hyper-parameter

Soft SVM

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

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

Soft SVM

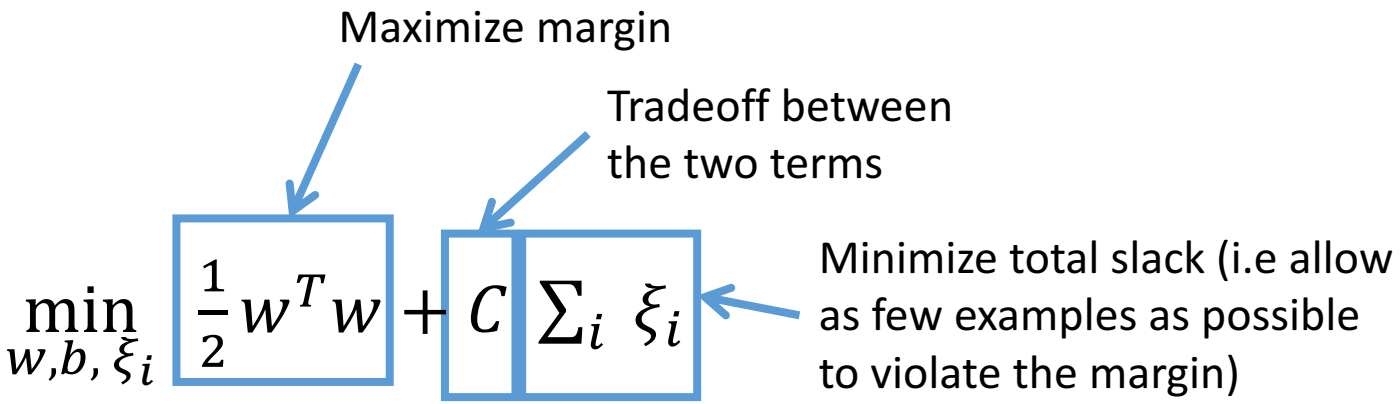


$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

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

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

Soft SVM



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

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

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

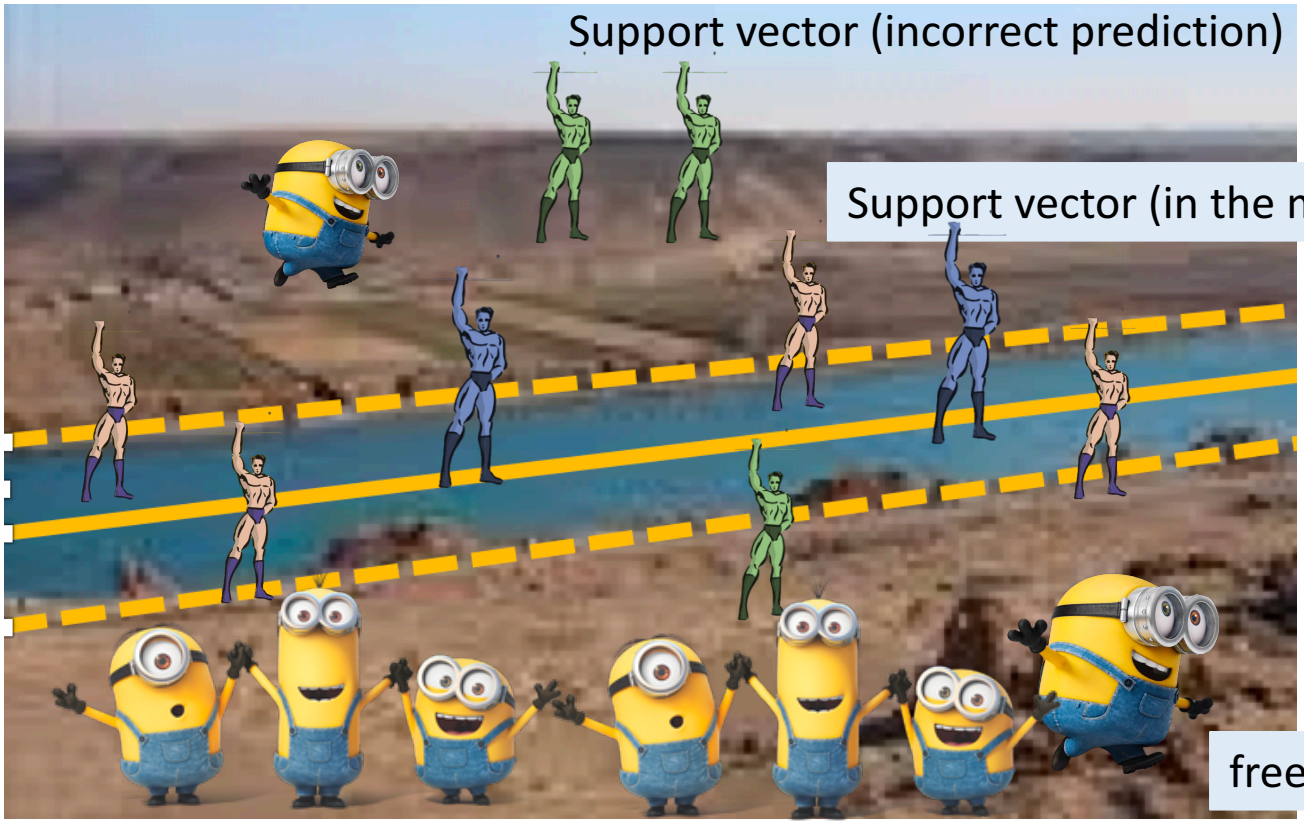
We can consider three cases

- 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 = $1 - y_i(w^T x_i + b)$

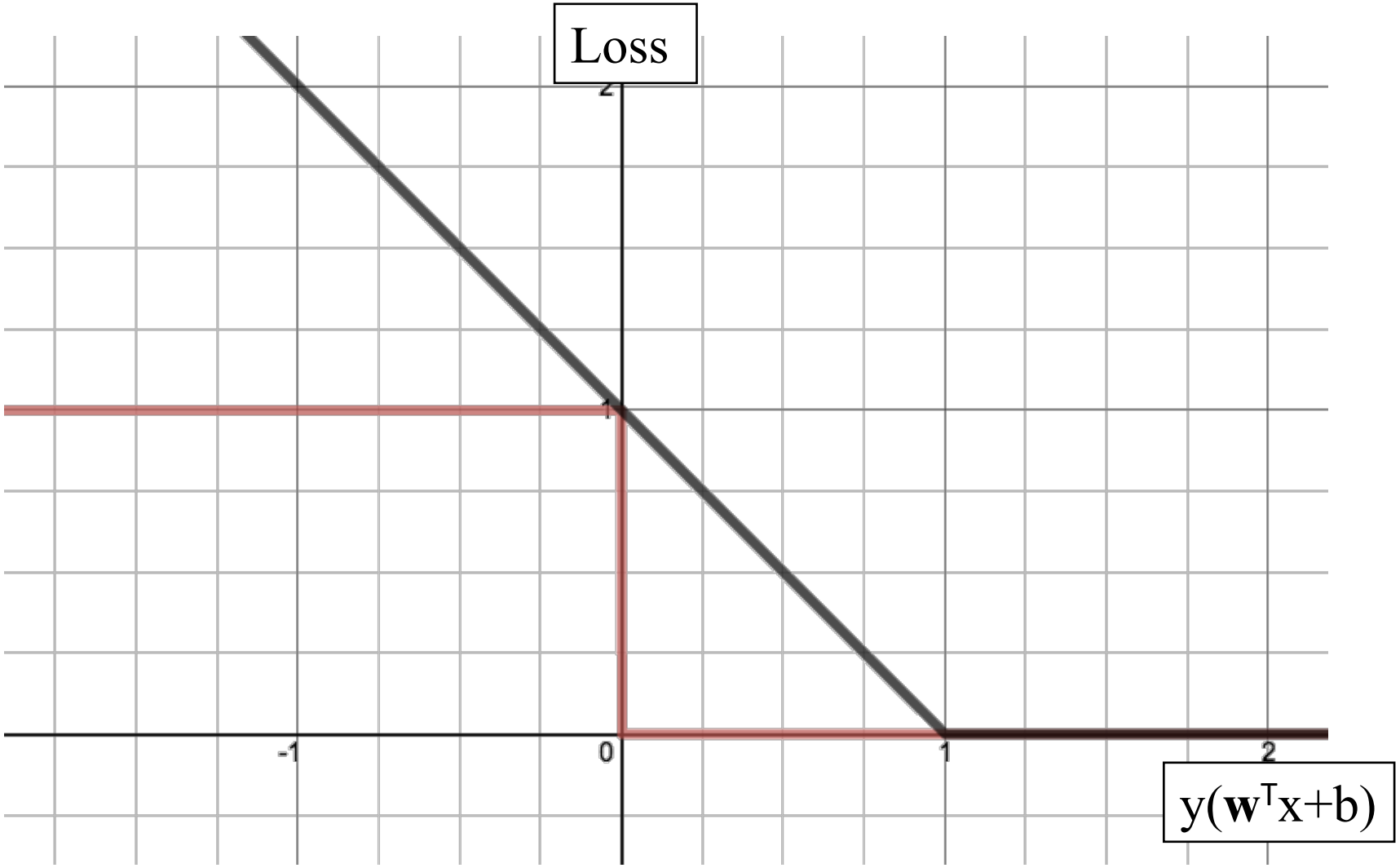
This is the hinge loss function

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

Why it called support vector machines?

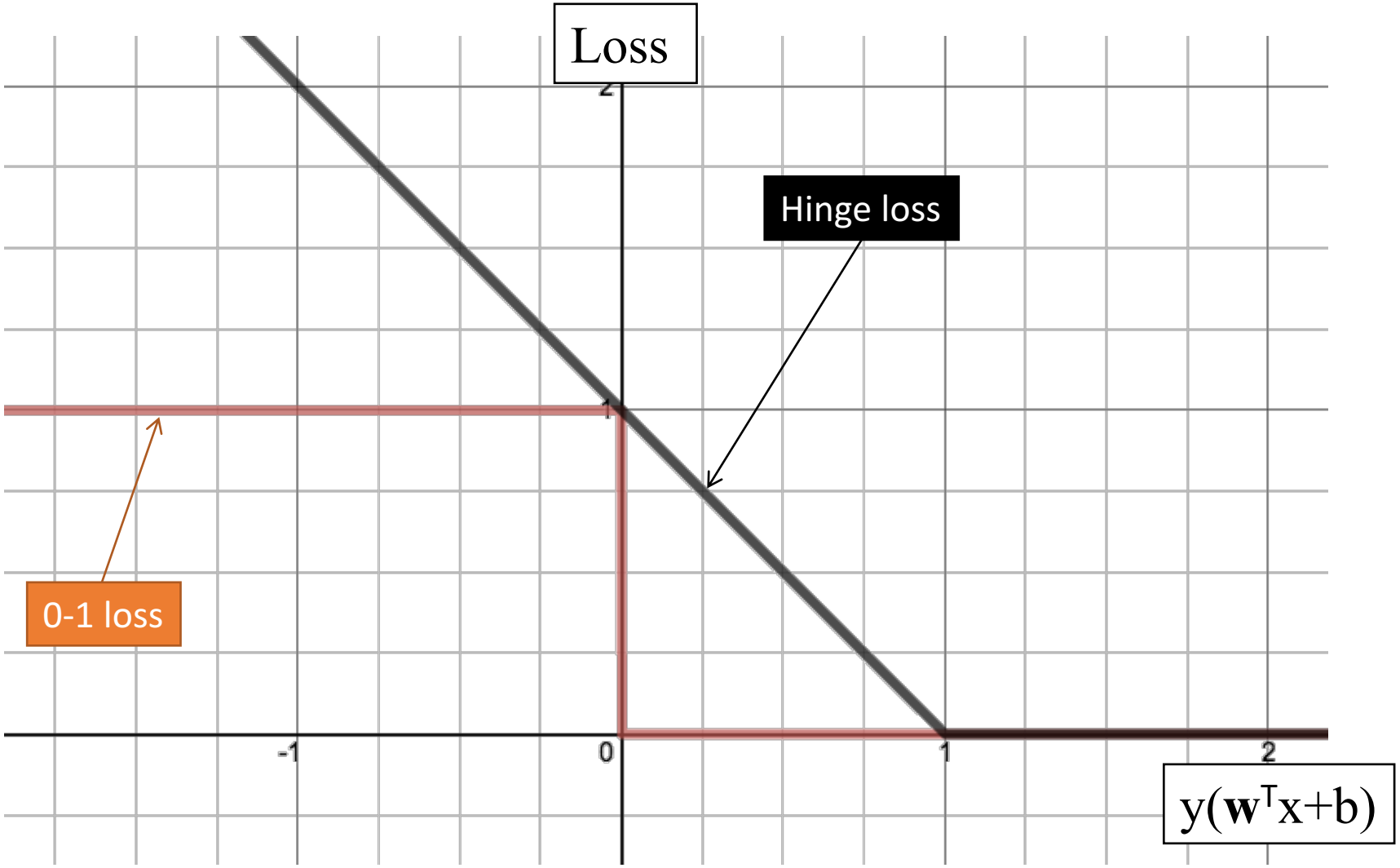


The Hinge Loss



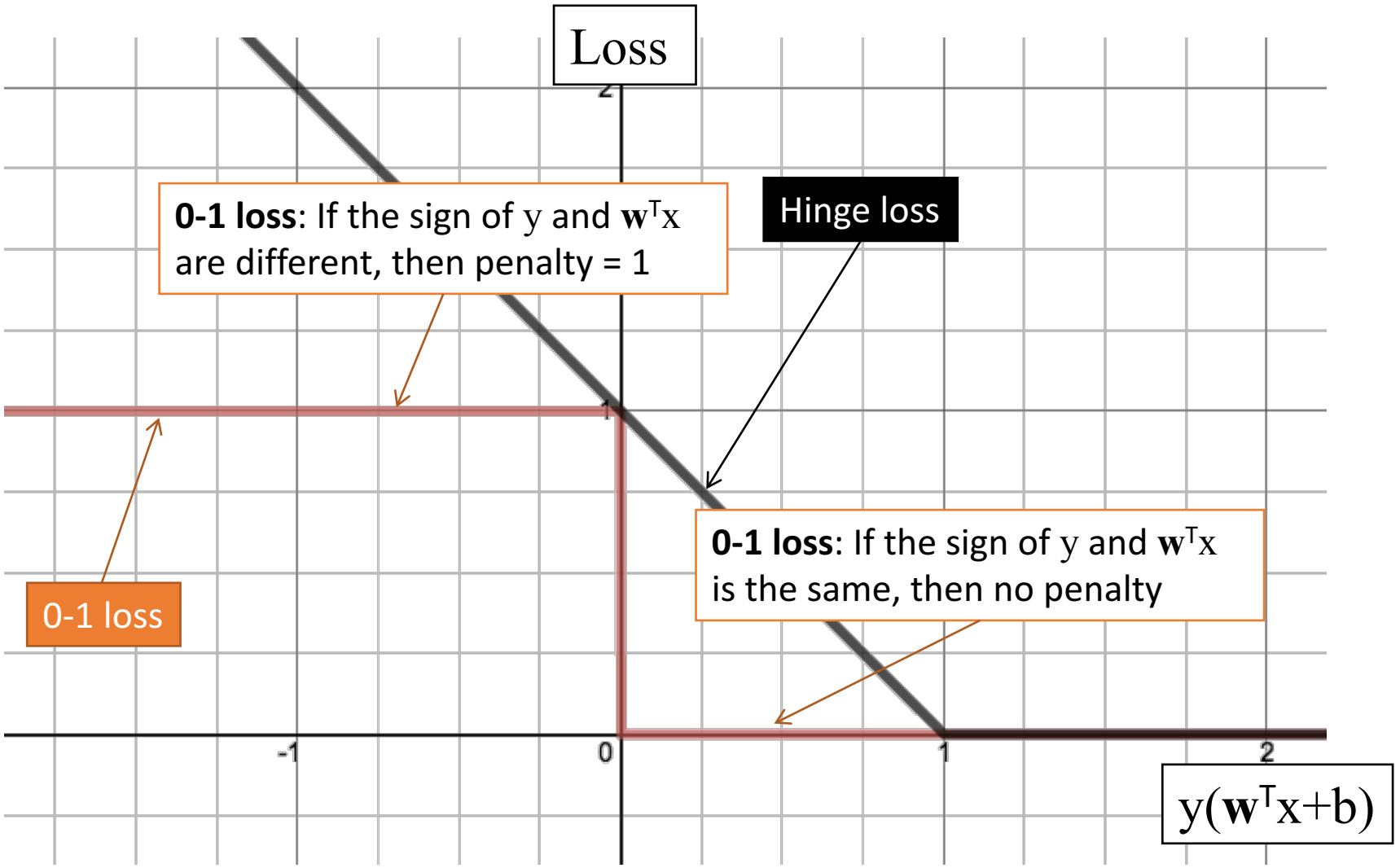
The Hinge Loss

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



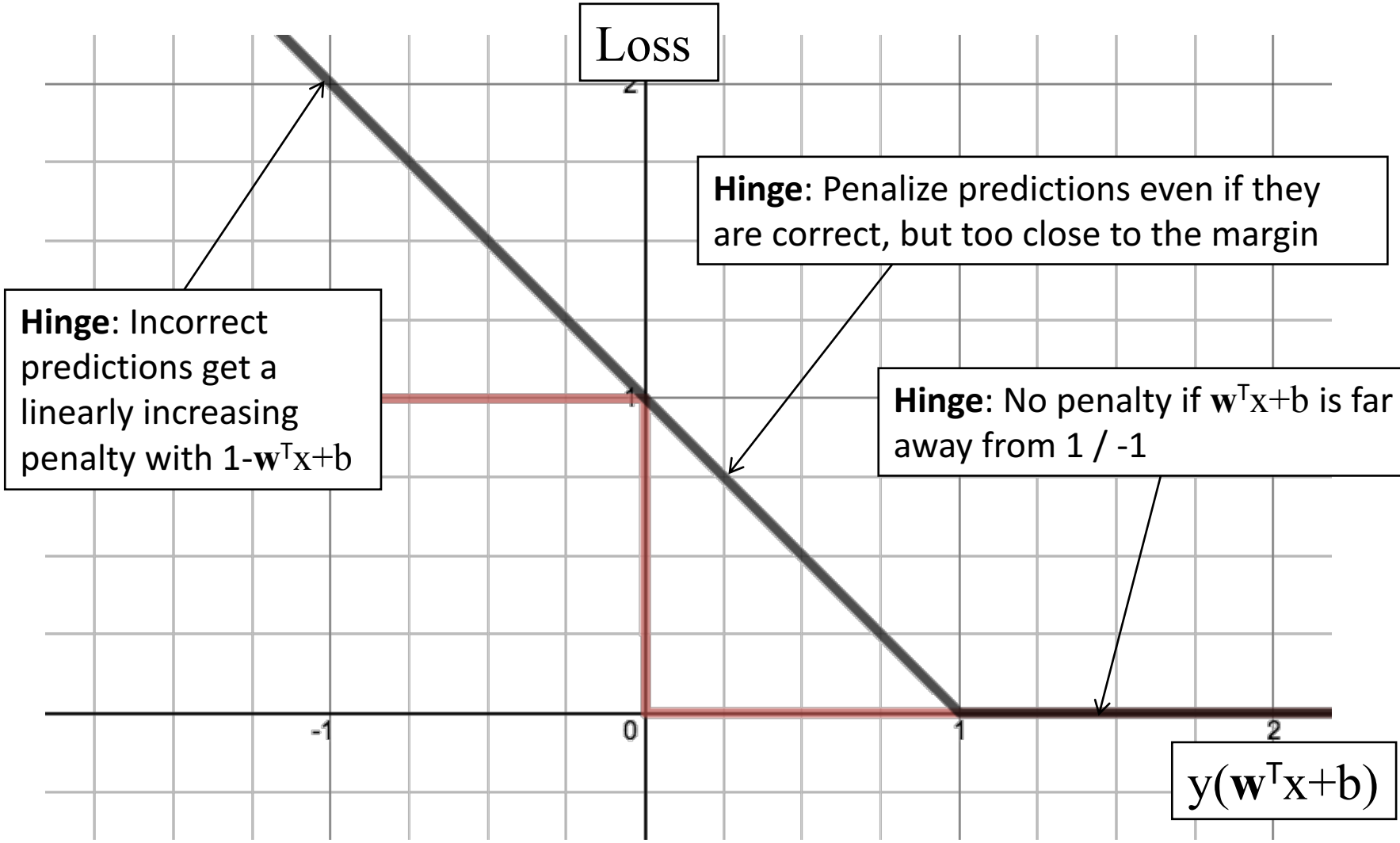
The Hinge Loss

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



The Hinge Loss

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



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 $\mathbf{w}$

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

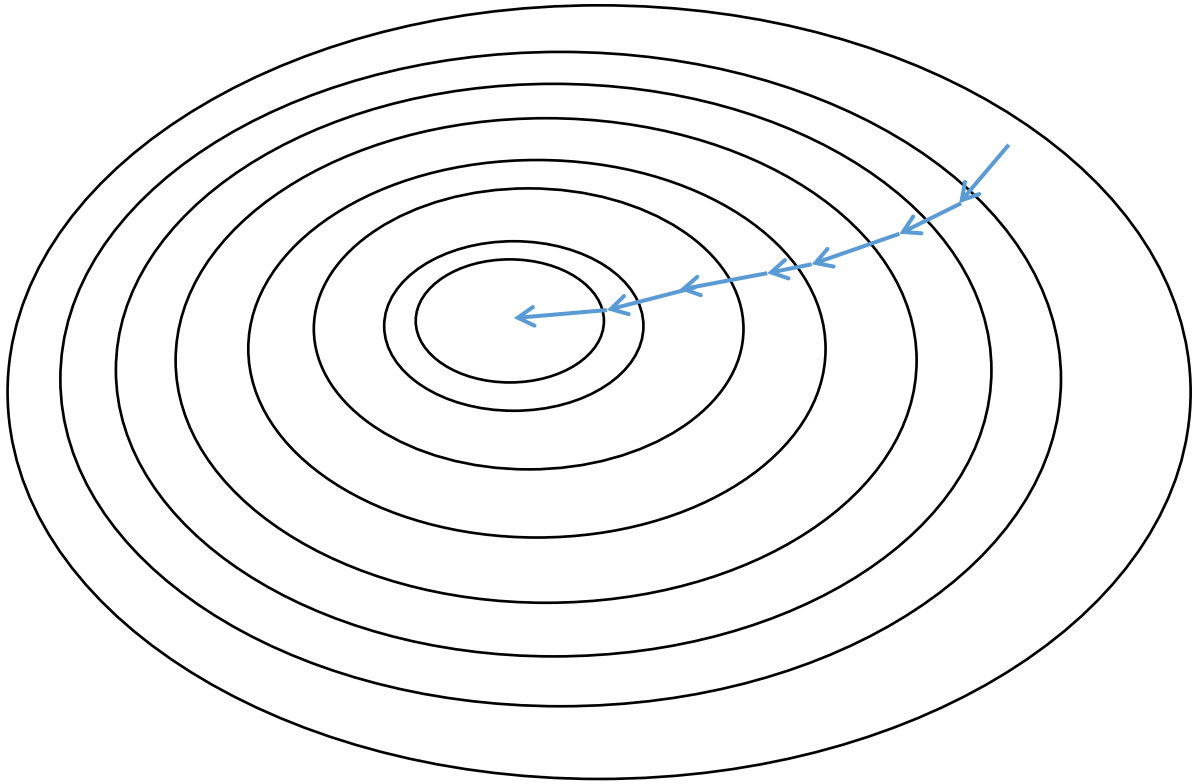
Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

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

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

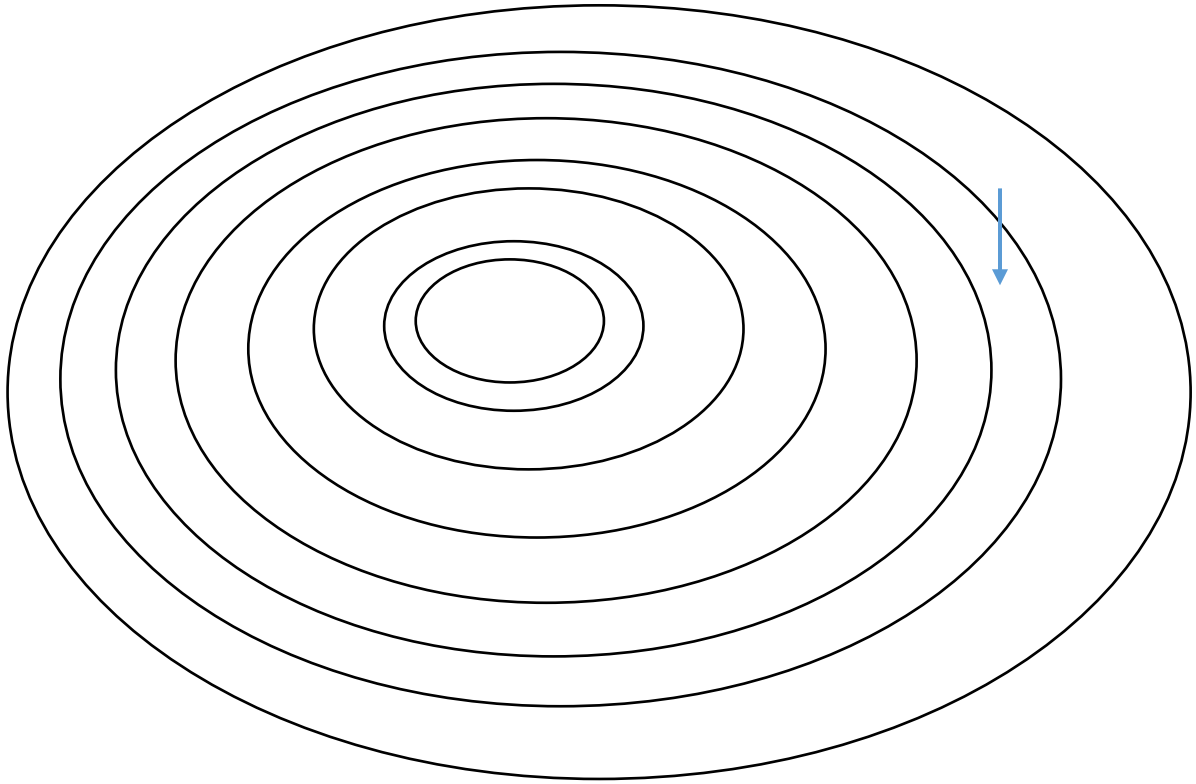
We will see more example later in this lecture

Gradient Descent vs SGD



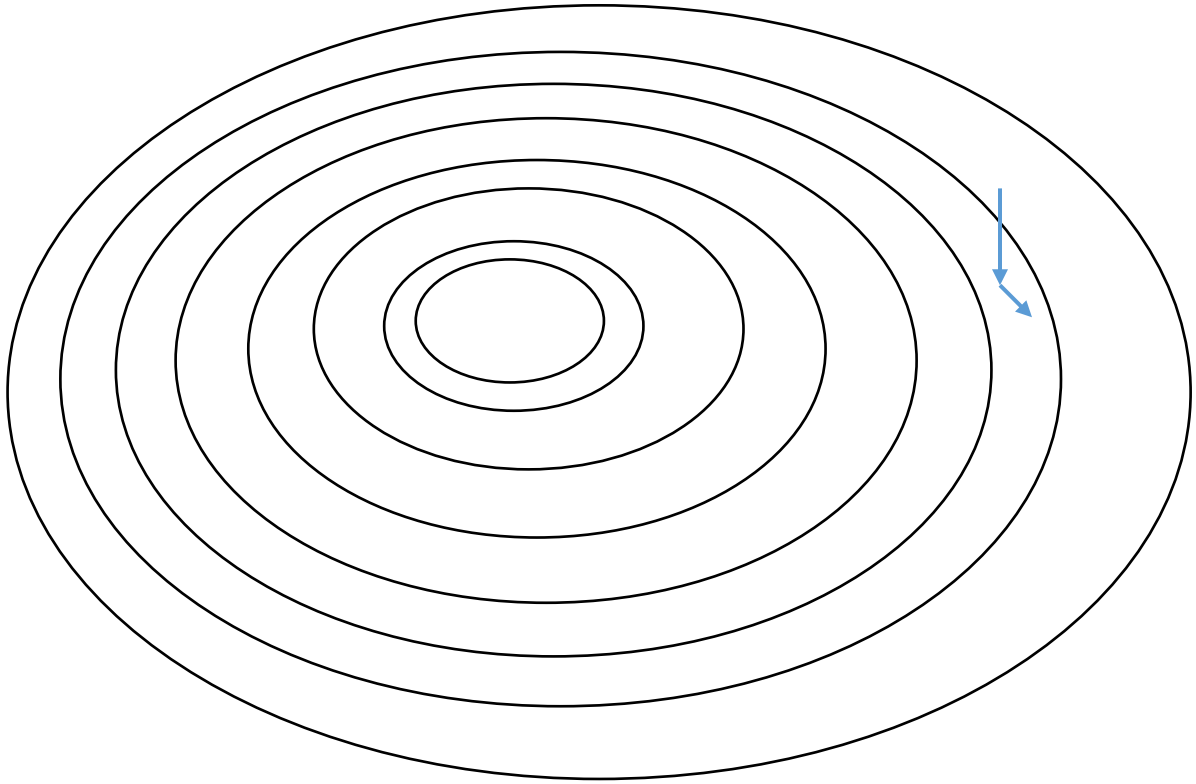
Gradient descent

Gradient Descent vs SGD



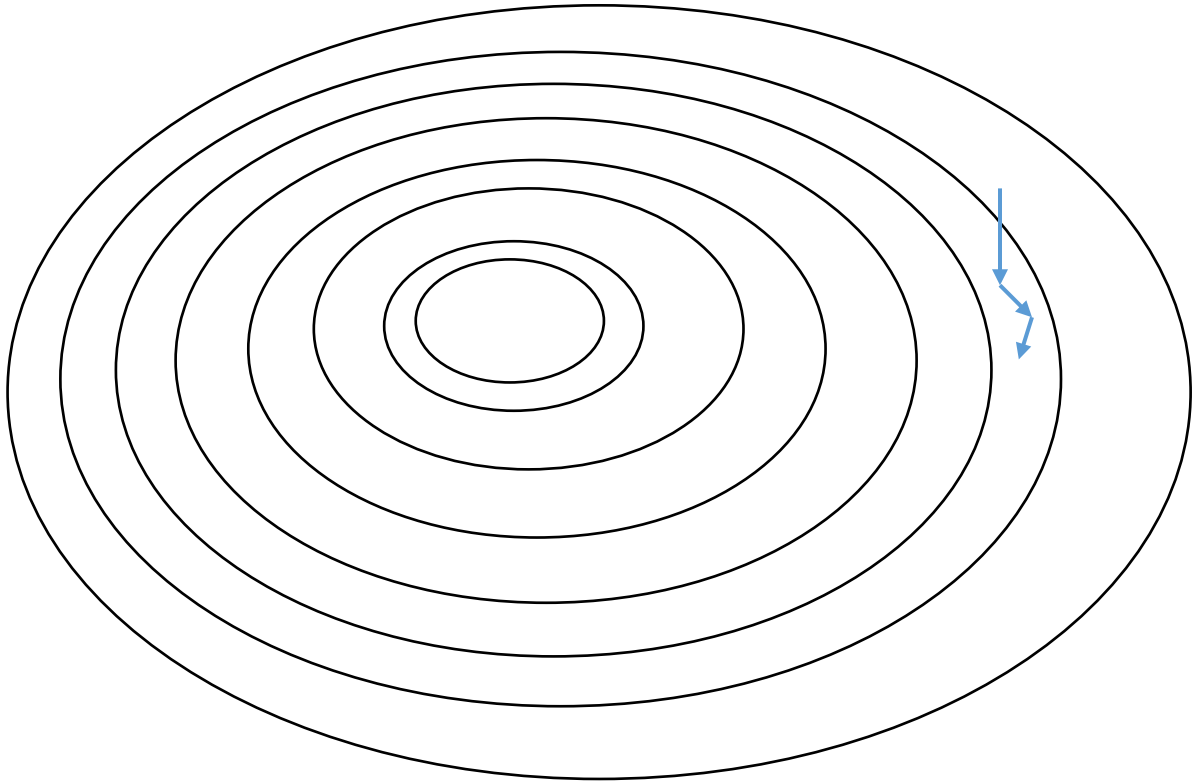
Stochastic Gradient descent

Gradient Descent vs SGD



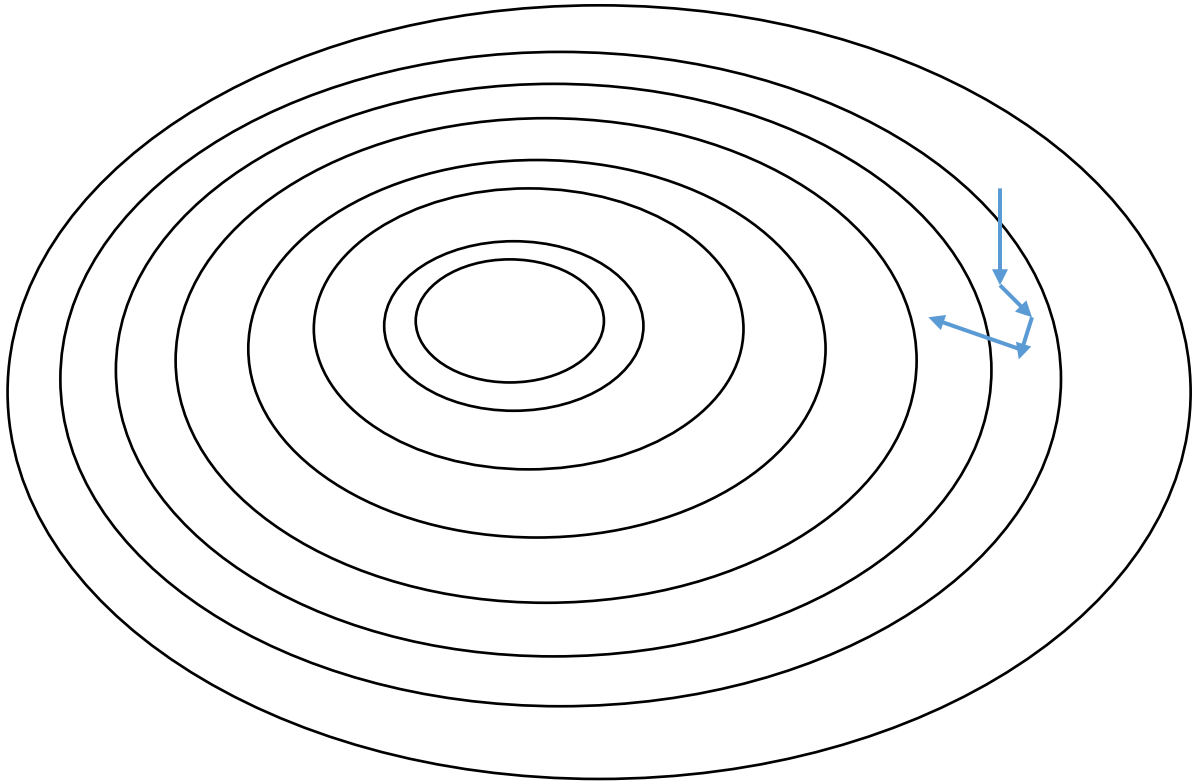
Stochastic Gradient descent

Gradient Descent vs SGD



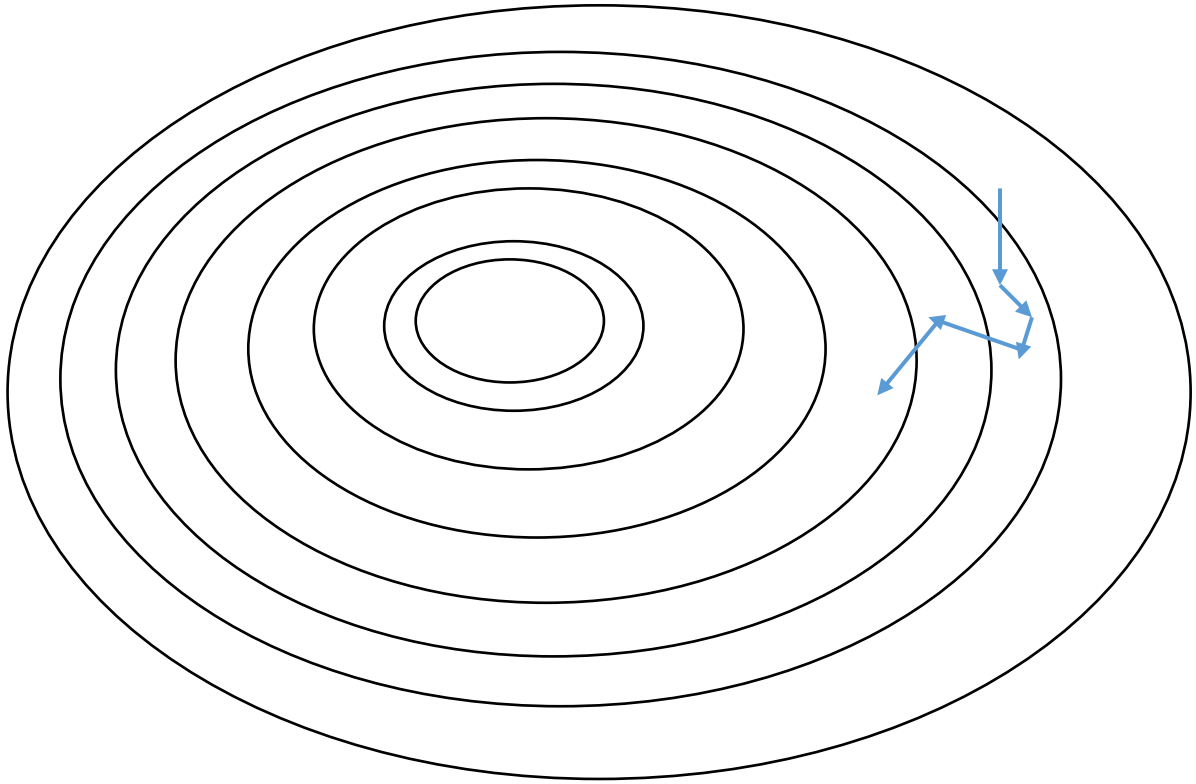
Stochastic Gradient descent

Gradient Descent vs SGD



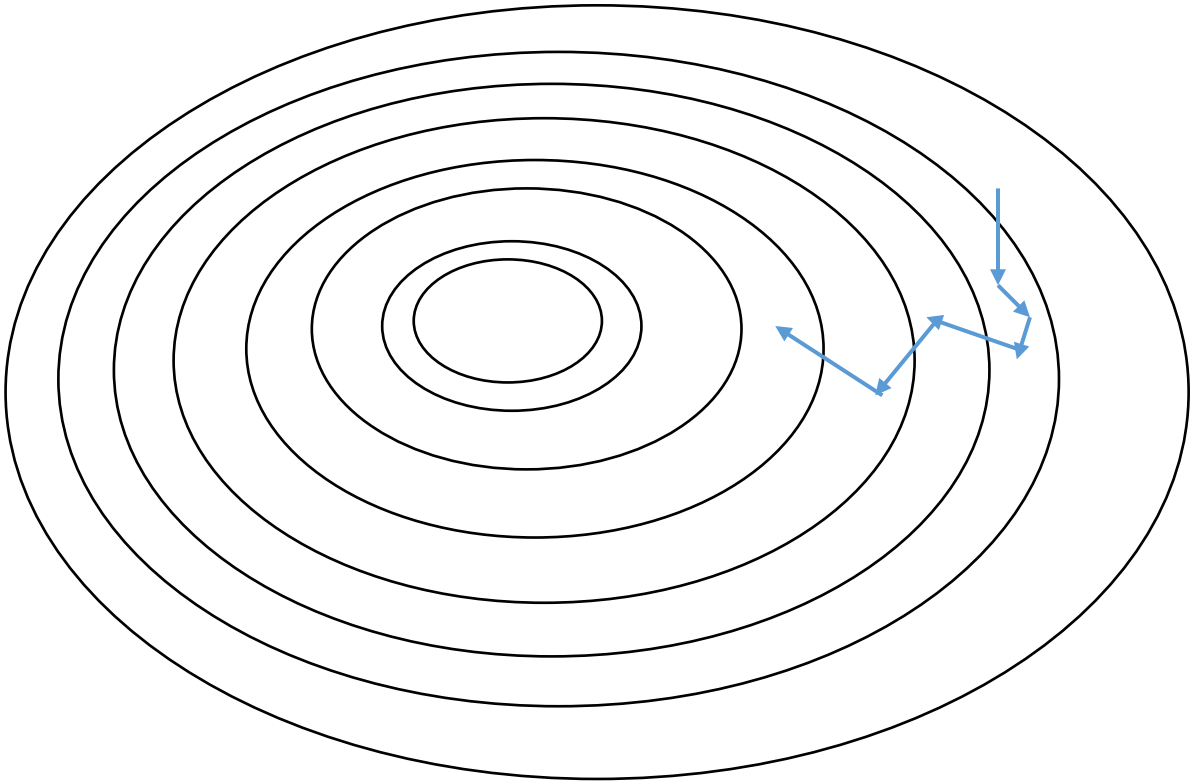
Stochastic Gradient descent

Gradient Descent vs SGD



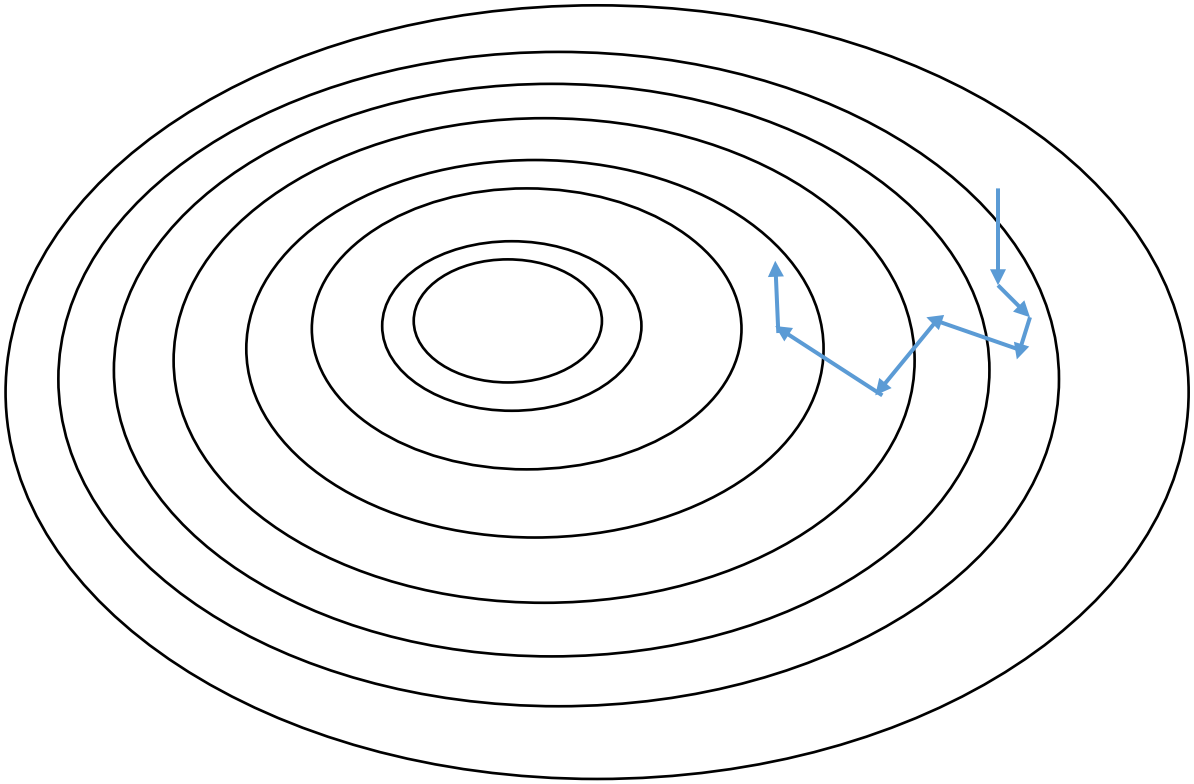
Stochastic Gradient descent

Gradient Descent vs SGD



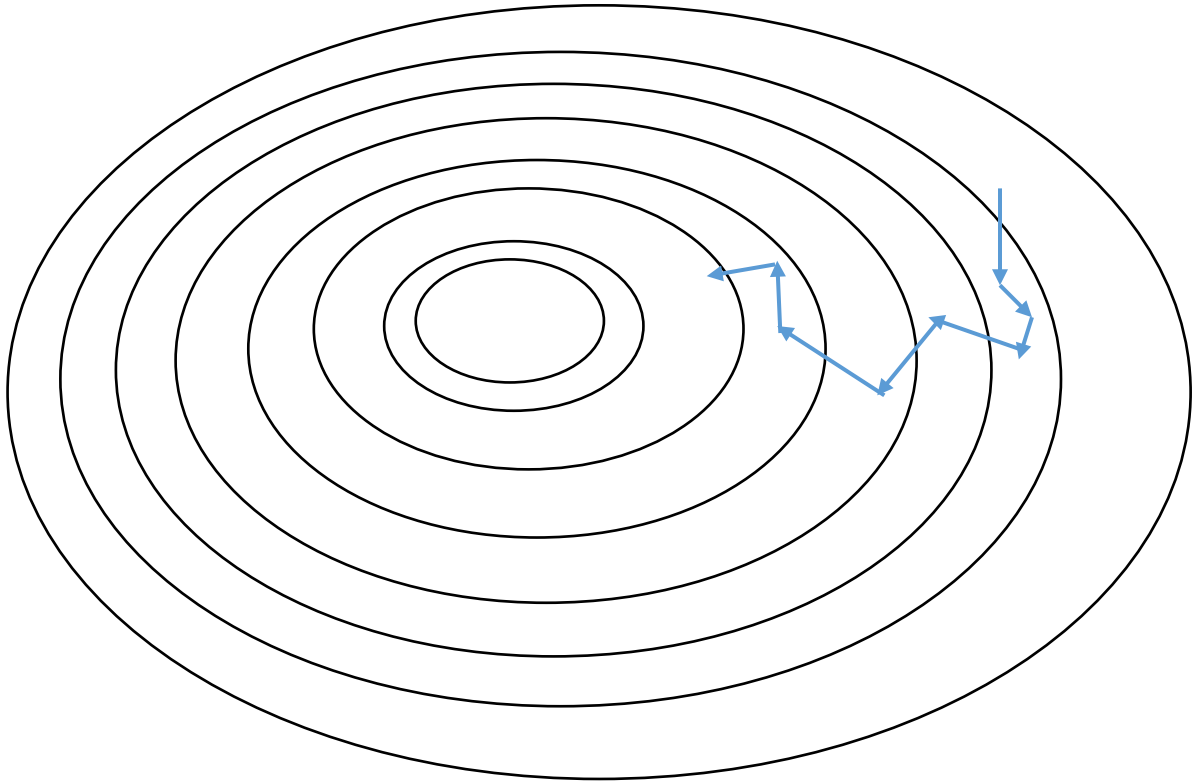
Stochastic Gradient descent

Gradient Descent vs SGD



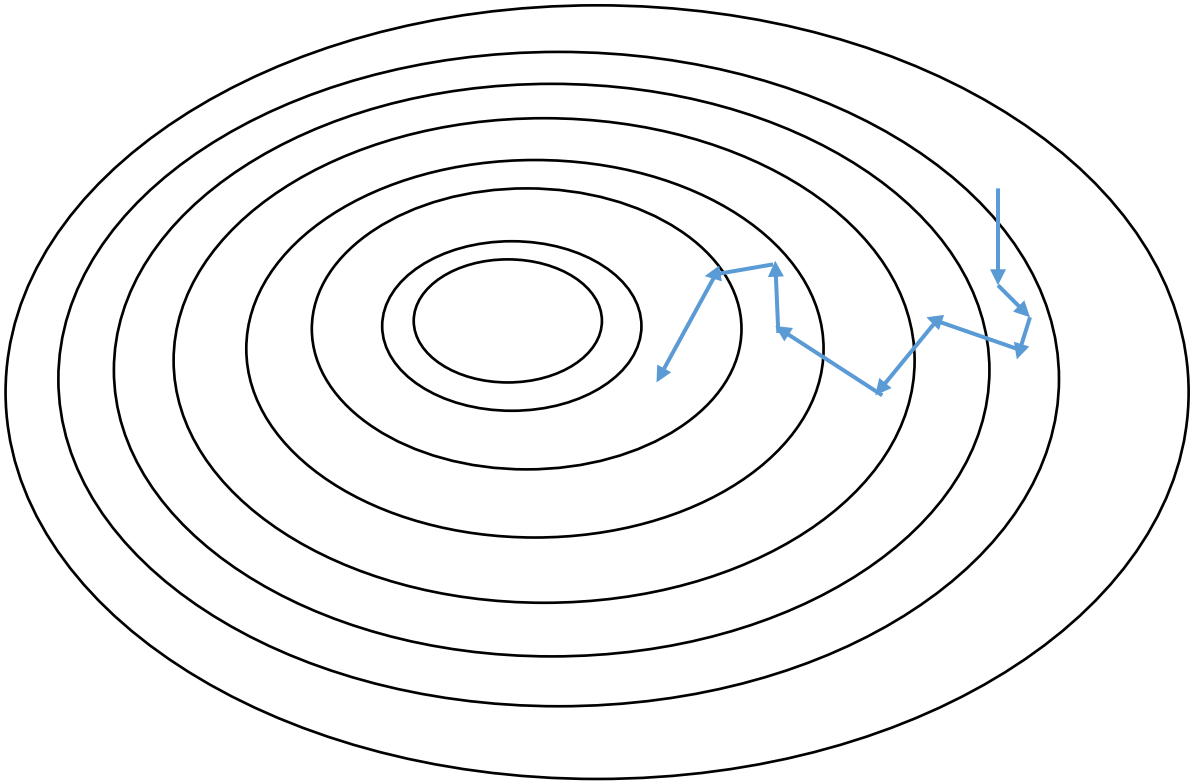
Stochastic Gradient descent

Gradient Descent vs SGD



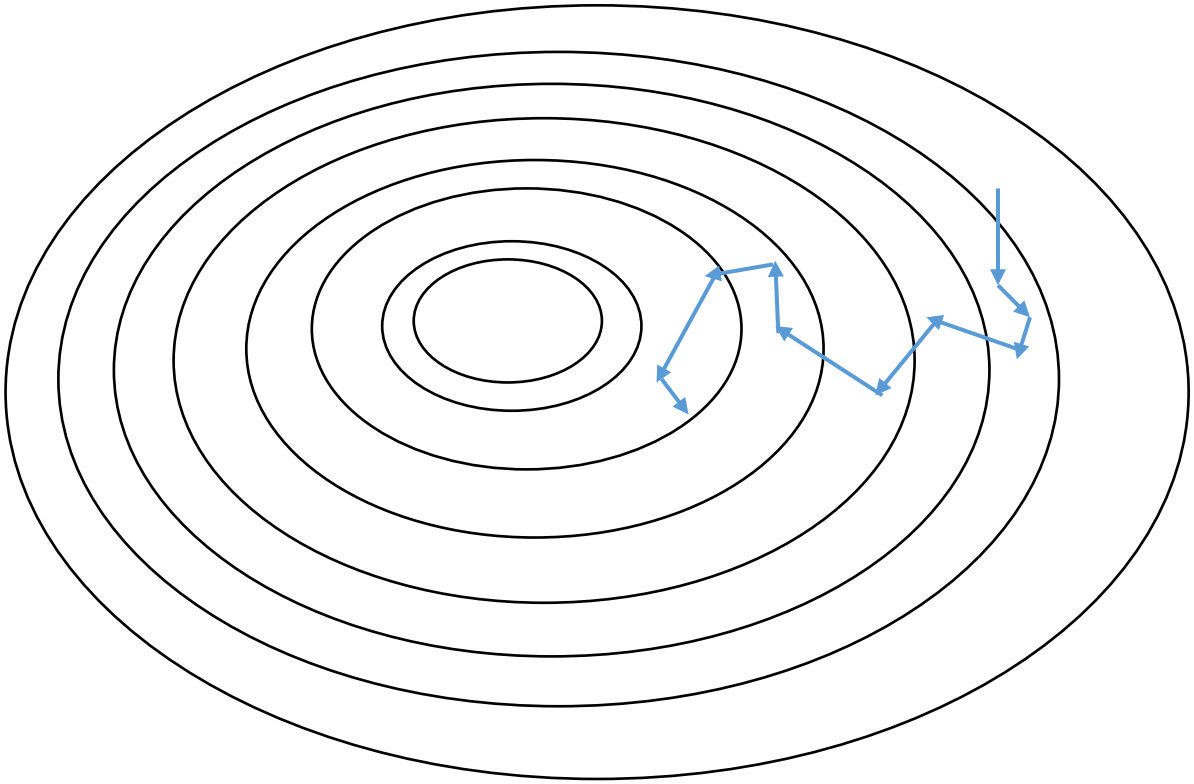
Stochastic Gradient descent

Gradient Descent vs SGD



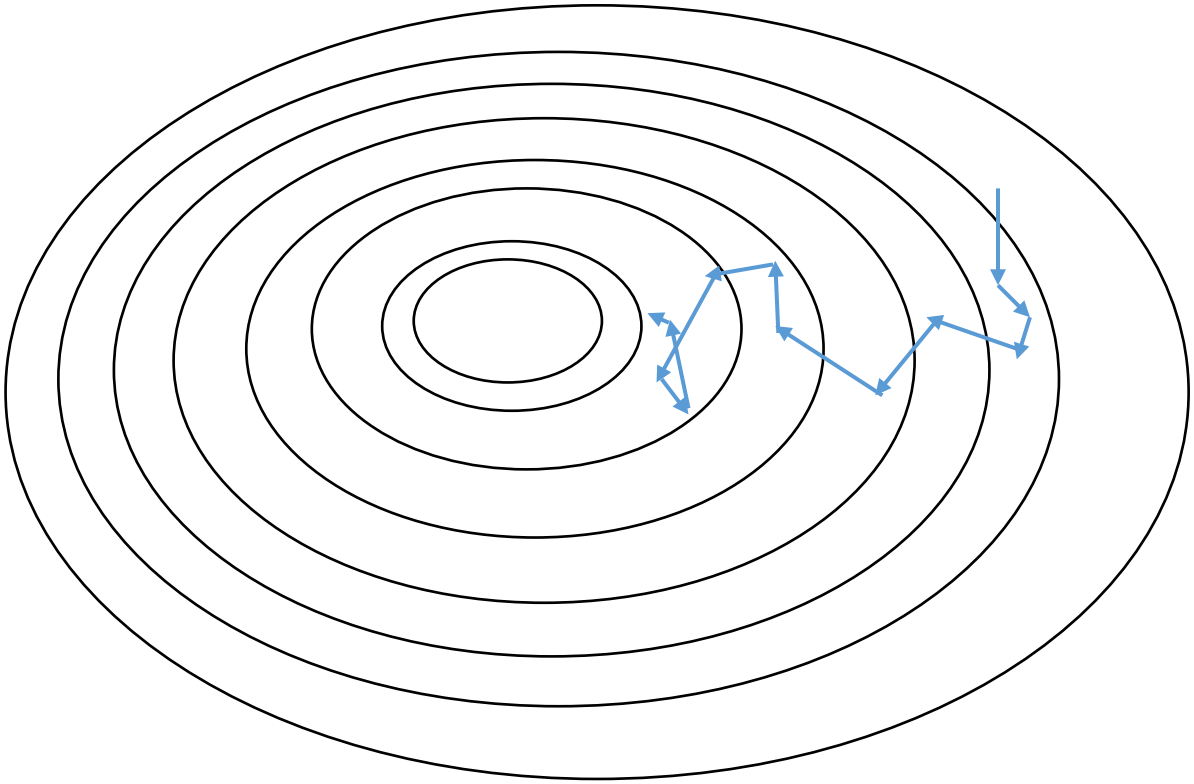
Stochastic Gradient descent

Gradient Descent vs SGD



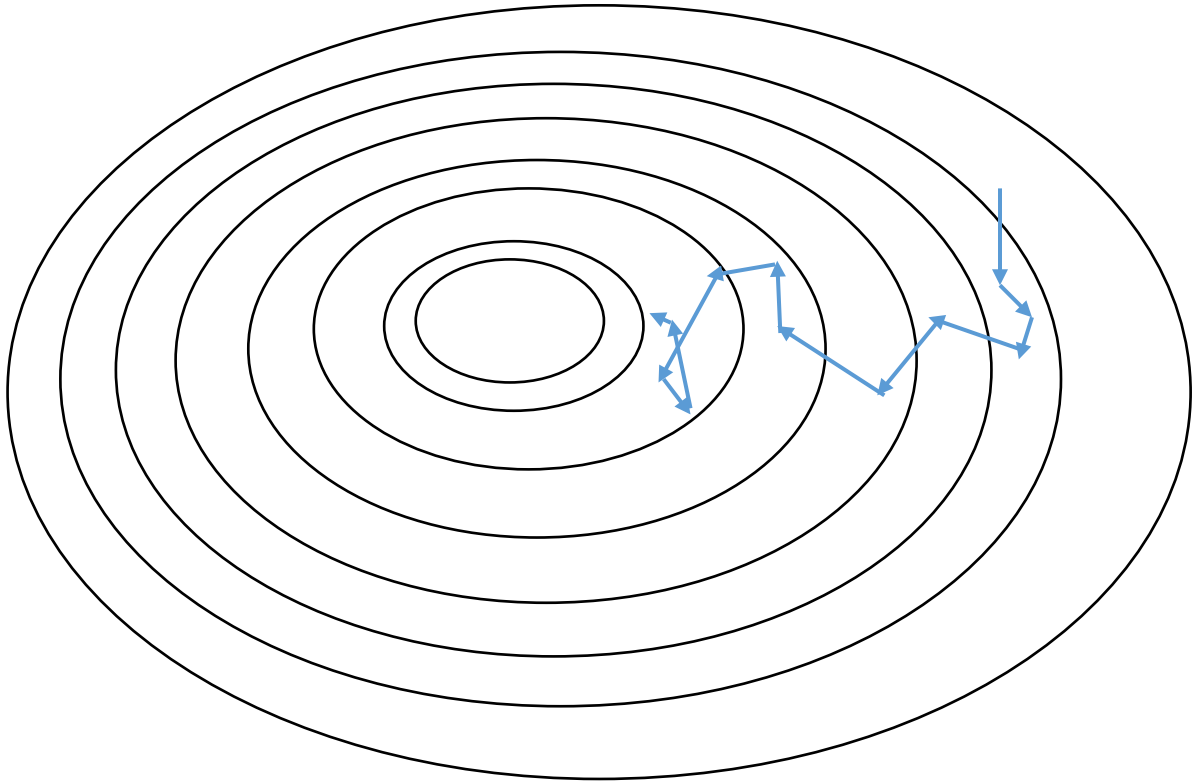
Stochastic Gradient descent

Gradient Descent vs SGD



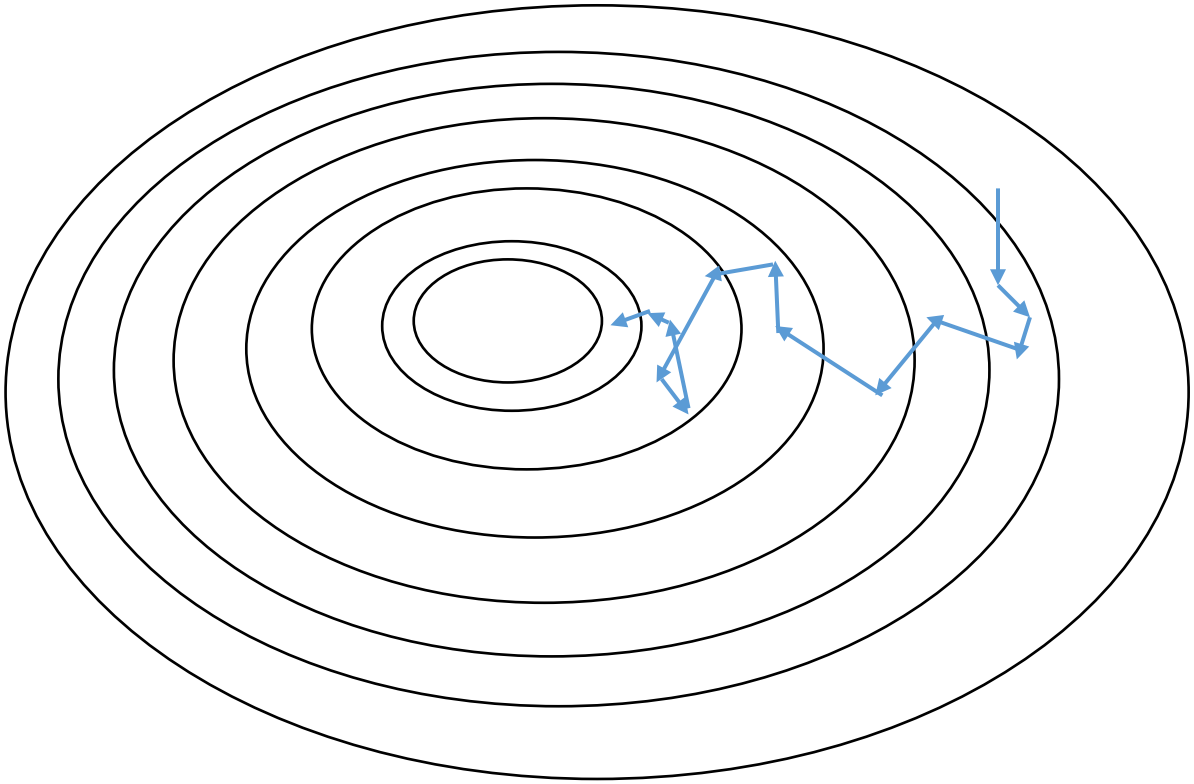
Stochastic Gradient descent

Gradient Descent vs SGD



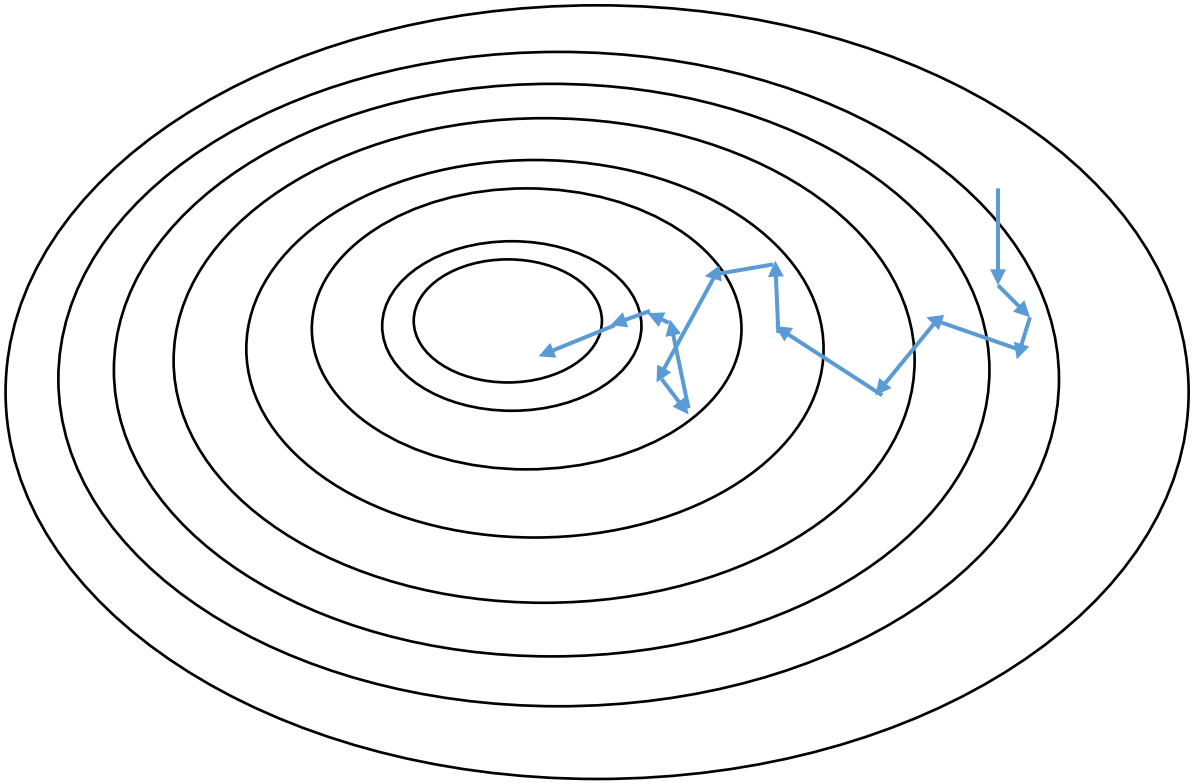
Stochastic Gradient descent

Gradient Descent vs SGD



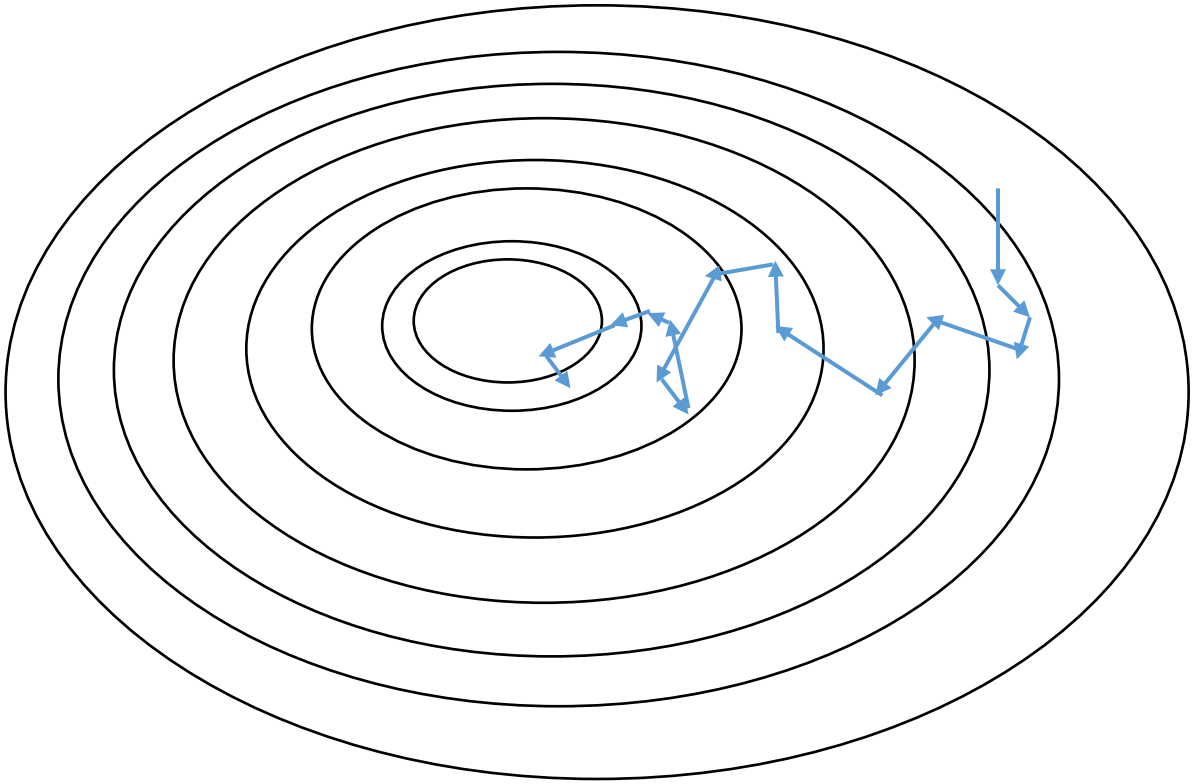
Stochastic Gradient descent

Gradient Descent vs SGD



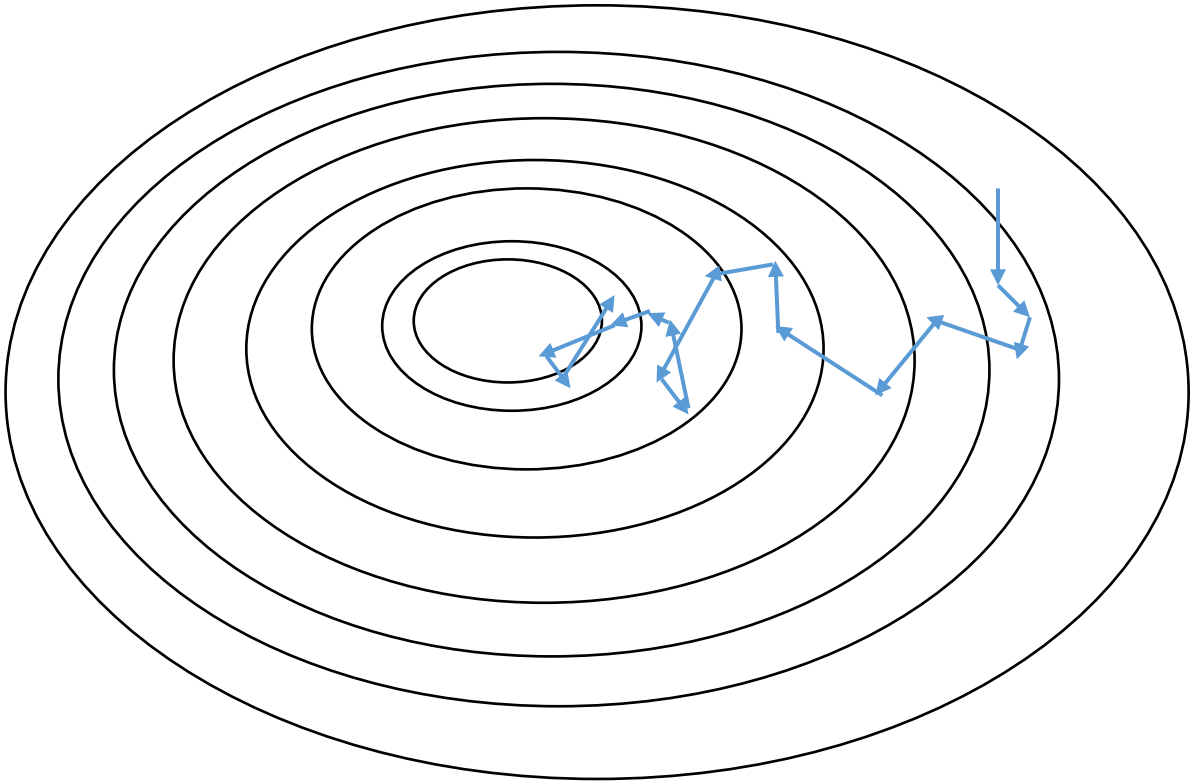
Stochastic Gradient descent

Gradient Descent vs SGD



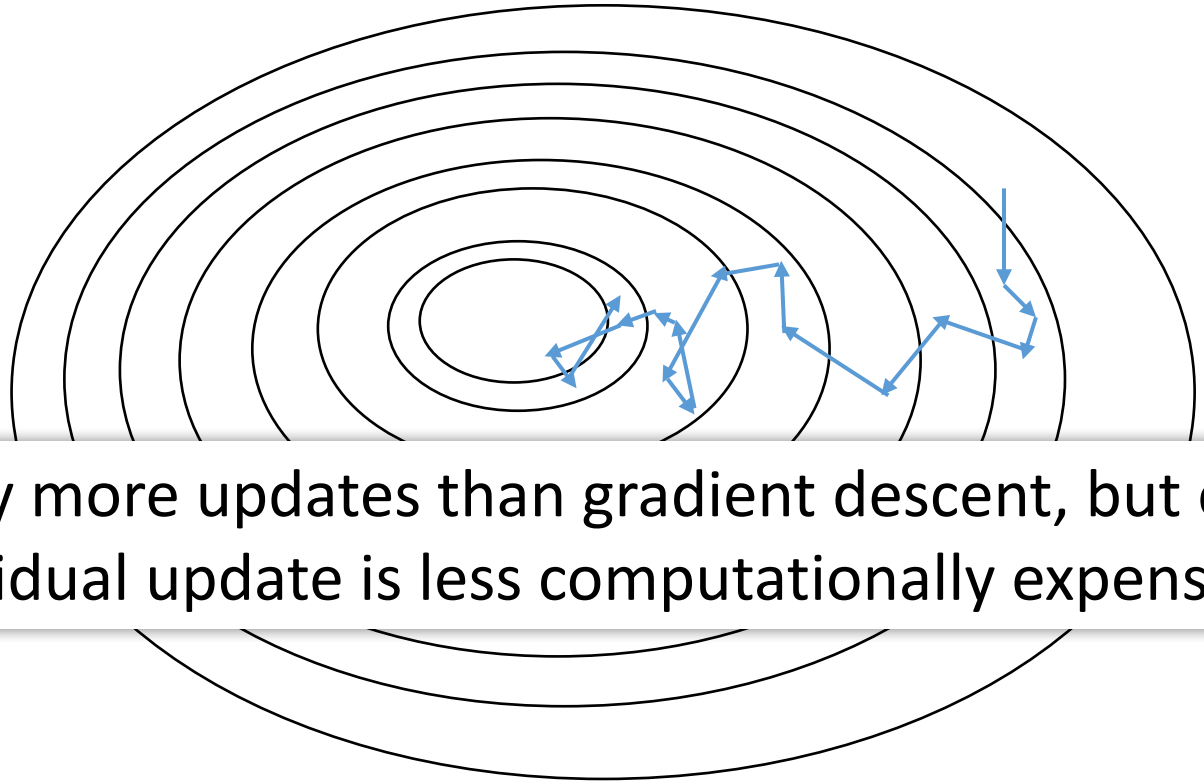
Stochastic Gradient descent

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!

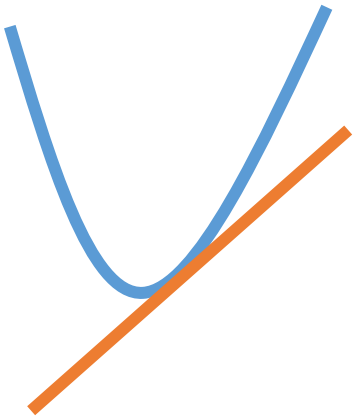
What is the derivative of the hinge loss with respect to w?

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

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

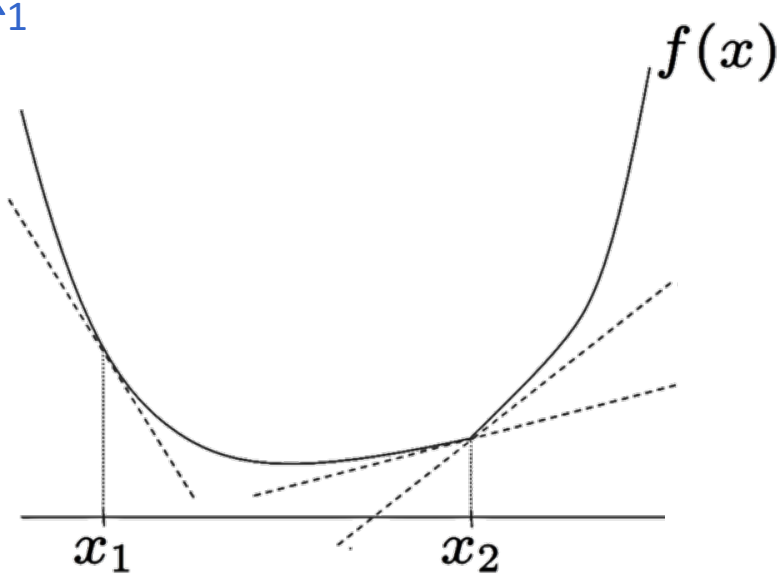
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

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



[Example from Boyd]

Sub-gradients

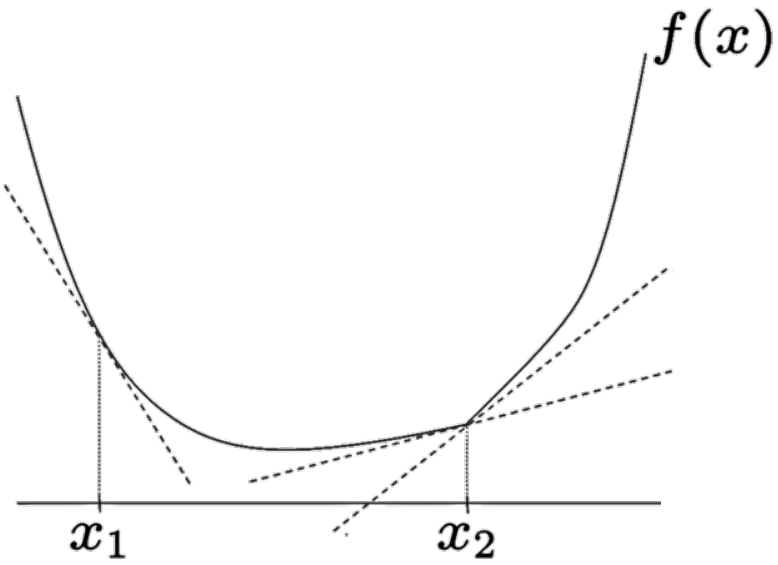
Formally, g is a subgradient to f at x if

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

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



[Example from Boyd]

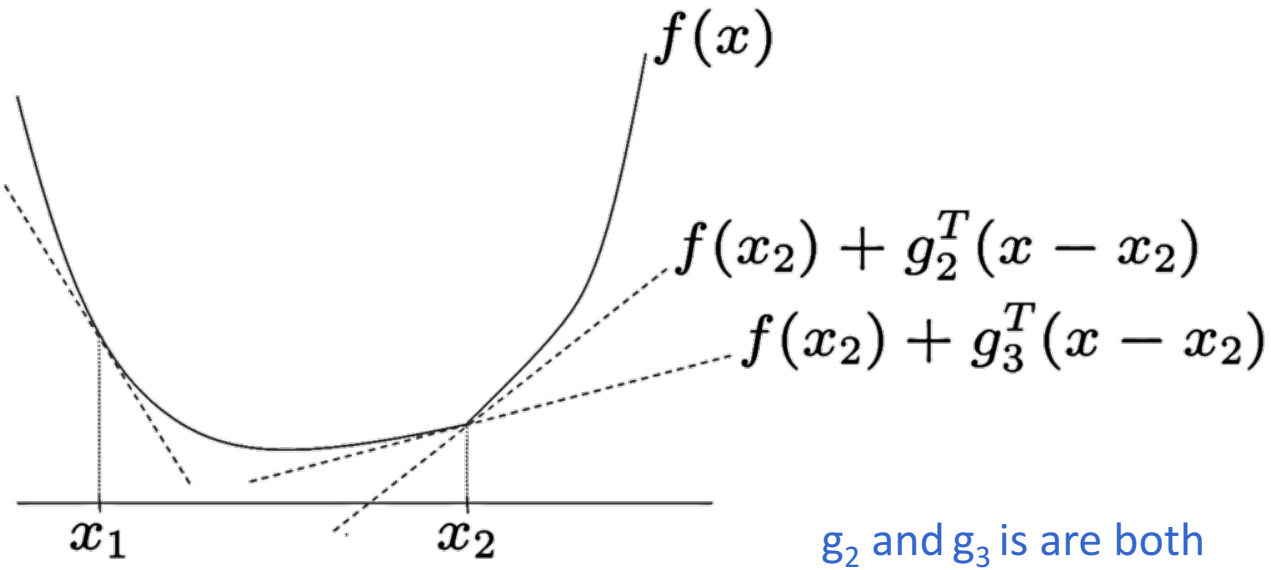
Sub-gradients

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

g_2 and g_3 is are both subgradients at x_2

[Example from Boyd]

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

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

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 $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

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

For epoch $1 \ldots T$:

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

if $y w^T x \geq 1$

$\mathrm{w} \leftarrow w - \eta \mathrm{w}$

else

$\mathrm{w} \leftarrow w - \eta(w + Cyx)$

Update $w \leftarrow w - \eta \nabla_{\mathrm{w}} f(x, y)$

Return $\boldsymbol{w}$

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

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 $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

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

For epoch $1 \ldots T$:

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

if $y(w^T x + b) \geq 1$

$\mathrm{w} \leftarrow w - \eta \mathrm{w}$

else

$\mathrm{w} \leftarrow w - \eta(w - Cyx)$

Update $w \leftarrow w - \eta \nabla_{\mathrm{w}} f(x, y)$

Return $\boldsymbol{w}$

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

Recap: The Perceptron Algorithm

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

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

SVM:
If $y(w^T x + b) < 1$
$\mathrm{w} \leftarrow w - \eta(w - Cyx)$
else: $\mathrm{w} \leftarrow w - \eta \mathrm{w}$

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

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

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

- SVM optimizes the hinge loss
  - With regularization

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

This lecture: Support vector machines

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

SVM: Primal and dual

The SVM objective

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

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

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

SVM: Primal and dual

The SVM objective

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

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

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

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

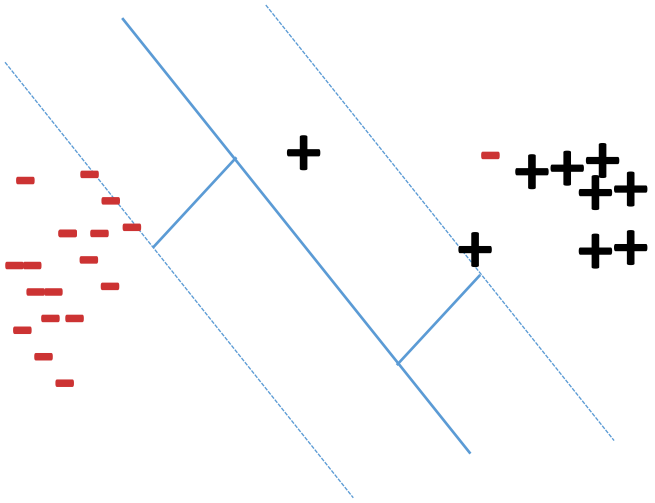
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Support vector machines

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

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

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



Furthermore,

$$\alpha_i = 0 \qquad \Rightarrow y_i \mathbf{w}^T x_i \geq 1$$

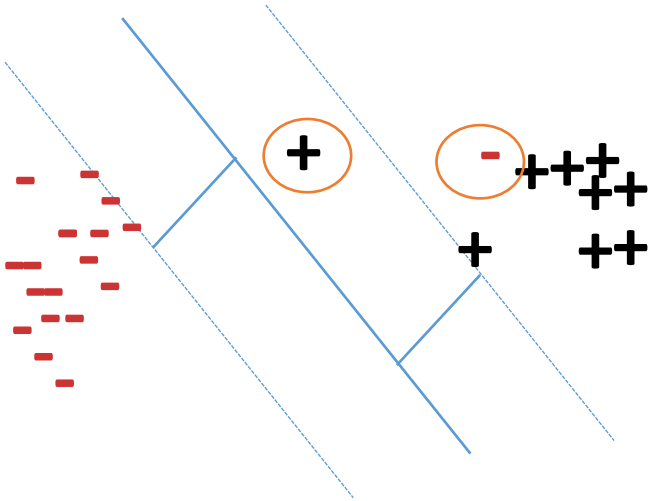
All points outside the margin

Support vector machines

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

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

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



Furthermore,

$$\alpha_i = 0 \qquad \Rightarrow y_i \mathbf{w}^T x_i \geq 1$$

$$\alpha_i = C \qquad \Rightarrow y_i \mathbf{w}^T x_i \leq 1$$

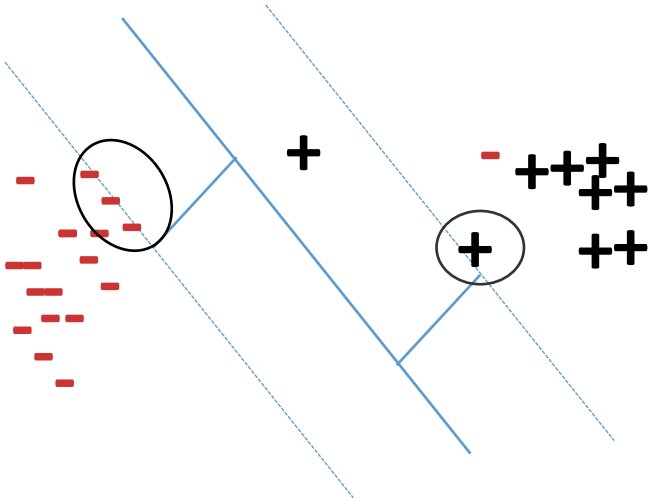
All points on the wrong side of the margin

Support vector machines

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

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

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



Furthermore,

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

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

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

All points on the margin

Support vectors

The weight vector is completely defined by training examples whose α_is are not zero

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

These examples are called the *support vectors*