Lecture 13:
Kernel Methods
Support Vector Machines
Fall 2022

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.

The Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

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

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

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

The Dual Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \boldsymbol{R}^m$
2. For $(\boldsymbol{x}_i, y_i)$ in $\mathcal{D}$:
3. if $y_i \sum_j \alpha_j y_j \phi(x_j)^T \phi(x_i) \leq \mathbf{0}$
4. $\alpha_i \leftarrow \alpha_i + 1$
5. Return $\boldsymbol{\alpha}$
6.

$\mathrm{w} \leftarrow \mathbf{0}$

$y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\phi(x)$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\top} \phi(x)) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x)$

$$\boxed{w = \sum_{j=1..m} \alpha_j\, y_j \phi(x_j)}$$

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

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

  - Prediction can be done by computing dot products between training examples and the new example **x**
- This is true if we map examples with $\phi(x)$

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

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 $\boldsymbol{\phi}(\cdot)$.*

In this example, $x \in R^2$, the benefit of using kernel is significant, but consider when $x \in R^{1000}$, $\phi(x) \in R^{500500}$

The Kernel Trick

Suppose we wish to compute

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

Here ϕ maps $\mathbf{x}$ and $\mathbf{z}$ to a high dimensional space

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

Kernel functions

- A kernel function $k(\cdot,\cdot)$ satisfies the following properties
- For any x_m, 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 $\phi(\cdot)$

Example: $(\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$ is a kernel, because it is the linear product of the following mapping

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

Exercise

- Let $x \in R^2$, show $\left(4 + 9x_i^T x_j\right)^2$ is a valid kernel.

Zoo of Kernel

Linear Kernel: $K(x, y) = x^T y$

Polynomial Kernel: $K(x, y) = (x^\mathsf{T} y + c)^d$

RBF Kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$

The proof will not be in exam

RBF Kernel maps data into an infinite-dimension space

$$\begin{aligned}
\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) &= \exp\left(\frac{2}{2}\mathbf{x}^\top\mathbf{x}' - \frac{1}{2}\|\mathbf{x}\|^2 - \frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \exp(\mathbf{x}^\top\mathbf{x}')\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\sum_{n_1+n_2+\cdots+n_k=j}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)\frac{{x'}_1^{n_1}\cdots {x'}_k^{n_k}}{\sqrt{n_1!\cdots n_k!}} \\
&= \langle\varphi(\mathbf{x}),\varphi(\mathbf{x}')\rangle
\end{aligned}$$

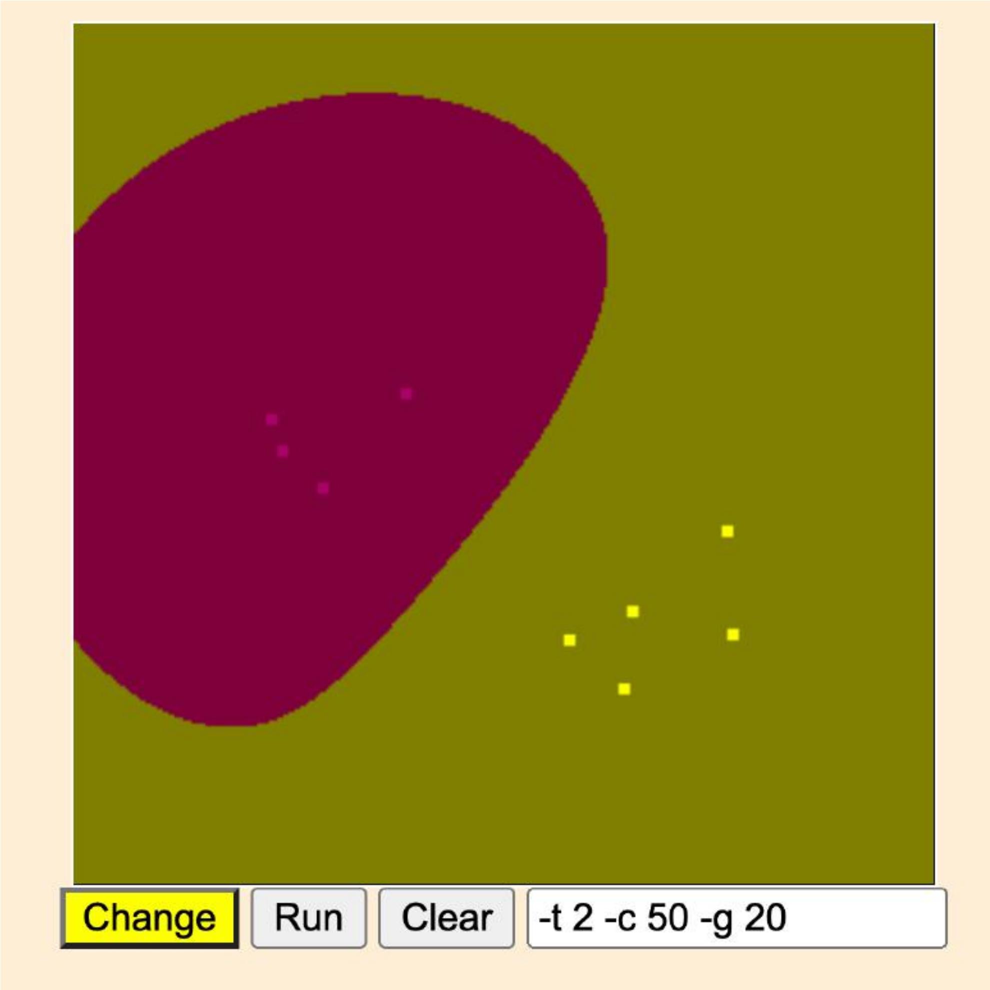
$$\varphi(\mathbf{x}) = \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\left(a_{l_0}^{(0)}, a_1^{(1)}, \ldots, a_{l_1}^{(1)}, \ldots, a_1^{(j)}, \ldots, a_{l_j}^{(j)}, \ldots\right)$$

where $l_j = \binom{k+j-1}{j}$,

$$a_l^{(j)} = \frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}} \quad \Big| \quad n_1+n_2+\cdots+n_k = j \wedge 1 \le l \le l_j$$

Demo – SVM (will be taught later)

- https://www.csie.ntu.edu.tw/~cjlin/libsvm/



The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \boldsymbol{R}^{2\mathrm{n}}$
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.
6.

Return $\boldsymbol{w}$

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

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\boldsymbol{w}^{\mathsf{T}} \begin{bmatrix} x \\ x^2 \end{bmatrix})$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

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

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

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

$$\boxed{w^T \phi(x) = \sum_i \alpha_i\, y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)}$$

The Dual Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \boldsymbol{R}^m$
2. For $(\boldsymbol{x}_i, y_i)$ in $\mathcal{D}$:
3. if $y_i \sum_j \alpha_j y_j \phi(x_j)^T \phi(x_i) \leq \mathbf{0}$
4. $\alpha_i \leftarrow \alpha_i + 1$
5. Return $\boldsymbol{\alpha}$
6.

$\boxed{\mathrm{w} \leftarrow \mathbf{0}}$

$\boxed{y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}}$

$\boxed{\boldsymbol{w} \leftarrow \boldsymbol{w} + y\, \phi(x)}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \phi(x)) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x)$

$$\boxed{w = \sum_{j=1..m} \alpha_j\, y_j \phi(x_j)}$$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

```
Initialize α ← 0 ∈ R^m
For (x_i, y_i) in D:
    if y_i Σ_j α_j y_j K(x_i, x_j) ≤ 0
        α_i ← α_i + 1
Return w
```

$\boxed{\mathrm{w} \leftarrow \mathbf{0}}$

$\boxed{y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}}$

$\boxed{\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\phi(x)}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \phi(x)) = \sum_i \alpha_i y_i K(x_i, x)$

$$\boxed{w^T \phi(x) = \sum_i \alpha_i\, y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)}$$

Lecture 13: Support Vector Machines

Fall 2022

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.

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

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.

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 $\boldsymbol{w}$

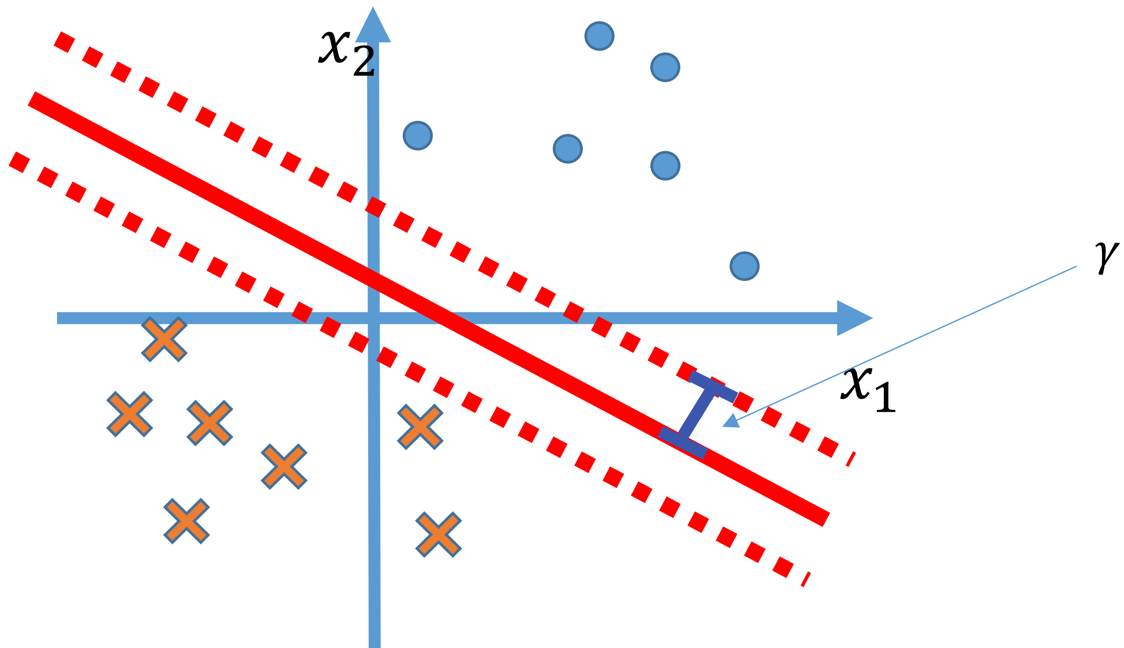
Assume $y \in \{1, -1\}$

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

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

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

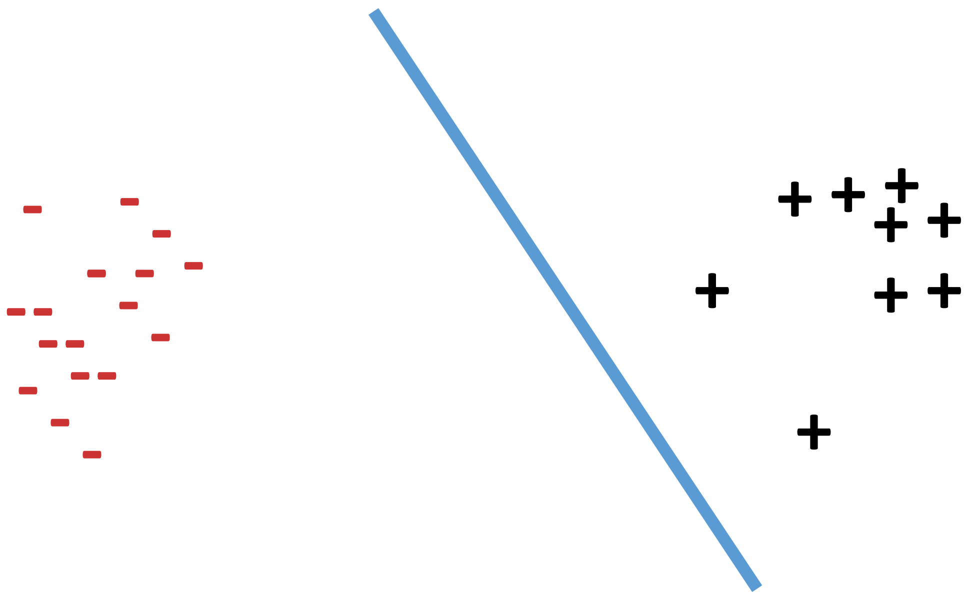
- The SVM objective

- Solving the SVM optimization problem

- Support vectors, duals and kernels

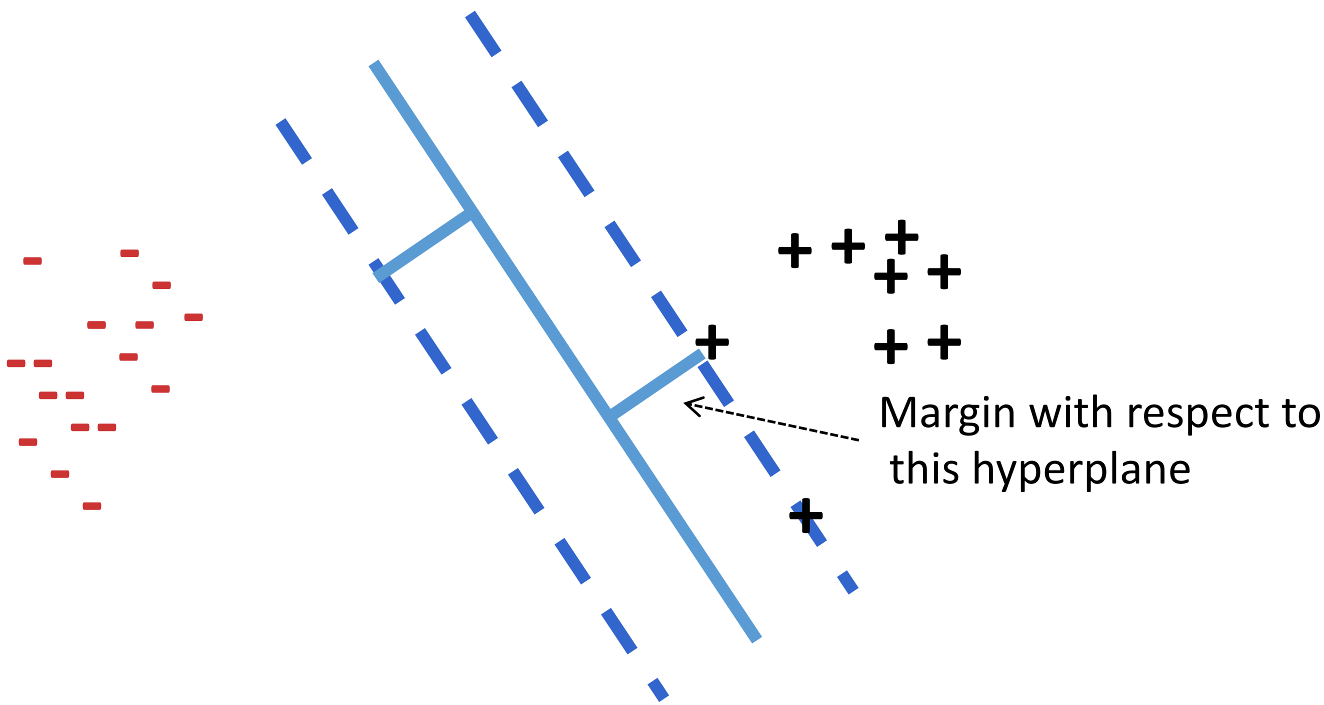
Recall: Margin

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

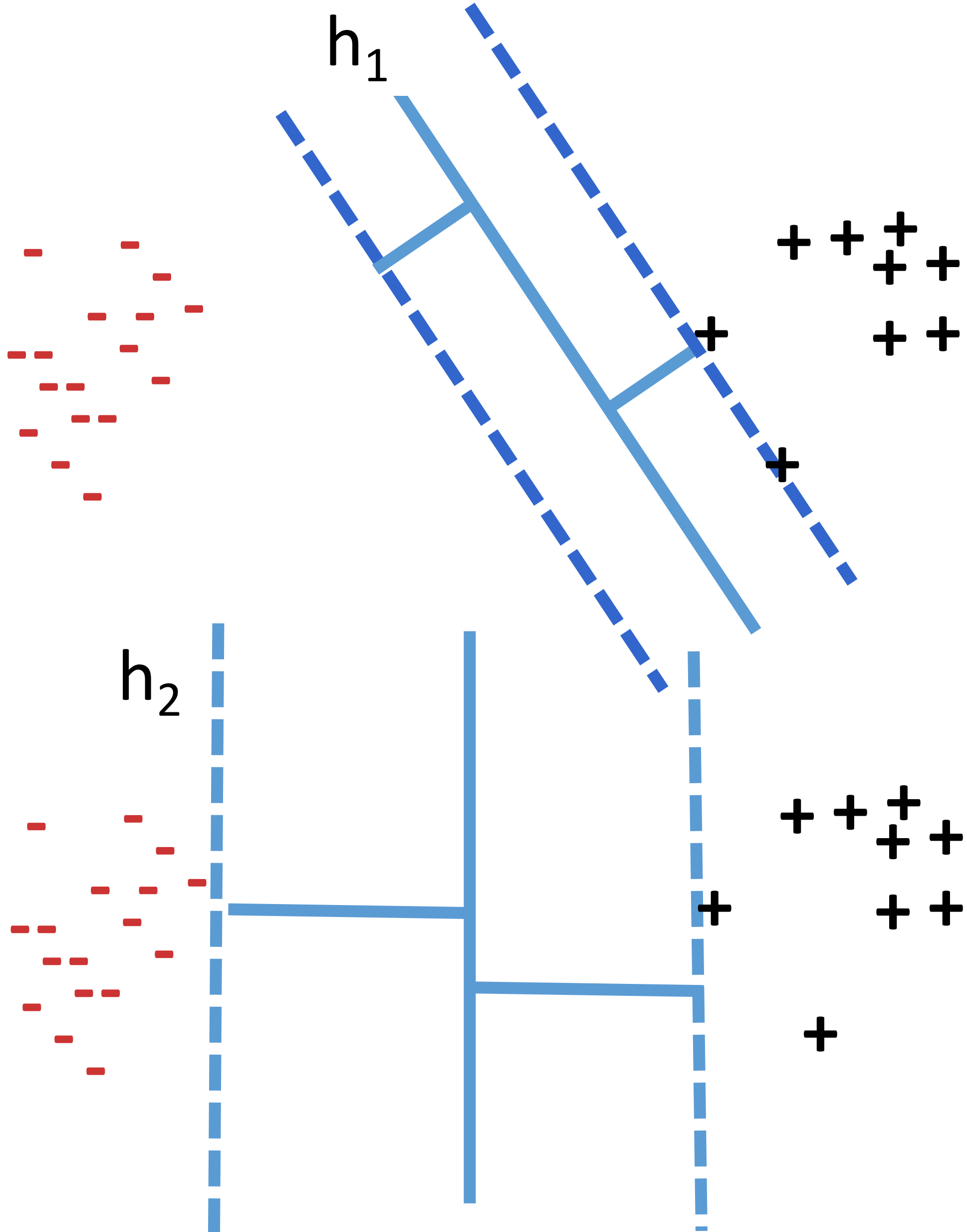


Recall: Margin

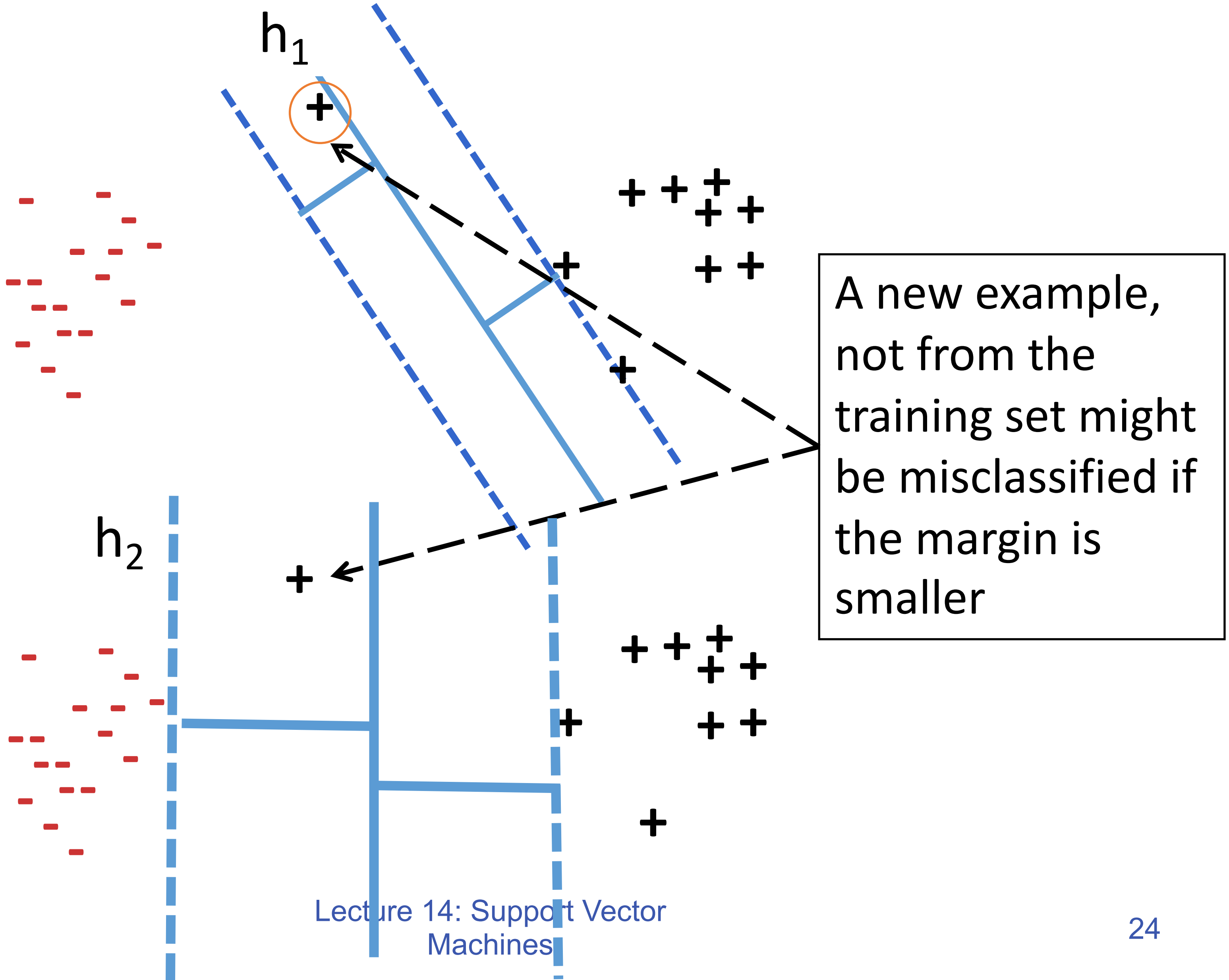
The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Which line is a better choice? Why?



Which line is a better choice? Why?



Learning strategy

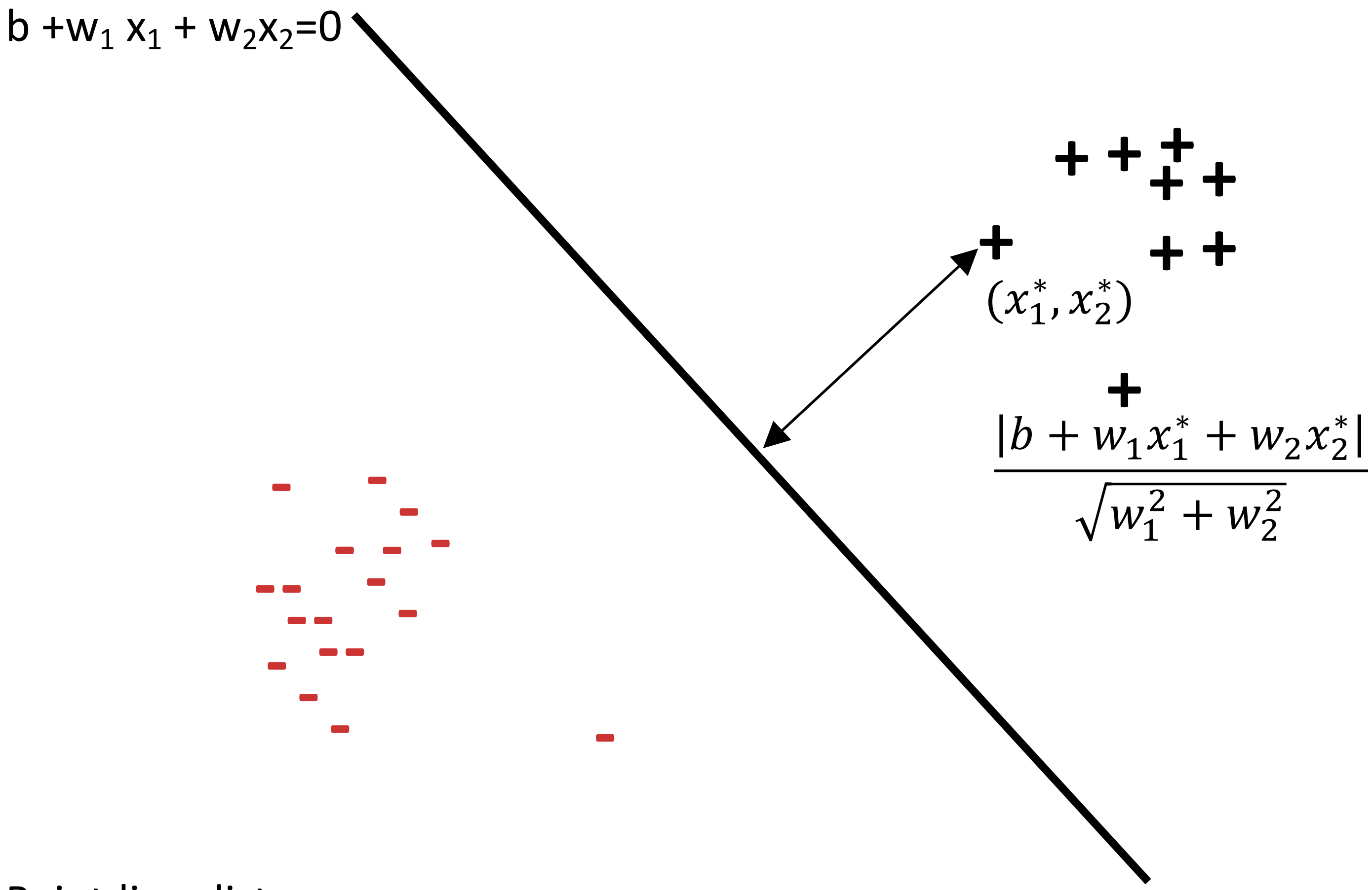
Find the linear separator that maximizes the margin

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

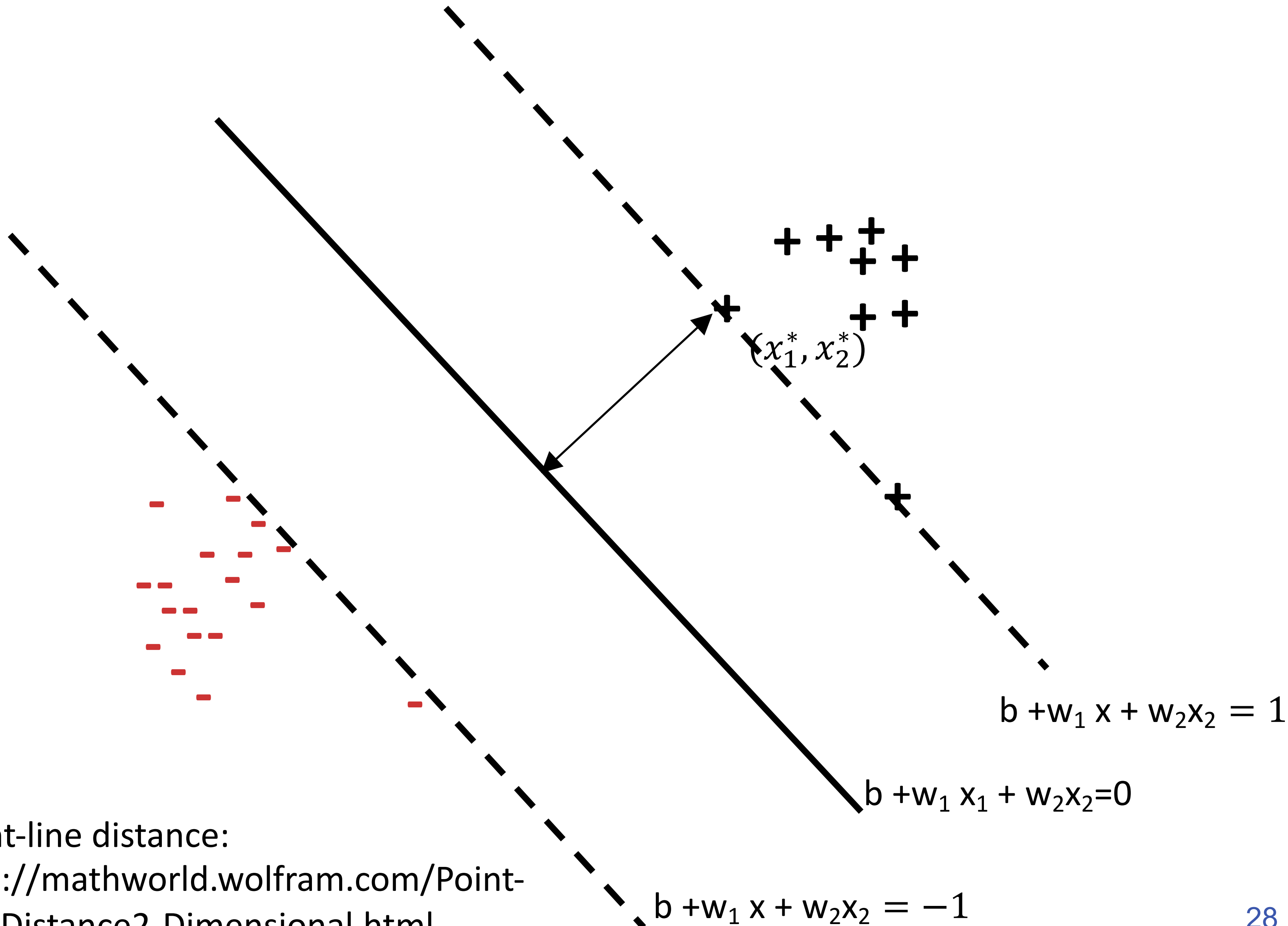
Prediction = $\mathrm{sgn}(b + w_1 x_1 + w_2 x_2)$



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

Margin

What is the distance between
$b + w_1 x + w_2 x_2 = 1$ and $b + w_1 x + w_2 x_2 = 0$

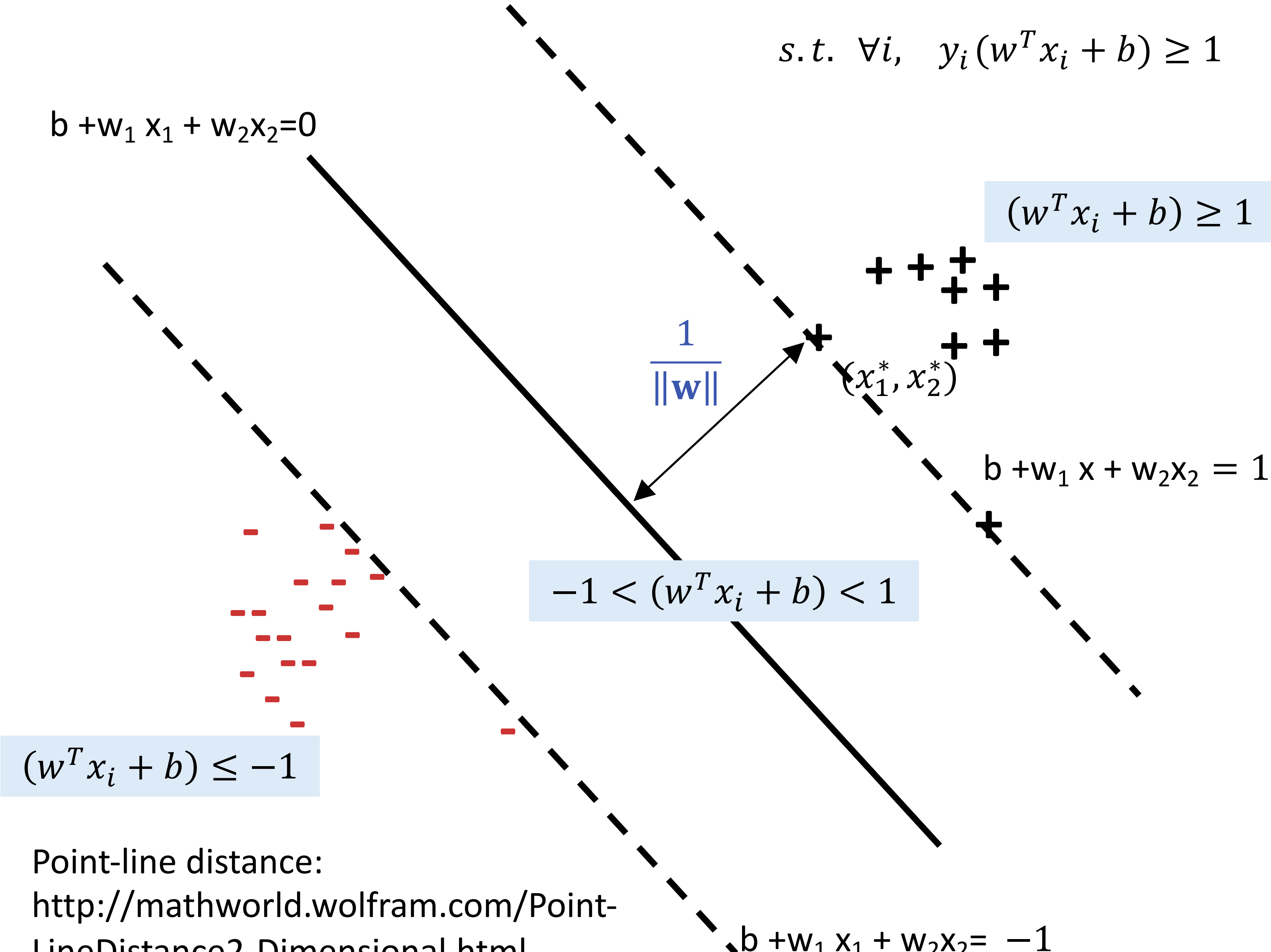


Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Hard SVM

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

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



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

Max-margin classifiers

- Learning problem:

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

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

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

Max-margin classifiers

- Learning problem:

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

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

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

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

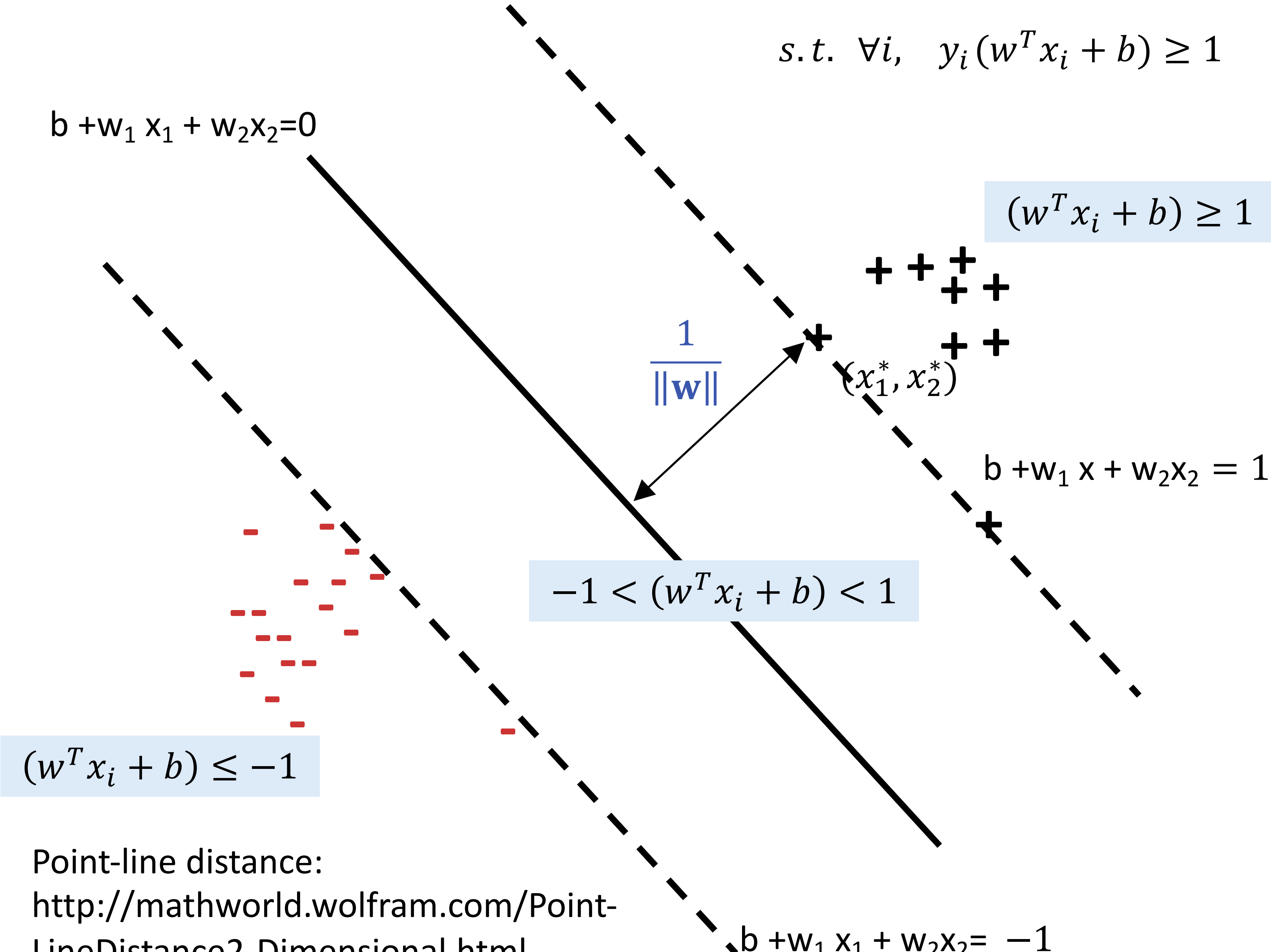
- This is called the “hard” Support Vector Machine

We will look at how to solve this optimization problem later

Hard SVM

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

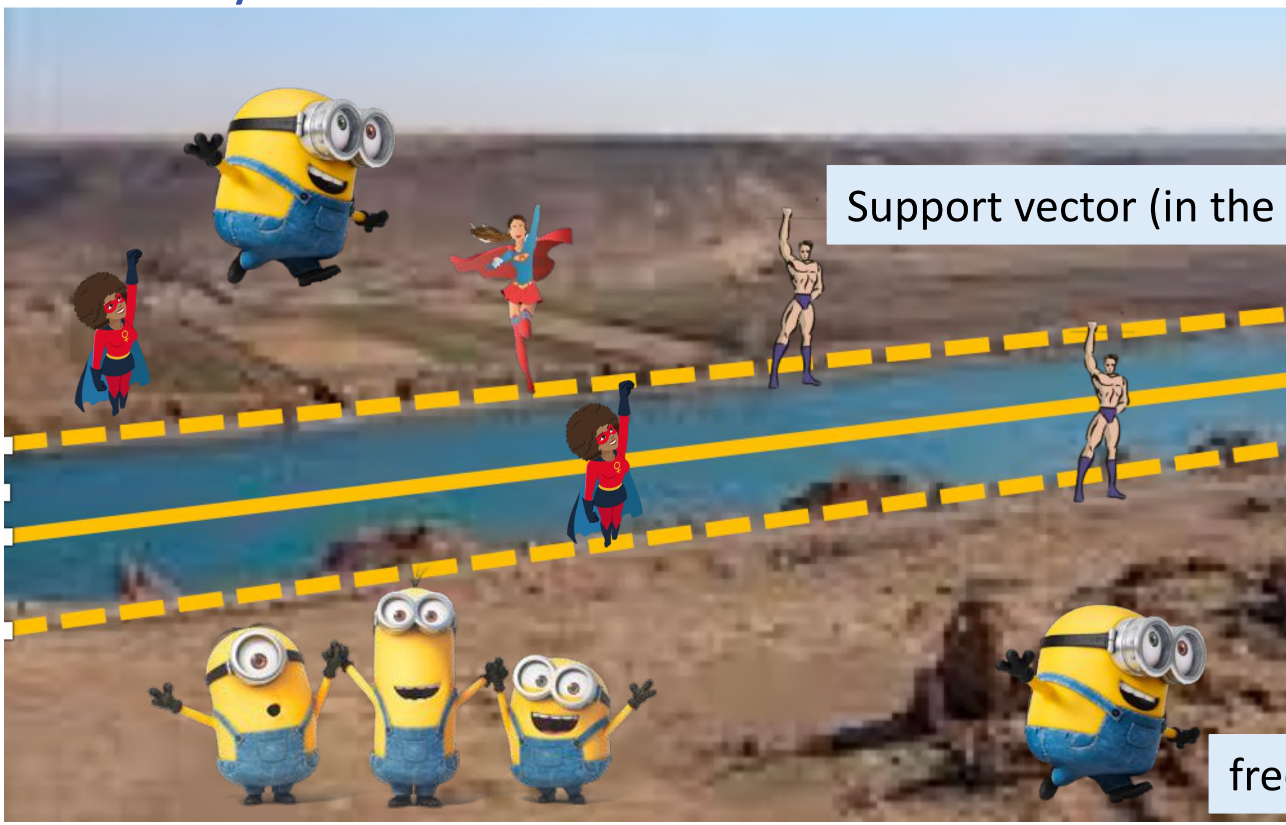
$$s.t. \ \forall i, \quad y_i(w^T x_i + b) \geq 1$$



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

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

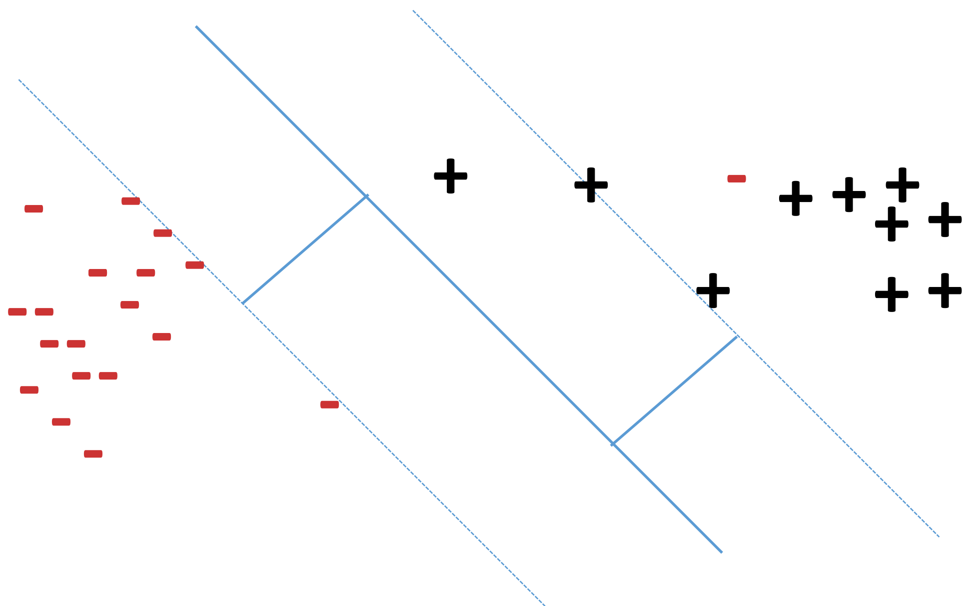
Every example has an functional margin of at least 1

- 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!

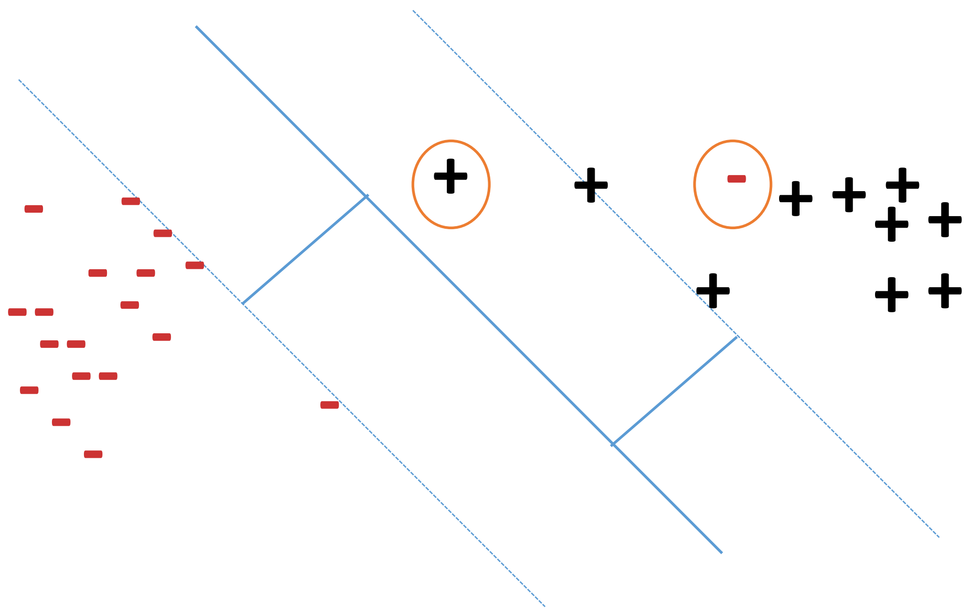
Dealing with non-separable data

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



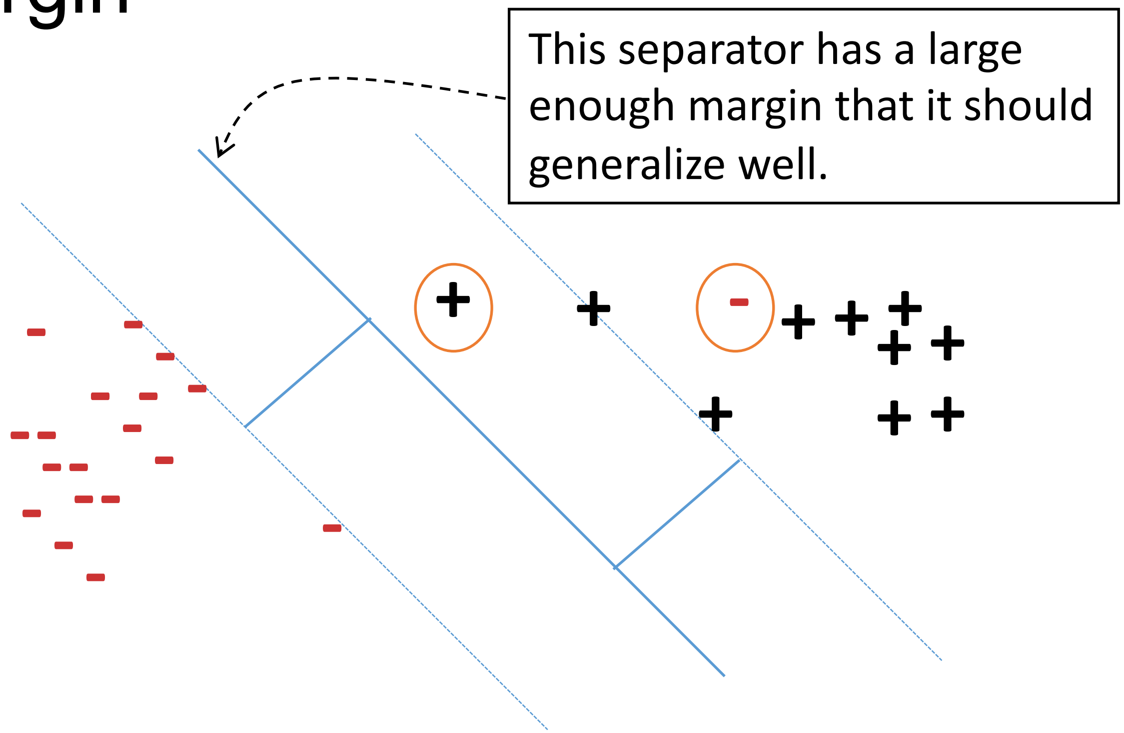
Dealing with non-separable data

Key idea: Allow some examples to "break into the margin"



Dealing with non-separable data

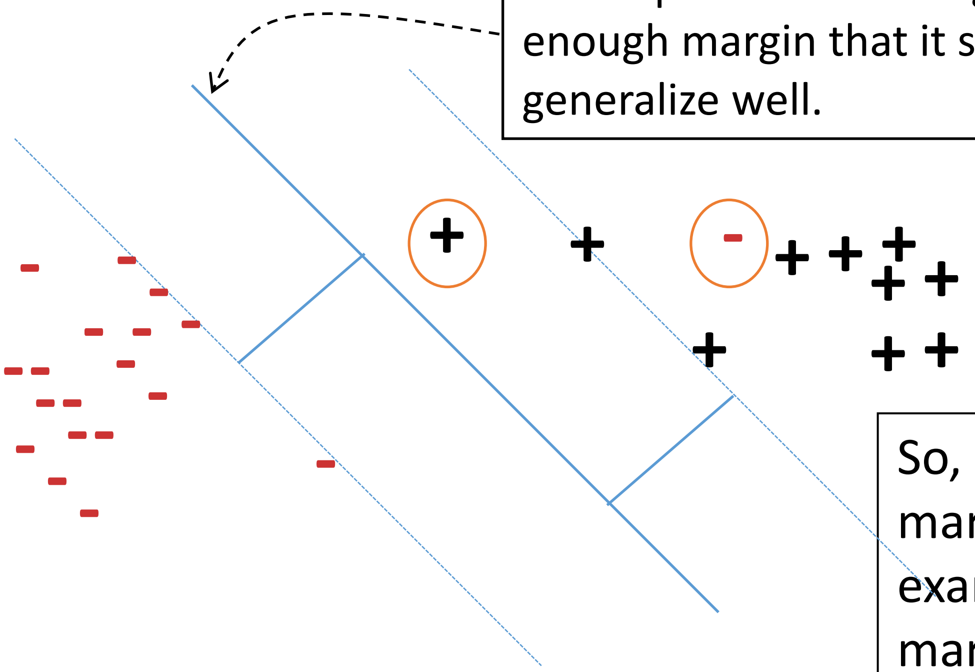
Key idea: Allow some examples to "break into the margin"



Dealing with non-separable data

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

This separator has a large enough margin that it should generalize well.



So, when computing margin, ignore the examples that make the margin smaller or the data inseparable.

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

Maximize margin

Tradeoff between the two terms

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

Minimize total slack (i.e allow as few examples as possible to violate the margin)

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

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

Soft SVM

Maximize margin

Tradeoff between the two terms

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

Minimize total slack (i.e allow as few examples as possible to violate the margin)

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

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

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

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

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

$$\Downarrow$$

$$\xi_i \geq \max(0, 1 - y_i(w^T x_i + b))$$

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

Maximizing margin and minimizing loss

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

Maximize margin

Penalty for the prediction

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

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.

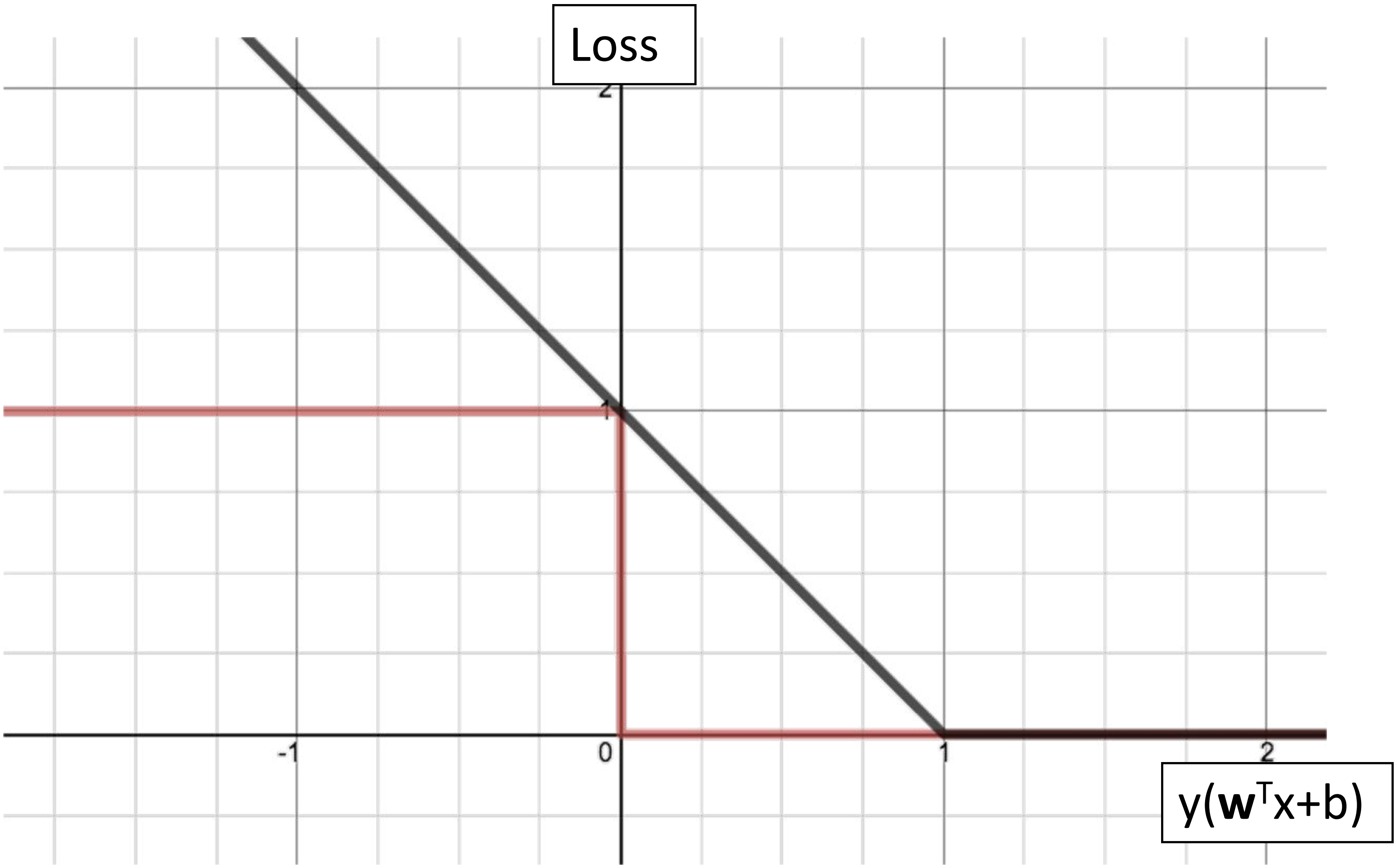
Capacity control gives better generalization

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]

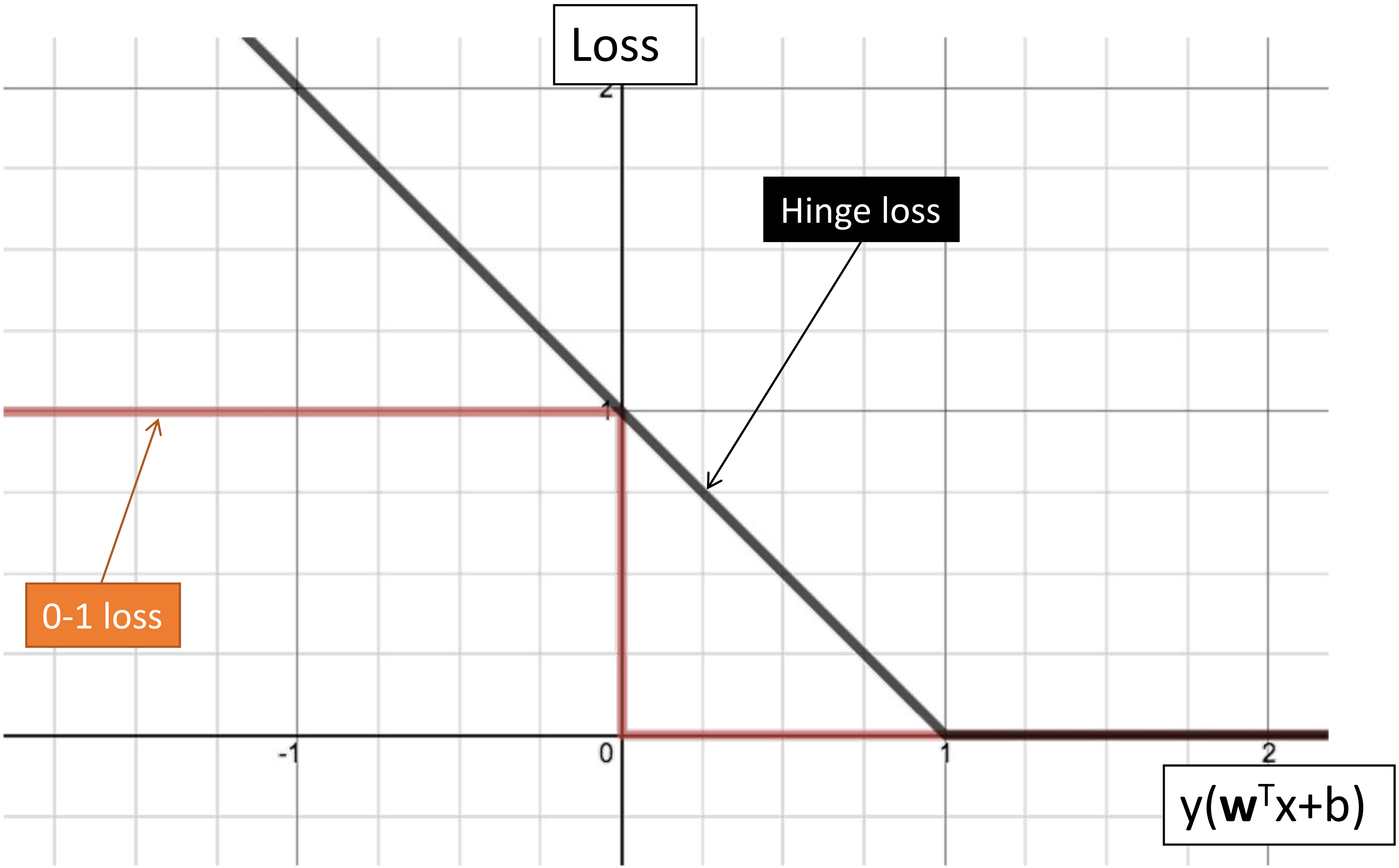
The Hinge Loss

$$L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))$$



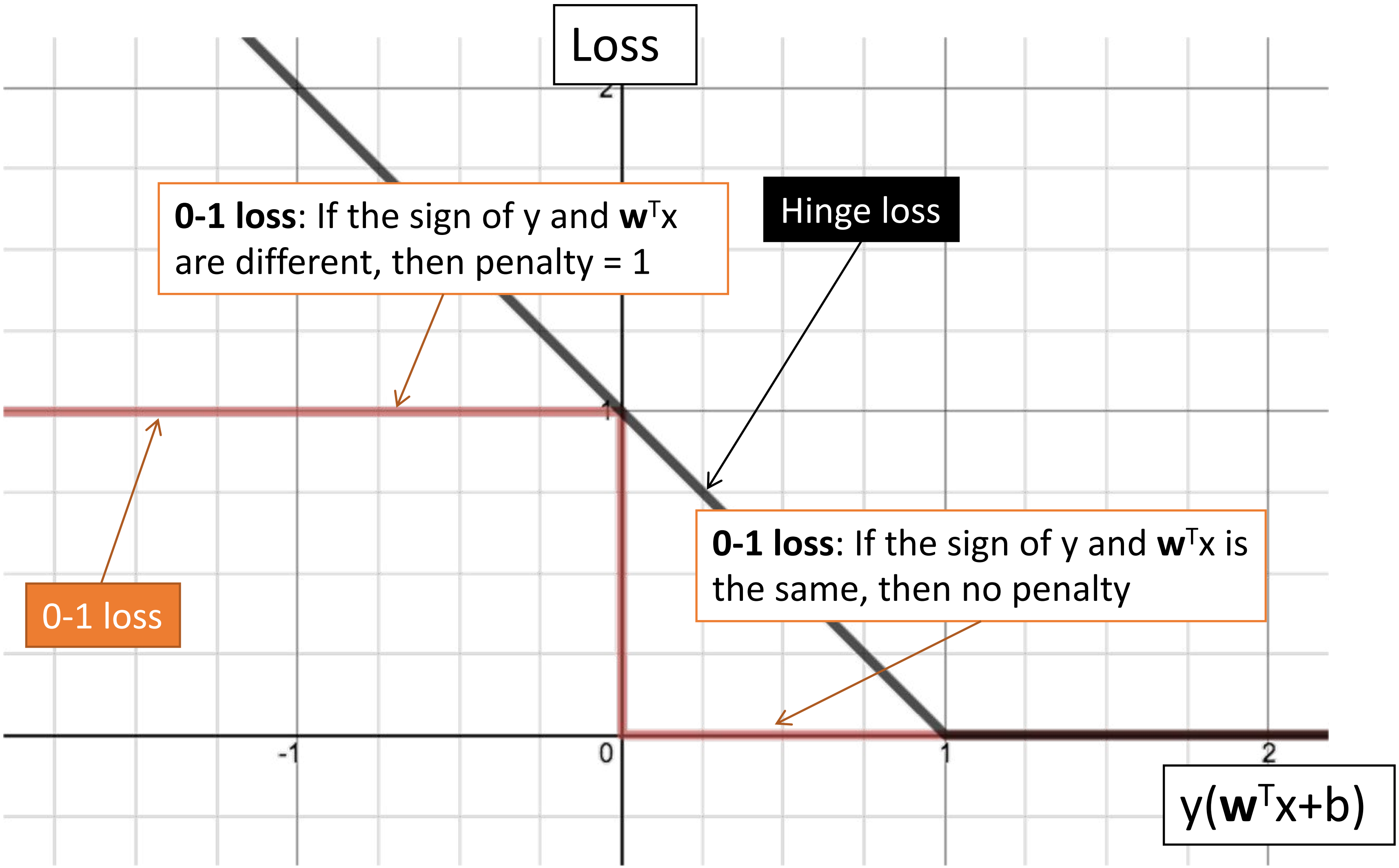
The Hinge Loss

$$\boxed{L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))}$$



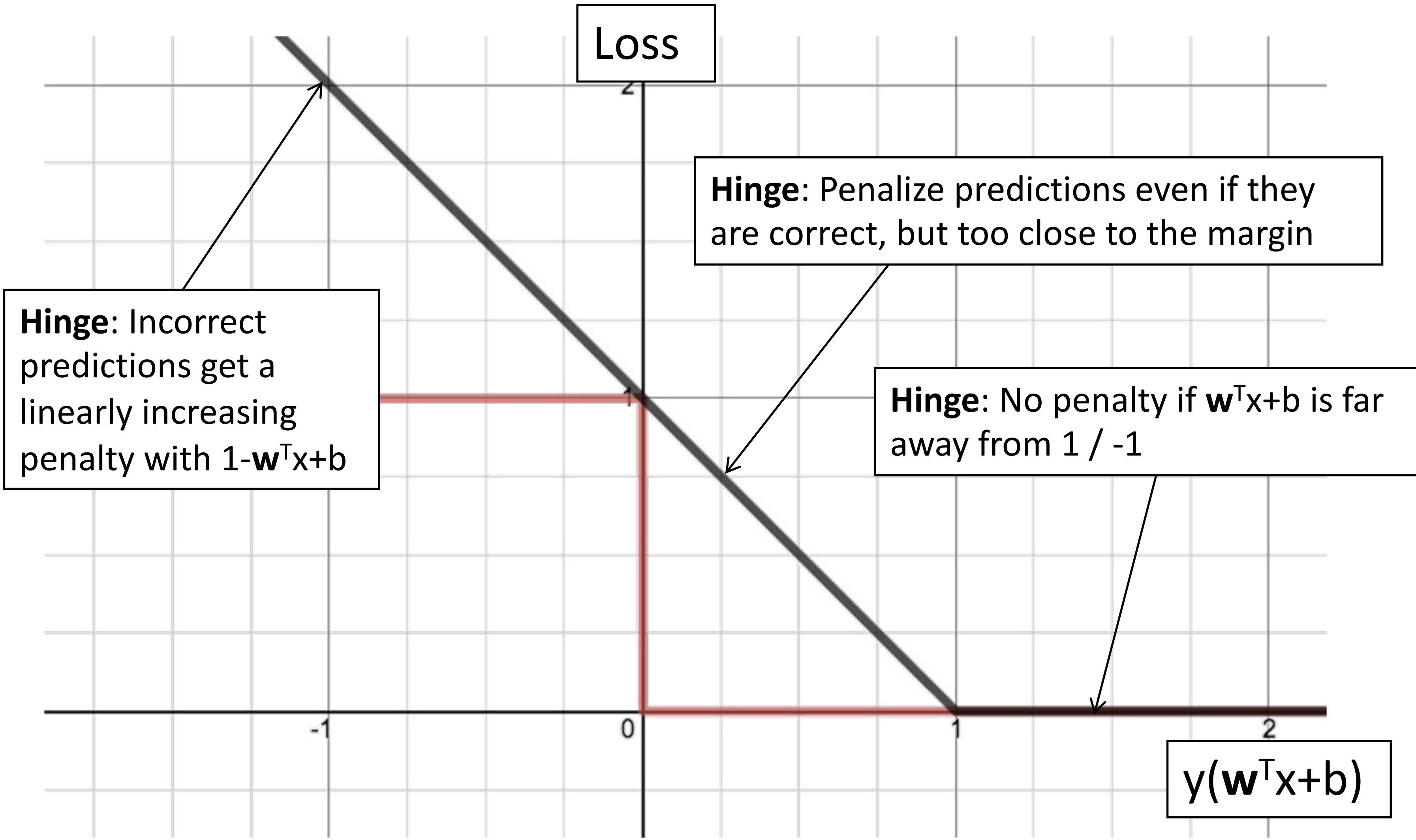
The Hinge Loss

$$L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))$$



The Hinge Loss

$$L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))$$



This lecture: Support vector machines

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

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**

Outline: Training SVM by optimization

1. Stochastic gradient descent
2. Sub-derivatives of the hinge loss
3. Stochastic sub-gradient descent for SVM
4. 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

Hinge loss is not differentiable!

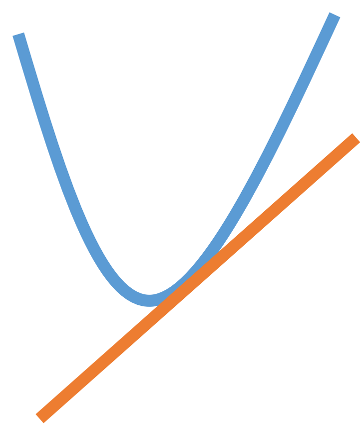
What is the derivative of the hinge loss with respect to w?

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

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

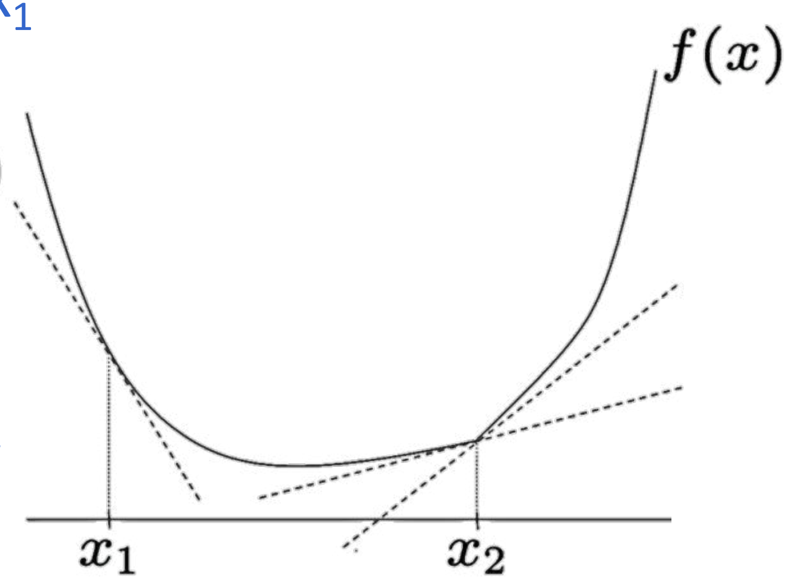
Formally, g is a subgradient to f at x if

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

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]

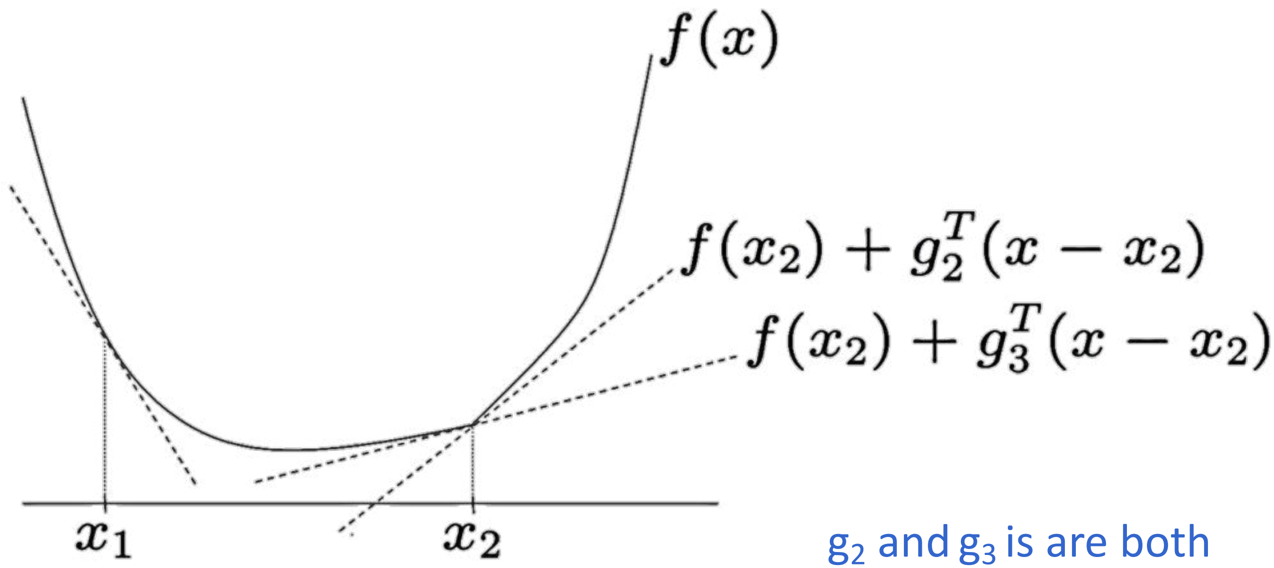
Sub-gradients

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

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. ~~Stochastic gradient descent~~
2. ~~Sub-derivatives of the hinge loss~~
3. Stochastic sub-gradient descent for SVM
4. 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

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

Return $\boldsymbol{w}$

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

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. b ← b + y
6. Return $\boldsymbol{w}$

SVM:

If $y(w^T x + b) < 1$

$w \leftarrow w - \eta(w - Cyx)$

else: $w \leftarrow w - \eta w$

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

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

$$\boxed{L_{Hinge}(y, x, w) = \max(0, -y_i(w^T x_i + b))}$$

- SVM optimizes the hinge loss
  - With regularization

$$\boxed{L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))}$$

This lecture: Support vector machines

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

Maximizing margin and minimizing loss

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

Maximize margin

Penalty for the prediction

There are 3 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 = $1 - y_i(w^T x_i + b)$

$$\boxed{L_{Hinge}(y, x, w) = \max(0, 1 - y_i(w^T x_i + b))}$$

This is the hinge loss function

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

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

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

$$\text{maximize}_\alpha \quad \textstyle\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\textstyle\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

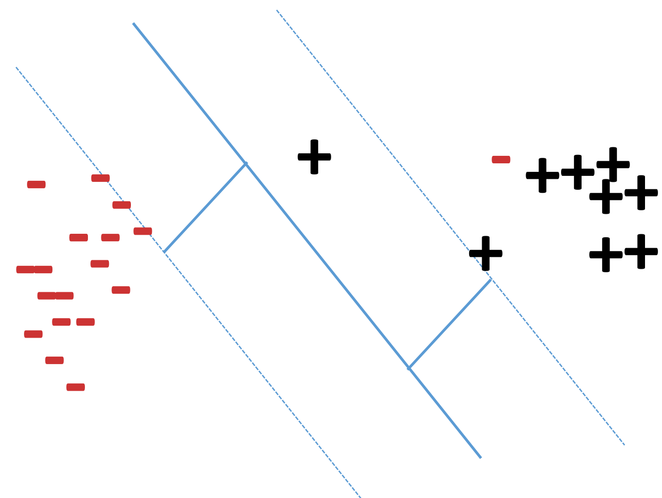
See the proof at https://www.csie.ntu.edu.tw/~cjlin/papers/bottou_lin.pdf

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,

$$y_i(w^T x_i + b) > 1 \Rightarrow \alpha_i = 0$$

$$y_i(w^T x_i + b) < 1 \Rightarrow \alpha_i = C$$

$$y_i(w^T x_i + b) = 1 \Rightarrow 0 \leq \alpha_i \leq C$$

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

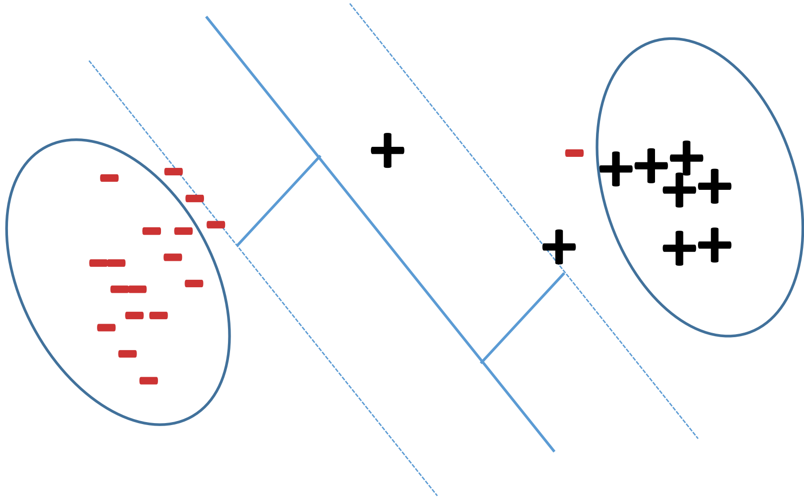
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\boxed{y_i(w^T x_i + b) > 1 \Rightarrow \alpha_i = 0}$$

$$y_i(w^T x_i + b) < 1 \Rightarrow \alpha_i = C$$

$$y_i(w^T x_i + b) = 1 \Rightarrow 0 \leq \alpha_i \leq C$$

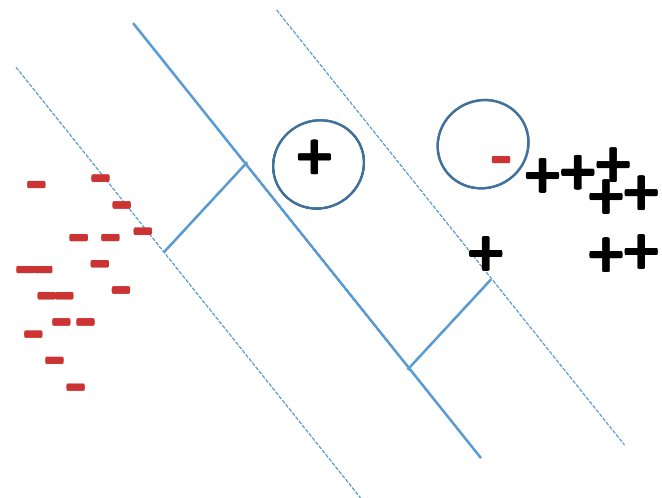


Support vector machines

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,

$$y_i(w^T x_i + b) > 1 \Rightarrow \alpha_i = 0$$

$$\boxed{y_i(w^T x_i + b) < 1 \Rightarrow \alpha_i = C}$$

$$y_i(w^T x_i + b) = 1 \Rightarrow 0 \leq \alpha_i \leq C$$

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

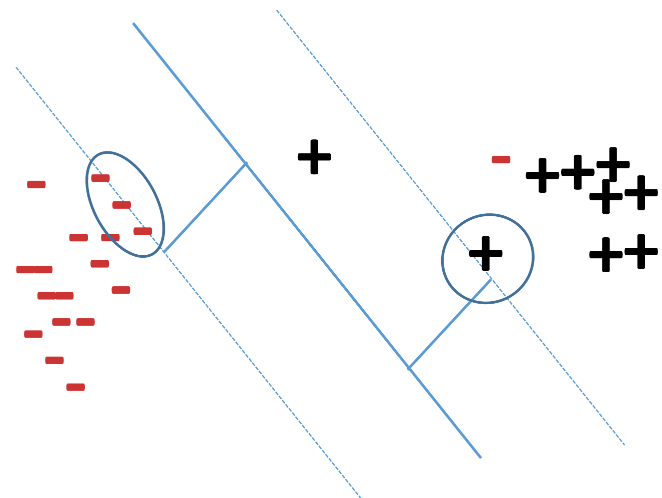
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$y_i(w^T x_i + b) > 1 \Rightarrow \alpha_i = 0$$

$$y_i(w^T x_i + b) < 1 \Rightarrow \alpha_i = C$$

$$\boxed{y_i(w^T x_i + b) = 1 \Rightarrow 0 \leq \alpha_i \leq C}$$



Support vectors

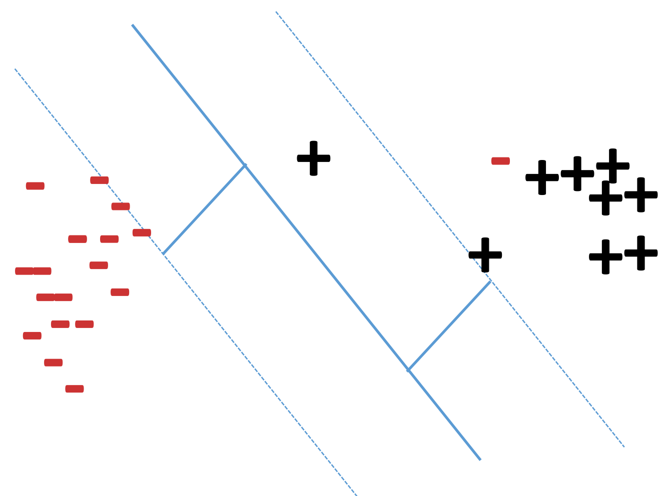
The weight vector is completely defined by training examples whose α_is are not zero

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

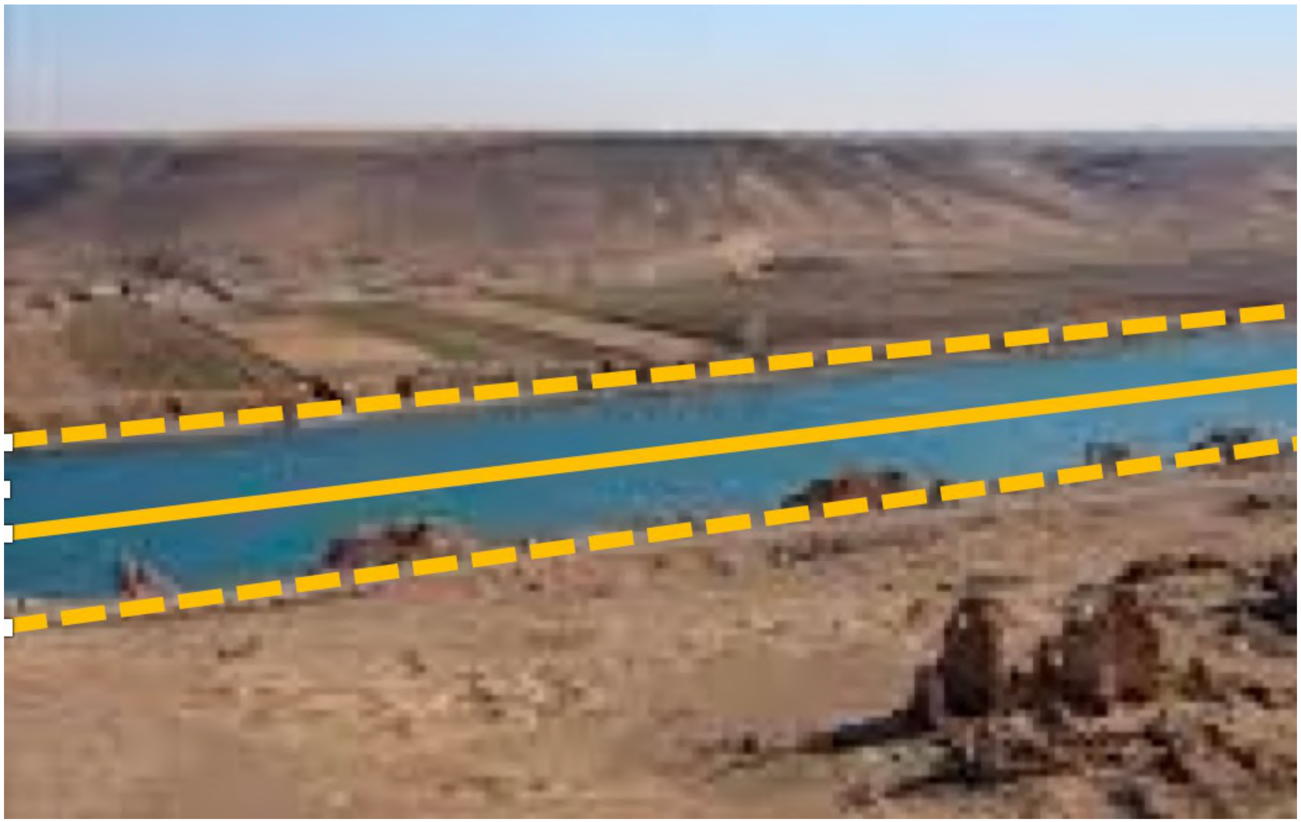
These examples are called the *support vectors*

$$y_i(w^T x_i + b) > 1 \Rightarrow \alpha_i = 0$$

$$\boxed{\begin{aligned} y_i(w^T x_i + b) < 1 &\Rightarrow \alpha_i = C \\ y_i(w^T x_i + b) = 1 &\Rightarrow 0 \le \alpha_i \le C \end{aligned}}$$



Why it called support vector machines?



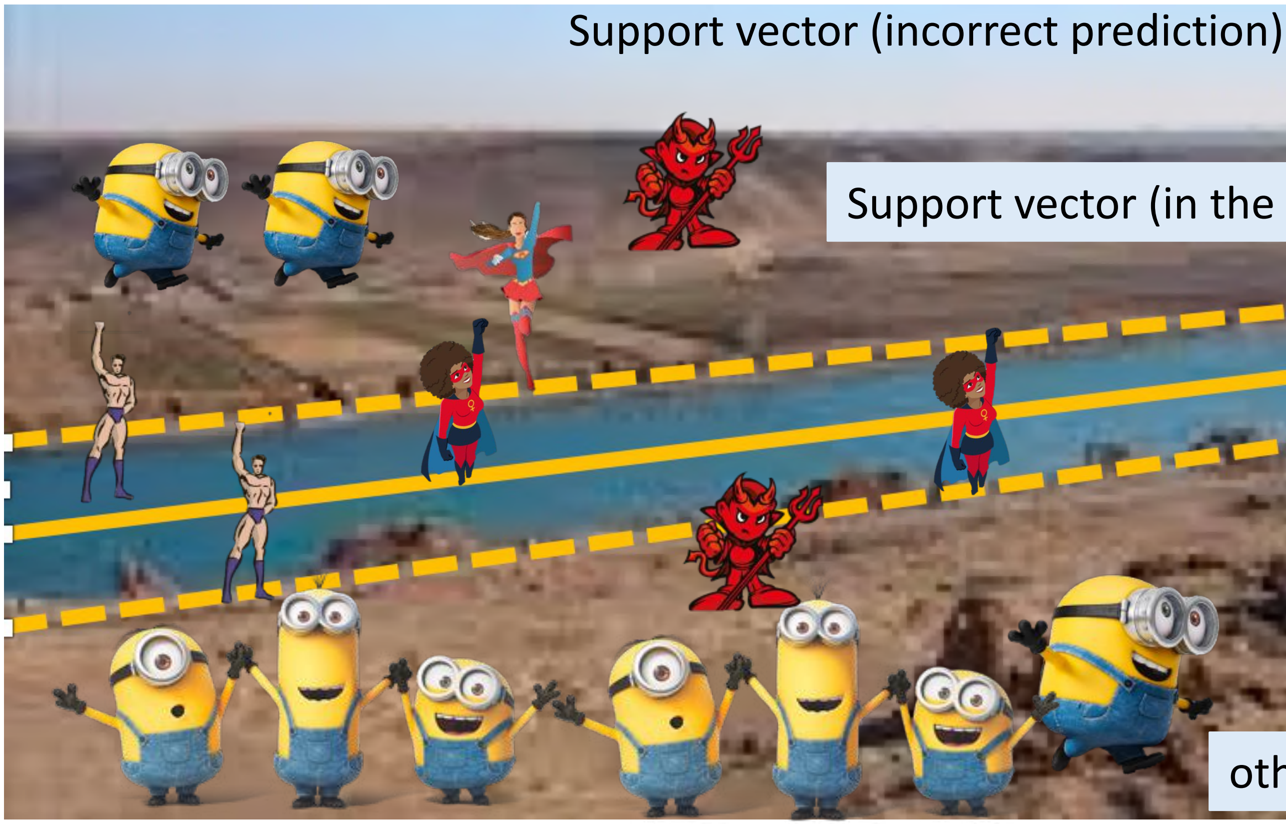
margin (upper)

Decision boundary

margin (lower)

Why it called support vector machines?

ohter vector
(correct samples
outside margin)



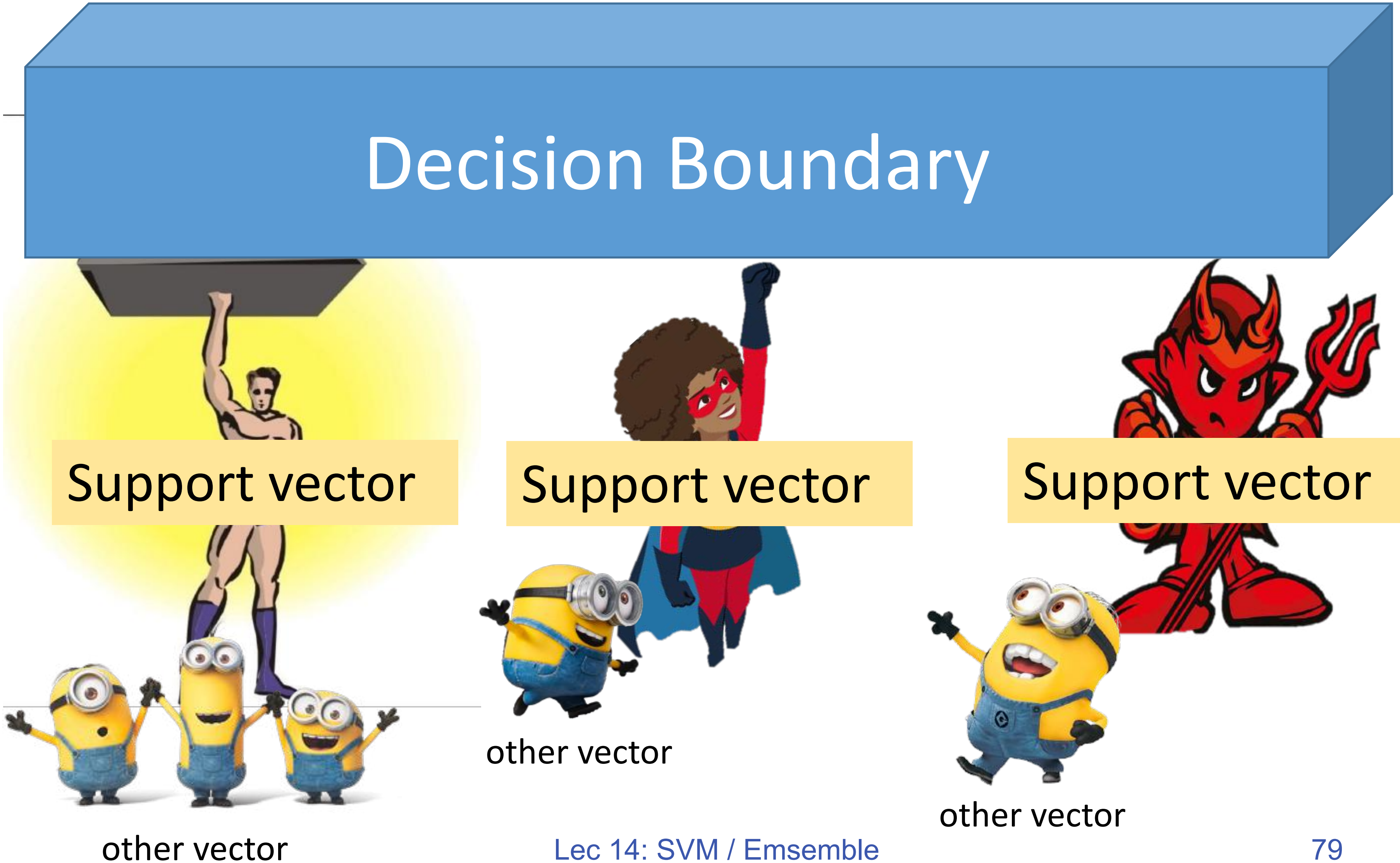
margin (upper)

Decision boundary

margin (lower)

Support vector (in the margin)

Support vector (incorrect prediction)


Decision Boundary
Support vector
Support vector
Support vector
other vector
other vector
other vector