Lecture 14: Support Vector Machines, Multi-Class Classification

Winter 2018

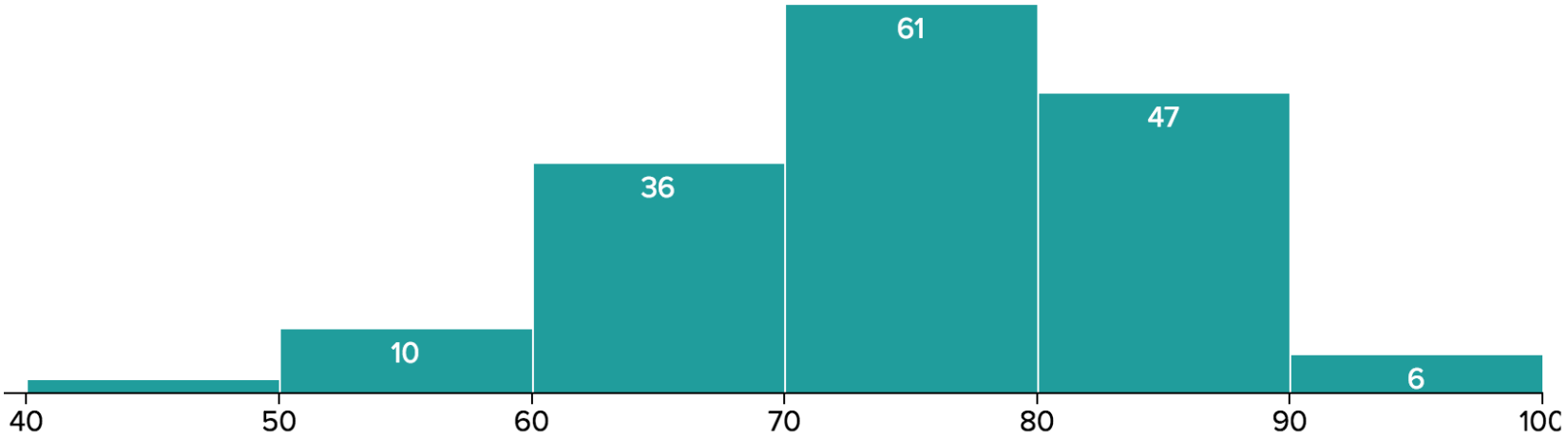
Kai-Wei Chang

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.

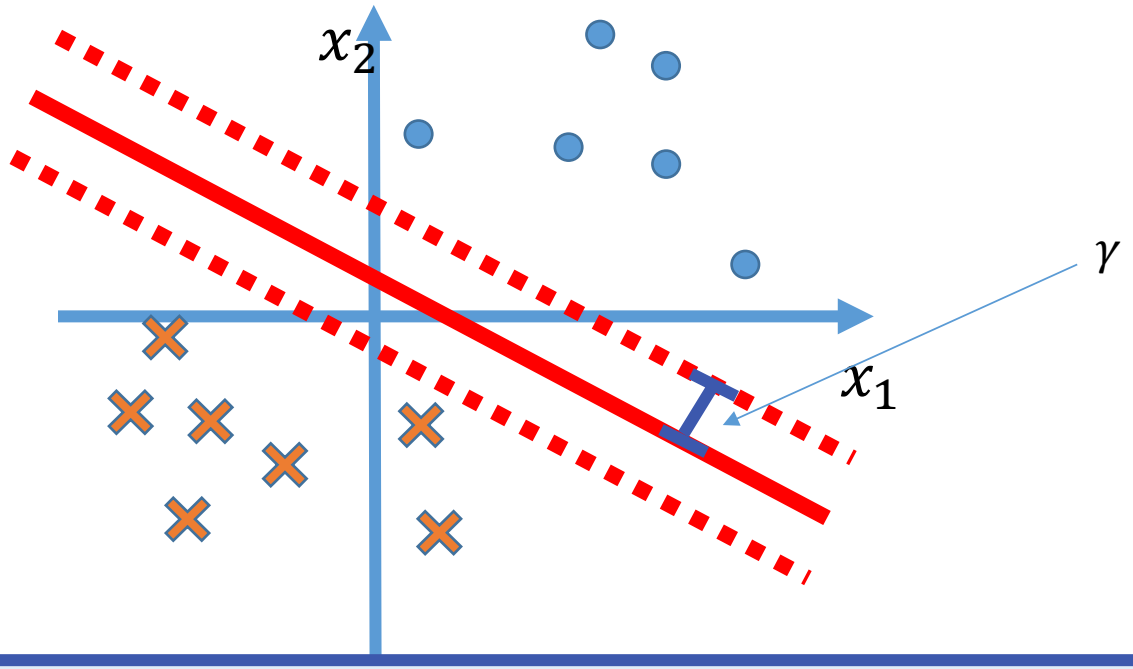
Announcement



MEDIAN	MAXIMUM	MEAN	STD DEV
75.25	100.0	74.57	9.86

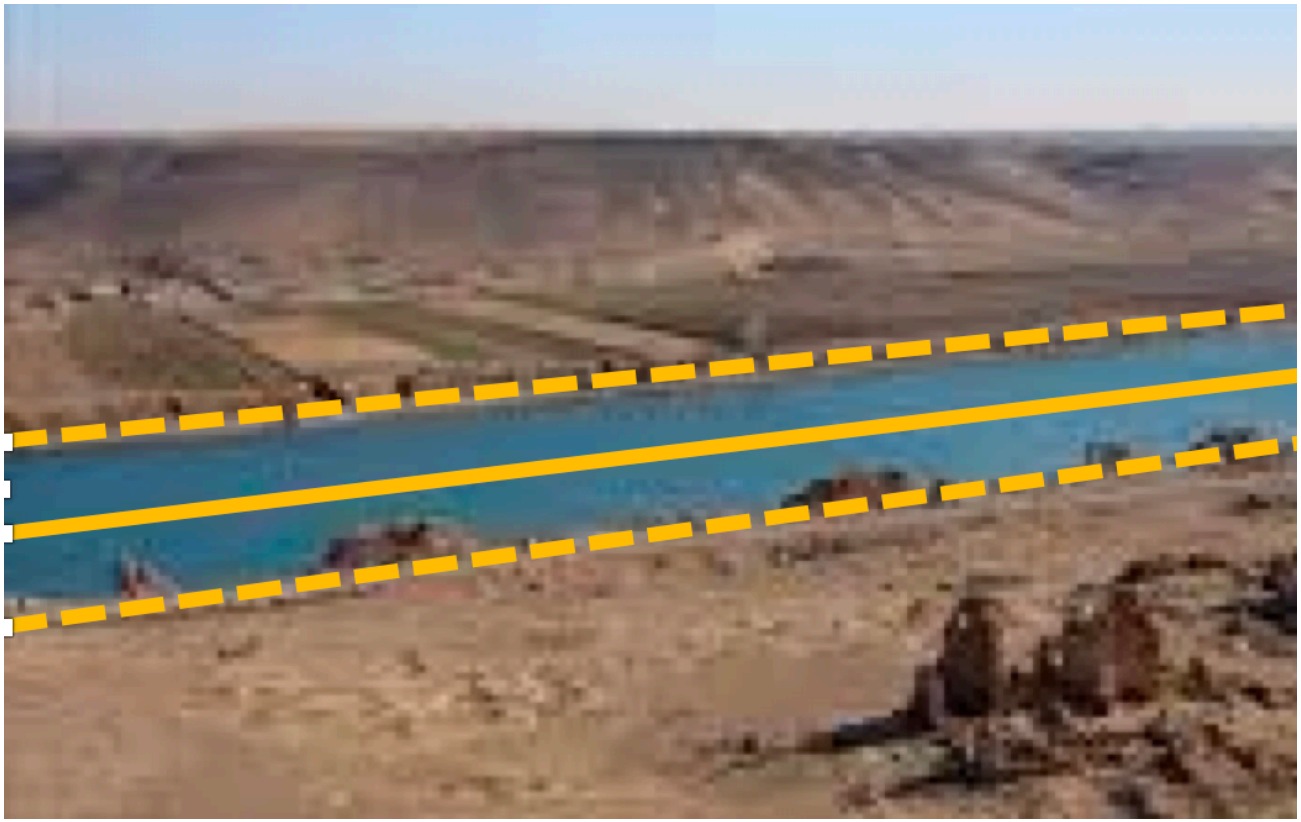
Support Vector Machine

Large Margin



Is there a way to find out the best γ automatically?

Why it called support vector machines?



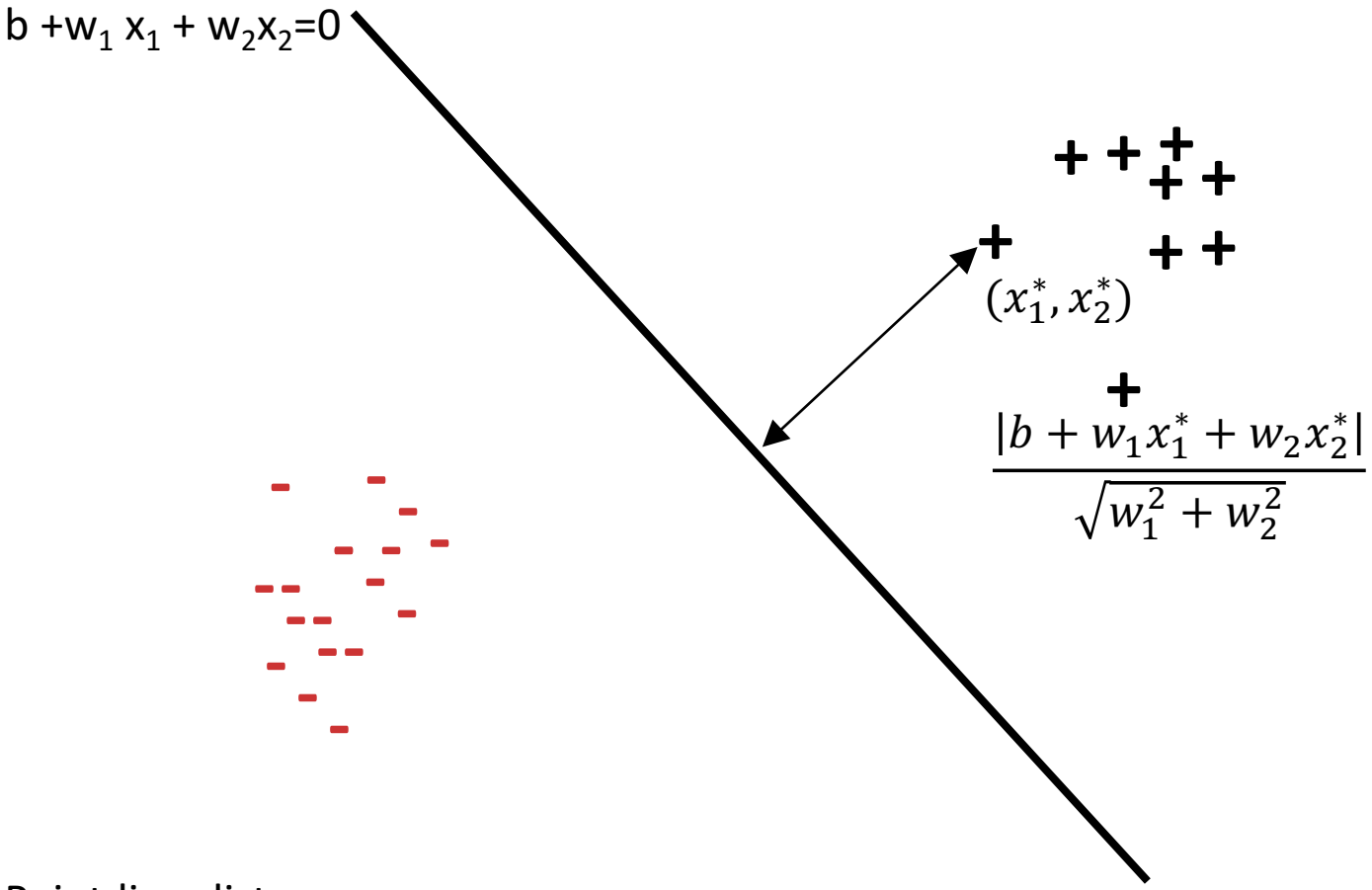
margin (upper)

Decision boundary

margin (lower)

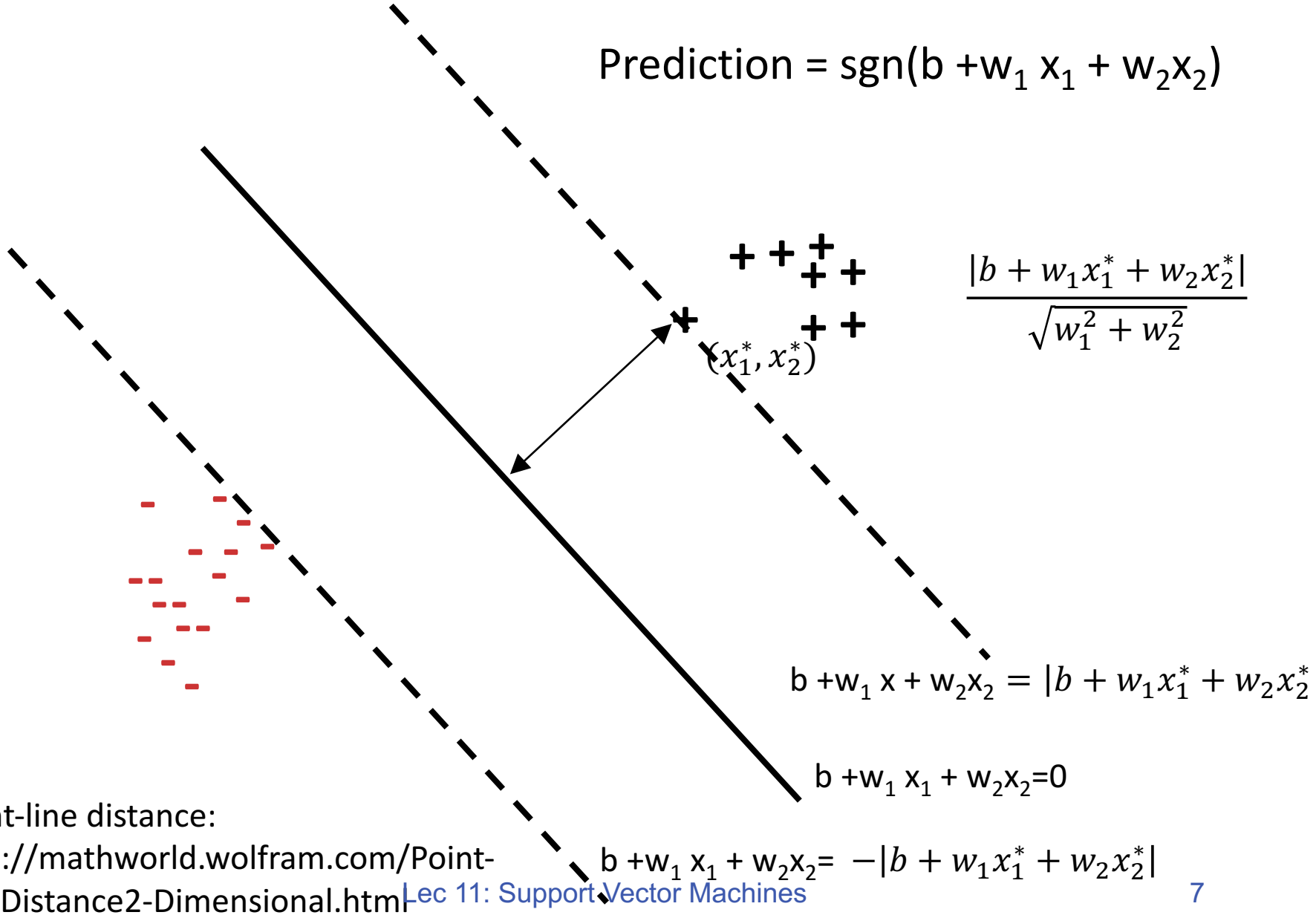
Recall: The geometry of a linear classifier

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



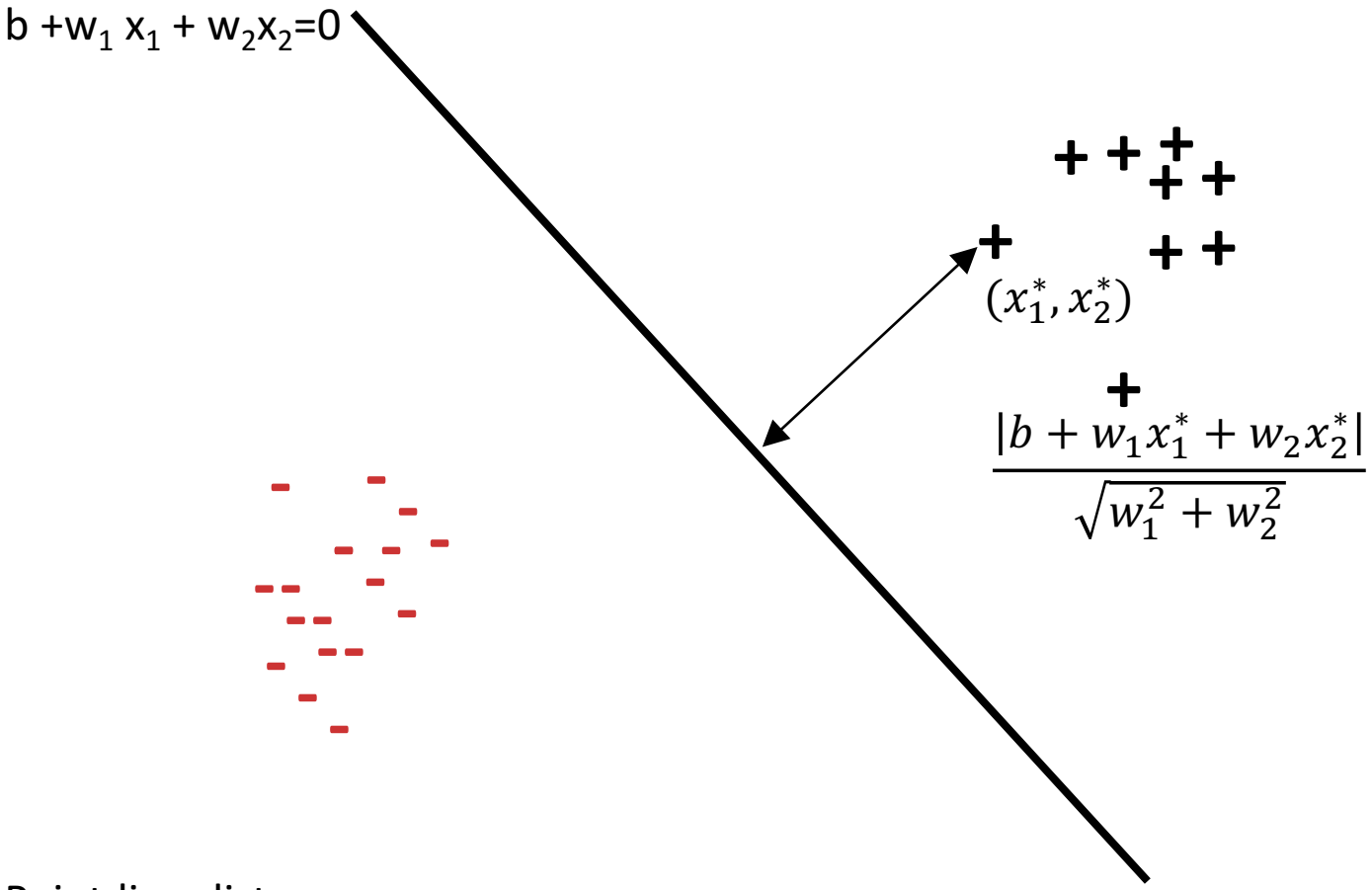
Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

This lecture: Support vector machines

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

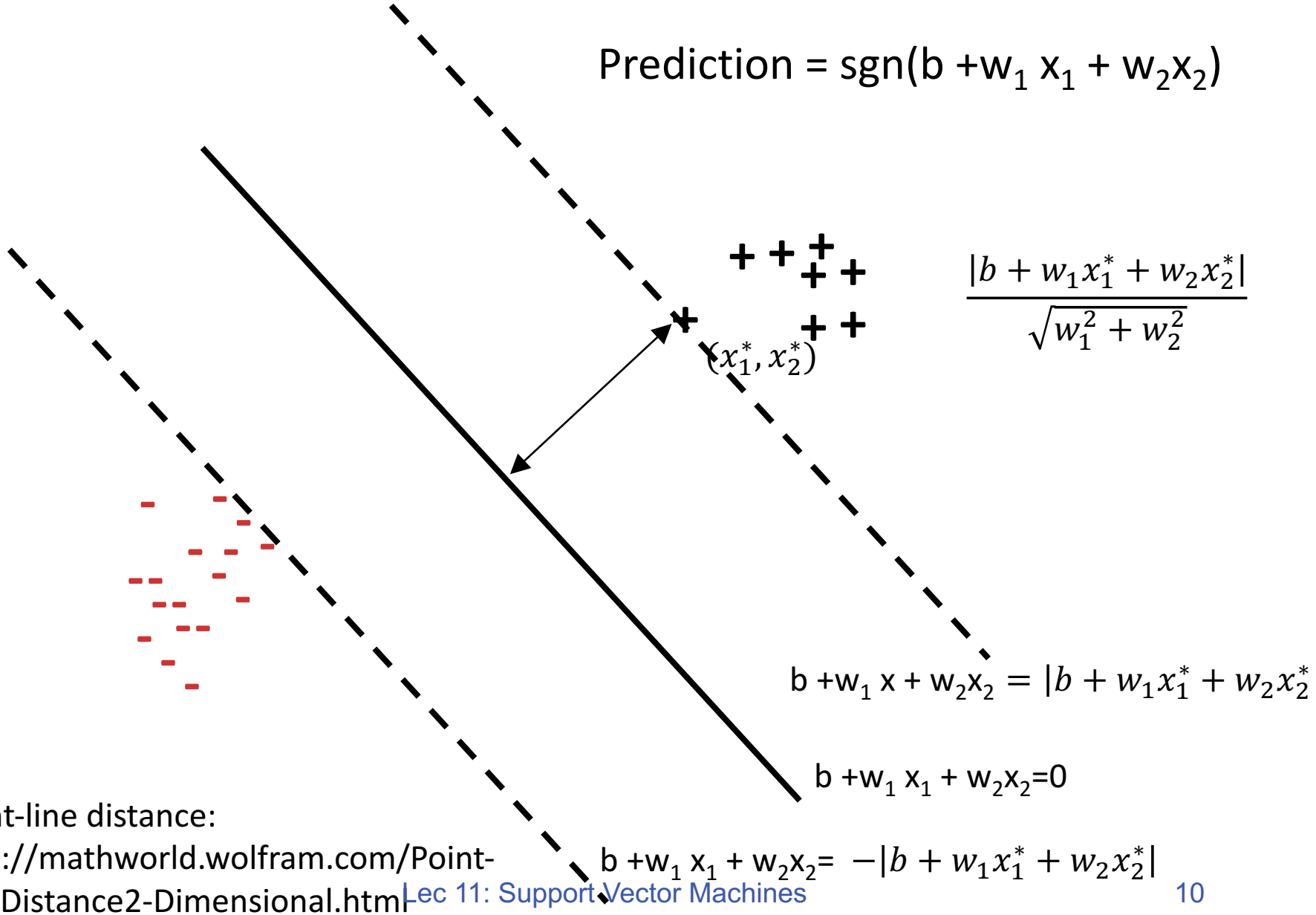
Recall: The geometry of a linear classifier

Prediction = sgn(b +w_1 x_1 + w_2x_2)



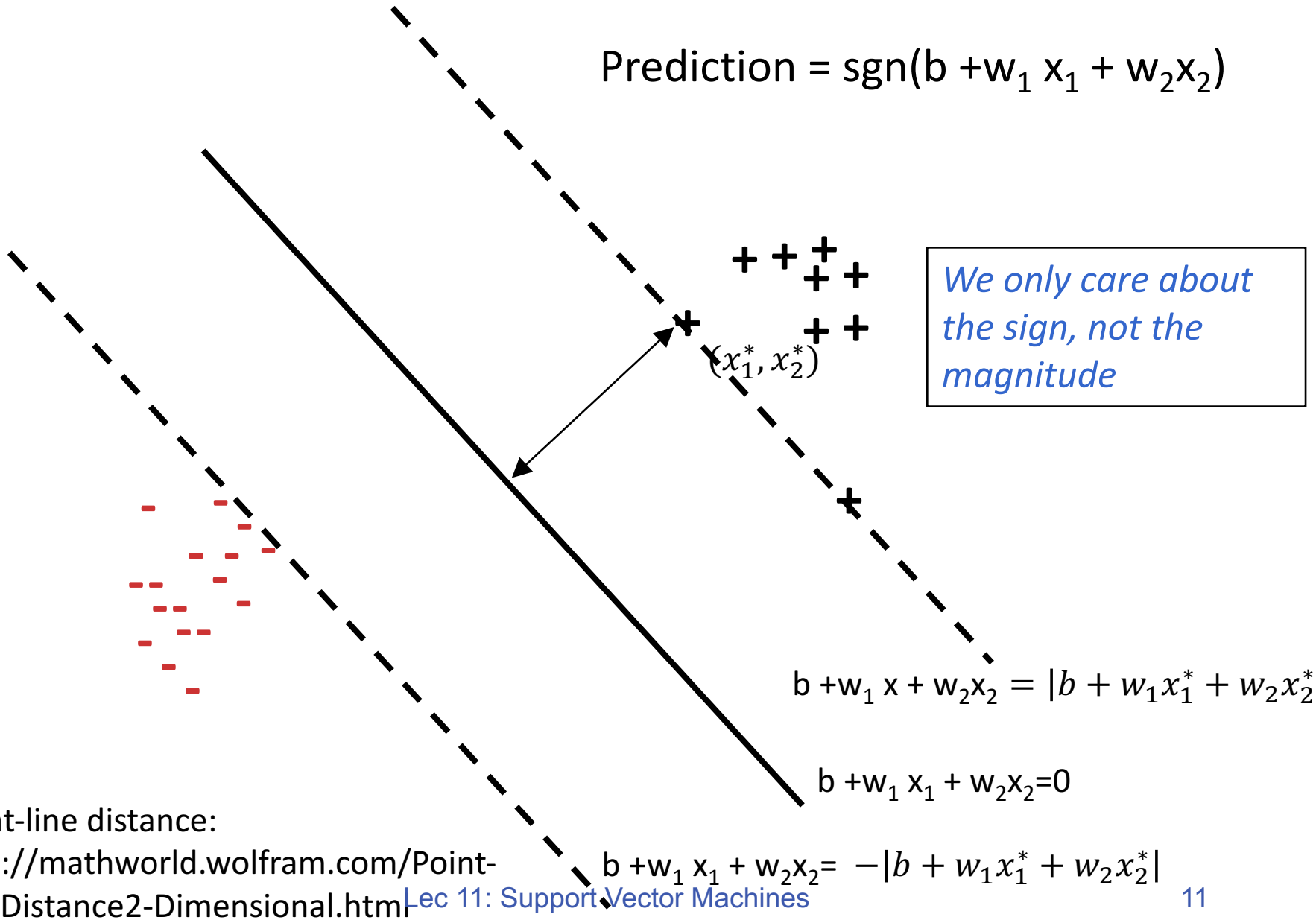
Point-line distance:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Recall: The geometry of a linear classifier



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

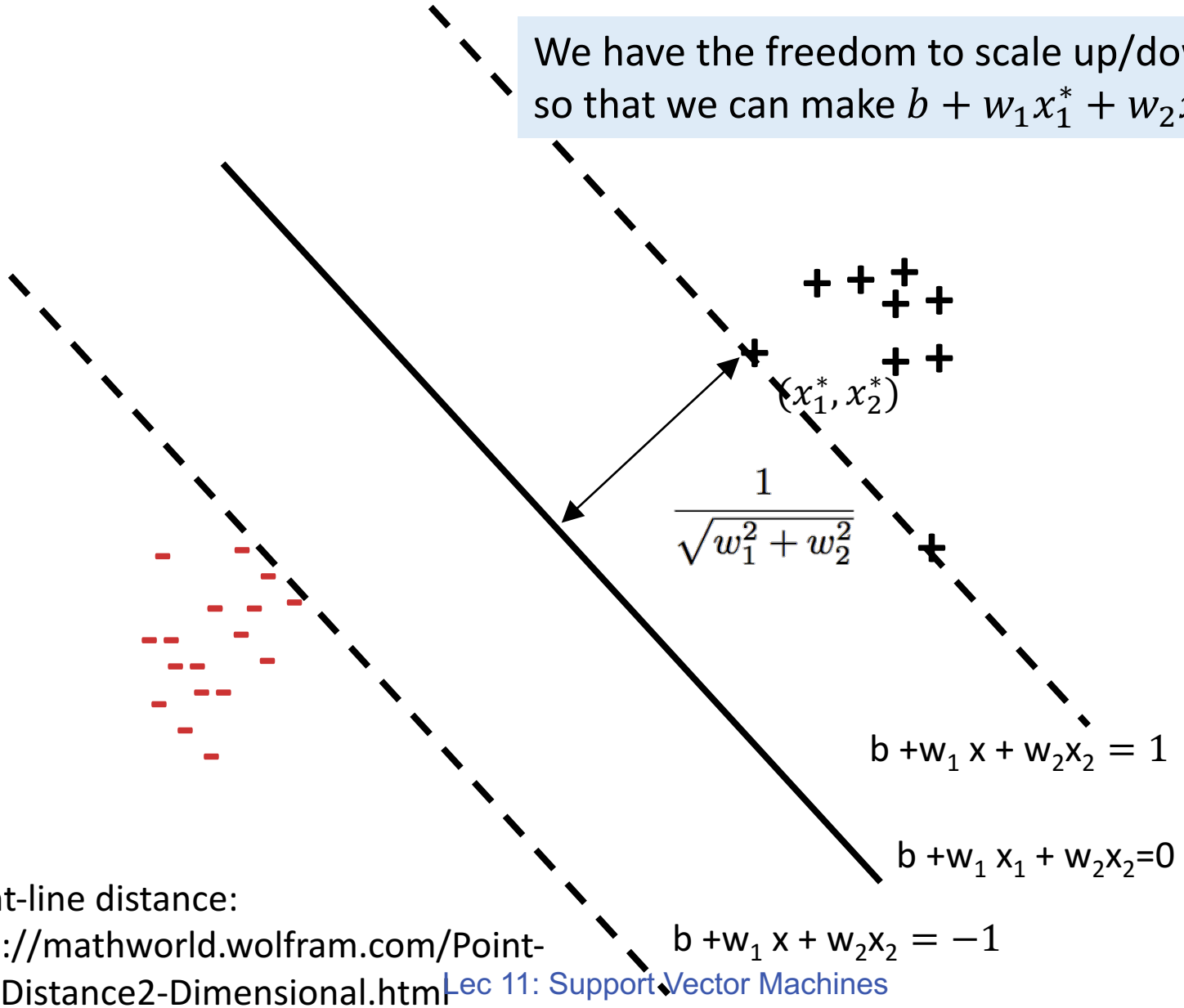
Recall: The geometry of a linear classifier



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

Recall: The geometry of a linear classifier

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

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

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

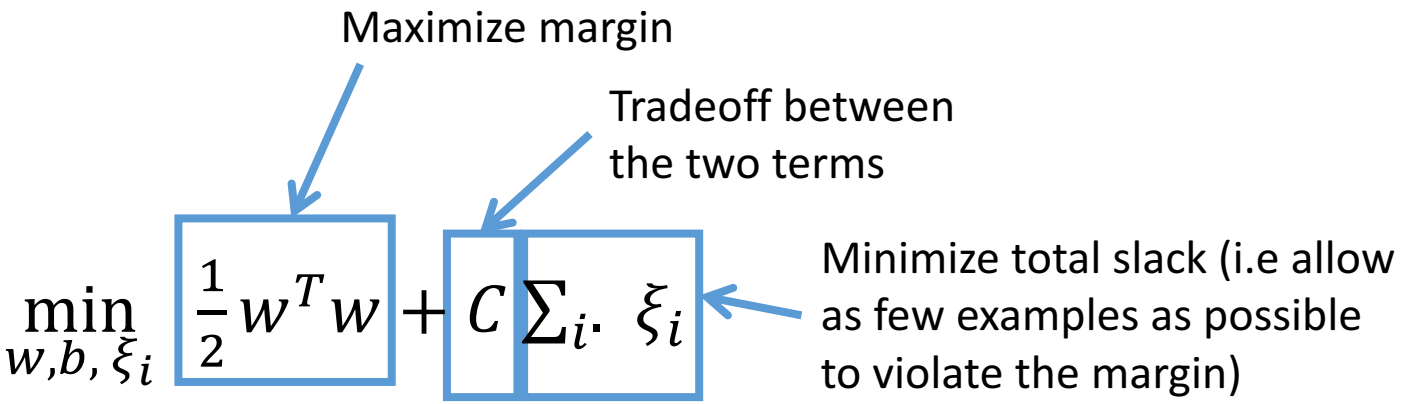
- New optimization problem for learning

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

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

C is the hyper-parameter

Soft SVM

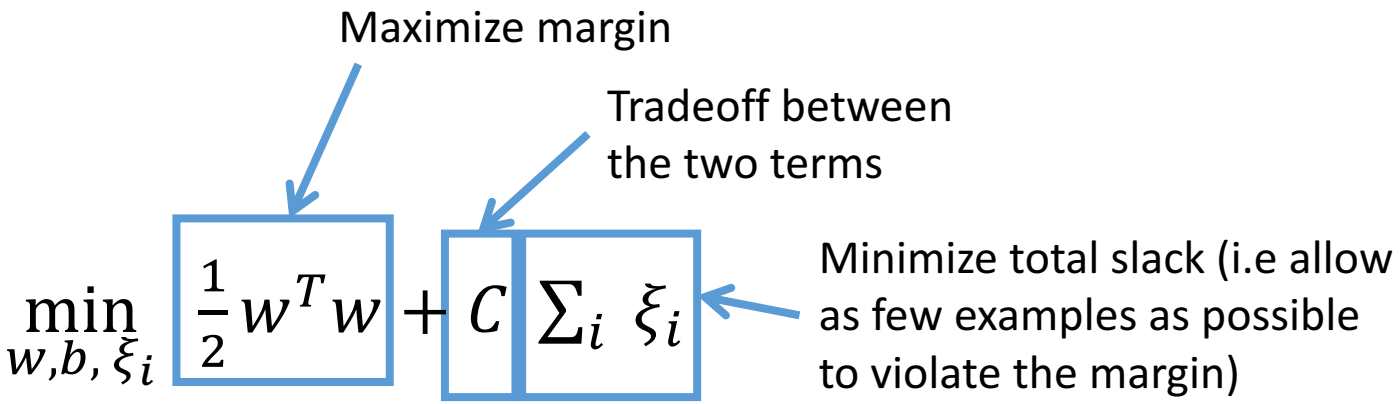


$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

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

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

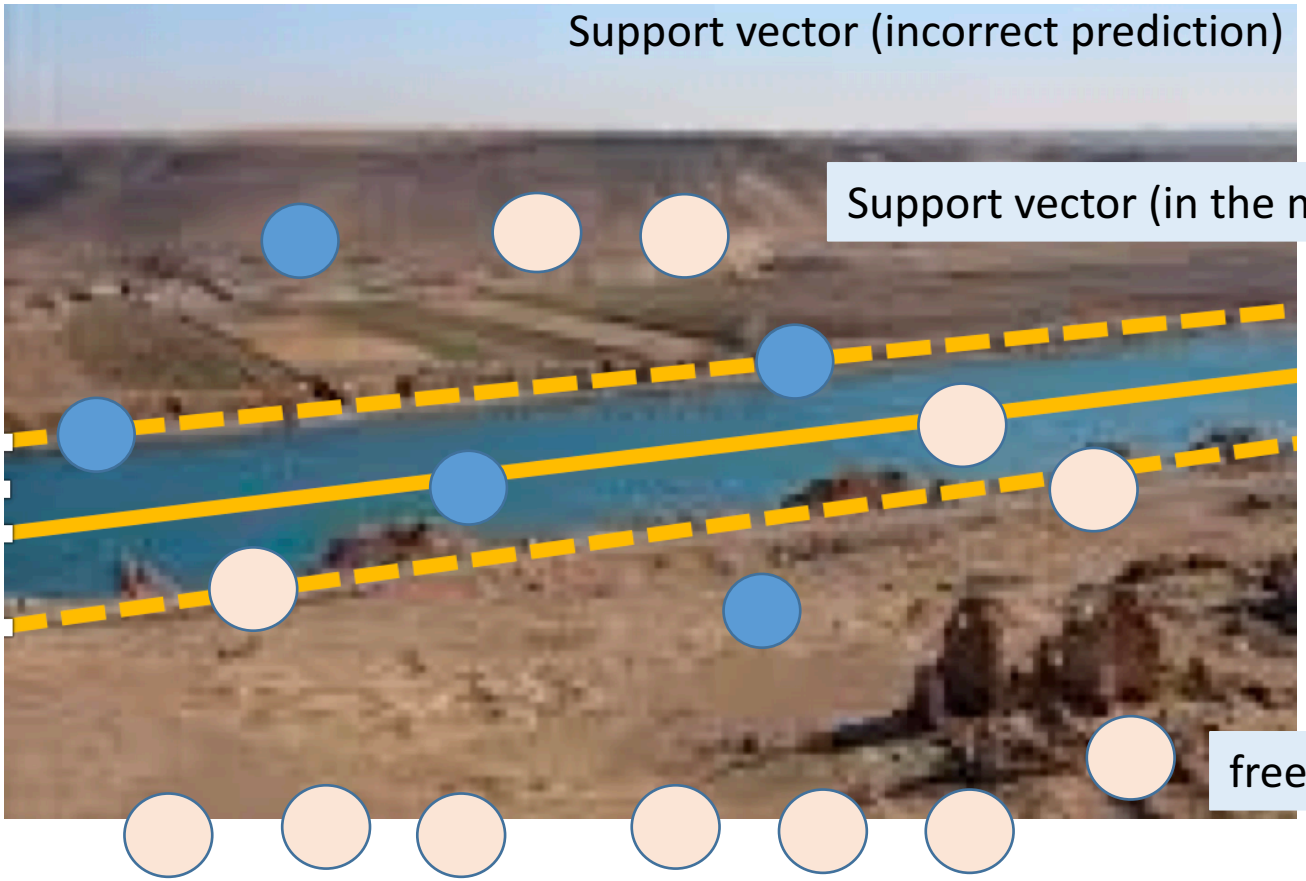
We can consider three cases

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

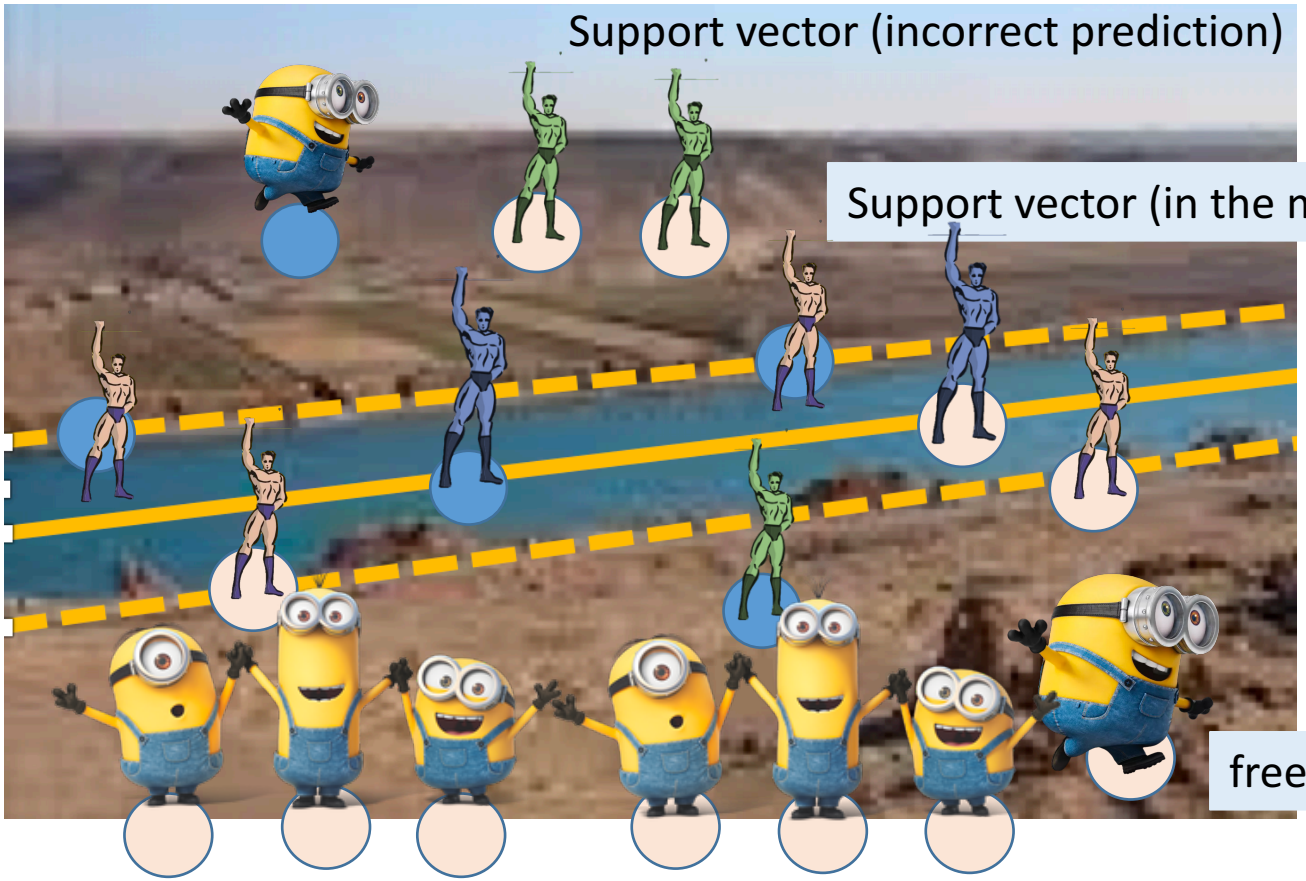
This is the hinge loss function

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

Recap: support vector machines

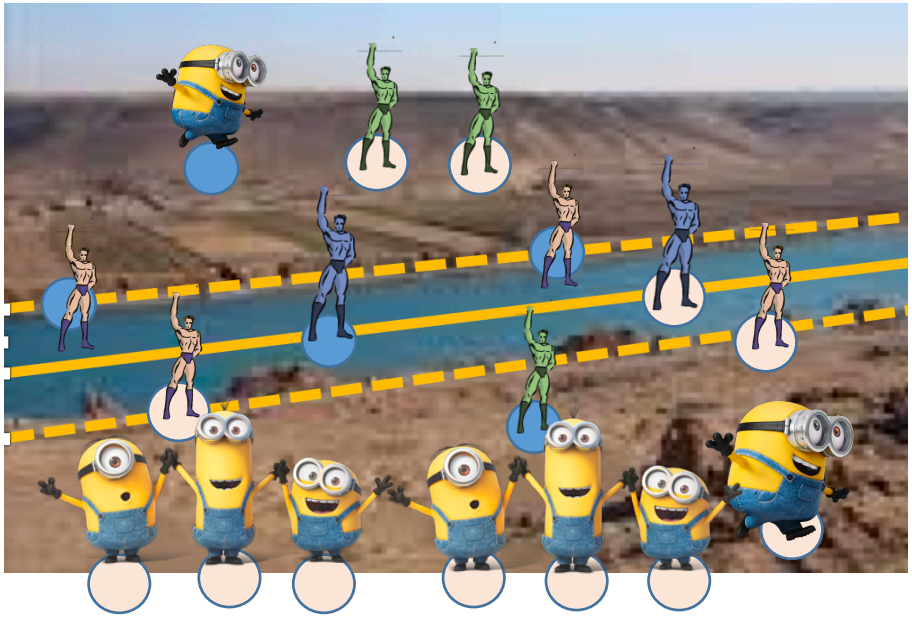


Recap: support vector machines



Recap: support vector machines

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



Match-up quiz

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

$b)\ y_i(w^T x_i + b) < 1$

$c)\ y_i(w^T x_i + b) > 1$

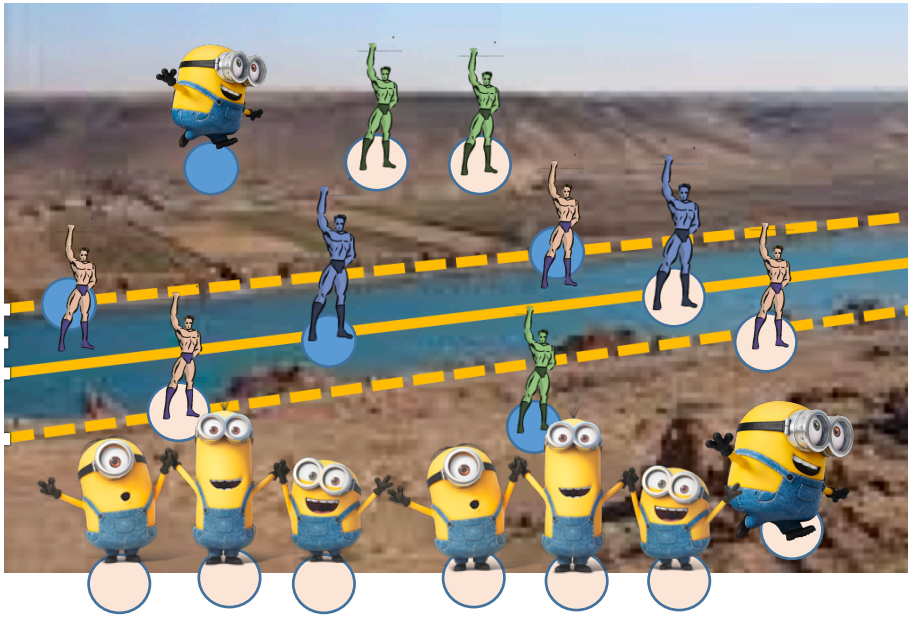
Instances with correct prediction

Instances on the Decision boundary

Instances inside the margin or on the wrong side

Recap: support vector machines

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



Match-up quiz

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

$b)\ y_i(w^T x_i + b) < 1$

$c)\ y_i(w^T x_i + b) > 1$

$c)\ y_i(w^T x_i + b) > 1$

Instances with correct prediction

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

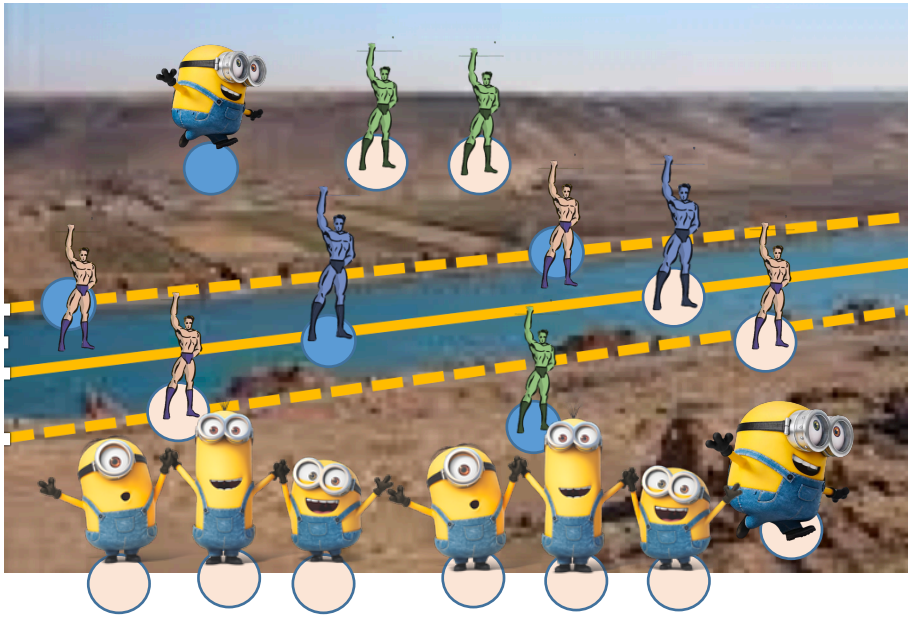
Instances on the Decision boundary

$b)\ y_i(w^T x_i + b) < 1$

Instances inside the margin or on the wrong side

Recap: support vector machines

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



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

Match-up quiz

$a) \ 0 < \xi < 1$

b) $\xi = 0$

C) $\xi > 1$

Instances with correct prediction

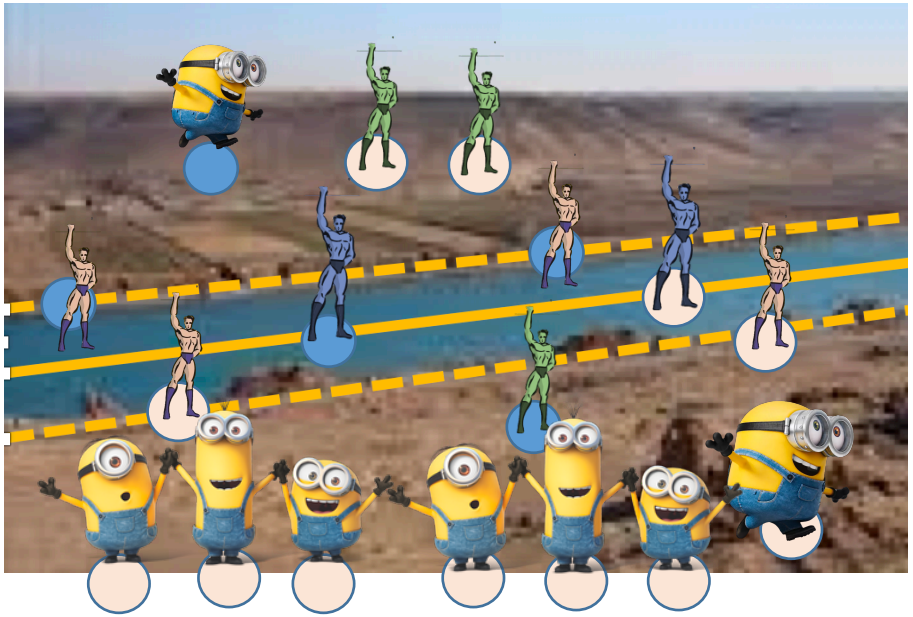
Instances on the Decision boundary

Instances inside the margin or on the wrong side

Recap: support vector machines

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

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



Match-up quiz

$a) \ 0 < \xi < 1$

b) $\xi = 0$

C) $\xi > 1$

b) $\xi = 0$

Instances with correct prediction

b) $\xi = 0$

Instances on the Decision boundary

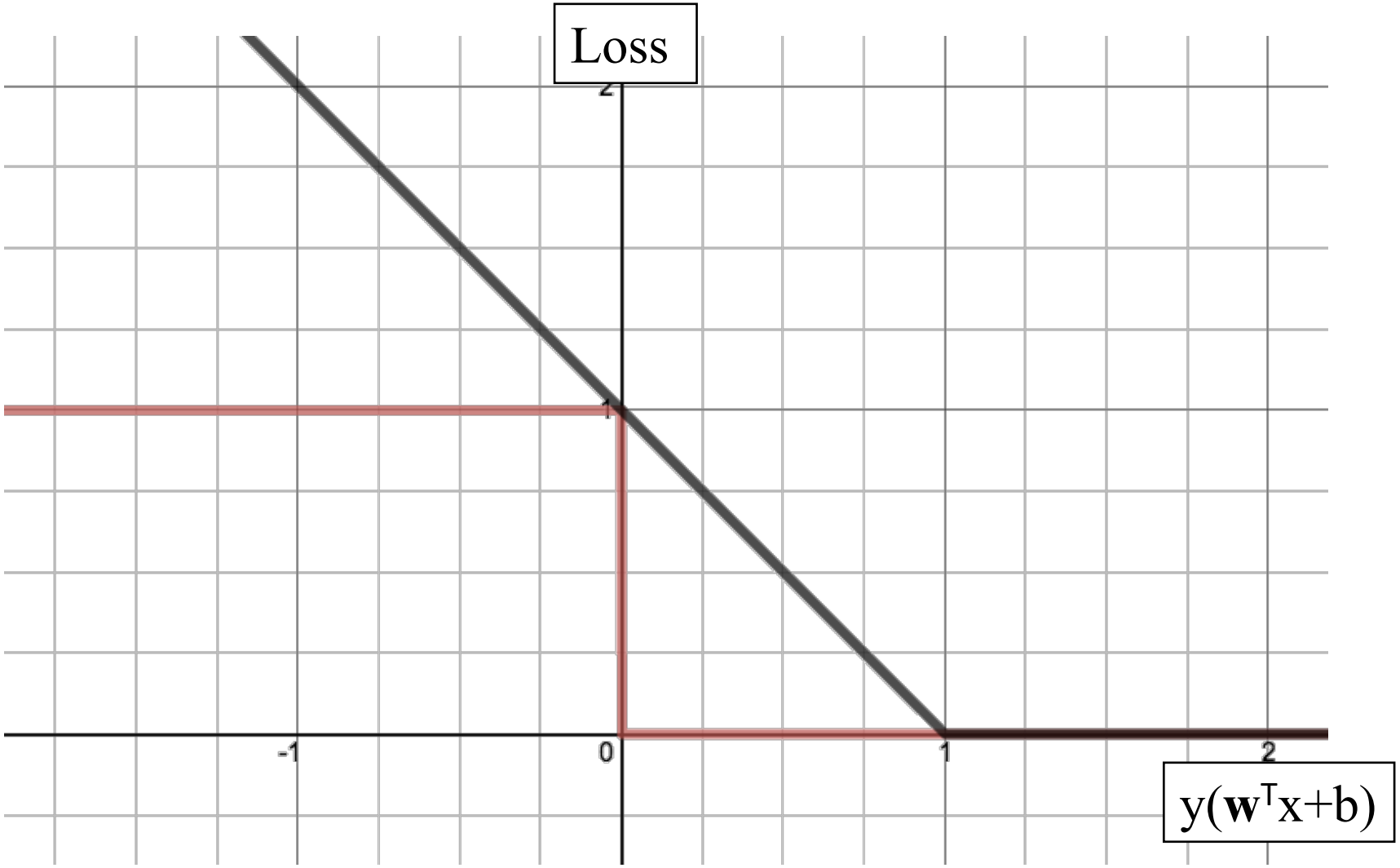
$a) \ 0 < \xi < 1$

Instances inside the margin

C) $\xi > 1$

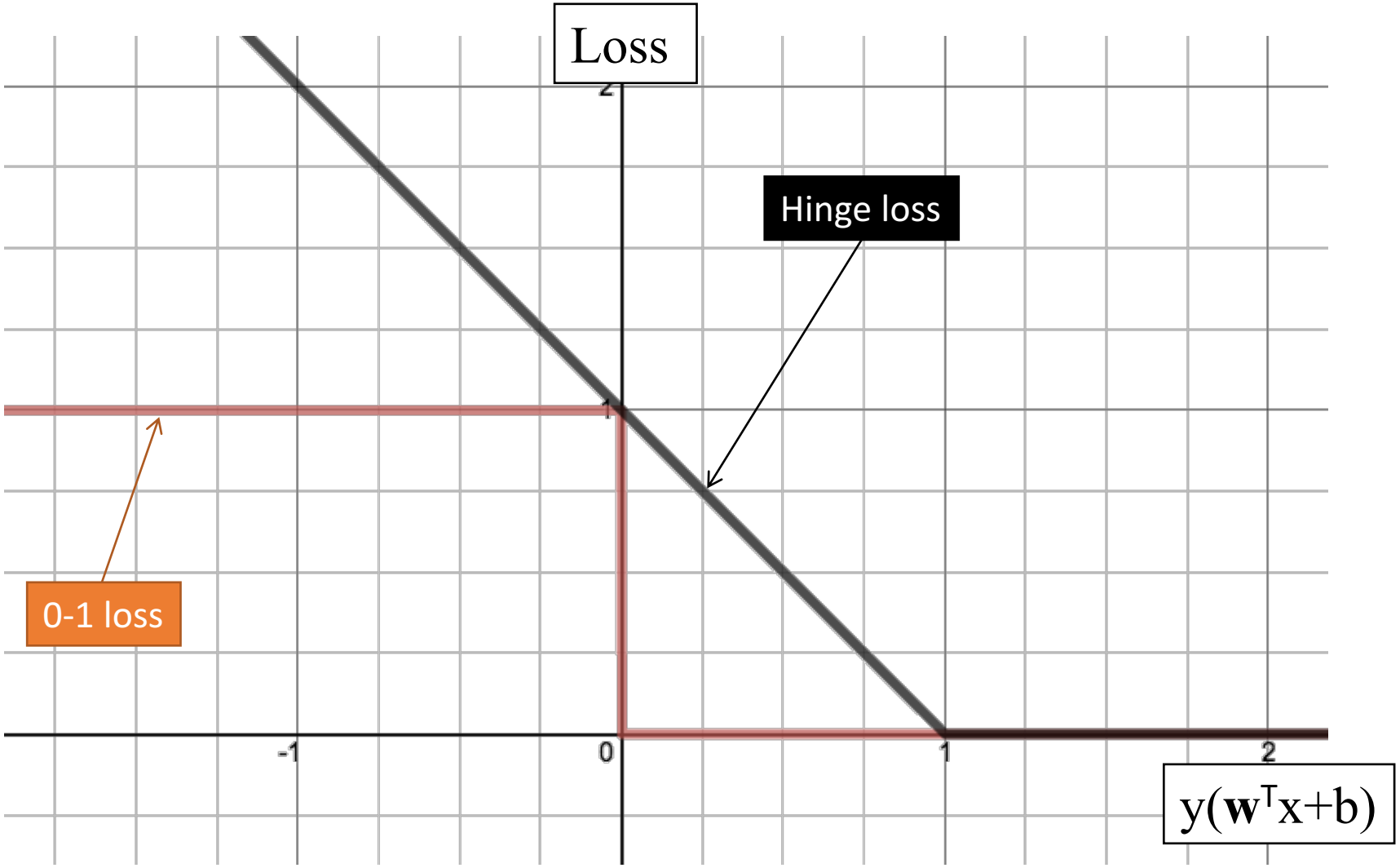
Training error

The Hinge Loss



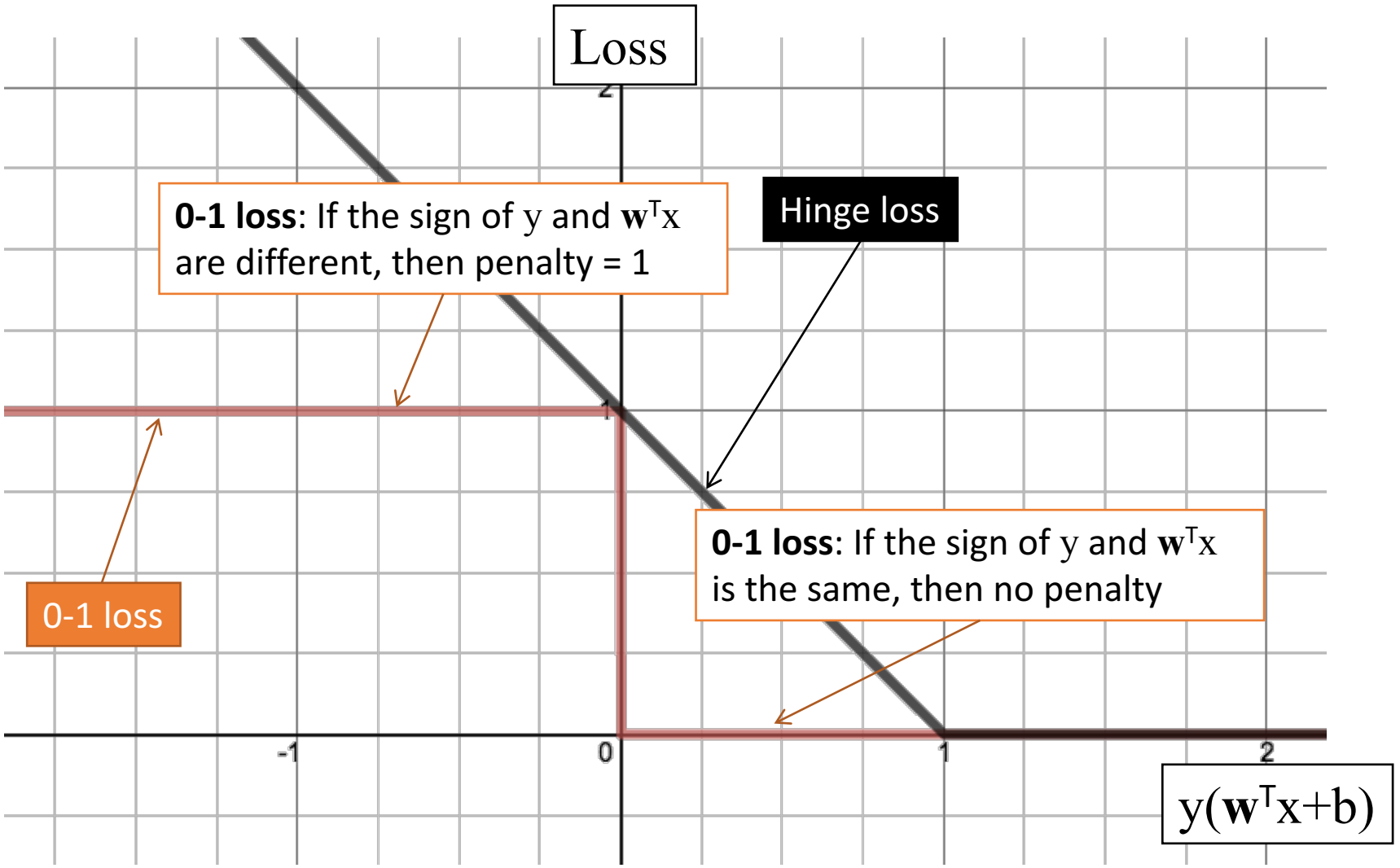
The Hinge Loss

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



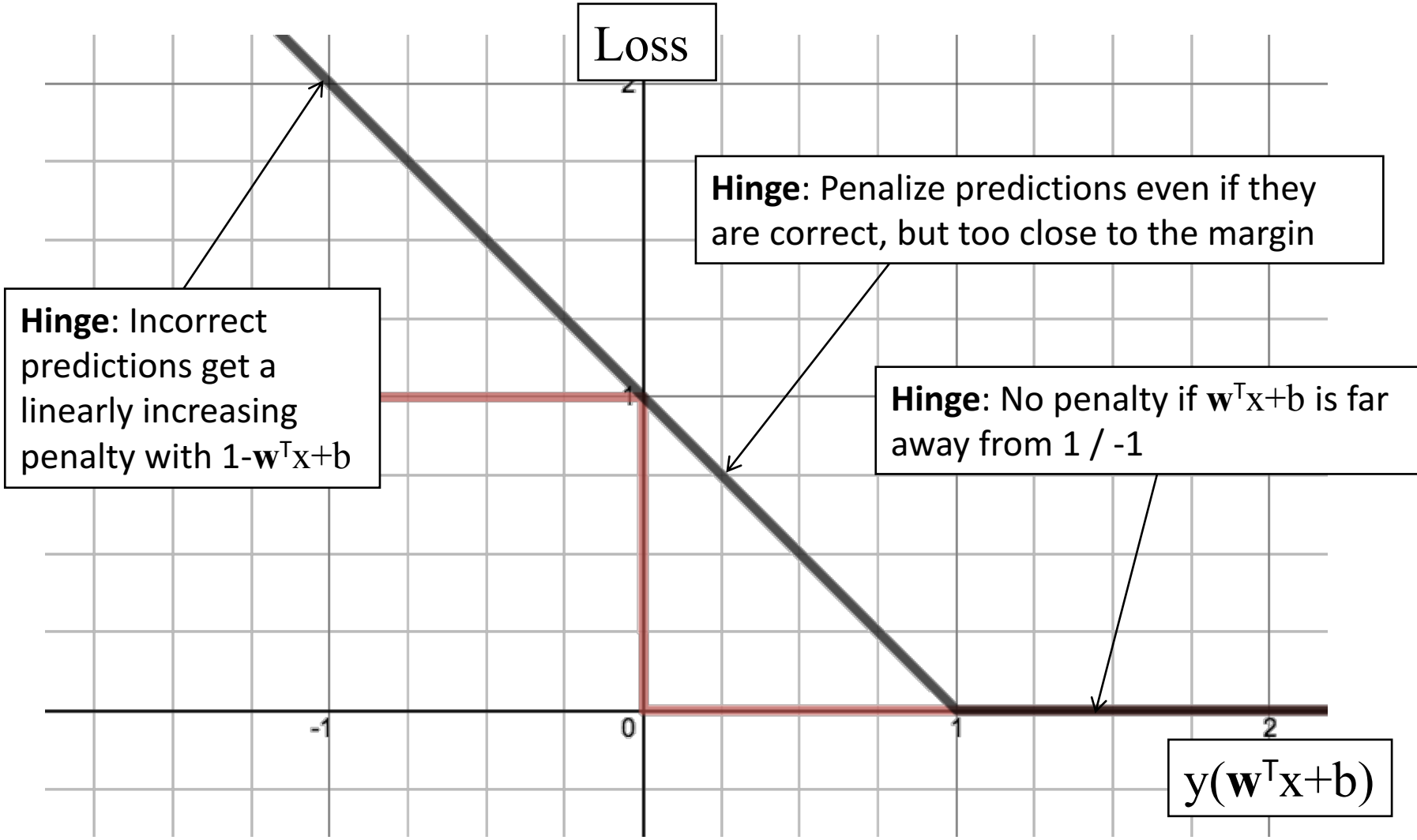
The Hinge Loss

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



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

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

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!

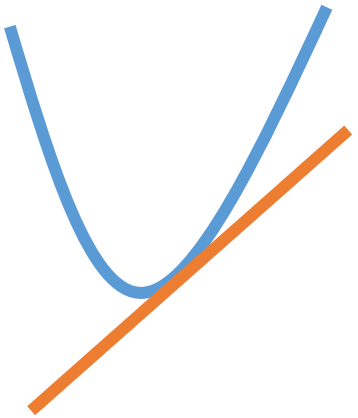
What is the derivative of the hinge loss with respect to w?

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

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

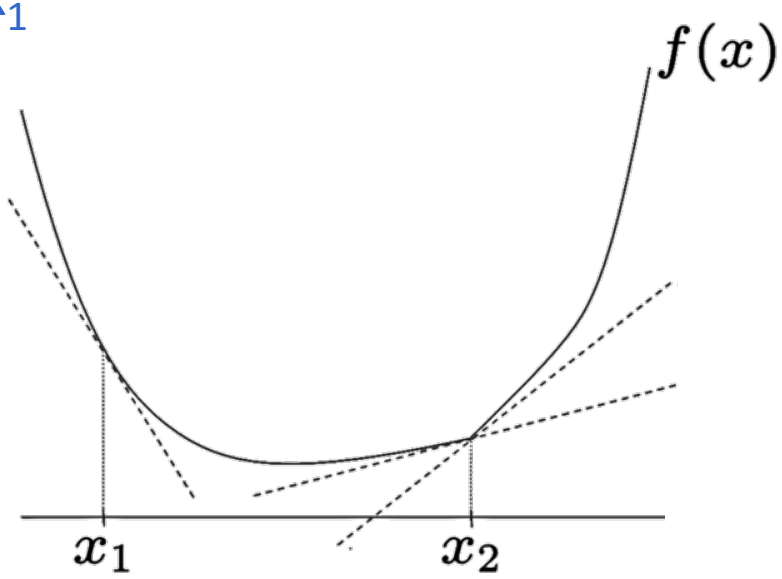
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

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]

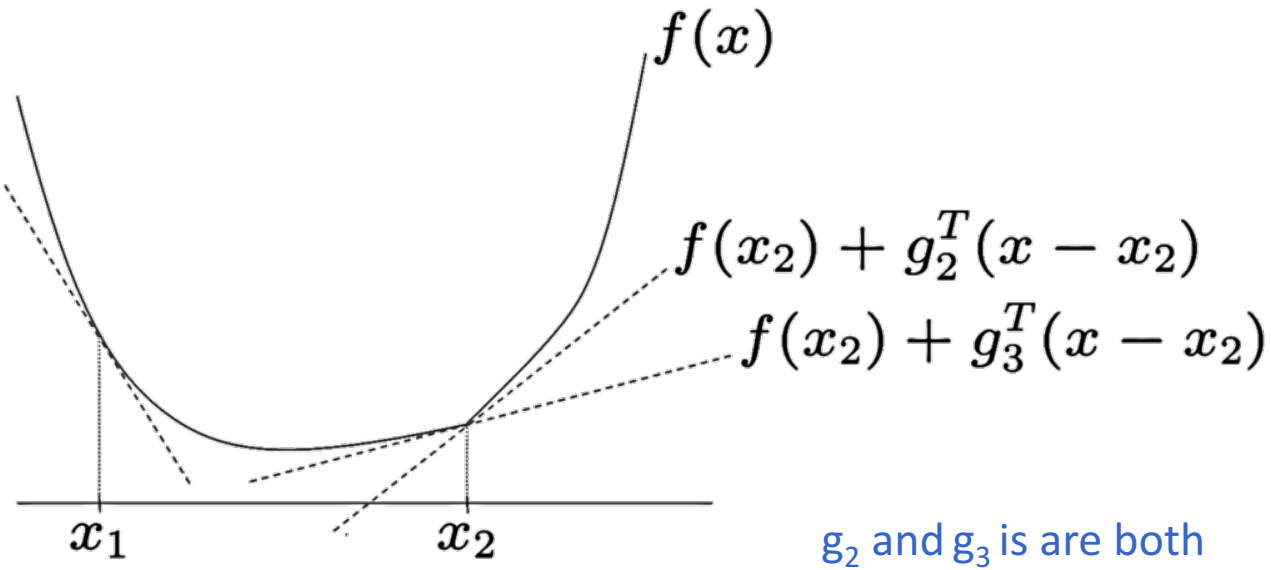
Sub-gradients

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

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

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

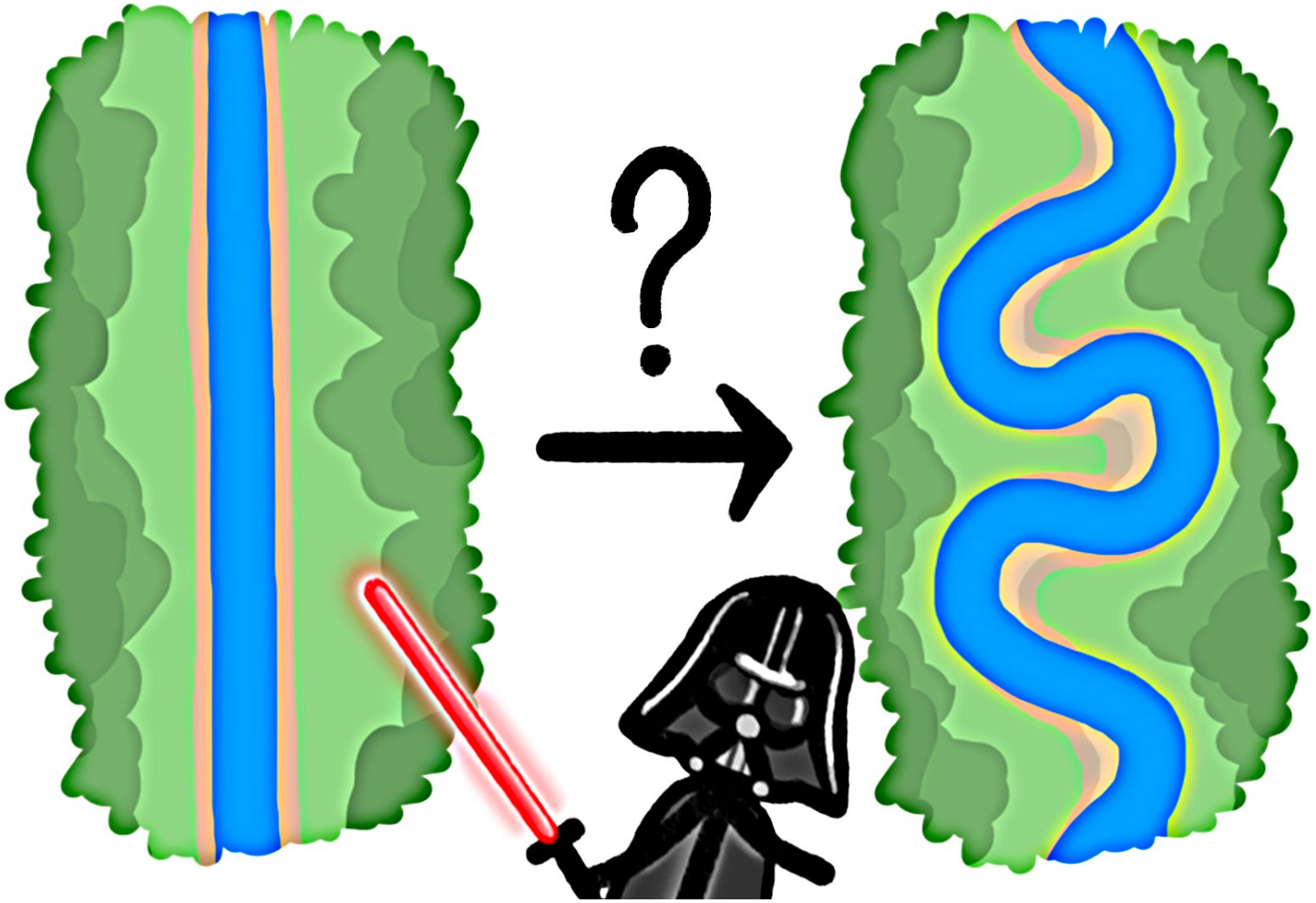
- SVM optimizes the hinge loss
  - With regularization

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

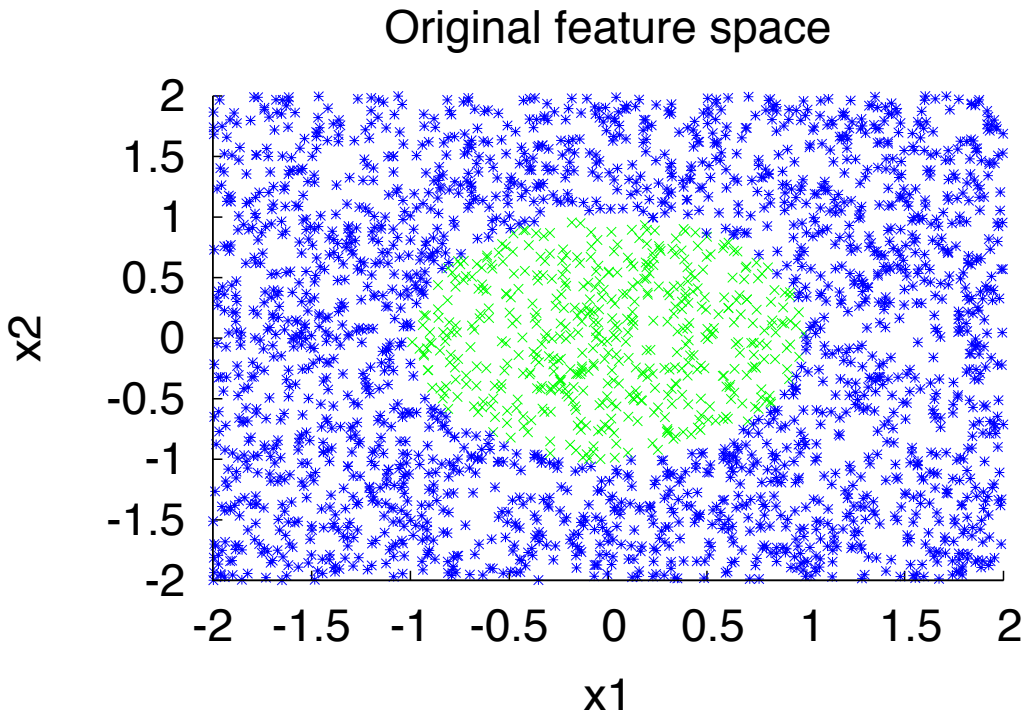
This lecture: Support vector machines

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

How about non-linearly separated data?

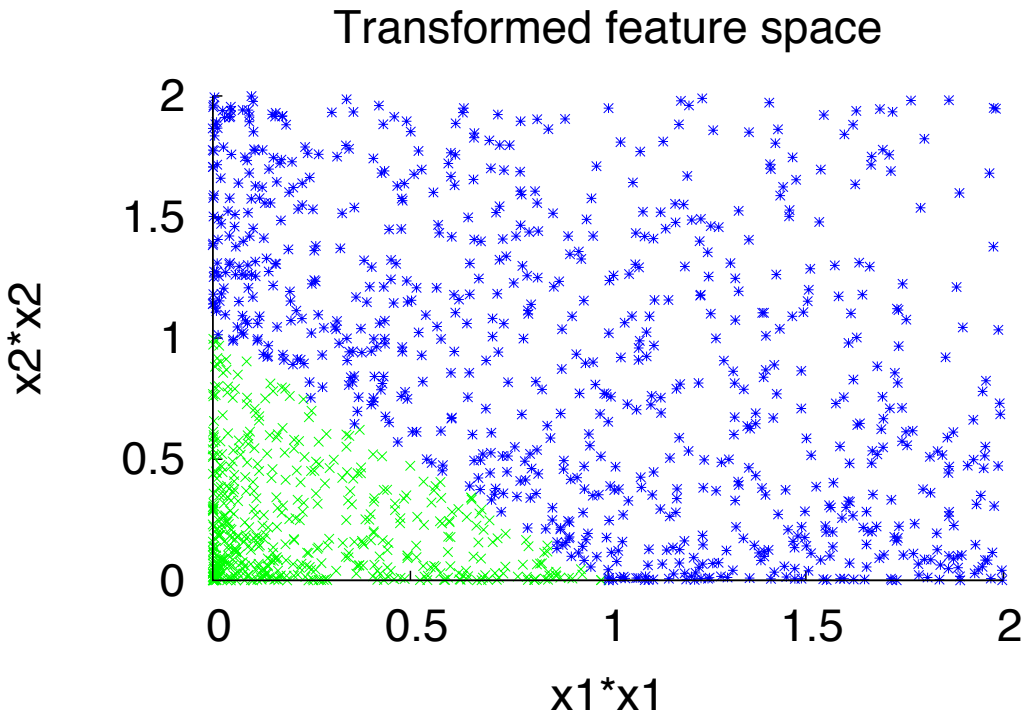


Recap: Making data linearly separable



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Recap: Making data linearly separable



Transform data: $\mathbf{x} = (x_1, x_2)$ => $\mathbf{x'} = (x_1^2, x_2^2)$

$f(\mathbf{x'}) = 1$ iff $x'_1 + x'_2 \leq 1$

Can we map data to very high dimensional space?

- Yes – we can map input to an infinite dimensional space
- For example in RBF kernel:

$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T$$

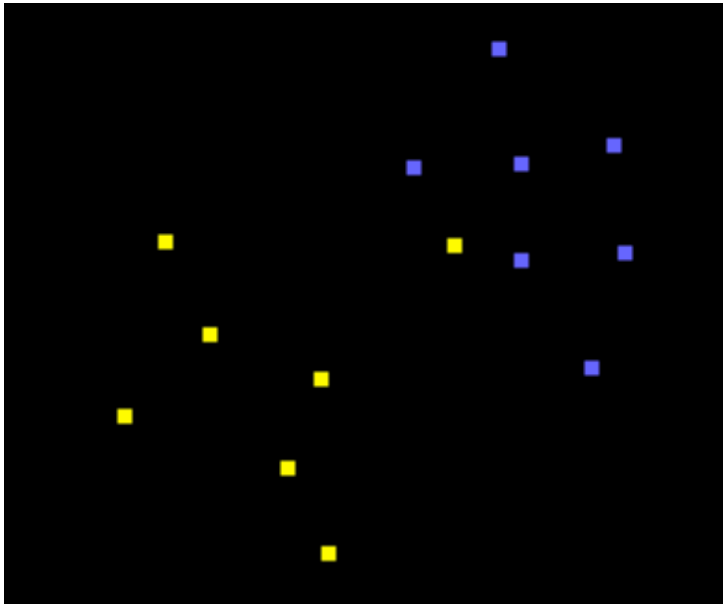
(γ: a hyper-parameter)

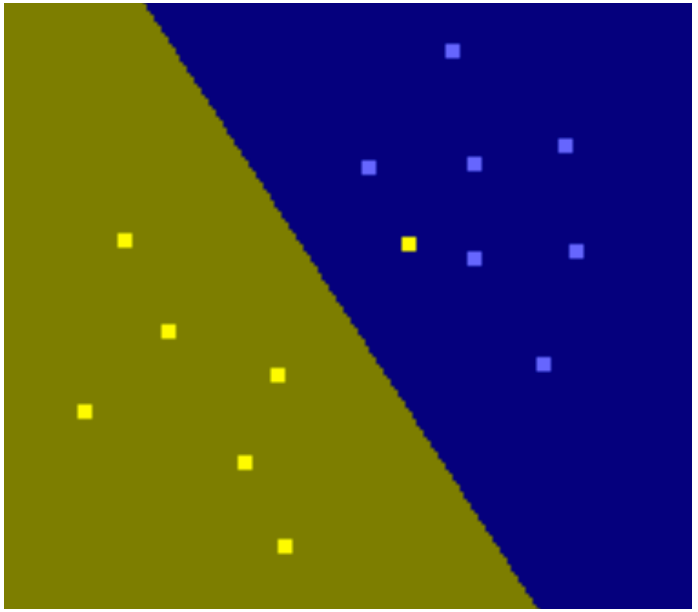
How can we learn a model in an infinite dimensional space?

DEMO – SVM

- Linear Kernel (C= 1)

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



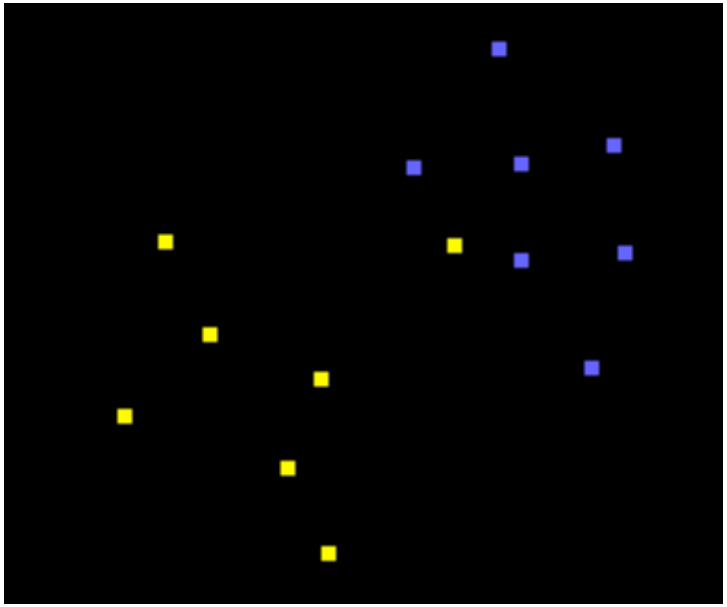


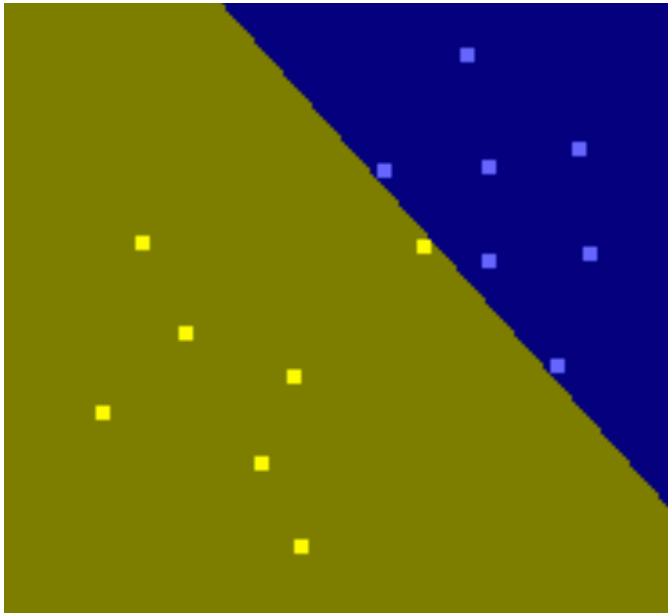
https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

Example – SVM

- Linear Kernel (C= 10000)

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

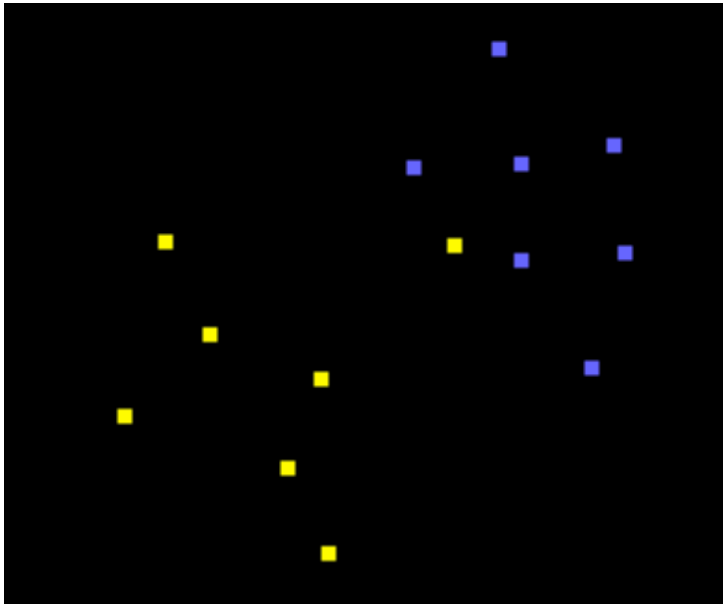


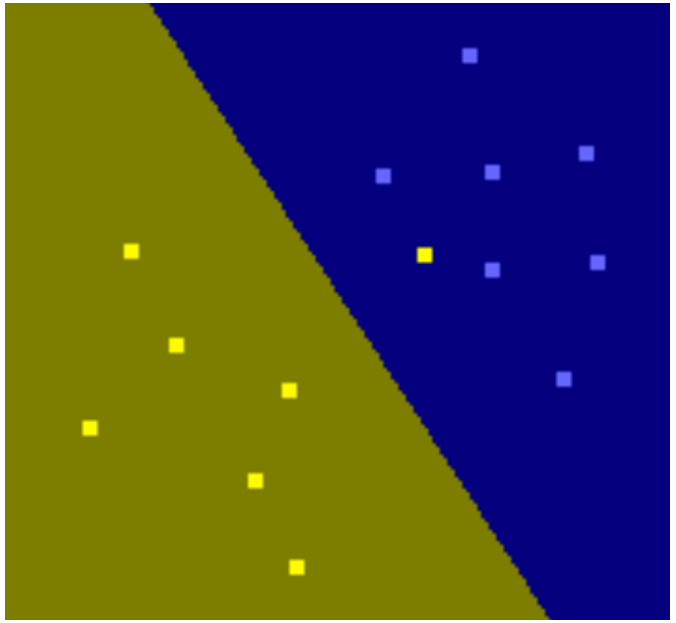


Example – SVM

- RBF Kernel (C= 1 $\gamma = 0.01$)

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

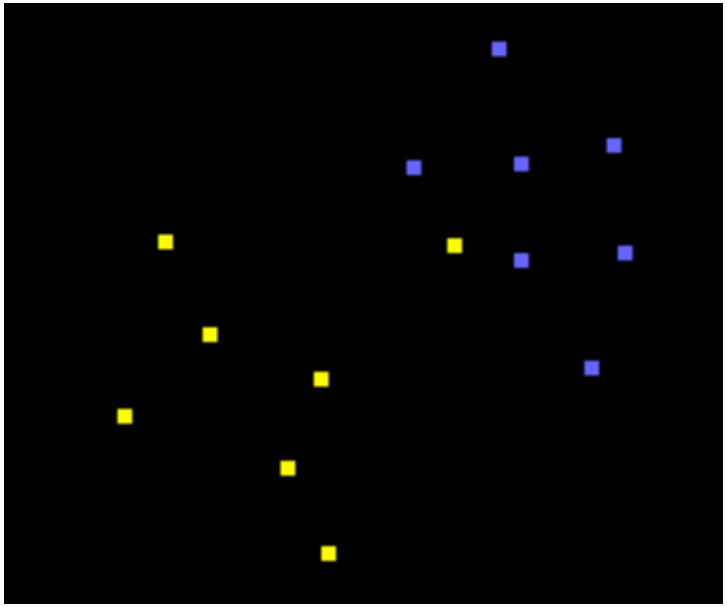


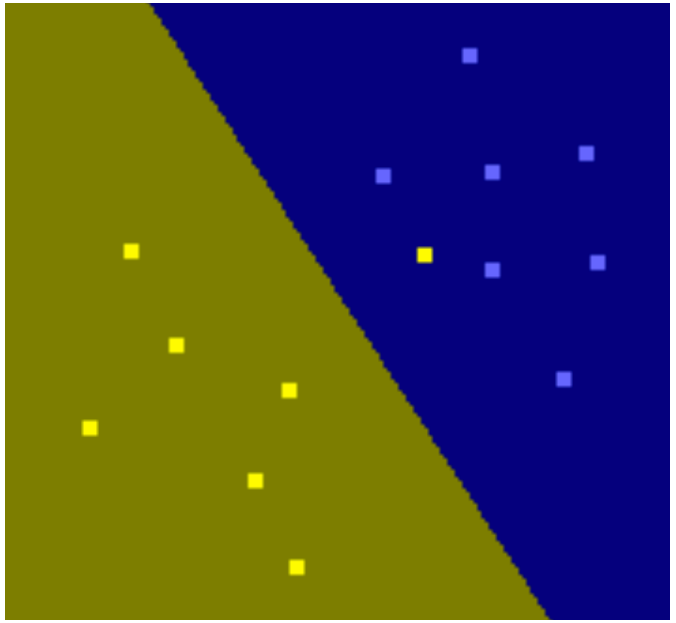


Example – SVM

- RBF Kernel (C= 1 $\gamma = 0.01$)

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

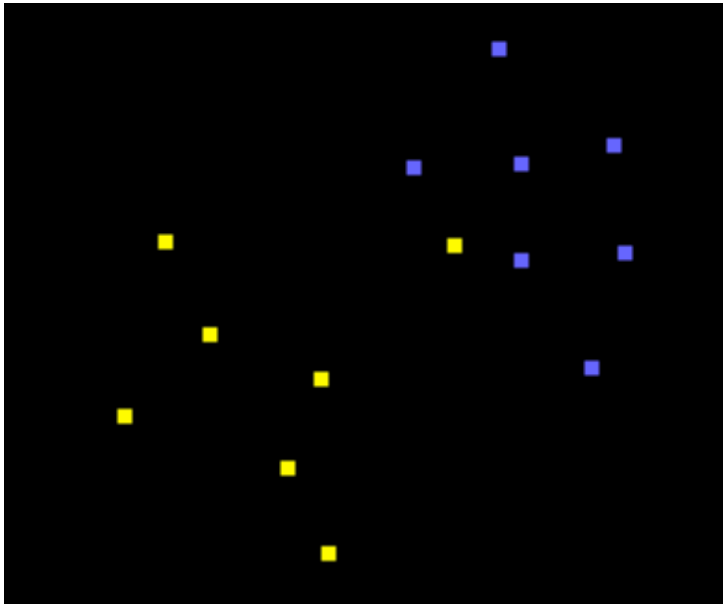


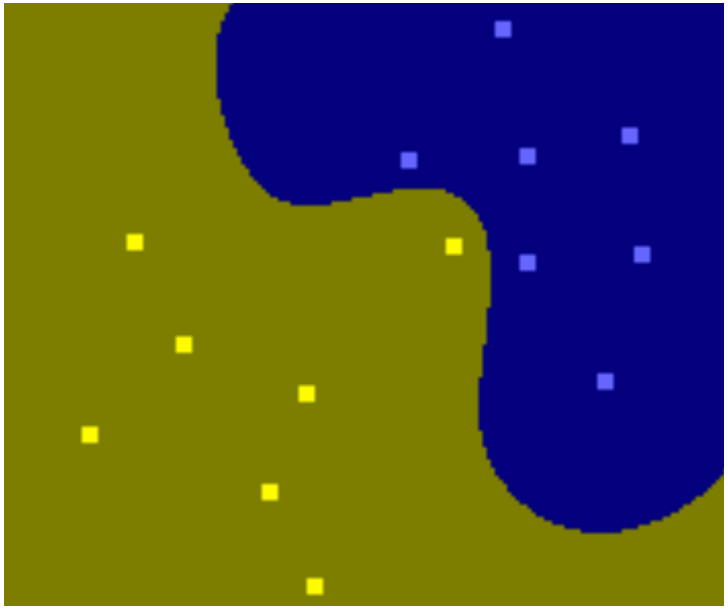


Example – SVM

- RBF Kernel (C= 10 $\gamma = 100$)

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

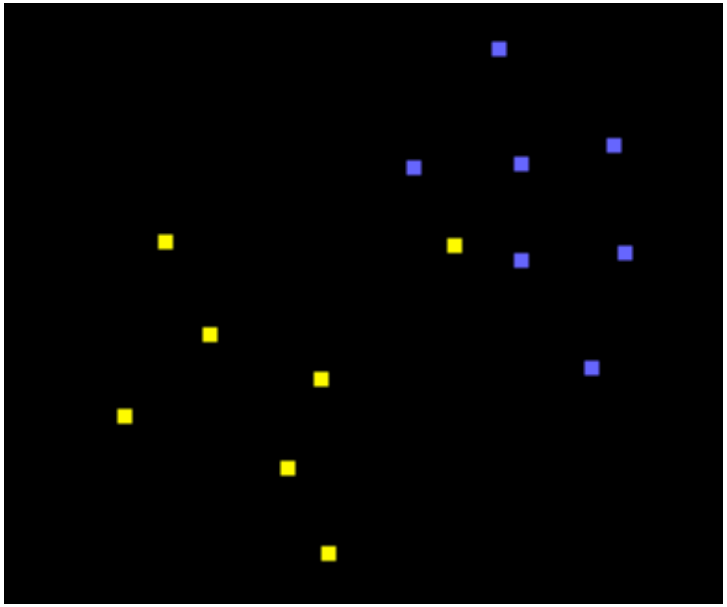


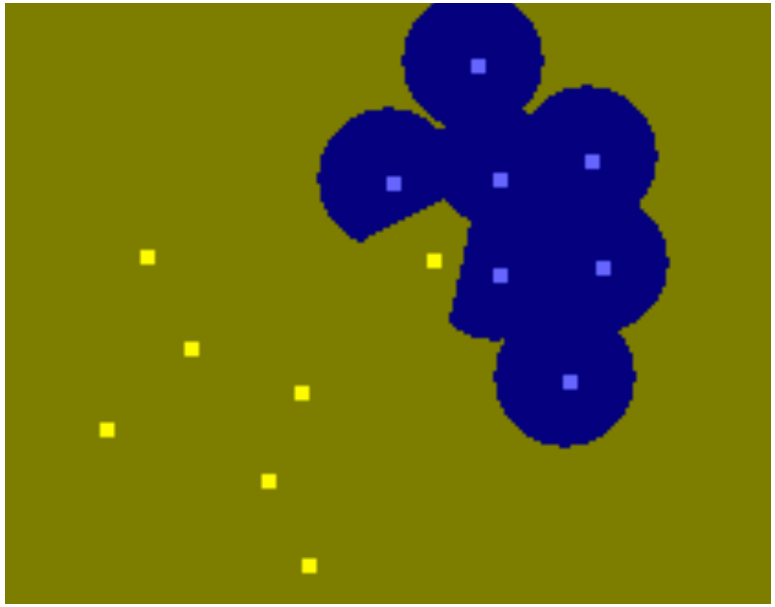


Example – SVM

- RBF Kernel (C= 10 $\gamma = 10000$)

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

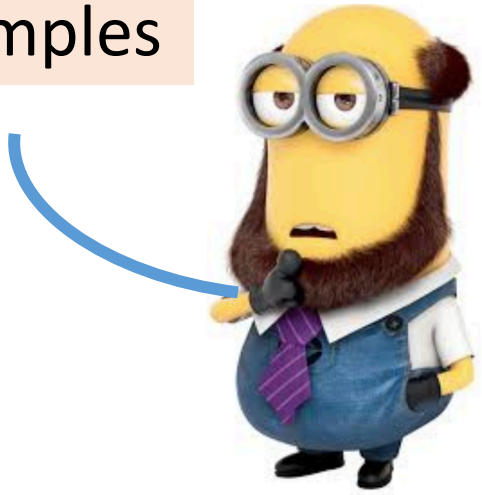




Try it by yourself:
https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

How can we learn a model in an infinite dimensional space?

Represent the model by training samples



We don't need to represent w explicitly, but only need a way to compute $w^T x$

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

Example: Perceptron Algorithm with Mapping

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

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

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

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

Kernel Perceptron Algorithm

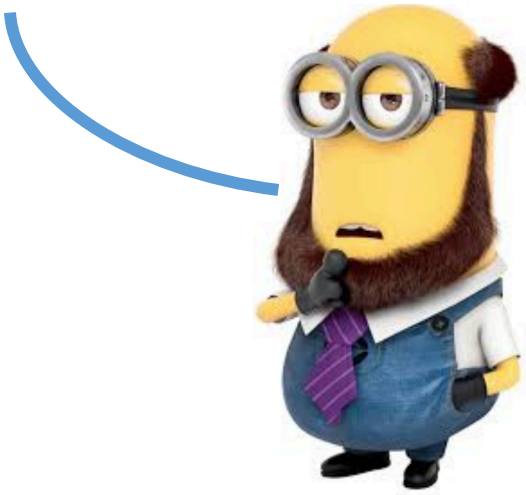
Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$, $\boldsymbol{\phi}(.)$

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

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

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

Using α_i instead of w



Kernel Perceptron Algorithm

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

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

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

Prediction: $y^{\text{test}} \leftarrow \text{sgn}((\sum_{1..m} \alpha_i \, y_i \phi(x_i))^{\top} \boldsymbol{\phi}(\boldsymbol{x})^{\text{test}})$

Kernel Perceptron Algorithm

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

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\boldsymbol{m}}$
2. For $(\boldsymbol{x_j}, y_j)$ in $\mathcal{D}$: $(\mathrm{j} = 1 \ldots m)$
3. if $y_j\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x_j)\right) \leq \mathbf{0}$
4. $\alpha_j \leftarrow \alpha_j + 1$
5. Return $\boldsymbol{\alpha}$
6.

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

Let $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x^{Test})\right)$

Kernel Perceptron Algorithm

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

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\boldsymbol{m}}$
2. For $(\boldsymbol{x_j}, y_j)$ in $\mathcal{D}$: $(\mathrm{j} = 1 \ldots m)$
3. if $y_j\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x_j)\right) \leq \mathbf{0}$
4. $\alpha_j \leftarrow \alpha_j + 1$
5. Return $\boldsymbol{\alpha}$
6.

Even if the dimentionality of $\phi(x)$ is infinite,
If we can compute $\boldsymbol{K(x_i, x_j) = \phi(x_i)^T \phi(x_j)}$
we can learn the model and make predictions

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\sum_{1..m} \alpha_i \, y_i K(x_i, x^{Test}))$

Kernel SVM

- Can we derive dual representation of SVM?

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

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

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

Dual SVM problem

- w may be infinite variables
- We can derive the Lagrangian dual problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l \\ & \mathbf{y}^T\boldsymbol{\alpha} = 0, \end{aligned}$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$

https://www.csie.ntu.edu.tw/~cjlin/talks/MLSS.pdf

Decision function in Kernel SVM

- In the optimum, the solutions of the primal and dual problem have the following relation:

$$w^* = \sum_{1..m} \alpha_i^* \, y_i \phi(x_i)$$

- The original decision function: $w^T \phi(x) + b$
- Can be rewrite as:

$$\sum_{i=1..m} \alpha_i \, y_i K(x_i, x^{Test})$$

Support vectors

$$\sum_{i=1..m} \alpha_i \, y_i K(x_i, x^{Test})$$

- If $\alpha_i = 0 \Rightarrow$ the training sample doesn't affect the prediction

- $\alpha_i > 0 \Rightarrow$ support vectors

Common choice of Kernels

$$e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}, \text{ (Radial Basis Function)}$$

$$(\mathbf{x}_i^T\mathbf{x}_j/a+b)^d \text{ (Polynomial kernel)}$$

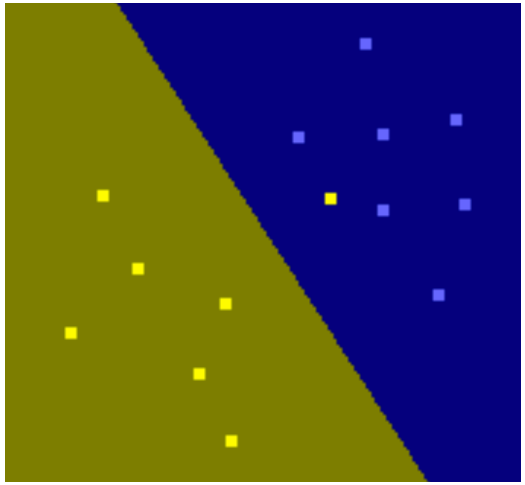
- RBF kernel implicitly map data into a feature space with infinity dimensions (by Taylor explanation)

$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T$$

RBF Kernel

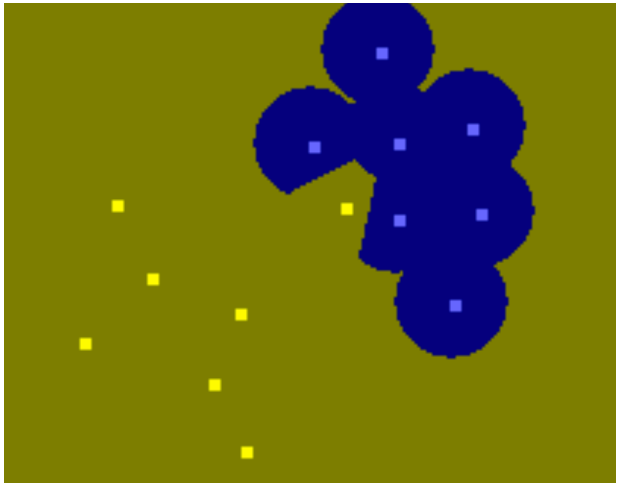
$$e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$$

Try it out:
https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html



(C= 1 $\gamma = 0.01$)

(C= 10 $\gamma = 100$)



(C= 10 $\gamma = 10000$)

What you need to know about SVMs

- What are support vectors
- SVM objectives
  (constraint and unconstraint version)
- Kernel trick, Kernel Perceptron, and Kernel SVM

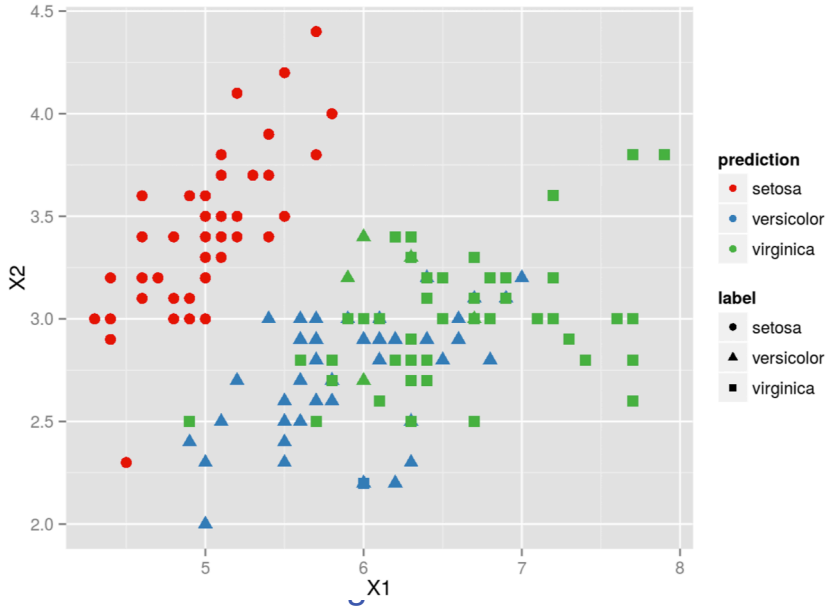
Multi-Class Classification

This Lecture

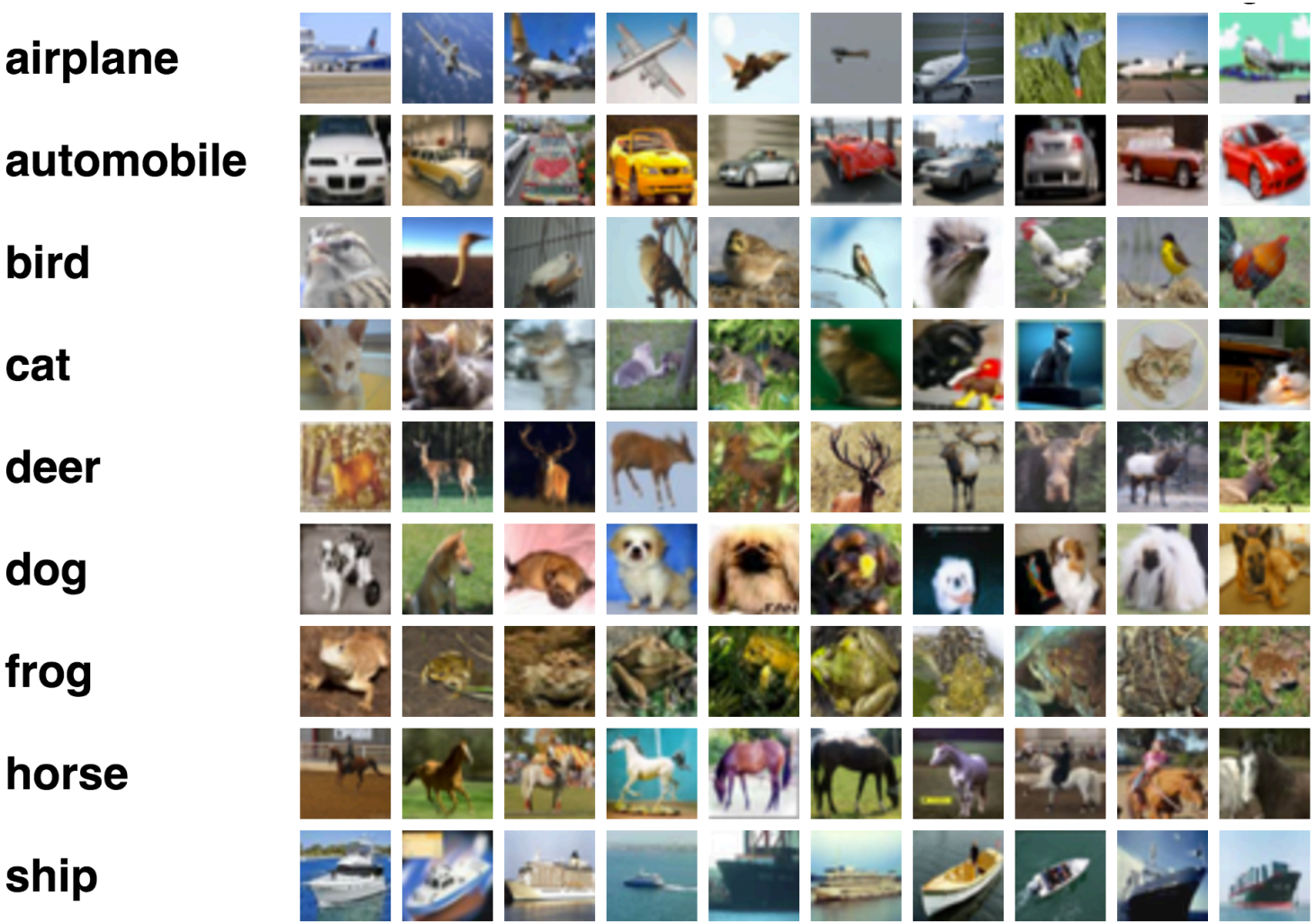
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one

What is multiclass

- Output $\in \{1,2,3, \ldots K\}$
  - In some cases, output space can be very large (i.e., K is very large)
- Each input belongs to exactly one class (c.f. in multilabel, input belongs to many classes)



Example applications



Two key ideas to solve multiclass

- Reducing multiclass to binary
  - Decompose the multiclass prediction into multiple binary decisions
  - Make final decision based on multiple binary classifiers
- Training a single classifier
  - Minimize the empirical risk
  - Consider all classes simultaneously

Reduction v.s. single classifier

- Reduction
  - Future-proof: binary classification improved so does muti-class
  - Easy to implement
- Single classifier
  - Global optimization: directly minimize the empirical loss; easier for joint prediction
  - Easy to add constraints and domain knowledge

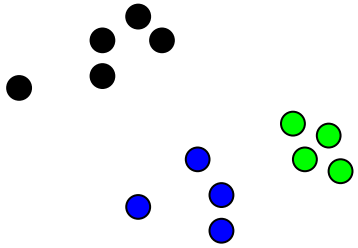
This Lecture

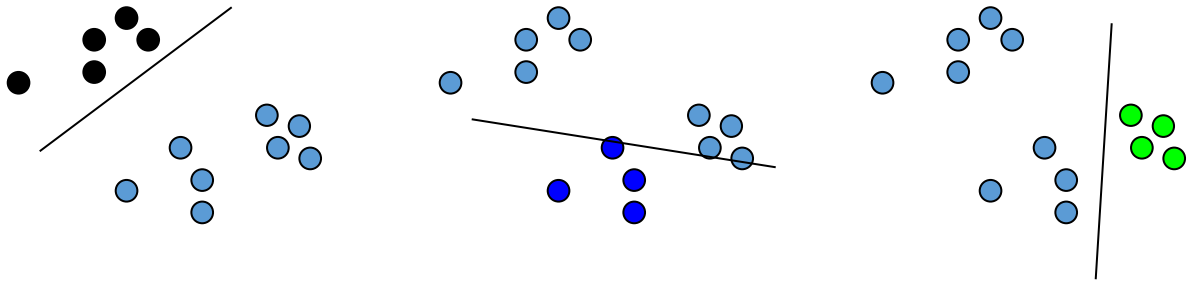
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one

One against all strategy

One against All learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems



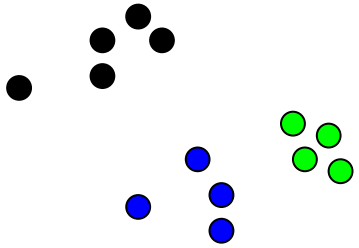


One-again-All learning algorithm

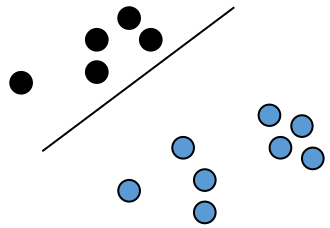
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - The binary classification can be solved by any algorithm we have seen

One against All learning

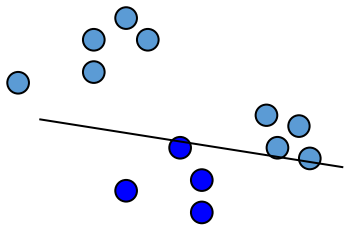
- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems



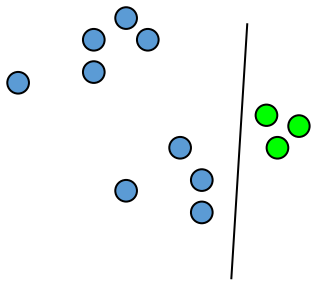
Ideal case: only the correct label will have a positive score



$w_{black}^T x > 0$



$w_{blue}^T x > 0$



$w_{green}^T x > 0$

One-again-All Inference

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \dots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \dots w_K$
- Inference: "Winner takes all"
  - $\hat{y} = \text{argmax}_{y \in \{1,2,\dots K\}} \, w_y^T \, x$

For example: $\text{y} = \text{argmax}(w_{black}^T \, x \, , \;\; w_{blue}^T \, x, \;\; w_{green}^T \;\; x)$

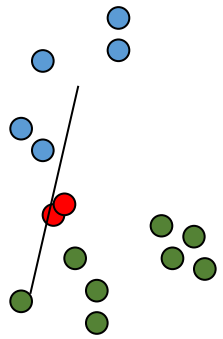
  - An instance of the general form

$$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \, f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$$

$w = \{w_1, w_2, \dots w_K\}, \; f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w}_y^T \, x$

One-again-All analysis

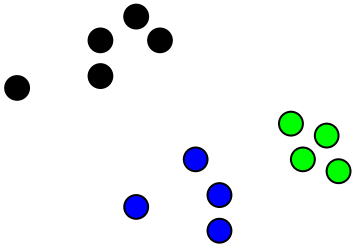
- Not always possible to learn
  - Assumption: each class individually separable from all the others
- No theoretical justification
  - Need to make sure the range of all classifiers is the same – we are comparing scores produced by K classifiers trained independently.
- Easy to implement; work well in practice

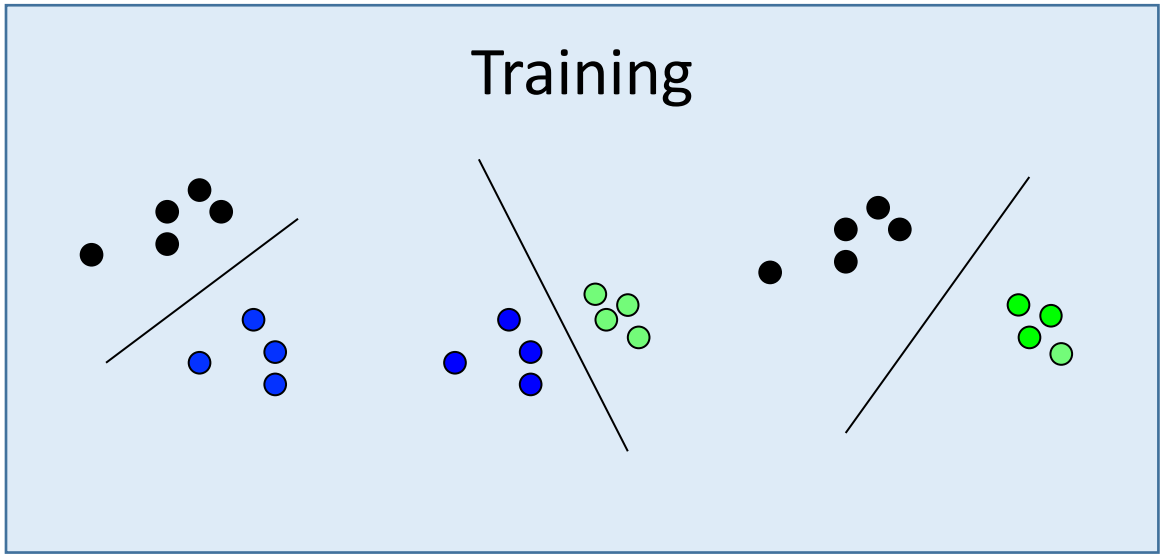


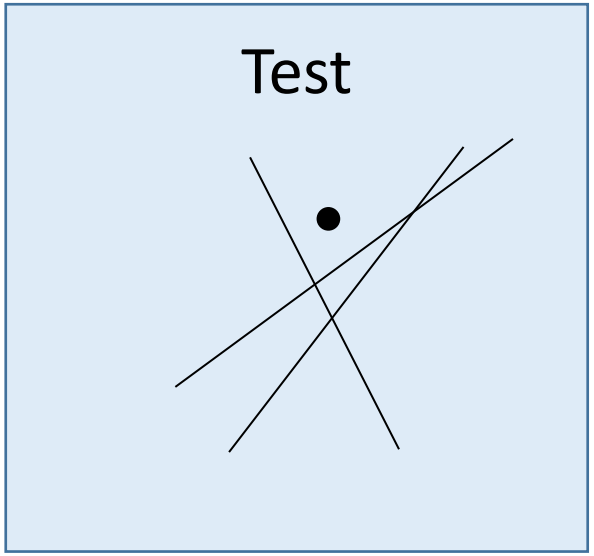
One v.s. One (All against All) strategy

One v.s. One learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems







One-v.s-One learning algorithm

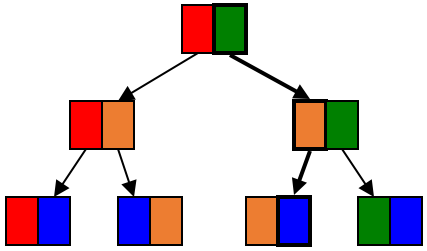
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into C(K,2) binary classification tasks
  - Learn C(K,2) models: $w_1, w_2, w_3, \ldots w_{K*(K-1)/2}$
  - For each class pair (i,j), construct a binary classification task as:
    - Positive examples: Elements of D with label i
    - Negative examples Elements of D with label j
    - The binary classification can be solved by any algorithm we have seen

One-v.s-One Inference algorithm

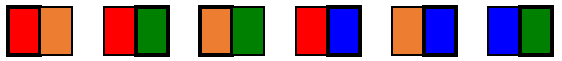
- Decision Options:
  - More complex; each label gets k-1 votes
  - Output of binary classifier may not cohere.
  - Majority: classify example x to take label i if i wins on x more often than j (j=1,…k)
  - A tournament: start with n/2 pairs; continue with winners

Classifying with One-vs-one

Tournament



Majority Vote



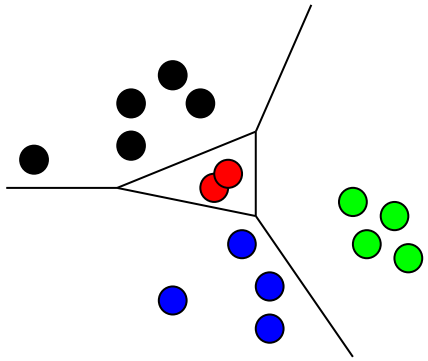
1 red, 2 yellow, 2 green

➔ ?

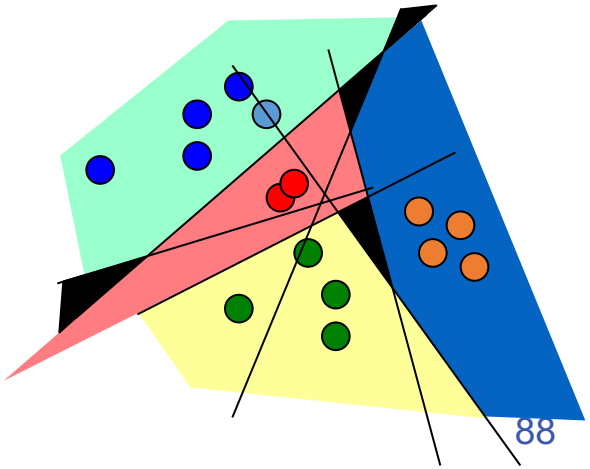
All are post-learning and *might* cause weird stuff

One-v.s.-one Assumption

- Every pair of classes is separable



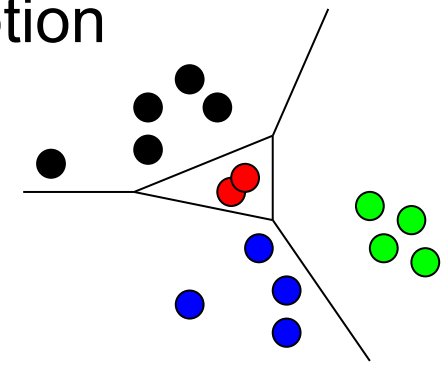
It is possible to separate all k classes with the $O(k^2)$ classifiers



Decision Regions

Comparisons

- One against all
  - O(K) weight vectors to train and store
  - Training set of the binary classifiers may unbalanced
  - Less expressive; make a strong assumption



- One v.s. One (All v.s. All)
  - O(K^2) weight vectors to train and store
  - Size of training set for a pair of labels could be small
    ⇒ overfitting of the binary classifiers
  - Need large space to store model

Problems with Decompositions

- Learning optimizes over *local* metrics
  - Does not guarantee good *global* performance
  - We don't care about the performance of the *local* classifiers
- Poor decomposition $\Rightarrow$ poor performance
  - Difficult local problems
  - Irrelevant local problems
- Efficiency: e.g., All vs. All vs. One vs. All
- Not clear how to generalize multi-class to problems with a very large # of output

Still an ongoing research direction

Key questions:

- How to deal with large number of classes
- How to select "right samples" to train binary classifiers

- Error-correcting tournaments
  [Beygelzimer, Langford, Ravikumar 09]
- Logarithmic Time One-Against-Some
  [Daume, Karampatziakis, Langford, Mineiro 16]
- Label embedding trees for large multi-class tasks.
  [Bengio, Weston, Grangier 10]
- …

Decomposition methods: Summary

- General Ideas:
  - Decompose the multiclass problem into many binary problems
  - Prediction depends on the decomposition
    - Constructs the multiclass label from the output of the binary classifiers
- Learning optimizes local correctness
  - Each binary classifier don't need to be globally correct and isn't aware of the prediction procedure