

# Support Vector Machines

---

Machine Learning 1 — Lecture 11

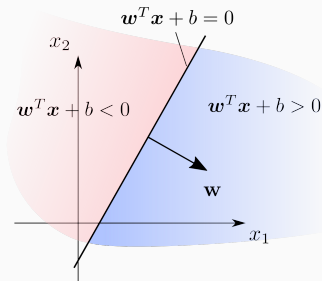
6<sup>th</sup> June 2023

Robert Peharz

Institute of Theoretical Computer Science

Graz University of Technology

- Basically, **support vector machines** (SVMs) are linear models
- Beautiful theory
- Convex problem, i.e. a global minimum can be found
- They are often used to introduce the **kernel trick**, a powerful technique to work implicitly in high-dimensional spaces
- SVMs were the state-of-the-art from about 1995-2010
- Neural networks didn't take off back then
- We will discuss SVMs in the context of binary classification – multi-class extensions are available and relatively straight-forward



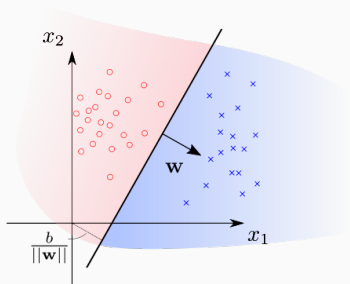
A **vector**  $w$  and a **bias**  $b$  describe a **hyperplane** via

$$w^T x + b = 0$$

Vector  $w$  is the **normal vector** of the hyperplane.

The positive and negative **half-spaces** are described as

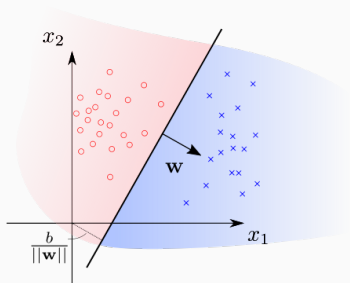
$$w^T x + b > 0 \quad \text{and} \quad w^T x + b < 0 \quad \text{respectively.}$$



**Linearly separable datasets:** means that there exists at least one **separating hyperplane**, which classifies all samples correctly (positive samples in positive half-space, negative samples in negative half-space).

We will first assume that the training set is linearly separable and relax this assumption later.

# Support Vector Machines



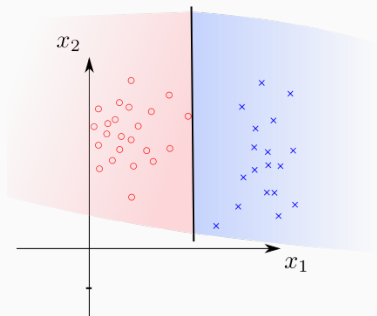
**Support vector machines** classify with a **separating hyperplane**:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

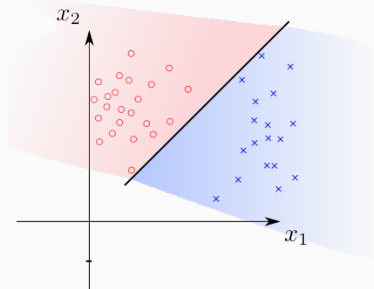
Separating hyperplane is usually not unique.

What is a “good” separating hyperplane?

Is this a good hyperplane?

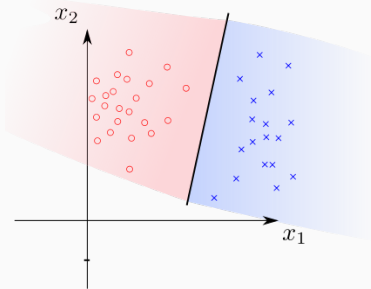


Or this?

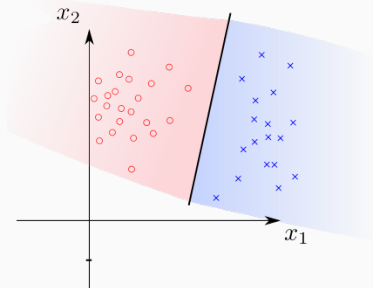


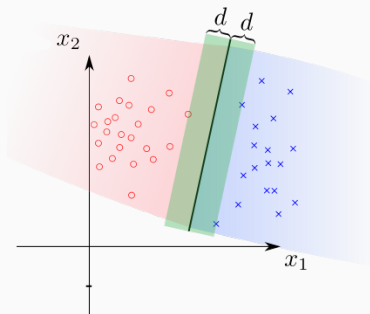


Or this?



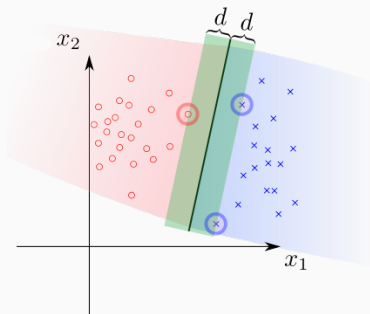
The last one looks somehow good.  
But why?





**Classification Margin (Margin):**  
minimal distance  $d$  to any training example.

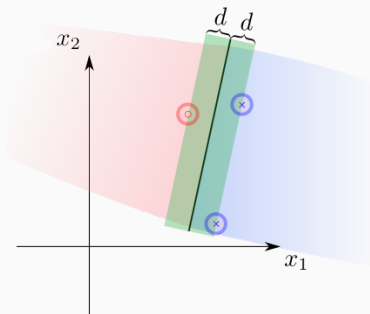
**Max-Margin Hyperplane:**  
maximizes the margin; uniquely defined; physical interpretation: thickest board fitting between training vectors.



**Classification Margin (Margin):**  
minimal distance  $d$  to any training example.

**Max-Margin Hyperplane:**  
maximizes the margin; uniquely defined; physical interpretation: thickest board fitting between training vectors.

The “board” is **supported** by a few **support vectors** on the margin, hence the name **Support Vector Machine (SVM)**.



SVM is entirely determined by support vectors – same solution if other points are removed.

**Classification Margin (Margin):**  
minimal distance  $d$  to any training example.

**Max-Margin Hyperplane:**  
maximizes the margin; uniquely defined; physical interpretation: thickest board fitting between training vectors.

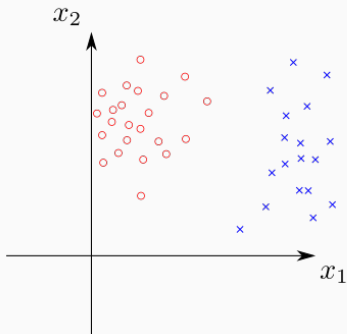
The “board” is **supported** by a few **support vectors** on the margin, hence the name **Support Vector Machine (SVM)**.

# Learning Support Vector Machines

---

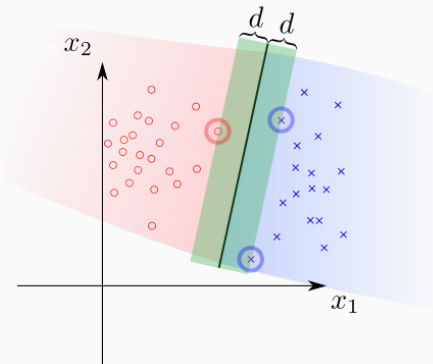
# Setting

Assume linearly separable data for now.



# Setting

Goal: find the **max-margin hyperplane**.



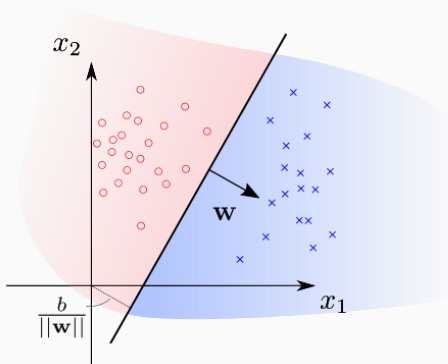


# Setting

Datapoints  $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots (\mathbf{x}^{(N)}, y^{(N)})$

$$y^{(i)} = \begin{cases} +1 & \text{for positive class } (\times) \\ -1 & \text{for negative class } (\circ) \end{cases}$$

$$\hat{y} = f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

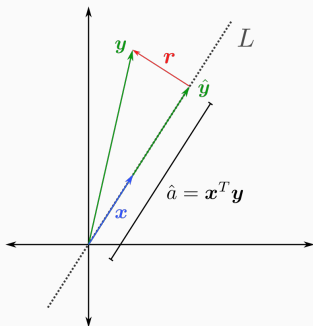


# Optimization Problem 1

We can describe SVM learning as follows:

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{maximize}} && \overbrace{\min_i d(\mathbf{x}^{(i)}, \mathbf{w}, b)}^{\text{margin}} \\ & \text{s.t.} && \text{all samples correctly classified} \end{aligned}$$

- $d(\mathbf{x}, \mathbf{w}, b) :=$  Euclidean distance of  $\mathbf{x}$  to hyperplane  $\mathbf{w}, b$
- Correct formulation, but too abstract for optimizers. . .



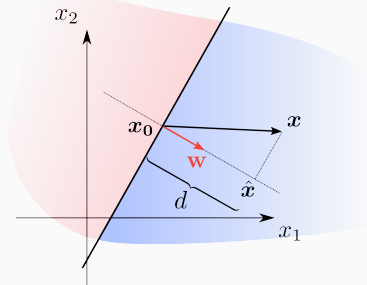
If  $x$  is a normalized vector, then the projection of  $y$  onto  $x$  is

$$\hat{y} = x^T y x$$

If  $x$  is not normalized, we need to divide  $x$  by its norm, yielding:

$$\hat{y} = \frac{x^T y x}{\|x\| \|x\|} = \frac{x^T y x}{\|x\|^2}$$

# Distance of Point to Hyperplane



- Let  $w$  and  $b$  be given, describing hyperplane
- Let  $x$  be an arbitrary point and  $x_0$  an arbitrary point on the hyperplane
- $x - x_0$  is the vector starting at  $x_0$  and pointing to  $x$
- Let  $\hat{x}$  be the projection of  $x - x_0$  onto  $w$
- $\|\hat{x}\|$  is the Euclidean distance between  $x$  and the hyperplane

## Distance of Point to Hyperplane cont'd

Euclidean distance of  $\mathbf{x}$  to hyperplane:

$$d = \left\| \frac{\overbrace{\mathbf{w}^T (\mathbf{x} - \mathbf{x}_0) \mathbf{w}}^{\hat{\mathbf{x}}}}{\|\mathbf{w}\|^2} \right\| = |\mathbf{w}^T (\mathbf{x} - \mathbf{x}_0)| \frac{\|\mathbf{w}\|}{\|\mathbf{w}\|^2} = \frac{|\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mathbf{x}_0|}{\|\mathbf{w}\|}$$

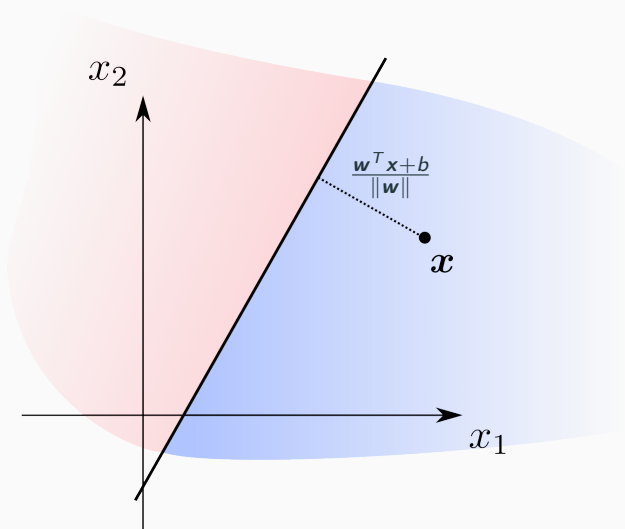
Since  $\mathbf{x}_0$  lies on the hyperplane,  $\mathbf{w}^T \mathbf{x}_0 + b = 0$ . Thus  $\mathbf{w}^T \mathbf{x}_0 = -b$  and

$$d = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|}$$

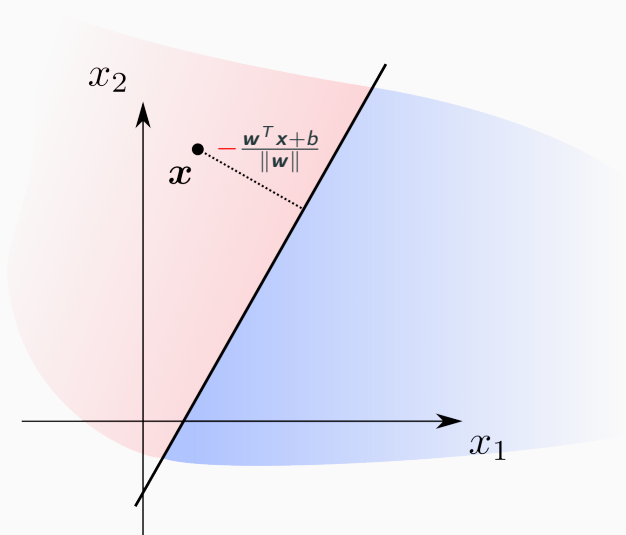
If we remove the absolute value we get a **signed distance**

$$d_{\text{signed}} = \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

## (Signed) Distance between Point and Hyperplane Example



## (Signed) Distance between Point and Hyperplane Example



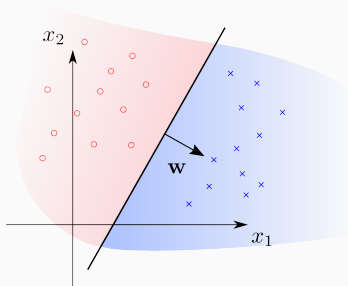
## Optimization Problem 2 (formalizing Problem 1)

$$\underset{\mathbf{w}, b}{\text{maximize}} \quad \overbrace{\min_i \frac{y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}}^{\text{margin}}$$

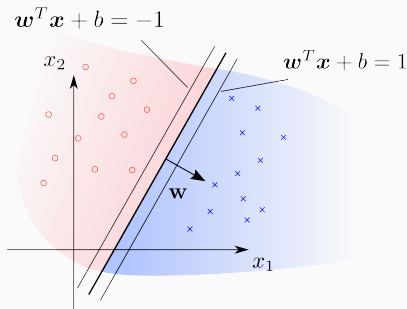
- For any separating hyperplane,  $\frac{y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}$  is the correct distance, due to multiplication with  $y_i \in \{-1, +1\}$
- Hence, the objective  $\min_i \frac{y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}$  is positive exactly for separating hyperplanes
- Since we **maximize** the objective we will find
  1. a separating hyperplane,
  2. which maximizes the margin
- However, the  $\min_i$  and division by  $\|\mathbf{w}\|$  make the problem **non-convex**, i.e. it is hard to find a global optimum



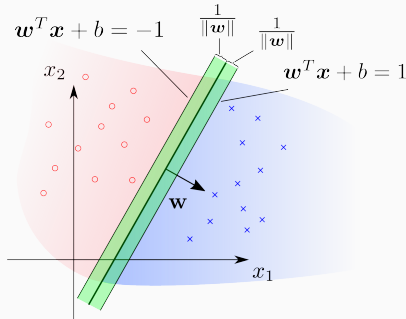
- Let  $\mathbf{w}^T \mathbf{x} + b = 0$  be a separating hyperplane



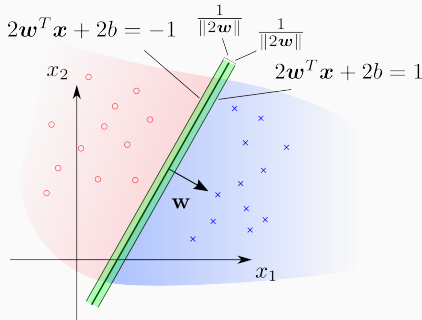
- Let  $\mathbf{w}^T \mathbf{x} + b = 0$  be a separating hyperplane
- Introduce **margin hyperplanes**  $\mathbf{w}^T \mathbf{x} + b = -1$  and  $\mathbf{w}^T \mathbf{x} + b = 1$ , i.e. shifted versions of the separating hyperplane



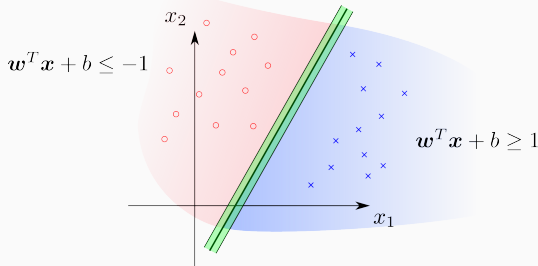
- Let  $\mathbf{w}^T \mathbf{x} + b = 0$  be a separating hyperplane
- Introduce **margin hyperplanes**  $\mathbf{w}^T \mathbf{x} + b = -1$  and  $\mathbf{w}^T \mathbf{x} + b = 1$ , i.e. shifted versions of the separating hyperplane
- The distance of the margin hyperplanes to the separating hyperplane, is given as  $\frac{1}{\|\mathbf{w}\|}$  (to see this, plug in  $\mathbf{w}^T \mathbf{x} + b = \pm 1$  into the distance formula)



- Scaling  $\mathbf{w}$  and  $b$  does not change the separating hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$ , but moves the margin hyperplanes closer to the separating hyperplane



- Scaling  $\mathbf{w}$  and  $b$  does not change the separating hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$ , but moves the margin hyperplanes closer to the separating hyperplane
- Thus, by appropriately scaling  $\mathbf{w}$  and  $b$  we can always achieve that all points satisfy the **margin constraints**, i.e.
  - all negative points satisfy  $\mathbf{w}^T \mathbf{x} + b \leq -1$
  - all positive points satisfy  $\mathbf{w}^T \mathbf{x} + b \geq 1$



## Margin Constraints:

- All negative points must satisfy  $\mathbf{w}^T \mathbf{x} + b \leq -1$
- All positive points must satisfy  $\mathbf{w}^T \mathbf{x} + b \geq 1$
- Thus, all points must satisfy:

$$y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1$$

Thus, SVM learning can now be described as

- maximizing distance of margin hyperplanes  $\frac{1}{\|\mathbf{w}\|}$
- while satisfying all margin constraints

## Optimization Problem 3 (equivalent Problem 2)

$$\begin{array}{ll}\text{maximize}_{\mathbf{w}, b} & \frac{1}{\|\mathbf{w}\|} \\ \text{s.t.} & y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N\end{array}$$

## Optimization Problem 3 (equivalent Problem 2)

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{maximize}} && \frac{1}{\|\mathbf{w}\|} \\ & \text{s.t.} && y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N \end{aligned}$$

- Still not convex, due to  $\frac{1}{\|\mathbf{w}\|}$
- However, this is easily fixed:
  - $\frac{1}{\|\mathbf{w}\|} \rightarrow \|\mathbf{w}\|$
  - $\max \rightarrow \min$



## Optimization Problem 3 (equivalent Problem 2)

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{maximize}} && \frac{1}{\|\mathbf{w}\|} \\ & \text{s.t.} && y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N \end{aligned}$$

- Still not convex, due to  $\frac{1}{\|\mathbf{w}\|}$
- However, this is easily fixed:
  - $\frac{1}{\|\mathbf{w}\|} \rightarrow \|\mathbf{w}\|$
  - $\max \rightarrow \min$
- Further, minimizing  $\|\mathbf{w}\|$  is equivalent to minimizing  $\frac{1}{2}\|\mathbf{w}\|^2$ , which is smooth and differentiable everywhere.

## Optimization Problem 4 (final, equivalent to Problem 3)

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{s.t.} && y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N \end{aligned}$$

- Standard formulation of SVMs
- **Quadratic program**
- Quadratic objective  $\frac{1}{2} \|\mathbf{w}\|^2$
- Linear constraints  $y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1$
- Convex optimization problem
- Amenable for standard solvers (e.g. Gurobi)
- Many specialized SVM solvers exist (e.g. scikit-learn)

# SVMs with Soft-Margin

---

## What if data is not linearly separable?

$$\begin{array}{ll} \underset{\mathbf{w}, b}{\text{minimize}} & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} & y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N \end{array}$$

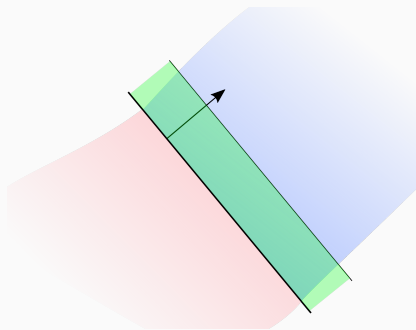
## What if data is not linearly separable?

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{s.t.} && y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1 \dots N \end{aligned}$$

If data is not linearly separable, there doesn't exist a **feasible solution**, i.e. no  $\mathbf{w}$  and  $b$  can satisfy all margin constraints simultaneously.

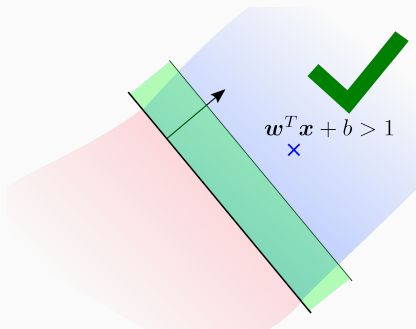
How can we deal with that?

## Margin Constraints $y^{(i)} (w^T x^{(i)} + b) \geq 1$



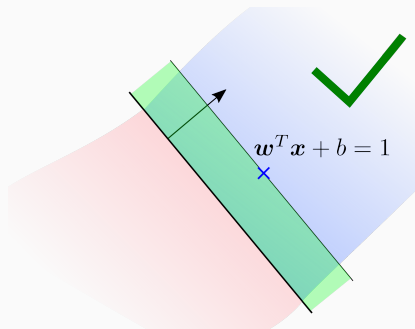
# Margin Constraints $y^{(i)} (w^T x^{(i)} + b) \geq 1$

- Safe point  $w^T x^{(i)} + b > 1$



## Margin Constraints $y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1$

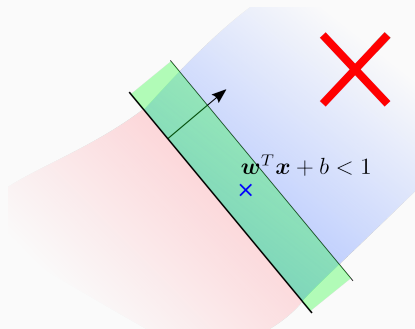
- Safe point  $\mathbf{w}^T \mathbf{x}^{(i)} + b > 1$
- Support vector  $\mathbf{w}^T \mathbf{x}^{(i)} + b = 1$





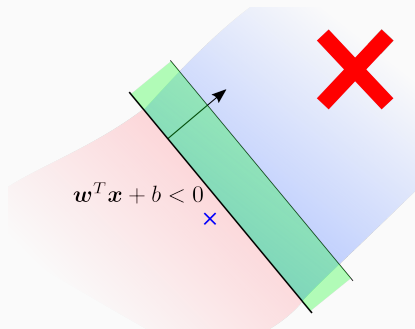
## Margin Constraints $y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1$

- Safe point  $\mathbf{w}^T \mathbf{x}^{(i)} + b > 1$
- Support vector  $\mathbf{w}^T \mathbf{x}^{(i)} + b = 1$
- Violates constraint  $\mathbf{w}^T \mathbf{x}^{(i)} + b < 1$



## Margin Constraints $y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1$

- Safe point  $\mathbf{w}^T \mathbf{x}^{(i)} + b > 1$
- Support vector  $\mathbf{w}^T \mathbf{x}^{(i)} + b = 1$
- Violates constraint  $\mathbf{w}^T \mathbf{x}^{(i)} + b < 1$
- Wrongly classified  $\mathbf{w}^T \mathbf{x}^{(i)} + b < 0$



- In non-separable data, we cannot achieve **all** margin constraints simultaneously
- But we are the “happier” the less each constraint is violated
- Let’s introduce **slack variables**, relaxing the constraints:

$$y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 - \xi^{(i)} \quad \text{where } \xi^{(i)} \geq 0$$

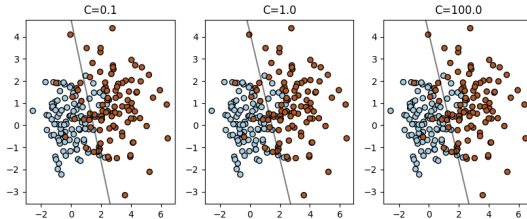
- We want  $\xi^{(i)}$  to be small  $\Rightarrow$  penalize them

# SVM with Soft-Margin

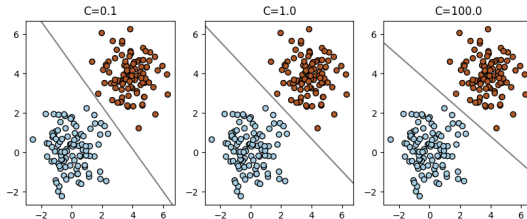
$$\begin{aligned} \underset{\mathbf{w}, b, \xi}{\text{minimize}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi^{(i)} \\ \text{s.t.} \quad & y_i \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 - \xi^{(i)} \\ & \xi^{(i)} \geq 0 \quad \text{for } i = 1 \dots N \end{aligned}$$

- Hyperparameter  $C$  trades off  $\frac{1}{2} \|\mathbf{w}\|^2$  and slacks  $\sum_i \xi^{(i)}$
- $C \rightarrow \infty$ : hard margin
- Select e.g. with cross-validation
- Still a Quadratic Program (convex)
- Amenable for standard solvers, and specialized solvers exist

### Linearly Non-separable Data



### Linearly Separable Data



## Dual SVM and the Kernel Trick

---

The SVM optimization problem (**primal problem**):

$$\begin{aligned} \underset{\mathbf{w}, b, \xi}{\text{minimize}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi^{(i)} \\ \text{s.t.} \quad & y_i \left( \mathbf{w}^T \mathbf{x}^{(i)} + b \right) \geq 1 - \xi^{(i)} \\ & \xi^{(i)} \geq 0 \quad \text{for } i = 1 \dots N \end{aligned}$$

It can be shown that the following is equivalent (**dual problem**):

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} + \sum_{i=1}^N \lambda^{(i)} \\ \text{s.t.} \quad & \sum_{i=1}^N \lambda^{(i)} y^{(i)} = 0 \\ & 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

## Dual SVM Problem

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} + \sum_i \lambda^{(i)} \\ \text{s.t.} \quad & \sum_i \lambda^{(i)} y^{(i)} = 0 \\ & 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

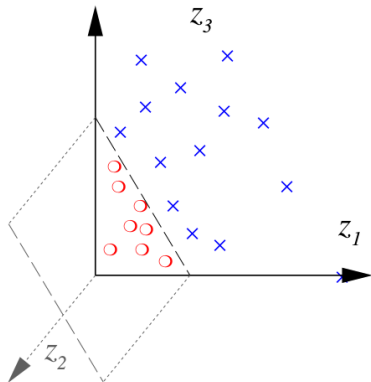
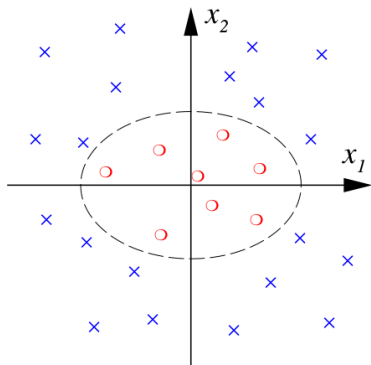
- Optimization variables  $\lambda^{(i)}$ , corresponding to **Lagrange-multiplier** for margin constraint of  $i^{\text{th}}$  sample
- Still a quadratic program (convex)
- Optimal  $\lambda^*$  delivers optimal solution for primal:
  - $\mathbf{w}^* = \sum_i \lambda^{(i)*} y^{(i)} \mathbf{x}^{(i)}$
  - $b^* = \mathbf{w}^{*T} \mathbf{x}^{(i)} - y^{(i)}$  (for any  $i$  where  $\lambda^{*(i)} > 0$ )
- $\lambda^{(i)*} > 0$  indicates a support vector



## Dual vs. Primal SVM

- Solving primal SVM:  $\Omega(DN)$
- Solving dual SVM:  $\Omega(N^2)$
- Dual SVM depends only on inner products  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)}$ , i.e., on pair-wise similarities between samples, not on samples themselves
- This allows us to apply the so-called **kernel trick**
- In a nutshell, the kernel trick computes non-linear features implicitly

- Apply non-linear feature transforms first, transforming learning problem into higher dimensional feature space
- High dimensional data tends to be linearly separable
- For example, use
  - radial basis functions
  - polynomials
  - exp, log
  - trigonometric functions, cos, sin, tanh, ...
- Non-linear features + linear classifier = non-linear classifier



Left: unsuitable for linear classifiers.

[Schölkopf, MLSS 2013]

Right: non-linear features yield linearly separable data.

$$\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$$

$$\Phi(\mathbf{x}) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) = (z_1, z_2, z_3)$$

# The Problem with Feature Transforms

- We want many feature transforms, since we don't know which are the “good” ones
- This quickly explodes. For example:
  - $D$  original features
  - We take all polynomials of degree  $K$
  - $\binom{D+K-1}{K}$  features
  - E.g.,  $D = 100$ ,  $K = 5$ , yields  $75 \times 10^6$  features
- Exhaustive feature transforms are inefficient

Can we work **implicitly** in high-dimensional feature spaces, without computing features explicitly?

Can we work **implicitly** in high-dimensional feature spaces, without computing features explicitly?

**Yes, using the kernel trick!**

# The Kernel Trick in a Nutshell

- Prerequisite: learning algorithms can be re-formulated such that it works only with
  - labels, and
  - pairwise inner products  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)}$  of input vectors
- Each  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)}$  is a scalar, regardless of input dimensionality, and is a **similarity measure** between  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$
- We can replace now  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)}$  with  $\phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$ , i.e. the inner products in feature space
- The algorithm still works the same, we only replaced the similarity measure!
- Can we “shortcut” the computation of  $\phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$ ?

## Kernel

A **kernel**  $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$  is a function which maps two vectors  $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^D$  to a real number.

## Positive Definite Kernel

A kernel is **positive definite**, if for any finite collection of vectors  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}$  and any collection of real numbers  $a^{(1)}, \dots, a^{(N)}$  we have

$$\sum_{i,j} a^{(i)} a^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq 0$$



# Representer Theorem

A kernel  $k$  is **positive definite** if and only if

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$$

for **some** feature transformation  $\phi(\mathbf{x})$ .

# The Kernel Trick

- **Kernel Trick:** Replace direct computation  $\phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$  with cheap computation  $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- Many correspondences between kernels and feature transforms are known:

- **Linear kernel** corresponds to  $\phi(\mathbf{x}) = \mathbf{x}$

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbf{x}^{(i)T} \mathbf{x}^{(j)}$$

- **Polynomial kernel** corresponds to  $\phi$  computing all polynomials up to degree  $p$

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = (1 + \mathbf{x}^{(i)T} \mathbf{x}^{(j)})^p$$

- **Gaussian kernel** correspond to infinite feature space  $\phi$  (functional space)

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2)$$

- ...

# Advantages of the Kernel Trick

- Large savings in compute, or enabling certain abstract feature space at all
- As long as  $k$  is positive definite, **we don't even need to know  $\phi$  explicitly!**
- **Generalizes many learning algorithms:**
  - SVMs
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - ...
- **Embed “non-vector” data in vector space:**
  - Strings
  - Graphs
  - ...

$$\begin{aligned} & \underset{\lambda}{\text{maximize}} && -\frac{1}{2} \sum_{i,j} \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} + \sum_i \lambda^{(i)} \\ & \text{s.t.} && \sum_i \lambda^{(i)} y^{(i)} = 0 \\ & && 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

## Kernelized SVMs

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i,j} \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)}) + \sum_i \lambda^{(i)} \\ \text{s.t.} \quad & \sum_i \lambda^{(i)} y^{(i)} = 0 \\ & 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

## Kernelized SVMs

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i,j} \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sum_i \lambda^{(i)} \\ \text{s.t.} \quad & \sum_i \lambda^{(i)} y^{(i)} = 0 \\ & 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

## Kernelized SVMs

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i,j} \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sum_i \lambda^{(i)} \\ \text{s.t.} \quad & \sum_i \lambda^{(i)} y^{(i)} = 0 \\ & 0 \leq \lambda^{(i)} \leq C \quad \text{for } i = 1 \dots N \end{aligned}$$

- Quadratic program (convex)
- Delivers globally optimal solution  $\lambda^*$

# Classifying with Kernelized SVMs

Recall that

- $\mathbf{w}^* = \sum_i \lambda^{(i)*} y^{(i)} \phi(\mathbf{x}^{(i)})$
- $b^* = \mathbf{w}^{*T} \phi(\mathbf{x}^{(i)}) - y^{(i)}$  (for any  $i$  where  $\lambda^{(i)} > 0$ )

Classification for new  $\mathbf{x}$  via  $\text{sgn}(\mathbf{w}^T \phi(\mathbf{x}) + b)$ . We can express this again via the kernel as:

$$f(\mathbf{x}) = \text{sgn} \left( \sum_i \lambda^{(i)*} y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x}) - b^* \right)$$

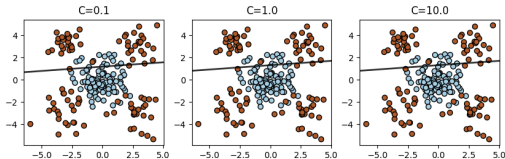
where

$$b^* = \sum_j \lambda^{(j)*} y^{(j)} k(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}) - y^{(i)} \quad (\text{for any } i \text{ where } \lambda^{(i)} > 0)$$

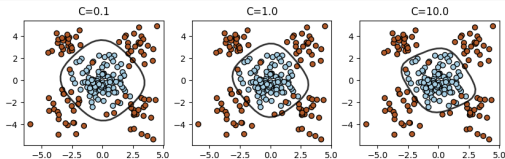


# Effect of Different Kernels (sklearn)

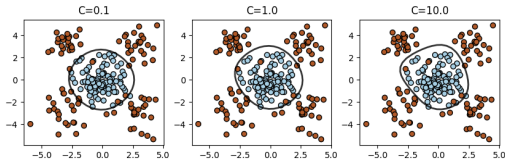
```
classifier = svm.SVC(kernel='linear', C=C)
```



```
classifier = svm.SVC(kernel='poly', degree=4, C=C)
```



```
classifier = svm.SVC(kernel='rbf', C=C)
```



- Primal SVM and Dual SVM
  - Size Primal  $\Omega(DN)$
  - Size Dual  $\Omega(N^2)$
  - Both are quadratic programs (convex, global optimum)
- Dual SVM depends only on inner products  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)} \Rightarrow$  amenable to the kernel trick
- The Kernel Trick
  - replace  $\mathbf{x}^{(i)T} \mathbf{x}^{(j)}$  with  $\phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$  which can be cheaply computed via  $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
  - Representer Theorem: Kernel corresponds to inner product in some feature space, if and only if  $k$  is positive definite
  - Kernel SVMs, Kernel PCA, Kernel LDA, ...
- Kernelized SVM is one of the strongest classifiers
- High time of SVMs and Kernels: 90's and early 2000's