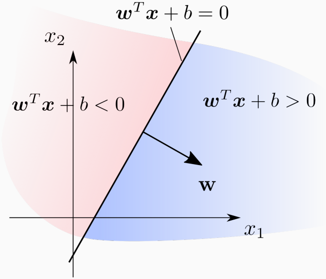
Support Vector Machines

Machine Learning 1 — Lecture 11

4^{th} June 2024

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



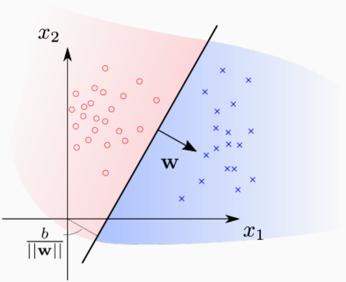
A **vector** $\boldsymbol{w}$ and a **bias** b describe a **hyperplane** via

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0$$

Vector $\boldsymbol{w}$ is the **normal vector** of the hyperplane.

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

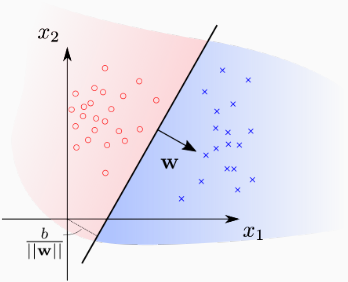
$$\boldsymbol{w}^T \boldsymbol{x} + b > 0 \quad \text{and} \quad \boldsymbol{w}^T \boldsymbol{x} + b < 0 \quad \text{respectively.}$$



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

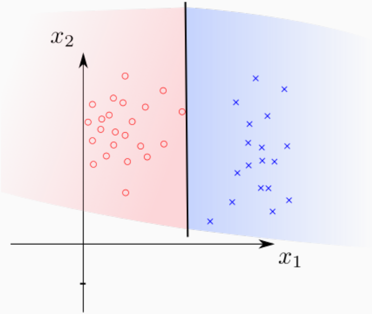


Support vector machines classify with a **separating hyperplane**:

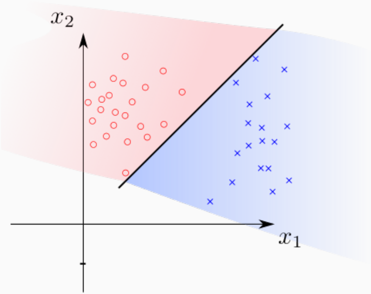
$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^T\boldsymbol{x} + b) = \begin{cases} +1 & \text{if } \boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Separating hyperplane is usually not unique.

What is a “good” separating hyperplane?

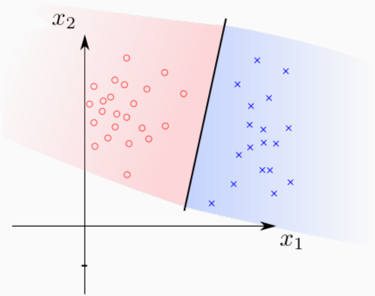


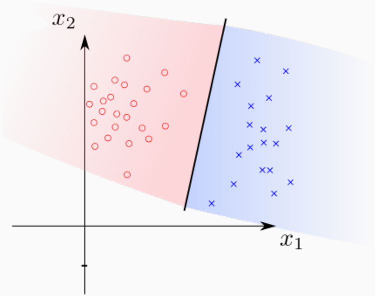
Is this a good hyperplane?



Or this?

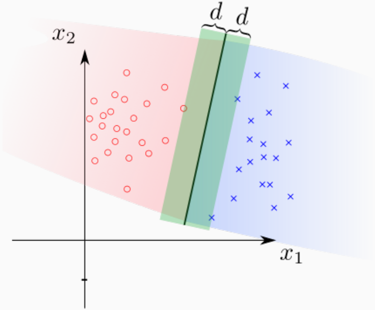
Or this?





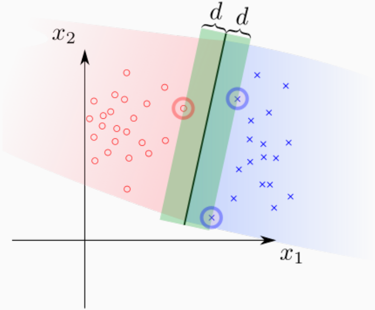
The last one looks somehow good.

But why?



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

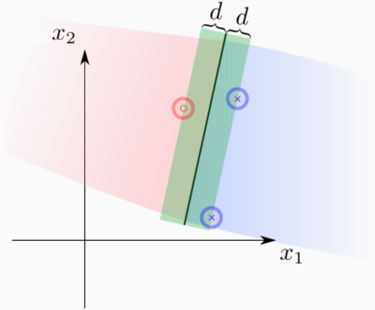
Max-Margin Hyperplane: maximizes the margin; uniquely defined; physical interpretation: thickest board fitting between training vectors.



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.

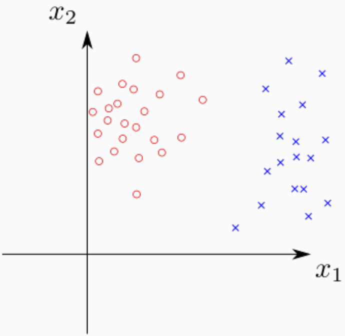
Max-Margin Hyperplane: maximizes the margin; uniquely defined; physical interpretation: thickest board fitting between training vectors.

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

Learning Support Vector Machines

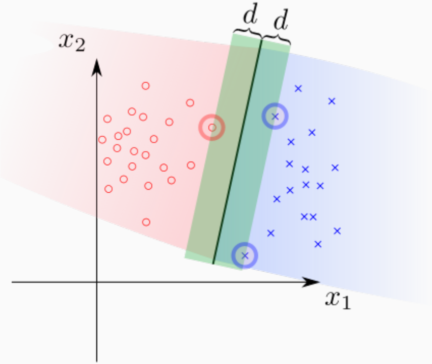
Setting

Assume linearly separable data for now.



Setting

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

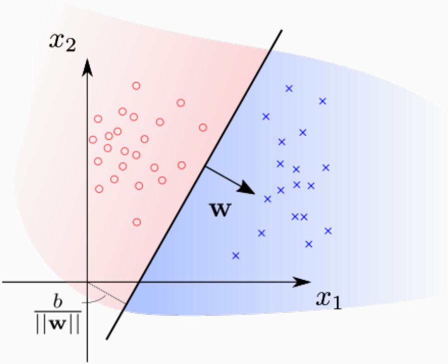


Setting

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

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

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

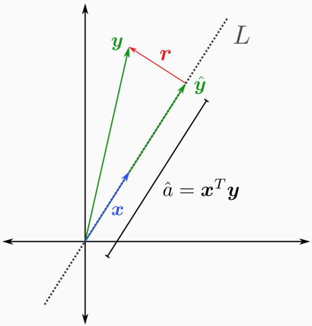


Optimization Problem 1

We can describe SVM learning as follows:

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

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



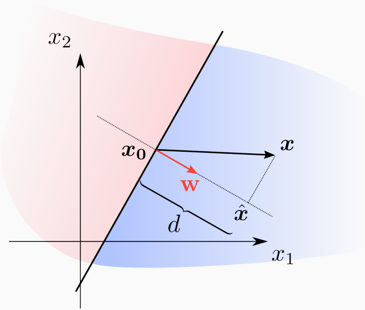
If $\boldsymbol{x}$ is a normalized vector, then the projection of $\boldsymbol{y}$ onto $\boldsymbol{x}$ is

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

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

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

Distance of Point to Hyperplane



- Let $\boldsymbol{w}$ and b be given, describing hyperplane
- Let $\boldsymbol{x}$ be an arbitrary point and $\boldsymbol{x}_0$ an arbitrary point on the hyperplane
- $\boldsymbol{x} - \boldsymbol{x}_0$ is the vector starting at $\boldsymbol{x}_0$ and pointing to $\boldsymbol{x}$
- Let $\hat{x}$ be the projection of $\boldsymbol{x} - \boldsymbol{x}_0$ onto $\boldsymbol{w}$
- $\|\hat{\boldsymbol{x}}\|$ is the Euclidean distance between $\boldsymbol{x}$ and the hyperplane

Distance of Point to Hyperplane cont'd

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

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

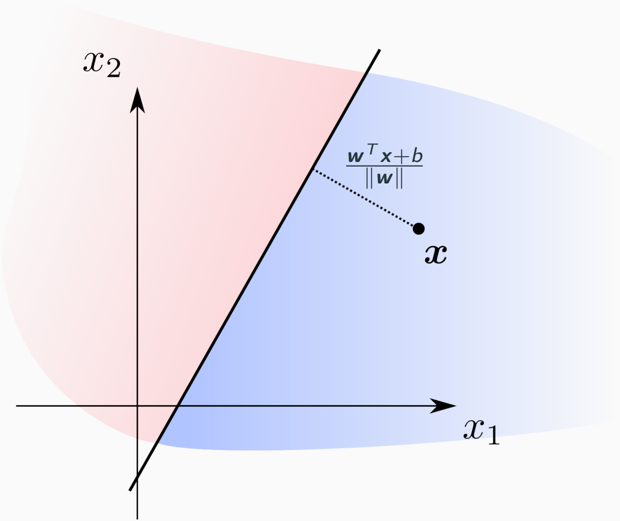
Since $\boldsymbol{x}_0$ lies on the hyperplane, $\boldsymbol{w}^T\boldsymbol{x}_0 + b = 0$. Thus $\boldsymbol{w}^T\boldsymbol{x}_0 = -b$ and

$$d = \frac{\left|\boldsymbol{w}^T\boldsymbol{x} + b\right|}{\|\boldsymbol{w}\|}$$

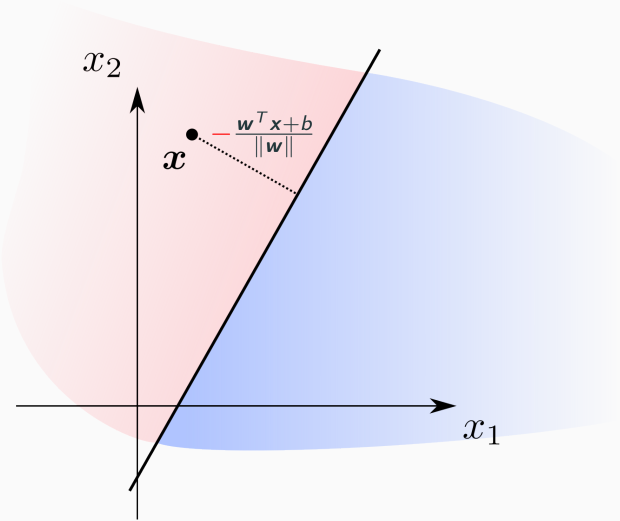
If we remove the absolute value we get a **signed distance**

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

(Signed) Distance between Point and Hyperplane Example



(Signed) Distance between Point and Hyperplane Example

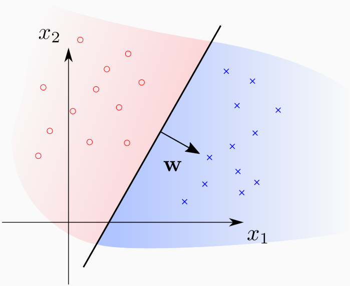


Optimization Problem 2 (formalizing Problem 1)

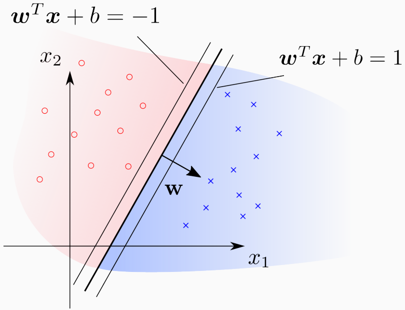
$$\underset{\boldsymbol{w},b}{\text{maximize}} \quad \overbrace{\min_i \frac{y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)}{\|\boldsymbol{w}\|}}^{\text{margin}}$$

- For any separating hyperplane, $\frac{y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)}{\|\boldsymbol{w}\|}$ is the correct distance, due to multiplication with $y_i \in \{-1,+1\}$
- Hence, the objective $\min_i \frac{y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)}{\|\boldsymbol{w}\|}$ 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 $\|\boldsymbol{w}\|$ make the problem **non-convex**, i.e. it is hard to find a global optimum

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

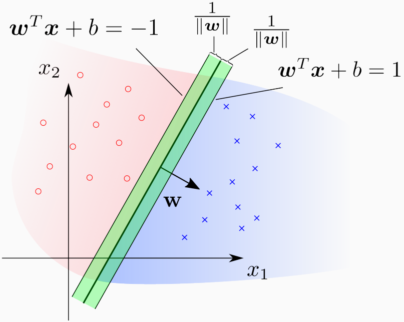


- Let $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ be a separating hyperplane
- Introduce **margin hyperplanes** $\boldsymbol{w}^T\boldsymbol{x} + b = -1$ and $\boldsymbol{w}^T\boldsymbol{x} + b = 1$, i.e. shifted versions of the separating hyperplane

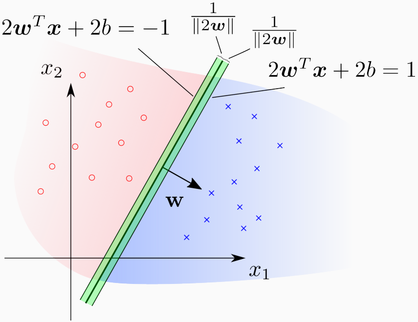


Margin Hyperplanes

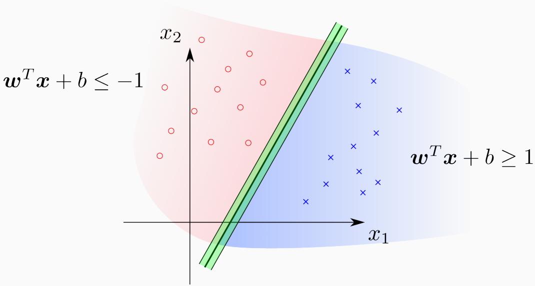
- Let $\boldsymbol{w}^T \boldsymbol{x} + b = 0$ be a separating hyperplane
- Introduce **margin hyperplanes** $\boldsymbol{w}^T \boldsymbol{x} + b = -1$ and $\boldsymbol{w}^T \boldsymbol{x} + b = 1$, i.e. shifted versions of the separating hyperplane
- The distance of the margin hyperplanes to the separating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



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



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



Working with Margins

Margin Constraints:

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

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

Thus, SVM learning can now be described as

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

Optimization Problem 3 (equivalent Problem 2)

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

Optimization Problem 3 (equivalent Problem 2)

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

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

Optimization Problem 3 (equivalent Problem 2)

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

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

Optimization Problem 4 (final, equivalent to Problem 3)

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

- Standard formulation of SVMs
- **Quadratic program**
- Quadratic objective $\frac{1}{2}\|\boldsymbol{w}\|^2$
- Linear constraints $y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1$
- 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?

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

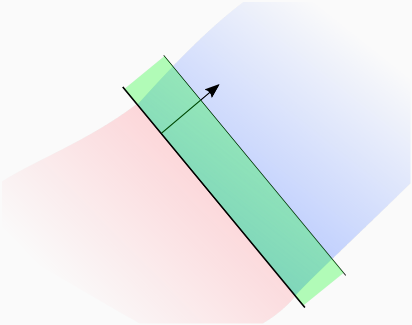
What if data is not linearly separable?

$$\underset{\boldsymbol{w},b}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 \qquad \text{for } i = 1 \ldots N$$

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

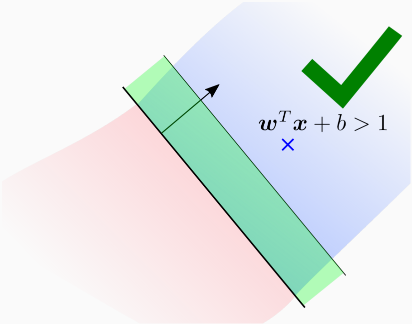
How can we deal with that?

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



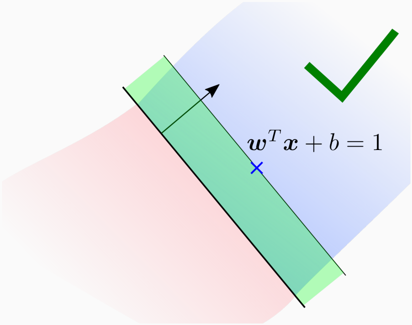
Margin Constraints $y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)\geq 1$

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



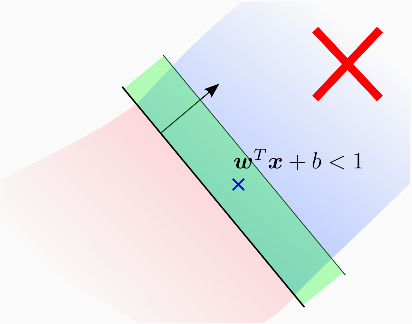
Margin Constraints $y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)\geq 1$

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



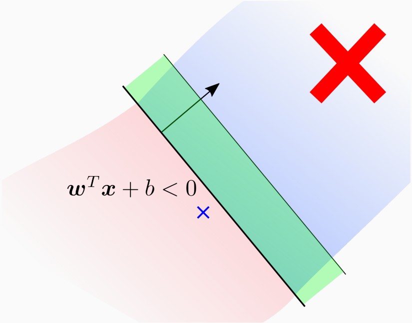
Margin Constraints $y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)\geq 1$

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



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

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



- 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:

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

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

SVM with Soft-Margin

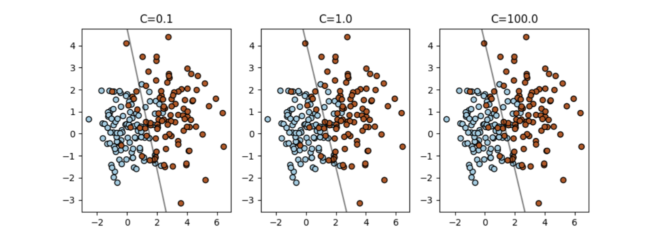
$$
\begin{aligned}
\underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_i \xi^{(i)} \\
\text{s.t.} \quad & y_i \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b \right) \geq 1 - \xi^{(i)} \\
& \xi^{(i)} \geq 0 \qquad\qquad\qquad \text{for } i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

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

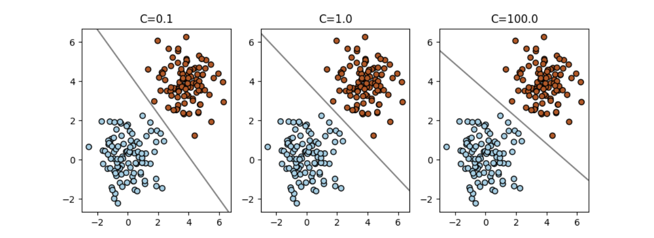
SVM with Soft-Margin (scikit-learn)

Example

Linearly Non-separable Data



Linearly Separable Data



Dual SVM and the Kernel Trick

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

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

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

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

Dual SVM Problem

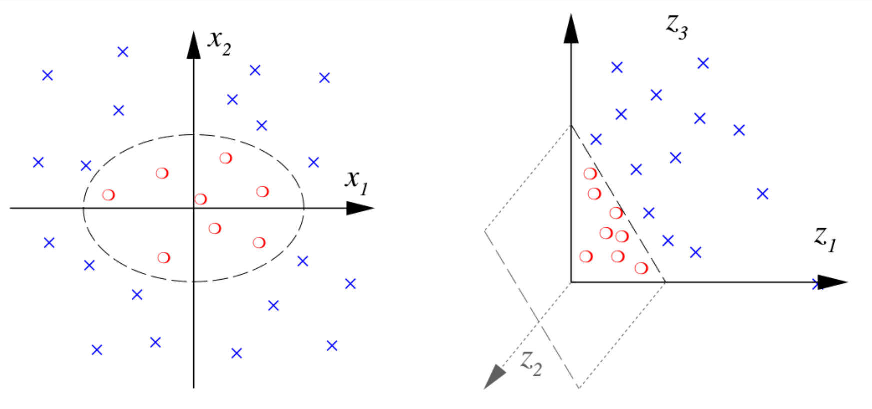
$$\begin{aligned}\underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda^{(i)}\lambda^{(j)}\, y^{(i)}y^{(j)}\, \boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)} + \sum_i \lambda^{(i)} \\ \text{s.t.} \quad & \sum_i \lambda^{(i)}y^{(i)} = 0 \\ & 0 \le \lambda^{(i)}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

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

Dual vs. Primal SVM

- Solving primal SVM: $\Omega(DN)$
- Solving dual SVM: $\Omega(N^2)$
- Dual SVM depends only on inner products ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)}$, 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.

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

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

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 75×10^6 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 ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)}$ of input vectors
- Each ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)}$ is a scalar, regardless of input dimensionality, and is a **similarity measure** between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$
- We can replace now ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)}$ with $\phi(\boldsymbol{x}^{(i)})^T \phi(\boldsymbol{x}^{(j)})$, 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 $\phi(\boldsymbol{x}^{(i)})^T \phi(\boldsymbol{x}^{(j)})$?

Kernel

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

Positive Definite Kernel

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

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

Representer Theorem

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

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

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

The Kernel Trick

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

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

    - **Polynomial kernel** corresponds to ϕ computing all polynomials up to degree p

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

    - **Gaussian kernel** correspond to infinite feature space ϕ (functional space)

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

    - ...

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 ϕ explicitly!**
- **Generalizes many learning algorithms:**
  - SVMs
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - ...
- **Embed "non-vector" data in vector space:**
  - Strings
  - Graphs
  - ...

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

Kernelized SVMs

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

Kernelized SVMs

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

Kernelized SVMs

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

- Quadratic program (convex)
- Delivers globally optimal solution $\boldsymbol{\lambda}^*$
- How do we classify with $\boldsymbol{\lambda}^*$?

Classifying with Kernelized SVMs

Recall that

- $\boldsymbol{w}^* = \sum_i \lambda^{(i)*} y^{(i)} \phi(\boldsymbol{x}^{(i)})$
- $b^* = \boldsymbol{w}^{*T} \phi(\boldsymbol{x}^{(i)}) - y^{(i)}$ (for any support vector)

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

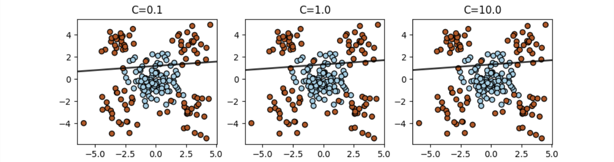
$$f(\boldsymbol{x}) = sgn\left(\sum_i \lambda^{(i)*} y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) - b^*\right)$$

where

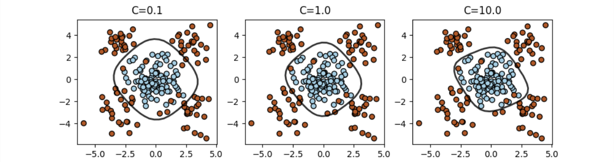
$$b^* = \sum_j \lambda^{(j)*} y^{(j)} k(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)}) - y^{(i)} \quad \text{(for any } i \text{ where } \lambda^{(i)} > 0)$$

Effect of Different Kernels (sklearn)

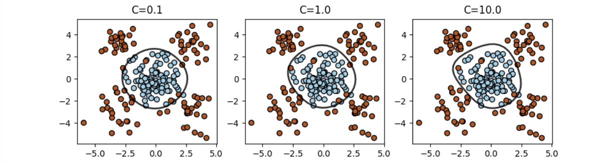
```
classifier = svm.SVC(kernel='linear', C=C)
```




```
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 ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)} \Rightarrow$ amenable to the kernel trick
- The Kernel Trick
    - replace ${\boldsymbol{x}^{(i)}}^T \boldsymbol{x}^{(j)}$ with $\phi(\boldsymbol{x}^{(i)})^T \phi(\boldsymbol{x}^{(j)})$ which can be cheaply computed via $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$
    - 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