# Support vector machines

Pavlo Mozharovskyi[1]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris

Machine learning

Paris, March 12, 2022

# Today

Vapnik-Chervonenkis theory, simplest case

The support vector machine
- Optimal margin classifier
- Introducing kernels (1992)
- Allowing for misclassification: soft margin (1995)

Implementation

# Literature

**Learning materials** include but are not limited to:

- Hastie, T., Tibshirani, R., and Friedman, J. (2009).
  *The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition)*.
  Springer.
  - Section $12.\{1, 2, 3.1, 3.2\}$.
- Slides of the lecture.
- Boser, B. E., Guyon, I. M., and V. N. Vapnik (1992).
  A training algorithm for optimal margin classifiers.
  In: *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, Pittsburgh, ACM, 5, 144–152.
- Cortes, C. and Vapnik, V. (1995).
  Support-vector networks.
  *Machine learning*, 20, 273–297.
- Vapnik, V. N. (1998).
  *Statistical Learning Theory*.
  John Wiley & Sons.

# Binary supervised classification (reminder)

Notation:

- **Given:** for the random pair $(X, Y)$ in $\mathbb{R}^d \times \{-1, 1\}$ consisting of a random observation $X$ and its random binary label $Y$ (class), a sample of $n$ i.i.d.: $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$.
- **Goal:** predict the label of the new (unseen before) observation $\boldsymbol{x}$.
- **Method:** construct a classification rule:

$$g \,:\, \mathbb{R}^d \to \{-1, 1\}\,,\, \boldsymbol{x} \mapsto g(\boldsymbol{x})\,,$$

  so $g(\boldsymbol{x})$ is the prediction of the label for observation $\boldsymbol{x}$.
- **Criterion:** of the performance of $g$ is the **error probability**:

$$R(g) = \mathbb{P}[g(X) \neq Y] = \mathbb{E}[\mathbb{1}\big(g(X) \neq Y\big)]\,.$$

- **The best solution:** is to know the distribution of (X,Y):

$$g(\boldsymbol{x}) = \text{sign}\big(2\mathbb{E}[Y|X = \boldsymbol{x}] - 1 > 0\big)\,.$$

# Contents

# Glivenko-Cantelli theorem

Consider the classification rule for a new observation $\boldsymbol{x}$ given the weights vector $\boldsymbol{w}$:

$$g(\boldsymbol{x}, \boldsymbol{w}) = \begin{cases} 1 & \text{if } \boldsymbol{w}^T \boldsymbol{x} > 0\,, \\ -1 & \text{otherwise}\,. \end{cases}$$

What can be said about the **error probability**, *i.e.* about the relationship between

$$\mathbb{P}\big(g(X, \boldsymbol{w}) \neq Y\big) = \int_{\mathbb{R}^d} \mathbb{1}\big(g(\boldsymbol{x}, \boldsymbol{w}) \neq Y\big)\, dF_X \quad \text{and} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\big(g(\boldsymbol{x}_i, \boldsymbol{w}) \neq y_i\big)\,?$$

Let $X_1, \ldots, X_n$ be a random sample on $\mathbb{R}$. The **empirical distribution function** is defined as

$$\mathbb{F}_n(t) = \frac{1}{n}\sum \mathbb{1}(X_i \leq t)\,.$$

## Theorem (Glivenko-Cantelli)

*If $X_1, X_2, \ldots$ are i.i.d. random variables with distribution function $F$, then*

$$\|\mathbb{F}_n - F\|_\infty = \sup_{x \in \mathbb{R}} |\mathbb{F}_n(x) - F(x)| \xrightarrow{a.s.} 0\,.$$

# Uniform one-sided convergence

Under *additional* conditions, for $g(\boldsymbol{x}, \boldsymbol{w})$ and a probability measure $F_X$, for any $\epsilon > 0$ it holds

$$\mathbb{P}\Big\{\sup_{\boldsymbol{w}}\Big( \underbrace{\mathbb{P}\big(g(X, \boldsymbol{w}) \neq Y\big)}_{L\big(g(\cdot,\boldsymbol{w})\big)} - \underbrace{\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\big(g(X_i, \boldsymbol{w}) \neq Y_i\big)}_{L_{emp}\big(g(\cdot,\boldsymbol{w})\big)} \Big) > \epsilon\Big\} \underset{n\to\infty}{\longrightarrow} 0\,.$$

What can be said about the **rate of convergence**?

Regard **finite set of classification rules** $g(\boldsymbol{x}, \boldsymbol{w}_k)$, $k = 1, ..., N$. The restriction is naturally posed by the finite number of elements in the training set.

$$\begin{aligned}
&\mathbb{P}\Big\{\sup_{k\in\{1,\ldots,N\}}\Big( L(g(\cdot, \boldsymbol{w}_k)) - L_{emp}(g(\cdot, \boldsymbol{w}_k)) \Big) > \epsilon\Big\} \\
&\quad \leq \sum_{k=1}^{N} \mathbb{P}\Big\{\Big( L(g(\cdot, \boldsymbol{w}_k)) - L_{emp}(g(\cdot, \boldsymbol{w}_k)) \Big) > \epsilon\Big\}
\end{aligned}$$

# Uniform one-sided convergence

## Theorem (Chernoff-Hoeffding, Bernoulli scheme)

*If $X_1, \ldots, X_n$ are i.i.d. random variables taking values in $\{0,1\}$, then for any $\epsilon > 0$ it holds*

$$\mathbb{P}\Big(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \epsilon\Big) < e^{-2\epsilon^2 n}.$$

This allows for:

$$\begin{aligned}
& \sum_{k=1}^{N} \mathbb{P}\Big\{\Big( L(g(\cdot,\boldsymbol{w}_k)) - L_{emp}(g(\cdot,\boldsymbol{w}_k)) \Big) > \epsilon\Big\} \\
= \quad & \sum_{k=1}^{N} \mathbb{P}\Big\{\Big(\underbrace{\mathbb{P}(g(X,\boldsymbol{w}_k) \neq Y)}_{\mathbb{E}\left[\mathbb{1}\left(g(X,\boldsymbol{w}_k)\neq Y\right)\right]} - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(g(X_i,\boldsymbol{w}_k) \neq Y_i)\Big) > \epsilon\Big\} \\
\leq \quad & N e^{-2\epsilon^2 n}.
\end{aligned}$$

# Vapnik-Chervonenkis inequality

So:

$$\mathbb{P}\Big\{ \sup_{k\in\{1,\ldots,N\}} \Big( L\big(g(\cdot,\boldsymbol{w}_k)\big) - L_{emp}\big(g(\cdot,\boldsymbol{w}_k)\big) \Big) > \epsilon \Big\} \leq Ne^{-2\epsilon^2 n}\,.$$

Let us fix this probability having chosen $0 < \eta \leq 1$, by that maintaining reliability $1-\eta$:

$$Ne^{-2\epsilon^2 n} = \eta \qquad \text{or equivalently} \qquad \epsilon = \sqrt{\frac{\log N - \log\eta}{2n}}\,.$$

This allows for the following result:

## Theorem (Vapnik-Chervonenkis, 1974)

*If from a set consisting of $N$ classification rules a rule $g(\cdot,\boldsymbol{w})$ is chosen, which delivers empirical risk $L_{emp}\big(g(\cdot,\boldsymbol{w})\big)$, then with reliability $1-\eta$ one can state that the error probability $L\big(g(\cdot,\boldsymbol{w})\big)$ is bounded from above as follows*

$$L\big(g(\cdot,\boldsymbol{w})\big) \leq L_{emp}\big(g(\cdot,\boldsymbol{w})\big) + \sqrt{\frac{\log N - \log\eta}{2n}}\,.$$

# Particular case: linear rule

Let us try to estimate $N$ for the linear classification rule.

The number $\Phi(d, n)$ of all possible separations of $n$ points in $\mathbb{R}^d$ by a hyperplane via the origin is computed as

$$\Phi(d, n) = \begin{cases} 2\sum_{l=0}^{d-1} \binom{n-1}{l} & \text{if } d \leq n\,, \\ 2^n & \text{otherwise}\,. \end{cases}$$

For $d \leq n$, one can approximate it from above using:

$$\Phi(d, n) \leq 3\frac{n^{d-1}}{(d-1)!} \leq n^d\,.$$

Plugging this into the Vapnik-Chervonenkis inequality gives:

$$L\big(g(\cdot, \boldsymbol{w})\big) \leq L_{emp}\big(g(\cdot, \boldsymbol{w})\big) + \sqrt{\frac{d\log n - \log\eta}{2n}}\,.$$

# Contents

# The principle

- The conservative upper **bound of Vapnik and Chervonenkis** is very **pessimistic**, as even for a linear classification rule a very large training data set is required to guarantee meaningfulness of the achieved empirical risk.

- As an example, consider the case of two **linearly separable** training classes. Even in this case, only **little can be said** about probability of **points from one class inside the other** one.

- Sticking to this "trivial" case, the safest **separating hyperplane** would be the one having **maximal** and equal **margin** to each of the classes.

- Finding such a hyperplane in a systematic way constitutes the main idea of the **optimal margin hyperplane** algorithm.

# Contents

# Optimal margin hyperplane

- Let the training sample consist of $n$ pairs $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$ taking values in $\mathbb{R}^d \times \{-1, 1\}$.
- This set is said to be **linearly separable** if there exist a non-zero vector $\boldsymbol{\psi} \in \mathbb{R}^d$ and a scalar $b \in \mathbb{R}$ such that the $n$ following inequalities hold:

$$\begin{aligned} \boldsymbol{\psi}^T \boldsymbol{x}_i + b &\geq 0 && \text{if} \quad y_i = 1\,, \\ \boldsymbol{\psi}^T \boldsymbol{x}_i + b &\leq 0 && \text{if} \quad y_i = -1\,. \end{aligned}$$

- Instead of simply requiring separation (the parts "$\geq 0$" and "$\leq 0$" in the above inequality) one can **introduce margin** $M > 0$, *i.e.*, require the distance between any two points stemming from different classes — in projection onto $\boldsymbol{\psi}$ — be at least $2M$.
- Involving the output (in this notation corresponding to the sign) allows for rewriting the above (restricting) inequalities in the following way:

$$\frac{y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b)}{\|\boldsymbol{\psi}\|} \geq M\,, \quad i = 1, \ldots, n\,.$$

# Optimal margin hyperplane

- The objective of the training algorithm is then to find the parameter vector $\boldsymbol{\psi}$ that maximizes $M$:

$$
\begin{aligned}
& M^* = \max M \\
& \text{subject to} \quad \|\boldsymbol{\psi}\| = 1\,, \\
& \qquad\qquad\quad y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) \geq M\,, \quad i = 1, \dots, n\,.
\end{aligned}
$$

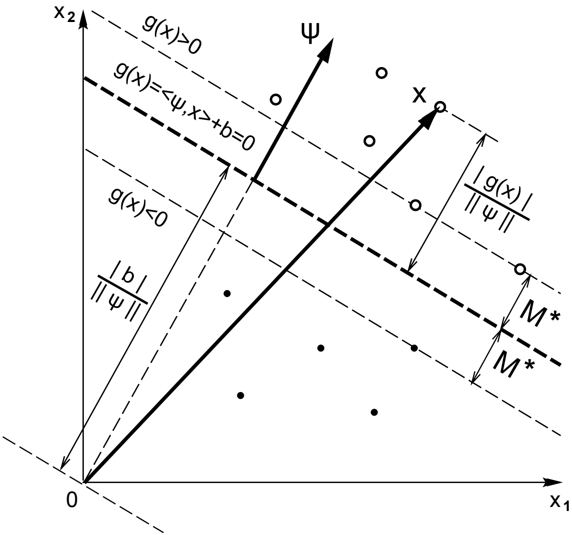
- The (last) bound is attained for those patterns satisfying

$$\min_{i \in \{1,\dots,n\}} y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) = M^*\,.$$

- These patterns are called the **support vectors** of the decision boundary.
- Thus, the problem of finding a hyperplane with maximum margin can be seen as a **minimax** problem:

$$\max_{\boldsymbol{\psi} \in \mathbb{R}^d,\, \|\boldsymbol{\psi}\|=1} \ \min_{i \in \{1,\dots,n\}} \ y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b)\,.$$

# Optimal margin hyperplane : illustration



# Optimal margin hyperplane

- Instead of fixing the norm of $\boldsymbol{\psi}$, the product of the margin $M$ and the norm of $\boldsymbol{\psi}$ can be fixed, *e.g.* by:
$$M\|\boldsymbol{\psi}\| = 1\,.$$
- Now, **maximizing** the **margin** $M$ is **equivalent to minimizing** the **norm** $\|\boldsymbol{\psi}\|$.
- Then the problem of finding a maximum margin separating hyperplane, characterized by $\boldsymbol{\psi}$, reduces to solving the following **quadratic optimization problem**:
$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{\psi}\|^2 \\ &\text{subject to} \quad y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) \geq 1\,, \quad i = 1, ..., n\,. \end{aligned}$$
- The maximum margin is:
$$M^* = \frac{1}{\|\boldsymbol{\psi}^*\|}\,.$$
- This approach is **impractical**:
  - if the **dimension** $d$ is large or **infinite**,
  - because no information about **support vectors** is gained.

# Optimal margin hyperplane: Lagrangian

- Construct a Lagrangian:

$$L(\boldsymbol{\psi}, b, \boldsymbol{\Lambda}) = \frac{1}{2}\boldsymbol{\psi}^T\boldsymbol{\psi} - \sum_{i=1}^{n} \alpha_i \big(y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) - 1\big)$$

  with $\boldsymbol{\Lambda} = (\alpha_1, ..., \alpha_n)^T$ being the vector of non-negative **Lagrange multipliers** corresponding to the inequality constraints.

- The solution to the optimization problem is determined by the saddle point of this Lagrangian in the $(d + 1 + n)$-dimensional space of $\boldsymbol{\psi}$, $b$, and $\boldsymbol{\Lambda}$.

- The **minimum** should be taken w.r.t. the parameters $\boldsymbol{\psi}$ and $b$, the **maximum** should be taken w.r.t. the Lagrange multipliers $\boldsymbol{\Lambda}$.

# Optimal margin hyperplane: Lagrangian

- At the point of minimum (w.r.t. $\boldsymbol{\psi}$ and $b$) one obtains:

$$\left.\frac{\partial L(\boldsymbol{\psi}, b, \boldsymbol{\Lambda})}{\partial \boldsymbol{\psi}}\right|_{\boldsymbol{\psi}=\boldsymbol{\psi}^*} = \left(\boldsymbol{\psi}^* - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i\right) = 0\,,$$

$$\left.\frac{\partial L(\boldsymbol{\psi}, b, \boldsymbol{\Lambda})}{\partial b}\right|_{b=b^*} = \sum_{i=1}^{n} y_i \alpha_i = 0\,.$$

- From the upper equality one can derive:

$$\boldsymbol{\psi}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i\,.$$

- This means that the optimal hyperplane can be written as a **linear combination of training observations**.

- Only training observations $\boldsymbol{x}_i$ with (strictly) positive Lagrange multipliers (*i.e.* with $\alpha_i > 0$) have an efficient contribution to the sum — the **support vectors**.

# Optimal margin hyperplane: Lagrangian

- Substitution of the minimum conditions into the Lagrangian yields the following optimization problem:

$$
\begin{aligned}
\max W(\boldsymbol{\Lambda}) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\
\text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \,, \\
& \alpha_i \geq 0 \,, \quad i = 1, \ldots, n \,.
\end{aligned}
$$

- Usually it is written in the matrix form:

$$
\begin{aligned}
\max W(\boldsymbol{\Lambda}) = & \boldsymbol{\Lambda}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\Lambda}^T \boldsymbol{D} \boldsymbol{\Lambda} \\
\text{subject to} \quad & \boldsymbol{\Lambda}^T \boldsymbol{Y} = 0 \,, \\
& \boldsymbol{\Lambda} \geq \mathbf{0}
\end{aligned}
$$

with $\boldsymbol{D}$ being a $(n \times n)$-dimensional matrix with entries $D_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$, $\boldsymbol{Y} = (y_1, \ldots, y_n)^T$, and $\mathbf{0}$ and $\mathbf{1}$ standing for $n$-dimensional vectors of zeros and ones.

# Optimal margin hyperplane: classification

- After the optimal pair $(\boldsymbol{\psi}^*, b^*)$ is obtained, classification of an observation $\boldsymbol{x} \in \mathbb{R}^d$ reduces to determining its position in the projection onto $\boldsymbol{\psi}^*$:

$$\begin{aligned} g(\boldsymbol{x}) &= \operatorname{sign}({\boldsymbol{\psi}^*}^T \boldsymbol{x} + b^*) \\ &= \operatorname{sign}\left( \sum_{i=1}^{n} y_i \alpha_i^* \boldsymbol{x}_i^T \boldsymbol{x} + b^* \right). \end{aligned}$$

- From this it becomes clear how to calculate $b^*$: it should position the separating hyperplane exactly in the middle between two support vectors from different classes, in the projection onto $\boldsymbol{\psi}^*$:

$$\begin{aligned} b^* &= -\frac{1}{2} \left( {\boldsymbol{\psi}^*}^T \boldsymbol{x}_A + {\boldsymbol{\psi}^*}^T \boldsymbol{x}_B \right) \\ &= -\frac{1}{2} \sum_{i=1}^{n} y_i \alpha_i^* (\boldsymbol{x}_i^T \boldsymbol{x}_A + \boldsymbol{x}_i^T \boldsymbol{x}_B). \end{aligned}$$

  with $\boldsymbol{x}_A \in \{\boldsymbol{x}_i \,:\, y_i = 1\,,\, \alpha_i^* > 0,\, i = 1, \ldots, n\}$ and $\boldsymbol{x}_B \in \{\boldsymbol{x}_i \,:\, y_i = -1\,,\, \alpha_i^* > 0,\, i = 1, \ldots, n\}$.
- Only **support vectors** influence the classification rule. (Analogy with a mine field on the front line between two enemies.)

# Optimal margin classifier (algorithm)

## Finding the optimal margin hyperplane (training)

**Input:** Training sample $\big((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\big) \subset \mathbb{R}^d \times \{-1, 1\}$.

1. Solve the constraint quadratic optimization problem to obtain $\boldsymbol{\Lambda}^* = (\alpha_1^*, ..., \alpha_n^*)^T$:

$$\begin{aligned} \max \quad & \boldsymbol{\Lambda}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\Lambda}^T \boldsymbol{D} \boldsymbol{\Lambda} \\ \text{subject to} \quad & \boldsymbol{\Lambda}^T \boldsymbol{Y} = 0\,, \\ & \boldsymbol{\Lambda} \geq \mathbf{0}\,. \end{aligned}$$

2. Taking any two support vectors from opposite classes $\boldsymbol{x}_A \in \{\boldsymbol{x}_i \,:\, y_i = 1\,,\, \alpha_i^* > 0\}$ and $\boldsymbol{x}_B \in \{\boldsymbol{x}_i \,:\, y_i = -1\,,\, \alpha_i^* > 0\}$, calculate the threshold:

$$b^* = -\frac{1}{2} \sum_{i=1}^{n} y_i \alpha_i^* (\boldsymbol{x}_i^T \boldsymbol{x}_A + \boldsymbol{x}_i^T \boldsymbol{x}_B)\,.$$

**Output:** The classifier: $g(\boldsymbol{x}) = \text{sign}\Big(\sum_{i=1}^n y_i \alpha_i^* \boldsymbol{x}_i^T \boldsymbol{x} + b^*\Big)$.

# Optimal margin classifier: some comments

- The training phase is reduced to solving a problem of **quadratic optimization**, which **is** usually **computationally tractable**.
- The time of the training algorithm depends on dimension $d$ only when calculating the matrix of quadratic coefficients $\boldsymbol{D}$, the **dimension** of the original space **is irrelevant for the optimization time**.
- As **only** the **support vectors** are relevant, only these **should be stored** for the classification rule.
- The problem **can be solved iteratively by chunks**, as in each (previous) chunk only support vectors are important (for further chunks).

But:

- **Linear** classification **rule has poor** approximation **performance**.
- **Misclassification is not allowed**.

# Contents

# Convolution of the inner product

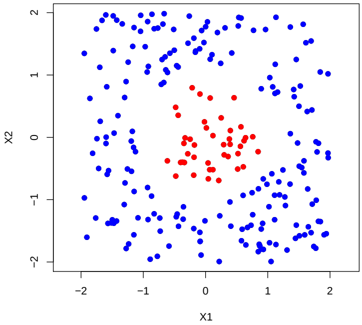
- The algorithm described above constructs a hyperplane (defining by that linear classification rule) in the input space $\mathbb{R}^d$.

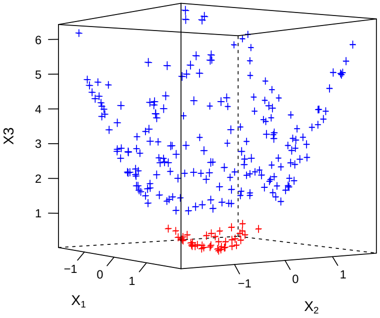
- **Idea:** To increase the approximation power of the classification rule but keep its algorithmic linearity, one maps the input space to a feature space, *i.e.* transforms the $d$-dimensional input vector $\boldsymbol{x}$ into a $D$-dimensional feature space through a choice of a $D$-dimensional vector function $\phi$:
$$\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D .$$

- Then, a $D$-dimensional linear separator $(\boldsymbol{\psi}, b) \in \mathbb{R}^D \times \mathbb{R}$ is constructed for the set of transformed vectors:
$$\boldsymbol{\phi}(\boldsymbol{x}_i) = \big(\phi_1(\boldsymbol{x}_i), \phi_2(\boldsymbol{x}_i), ..., \phi_D(\boldsymbol{x}_i)\big) \in \mathbb{R}^D , \quad i = 1, ..., n .$$

- Note: $\mathbb{R}^D$ can be of infinite dimension.

# Convolution of the inner product




$$(x_1, x_2)$$

$$\mapsto \big(\phi(x_1,\, x_2)\big)$$
$$= (x_1,\, x_2,\, x_1^2 + x_2^2)$$

# Convolution of the inner product

- The classification of an unknown vector $\boldsymbol{x}$ is done by first transforming it into the feature space

$$\boldsymbol{x} \mapsto \phi(\boldsymbol{x}),$$

  and then classifying the featured vector with

$$g(\boldsymbol{x}) = \operatorname{sign}\left(\boldsymbol{\psi}^{*T} \phi(\boldsymbol{x}) + b^*\right).$$

- According to the properties of the classifier, it can be written as a linear combination of the support vectors (in the feature space):

$$\boldsymbol{\psi}^* = \sum_{i=1}^{n} y_i \alpha_i^* \phi(\boldsymbol{x}_i).$$

- The linearity of the inner product implies that the **classifier** $g(\boldsymbol{x})$ **only depends on the inner products**:

$$g(\boldsymbol{x}) = \operatorname{sign}\left(\boldsymbol{\psi}^{*T} \phi(\boldsymbol{x}) + b^*\right) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i \alpha_i^* \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + b^*\right).$$

- The **quadratic problem depends only inner products** as well.

# Convolution of the inner product

- Consider the general form of the inner product in a Hilbert space:

$$\phi(\boldsymbol{u})^T\phi(\boldsymbol{v}) = K(\boldsymbol{u}, \boldsymbol{v}) .$$

- According to Hilbert-Schmidt Theory any symmetric function $K(\boldsymbol{u}, \boldsymbol{v})$, with $K(\boldsymbol{u}, \boldsymbol{v}) \in L_2$, can be expanded in the form:

$$K(\boldsymbol{u}, \boldsymbol{v}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\boldsymbol{u}) \phi_j(\boldsymbol{v})$$

  with $\lambda_i \in \mathbb{R}$ and $\phi_i$ being eigenvalues and eigenfunctions of the integral operator defined by the kernel $K(\boldsymbol{u}, \boldsymbol{v})$, *i.e.*

$$\int K(\boldsymbol{u}, \boldsymbol{v}) \phi_j(\boldsymbol{u}) d\boldsymbol{u} = \lambda_j \phi_j(\boldsymbol{v}) .$$

- A sufficient condition to ensure that $K(\boldsymbol{u}, \boldsymbol{v})$ defines an inner product in the feature space is that all the eigenvalues $\lambda_i$ are positive.

# Convolution of the inner product

- According to Mercer's theorem, for $\lambda_i$s to be positive, it is necessary and sufficient that

$$\int\int K(\boldsymbol{u},\boldsymbol{v})h(\boldsymbol{u})h(\boldsymbol{v})d\boldsymbol{u}d\boldsymbol{v} > 0$$

  holds for all $h$ such that

$$\int h^2(\boldsymbol{u})d\boldsymbol{u} < \infty\,.$$

- Thus, **functions that satisfy the Mercer's theorem can be used as inner products in the feature space**.

**Examples** of such functions:

- **Gaussian kernel = potential function = radial basis function**:

$$K(\boldsymbol{u},\boldsymbol{v}) = e^{-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2}} = e^{-\gamma\|\boldsymbol{u}-\boldsymbol{v}\|^2}\,.$$

- **Polynomial kernel**:

$$K(\boldsymbol{u},\boldsymbol{v}) = (\boldsymbol{u}^T\boldsymbol{v}+1)^{\beta}\,.$$

# Convolution of the inner product

- Using **different kernel functions** $K(\boldsymbol{u}, \boldsymbol{v})$ as inner products (**with different parameters**, *e.g.*, $\sigma$, $\gamma$, $\beta$) one can construct different learning machines with arbitrary types of decision surfaces.
- To find the optimal coefficient vector $\boldsymbol{\Lambda}^* = (\alpha_1^*, ..., \alpha_n^*)$, threshold $b^*$, and support vectors $\boldsymbol{x}_i$s, one follows the same solution scheme as for the original optimal margin classifier by solving the quadratic optimization problem.
- The only difference consists in using the matrix $\boldsymbol{D}$ with entries:

$$D_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\,, \quad i, j = 1, ..., n\,.$$

- The **decision rule** has then form:

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} y_i \alpha_i^* K(\boldsymbol{x}, \boldsymbol{x}_i) + b^*\,.$$

where one can only restrict to support vectors $\boldsymbol{x}_i$ and their coefficients $\alpha_i^* > 0$.

# Contents

# Soft margin classifier

- The **kernel trick** allows to “ignore” the transform on the algorithmic level.
- Consider the case where the **training data cannot be separated** without error.
- In this case one may want to **separate** the training set **with a minimal number of errors**.
- Let us introduce non-negative variables $\xi_i \geq 0$, $i = 1, ..., n$.
- We can then **minimize** the functional

$$\sum_{i=1}^{n} \xi_i^{\sigma}$$

  for some small $\sigma > 0$ subject to constraints

$$\begin{aligned} y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) &\geq 1 - \xi_i \,, & i &= 1, ..., n \,, \\ \xi_i &\geq 0 \,, & i &= 1, ..., n \,. \end{aligned}$$

- For sufficiently small $\sigma$ the minimized functional describes the number of errors on the training set.

## Soft margin classifier

- In the minimum, strictly positive $\xi_{i_j} > 0$, $j = 1, \ldots, k$ will identify the **minimal subset of training errors**:

$$(\boldsymbol{x}_{i_1}, y_{i_1}),\ (\boldsymbol{x}_{i_2}, y_{i_2}),\ \ldots,\ (\boldsymbol{x}_{i_k}, y_{i_k})\,.$$

- After these data are **excluded**, one can separate the remaining part of the training set without errors using the usual optimal separating hyperplane.
- Formally this can be expressed as:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{\psi}\|^2 + CF\left(\sum_{i=1}^{n} \xi_i^{\sigma}\right) \\ \text{subject to} \quad & y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i\,, \qquad i = 1, \ldots, n\,, \\ & \xi_i \geq 0\,, \qquad i = 1, \ldots, n\,. \end{aligned}$$

  with $F(u)$ being a monotonic convex function and $C$ being a positive constant.
- For sufficiently large $C$ and sufficiently small $\sigma$, the pair $(\boldsymbol{\psi}^*, b^*)$ minimizing this functional will determine the **hyperplane minimizing the number of errors and separating the rest with maximum margin**.

# Soft margin classifier

- However, the problem of finding a hyperplane minimizing number of errors is **NP-complete**.
- For the reasons of computational tractability, we consider the (most commonly used) case:

$$
\begin{aligned}
F(u) &= u \,, \\
\sigma &= 1 \,,
\end{aligned}
$$

  and choose appropriate value for the regularizing constant $C$.

- The problem then becomes:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{\psi}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\boldsymbol{\psi}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \,, \qquad i = 1, \ldots, n \,, \\
& \xi_i \geq 0 \,, \qquad i = 1, \ldots, n \,.
\end{aligned}
$$

# Soft margin classifier

- The corresponding Lagrangian is:

$$L(\boldsymbol{\psi}, b, \boldsymbol{\xi}, \boldsymbol{\Lambda}, \boldsymbol{r}) = \frac{1}{2}\boldsymbol{\psi}^T\boldsymbol{\psi} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\big(y_i(\boldsymbol{\psi}^T\boldsymbol{x}_i + b) - 1 + \xi_i\big) - \sum_{i=1}^{n} r_i\xi_i$$

  with $\boldsymbol{\Lambda} = (\alpha_1, ..., \alpha_n)^T$ and $\boldsymbol{r} = (r_1, ..., r_n)^T$ being the vectors of non-negative **Lagrange multipliers** corresponding to the two groups of inequality constraints.

- The solution to the optimization problem is determined by the saddle point of this Lagrangian in the $(d + 1 + n + n + n)$-dimensional space of $\boldsymbol{\psi}$, $b$, $\boldsymbol{\xi}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{r}$.

- The **minimum** should be taken w.r.t. the parameters $\boldsymbol{\psi}$, $b$, and $\boldsymbol{\xi}$, the **maximum** should be taken w.r.t. the Lagrange multipliers $\boldsymbol{\Lambda}$ and $\boldsymbol{r}$.

# Soft margin classifier

- At the point of minimum (w.r.t. $\boldsymbol{\psi}$, $b$, and $\boldsymbol{\xi}$) one obtains:

$$\frac{\partial L(\boldsymbol{\psi}, b, \boldsymbol{\xi}, \boldsymbol{\Lambda}, \boldsymbol{r})}{\partial \boldsymbol{\psi}}\Big|_{\boldsymbol{\psi}=\boldsymbol{\psi}^*} = \left(\boldsymbol{\psi}^* - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i\right) = 0\,,$$

$$\frac{\partial L(\boldsymbol{\psi}, b, \boldsymbol{\xi}, \boldsymbol{\Lambda}, \boldsymbol{r})}{\partial b}\Big|_{b=b^*} = \sum_{i=1}^{n} y_i \alpha_i = 0\,,$$

$$\frac{\partial L(\boldsymbol{\psi}, b, \boldsymbol{\xi}, \boldsymbol{\Lambda}, \boldsymbol{r})}{\partial \xi_i}\Big|_{\xi_i=\xi_i^*} = C - \alpha_i - r_i = 0\,, \quad i = 1, \ldots, n\,.$$

- This leads to the following quadratic problem:

$$\max W(\boldsymbol{\Lambda}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0\,,$$

$$0 \le \alpha_i \le C\,, \quad i = 1, \ldots, n\,.$$

# Support vector machine (SVM)

- The **training** phase:

$$\begin{aligned} \max \quad & \boldsymbol{\Lambda}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\Lambda}^T \boldsymbol{D} \boldsymbol{\Lambda} \\ \text{subject to} \quad & \boldsymbol{\Lambda}^T \boldsymbol{Y} = 0\,, \\ & \mathbf{0} \leq \boldsymbol{\Lambda} \leq C \mathbf{1}\,, \end{aligned}$$

with $\boldsymbol{Y} = (y_1, ..., y_n)^T$, $\mathbf{0}$ and $\mathbf{1}$ standing for $n$-dimensional vectors of zeros and ones, $C$ being a properly chosen constant, and $\boldsymbol{D}$ being a $(n \times n)$-dimensional matrix with entries

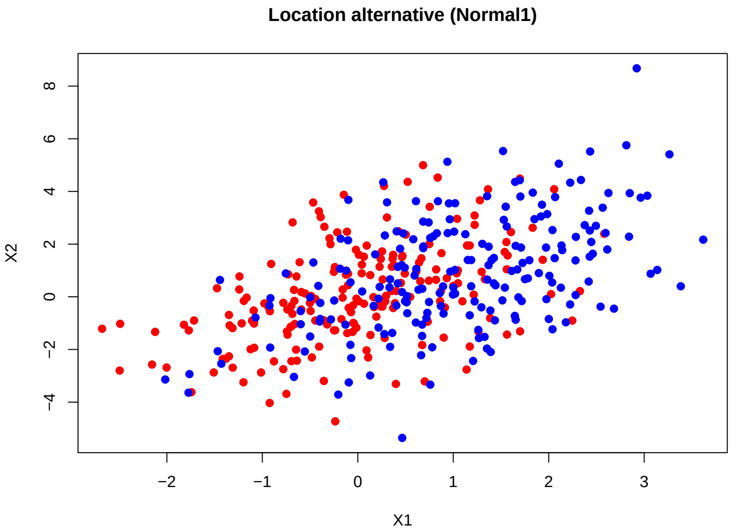
$$D_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\,, \quad i, j = 1, ..., n\,,$$

where $K(\boldsymbol{u}, \boldsymbol{v})$ is a properly chosen kernel function.
The result is the optimal vector $\boldsymbol{\Lambda}^* = (\alpha_1^*, ..., \alpha_n^*)^T$.

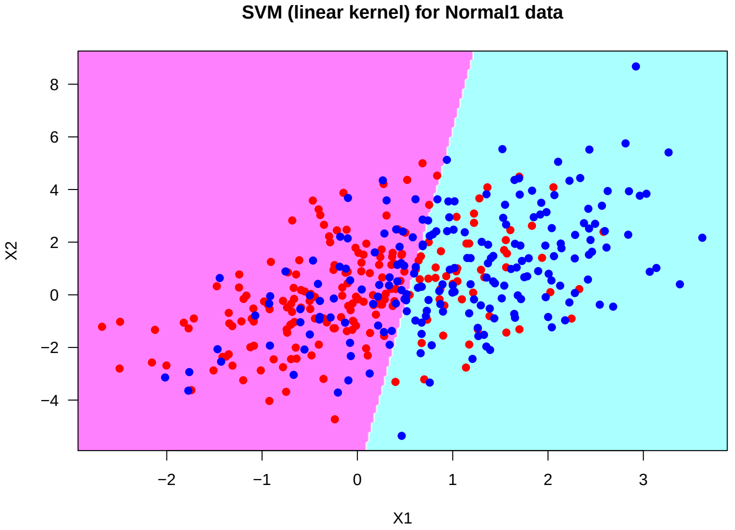
Then, taking any two support vectors $\boldsymbol{x}_{i_A}$ and $\boldsymbol{x}_{i_B}$ from opposite classes, *i.e.* with $i_A \in \arg\max_{j\,:\,y_j=1, \alpha_j^*>0} \sum_{i=1}^n y_i \alpha_i^* K(\boldsymbol{x}_j, \boldsymbol{x}_i)$ and $i_B \in \arg\min_{j\,:\,y_j=-1, \alpha_j^*>0} \sum_{i=1}^n y_i \alpha_i^* K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, calculate threshold:

$$b^* = -\frac{1}{2} \sum_{i=1}^{n} y_i \alpha_i^* \left( K(\boldsymbol{x}_i, \boldsymbol{x}_{i_A}) + K(\boldsymbol{x}_i^T \boldsymbol{x}_{i_B}) \right).$$

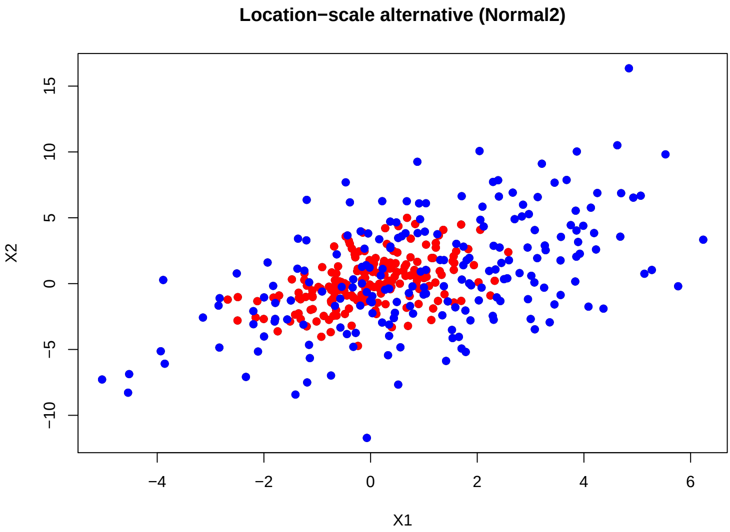
# Normal location alternative



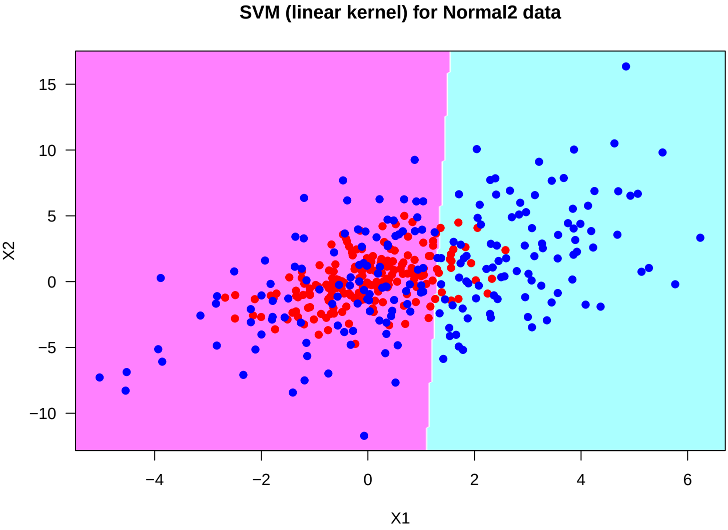
# SVM: normal location alternative



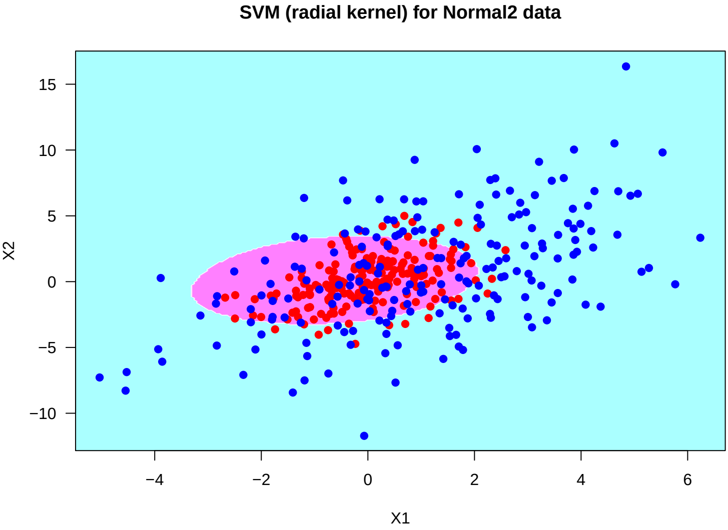
# Normal location-scale alternative



# SVM: normal location-scale alternative



# SVM: normal location-scale alternative



# Contents

# Implementing SVM

**When** implementing and **applying SVM**, its **parameters have to be chosen**:

- **kernel** function,
- **kernel parameter**,
- **regularization constant** (=box constraint).

In practice, these parameters are usually chosen by cross-validation. This process is called **tuning of the SVM**. The SVM possesses certain degree of insensitivity w.r.t. parameters, which can be limited depending on the application of interest.

For `R`-software, SVM is implemented in such packages as, *e.g.*, `e1071`, `kernlab`, `klaR`, `svmpath`.
For an overview, see, *e.g.*:

- Karatzoglou, A., Meyer, D., and Hornik, K. (2006).
  Support vector machines in `R`.
  *Journal of Statistical Software*, 15(9).

# Thank you for your attention!

Thank you for your attention!

# And some references


- Hastie, T., Tibshirani, R., and Friedman, J. (2009).
  *The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition)*.
  Springer.

- Devroye, L., Gyöfri, L., Lugosi, G. (1996).
  *A Probabilistic Theory of Pattern Recognition*.
  Springer.

- Vapnik, V. N. (1998).
  *Statistical Learning Theory*.
  John Wiley & Sons.

- Haykin, S. (2009).
  *Neural Networks and Learning Machines (Third Edition)*.
  Pearson.