Pattern Recognition Advanced

The Support Vector Machine
Introduction to Kernel Methods

mcuturi@i.kyoto-u.ac.jp

Supervised Learning

Many observations of the same data type, with labels

- we consider a database $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$,
- each datapoint $\mathbf{x}_j$ is represented as a vector of features $\mathbf{x}_j = \begin{bmatrix} x_{\mathbf{1},j} \\ x_{\mathbf{2},j} \\ \vdots \\ x_{\boldsymbol{d},j} \end{bmatrix}$
- To each observation is associated a **label** y_j...
  - If $y_j \in \mathbb{R}$, we have a **regression** problem.
  - If $y_j \in \mathcal{S}$ where $\mathcal{S}$ is a finite set, **multiclass classification**.
  - If $\mathcal{S}$ only has two elements, **binary classification**.

Supervised Learning: Binary Classification

$\boxed{\text{Examples of Binary Classes}}$

- Using elementary measurements, guess if someone **has or not** a disease that is
  - difficult to detect at an early stage
  - difficult to measure directly (fetus)
- Classify chemical compounds into **toxic / nontoxic**
- Classify a scanned piece of luggage as **suspicious/not suspicious**
- Classify body tumor as **begign/malign** to detect cancer
- Detect whether an image's primary content is **people** or any other **object**
- *etc.*

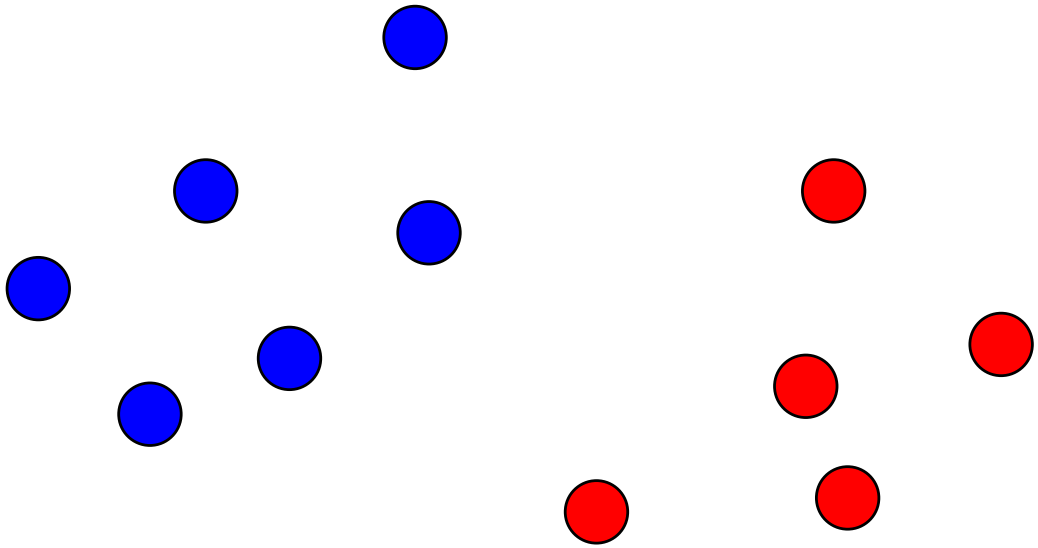
Data

- **Data**: instances $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_N$.
- To infer a "yes/no" rule, we need the **corresponding answer** for each vector.
- We consider thus a set of **pairs of (vector,bit)**

$$\text{"training set"} = \left\{ \left(\mathbf{x}_j = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{d,j} \end{bmatrix} \in \mathbb{R}^d, \ \mathbf{y}_j \in \{0,1\} \right)_{j=1..N} \right\}$$

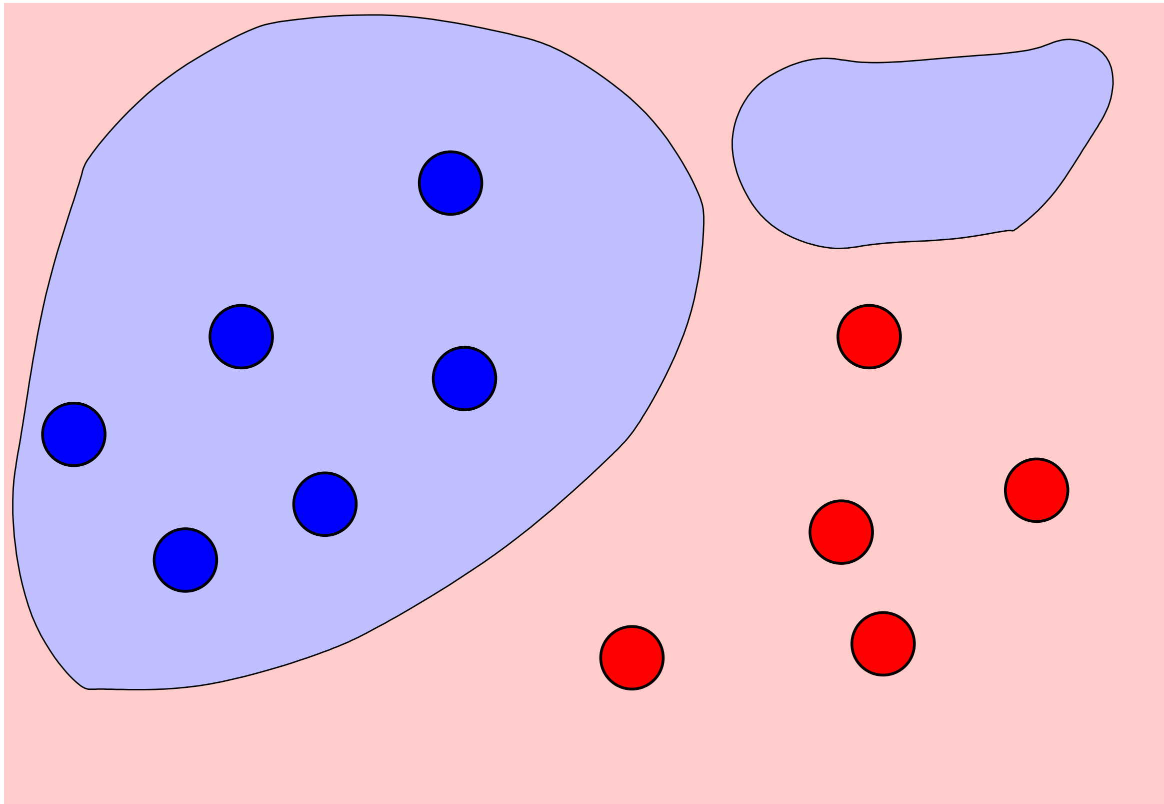
- For illustration purposes **only** we will consider **vectors in the plane**, $d = 2$.
- The ideas for $d \gg 3$ are **conceptually the same**.

Binary Classification Separation Surfaces for Vectors



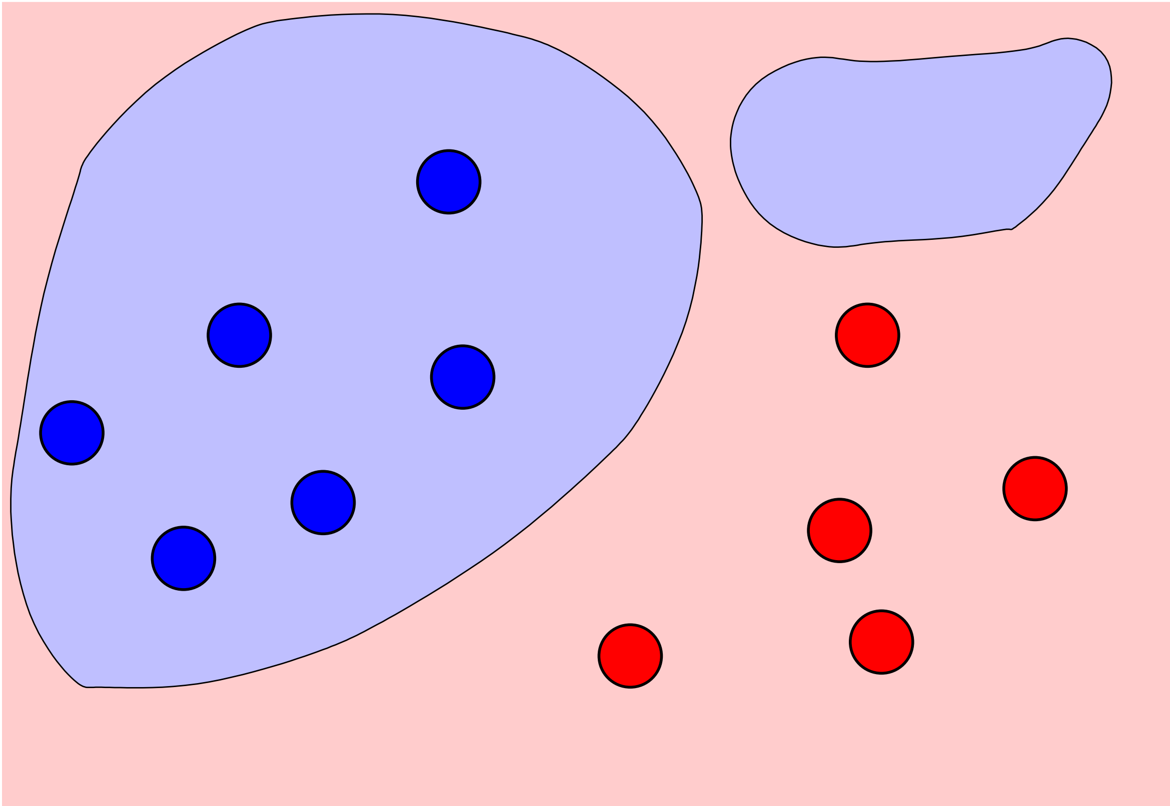
What is a classification rule?

Binary Classification Separation Surfaces for Vectors



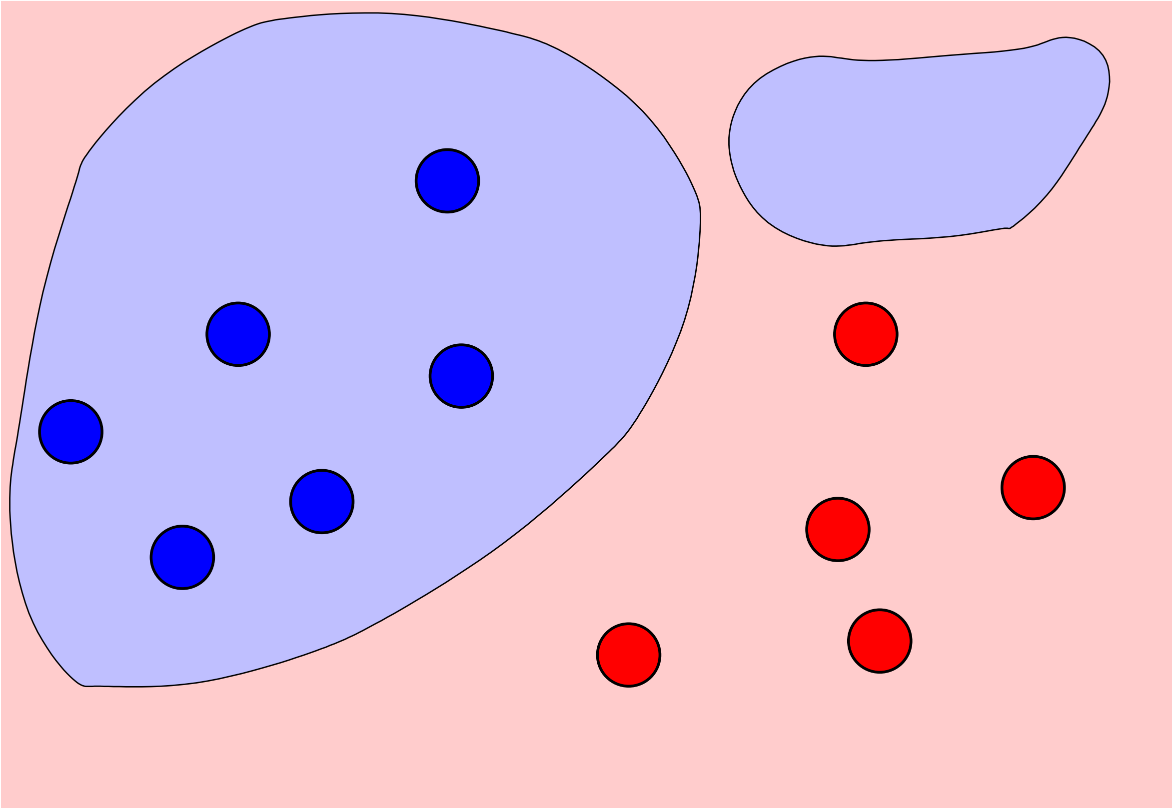
Classification rule = a partition of $\mathbb{R}^d$ into two sets

Binary Classification Separation Surfaces for Vectors



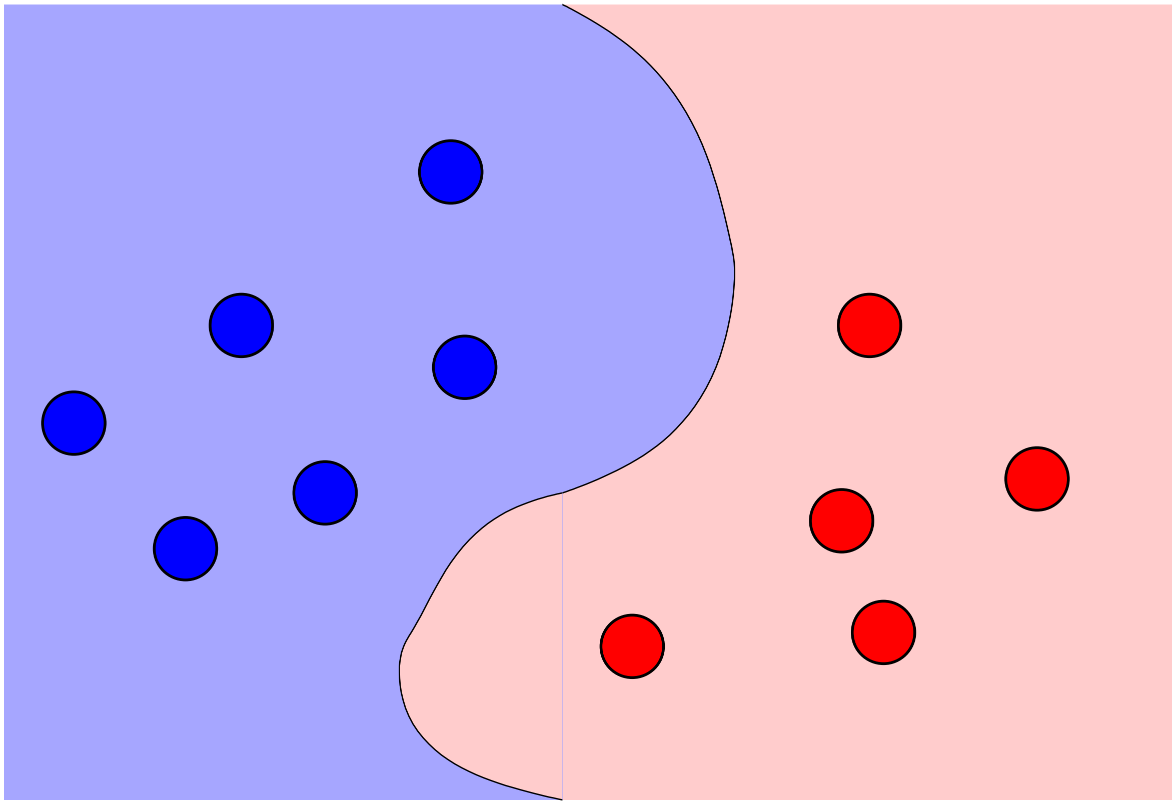
This partition is **encoded** as the **level set** of a function on $\mathbb{R}^d$

Binary Classification Separation Surfaces for Vectors



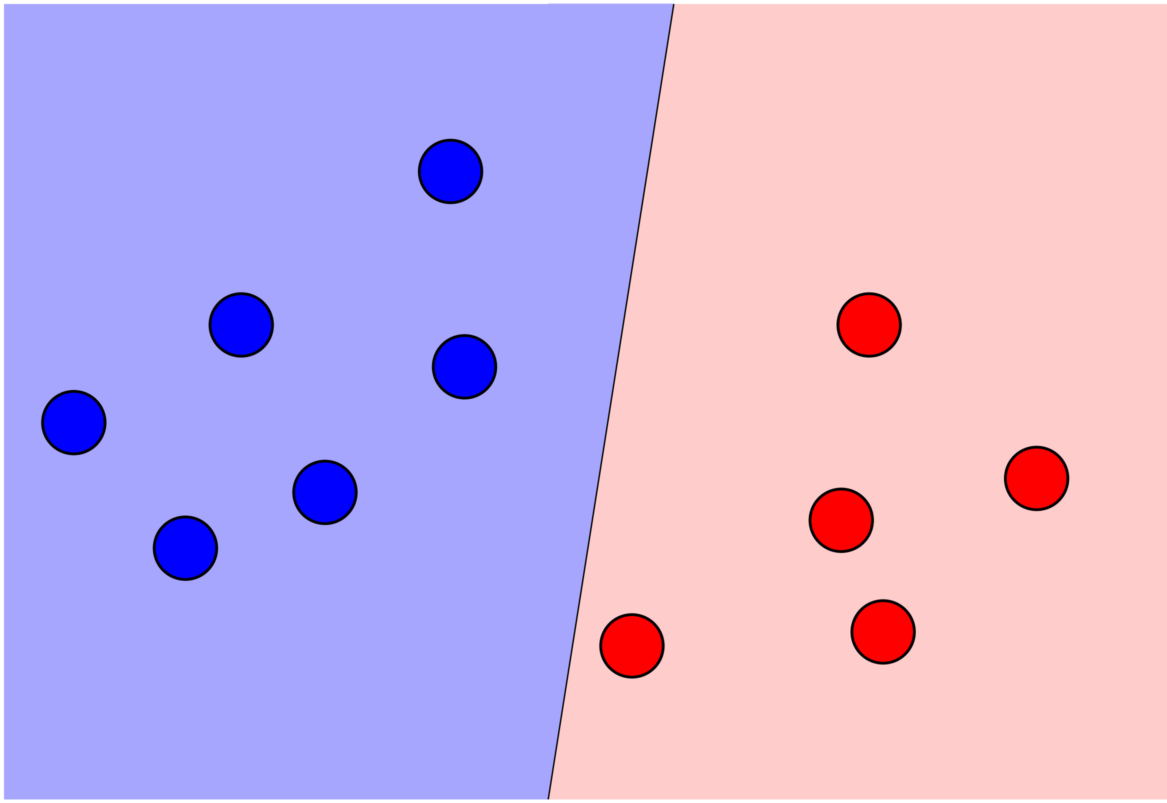
Namely, $\{\mathbf{x} \in \mathbb{R}^d | \boldsymbol{f}(\mathbf{x}) > 0\}$ and $\{\mathbf{x} \in \mathbb{R}^d | \boldsymbol{f}(\mathbf{x}) \leq 0\}$

Classification Separation Surfaces for Vectors



What kind of function? any smooth function works. For instance, a curved line

Classification Separation Surfaces for Vectors



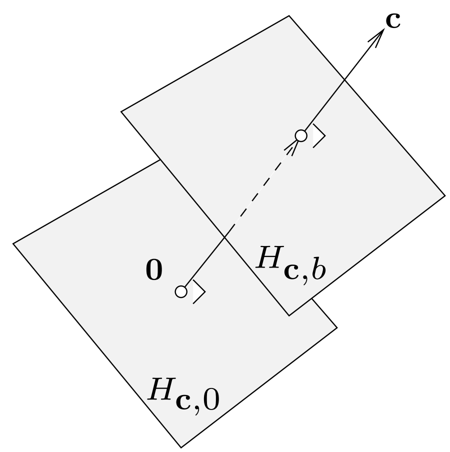
Even more **simple**: using **affine functions** that define halfspaces.

Linear Classifiers

- **lines** (hyperplanes when $d > 2$) provide **the simplest type** of classifiers.
- A hyperplane $H_{\mathbf{c},b}$ is a set in $\mathbb{R}^d$ defined by
  - a normal vector $\mathbf{c} \in \mathbb{R}^d$
  - a constant $b \in \mathbb{R}$. as

$$H_{\mathbf{c},b} = \{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} \;=\; b\}$$

- Letting b vary we can "slide" the hyperplane across $\mathbb{R}^d$

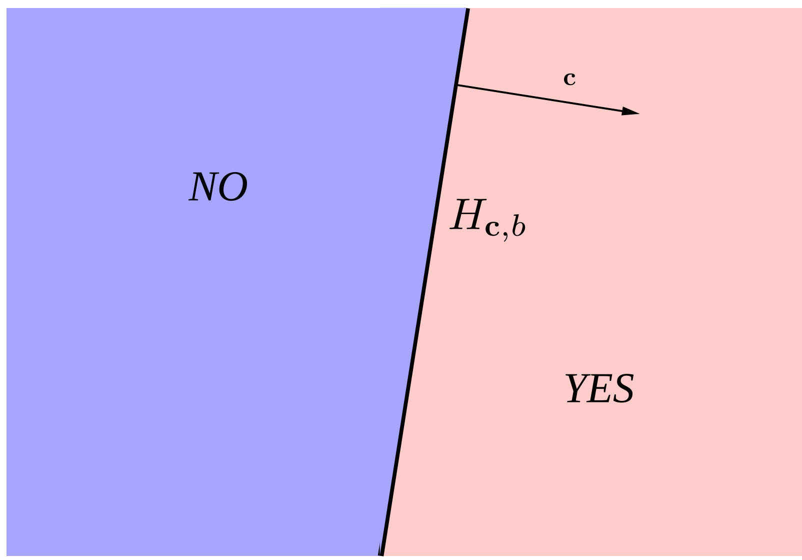


Linear Classifiers

- Exactly like lines in the plane, hyperplanes **divide** $\mathbb{R}^d$ into **two** halfspaces,

$$\{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} < b\} \cup \{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} \geq b\} = \mathbb{R}^d$$

- Linear classifiers answer *"yes"* or *"no"* given $\mathbf{x}$ and pre-defined $\mathbf{c}$ and b.



how can we ***choose*** *a* ***"good"*** $(\mathbf{c}^\star, b^\star)$ *given the* ***training set****?*

Linear Classifiers

- This specific question,

$$\text{“training set”}\ \left\{\left(\mathbf{x}_i \in \mathbb{R}^d,\ \mathbf{y}_i \in \{0,1\}\right)_{i=1..N}\right\} \xRightarrow{????} \text{“best”}\,\mathbf{c}^\star,\ b^\star$$

  has different answers. A (non-exhaustive!) selection of techniques:

- **Linear Discriminant Analysis** (or Fisher's Linear Discriminant);
- **Logistic regression** maximum likelihood estimation;
- **Perceptron**, a one-layer neural network;
- **Support Vector Machine**, the result of a convex program.

Linear Classifiers

- This specific question,

$$\text{“training set”} \;\left\{\left(\mathbf{x}_i \in \mathbb{R}^d,\; \mathbf{y}_i \in \{0,1\}\right)_{i=1..N}\right\} \overset{????}{\Longrightarrow} \text{“best”}\, \mathbf{c}^\star,\, b^\star$$

  has different answers. A (non-exhaustive!) selection of techniques:

- **Linear Discriminant Analysis** (or Fisher's Linear Discriminant);
- **Logistic regression** maximum likelihood estimation;
- **Perceptron**, a one-layer neural network;
- **Support Vector Machine**, the result of a convex program.
- *etc.*

Which one should I use?

Linear Classifiers

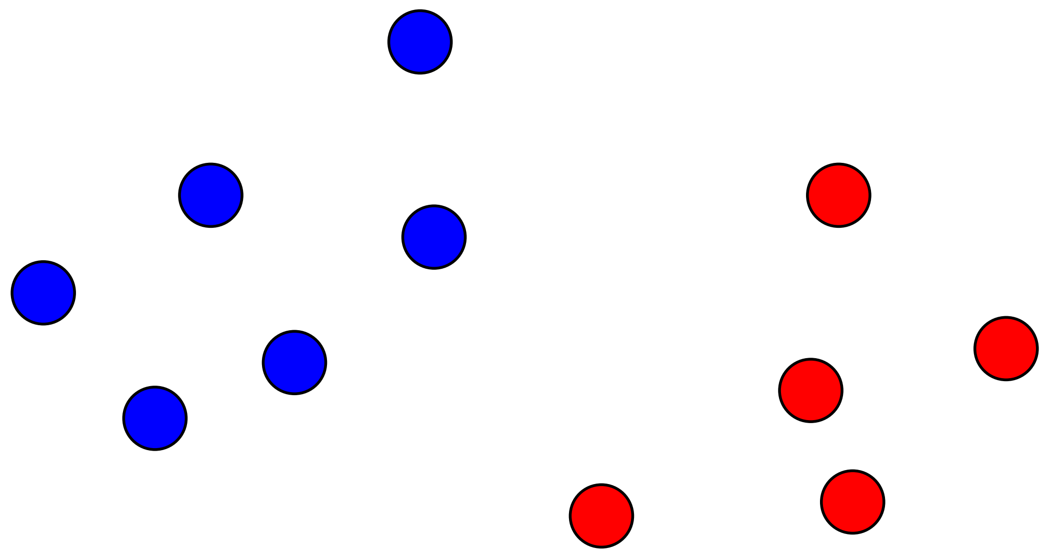
$\boxed{\textbf{Which one should I use}?}$

Too many criteria to answer that question.

Computational speed. Interpretability of the results. Batch or online training. Theoretical guarantees and learning rates. Applicability of assumptions underpinning the technique to your dataset. Source code availability. Parallel?. Size (in bits) of the solution. Reproducibility of results & Numerical stability...*etc.*

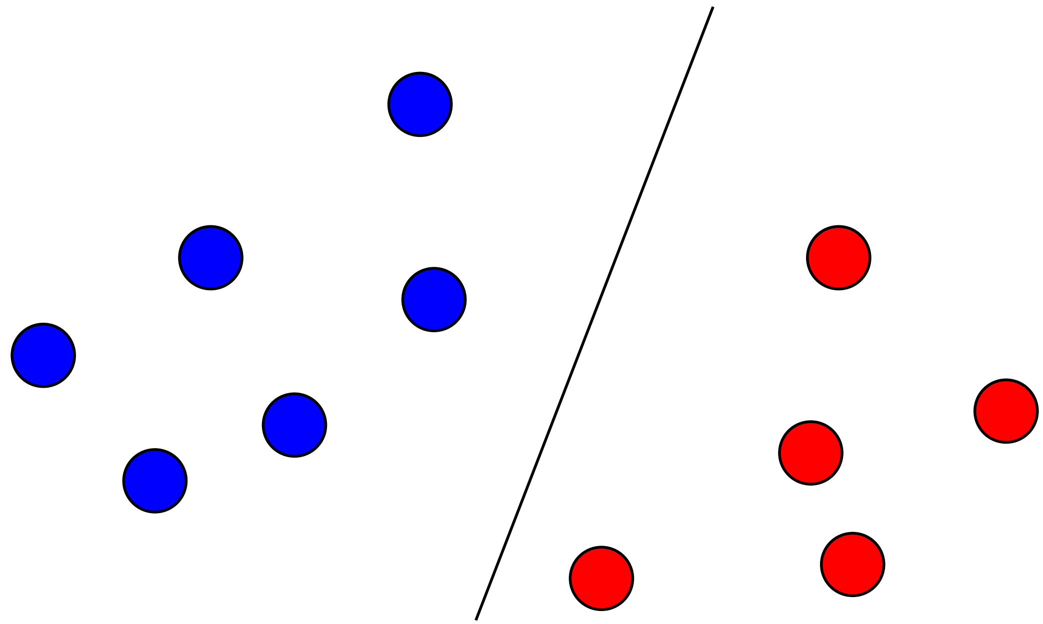
Choosing the **adequate tool** for a **specific dataset** is where your added value as a researcher in pattern recognition lies.

Classification Separation Surfaces for Vectors



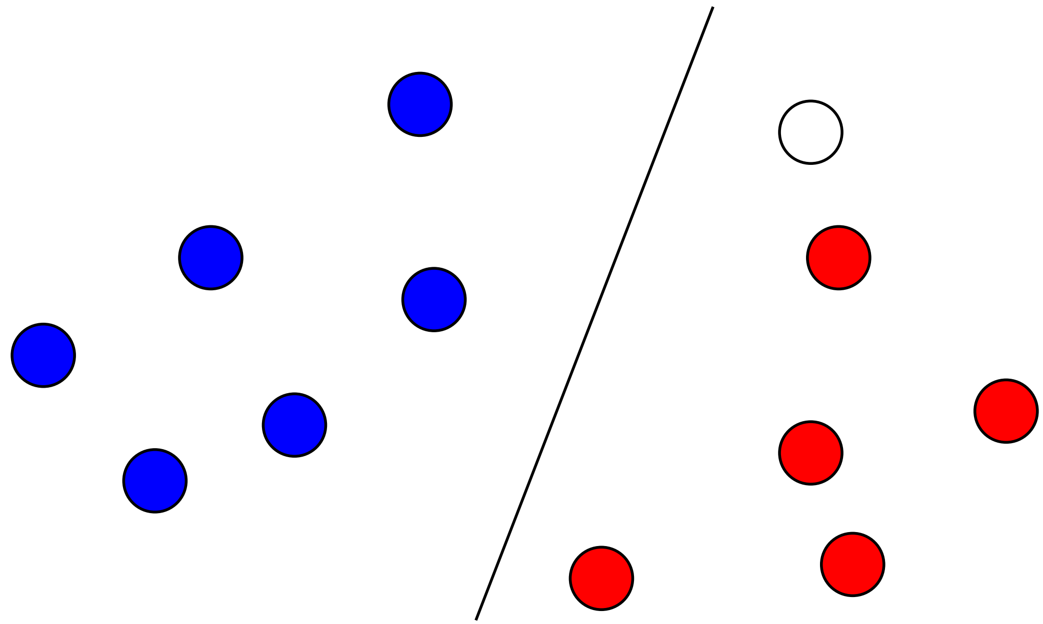
Given two sets of points...

Classification Separation Surfaces for Vectors



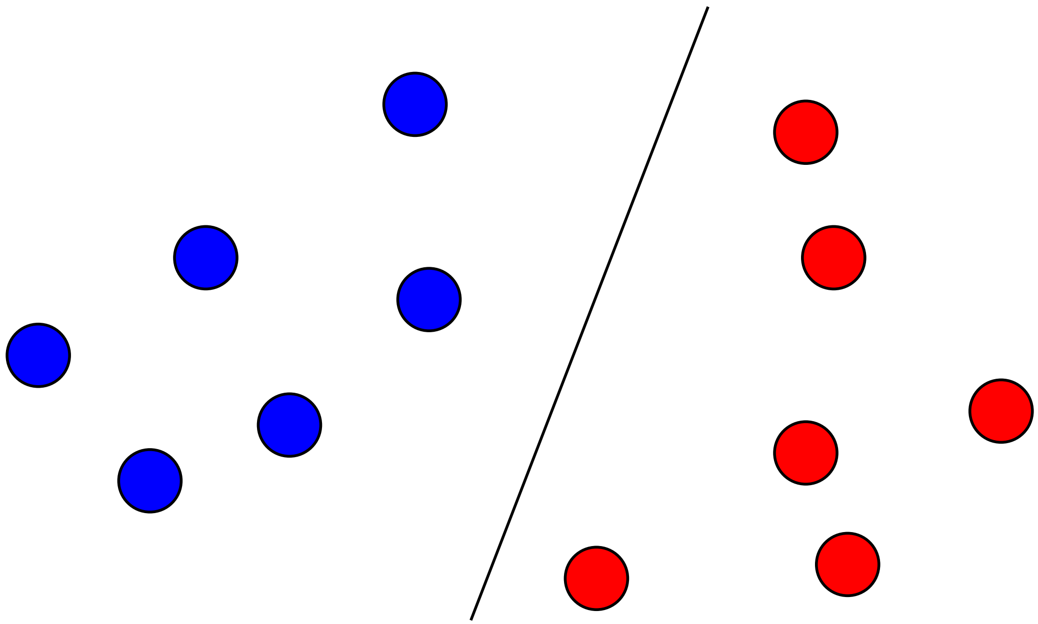
It is sometimes possible to separate them perfectly

Classification Separation Surfaces for Vectors

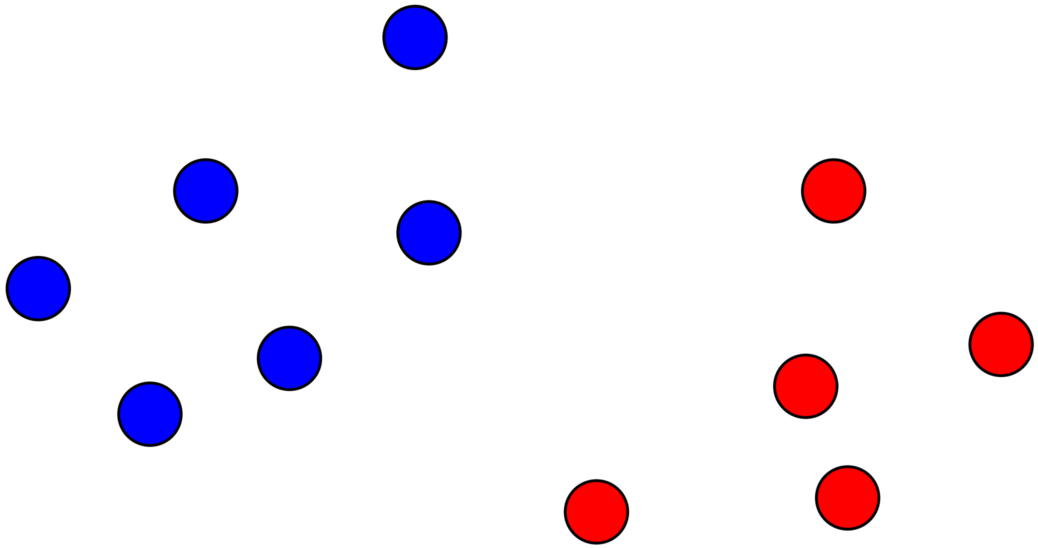


Each choice might look equivalently good on the training set,
but it will have obvious impact on new points

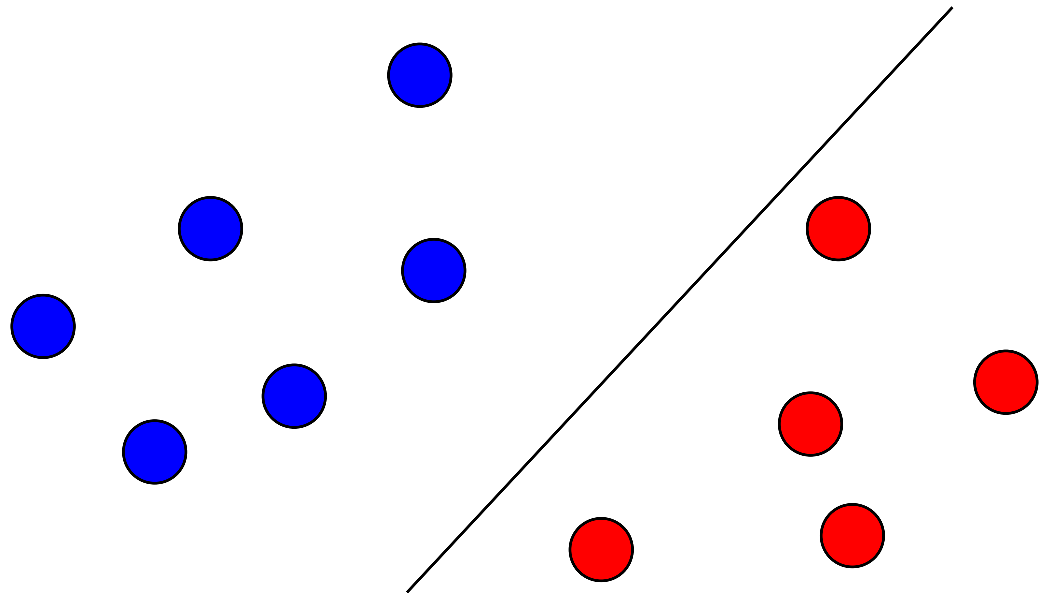
Classification Separation Surfaces for Vectors



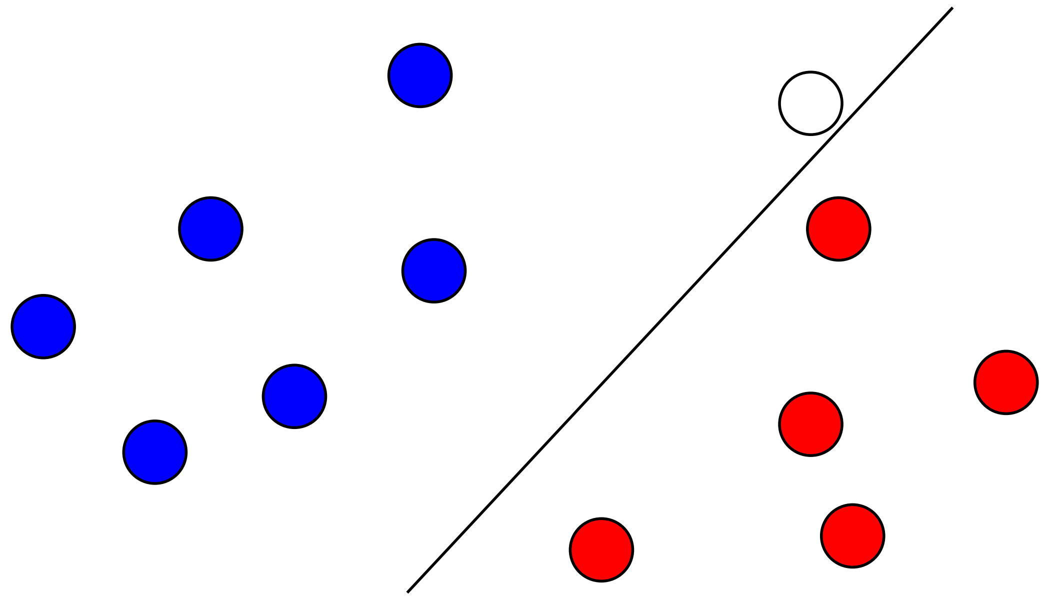
Linear classifier, some degrees of freedom



Linear classifier, some degrees of freedom

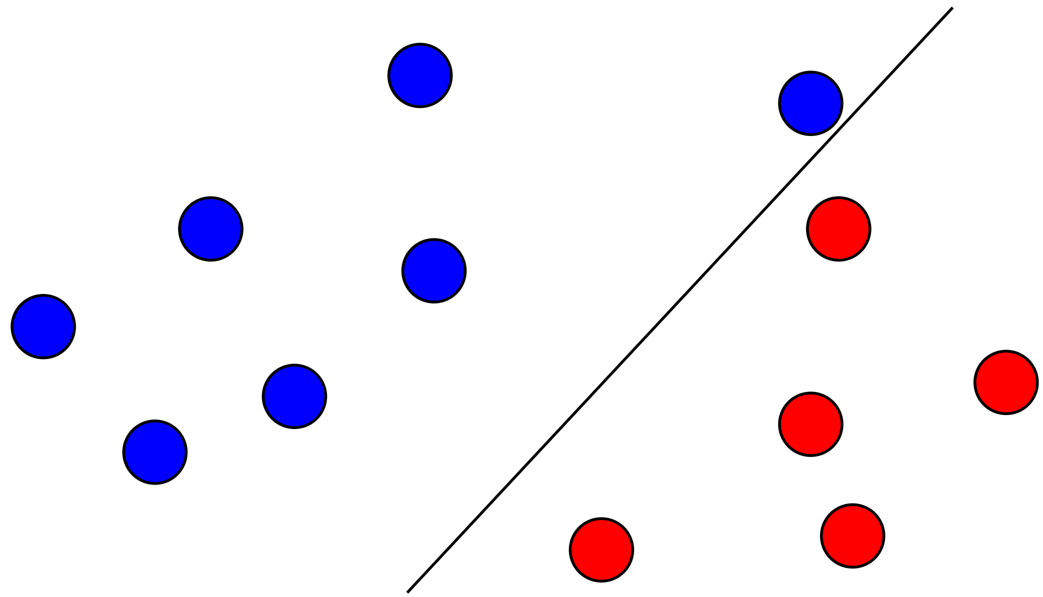


Linear classifier, some degrees of freedom

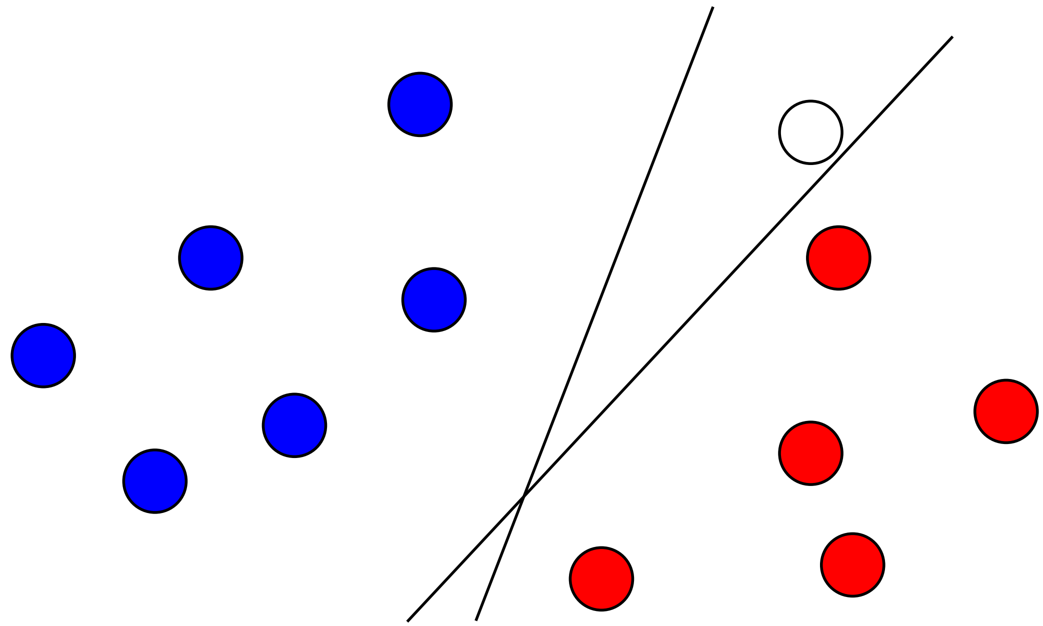


Specially close to the border of the classifier

Linear classifier, some degrees of freedom



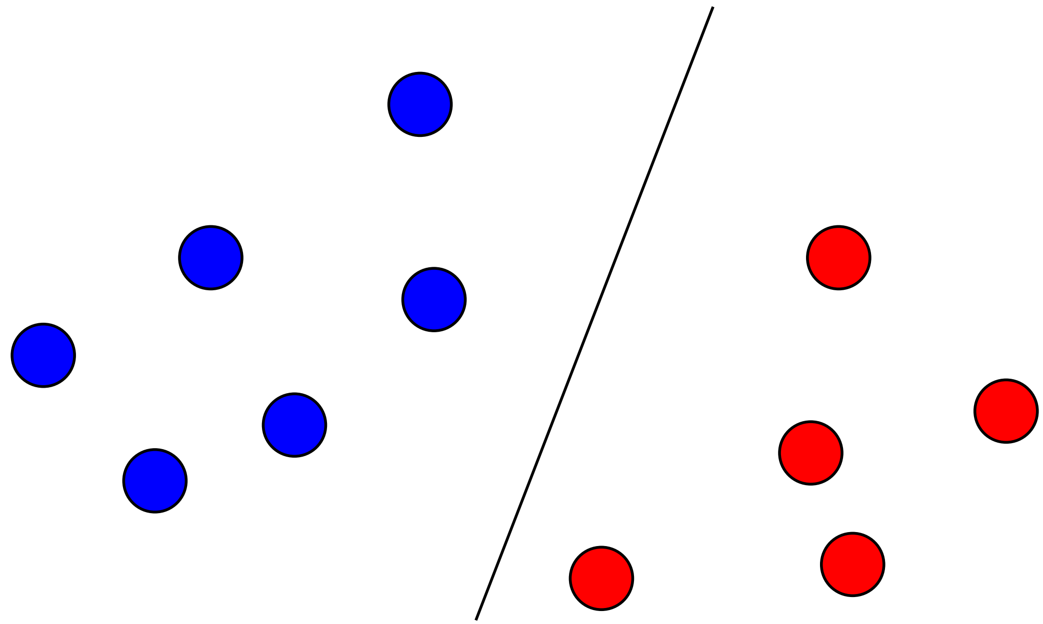
Linear classifier, some degrees of freedom



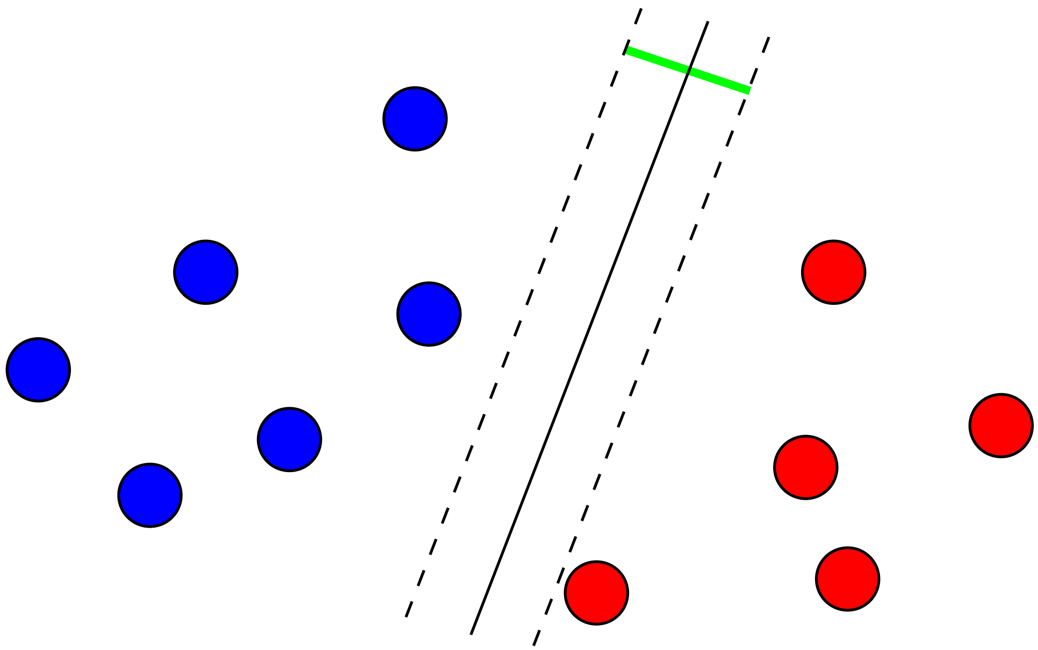
For each different technique, different results, different performance.

Support Vector Machines
The linearly-separable case

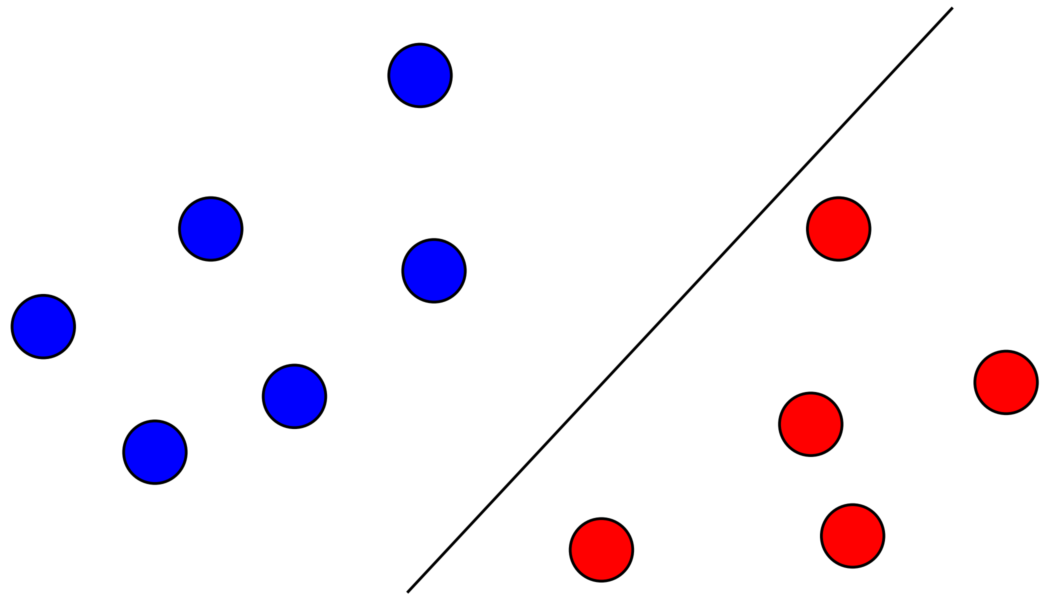
A criterion to select a linear classifier: the margin ?



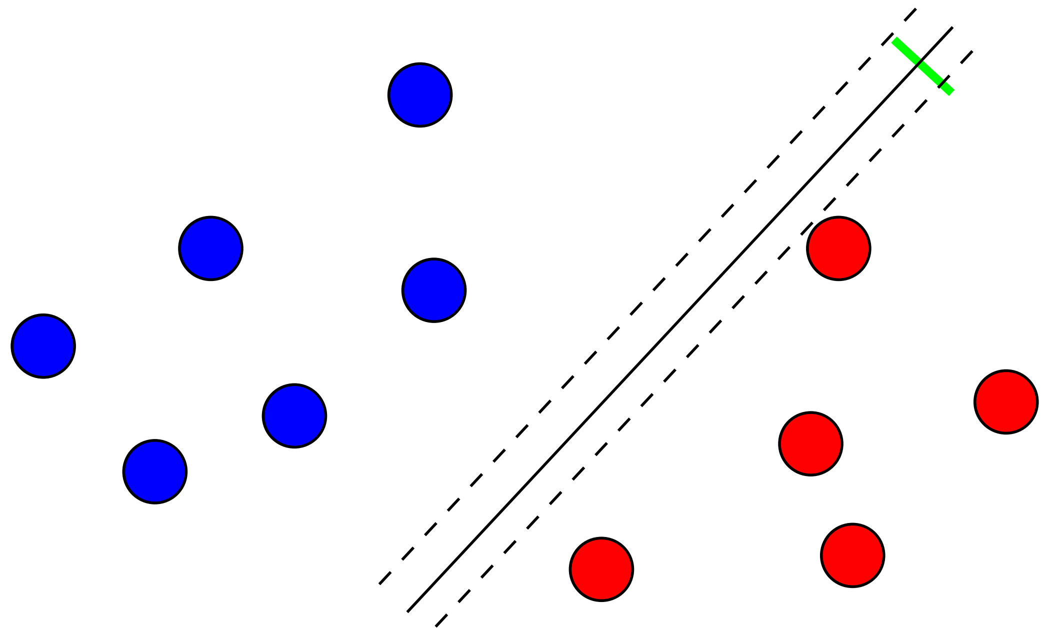
A criterion to select a linear classifier: the margin ?



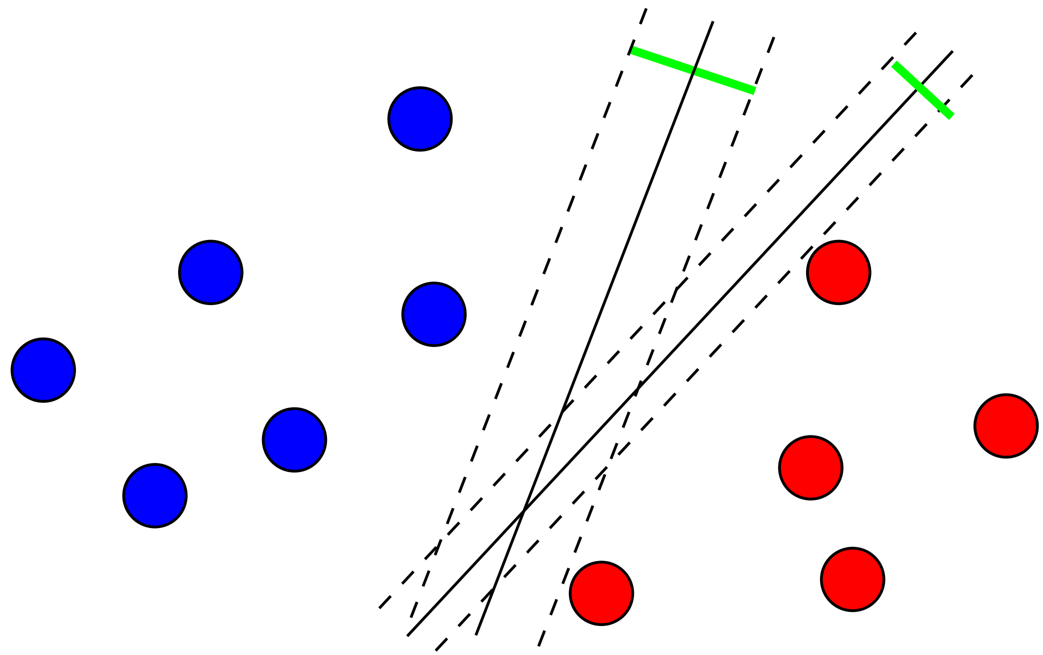
A criterion to select a linear classifier: the margin ?



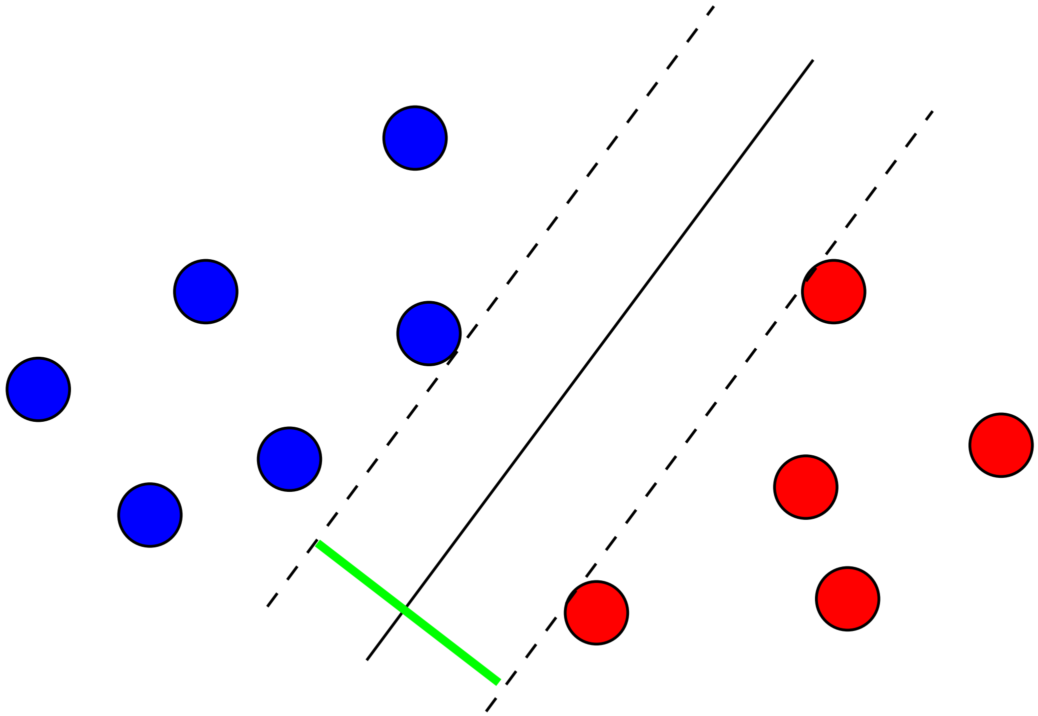
A criterion to select a linear classifier: the margin ?



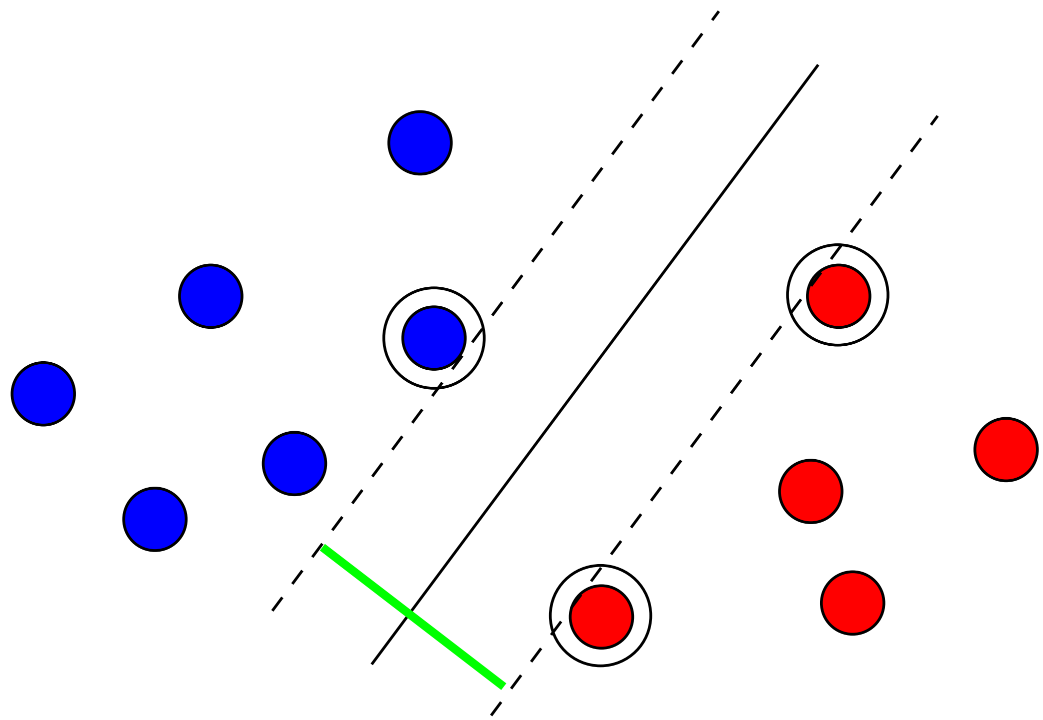
A criterion to select a linear classifier: the margin ?



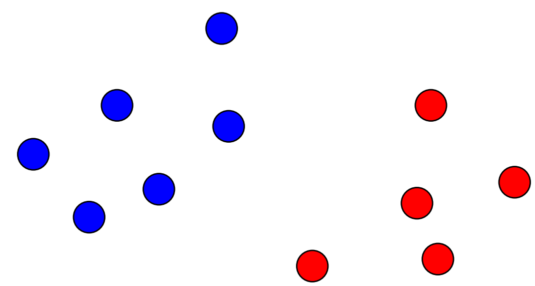
Largest Margin Linear Classifier ?



Support Vectors with Large Margin



In Mathematical Equations



- The **training set** is a finite set of n data/class pairs:

$$\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\},$$

where $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in \{-1, 1\}$.

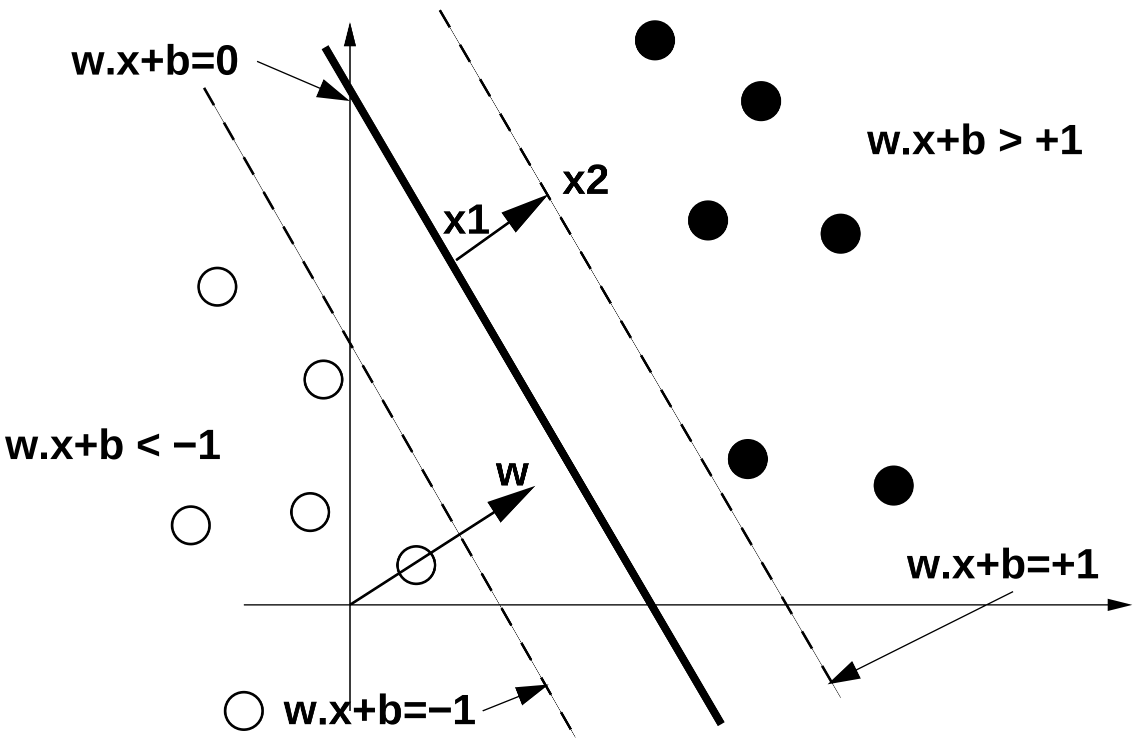
- We assume (for the moment) that the data are **linearly separable**, i.e., that there exists $(\mathbf{w}, b) \in \mathbb{R}^d \times \mathbb{R}$ such that:

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b > 0 & \text{if } \mathbf{y}_i = 1\,, \\ \mathbf{w}^T\mathbf{x}_i + b < 0 & \text{if } \mathbf{y}_i = -1\,. \end{cases}$$

How to find the largest separating hyperplane?

For the linear classifier $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ consider the *interstice* defined by the hyperplanes

- $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = +1$
- $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = -1$



The margin is $2/||\mathbf{w}||$

- Indeed, the points $\mathbf{x}_1$ and $\mathbf{x}_2$ satisfy:

$$\begin{cases} \mathbf{w}^T \mathbf{x}_1 + b = 0, \\ \mathbf{w}^T \mathbf{x}_2 + b = 1. \end{cases}$$

- By subtracting we get $\mathbf{w}^T(\mathbf{x}_2 - \mathbf{x}_1) = 1$, and therefore:

$$\gamma = 2||\mathbf{x}_2 - \mathbf{x}_1|| = \frac{2}{||\mathbf{w}||}.$$

where γ is the margin.

$$\boxed{\text{Large margin } \gamma \Leftrightarrow \text{Small } ||\mathbf{w}|| \Leftrightarrow \text{Small } ||\mathbf{w}||^2}$$

All training points should be on the good side

- For positive examples ($y_i = 1$) this means:

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1$$

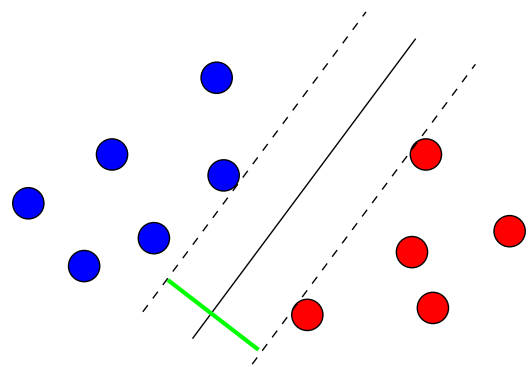
- For negative examples ($y_i = -1$) this means:

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1$$

- in both cases:

$$\forall i = 1, \ldots, n, \qquad \mathbf{y}_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1$$

Finding the optimal hyperplane



- Finding the optimal hyperplane is equivalent to finding $(\mathbf{w}, b)$ which minimize:

$$\|\mathbf{w}\|^2$$

under the constraints:

$$\forall i = 1, \ldots, n, \qquad \mathbf{y}_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1 \geq 0.$$

This is a classical quadratic program on $\mathbb{R}^{d+1}$
linear constraints - **quadratic objective**

Lagrangian

- In order to minimize:

$$\frac{1}{2}||\mathbf{w}||^2$$

under the constraints:

$$\forall i = 1, \ldots, n, \qquad y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1 \geq 0.$$

- introduce **one dual variable α_i for each constraint**,
- one constraint for **each training point**.
- the **Lagrangian** is, for $\alpha \succeq 0$ (that is for each $\alpha_i \geq 0$)

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1\right).$$

The Lagrange dual function

$$g(\alpha) = \inf_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right) \right\}$$

the saddle point conditions give us that at the minimum in $\mathbf{w}$ and b

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{y}_i \mathbf{x}_i, \quad (\text{ derivating w.r.t } \mathbf{w}) \quad (*)$$

$$0 = \sum_{i=1}^{n} \alpha_i \mathbf{y}_i, \quad (\text{derivating w.r.t } b) \qquad (**)$$

substituting $(*)$ in g, and using $(**)$ as a constraint, get the dual function $g(\alpha)$.

- To solve the dual problem, **maximize** g w.r.t. α.

Dual optimum

The dual problem is thus

$$\begin{array}{ll} \text{maximize} & g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{such that} & \alpha \succeq 0, \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0. \end{array}$$

This is a **quadratic program** in $\mathbb{R}^n$, with *box constraints*.
$\alpha^\star$ can be computed using optimization software
($e.g.$ built-in `matlab` function)

Dual optimum

The dual problem is thus

$$\begin{array}{ll} \text{maximize} & g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{such that} & \alpha \succeq 0, \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0. \end{array}$$

This is a **quadratic program** in $\mathbb{R}^n$, with *box constraints*.
$\alpha^\star$ can be computed using optimization software
($e.g.$ built-in `matlab` function)

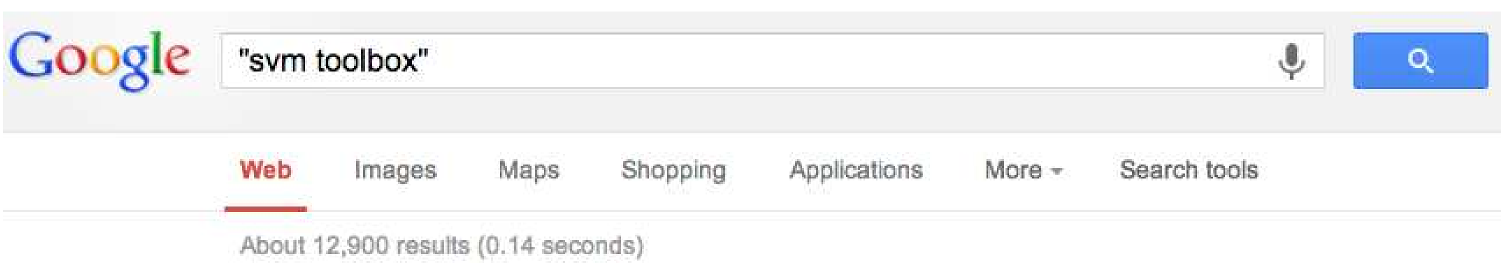
All SVM toolboxes available can be interpreted as *customized solvers* that handle *efficiently* this particular problem.

Dual optimum

The dual problem is thus

$$\begin{array}{ll} \text{maximize} & g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{such that} & \alpha \succeq 0, \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0. \end{array}$$

This is a **quadratic program** in $\mathbb{R}^n$, with *box constraints*.
$\alpha^\star$ can be computed using optimization software
($e.g.$ built-in `matlab` function)



Recovering the optimal hyperplane

- With $\alpha^\star$, we recover $(\mathbf{w}_\star^T, b_\star)$ corresponding to the **optimal hyperplane**.
- $\mathbf{w}_\star^T$ is given by $\mathbf{w}_\star^T = \sum_{i=1}^n y_i \alpha_i^\star \mathbf{x}_i^T$,
- $b_\star$ is given by the conditions on the support vectors $\alpha_i > 0$, $\mathbf{y}_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$,

$$b_\star = -\frac{1}{2}\left(\min_{\mathbf{y}_i=1,\alpha_i>0}(\mathbf{w}_\star^T \mathbf{x}_i) + \max_{\mathbf{y}_i=-1,\alpha_i>0}(\mathbf{w}_\star^T \mathbf{x}_i)\right)$$

- the **decision function** is therefore:

$$\begin{aligned} f^\star(\mathbf{x}) &= \mathbf{w}_\star^T \mathbf{x} + b_\star \\ &= \left(\sum_{i=1}^n y_i \alpha_i^\star \mathbf{x}_i^T\right) \mathbf{x} + b_\star. \end{aligned}$$

- Here the **dual** solution gives us directly the **primal** solution.

From optimization theory...

Studying the relationship between primal/dual problems is an important topic in optimization theory.

As a consequence, we can say many things about the optimal $\alpha^\star$ and the constraints of the original problem.

- **Strong duality** : primal and dual problems have the **same optimum**.
- **Karush-Kuhn-Tucker Conditions** give us that for every $i \leq n$

$$\alpha_i^\star(\mathbf{y}_i \left(\mathbf{w}_\star^T \mathbf{x}_i + b_\star\right) - 1) = 0.$$

From optimization theory...

Studying the relationship between primal/dual problems is an important topic in optimization theory.

As a consequence, we can say many things about the optimal $\alpha^\star$ and the constraints of the original problem.

- **Strong duality** : primal and dual problems have the **same optimum**.
- **Karush-Kuhn-Tucker Conditions** give us that for every $i \leq n$

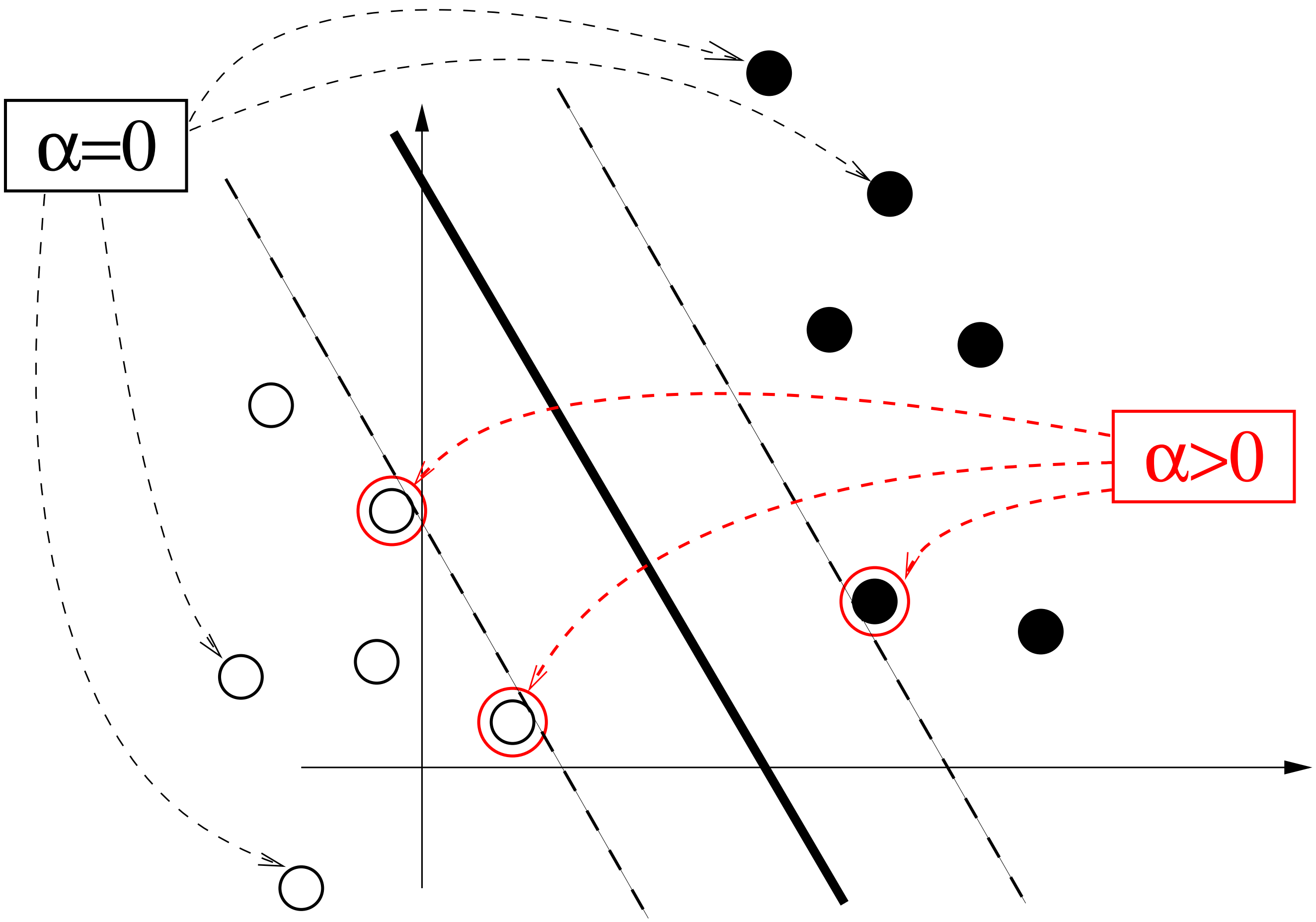
$$\underbrace{\alpha_i^\star}_{\text{multiplier } i} \times \underbrace{\left(\mathbf{y}_i \left(\mathbf{w}_\star^T \mathbf{x}_i + b_\star\right) - 1\right)}_{\text{constraint } i} = 0.$$

- Hence, either

$$\boldsymbol{\alpha_i^\star = 0} \text{ OR } \boldsymbol{\mathbf{y}_i \left(\mathbf{w}_\star^T \mathbf{x}_i + b_\star\right) = 1}.$$

- $\alpha_i^\star \neq 0$ **only** for points that lie **on the tube** (support vectors!).

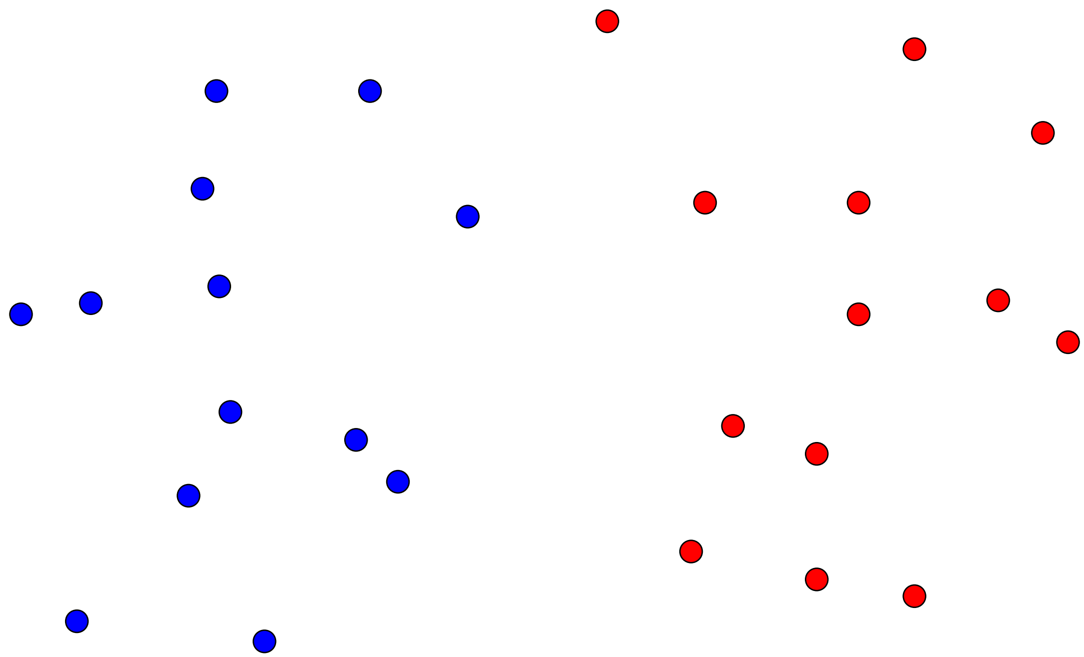
Visualizing Support Vectors



$$\mathbf{w}_\star = \sum_{i=1}^{n} \alpha_i^\star \mathbf{y}_i \mathbf{x}_i$$

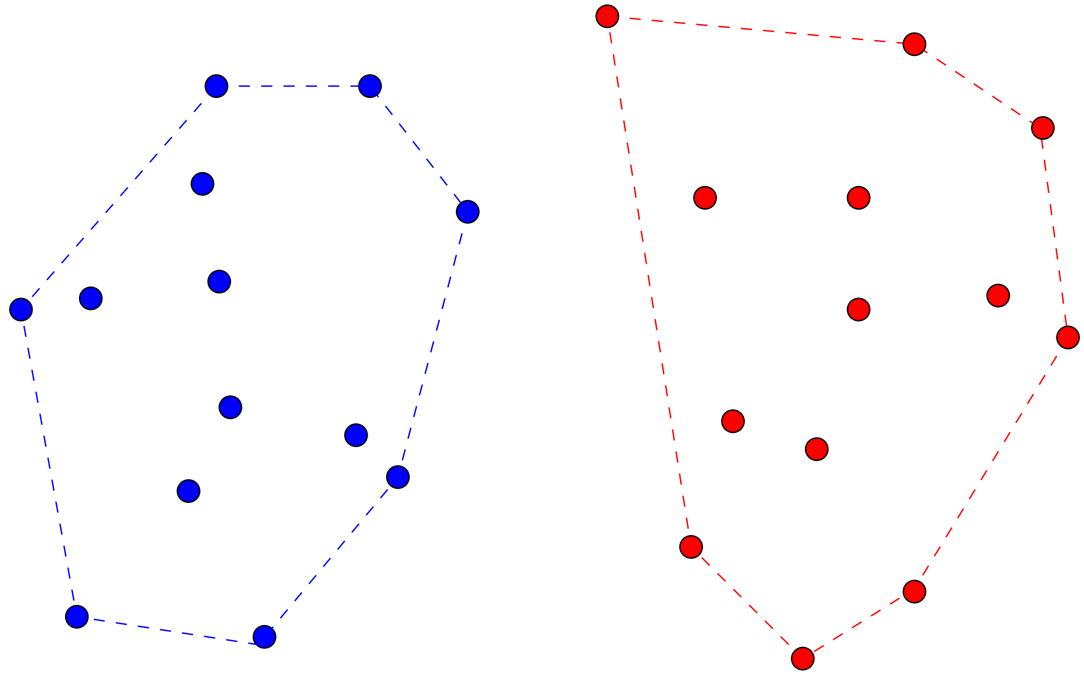
$\alpha_i^\star \neq 0$ **only** for points that lie **on the tube** (support vectors!).

Another interpretation: Convex Hulls



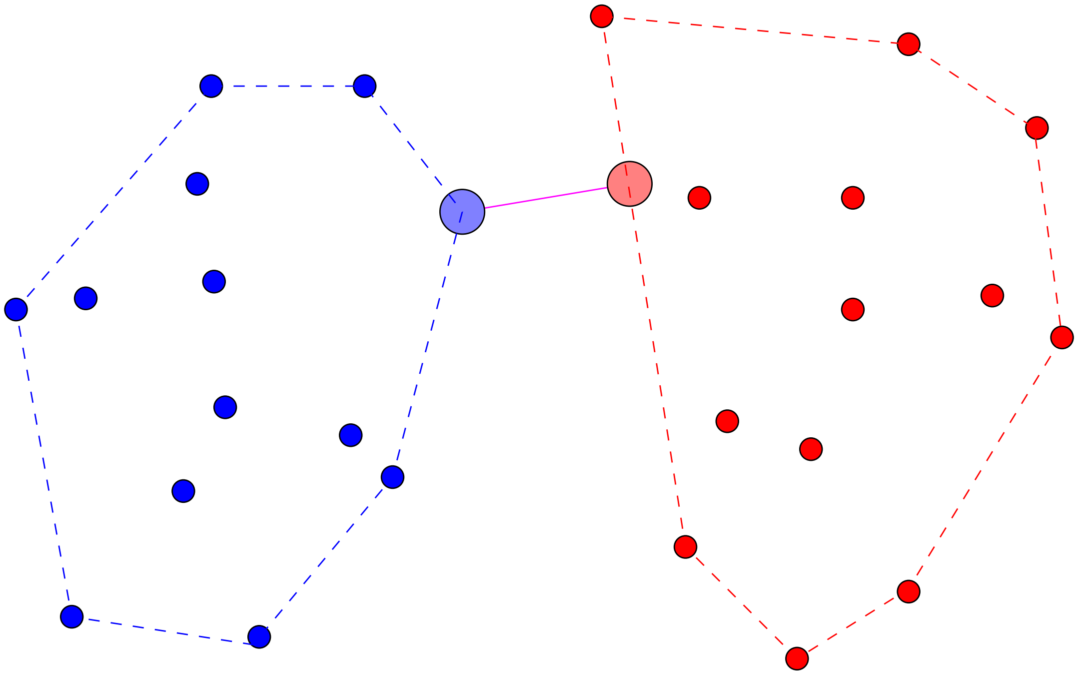
go back to 2 sets of points that are linearly separable

Another interpretation: Convex Hulls



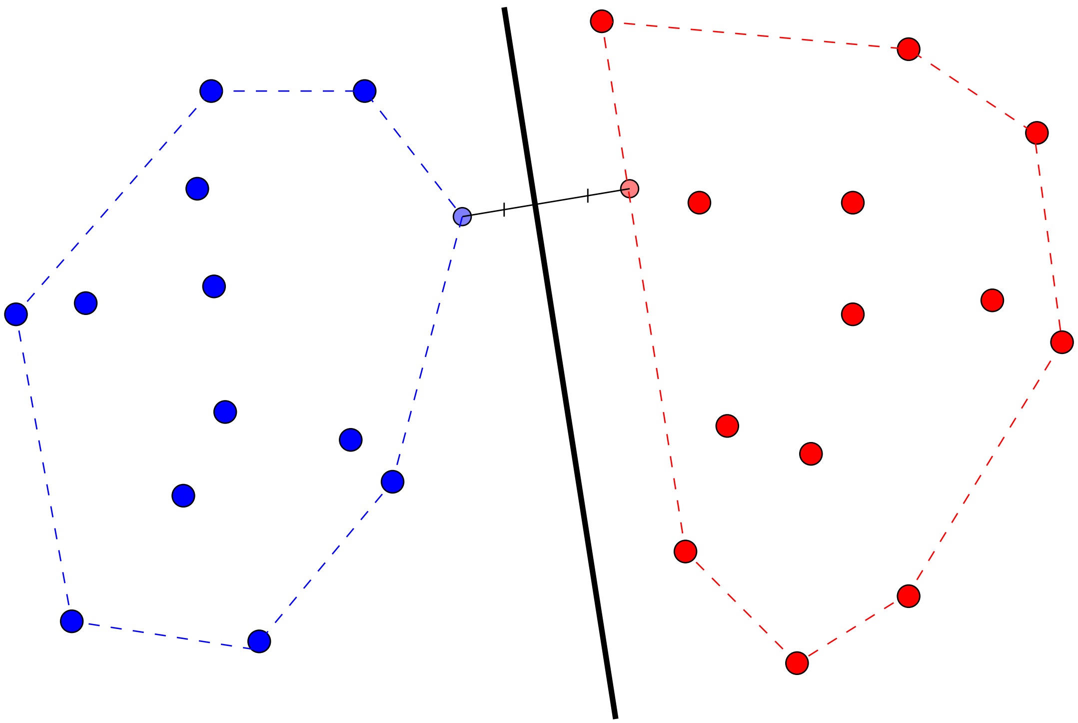
$\boxed{\text{Linearly separable} = \text{convex hulls do not intersect}}$

Another interpretation: Convex Hulls



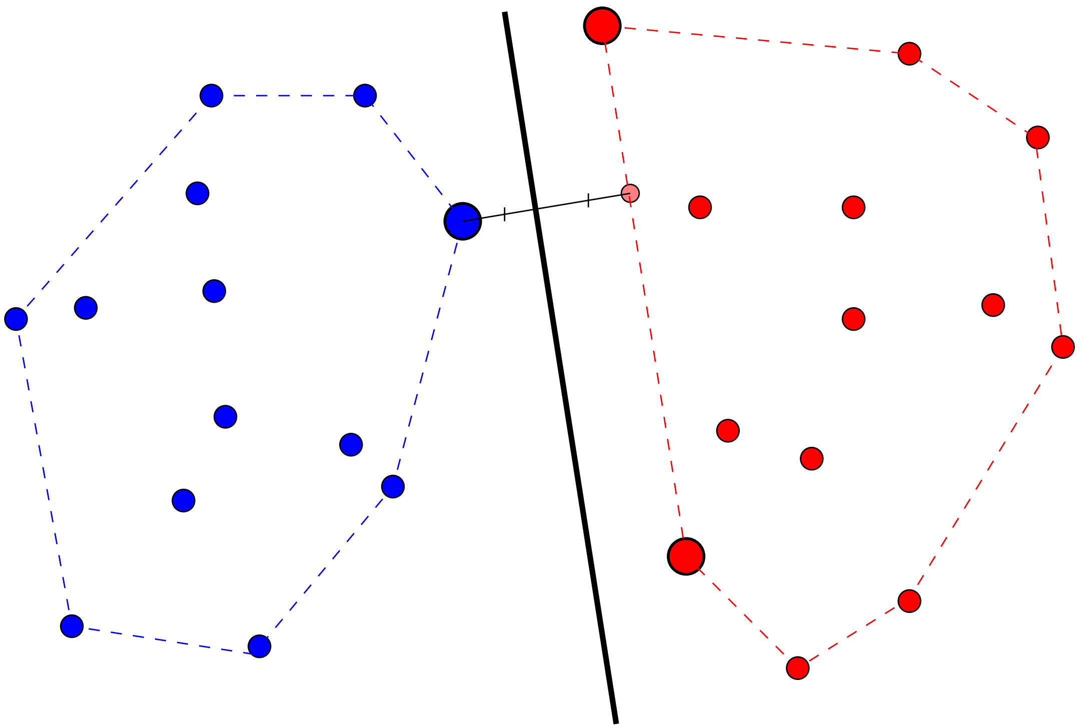
Find two closest points, one in each convex hull

Another interpretation: Convex Hulls



$\boxed{\text{The SVM = bisection of that segment}}$

Another interpretation: Convex Hulls

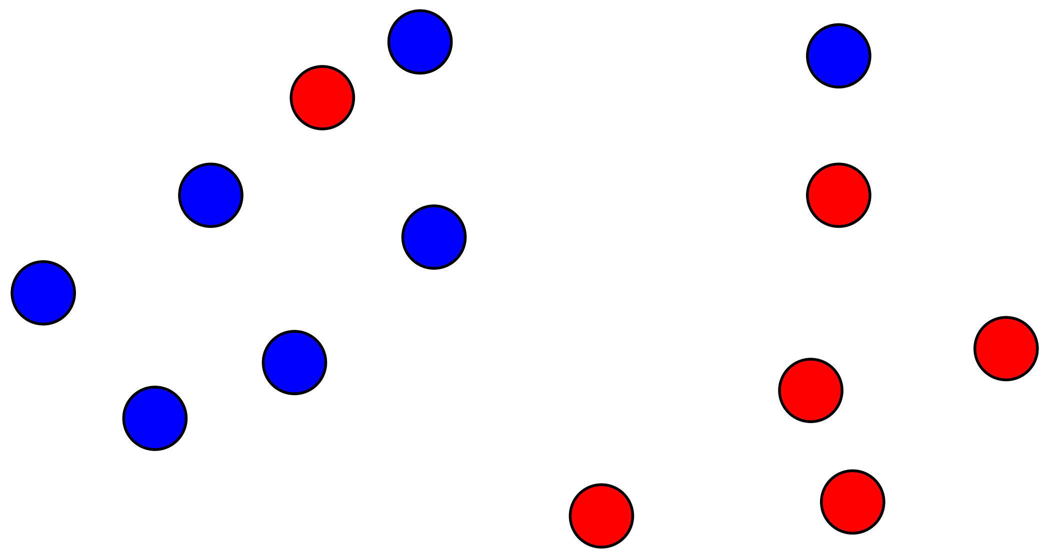


support vectors = extreme points of the faces on which the two points lie

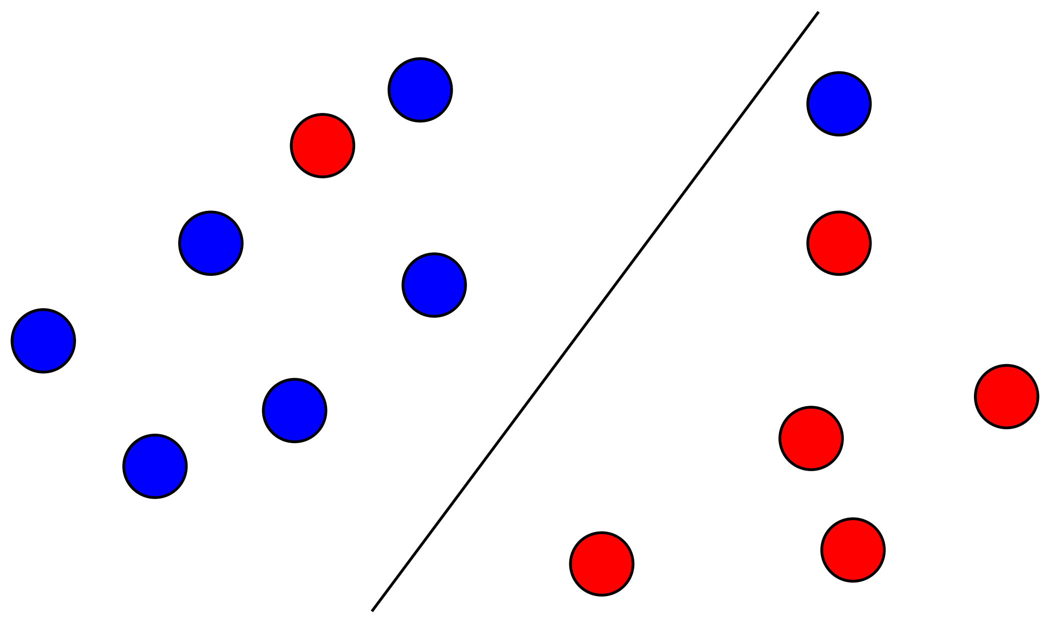
The non-linearly separable case

(when convex hulls intersect)

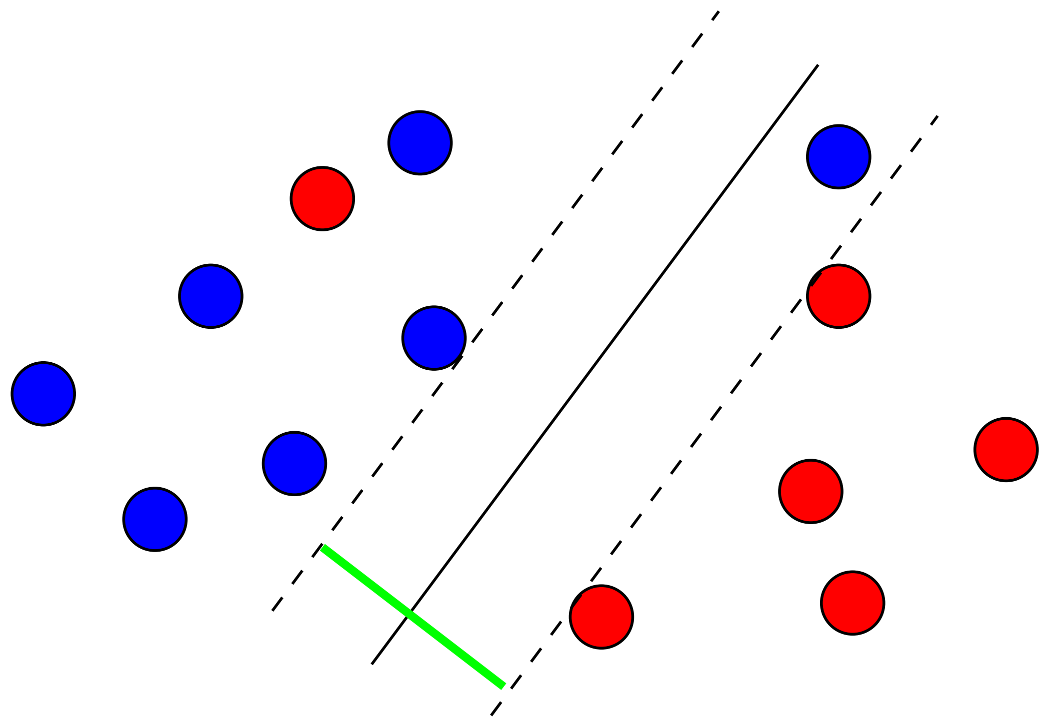
What happens when the data is not linearly separable?



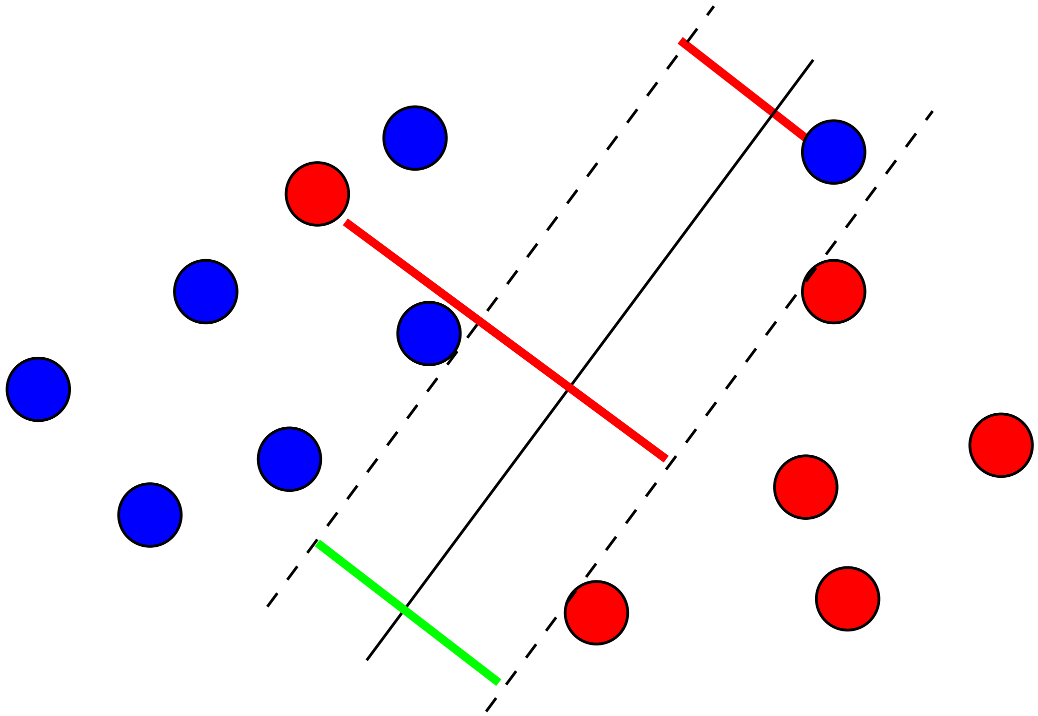
What happens when the data is not linearly separable?



What happens when the data is not linearly separable?



What happens when the data is not linearly separable?

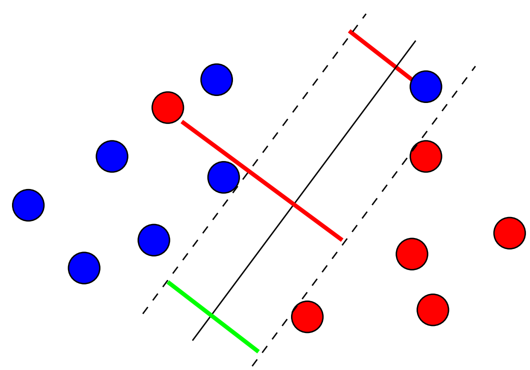


Soft-margin SVM ?

- Find a trade-off between **large margin** and **few errors**.
- Mathematically:

$$\min_f \left\{ \frac{1}{\text{margin}(f)} + C \times \text{errors}(f) \right\}$$

- C is a parameter



Soft-margin SVM formulation ?

- The **margin** of a labeled point $(\mathbf{x}, \mathbf{y})$ is

$$\text{margin}(\mathbf{x}, \mathbf{y}) = \mathbf{y}\left(\mathbf{w}^T\mathbf{x} + b\right)$$

- The **error** is
  - 0 if $\text{margin}(\mathbf{x}, \mathbf{y}) > 1$,
  - $1 - \text{margin}(\mathbf{x}, \mathbf{y})$ otherwise.

- The soft margin SVM solves:

$$\min_{\mathbf{w},b}\{\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\{0, 1 - \mathbf{y}_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\}$$

- $c(u, y) = \max\{0, 1 - yu\}$ is known as the **hinge loss**.

- $c(\mathbf{w}^T\mathbf{x}_i + b, \mathbf{y}_i)$ associates a mistake cost to the decision $\mathbf{w}, b$ for example $\mathbf{x}_i$.

Dual formulation of soft-margin SVM

- The soft margin SVM program

$$\min_{\mathbf{w},b}\{\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\{0, 1-\mathbf{y}_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\}$$

can be rewritten as

$$\begin{array}{rc} \text{minimize} & \|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{such that} & \mathbf{y}_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1-\xi_i \end{array}$$

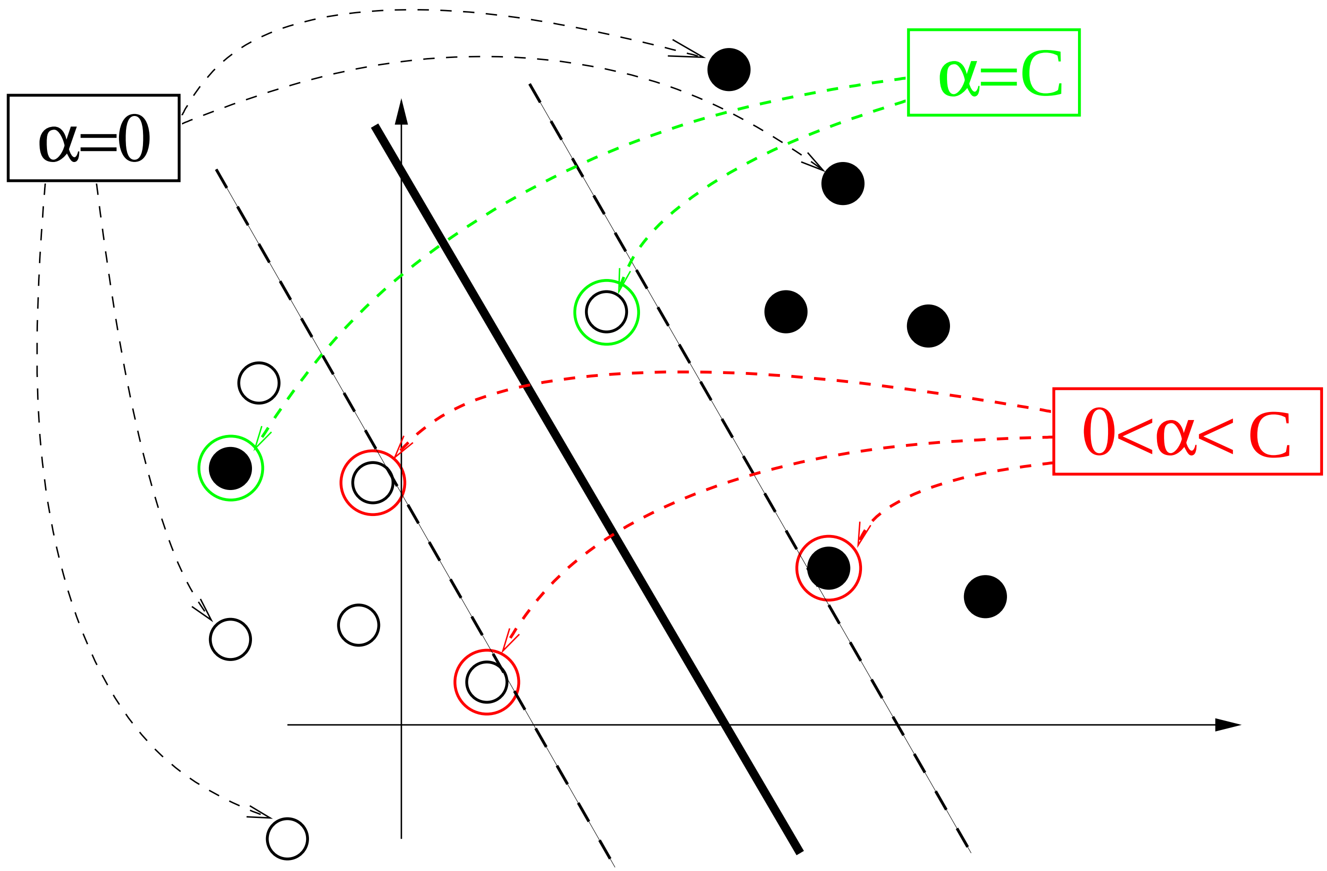
- In that case the dual function

$$g(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j\mathbf{y}_i\mathbf{y}_j\mathbf{x}_i^T\mathbf{x}_j,$$

which is finite under the constraints:

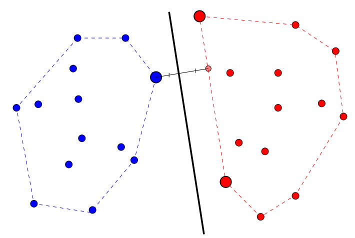
$$\begin{cases} 0 \leq \alpha_i \leq C, & \text{for } i = 1,\dots,n \\ \sum_{i=1}^{n}\alpha_i\mathbf{y}_i = 0. \end{cases}$$

Interpretation: bounded and unbounded support vectors

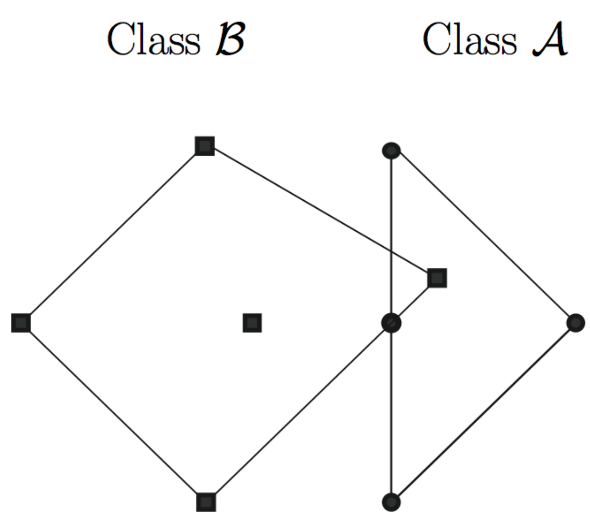


What about the convex hull analogy?

- Remember the separable case



- Here we consider the case where the two sets are not linearly separable, $i.e.$ their convex hulls **intersect.**



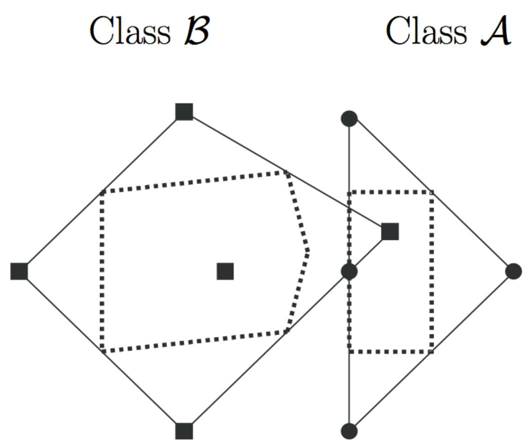
What about the convex hull analogy?

Definition 1. *Given a set of n points $\mathcal{A}$, and $0 \leq C \leq 1$, the set of finite combinations*

$$\sum_{i=1}^{n} \lambda_i \mathbf{x}_i, 1 \leq \lambda_i \leq C, \sum_{i=1}^{n} \lambda_i = 1,$$

is the (C) reduced convex hull of $\mathcal{A}$

- Using $C = 1/2$, the reduced convex hulls of $\mathcal{A}$ and $\mathcal{B}$,



- Soft-SVM with C = closest two points of C-reduced convex hulls.

Kernels

Kernel trick for SVM's

- use a mapping ϕ from $\mathcal{X}$ to a feature space,
- which corresponds to the **kernel** k:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \quad k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

- Example: if $\phi(\mathbf{x}) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$, then

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.$$

Training a SVM in the feature space

Replace each $\mathbf{x}^T\mathbf{x}'$ in the SVM algorithm by $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$

- **Reminder**: the dual problem is to maximize

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, \boldsymbol{k(\mathbf{x}_i, \mathbf{x}_j)},$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0. \end{cases}$$

- The **decision function** becomes:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \phi(x) \rangle + b_\star \\ &= \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{k(\mathbf{x}_i, \mathbf{x})} + b_\star. \end{aligned} \tag{1}$$

The Kernel Trick ?

The explicit computation of $\phi(\mathbf{x})$ is not necessary.
The kernel $k(\mathbf{x}, \mathbf{x}')$ is enough.

- the SVM optimization for α works **implicitly** in the feature space.
- the SVM is a kernel algorithm: only need to input $\boldsymbol{K}$ and $\mathbf{y}$:

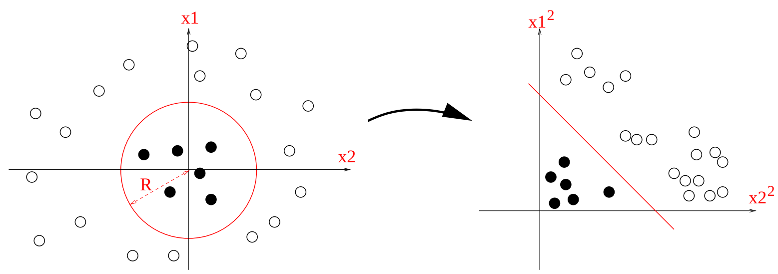
$$
\begin{array}{ll}
\text{maximize} & g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2}\alpha^T(\boldsymbol{K} \odot \mathbf{y}\mathbf{y}^T)\alpha \\
\text{such that} & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \dots, n \\
 & \sum_{i=1}^{n} \alpha_i \mathbf{y}_{\boldsymbol{i}} = 0.
\end{array}
$$

- $\boldsymbol{K}$'s **positive definite** $\Leftrightarrow$ **problem has an unique optimum**
- the decision function is $f(\cdot) = \sum_{i=1}^{n} \alpha_i \, \boldsymbol{k}(\mathbf{x}_i, \cdot) + b$.

Kernel example: polynomial kernel

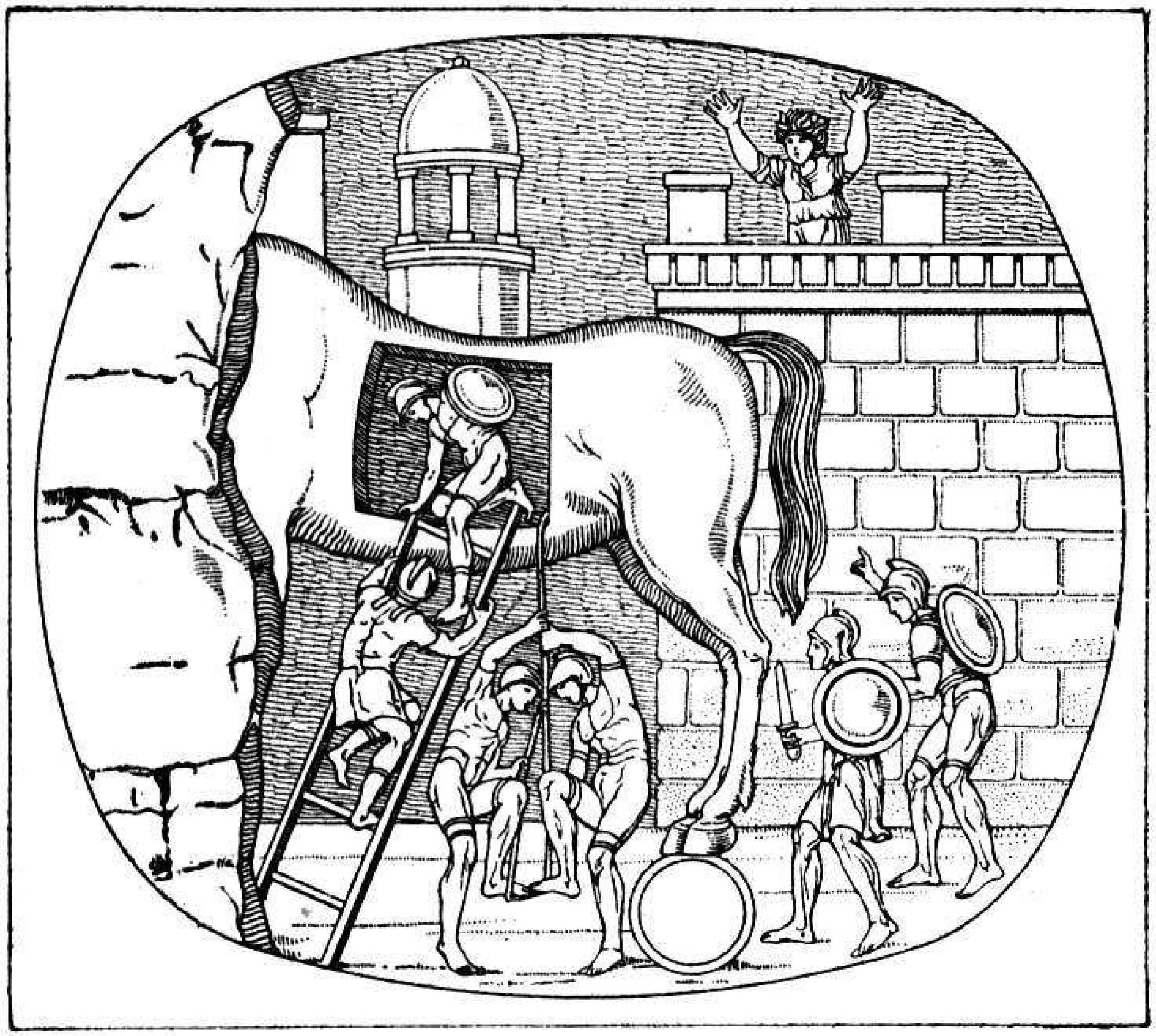
- For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$\begin{aligned}
\boldsymbol{K}(\mathbf{x}, \mathbf{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\
&= \{x_1x_1' + x_2x_2'\}^2 \\
&= \{\mathbf{x}^T\mathbf{x}'\}^2 .
\end{aligned}$$



Kernels are Trojan Horses onto Linear Models

- With kernels, complex structures can enter the realm of linear models

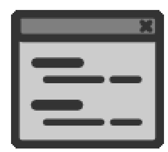
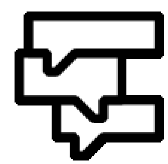
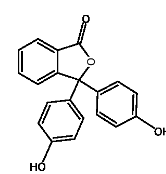


What is a kernel

$\boxed{\text{In the context of these lectures...}}$

- A kernel k is a function

$$
\begin{array}{rccc}
k: & \mathcal{X} \times \mathcal{X} & \longmapsto & \mathbb{R} \\
 & (\mathbf{x}, \mathbf{y}) & \longrightarrow & k(\mathbf{x}, \mathbf{y})
\end{array}
$$

- which compares two objects of a space $\mathcal{X}$, *e.g.*...
  - strings, texts and sequences, 
  - images, audio and video feeds,  
  - graphs, interaction networks and 3D structures 

- whatever actually... time-series of graphs of images? graphs of texts?...

Fundamental properties of a kernel

symmetric

$$k(\mathbf{x},\mathbf{y}) = k(\mathbf{y},\mathbf{x}).$$

positive-(semi)definite

for any *finite* family of points $\mathbf{x}_1,\cdots,\mathbf{x}_n$ of $\mathcal{X}$, the matrix

$$K = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_i) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_i) & \cdots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_i,\mathbf{x}_1) & k(\mathbf{x}_i,\mathbf{x}_2) & \cdots & k(\mathbf{x}_i,\mathbf{x}_i) & \cdots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & \cdots & k(\mathbf{x}_n,\mathbf{x}_i) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix} \succeq 0$$

is positive semidefinite (has a nonnegative spectrum).

$\boxed{K \text{ is often called the } \textbf{Gram matrix} \text{ of } \{\mathbf{x}_1,\cdots,\mathbf{x}_n\} \text{ using } k}$

What can we do with a kernel?

The setting

- Very loose setting: a set of objects $\mathbf{x}_1, \cdots, \mathbf{x}_n$ of $\mathcal{X}$
- **Sometimes** additional information on these objects
  - labels $\mathbf{y}_i \in \{-1, 1\}$ or $\{1, \cdots, \#(\text{classes})\}$,
  - scalar values $\mathbf{y}_i \in \mathbb{R}$,
  - associated object $\mathbf{y}_i \in \mathcal{Y}$
- A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

The Gram matrix perspective

- Imagine a little task: you have read 100 novels so far.

- You would like to know whether you will enjoy reading a **new** novel.
- A few options:
  - read the book...
  - have friends read it for you, read reviews.
  - try to guess, based on the novels you read, if you will like it

The Gram matrix perspective

Two distinct approaches

- Define what **features** can characterize a book.
  - Map each book in the library onto vectors

$$\longrightarrow \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

  typically the x_i's can describe...
  - ▷ # pages, language, year 1st published, country,
  - ▷ coordinates of the main action, keyword counts,
  - ▷ author's prizes, popularity, booksellers ranking

- Challenge: find a decision function using 100 ratings and features.

The Gram matrix perspective

- Define what makes **two novels similar**,
  - Define a kernel k which quantifies novel similarities.
  - Map the library onto a Gram matrix

$$\longrightarrow \quad K = \begin{bmatrix} k(b_1, b_1) & k(b_1, b_2) & \cdots & k(b_1, b_{100}) \\ k(b_2, b_1) & k(b_2, b_2) & \cdots & k(b_2, b_{100}) \\ \vdots & \vdots & \ddots & \vdots \\ k(b_n, b_1) & k(b_n, b_2) & \cdots & k(b_{100}, b_{100}) \end{bmatrix}$$

- Challenge: find a decision function that takes this 100×100 matrix as an input.

The Gram matrix perspective

Given a new novel,

- with the **features approach**, the prediction can be rephrased as **what are the features of this new book**? what **features** have I found in the past that were good indicators of my taste?

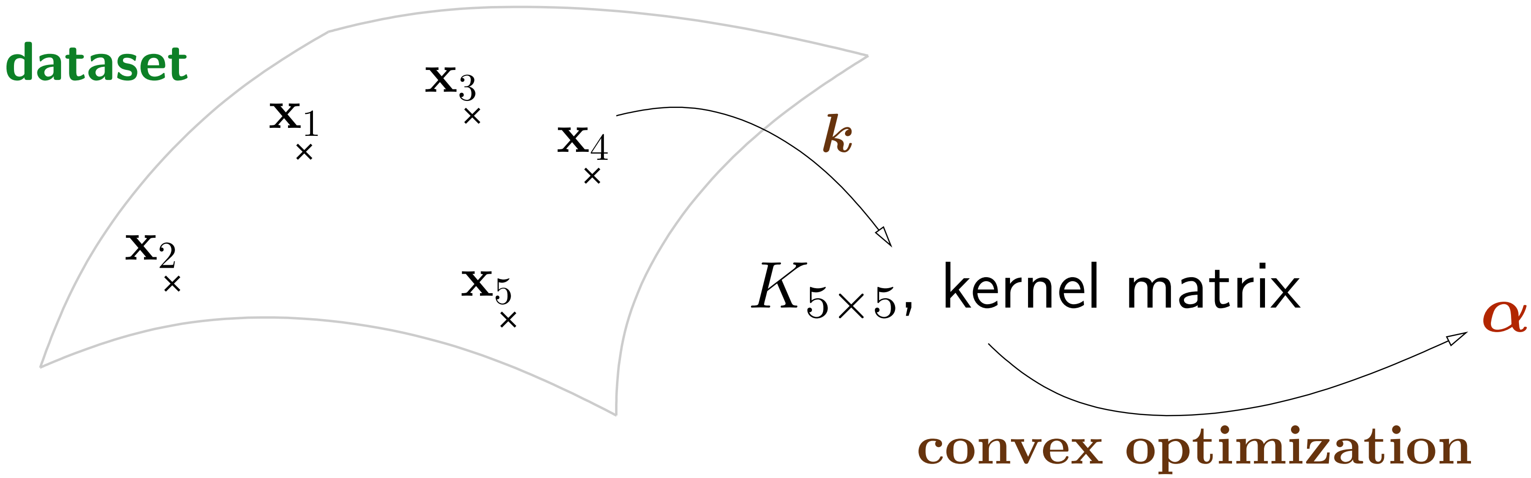
- with the **kernel approach**, the prediction is rephrased as **which novels this book is similar or dissimilar to?** what **pool of books** did I find the most influentials to define my tastes accurately?

kernel methods **only use kernel similarities**, do not consider features.

Features can help define similarities, but **never considered elsewhere**.

The Gram matrix perspective

in kernel methods, clear separation between the kernel...



and **Convex optimization** (thanks to psdness of K, more later) to output the α's.

Kernel Trick

Given a dataset $\{x_1, \cdots, x_N\}$ and a new instance x_{new}

Many data analysis methods only depend on $x_i^T x_j$ and $x_i^T \boldsymbol{x}_{\text{new}}$

- Ridge regression
- Principal Component Analysis
- Linear Discriminant Analysis
- Canonical Correlation Analysis
- *etc.*

Replace these evaluations by $\boldsymbol{k}(x_i, x_j)$ and $\boldsymbol{k}(x_i, \boldsymbol{x}_{\text{new}})$

- This will even work if the x_i's are not in a dot-product space! (strings, graphs, images *etc.*)