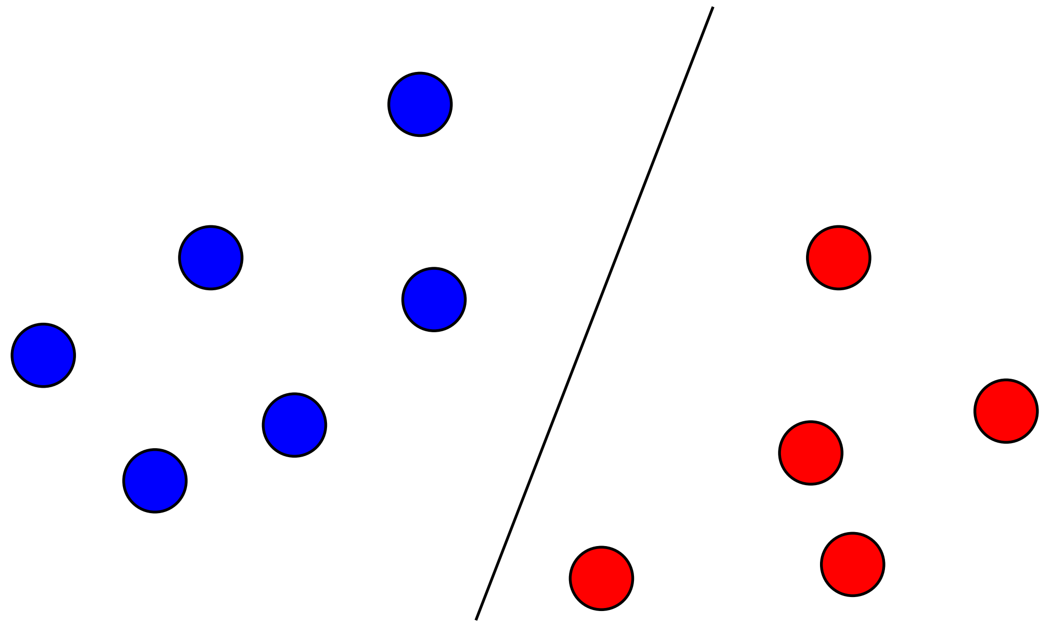
Foundation of Intelligent Systems, Part I

SVM's & Kernel Methods

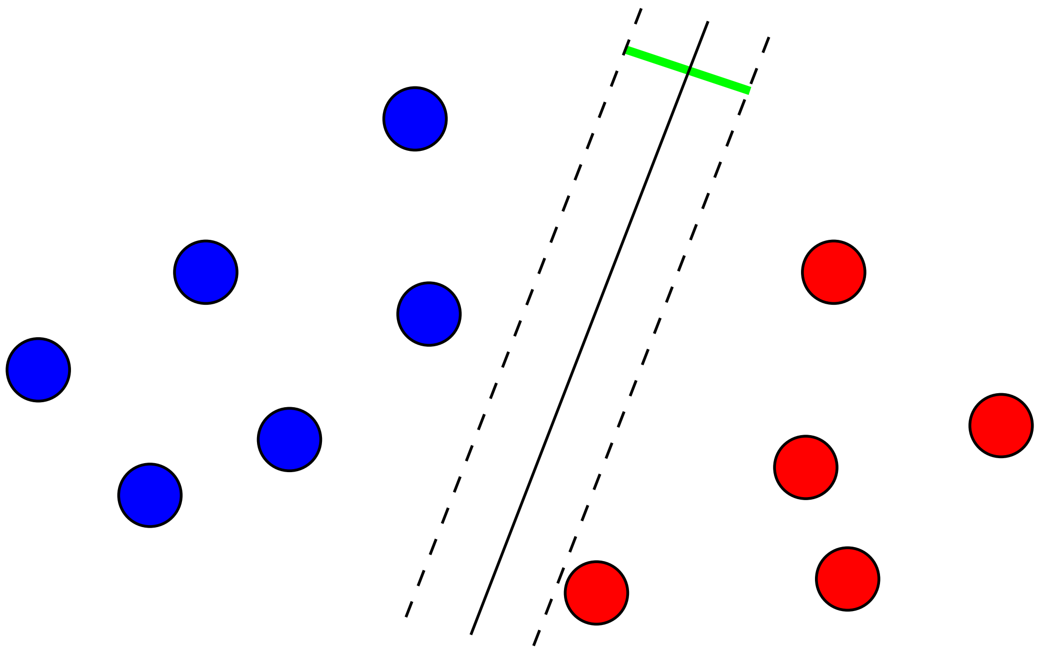
mcuturi@i.kyoto-u.ac.jp

Support Vector Machines
The linearly-separable case

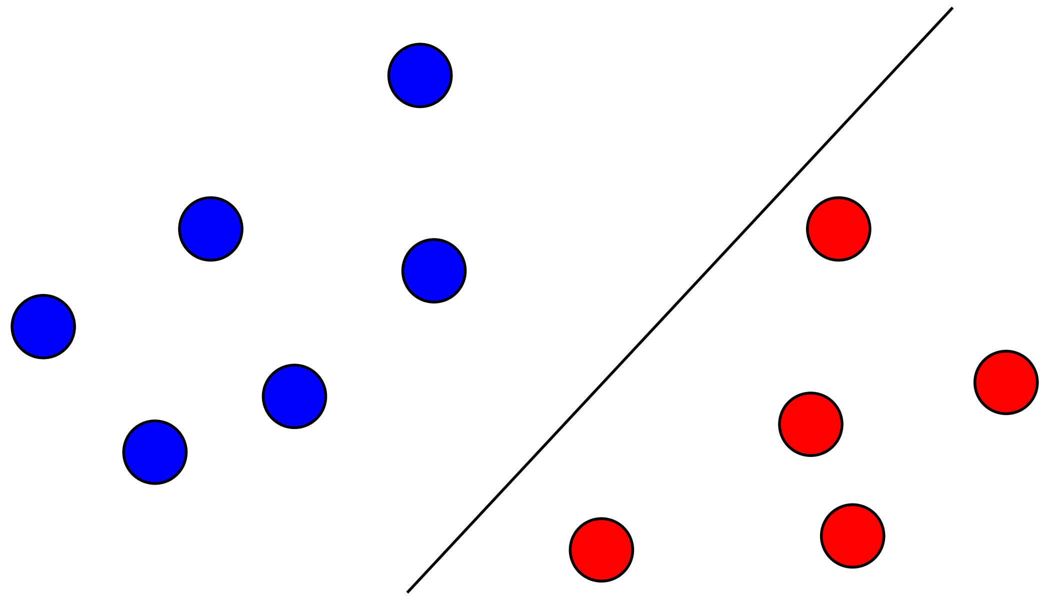
A criterion to select a linear classifier: the margin ?



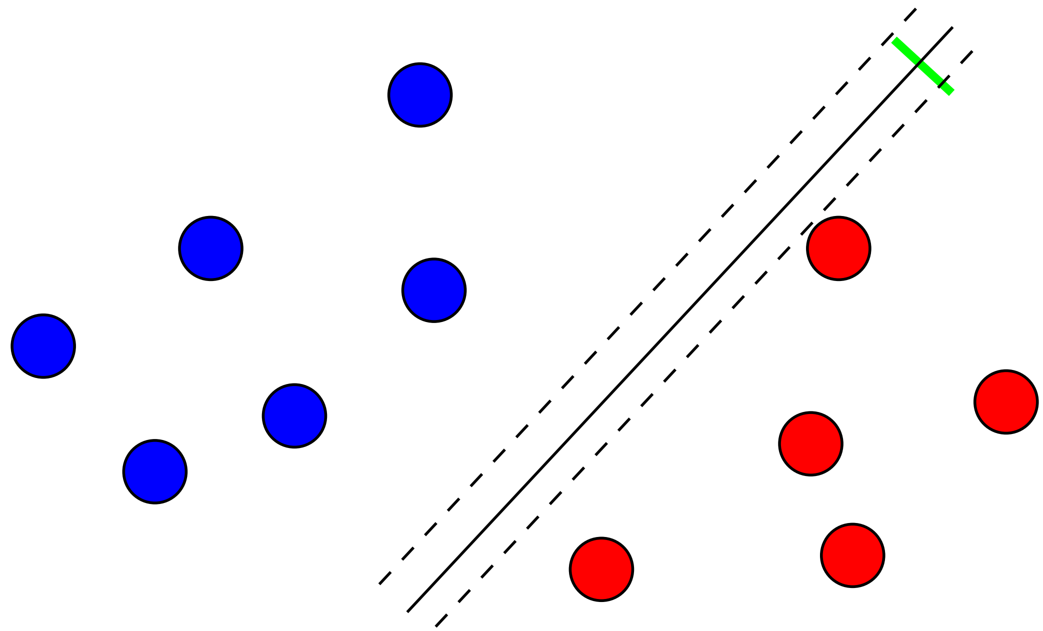
A criterion to select a linear classifier: the margin ?



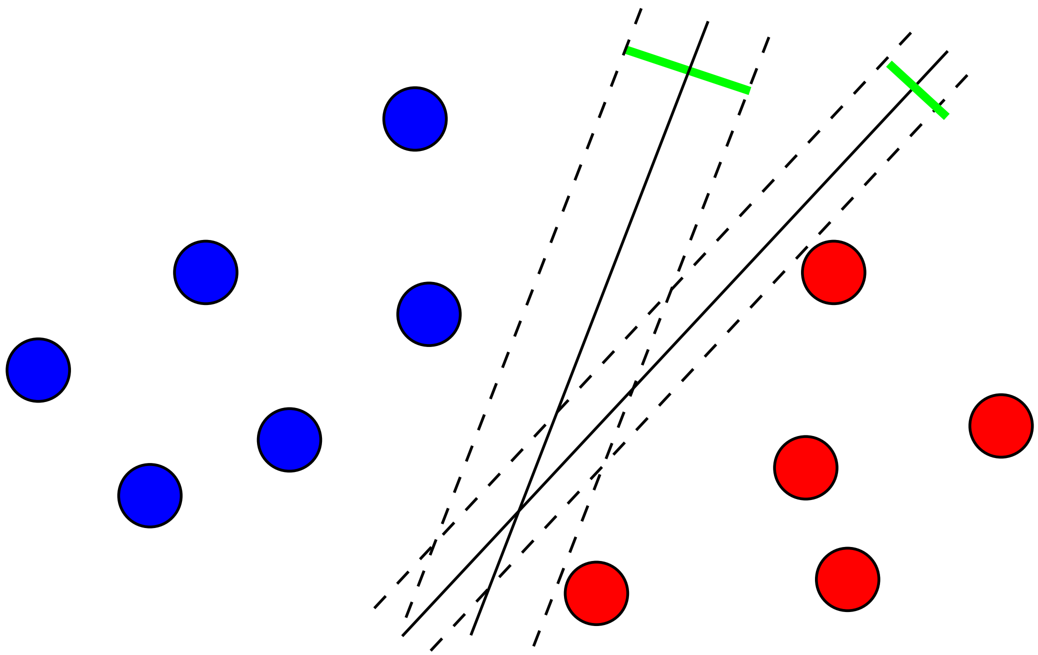
A criterion to select a linear classifier: the margin ?



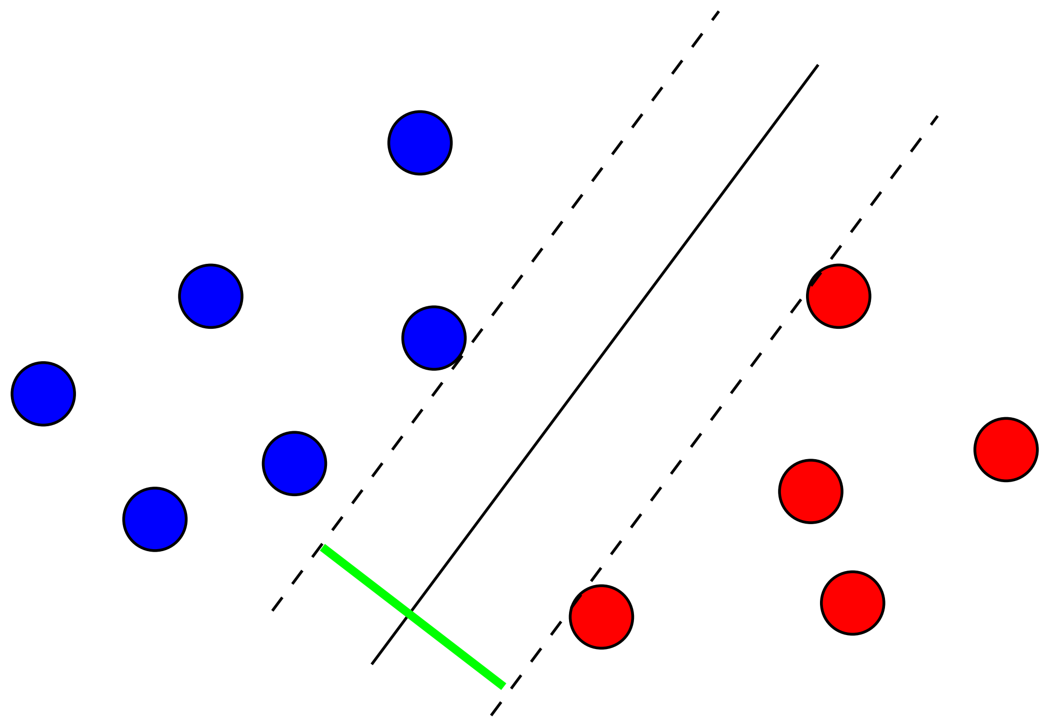
A criterion to select a linear classifier: the margin ?



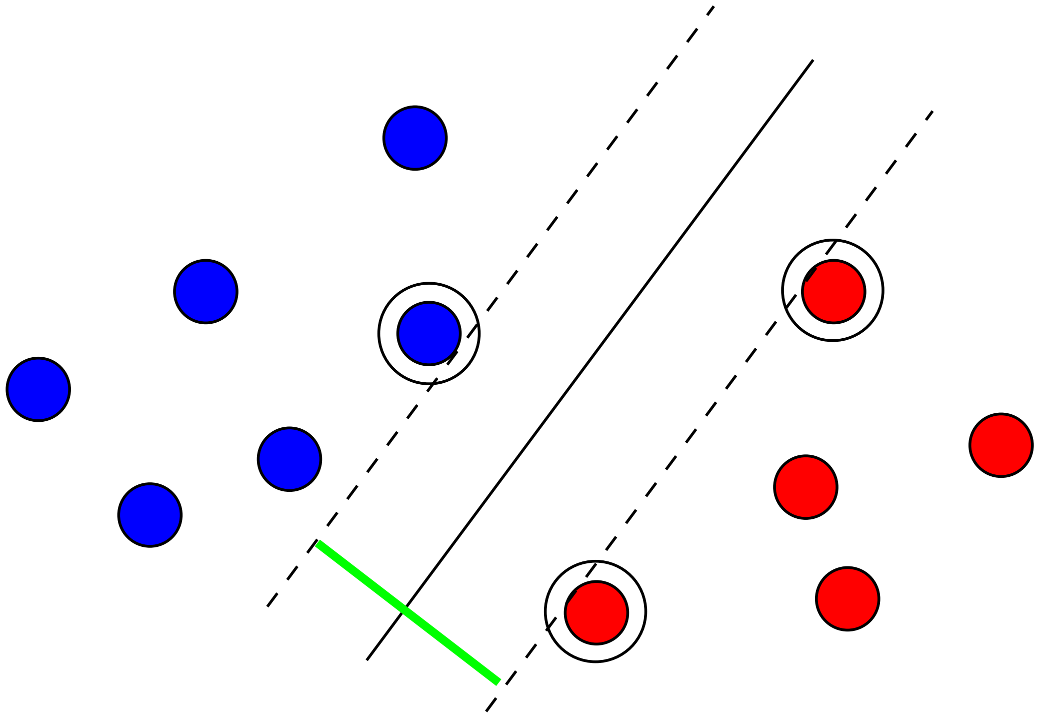
A criterion to select a linear classifier: the margin ?



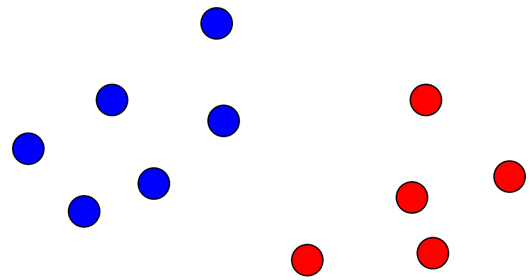
Largest Margin Linear Classifier ?



Support Vectors with Large Margin



In equations



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

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

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

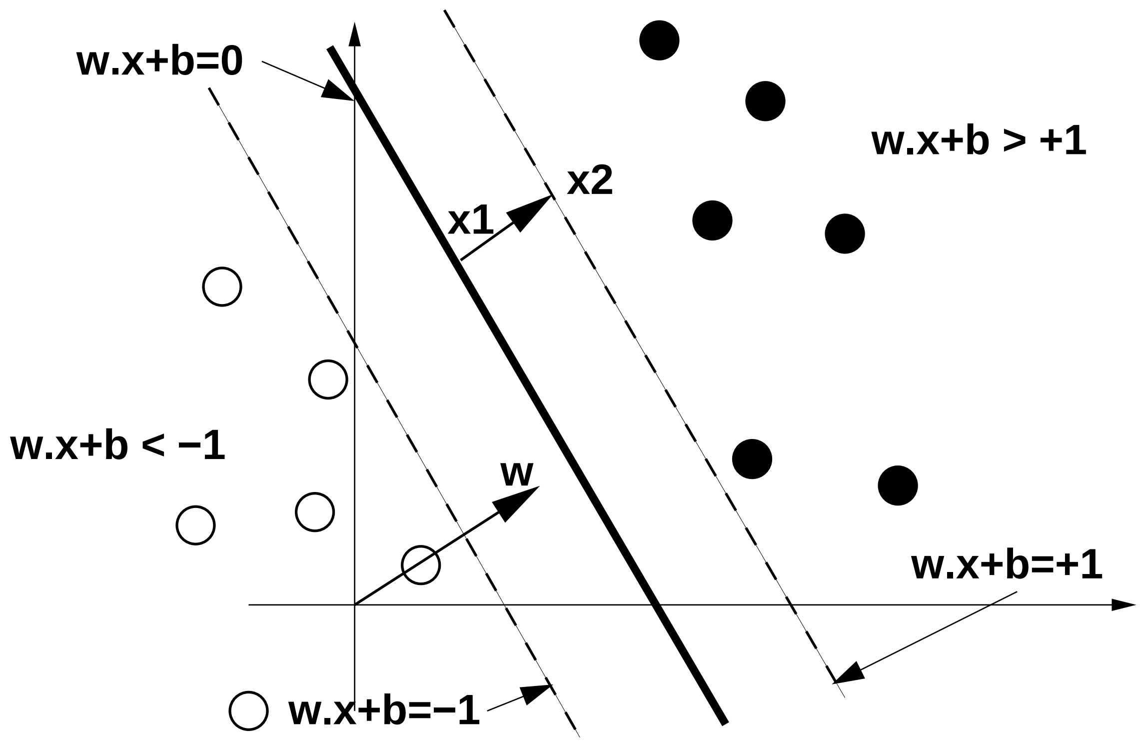
- We assume (for the moment) that the data are **linearly separable**, i.e., that there exists $(\mathbf{w}, b) \in \mathbb{R}^d \times \mathbb{R}$ such that:

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

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.

All training points should be on the appropriate side

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

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

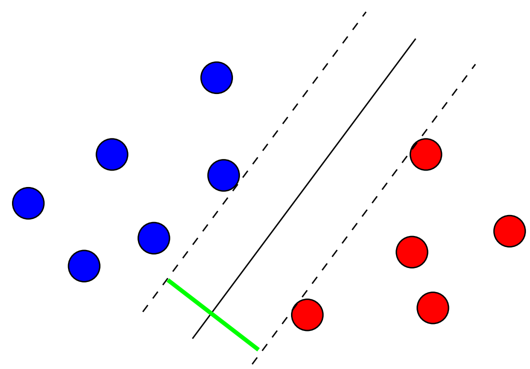
- For negative examples ($y_i = -1$) this means:

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

- 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:

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

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.
- **Strong duality holds** : primal and dual problems have the **same optimum**.
- KKT gives us $\alpha_i(\mathbf{y}_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1) = 0$,
  ...*hence*, either $\boldsymbol{\alpha_i = 0}$ or $\boldsymbol{\mathbf{y}_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) = 1}$.
- $\alpha_i \neq 0$ **only** for points on the support hyperplanes $\{(\mathbf{x}, \mathbf{y}) |\, \mathbf{y}_i(\mathbf{w}^T \mathbf{x}_i + b) = 1\}$.

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*.
> α^* can be computed using optimization software
> ($e.g.$ built-in `matlab` function)

Recovering the optimal hyperplane

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

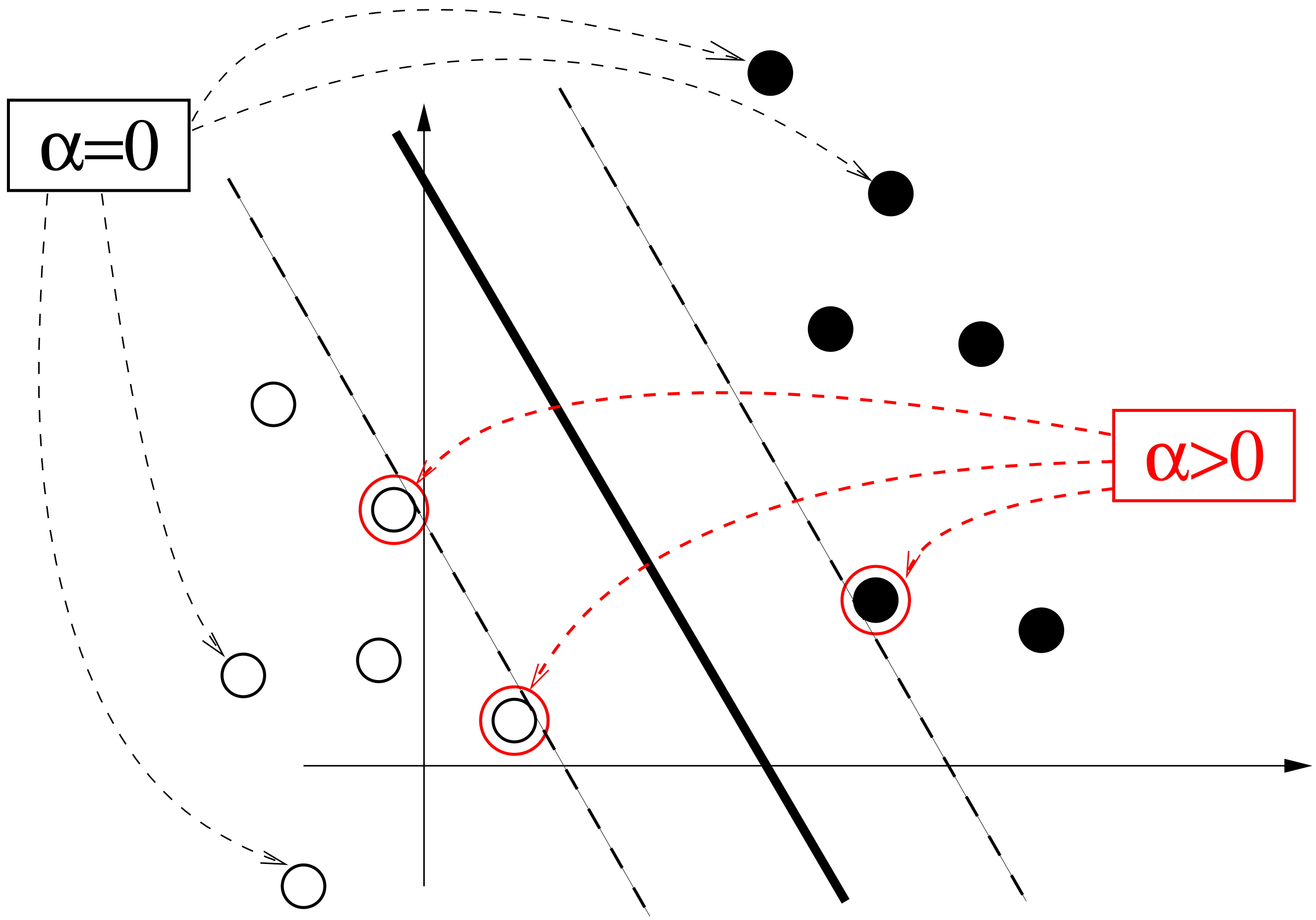
$$b^* = -\frac{1}{2}\left(\min_{\mathbf{y}_i=1,\alpha_i>0}(\mathbf{w}^T\mathbf{x}_i) + \max_{\mathbf{y}_i=-1,\alpha_i>0}(\mathbf{w}^T\mathbf{x}_i)\right)$$

- the **decision function** is therefore:

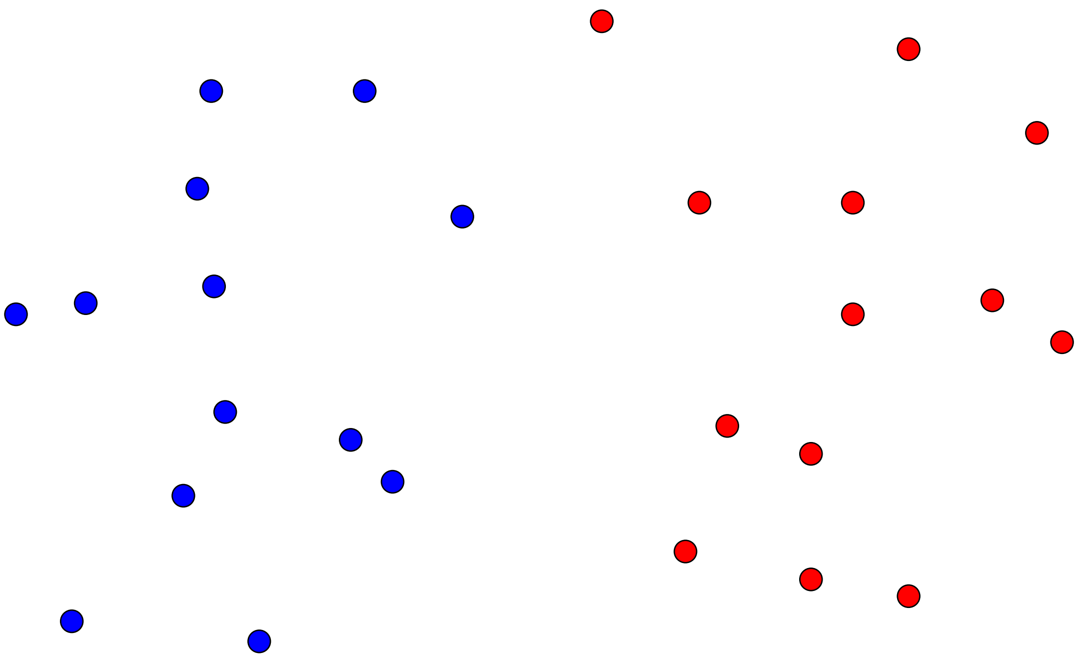
$$\begin{aligned} f^*(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} + b^* \\ &= \left(\sum_{i=1}^n y_i\alpha_i\mathbf{x}_i^T\right)\mathbf{x} + b^*. \end{aligned}$$

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

Interpretation: support vectors

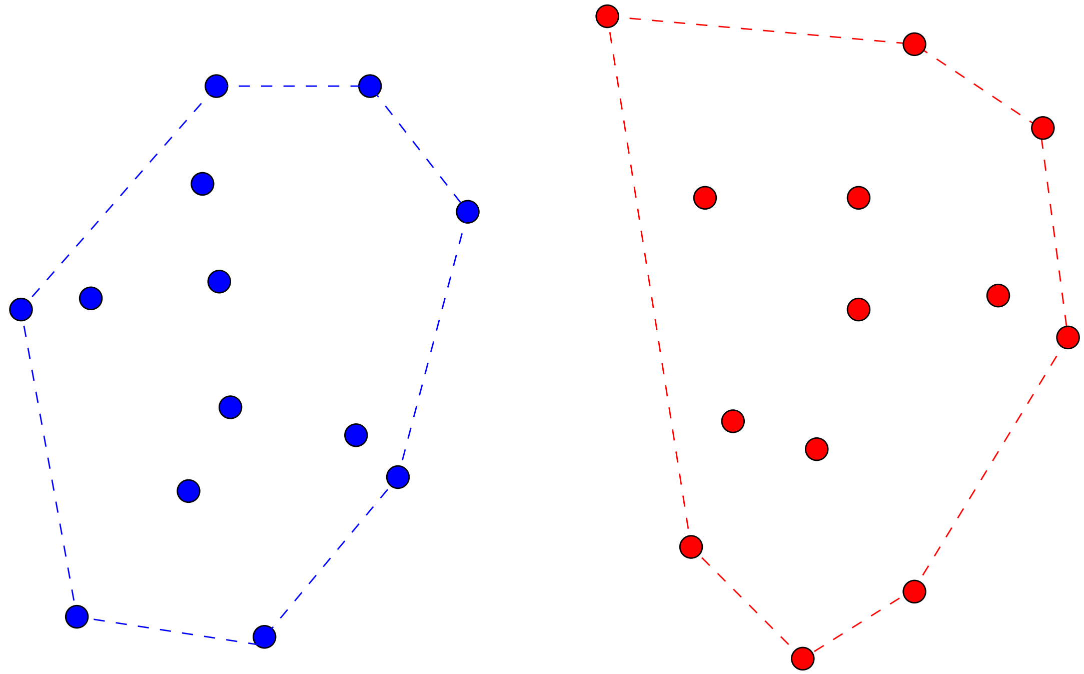


Another interpretation: Convex Hulls



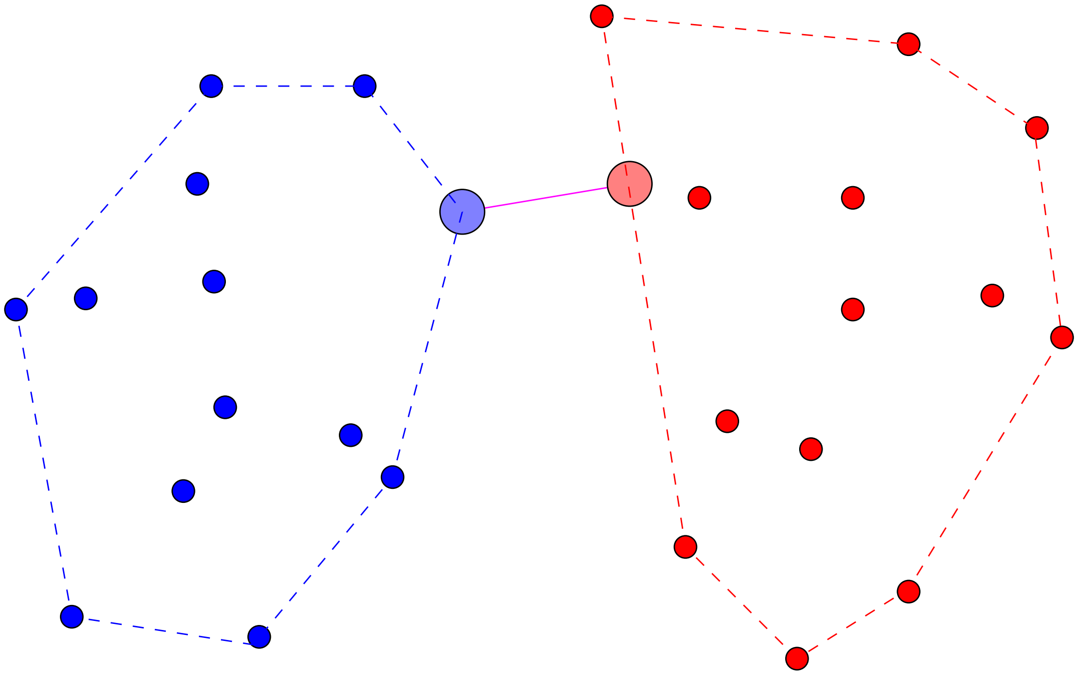
go back to 2 sets of points that are linearly separable

Another interpretation: Convex Hulls



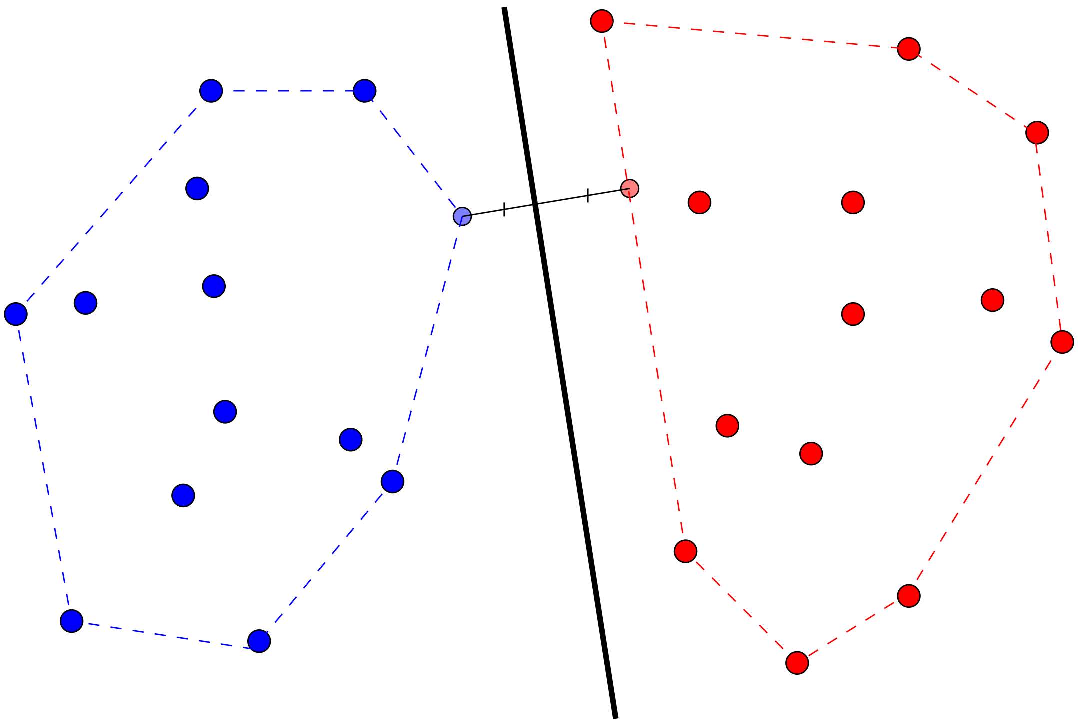
Linearly separable = convex hulls do not intersect

Another interpretation: Convex Hulls



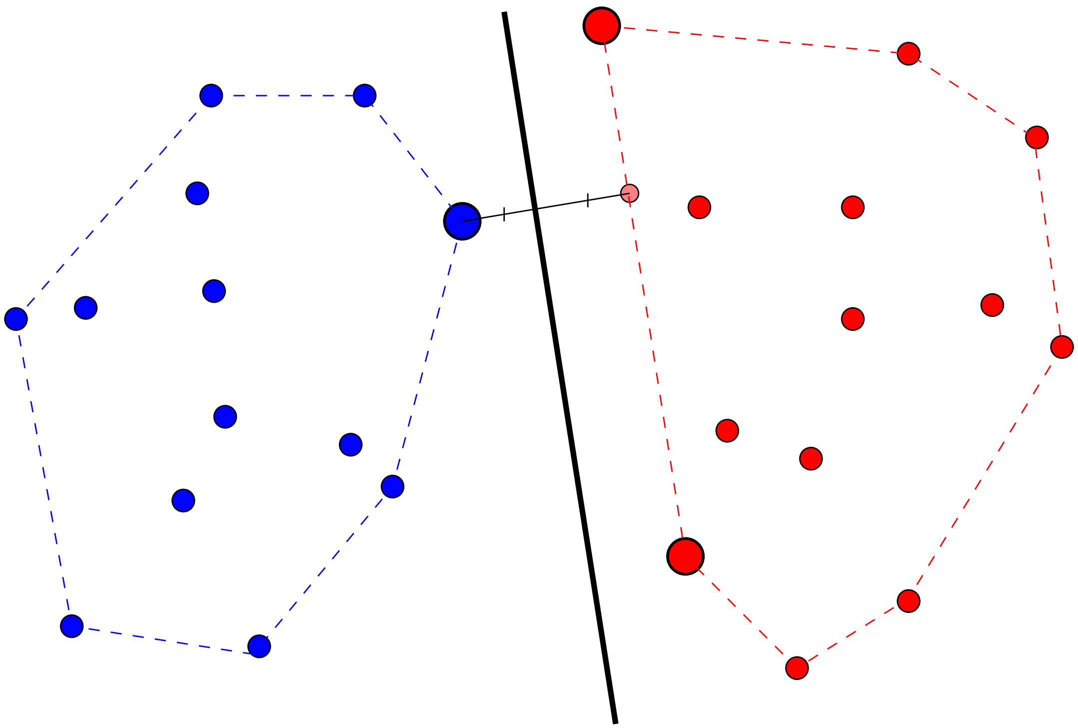
Find two closest points, one in each convex hull

Another interpretation: Convex Hulls



The SVM = bisection of that segment

Another interpretation: Convex Hulls

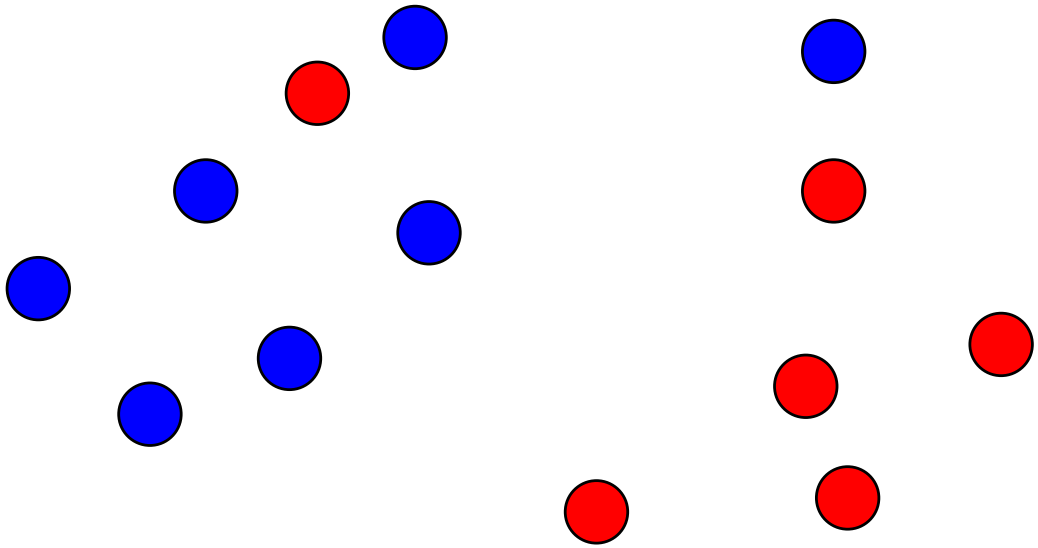


support vectors = extreme points of the faces on which the two points lie

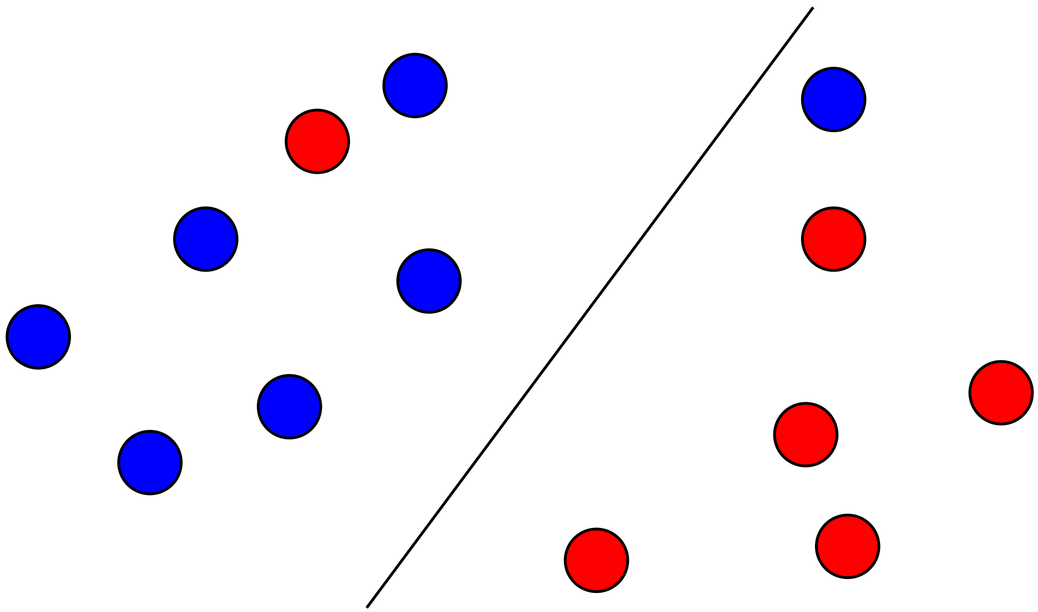
The non-linearly separable case

(when convex hulls intersect)

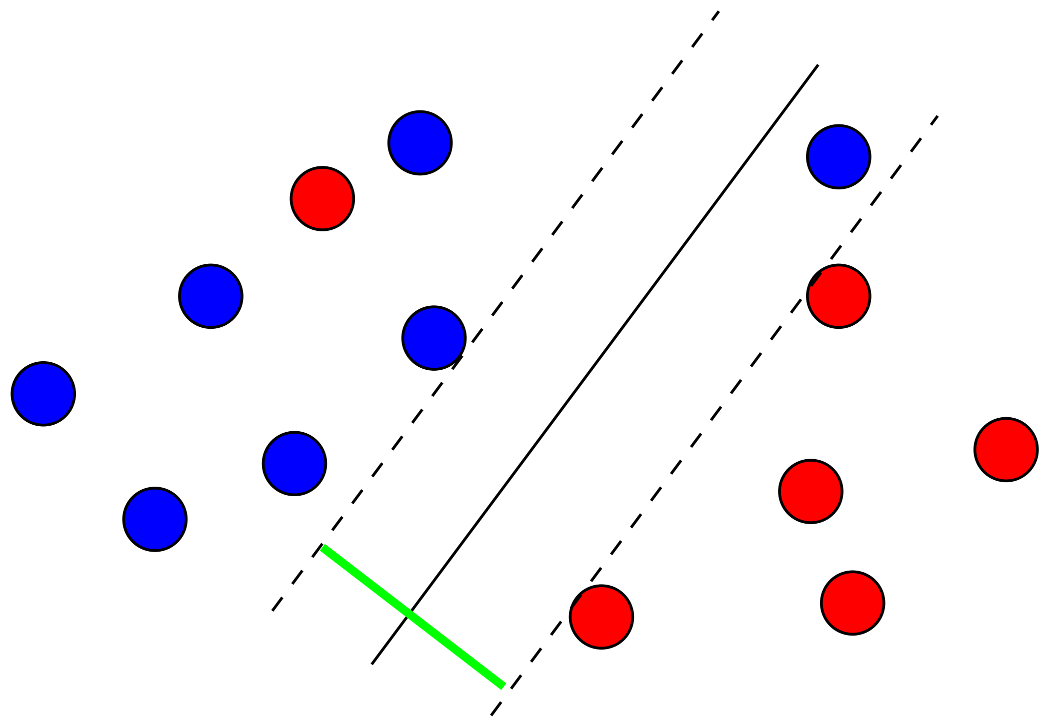
What happens when the data is not linearly separable?



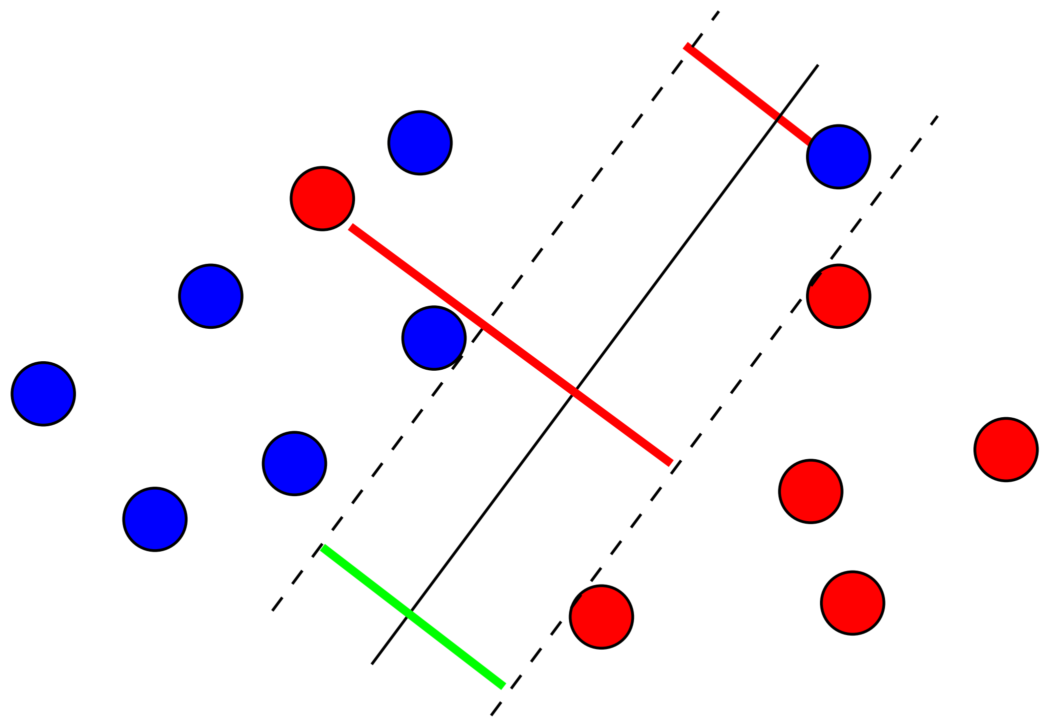
What happens when the data is not linearly separable?



What happens when the data is not linearly separable?



What happens when the data is not linearly separable?



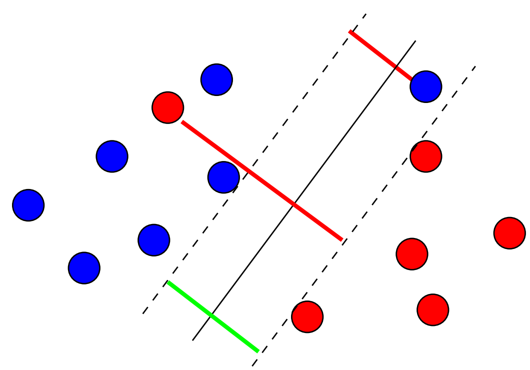
Soft-margin SVM ?

- 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

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

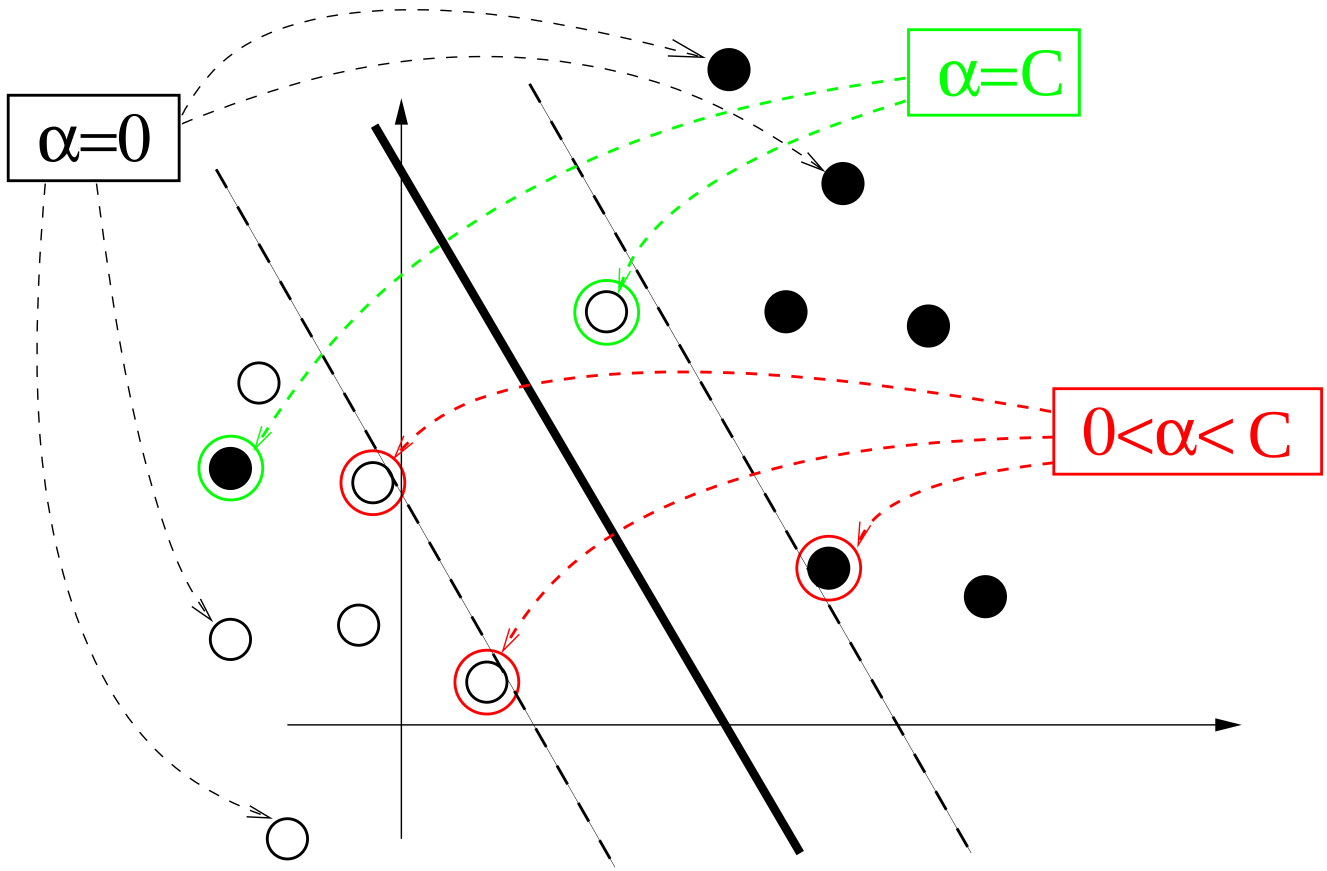
- In that case the dual function

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

which is finite under the constraints:

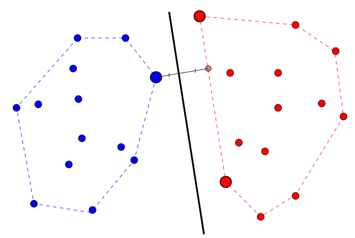
$$\begin{cases} 0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n}\alpha_i\mathbf{y}_i = 0. \end{cases}$$

Interpretation: bounded and unbounded support vectors

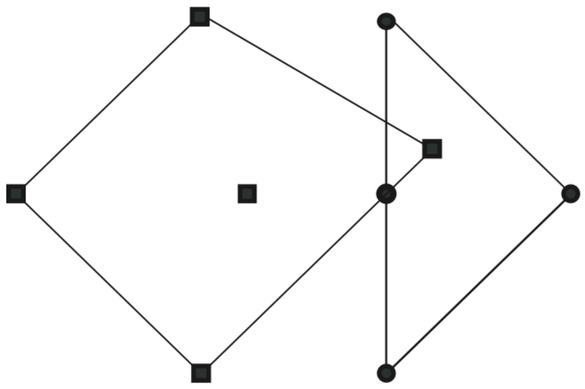


What about the convex hull analogy?

- Remember the separable case



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



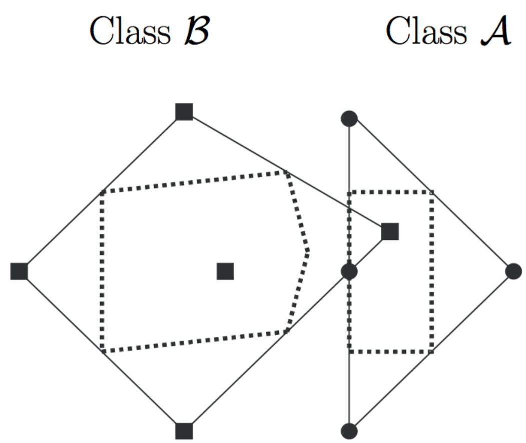
What about the convex hull analogy?

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:

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

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 $\mathbf{y}$:

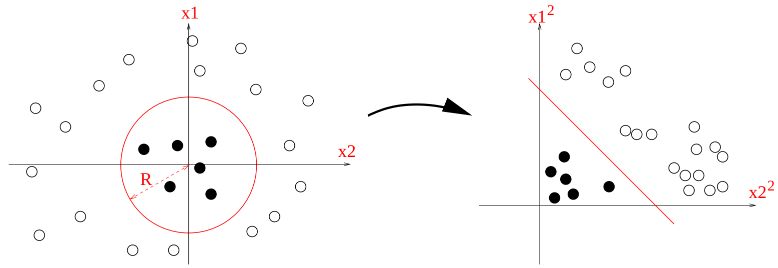
$$\begin{array}{ll} \text{maximize} & g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2}\alpha^T (\boldsymbol{K} \odot \mathbf{y}\mathbf{y}^T)\alpha \\ \text{such that} & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n \\ & \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0. \end{array}$$

- $\boldsymbol{K}$'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

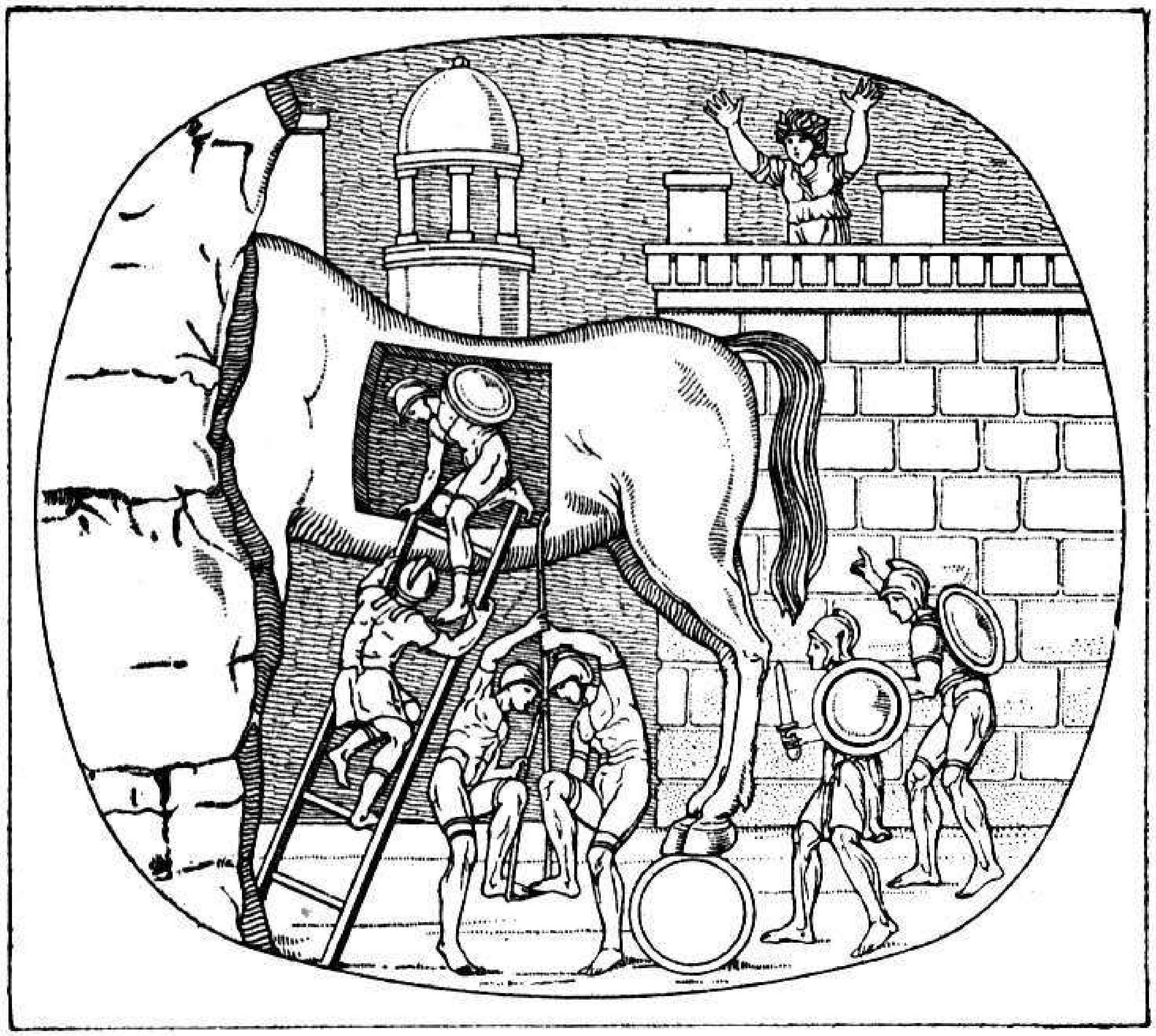
- For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

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



Kernels are Trojan Horses onto Linear Models

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

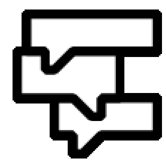
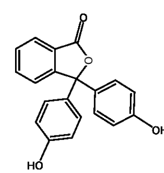


What is a kernel

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).

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

What can we do with a kernel?

The setting

- Pretty simple 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 $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

A few intuitions on the possibilities of kernel methods

Important concepts and perspectives

- The functional perspective: represent **points as functions**.
- **Nonlinearity** : linear combination of kernel evaluations.
- Summary of a sample through its **kernel matrix**.

Represent any point in $\mathcal{X}$ as a function

For every **x**, the map
$$\mathbf{x} \longrightarrow k(\mathbf{x}, \cdot)$$
associates to **x** a function $k(\mathbf{x}, \cdot)$ from $\mathcal{X}$ to $\mathbb{R}$.

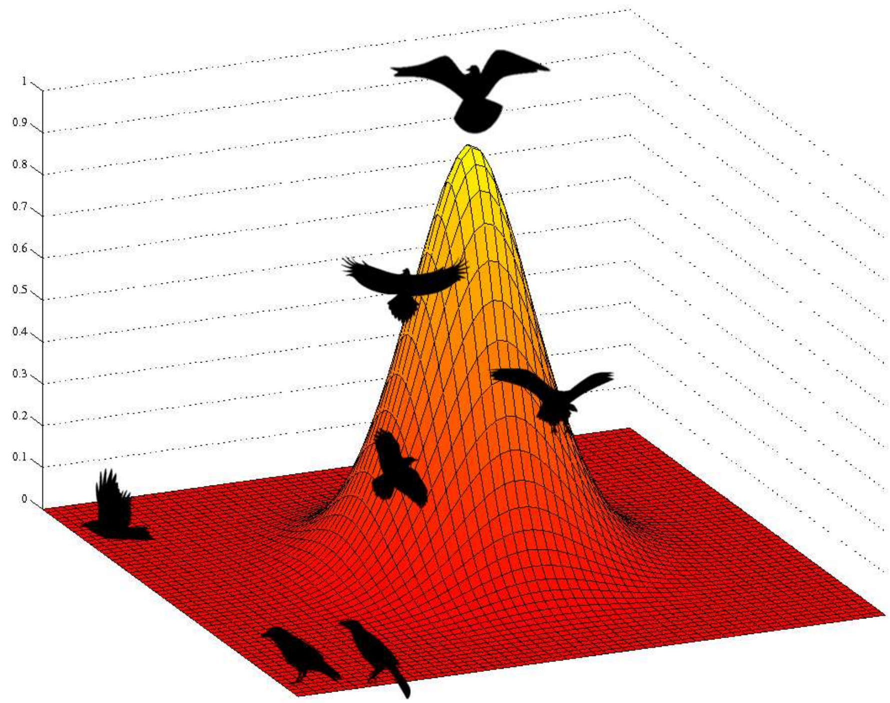
- Suppose we have a kernel k on bird images

- Suppose for instance

k ( , ) $= .32$

Represent any point in $\mathcal{X}$ as a function

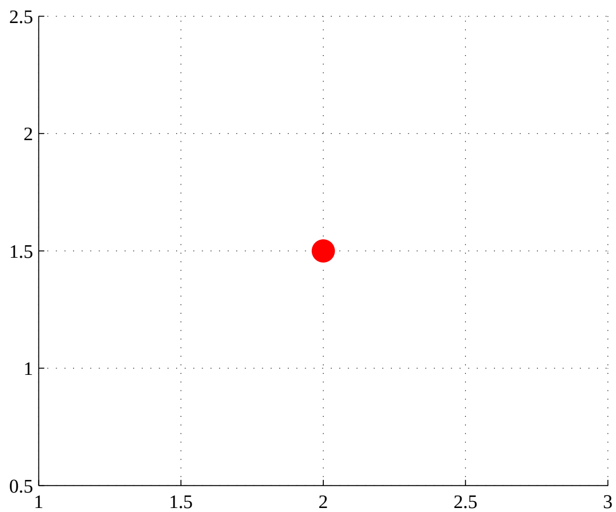
- We examine one image in particular: 
- With kernels, we get a **representation** of that bird as a real-valued function, defined on the space of birds, represented here as $\mathbb{R}^2$ for simplicity.



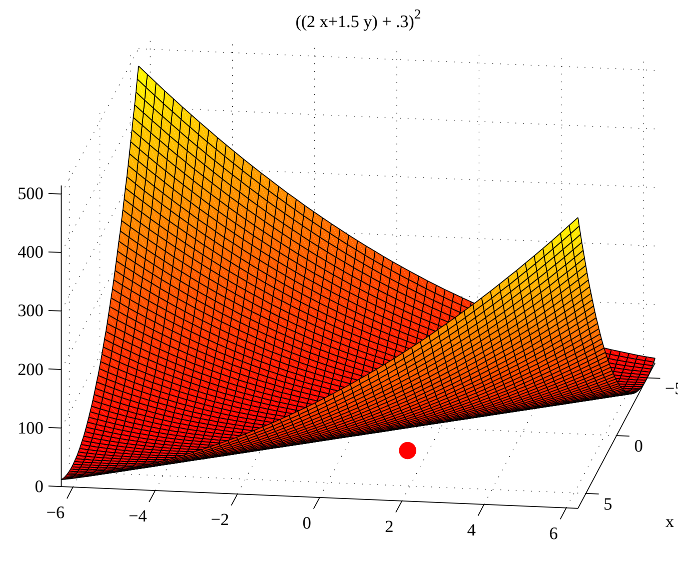
schematic plot of $k($  $, \cdot \,)$.

Represent any point in $\mathcal{X}$ as a function

- If the bird example was confusing...
- $k\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) = \left(\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} x' \\ y' \end{bmatrix} + .3\right)^2$
- From a point in $\mathbb{R}^2$ to a function defined over $\mathbb{R}^2$.



$\rightarrow$



- We assume implicitly that the **functional representation** will be more useful than the **original representation**.

Decision functions as linear combination of kernel evaluations

- Linear decisions functions are a major tool in statistics, that is functions

$$f(\mathbf{x}) = \beta^T \mathbf{x} + \beta_0.$$

- Implicitly, a point $\mathbf{x}$ is processed depending on its characteristics x_i,

$$f(\mathbf{x}) = \sum_{i=1}^{d} \boldsymbol{\beta_i} x_i + \boldsymbol{\beta_0}.$$

  the free parameters are scalars $\boldsymbol{\beta_0, \beta_1, \cdots, \beta_d}$.

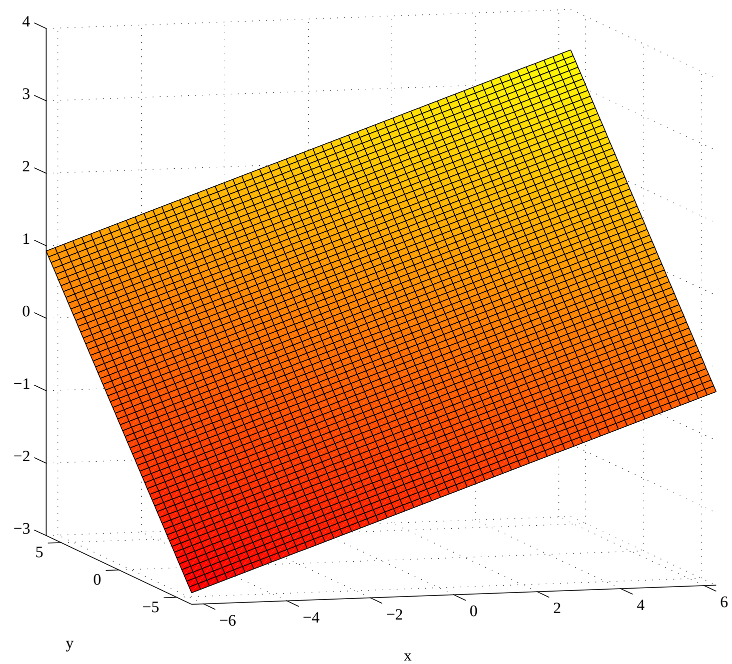
- Kernel methods yield candidate decision functions

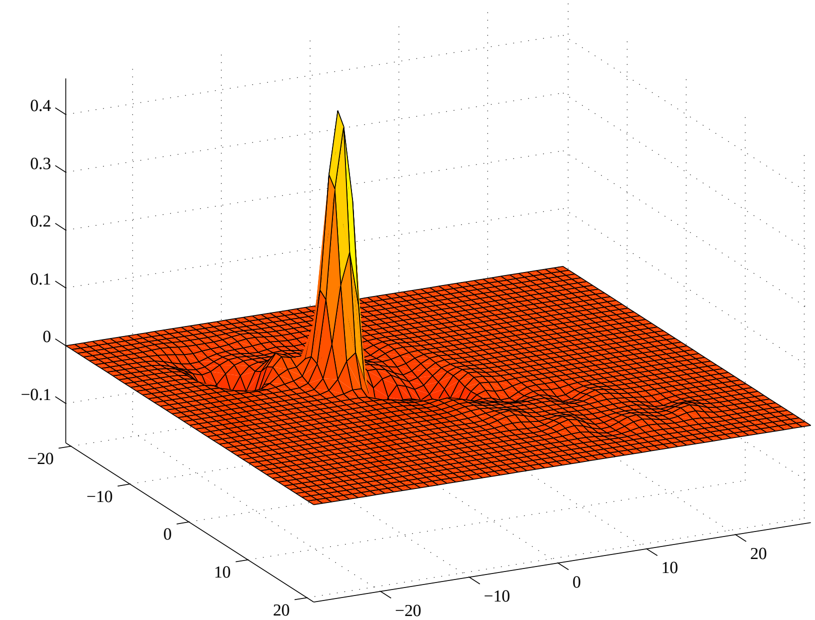
$$f(\mathbf{x}) = \sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\alpha_j} k(\mathbf{x}_j, \mathbf{x}) + \boldsymbol{\alpha_0}.$$

  the free parameters are scalars $\boldsymbol{\alpha_0, \alpha_1, \cdots, \alpha_n}$.

Decision functions as linear combination of kernel evaluations

- linear decision surface / linear expansion of **kernel surfaces** (here $k_G(\mathbf{x}_i, \cdot)$)





- Kernel methods are considered **non-linear** tools.
- Yet not completely "nonlinear" $\rightarrow$ only one-layer of nonlinearity.

kernel methods use the data as a functional base to define decision functions

Decision functions as linear combination of kernel evaluations

database $\{\mathbf{x}_i, i = 1, \ldots, N\}$

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \; k(\mathbf{x}_i, \mathbf{x})$$

kernel definition

weights α estimated
with a kernel machine

- f is any predictive function of interest of a new point **x**.
- Weights α are **optimized** with a kernel machine (*e.g.* support vector machine)

intuitively, kernel methods provide decisions based on how *similar* a point x is to each instance of the training set

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?

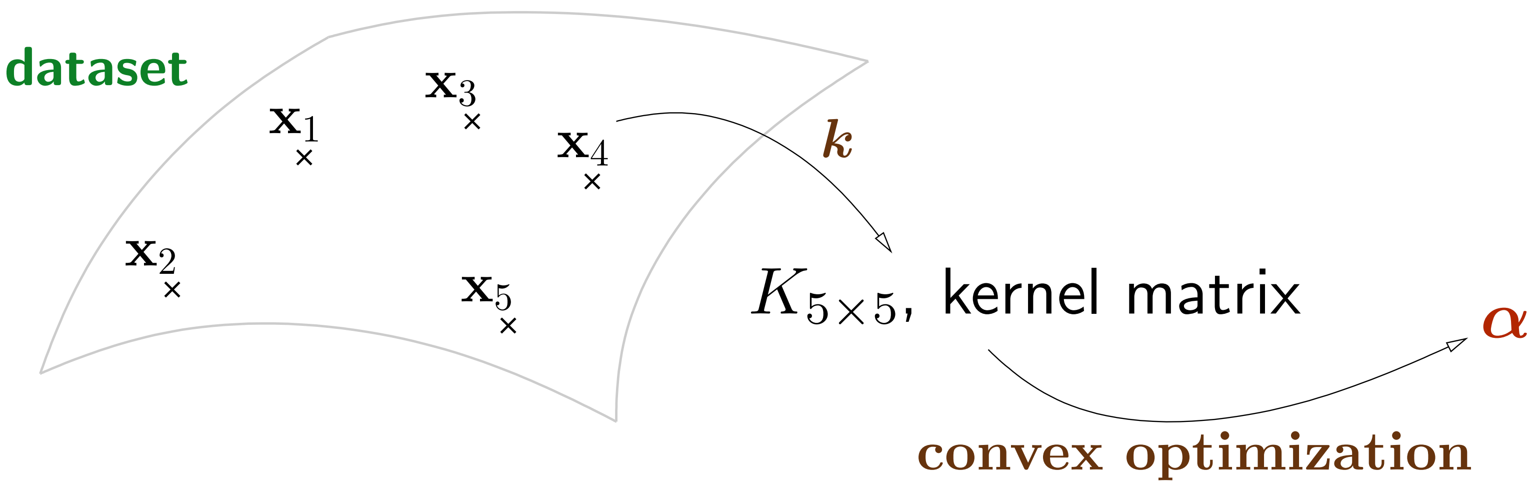
- with the **kernel approach**, the prediction is rephrased as **which novels this book is similar or dissimilar to?** what **pool of books** did I find the most influentials to define my tastes accurately?

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.