Foundation of Intelligent Systems, Part I

Classification

mcuturi@i.kyoto-u.ac.jp

Last Lecture : Regression

- Mentioned the **Maximum Likelihood** perspective on LS-regression

$$\log \mathcal{L}(\mathbf{a}, b) = C - \frac{1}{2\sigma^2} \underbrace{\sum_{j=1}^{N} \|y_j - (\mathbf{a}^T \mathbf{x}_j + b)\|^2}_{L(\mathbf{a},b)} .$$

- Provided a **geometric** perspective on LS regression through **projections**

Least Squares Regression
$\Updownarrow$
Projecting the vector of **observed predicted** variable in
$\text{span}\{$ vectors of **observed predictor** variables + constant vector$\}$

- Many issues with LS regression... Hence advanced regression techniques
  - Ridge Regression
  - **Subset selection**
  - **Lasso**
- we will talk about these in 3 lectures when discussing **sparsity**.

Today

- Classification, differences with regression
- Binary classification
- Linear classification algorithms
  - **Logistic Regression**
  - Ideally, **Linear Discriminant Analysis**, but no time.
  - **Perceptron rule**
  - **Support Vector Machine**
- Once this is done, we will move on to more theory in next lecture about statistical learning theory.

Classification

Starting Again With Regression

Many observations of the same data type, with label

- we still 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 **regression**
    - If $y_j \in \mathcal{S}$ where $\mathcal{S}$ is a finite set, **multiclass classification**.
    - If $\mathcal{S}$ only has two elements, **binary classification**.

Examples

Multiclass Classification

- Classify images of fruits into fruit category

- Classify images of handwritten digits into digits from 0 to 9
- Classify musical tunes, books, movies into genres
- Classify proteins into functional classes

img source

Examples

Binary Classification

- 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 passenger as **suspect/not suspect**
- Classify body tumor as **begign/malign** to detect cancer
- *etc.*

Why use a new name?

Our objective is to **build a function** $f : \mathbb{R}^d \rightarrow \mathcal{S}$
To do so, we need to evaluate the accuracy of a function,
how well $\boldsymbol{f}(\mathbf{x}_j)$ compares with the true answer y_j.

In **conventional regression - linear regression**

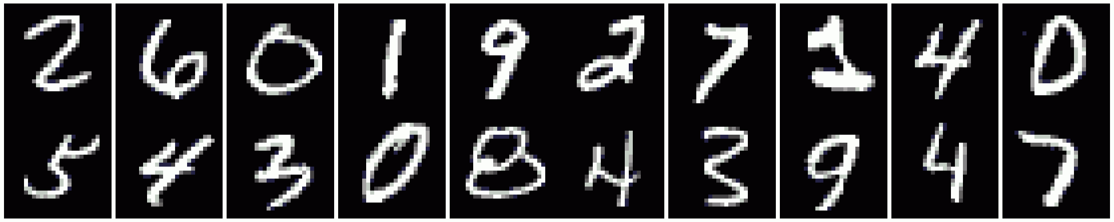
- We have used consistently $\sum_{j=1}^{N} \|\boldsymbol{f}(\mathbf{x}_j) - y_j\|^2$ to select a good $\boldsymbol{f}$.
- $\mathbb{R}$ is a **metric** space... $\|37.354 \text{ JPY} - 36.000 \text{ JPY}\| = \mathbf{1354}$
  - sense of closeness between possible answers
- $\mathbb{R}$ is a totally **ordered** set... $36.000 \text{ JPY} < 37.354 \text{ JPY}$
  - notion of total hierarchy between possible answers

In **discrete labels in classification**

- No distance, no order is assumed nor available in general.
- No order for musical genres jazz $>$ bossa-nova ?
- No distance between fruits $\|$kiwi $-$ banana$\|$?

Digits recognition

- Use a database such as



paired with the corresponding labels,

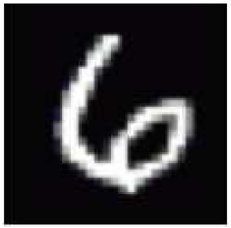
$$(2, 6, 0, 1, 9, 2, 7, 1, 4, 0, 5, 4, 3, 0, 8, 4, 3, 9, 4, 7).$$

to build an **automated recognition system** for handwritten digits.

- useful for post office, check recognition, tax office, *etc.*.

Labels are usually unordered and without a metric

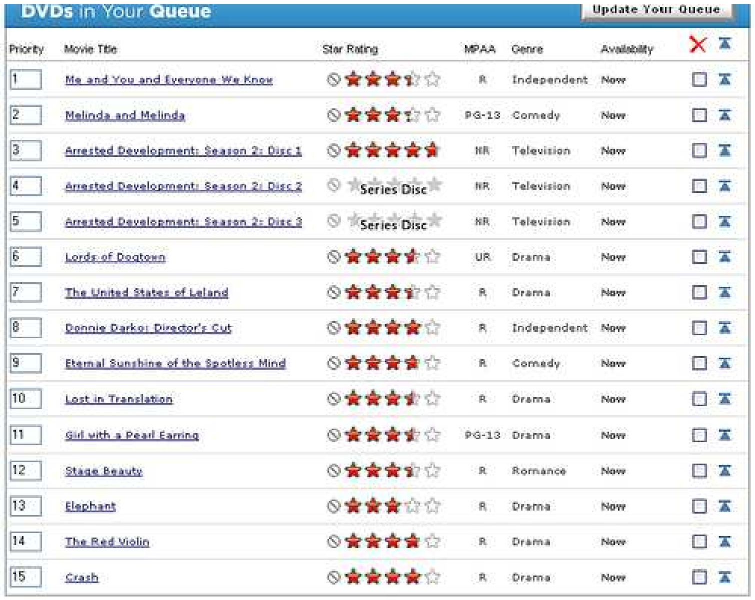
- The set of labels is $\mathcal{S} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Yet there is no distance/order in $\mathcal{S}$ for this task.
- Suppose the image given to the recognition system is



- The answer $\mathbf{5}$ is not better than $\mathbf{0}$ because the number $\mathbf{5}$ is closer to $\mathbf{6}$ than $\mathbf{0}$.

Sometimes discrete labels can be given with a metric

- Suppose the task is to guess the rating in stars of a movie



- User inputs are in $\mathcal{S} = \{1, 2, 3, 4, 5\}$
- In this case **standard regression techniques** may be applied because,
  - the natural metric $\|5 - 3\|$ works
  - linear regression works because the order is also valid.
  - the final user does not mind getting fractional predictions ($e.g.$ 3.85)

Binary Classification

$$\operatorname{card}\mathcal{S} = 2.$$

Usually $\mathcal{S} = \{0, 1\}$ or $\mathcal{S} = \{-1, 1\}$ or $\mathcal{S} = \{-, +\}$ or $\mathcal{S} = \{Y, N\}$

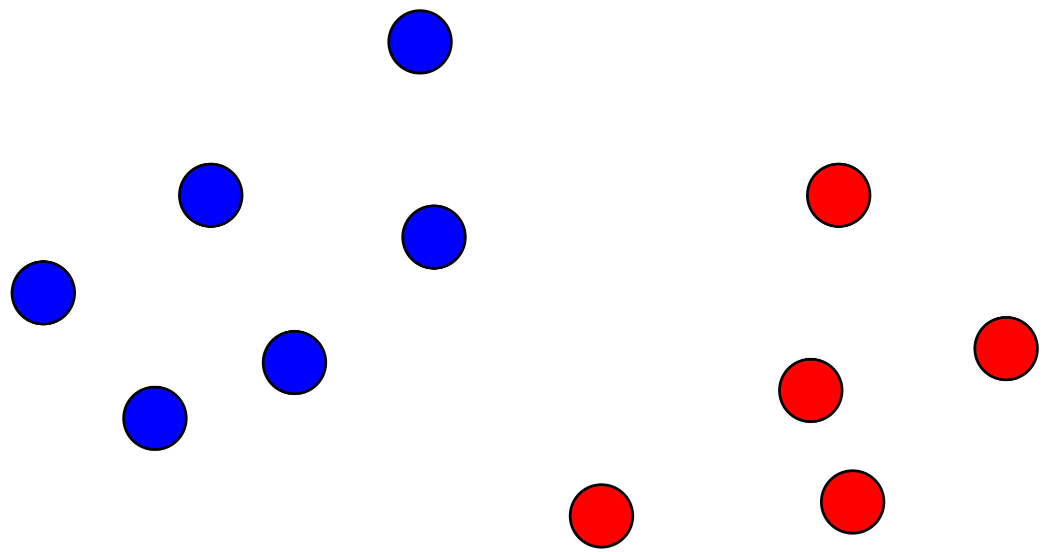
Data

- The **Data** we have: a bunch of vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_N$.
- Ideally, to infer a "yes/no" rule, we need the **correct answer** for each vector.
- We consider thus a set of **pairs of vector/bit**

$$\text{“training set”} = \left\{ \left(\mathbf{x}_i = \begin{bmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_d^i \end{bmatrix} \in \mathbb{R}^d, \ \mathbf{y}_i \in \{0, 1\} \right)_{i=1..N} \right\}$$

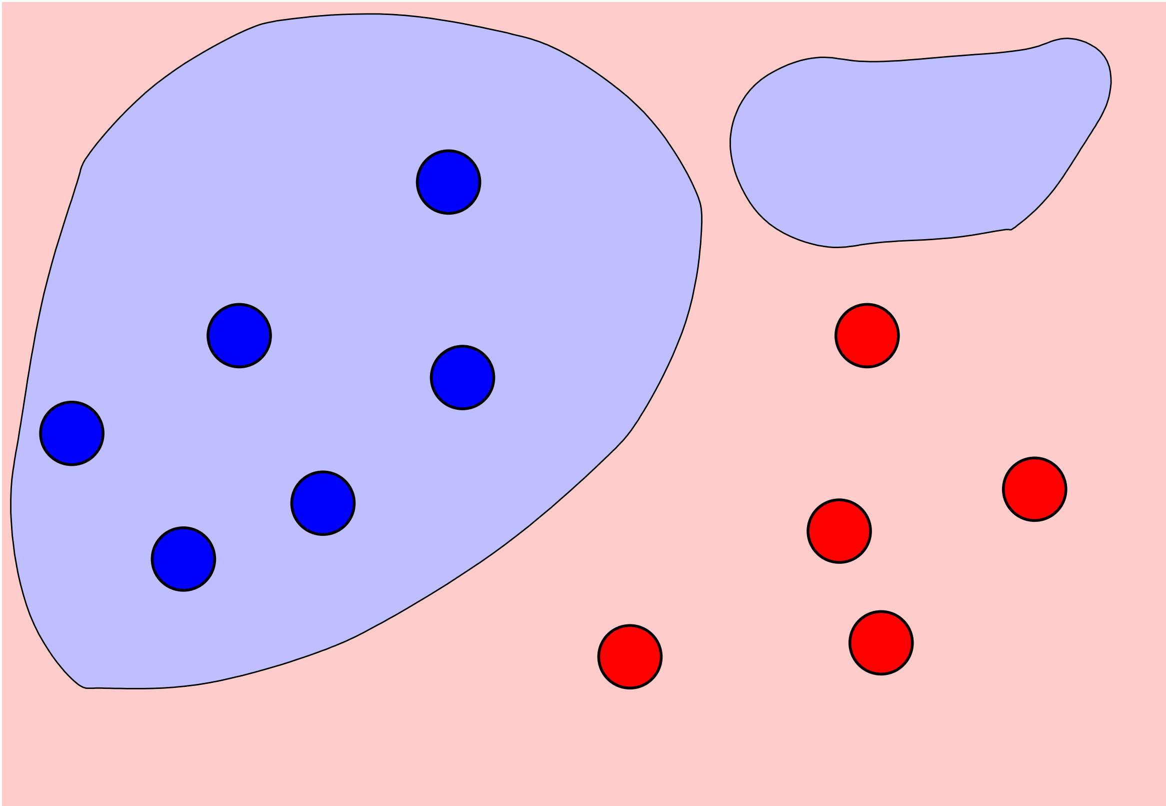
- For illustration purposes **only** we will consider **vectors in the plane**, $d = 2$.
- Points are easier to represent in 2 dimensions than in 20.000...
- The ideas for $d \gg 3$ are **exactly the same**.

Binary Classification Separation Surfaces for Vectors



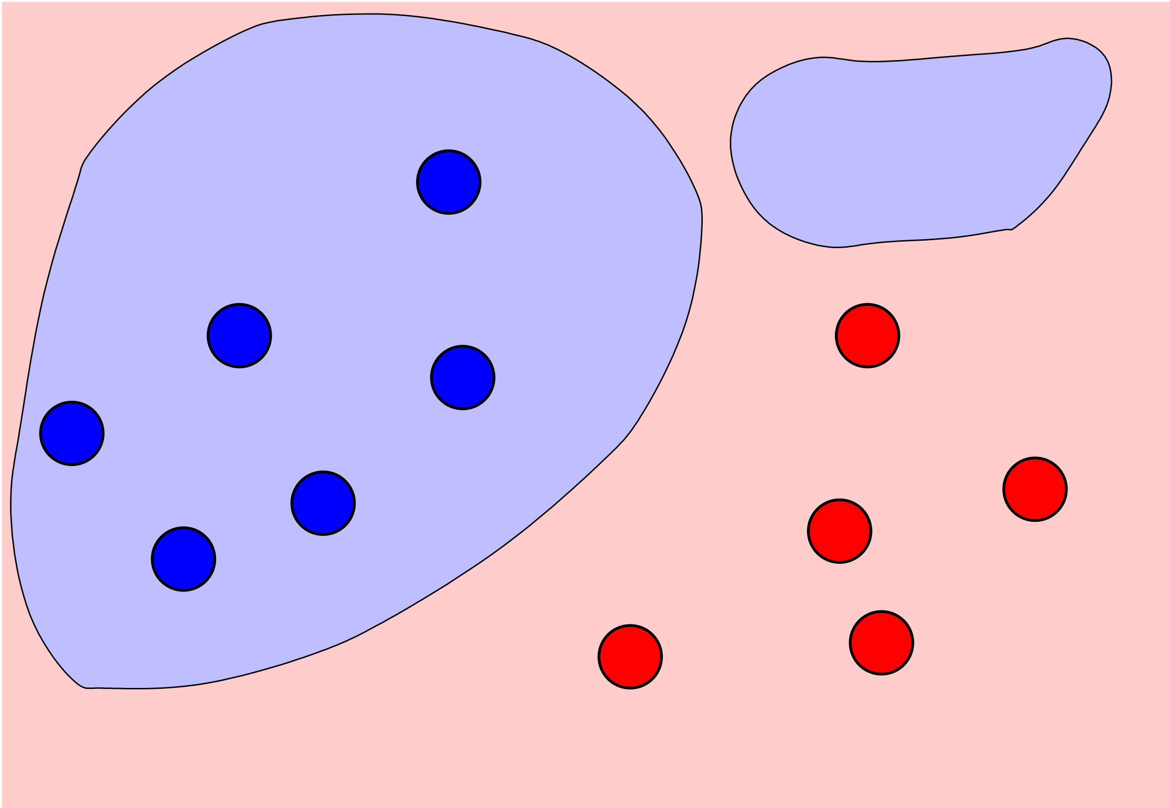
What is a classification rule?

Binary Classification Separation Surfaces for Vectors



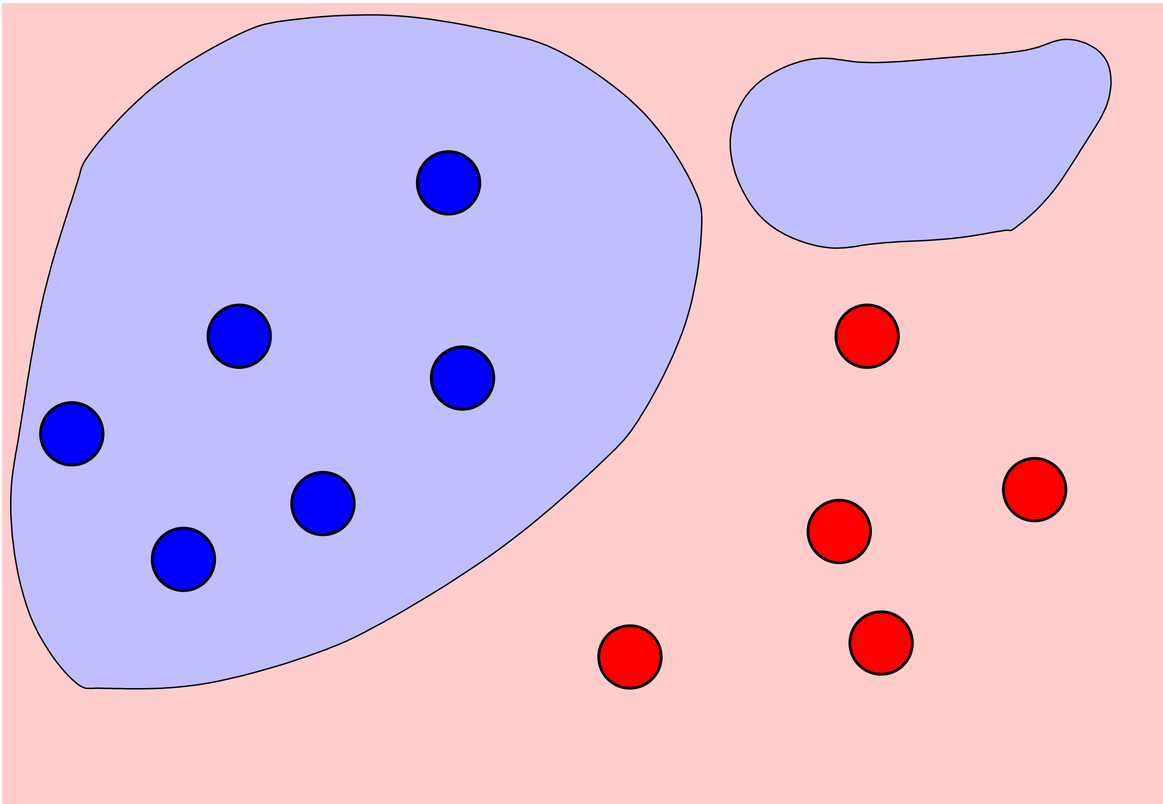
Classification rule = a partition of $\mathbb{R}^d$ into two sets

Binary Classification Separation Surfaces for Vectors



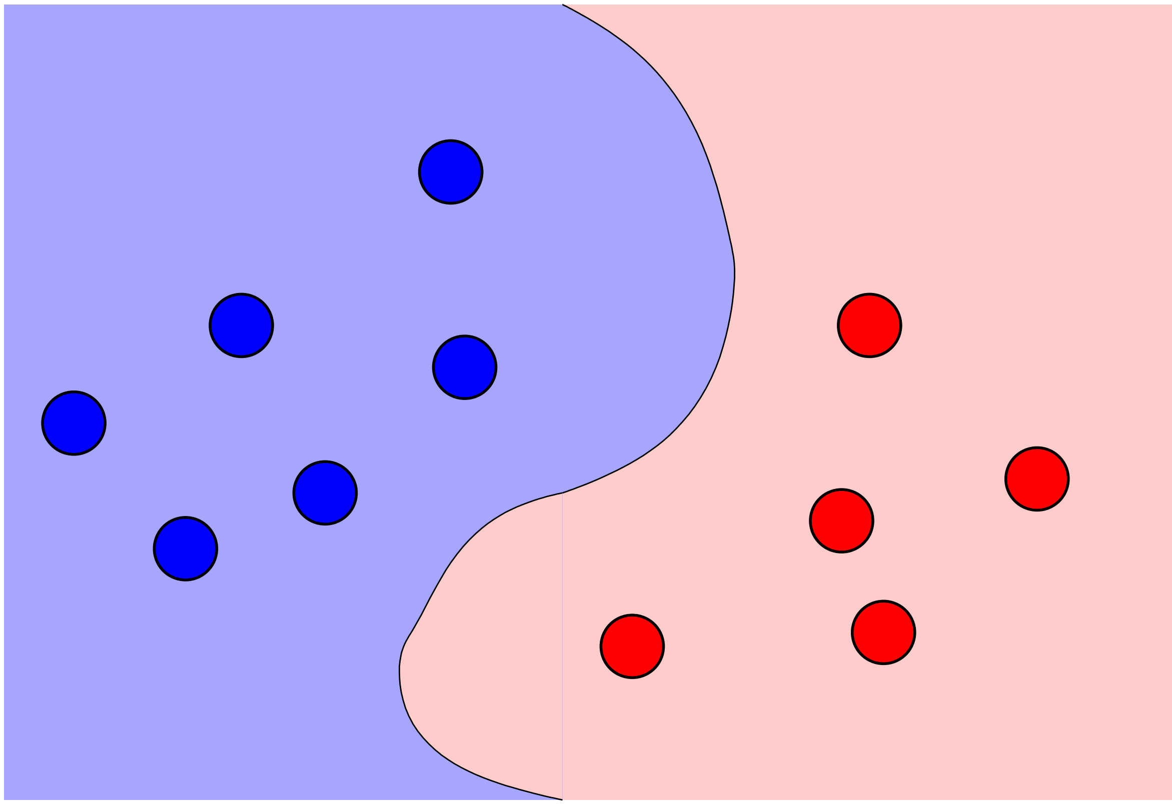
This partition is usually interpreted as the level set of function on $\mathbb{R}^d$

Binary Classification Separation Surfaces for Vectors



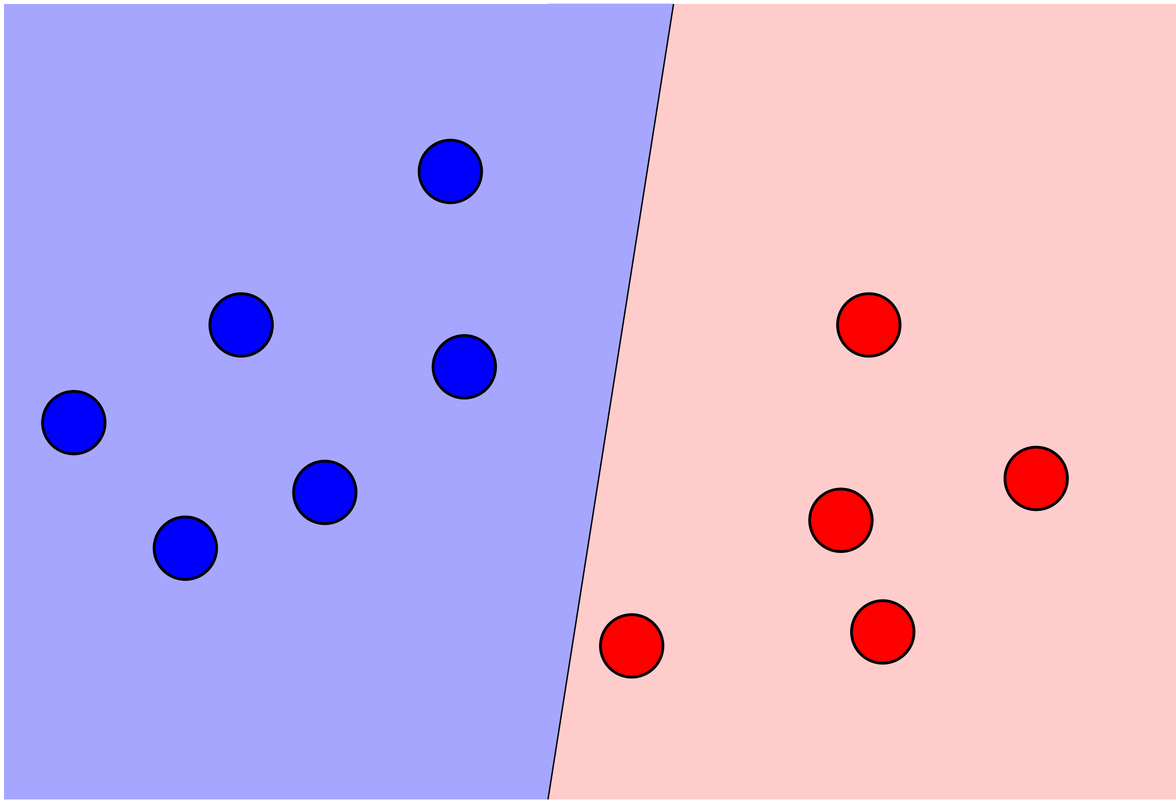
Typically, $\{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) > 0\}$ and $\{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) \leq 0\}$

Classification Separation Surfaces for Vectors



Can be defined by a single surface, *e.g.* a curved line

Classification Separation Surfaces for Vectors



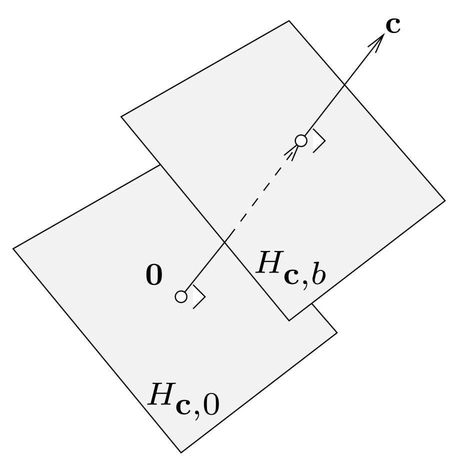
Even more **simple**: using **straight lines** and halfspaces.

Linear Classifiers

- **Straight lines** (hyperplanes when $d > 2$) are **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 $\mathbb{R}^d$

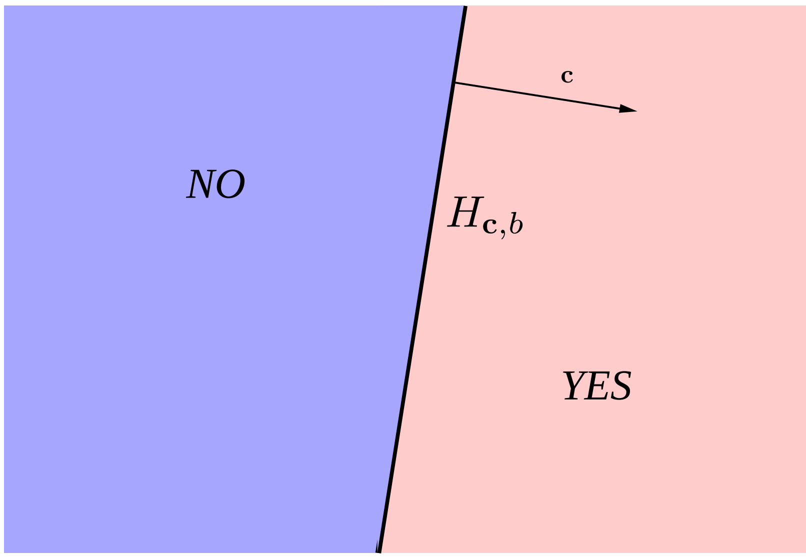


Linear Classifiers

- Exactly like lines in the plane, hypersurfaces **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 attribute the "yes" and "no" answers given arbitrary $\mathbf{c}$ and b.



- Assuming we only look at halfspaces for the decision surface...

  ...*how to **choose** the **"best"** $(\mathbf{c}^\star, b^\star)$ given a **training sample**?*

Linear Classifiers

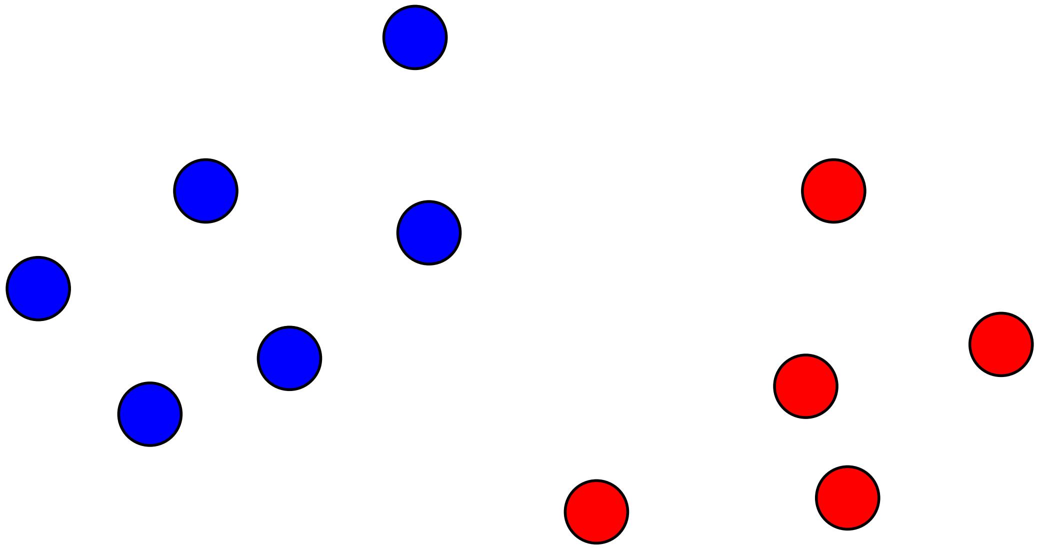
- This specific question,

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

  has different answers. Depends on the meaning of “best” **?**:

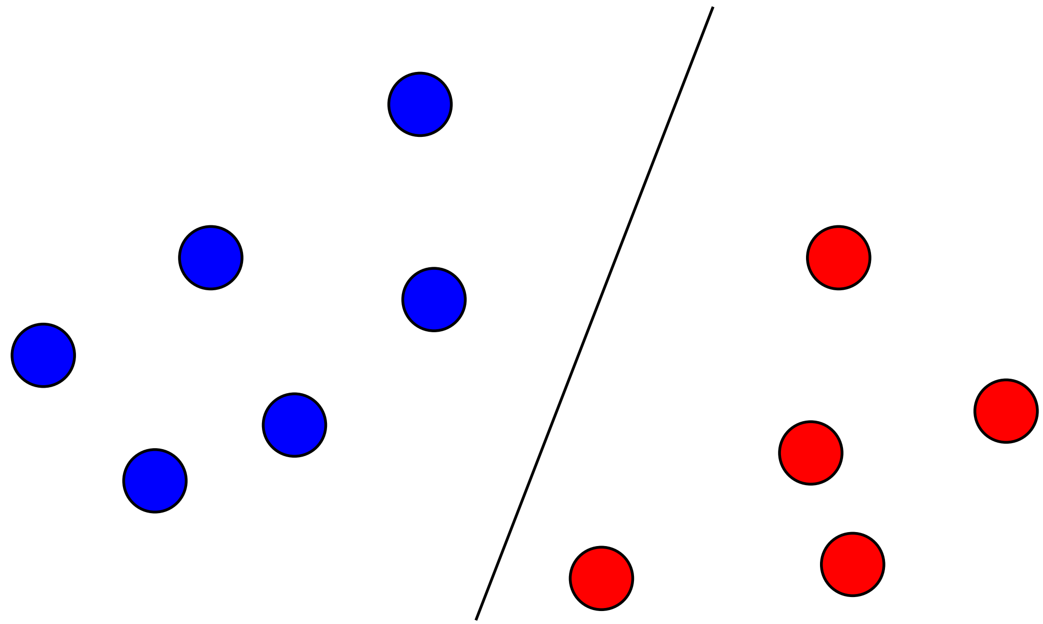
- **Linear Discriminant Analysis** (or Fisher's Linear Discriminant);
- **Logistic regression** maximum likelihood estimation;
- **Perceptron**, a one-layer neural network;
- **Support Vector Machine**, the result of a convex program
- *etc.*

Classification Separation Surfaces for Vectors



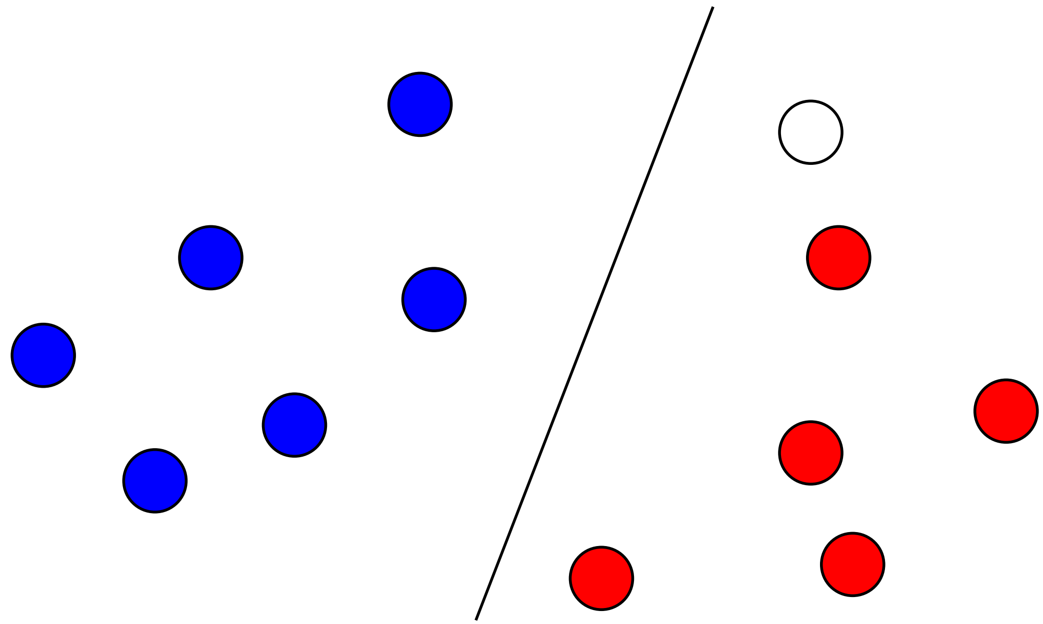
Given two sets of points...

Classification Separation Surfaces for Vectors



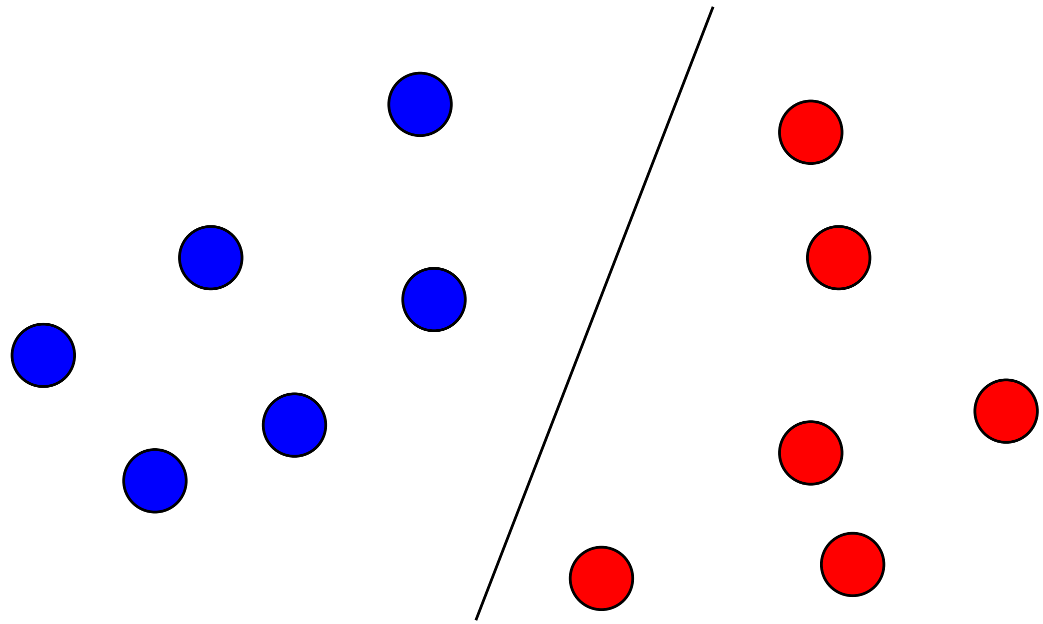
It is sometimes possible to separate them perfectly

Classification Separation Surfaces for Vectors

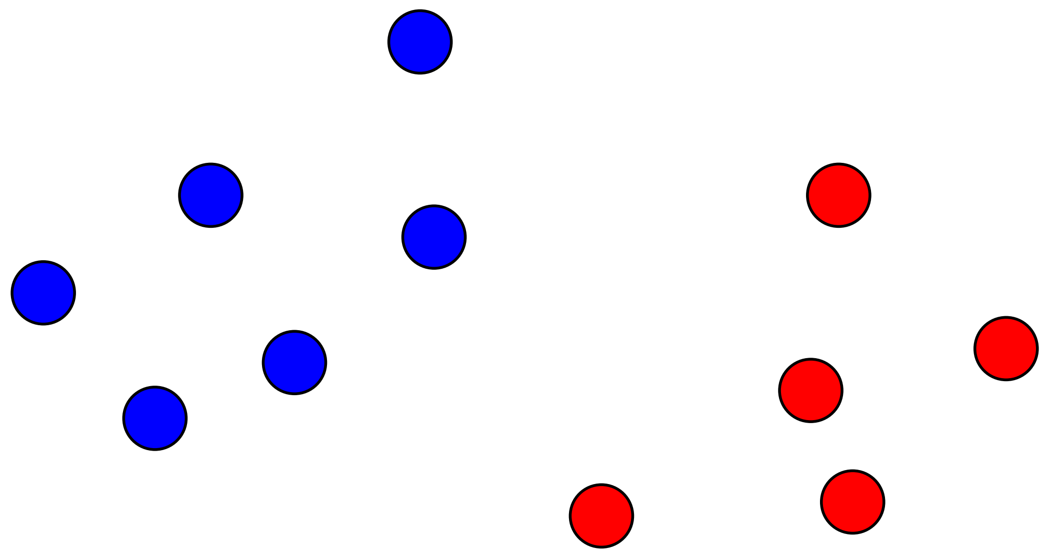


Each choice might look equivalently good on the training set,
but it will have obvious impact on new points

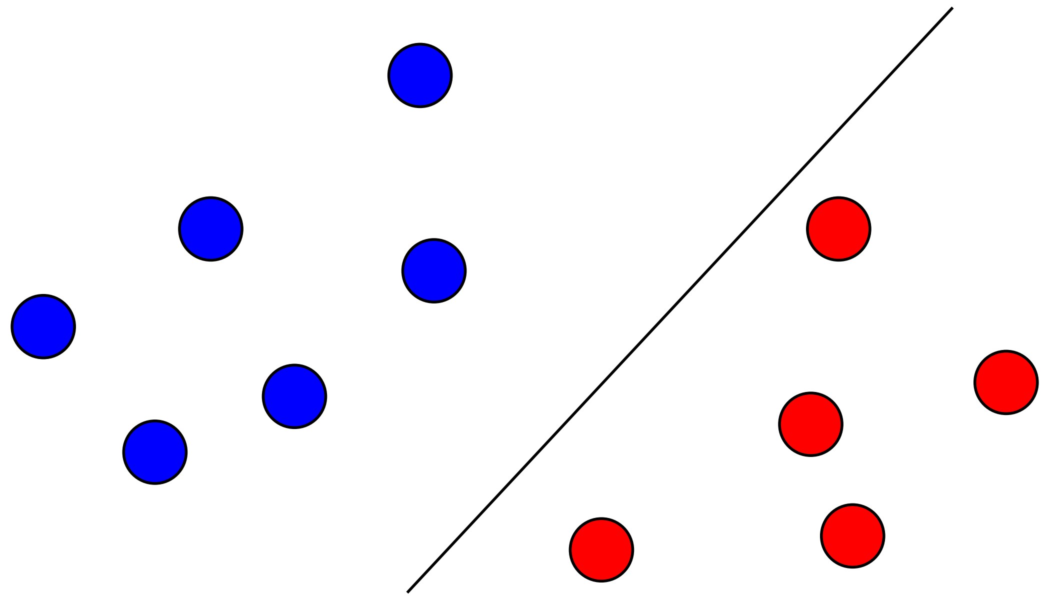
Classification Separation Surfaces for Vectors



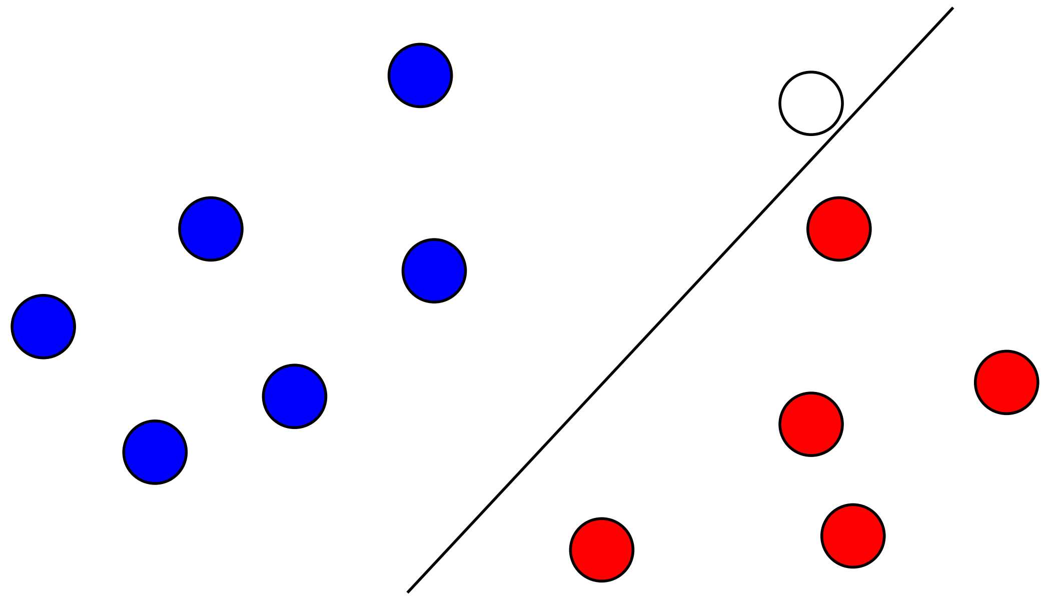
Linear classifier, some degrees of freedom



Linear classifier, some degrees of freedom

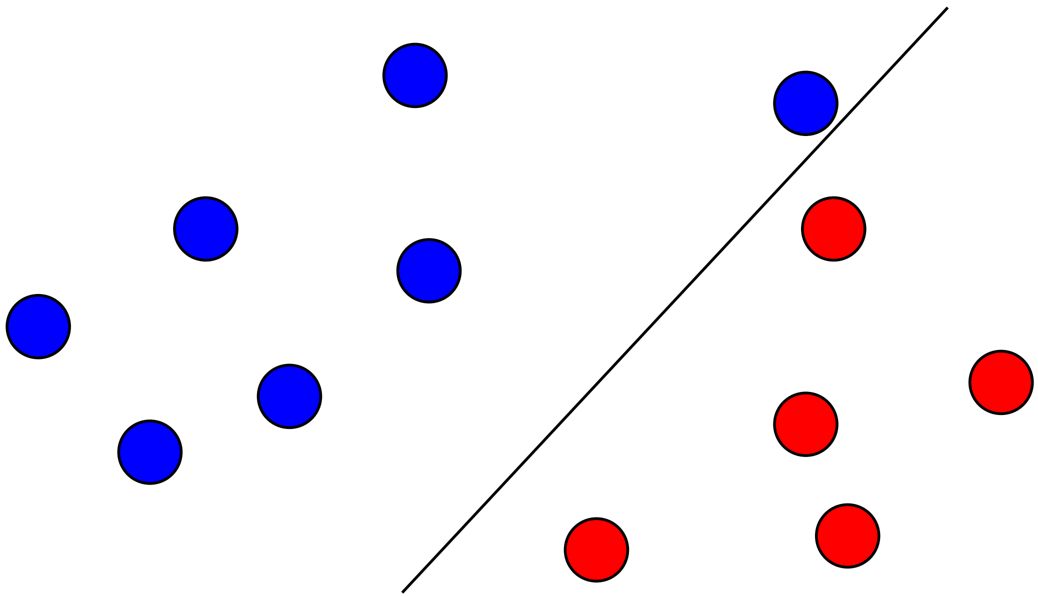


Linear classifier, some degrees of freedom

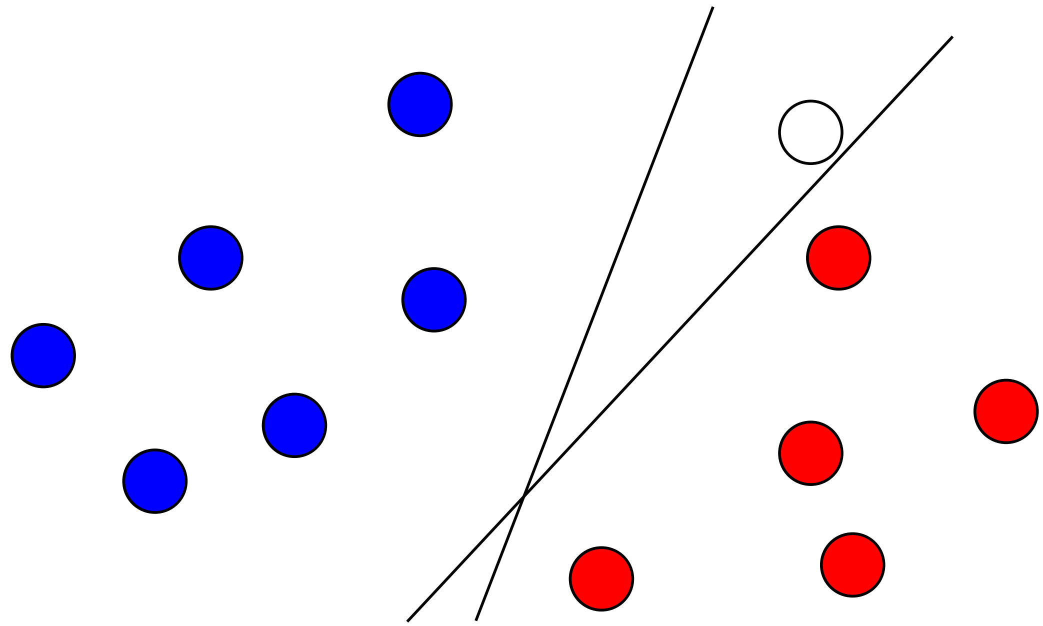


Specially close to the border of the classifier

Linear classifier, some degrees of freedom



Linear classifier, some degrees of freedom



For each different technique, different results, different performance.

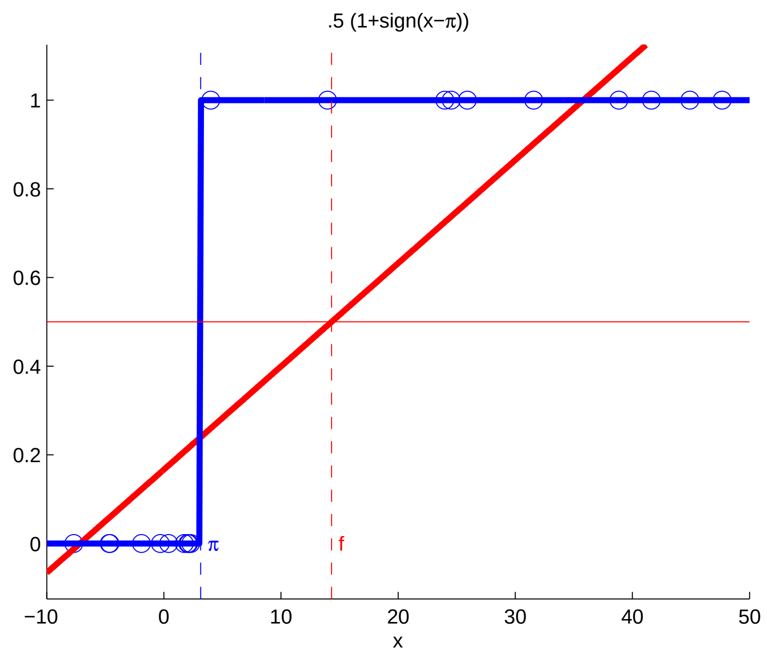
A few linear classifiers: Logistic Regression

Regression does not work

- Consider the toy classification example:
  - Points $\mathbf{x}_j$ are taken randomly between -10 and 50.
  - The label

$$y_j = \begin{cases} 0 \text{ if } \mathbf{x}_j < \pi, \\ 1 \text{ if } \mathbf{x}_j > \pi. \end{cases}$$

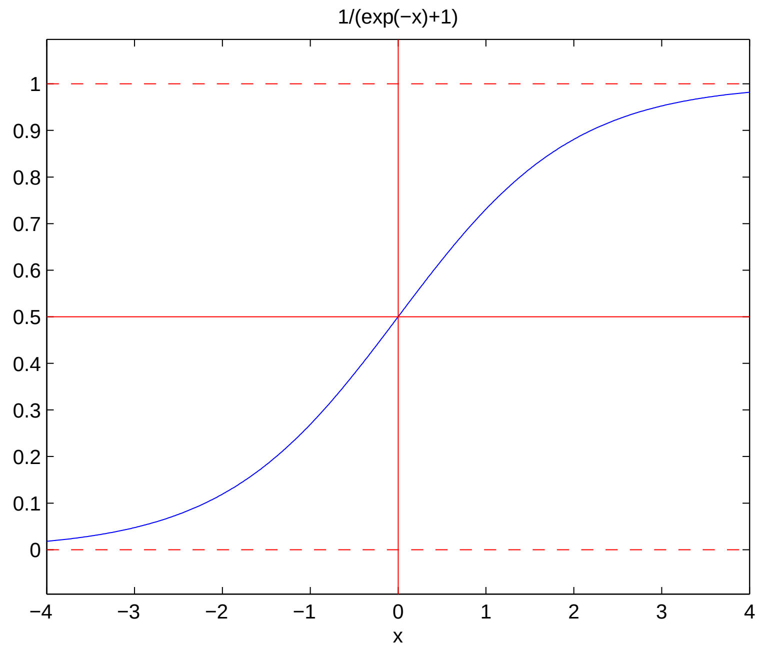
- What happens if we feed this directly to regression?... **matlab demo**



How can we adapt regression? logistic map

- Logistic map :

$$g(z) = \frac{e^z}{e^z + 1} = \frac{1}{e^{-z} + 1}$$



  - for any z, $0 \leq g(z) \leq 1$

How can we adapt regression? logistic map

Basic Idea

- Rather than find the best $\mathbf{c}$ and b such that

$$f(\mathbf{x}_j) = \mathbf{c}^T \mathbf{x}_j + b \quad \approx \quad y_j \in \{0, 1\}$$

- logistic regression considers instead the best $\mathbf{c}$ and b such that

$$g \circ f(\mathbf{x}_j) = \frac{1}{e^{-(\mathbf{c}^T \mathbf{x}_j + b)} + 1} \approx y_j \in \{0, 1\}.$$

- if for a new point $\mathbf{x}$,
  - $g \circ f(\mathbf{x}) > 1/2$, guess that the class is 1
  - $g \circ f(\mathbf{x}) < 1/2$, guess that the class is 0

Probabilistic Interpretation of Logistic Regression

- Suppose there is a probability density $\boldsymbol{p}(X, Y)$ on couples $(\mathbf{x}, y) \in \mathbb{R}^d \times \{0, 1\}$.
- Suppose for now that we **know** $\boldsymbol{p}$.
- The ratio

$$r(\mathbf{x}) = \frac{p(Y = 1 | X = \mathbf{x})}{p(Y = 0 | X = \mathbf{x})}$$

  is called the odds-ratio of a given point **x**.

- Obviously,
  - if $r(\mathbf{x}) > 1$, then it is more likely that $y = 1$ than $y = 0$.
  - if $r(\mathbf{x}) < 1$, then one is tempted to guess that $y = 0$ than $y = 1$.

Probabilistic Interpretation of Logistic Regression

- In other words...

$$\log \frac{p(Y=1|X=\mathbf{x})}{p(Y=0|X=\mathbf{x})}, \quad \begin{cases} >0 \text{ then } y=1 \text{ is the likely answer} \\ <0 \text{ then } y=0 \text{ is the likely answer} \end{cases}$$

- Logistic regression **assumes** that the log-odds ratio follows a **linear** relationship

$$\log \frac{p(Y=1|X=\mathbf{x})}{p(Y=0|X=\mathbf{x})} \approx \mathbf{c}^T\mathbf{x} + b$$

- This implies that the decision surface is **linear**.

> Note that Logistic Regression
> assumes a **model only for the log-odds ratio**,
> **not for the whole probability** p

Probabilistic Interpretation of Logistic Regression

- Since $p(Y = 0|X = \mathbf{x}) = 1 - p(Y = 1|X = \mathbf{x})$, we hence have

$$\log \frac{p(Y = 1|X = \mathbf{x})}{1 - p(Y = 1|X = \mathbf{x})} = \mathbf{c}^T \mathbf{x} + b$$

- which in turn implies

$$p(Y = 1|X = \mathbf{x}) = \frac{1}{e^{-(\mathbf{c}^T \mathbf{x} + b)} + 1} = g(\mathbf{c}^T \mathbf{x} + b).$$

Predictor variables contribute **linearly**
to the increase/decrease of the probability that $y = 1$.

Estimation of $\mathbf{c}$ and b through Maximum Likelihood

- Flip coin, setting $p(y=1)=p$ and $p(y=0)=1-p$ for binary random variable y,
  - Likelihood of a draw y knowing that probability is p,

$$p^y(1-p)^{1-y}$$

- In the context of **logistic regresion**, p depends on $\mathbf{c}, b$ and $\mathbf{x}_j$ for each point,

$$\mathcal{L}(\mathbf{c}, b) = \prod_{j=1}^{N} g(\mathbf{c}^T\mathbf{x}_j + b)^{y_j}(1 - g(\mathbf{c}^T\mathbf{x}_j + b))^{1-y_j}$$

Estimation of $\mathbf{c}$ and b through Maximum Likelihood

- Using again the log transformation,

$$\log \mathcal{L}(\mathbf{c}, b) = \sum_{j=1}^{N} y_j \log g(\mathbf{c}^T \mathbf{x}_j + b) + (1 - y_j) \log g(1 - (\mathbf{c}^T \mathbf{x}_j + b)).$$

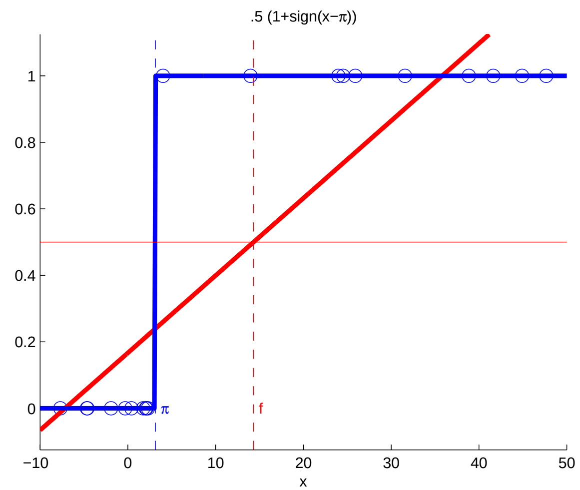
- Maximizing this log-likelihood is equivalent to

$$\max_{\mathbf{c}, b} \log \mathcal{L}(\mathbf{c}, b) \Leftrightarrow \max_{\mathbf{c}, b} \sum_{j=1}^{N} y_j (\mathbf{c}^T \mathbf{x}_j + b) - \log(1 + e^{\mathbf{c}^T \mathbf{x}_j + b}).$$

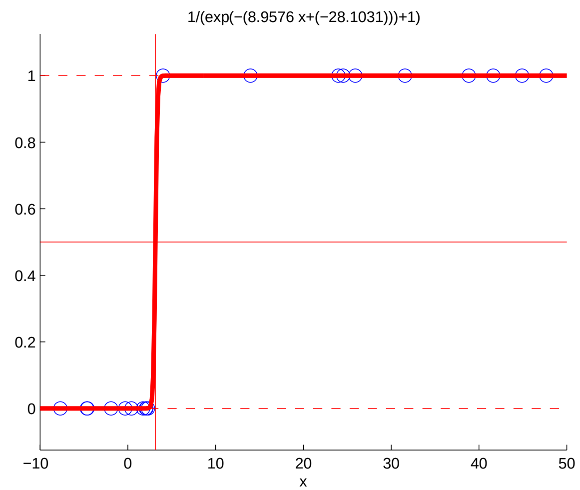
- No closed form solution for this unfortunately... need efficient optimization.
- For datasets of reasonable size, Newton method for instance.

Estimation of c and b through Maximum Likelihood

Compare...



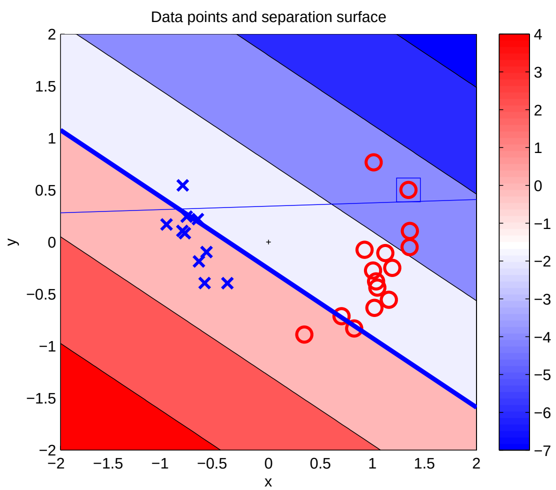
...with



A few linear classifiers: Perceptron Rule

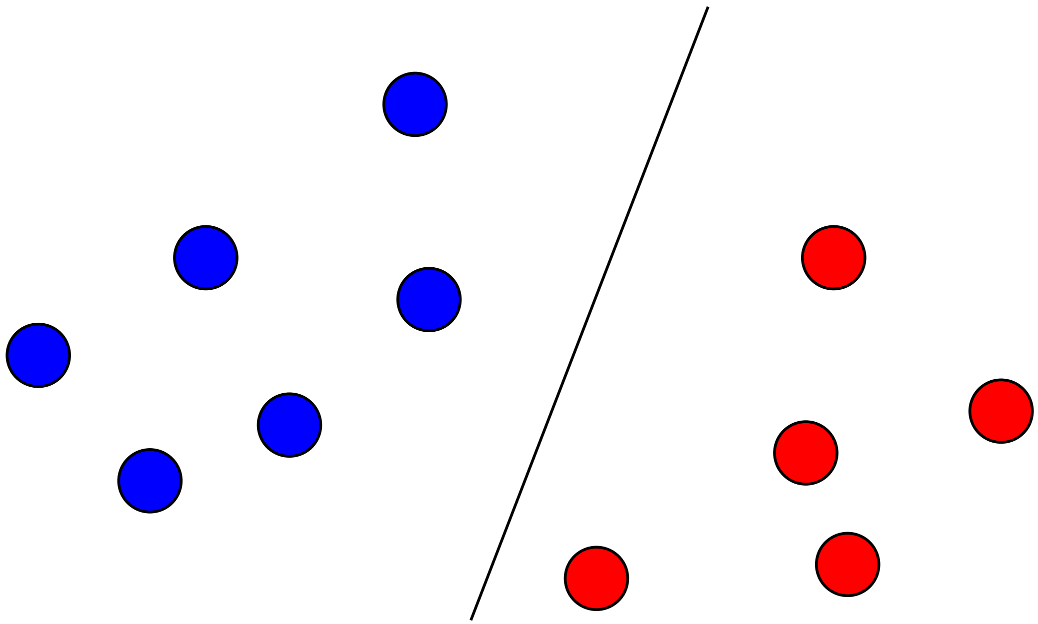
Estimation of $\mathbf{c}$ and b through iterative updates

- Iterative algorithm that considers each point successively.
- Here we consider $\mathcal{S} = \{-1, 1\}$
- Start from any arbitrary estimate $\omega = \begin{bmatrix} b \\ \mathbf{c} \end{bmatrix}$.
- Loop over j until ω does not change for a while...
  - Consider a point $\begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$ and his label y_j.
  - Do $u_j = \text{sign}(\omega^T \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix})$ and y_j match?
  - it not, set $\omega \leftarrow \omega + \rho(y_j - u_j) \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$.
- Not much more to add, better see in practice.

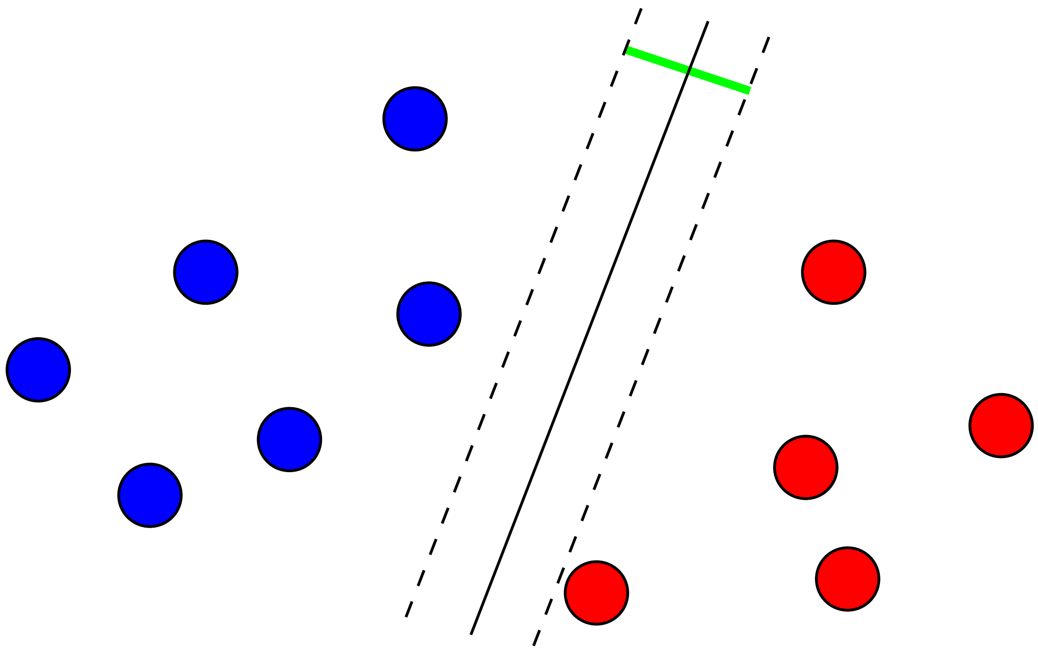


A few linear classifiers: Support Vector Machine

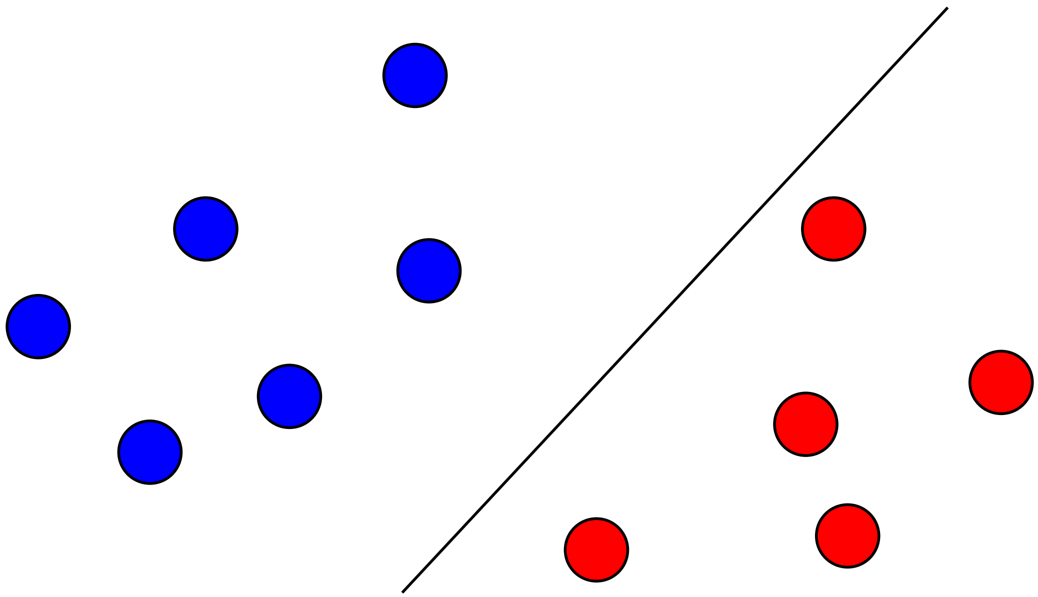
A criterion to select a linear classifier: the margin ?



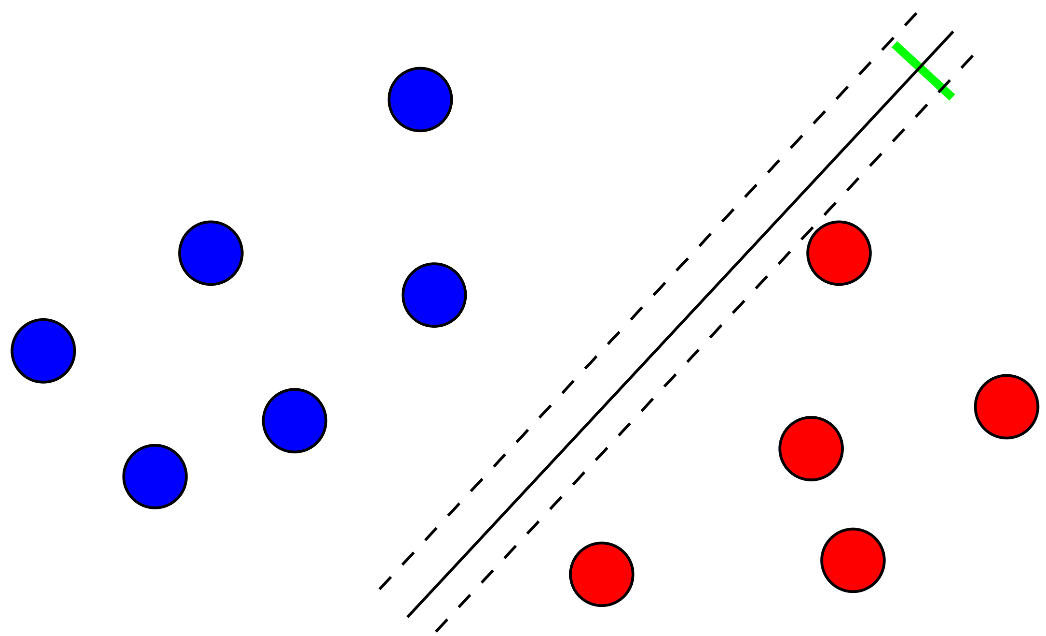
A criterion to select a linear classifier: the margin ?



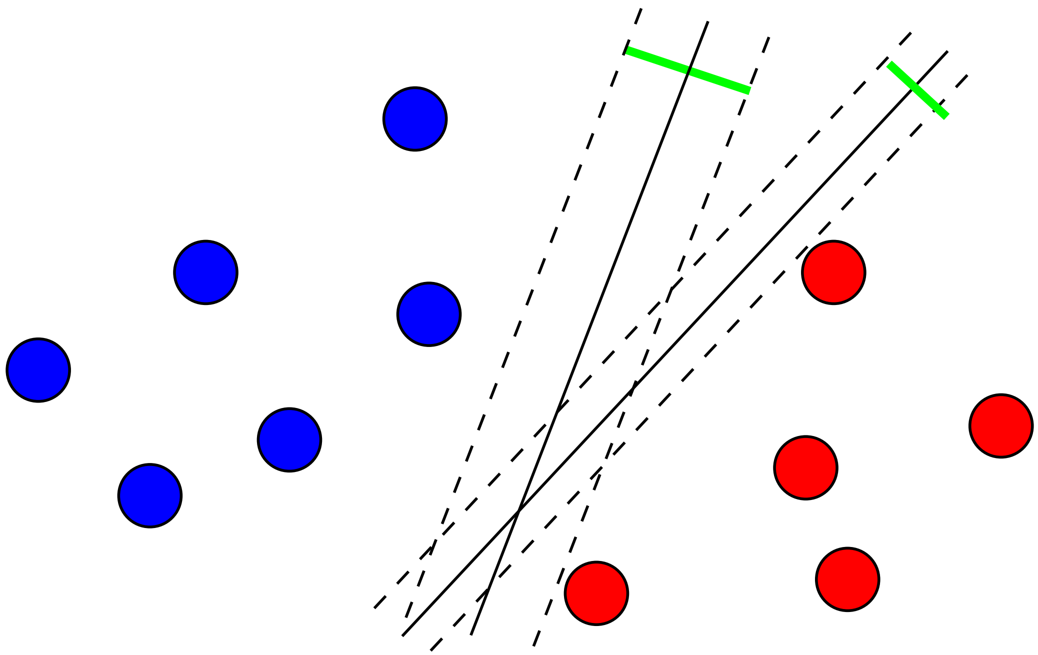
A criterion to select a linear classifier: the margin ?



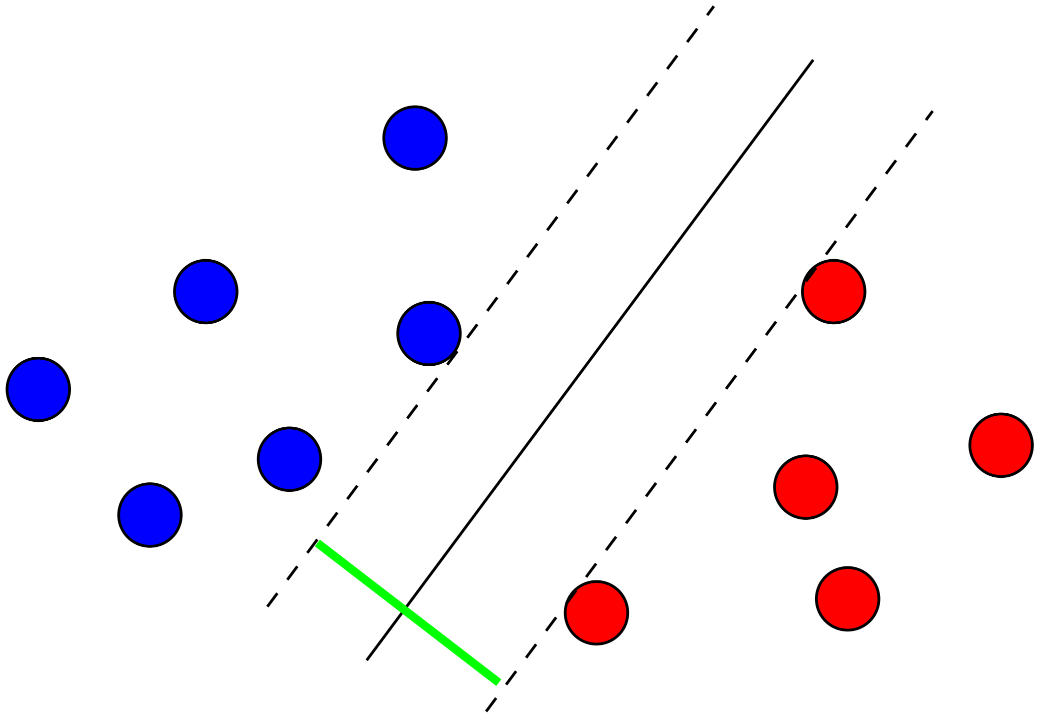
A criterion to select a linear classifier: the margin ?



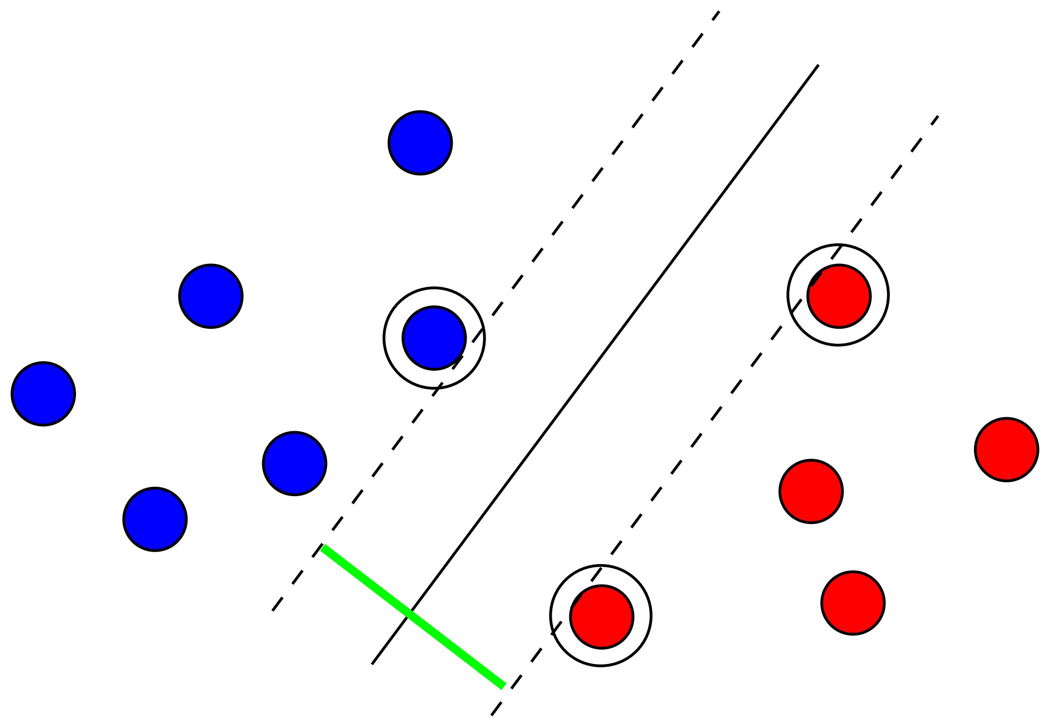
A criterion to select a linear classifier: the margin ?



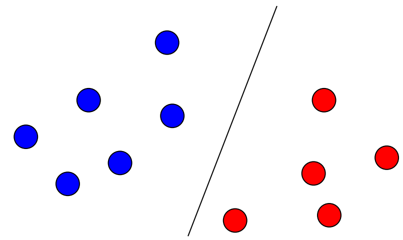
Largest Margin Linear Classifier ?



Support Vectors with Large Margin



In equations



- 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:

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

- Next, we give a formula to compute the margin as a function of $\mathbf{w}$.
- Obviously, for any $t \in \mathbb{R}$,

$$H_{\mathbf{w},b} = H_{t\mathbf{w},tb}$$

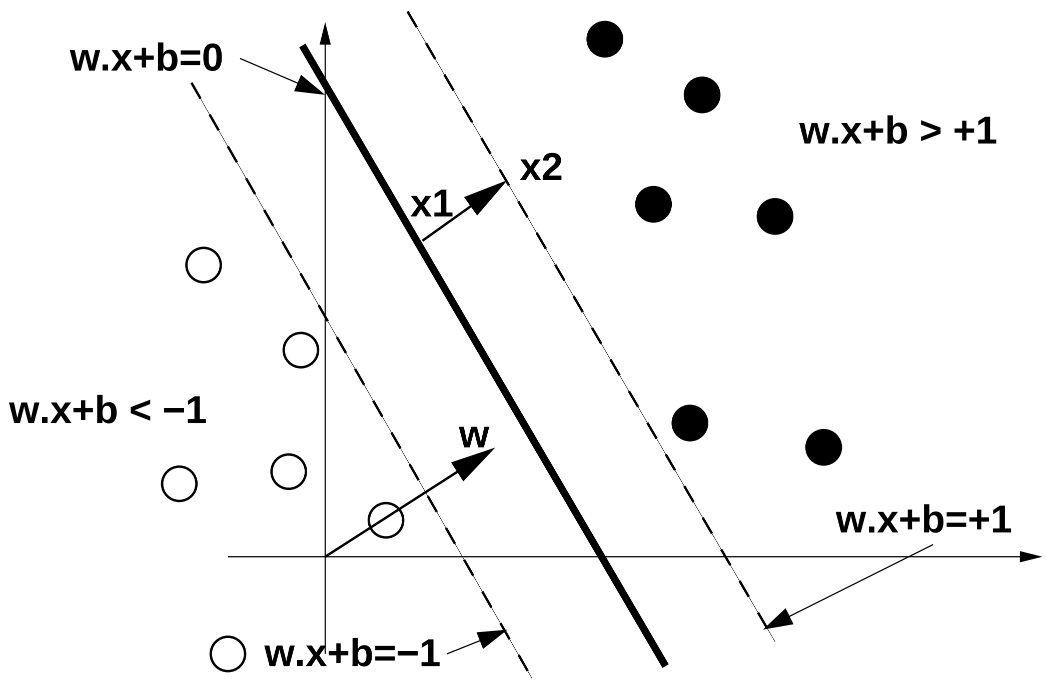
- Thus $\mathbf{w}$ and b are defined up to a multiplicative constant.
- We need to take care of this in the definition of the margin

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:

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



- Consider $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{x}_2 - \mathbf{x}_1$ is parallel to $\mathbf{w}$.

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

- Margin $= 2/\|\mathbf{w}\|$: 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:

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

where γ is by definition the **margin**.

All training points should be on the appropriate side

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

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

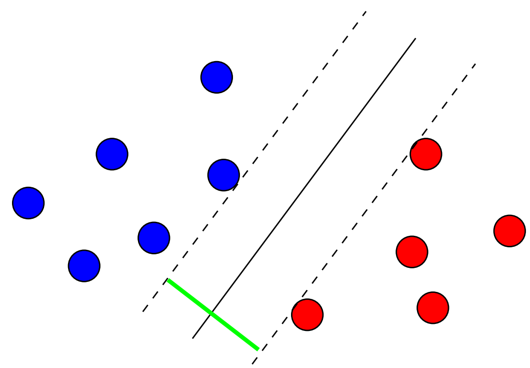
- For negative examples ($y_i = -1$) this means:

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

- 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



- Find $(\mathbf{w}, b)$ 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

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

is only defined when

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

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

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. 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 $\mathbf{y_i \left(w^T x_i + b\right) = 1}$.
- $\alpha_i \neq 0$ **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 $\alpha_i > 0$, $\mathbf{y}_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$,

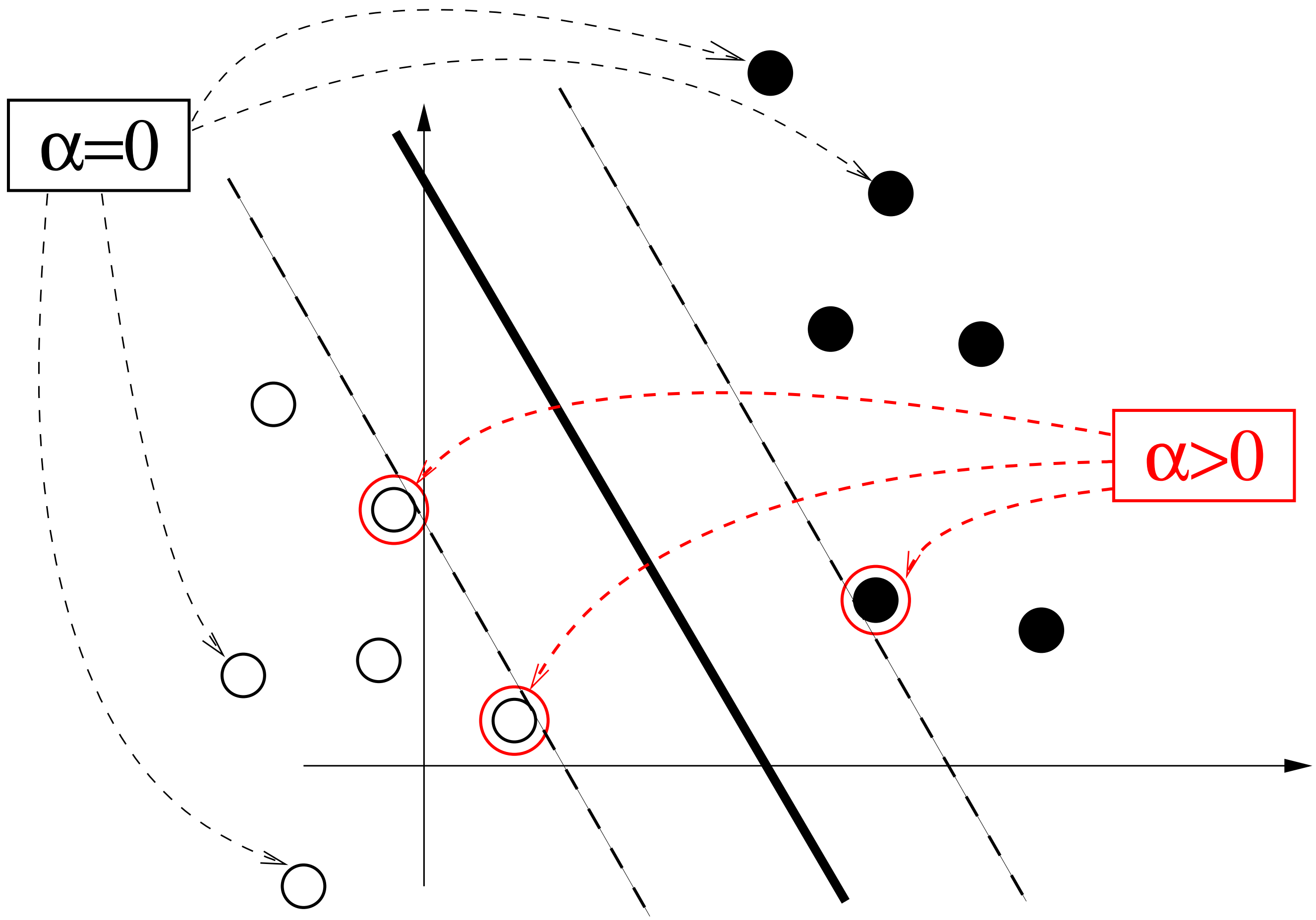
$$b^* = -\frac{1}{2}\left(\min_{\mathbf{y}_i=1,\alpha_i>0} (\mathbf{w}^T \mathbf{x}_i) + \max_{\mathbf{y}_i=-1,\alpha_i>0} (\mathbf{w}^T \mathbf{x}_i) \right)$$

- the **decision function** is therefore:

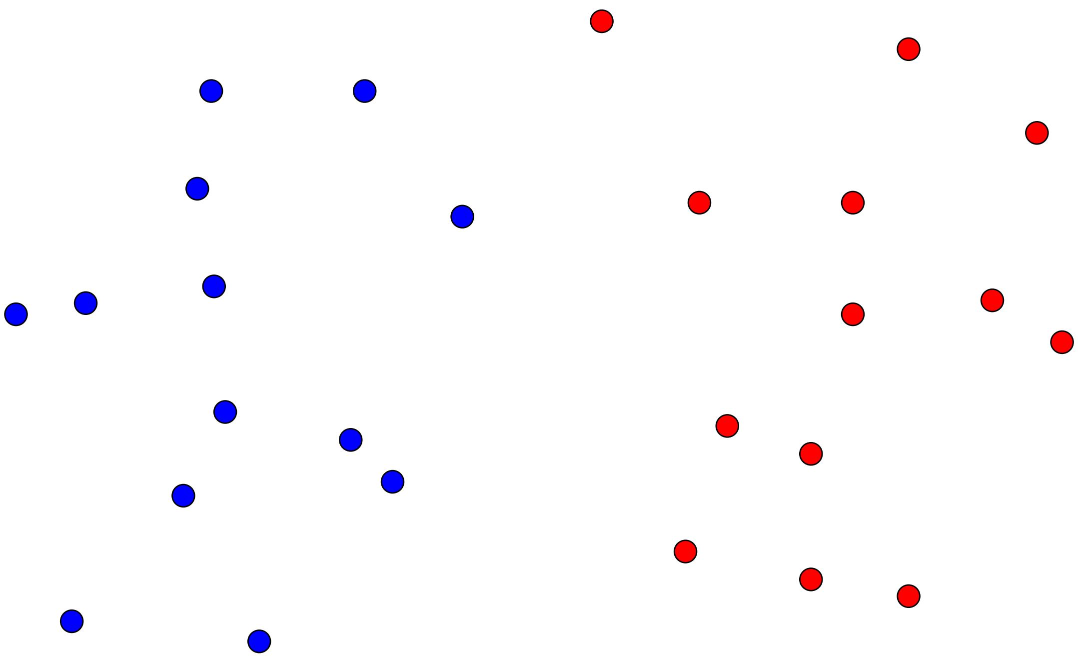
$$\begin{aligned} f^*(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b^* \\ &= \sum_{i=1}^n y_i \alpha_i \mathbf{x}_i^T \mathbf{x} + b^*. \end{aligned}$$

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

Interpretation: support vectors

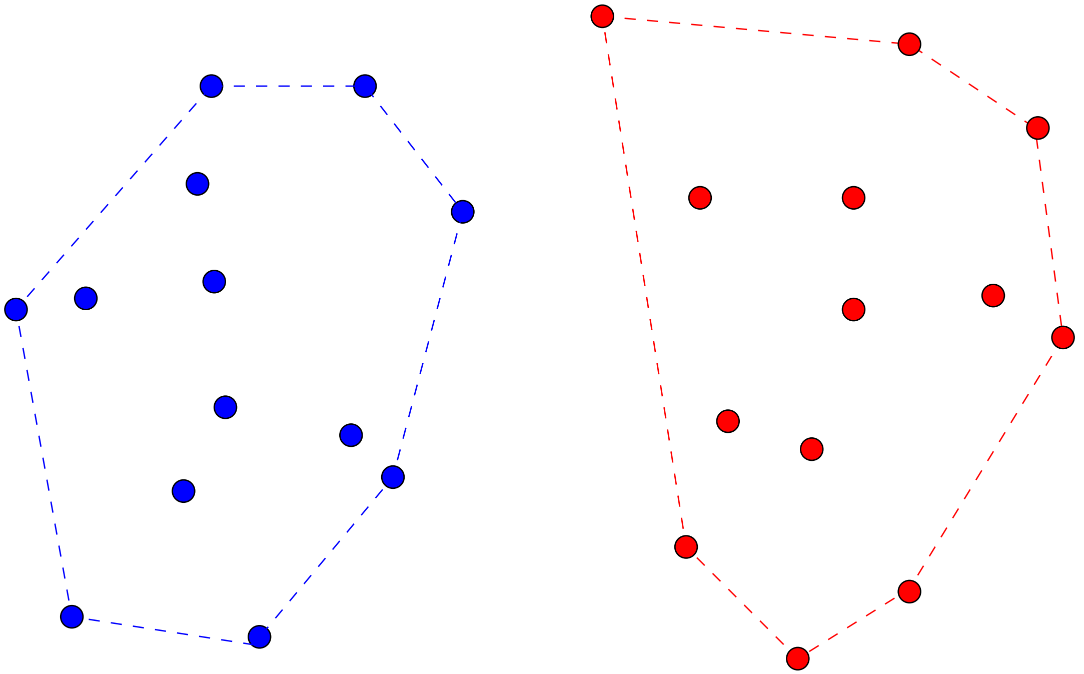


Another interpretation: Convex Hulls ?



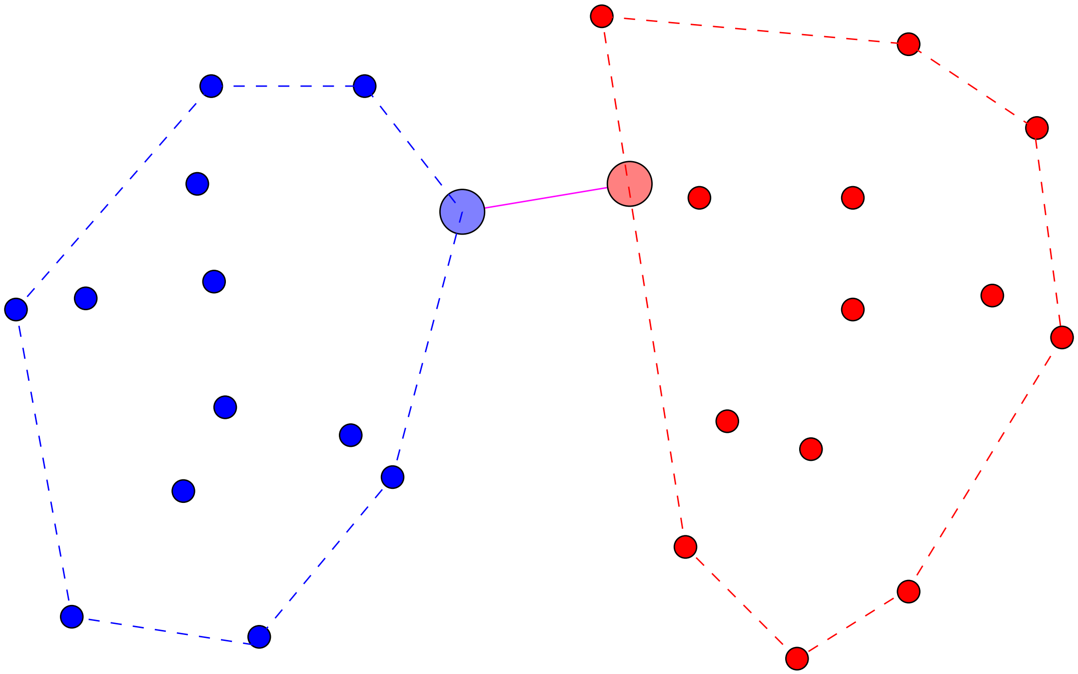
go back to 2 sets of points that are linearly separable

Another interpretation: Convex Hulls



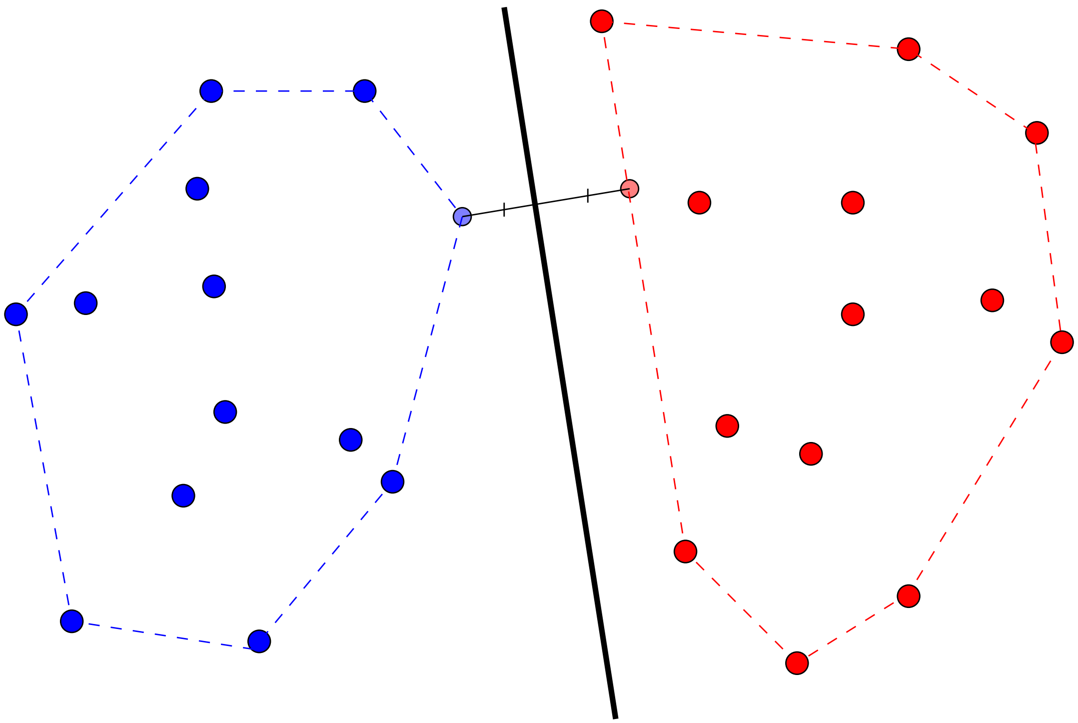
Linearly separable = convex hulls do not intersect

Another interpretation: Convex Hulls



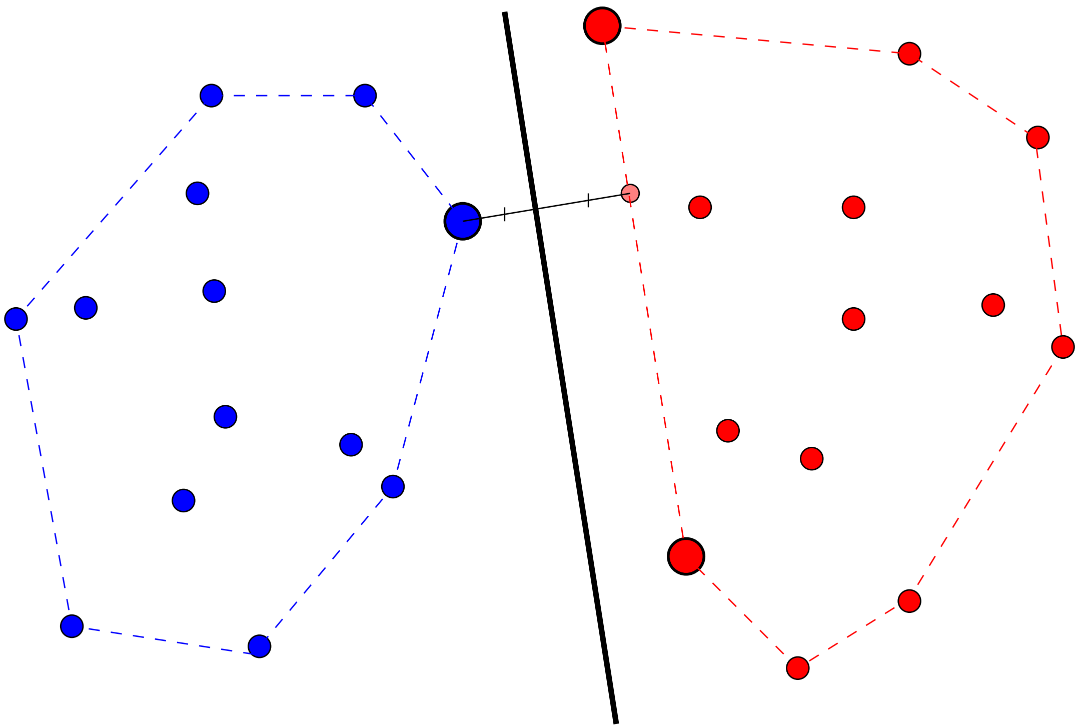
Find two closest points, one in each convex hull

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

Minimum Distance

- Suppose that
  - all the points of the blue set are in a matrix $A \in \mathbb{R}^{d\times n_{-1}}$,
  - all the points of the red set are in a matrix $B \in \mathbb{R}^{d\times n_1}$

$$\boldsymbol{A} = \begin{bmatrix} \vdots & \cdots & \vdots \\ \boldsymbol{x_1} & \cdots & \boldsymbol{x_{n_{-1}}} \\ \vdots & \cdots & \vdots \end{bmatrix} \in \mathbb{R}^{\boldsymbol{d\times n_{-1}}}, \quad B = \begin{bmatrix} \vdots & \cdots & \vdots \\ x'_1 & \cdots & x'_{n_1} \\ \vdots & \cdots & \vdots \end{bmatrix} \in \mathbb{R}^{d\times n_1}.$$

- Finding the two points in question, and the minimal distance, is given by

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{A}\mathbf{u} - B\mathbf{v}\|^2 \\ \text{subject to} & 1_{n_{-1}}^T \mathbf{u} = 1_{n_1}^T \mathbf{v} = 1 \\ & 0 \leq \mathbf{u} \in \mathbb{R}^{n_{-1}}, \mathbf{v} \in \mathbb{R}^{n_1} \end{array}$$

- Possible to prove that the primal SVM program, slightly modified, has this dual.
- A bit tedious unfortunately.

A brief hint through duality

- Remember that the dual of the SVM formulation is

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

- Suppose that the n_{-1} first points x_i have -1 label and n_1 have 1 label.
- rewrite $\alpha = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}$. The dual becomes:

$$\begin{array}{ll} \text{maximize} & 1_{n_{-1}}\mathbf{u} + 1_{n_1}\mathbf{v} - \frac{1}{2}\left[-\mathbf{u}^T, \mathbf{v}^T\right]\begin{bmatrix}A^T\\ B^T\end{bmatrix}\left[A,\, B\right]\begin{bmatrix}-\mathbf{u}\\ \mathbf{v}\end{bmatrix} \\ \text{such that} & 1_{n_{-1}}^T\mathbf{u} = 1_{n_1}^T\mathbf{v}, \\ & \mathbf{u}, \mathbf{v} \geq 0 \end{array}$$

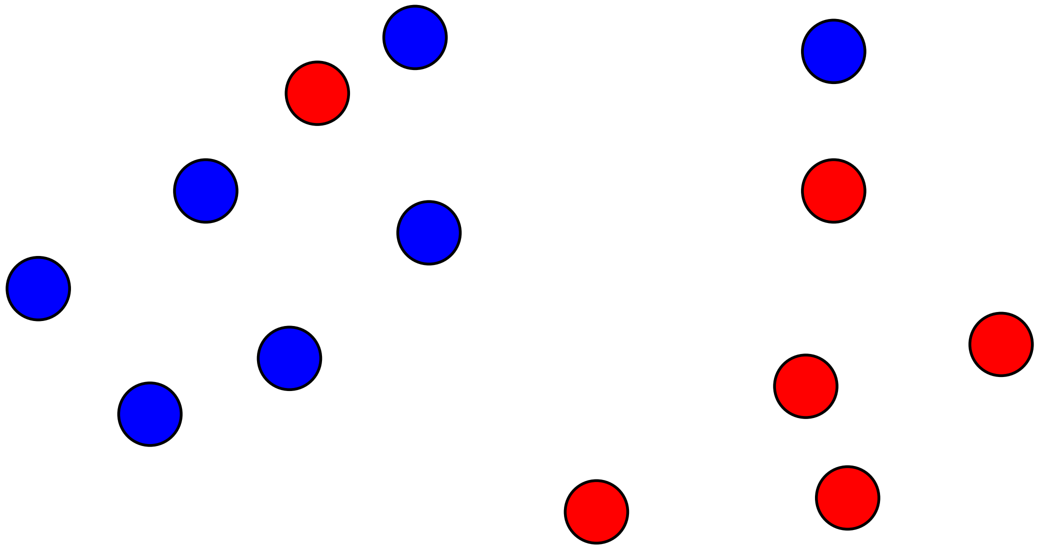
which is equivalent to

$$\begin{array}{ll} \text{minimize} & \|A\mathbf{u} - B\mathbf{v}\|^2 - 2(1_{n_{-1}}^T\mathbf{u} + 1_{n_1}^T\mathbf{v}) \\ \text{such that} & \mathbf{u}, \mathbf{v} \geq 0, \\ & 1_{n_{-1}}^T\mathbf{u} = 1_{n_1}^T\mathbf{v} \end{array}$$

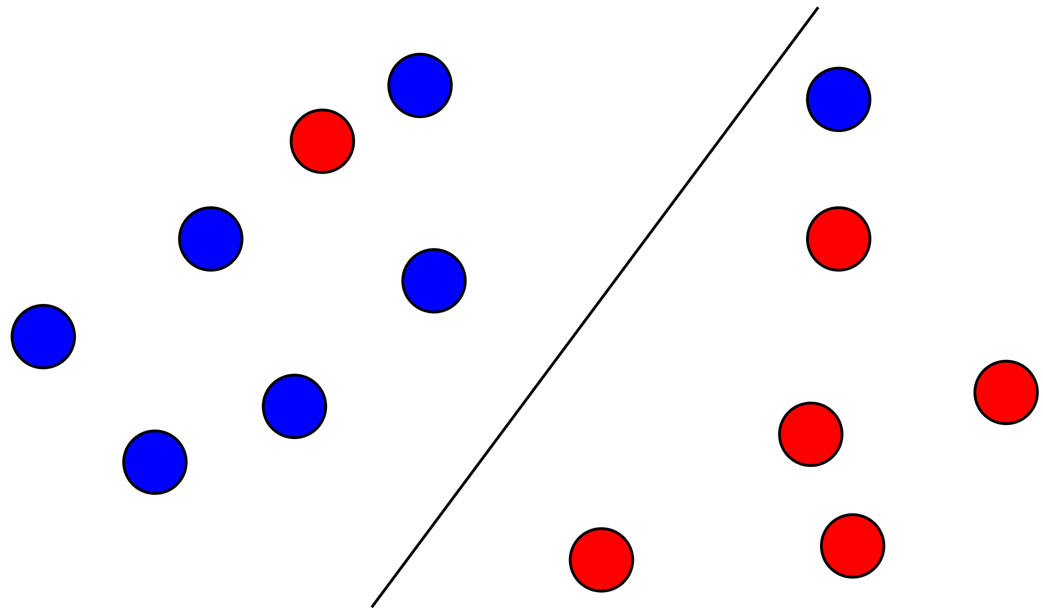
The non-linearly separable case

(when convex hulls intersect)

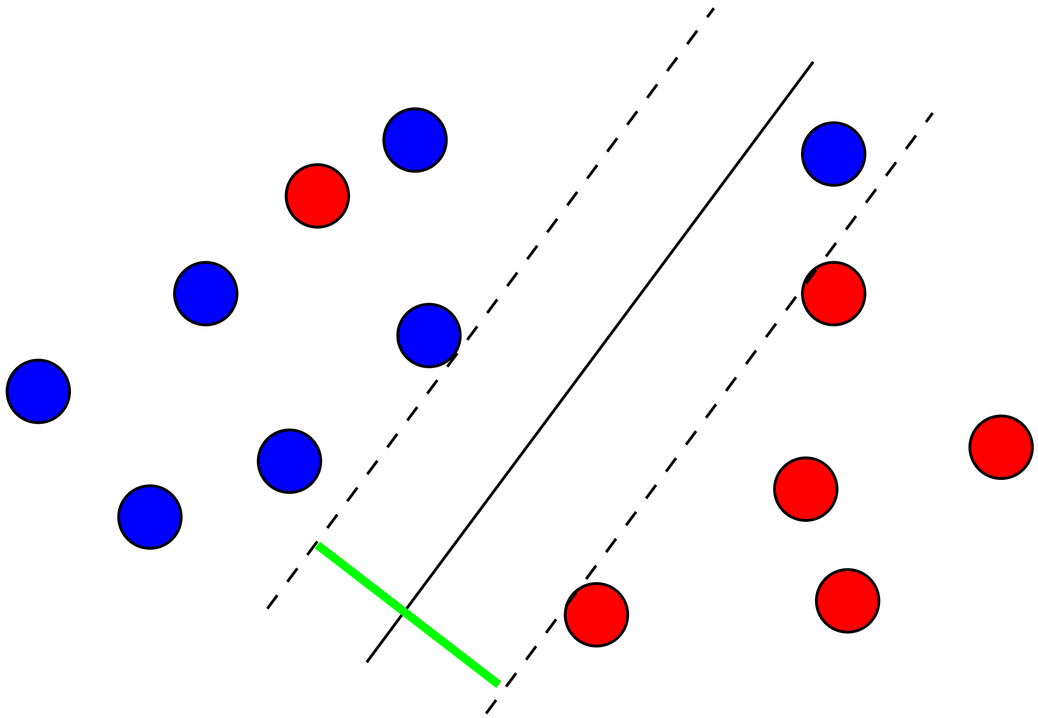
What happens when the data is not linearly separable?



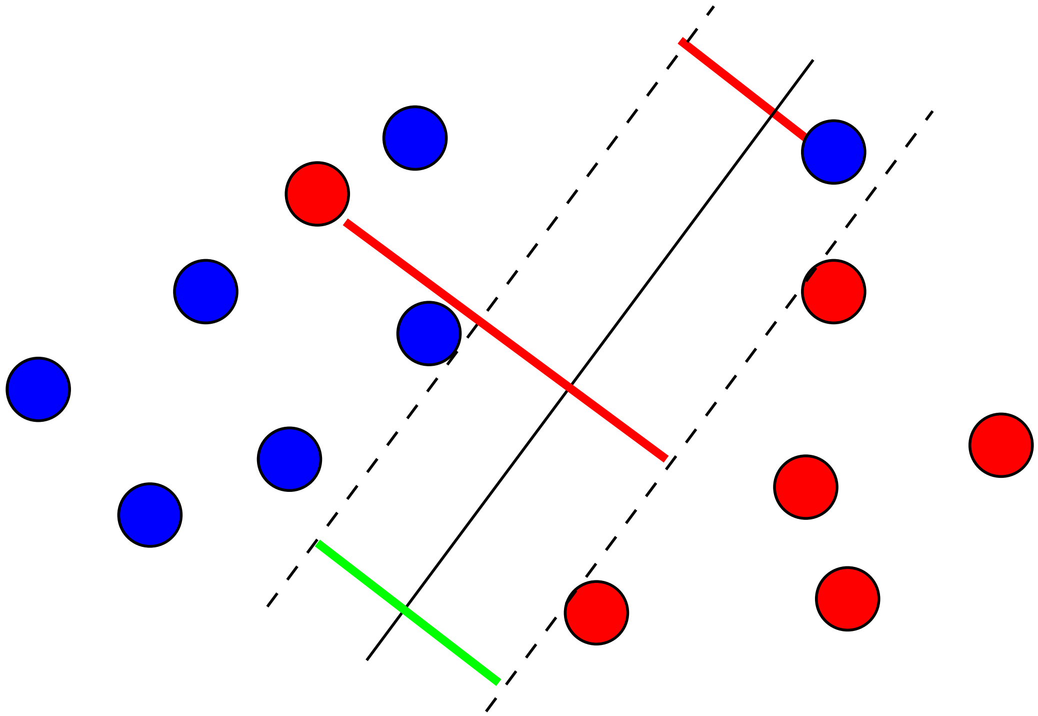
What happens when the data is not linearly separable?



What happens when the data is not linearly separable?



What happens when the data is not linearly separable?

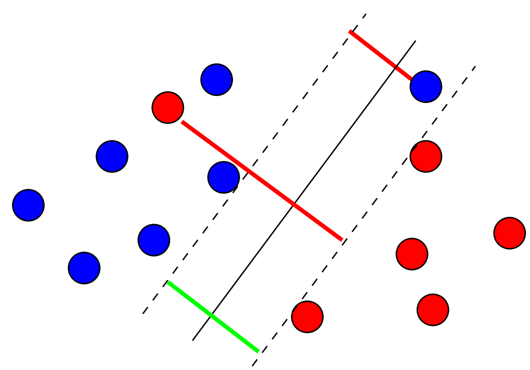


Soft-margin SVM ?

- 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:

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

Interpretation: bounded and unbounded support vectors