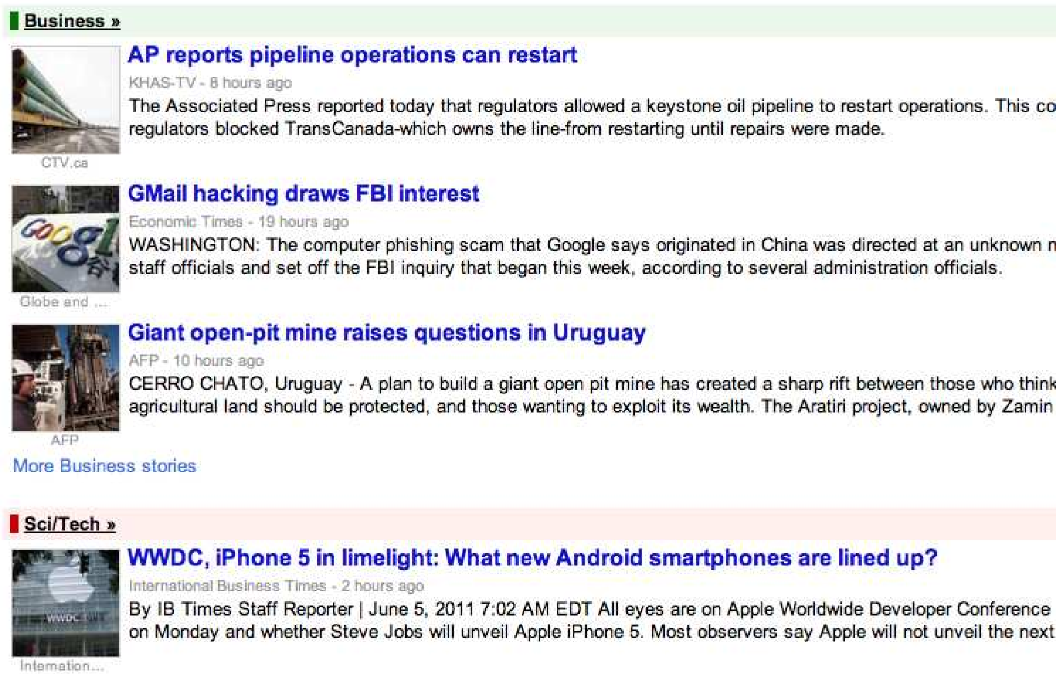
Language Information Processing, Advanced

Text Classifiers

mcuturi@i.kyoto-u.ac.jp

Today's talk

- Objective: supervised inference on text data.
  - **Ex.1** Given a large database of **news articles** about *business, sports, literature, politics* $etc.etc.$
    - ▷ Build a system that can classify **automatically** new articles.



Today's talk

- **Ex.2** Given a large set of e-mails in a mailbox, *family, friends, spam, ads, newletters* $etc.$
  - ▷ Build a system that **categorizes** automatically a new email.

Today's talk

- ◦ **Ex.3** Given a set of requests/messages sent to a retailer: complaints, need for technical support, praise
  - ▷ Build a system that **forwards directly** the message to the relevant department.

- Who is interested in this?
  - ◦ internet companies,
  - ◦ companies with large customer support receiving requests,
  - ◦ polling institutions,
  - ◦ social scientists who want to use text for their studies *etc.*

Text classification & probabilistic framework

- Assume that there is a probability p_{text} on texts on the internet

`Today will be a rainy day`

`In Ecuador tiger-hunters enjoy eating marshmallows`

`Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo`

- A probability quantifies how **likely** sentences are to appear
- Any idea on how this likelihood might be measured?

Text classification & probabilistic framework

- This probability takes into account **grammar and meaning**.
- Search engines are useful to have an idea about p_{text}

```
Today will be a rainy day
```

"today will be a rainy day"

About 288,000 results (0.24 seconds)

```
In Ecuador tiger-hunters enjoy eating marshmallows
```

"In Ecuador tiger-hunters enjoy eating marshmallows"

No results found for **"In Ecuador tiger-hunters enjoy eating marshmallows"**.

```
Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo
```

"Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo"

About 4,980,000 results (0.29 seconds)

Text classification & probabilistic framework

- We assume that there is **something to learn from data** (supervised inference)
- We assume our task is to categorize a given text among C given classes
  - agriculture, computer chips, energy, environment, sports, politics, gossip *etc.*
  - friends, family, spam, advertisements, newsletters *etc.*
- We also assume there is a probability $\boldsymbol{p}_{\text{cat}}$ **on categories**.

Text classification & probabilistic framework

- We assume that there is **something to learn from data** (supervised inference)
- We assume our task is to categorize a given text among C given classes
    - agriculture, computer chips, energy, environment, **sports**, **politics**, **gossip** *etc.*
    - friends, family, **spam, advertisements, newsletters** *etc.*
- Some documents appear more frequently than others.

$$\boldsymbol{p}_{\text{cat}}(\text{gossip}) > \boldsymbol{p}_{\text{cat}}(\text{philosophy})$$

Text classification & probabilistic framework

- Our goal will be to understand better the relationship betwee

$$\text{TEXT} \overset{?}{\leftrightarrow} \text{CATEGORY}$$

- Here, we assume also that there is a **joint** probability on texts and their category.

$$P(\texttt{text}, \texttt{category})$$

  which quantifies how likely the match between

  a text `text` and a category `category` is

- For instance,

$$P(\texttt{`I am feeling hungry these days'}, \texttt{'poetry'}) \approx 0$$

$$P(\texttt{`Manchester United's stock rose after their victory'}, \texttt{'business'})$$
$$\vee$$
$$P(\texttt{`Manchester United's stock rose after their victory'}, \texttt{'sports'})$$

Text classification & probabilistic framework

- Hence, given a sequence of words (including punctuation),

$$\mathbf{w} = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, \cdots, w_n)$$

- assuming we know P, the **joint** probability between texts and categories,
- an easy way to guess the category of $\mathbf{w}$ is by looking at

$$\text{category-prediction}(\mathbf{w}) = \operatorname*{argmax}_{\boldsymbol{C}} P(\boldsymbol{C}|\boldsymbol{w_1, w_2, \cdots, w_n})$$

Text classification & probabilistic framework

$$P(\text{'}\texttt{poetry}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.0037$$
$$P(\text{'}\texttt{business}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.005$$
$$P(\text{'}\texttt{sports}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.003$$
$$P(\text{'}\texttt{food}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.2$$
$$P(\text{'}\texttt{economy}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.04$$
$$P(\text{'}\texttt{society}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.08$$

Text classification & probabilistic framework

$$P(\text{'}\texttt{poetry}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.0037$$

$$P(\text{'}\texttt{business}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.005$$

$$P(\text{'}\texttt{sports}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.003$$

$$\rightarrow P(\text{'}\texttt{food}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.2$$

$$P(\text{'}\texttt{economy}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.04$$

$$P(\text{'}\texttt{society}\text{'}|\text{`}\texttt{I am feeling hungry these days}\text{'}) = 0.08$$

Bayes Rule

- Using Bayes theorem $p(A, B) = p(A|B)p(B)$,

$$P(\boldsymbol{C}|\boldsymbol{w_1, w_2, \cdots, w_n}) = \frac{P(\boldsymbol{C}, \boldsymbol{w_1, w_2, \cdots, w_n})}{P(\boldsymbol{w_1, w_2, \cdots, w_n})}$$

- When looking for the category C that best fits $\mathbf{w}$, we only focus on the numerator.
- Bayes theorem also gives that

$$\begin{aligned}
P(\boldsymbol{C}, \boldsymbol{w_1, \cdots, w_n}) &= P(\boldsymbol{C})P(\boldsymbol{w_1, w_2, \cdots, w_n}|\boldsymbol{C}) \\
&= P(\boldsymbol{C})P(\boldsymbol{w_1}|\boldsymbol{C})P(\boldsymbol{w_2, w_3, \cdots, w_n}|\boldsymbol{C}, \boldsymbol{w_1}) \\
&= P(\boldsymbol{C})P(\boldsymbol{w_1}|\boldsymbol{C})P(\boldsymbol{w_2}|\boldsymbol{C}, \boldsymbol{w_1})P(\boldsymbol{w_3, w_4, \cdots, w_n}|\boldsymbol{C}, \boldsymbol{w_1, w_2}) \\
&= \prod_{i=1}^{n} P(\boldsymbol{w_i}|\boldsymbol{C}, \boldsymbol{w_1, \cdots, w_{i-1}})
\end{aligned}$$

Examples

- Assume we have the beginning of this news title

$$w_1, \cdots, w_{12} = \text{`}\texttt{The weather was so bad that the organizers decided to close the}\text{'}$$

- If $C =$business, then

$$P(W_{13} = \text{`}\texttt{market}\text{'} | \texttt{business}, w_1, \cdots, w_{12})$$

should be quite high, as well as summit, meeting *etc.*.

- On the other hand, if we know $C =$sports, the probability for w_{13} changes significantly...

$$P(W_{13} = \text{`}\texttt{game}\text{'} | \texttt{sports}, w_1, \cdots, w_{12})$$

The Naive Bayes Assumption

- From a factorization

$$P(\boldsymbol{C}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_n}) = \prod_{i=1}^{n} P(\boldsymbol{w_i} | \boldsymbol{C}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_{i-1}})$$

  which handles all the **conditional** structures of text,

- we assume that each word appears **independently conditionally to** C,

$$\begin{aligned} P(\boldsymbol{w_i} | \boldsymbol{C}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_{i-1}}) &= P(\boldsymbol{w_i} | \boldsymbol{C}, \cancel{\boldsymbol{w_1}}, \cancel{\cdots}, \cancel{\boldsymbol{w_{i-1}}}) \\ &= P(\boldsymbol{w_i} | \boldsymbol{C}) \end{aligned}$$

- and thus

$$P(\boldsymbol{C}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_n}) = \prod_{i=1}^{n} P(\boldsymbol{w_i} | \boldsymbol{C})$$

The Naive Bayes Assumption Leads to Word Counts

- The factorization

$$P(\boldsymbol{w_i}|\boldsymbol{C}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_{i-1}}) = P(\boldsymbol{w_i}|\boldsymbol{C})$$

- means that we take for granted that

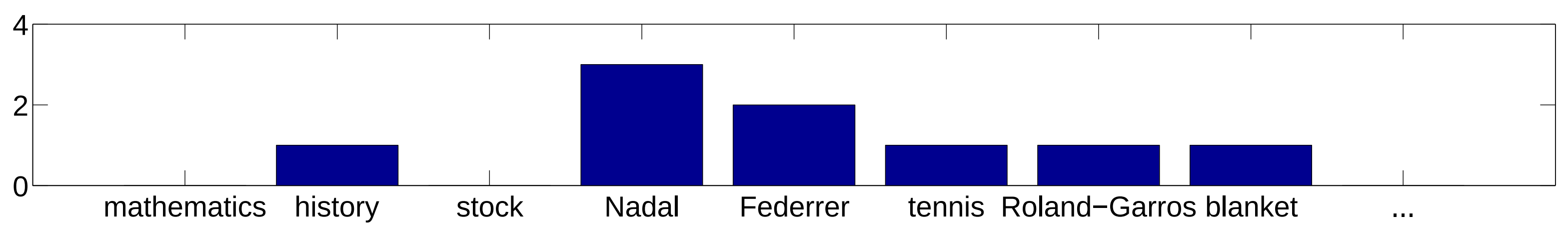
$$P(\boldsymbol{C}, \text{`}\texttt{The weather was bad so the meeting was closed}\text{'}) = P(\boldsymbol{C}, \text{`}\texttt{was The bad the closed meeting weather was so}\text{'})$$

The Naive Bayes Assumption Leads to Word Counts

- Assume we know $P(\boldsymbol{C}, w)$ for all words w in the dictionary and all categories.

$$P(\texttt{`business'}, \texttt{`stock'}) > P(\texttt{`sports'}, \texttt{`stock'})$$

- Given a text $\mathbf{T}$ =`But Federer has been quite a French Open security blanket for Nadal. Their rivalry is one of the greatest in tennis history, yet it has been decidedly short on suspense here. Nadal is now 5-0 against Federer at Roland-Garros. Nadal is the greatest ...`
- The only thing the Bayes classifier will consider is the **word histogram**



The Naive Bayes Assumption Leads to Word Counts

- To each text,
  - count the frequency of each word w in the dictionary $\mathcal{D}$, $\boldsymbol{h_w}$. Then

$$P(\mathbf{T}|\boldsymbol{C}) = \prod_{w \in \mathcal{D}} P(w|\boldsymbol{C})^{\boldsymbol{h_w}}$$

- In the example below, it seems obvious that the terms

$$P(W = \text{`}\texttt{Nadal}\text{'}|\texttt{tennis}), P(W = \text{`}\texttt{Federer}\text{'}|\texttt{tennis}), \cdots$$

  will be quite big.

- The Naive Bayes should easily classify this text as `tennis`...
  - **if the probabilities $P(w|\boldsymbol{C})$ were known!!!**

Term Frequencies

$\boxed{\text{We need to build an estimate of } P(w|\boldsymbol{C}) \text{ for } \textbf{all} \text{ words of } \mathcal{D}\text{, all categories}}$

Term Frequencies

We need to build an estimate of $P(w\|\boldsymbol{C})$ for **all** words of $\mathcal{D}$, all categories

A typical approach

- Consider a corpus of documents with different categories of text $\{(\mathbf{T}_1, c_1), \cdots, (\mathbf{T}_N, c_N)\}$.
- Build a reduced dictionary $\hat{\boldsymbol{\mathcal{D}}}$
  - using **all** words appearing in all $\mathbf{T}_i$,
  - usually removing non-informative words such as articles, prepositions *etc.*
- Compute histograms h_w^i for **each** $\mathbf{T}_i$ which only track words in $\hat{\mathcal{D}}$.
- Compute an estimate $\hat{\boldsymbol{p}}(\boldsymbol{w}|\boldsymbol{c})$ for each word $w \in \hat{\boldsymbol{\mathcal{D}}}$ and estimates $\hat{\boldsymbol{p}}(\boldsymbol{c})$.

Term Frequencies

- Use these elements, $\hat{p}$, $\hat{\mathcal{D}}$ to classify a new text $\mathbf{T}$ using his representation $h_w^{\mathbf{T}}$

$$\text{category-prediction}(\mathbf{T}) = \operatorname*{argmax}_c \left(\hat{p}(c) \prod_{w \in \hat{\mathcal{D}}} \hat{p}(w|c)^{h_w^{\mathbf{T}}} \right)$$

- of course, if we use the logarithm of the r.h.s., we get the rule

$$\text{category-prediction}(\mathbf{T}) = \operatorname*{argmax}_c \log \hat{p}(c) + \sum_{w \in \hat{\mathcal{D}}} h_w^{\mathbf{T}} \log \hat{p}(w|c)$$

$$\boxed{\text{Naive Bayes for text} \Leftrightarrow \text{Linear Classifier Using Term Frequencies as Features}}$$

- Once this is established... we could imagine **any** linear classifier using TF.

Term Frequency Data Seen from a Classification Perspective

- The **Data** we have:
  - texts $\mathbf{T}_i$ translated as histograms of words $h^1, h^2, h^3, \cdots, h^N$.
  - Each histogram is a vector of the simplex Σ_d where $d = \#\mathcal{D} - 1$ and

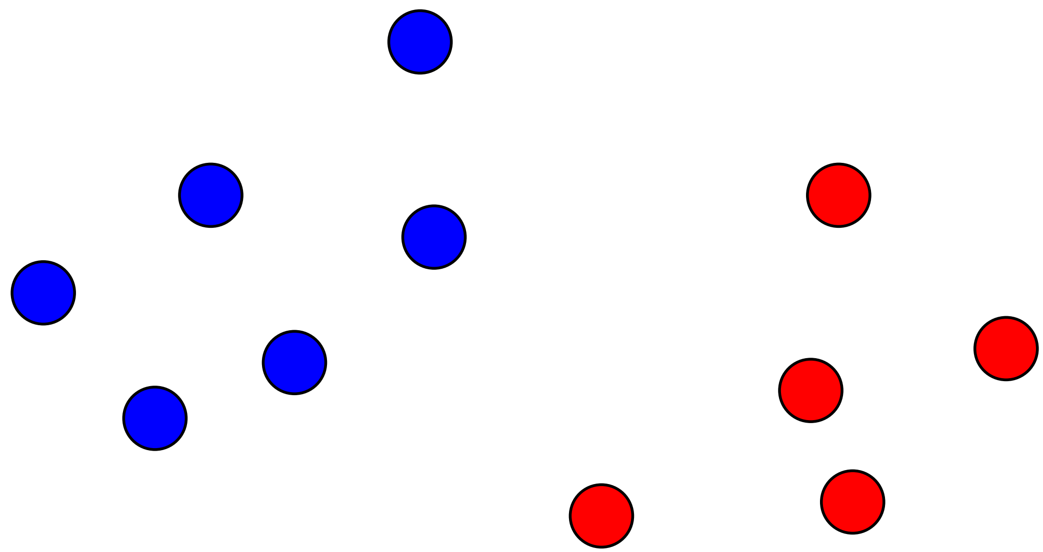
$$\Sigma_d = \{x \in \mathbb{R}^{d+1} | \, x_i \geq 0, \sum_{i=1}^{d+1} x_i = 1\}.$$

- We consider 2 categories only here, for instance "spam" vs "non-spam".
- The corpus consists in a large number of **histogram/bit** pairs

$$\text{"training set"} \;=\left\{\left(h_i = \begin{bmatrix} h^i_{w_1} \\ h^i_{w_2} \\ \vdots \\ h^i_{w_{d+1}} \end{bmatrix} \in \Sigma^d, \; \mathbf{y}_i \in \{0,1\} \right)_{i=1..N}\right\}$$

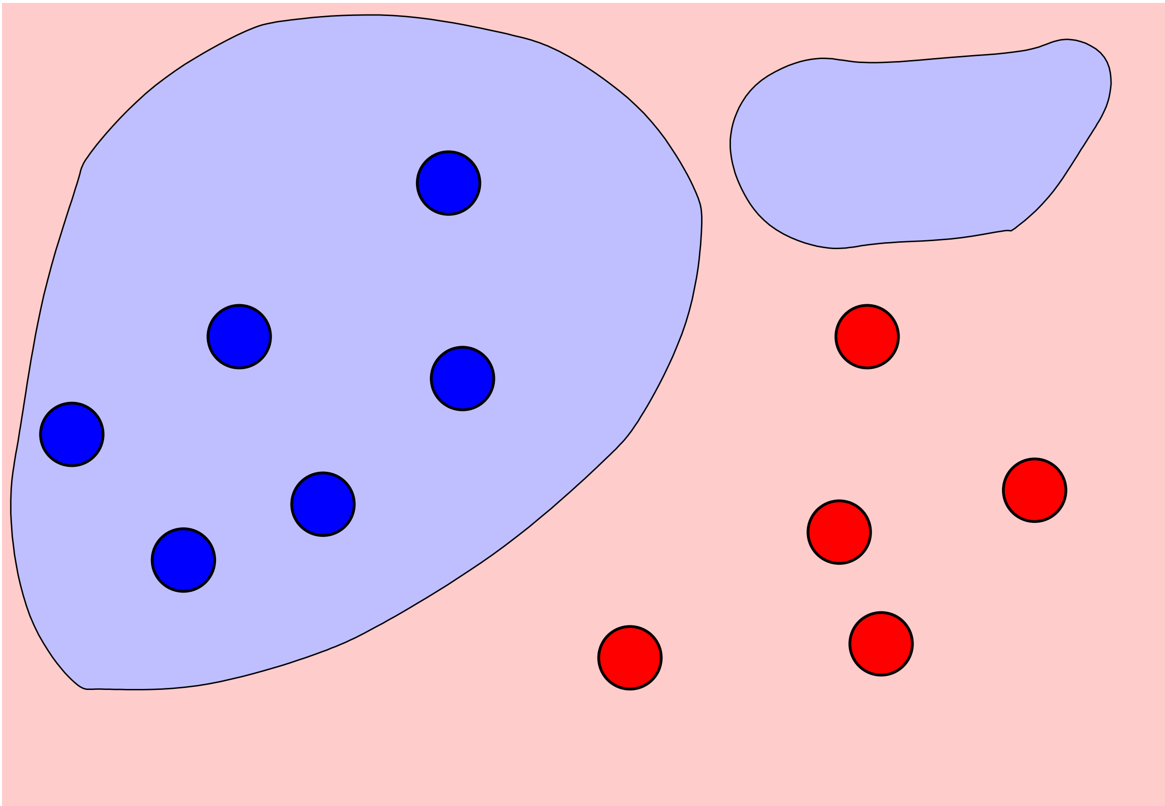
- For illustration purposes **only** we will consider the 2 dimensional simplex, that is $\#\mathcal{D} = 3$.

Binary Classification Separation Surfaces for Vectors



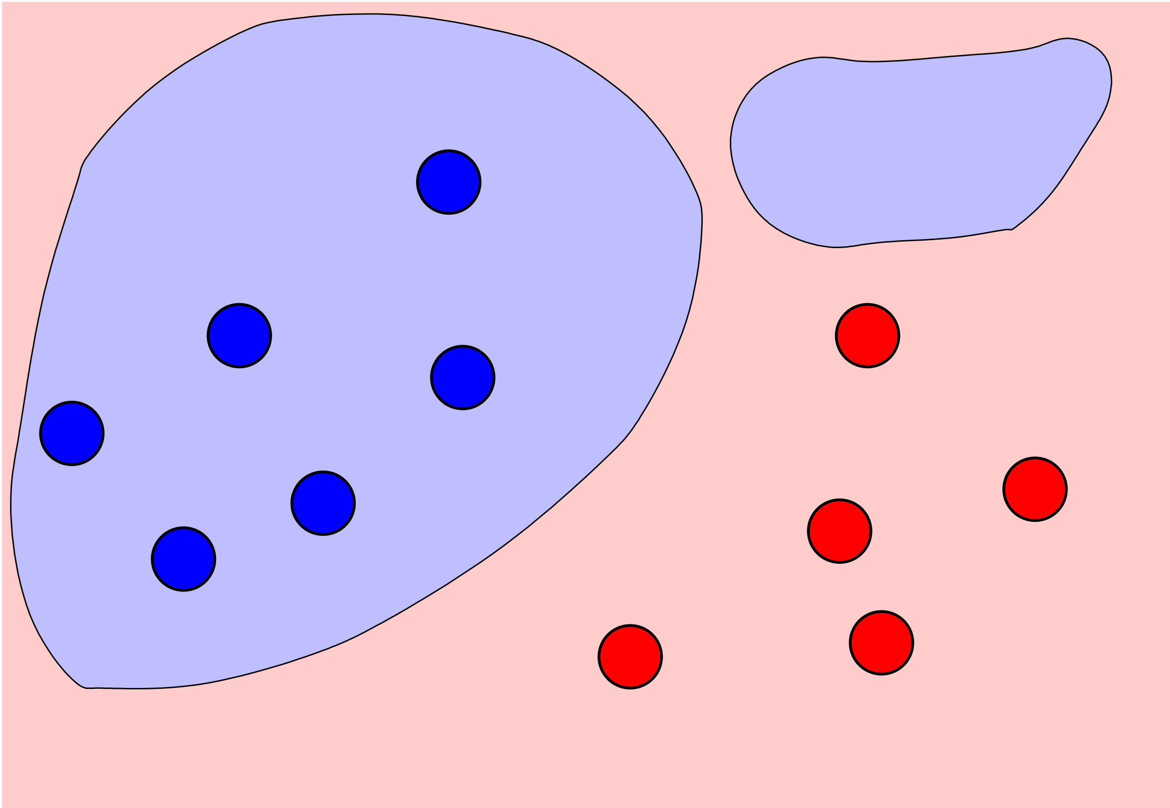
What is a classification rule?

Binary Classification Separation Surfaces for Vectors



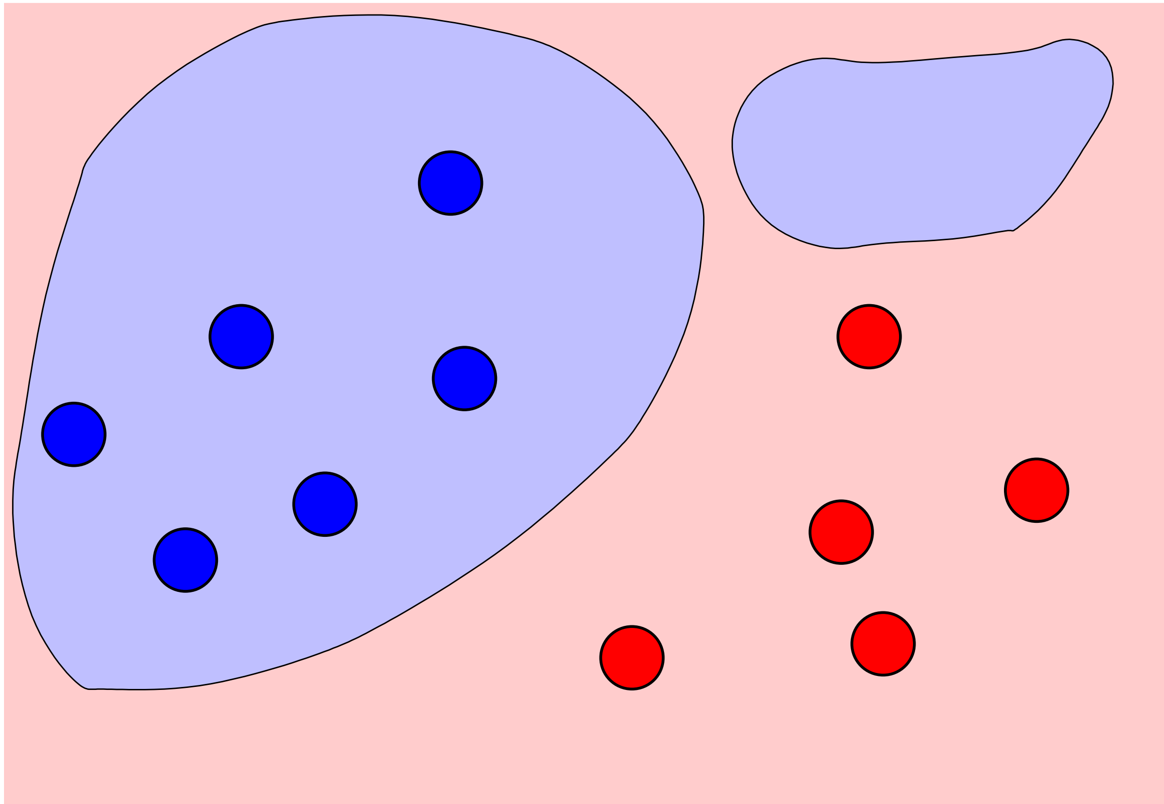
Classification rule = a partition of $\mathbb{R}^d$ into two sets

Binary Classification Separation Surfaces for Vectors



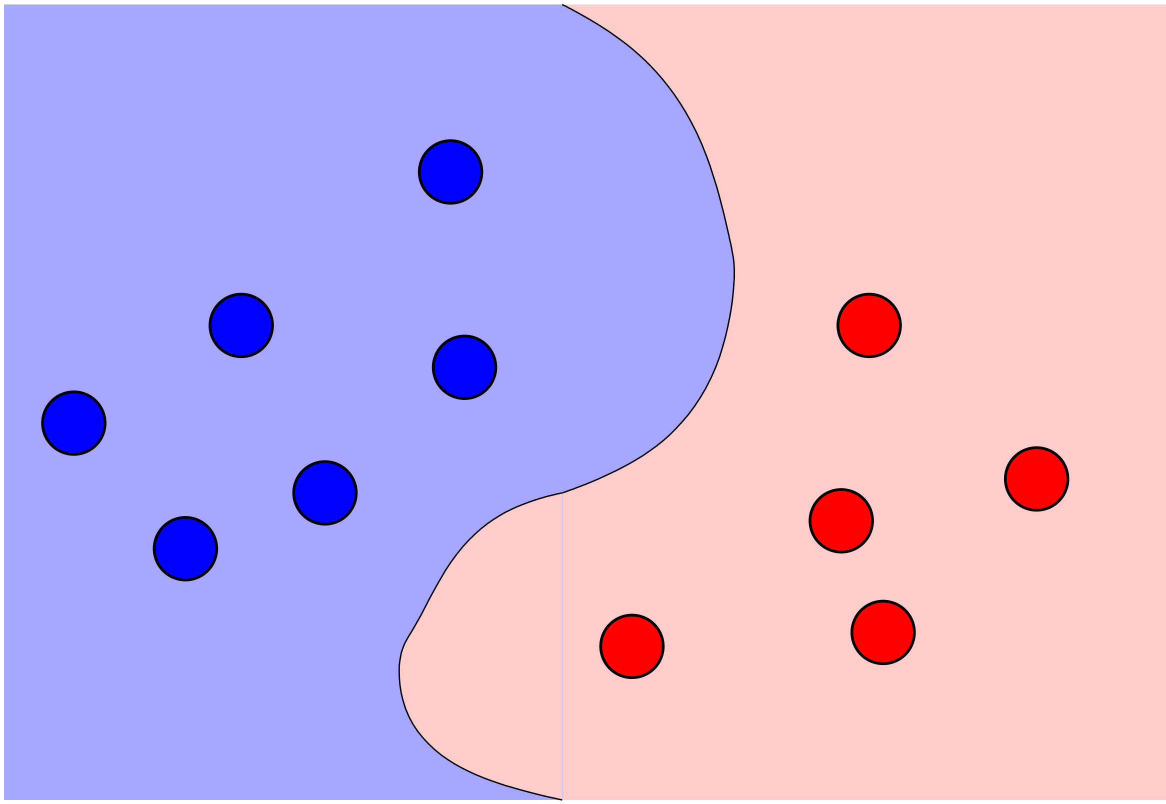
This partition is usually interpreted as the level set of a function

Binary Classification Separation Surfaces for Vectors



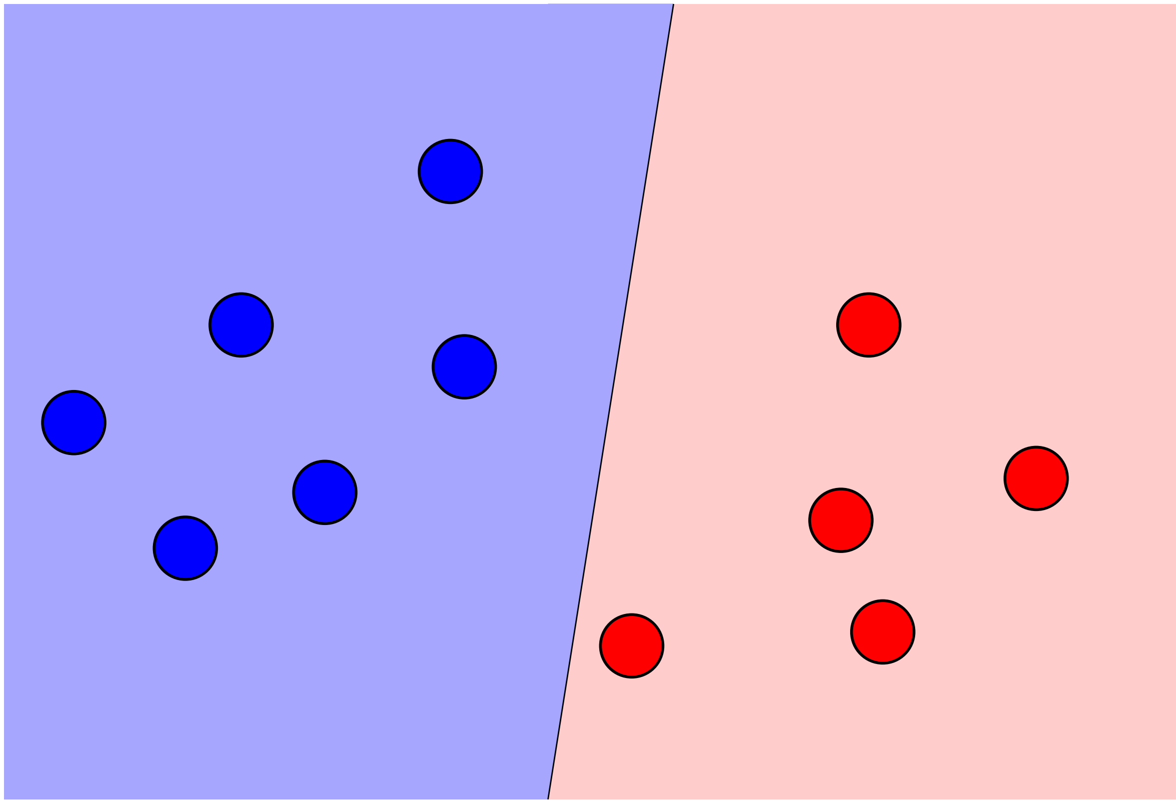
Typically, $\{h \in \Sigma_d | \boldsymbol{f}(h) > 0\}$ and $\{h \in \Sigma_d | \boldsymbol{f}(h) \leq 0\}$

Classification Separation Surfaces for Vectors



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

Classification Separation Surfaces for Vectors



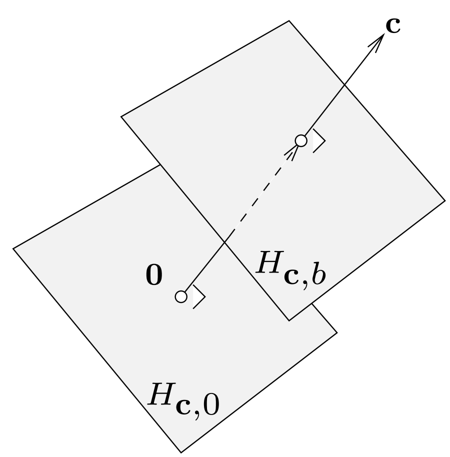
Even more **simple**: using **straight lines** and halfspaces.

Linear Classifiers

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

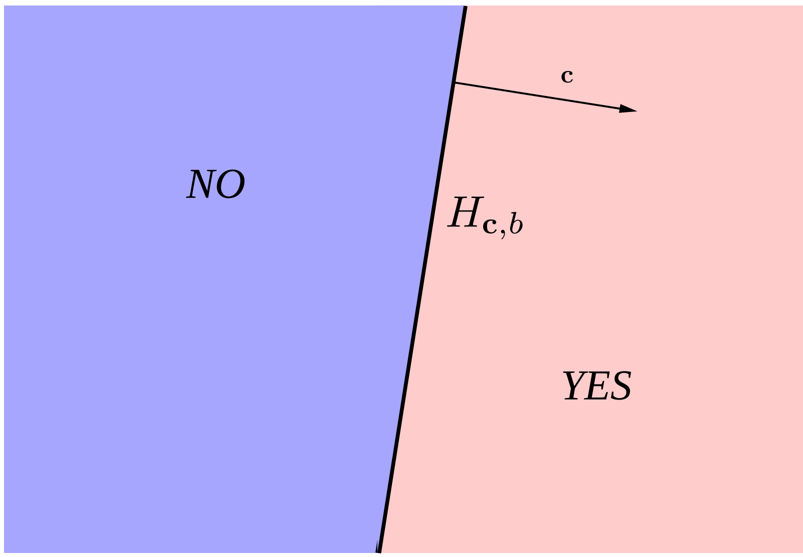


Linear Classifiers

- In Σ_d, things hypersurfaces **divide** $\mathbb{R}^d$ into **two** halfspaces,

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

- 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

- Training a classifier is mapping a dataset to a $\mathbf{c}$ and b.

$$\text{“training set”} \;\; \Big\{ \big(h^i \in \Sigma^d,\; \mathbf{y}_i \in \{0,1\} \big)_{i=1..N} \Big\} \overset{????}{\Longrightarrow} \text{“best”}\, \mathbf{c}^\star,\, b^\star$$

  has different answers.

- **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.*

What is special about natural text?

- Remember we have
  - A corpus of N documents $\{(\mathbf{T}_1, c_1), \cdots, (\mathbf{T}_N, c_N)\}$.
  - Build a reduced dictionary $\hat{\mathcal{D}}$ of M words
  - Compute histograms h_w^i for **each** $\mathbf{T}_i$ which only track words in $\hat{\mathcal{D}}$.
- What is difficult about text processing usually?

$$\boxed{\text{Usually, } M \text{ is very large, possible bigger than } N}$$

$$H = \begin{array}{l} \\ \text{eat} \\ \text{ball} \\ \text{dinosaur} \\ \text{genome} \\ \text{planet} \\ \text{Clooney} \\ \text{Guatemala} \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{T}_1 & \mathbf{T}_2 & \mathbf{T}_3 & \mathbf{T}_4 & \cdots & \mathbf{T}_N \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & 0 & \cdots & 0 \\ 4 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 2 & \cdots & 0 \\ 0 & 0 & 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

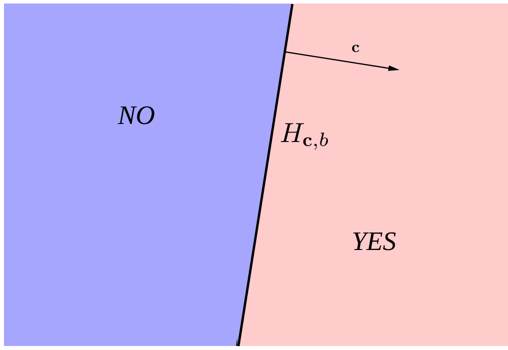
Sparse Classifiers

`sparse` (adj. sparser, sparsest)

Occurring, growing, or settled at widely spaced intervals; not thick or dense

What's the goal

- The goal when estimating linear classifiers: define $c \in \mathbb{R}^M$ and $b \in \mathbb{R}$.



- The number of words is M, defining a vector c means setting a value for:

$$c = \begin{bmatrix} c_{\text{eat}} \\ c_{\text{ball}} \\ c_{\text{dinosaur}} \\ c_{\text{genome}} \\ c_{\text{planet}} \\ c_{\text{Clooney}} \\ \vdots \end{bmatrix}$$

Sparse and non-sparse

- Without any constraint, defining $c^\star$ is simply:

$$\min_{c\in\mathbb{R}^M, b\in\mathbb{R}} \text{error}(c, b)$$

for instance, error can be the logistic error, the hinge loss (SVM) etc...

- With a **sparsity** constraint, we have

$$\min_{c,b\in\mathbb{R}, \|c\|_0\leq p} \text{error}(c, b), \text{ where } \|c\|_0 \overset{\text{def}}{=} \sum_{i=1}^{M} \mathbf{1}_{c_i\neq 0}$$

Sparse and non-sparse

- sparse vector:

$$c = \begin{bmatrix} 0 & 0 & 0 & 1.324 & 0 & 0 & -3.21 & 0 & 0 & 0 \end{bmatrix}$$

$$\|c\|_0 = 2$$

- dense vector

$$c = \begin{bmatrix} 0.21 & -4.65 & 3.2 & 6.982 & 5.43 & -9.1 & 0.004 & -0.37 & 12.1 & 3.94 \end{bmatrix}$$

$$\|c\|_0 = 10$$

- a **sparsity constraint** enforces the solution to be sparse and **not dense**

$$\min_{c,b\in\mathbb{R}, \|\boldsymbol{c}\|_0 \leq \boldsymbol{p}} \text{error}(c,b), \text{ where } \|c\|_0 \stackrel{\text{def}}{=} \sum_{i=1}^{M} \mathbf{1}_{c_i \neq 0}$$

Why we like sparse

$\boxed{\text{Sparse solutions for } c \text{ are desirable because}}$

- they are lighter in memory. computations only grow in p, not M anymore.

$$c = \begin{bmatrix} 0 & 0 & 0 & 1.324 & 0 & 0 & -3.21 & 0 & 0 & 0 \end{bmatrix}$$

$$c^T x = 1.324 \times x_4 - 3.21 \times x_7$$

- since only p words matter, these are **keywords** which can be interpreted
  - $c_4 > 0$, *genome* is the important word to predict positively
  - $c_7 > 0$, *Guatemala* is the important word to predict negatively

How we can solve a "sparsified" problem

How can we estimate sparse solutions $\mathbf{c}^\star$?

- **Direct** approach

$$\min_{c,b\in\mathbb{R},\|\boldsymbol{c}\|_0\leq \boldsymbol{p}} \text{error}(c,b),\ \text{where } \|c\|_0 \stackrel{\text{def}}{=} \sum_{i=1}^{M} \mathbf{1}_{c_i\neq 0}$$

is **computationally intractable**.

- **Alternative** approach: penalize with the l_1 norm

$$\min_{c,b\in\mathbb{R}} \text{error}(c,b) + \lambda\|\boldsymbol{c}\|_1,\ \text{where } \|c\|_1 \stackrel{\text{def}}{=} \sum_{i=1}^{M} |c_i|$$

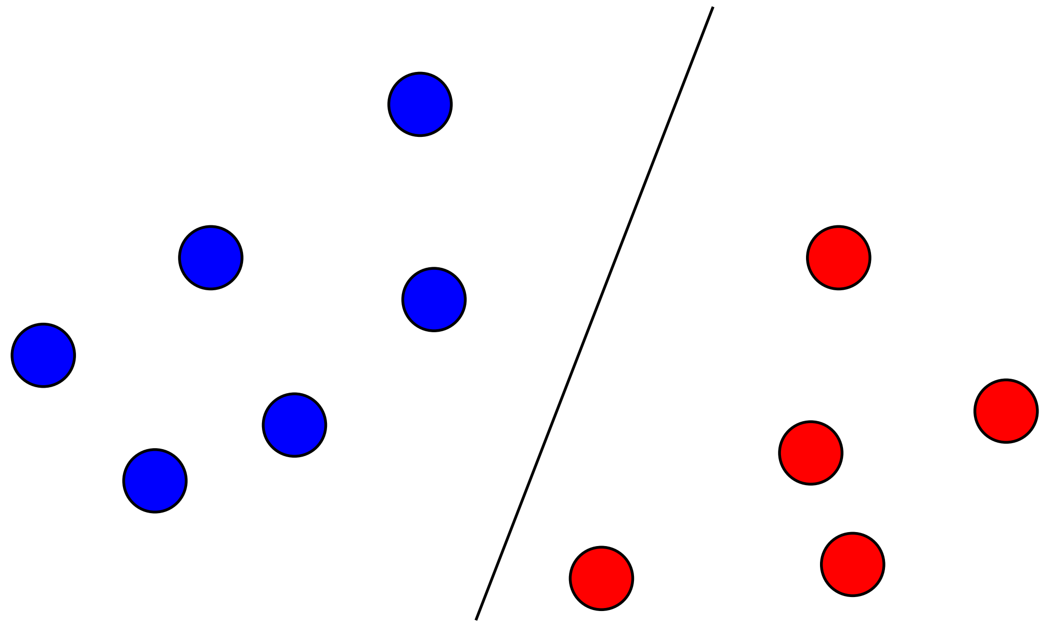
can prove that we can recover sparse solutions.

- Many algorithms: LASSO, FISTA... see literature on compressive sensing.
- Example: http://statnews.org/ website by Laurent El Ghaoui

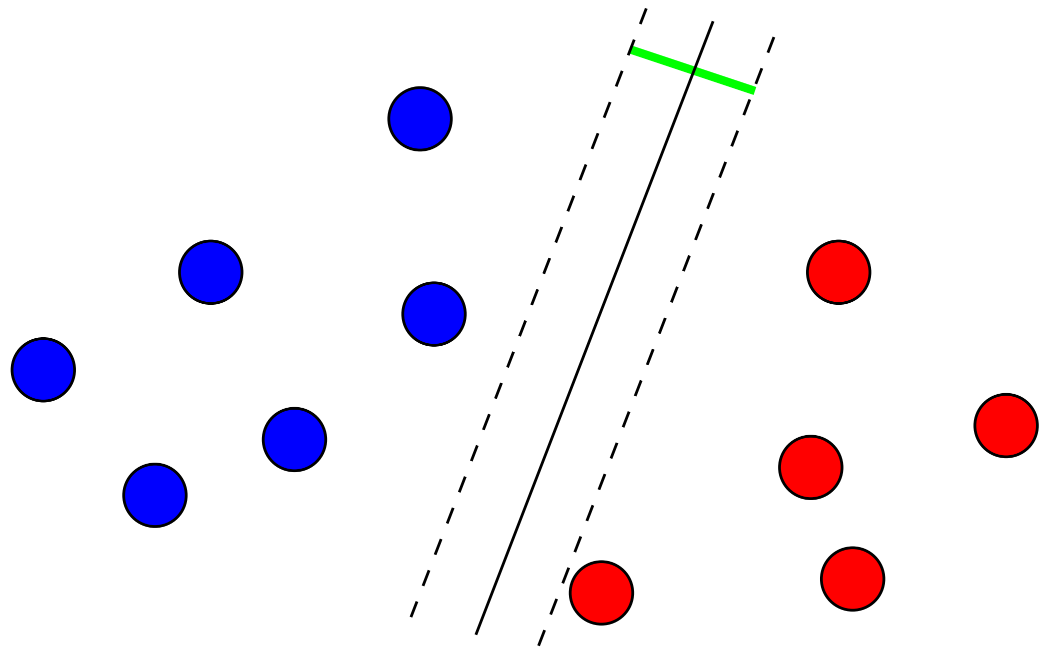
Support Vector Machine

Check the very nice book on the subject by T.Joachims. It's a bit old now but contains a lot of fundamental ideas.

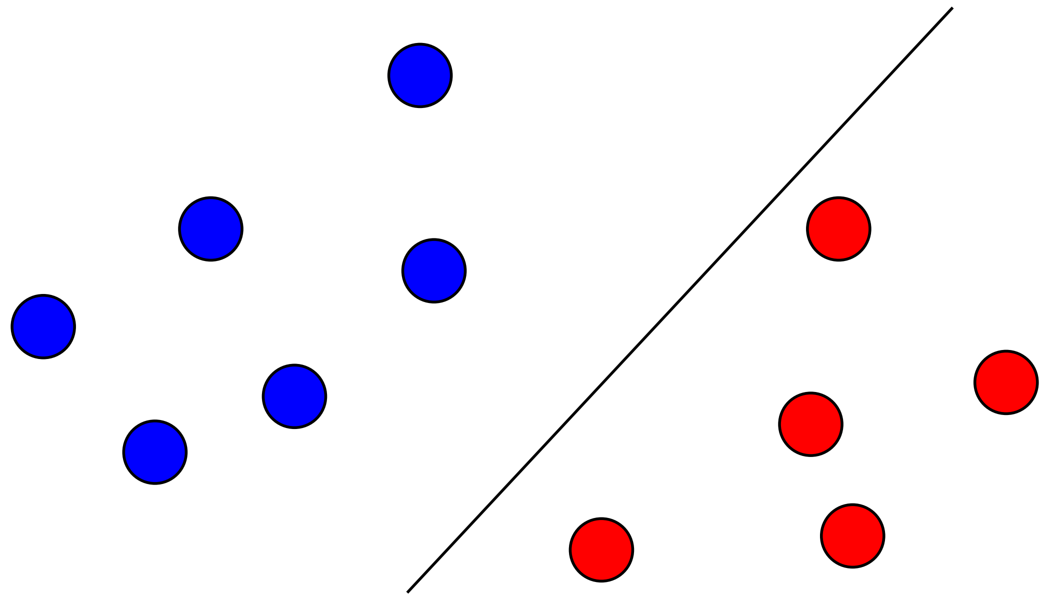
A criterion to select a linear classifier: the margin ?



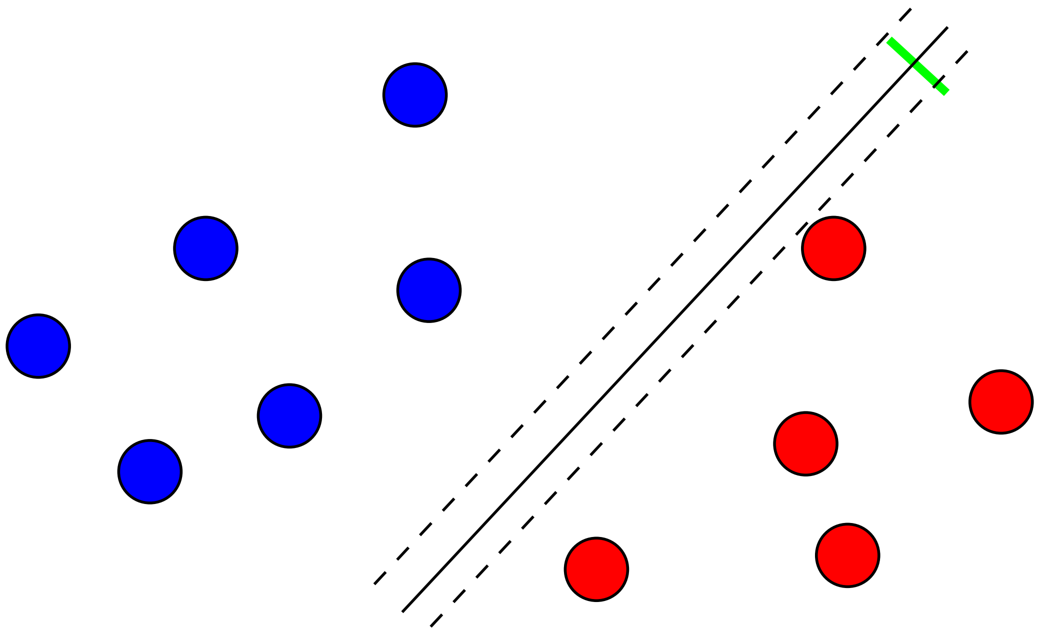
A criterion to select a linear classifier: the margin ?



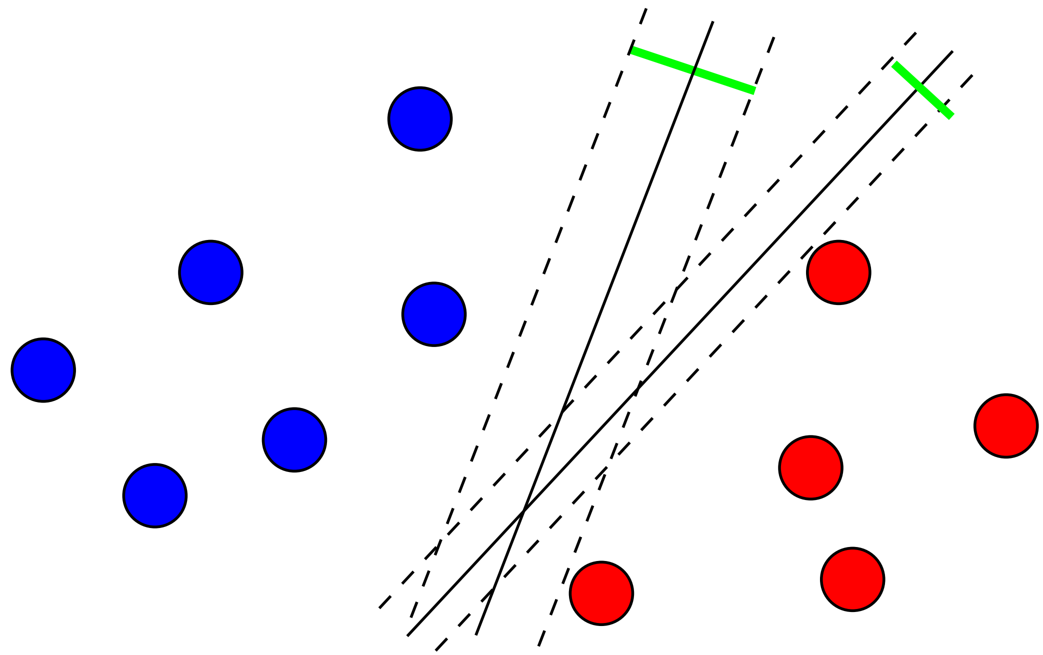
A criterion to select a linear classifier: the margin ?



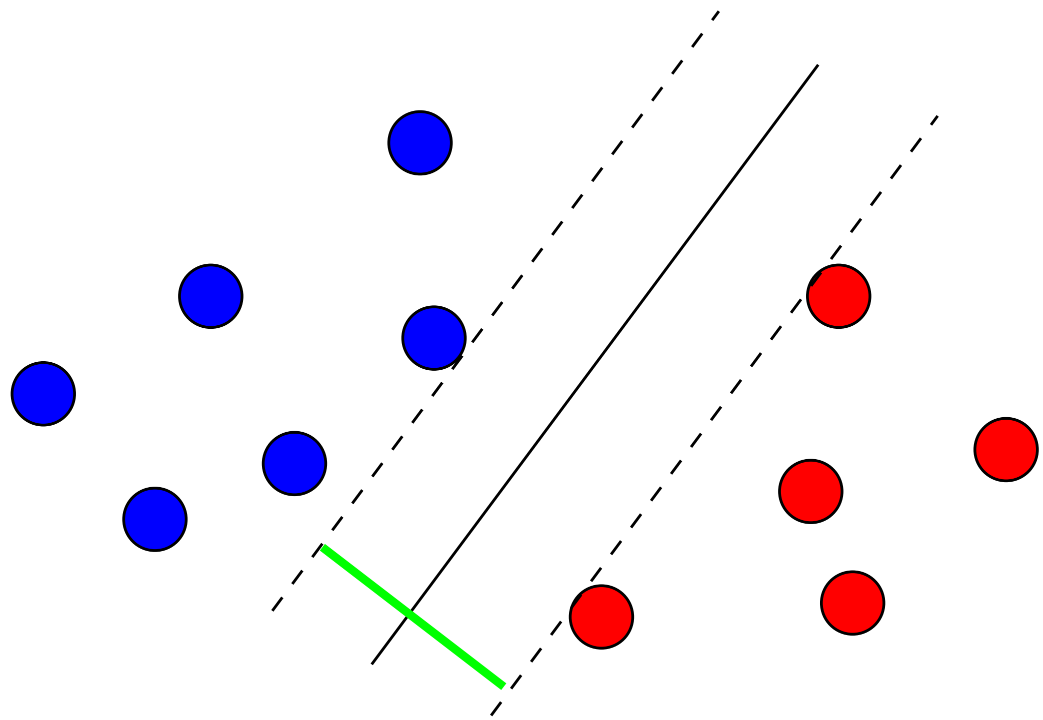
A criterion to select a linear classifier: the margin ?



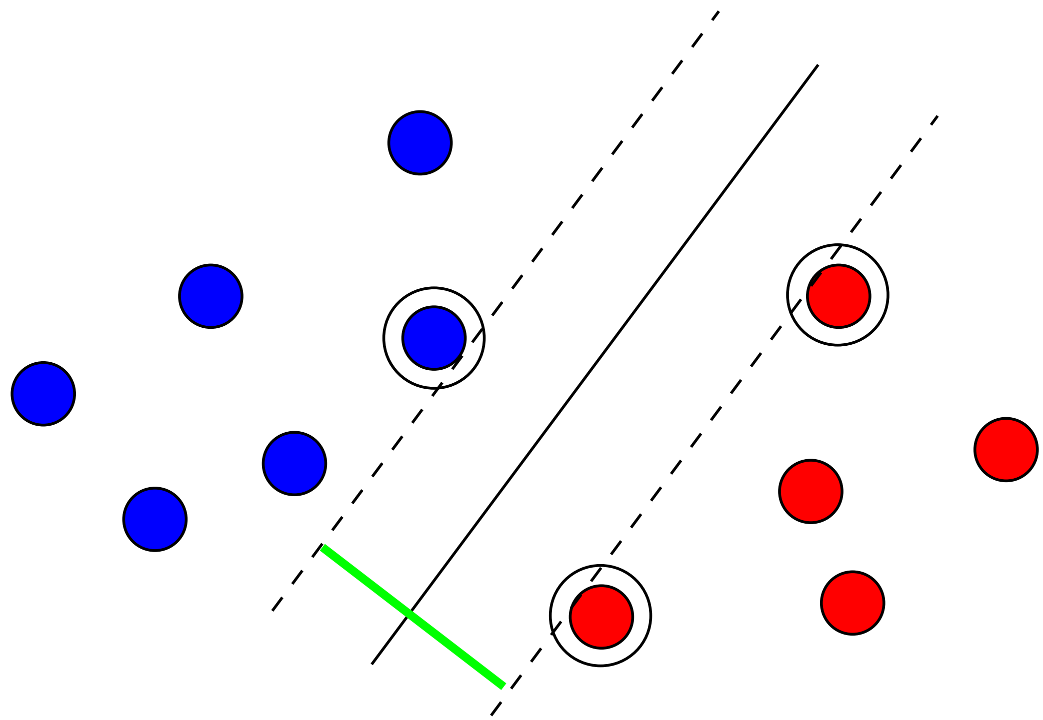
A criterion to select a linear classifier: the margin ?



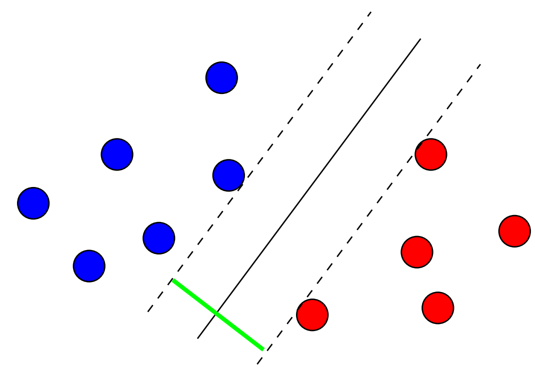
Largest Margin Linear Classifier ?



Support Vectors with Large Margin



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

- 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

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

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

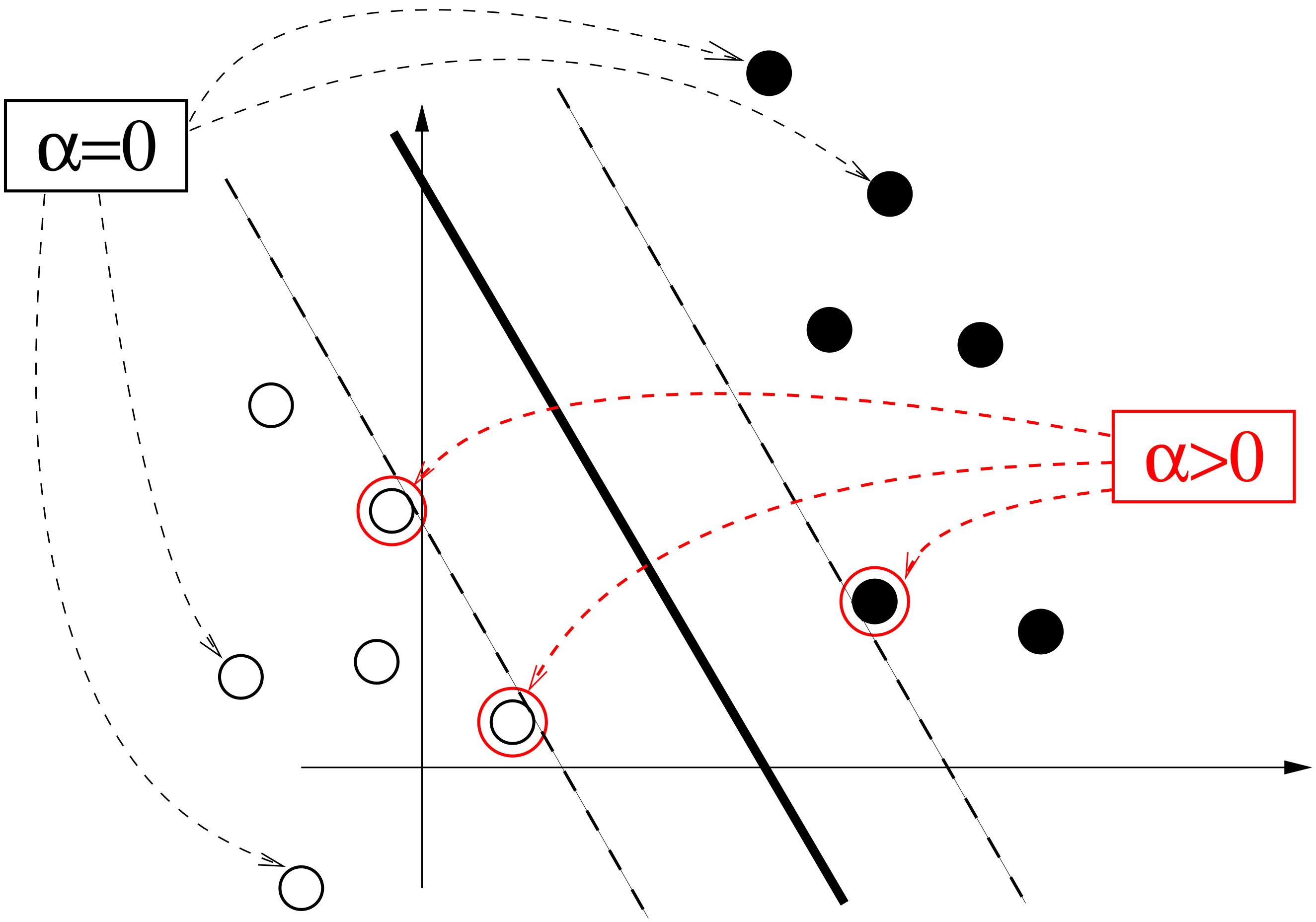
$$b^* = -\frac{1}{2}\left(\min_{\mathbf{y}_i=1,\alpha_i>0}(\mathbf{w}^T\mathbf{x}_i) + \max_{\mathbf{y}_i=-1,\alpha_i>0}(\mathbf{w}^T\mathbf{x}_i)\right)$$

- the **decision function** is therefore:

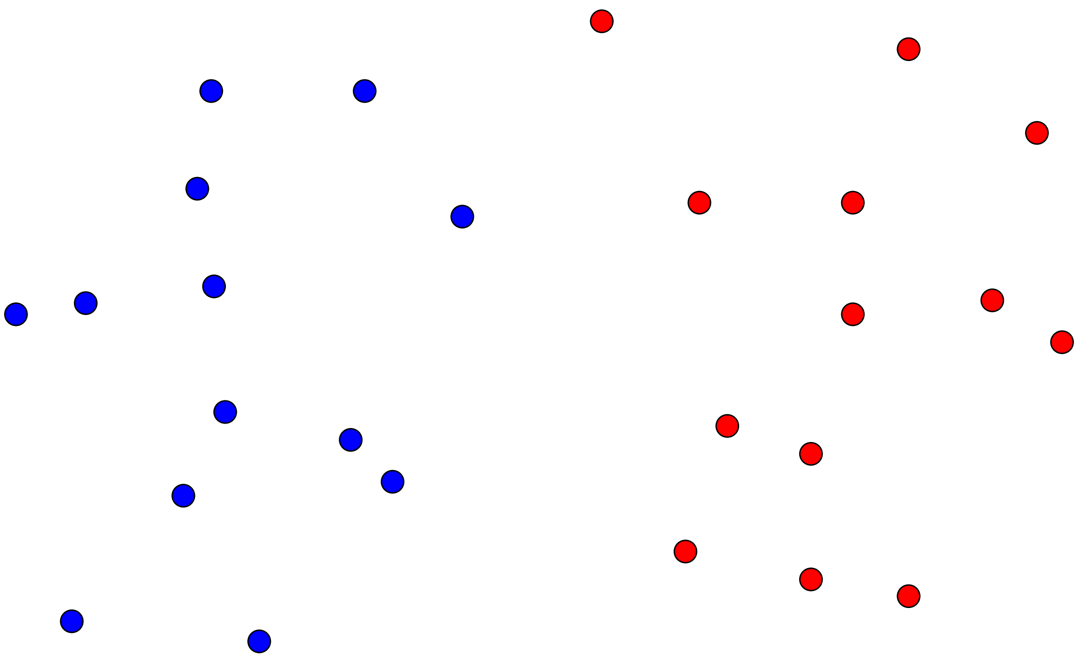
$$\begin{aligned} f^*(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} + b^* \\ &= \sum_{i=1}^n y_i \alpha_i \mathbf{x}_i^T \mathbf{x} + b^*. \end{aligned}$$

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

Interpretation: support vectors

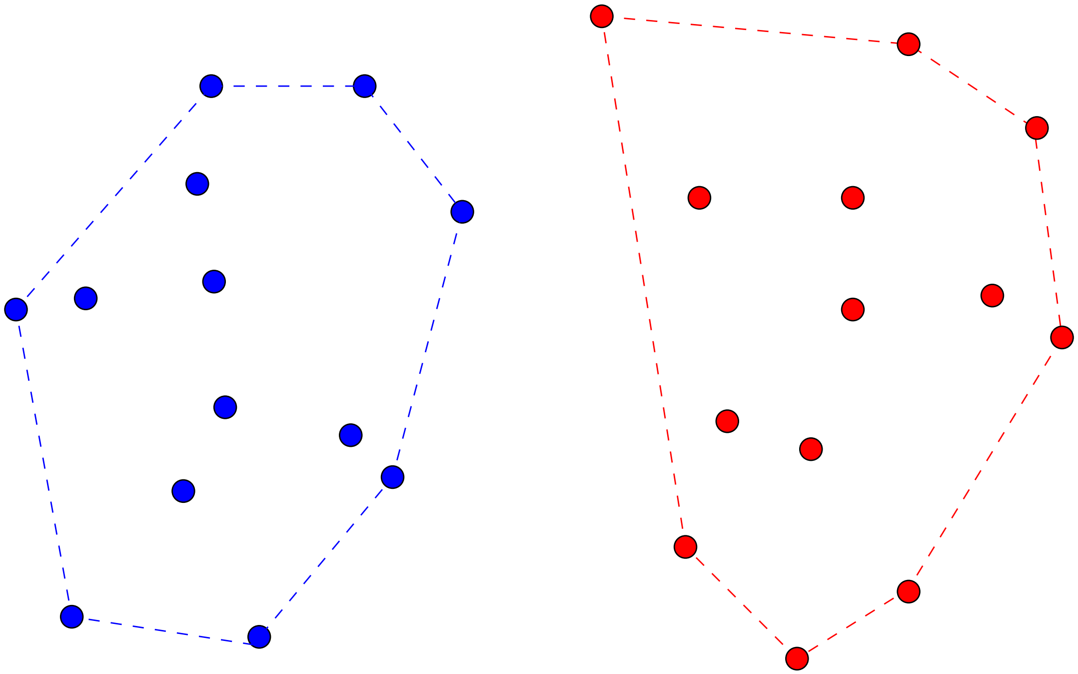


Another interpretation: Convex Hulls



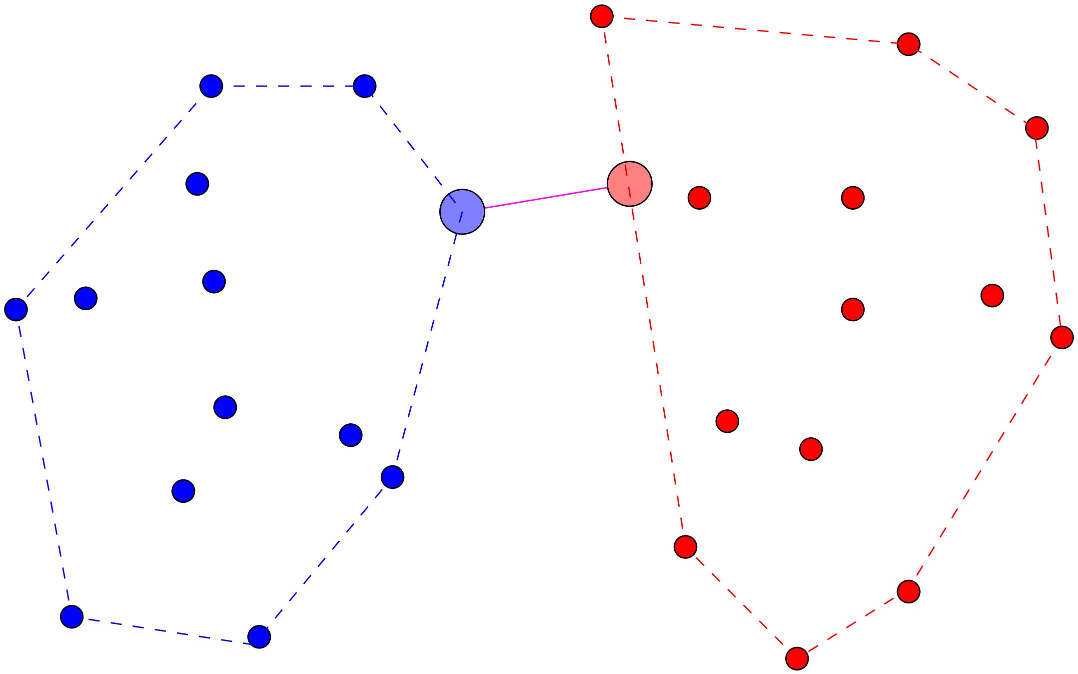
go back to 2 sets of points that are linearly separable

Another interpretation: Convex Hulls



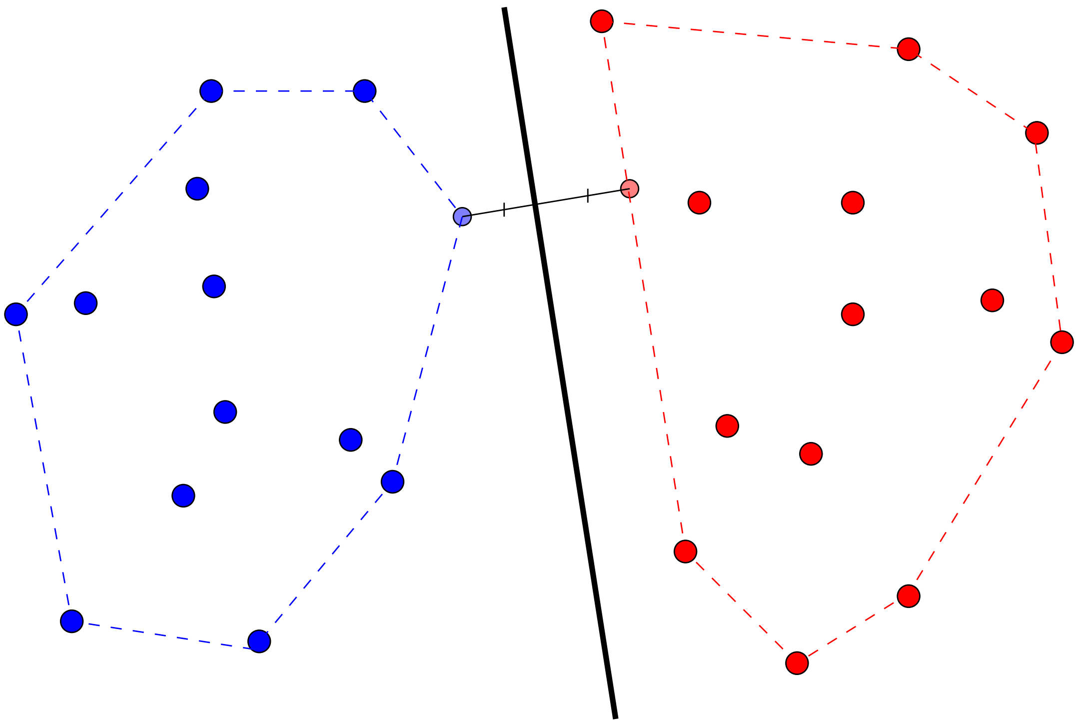
$\boxed{\text{Linearly separable} = \text{convex hulls do not intersect}}$

Another interpretation: Convex Hulls



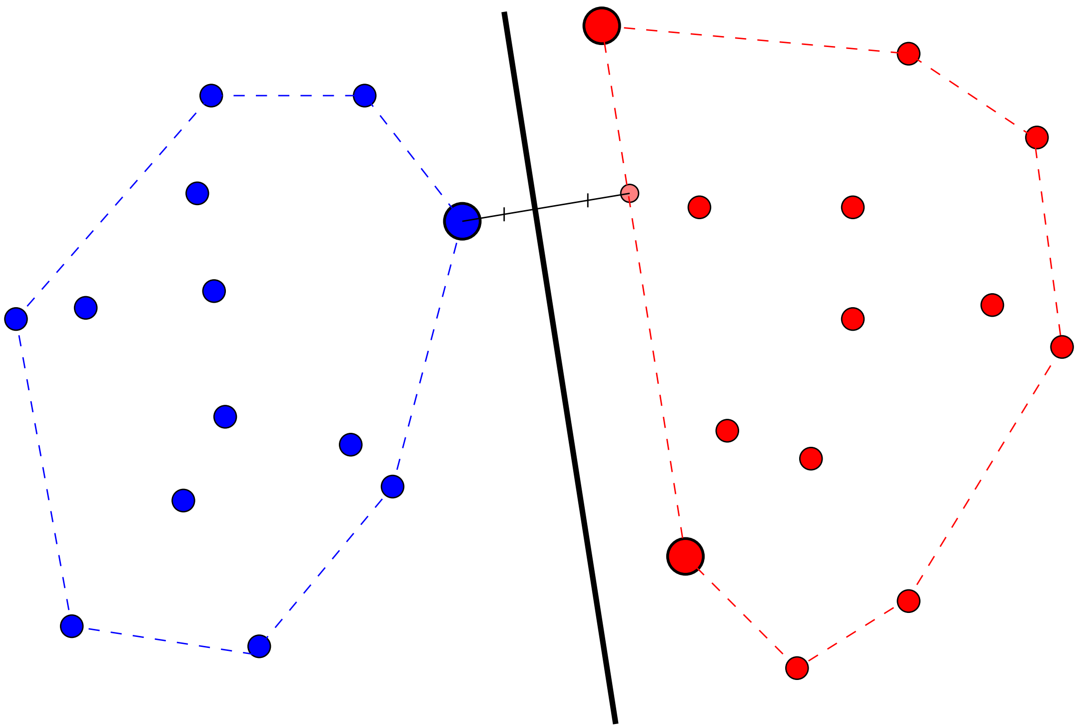
Find two closest points, one in each convex hull

Another interpretation: Convex Hulls



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

Another interpretation: Convex Hulls



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

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:

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

- 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 $\mathbf{y}$:

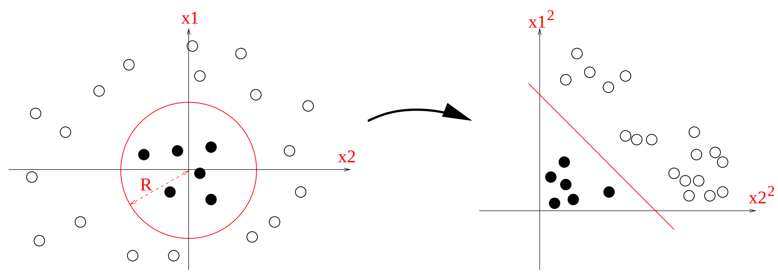
$$\begin{array}{ll} \text{maximize} & g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2}\alpha^T(\boldsymbol{K} \odot \mathbf{y}\mathbf{y}^T)\alpha \\ \text{such that} & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \dots, n \\ & \sum_{i=1}^{n} \alpha_i \mathbf{y}_{\boldsymbol{i}} = 0. \end{array}$$

- $\boldsymbol{K}$'s **positive definite** $\Leftrightarrow$ **problem has a 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

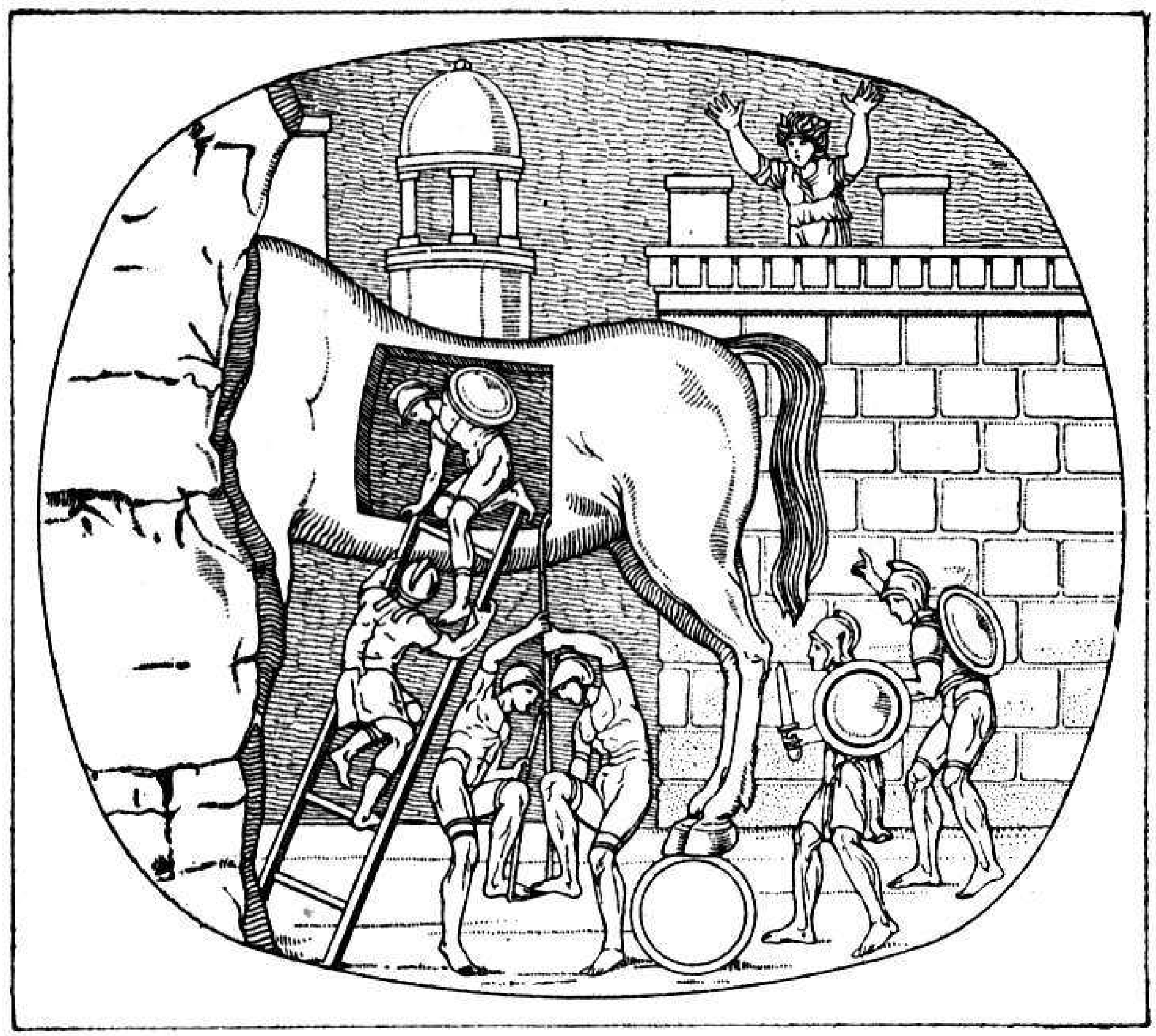
- For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

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



Kernels are Trojan Horses onto Linear Models

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



Kernels For Histograms

- An abridged bestiary of **negative definite distances** on the probability simplex:

$$\psi_{JD}(\theta,\theta') = h\left(\frac{\theta+\theta'}{2}\right) - \frac{h(\theta)+h(\theta')}{2},$$

$$\psi_{\chi^2}(\theta,\theta') = \sum_i \frac{(\theta_i-\theta_i')^2}{\theta_i+\theta_i'}, \quad \psi_{TV}(\theta,\theta') = \sum_i |\theta_i-\theta_i'|,$$

$$\psi_{H_2}(\theta,\theta') = \sum_i |\sqrt{\theta_i}-\sqrt{\theta_i'}|^2, \quad \psi_{H_1}(\theta,\theta') = \sum_i |\sqrt{\theta_i}-\sqrt{\theta_i'}|.$$

- Recover kernels through

$$k(\theta,\theta') = e^{-t\psi}, \quad t>0$$