SVM
(Support Vector Machine)

Wang Houfeng
Institute of Computational Linguistics
Peking University

Outline

- **Review of The Perceptron**
- **Preliminaries, Linear classifier & Definition of SVM**
- **Optimization Theory**
- **Non-Linear & Kernel Functions**
- **Implementation**
- **Introduction to NLP applications**

The Classification Setting

Class, Point, Example, Data Set, …

- Input Space: $X \subseteq R^n$
- (binary) Output Space: $Y = \{+1,-1\}$ or Multi-class: $Y = \{1, 2, \ldots, M\}$
- A point, pattern :

  $x \in X,\ x = (x_1, x_2, \ldots, x_n)$
- Example: (x, y) with $x \in X, y \in Y$
- Training Set: a set of examples generated i.i.d. according to an unknown distribution $P(x,y)$

$$D = \{(x_1, y_1), \ldots, (x_l, y_l)\}$$

The Classification Setting: Learning

- The hypotheses space, *F*, is the set of functions $f: X \to Y$, the learner pick out an optimal f from F (Good Training and good generalization). In SVM, f is of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + b = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

- Good training (**Learner**):

$f(x_i) = y_i$, i =1,2,…,*l, where* $S = \{(x_1, y_1), \ldots, (x_l, y_l)\} \subseteq (X \times Y)^l$

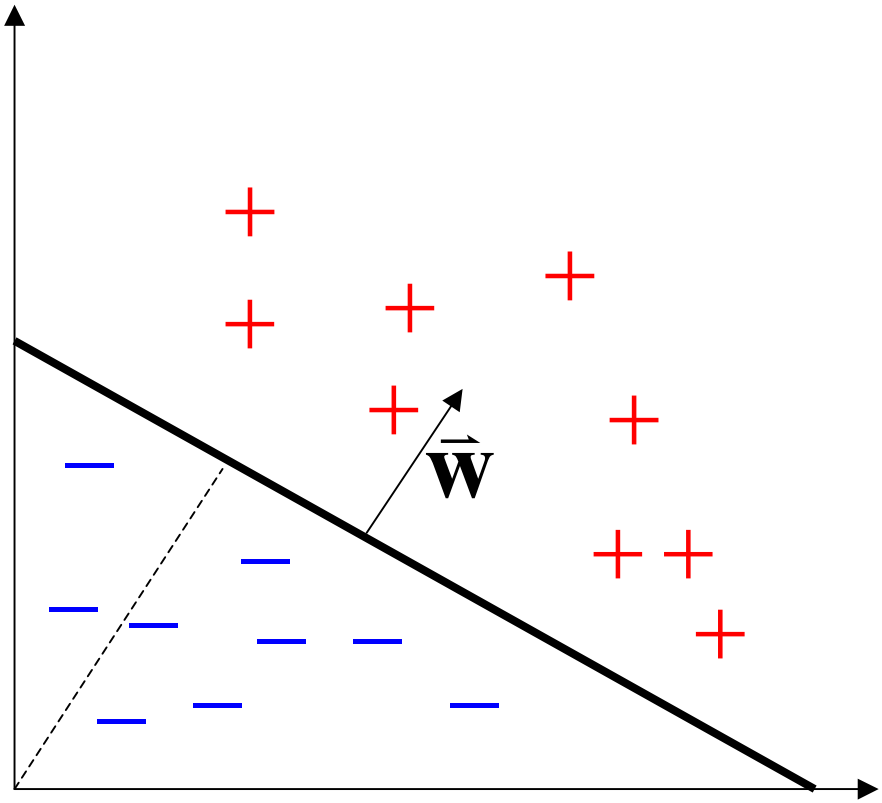
- Good generalization: for a **novel pair** (x, y)

$f(x) = y$

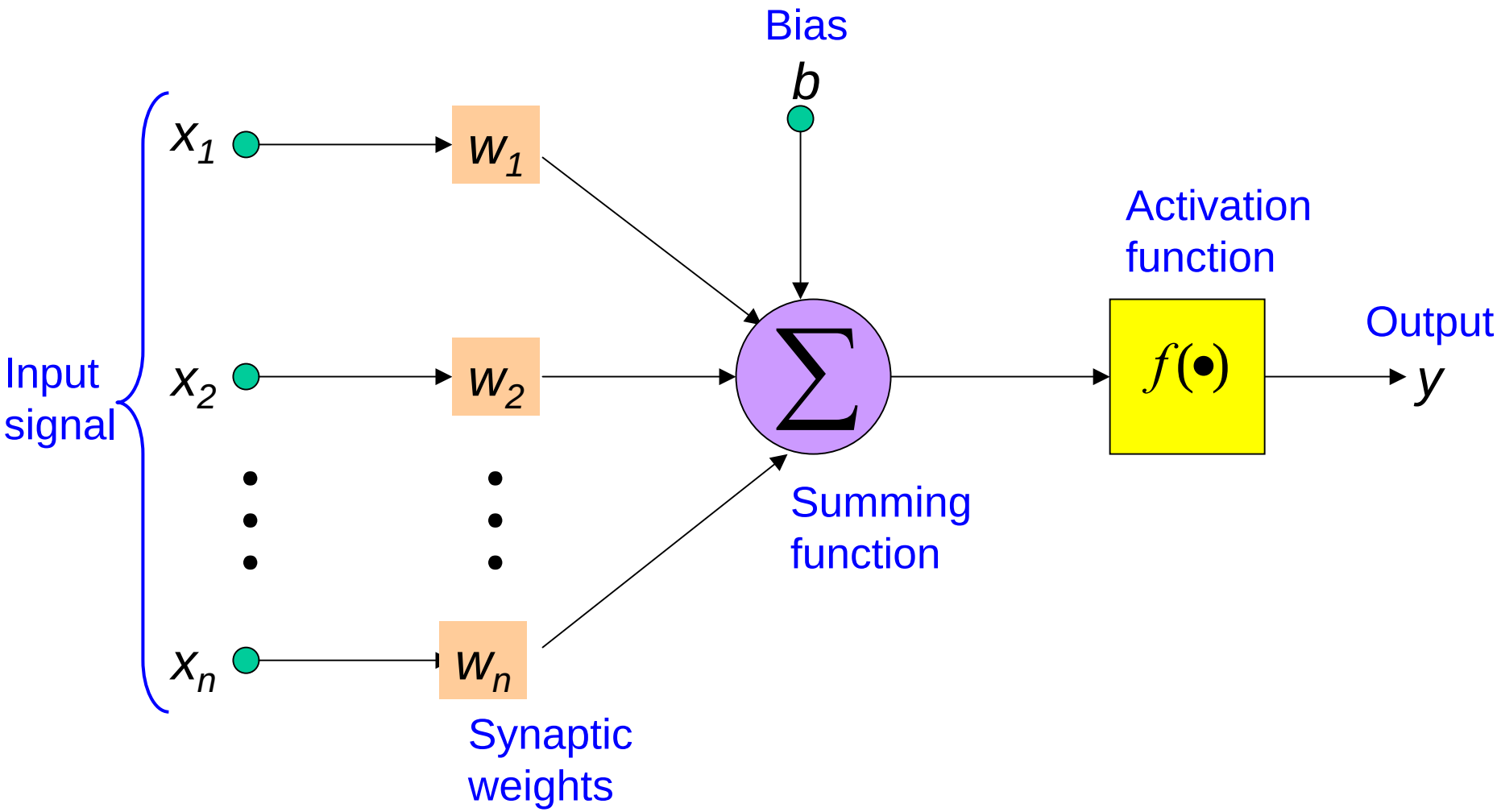
The Perceptron Algorithm

- Linear separation of the input space

$$f(x) = \langle w \cdot x \rangle + b$$

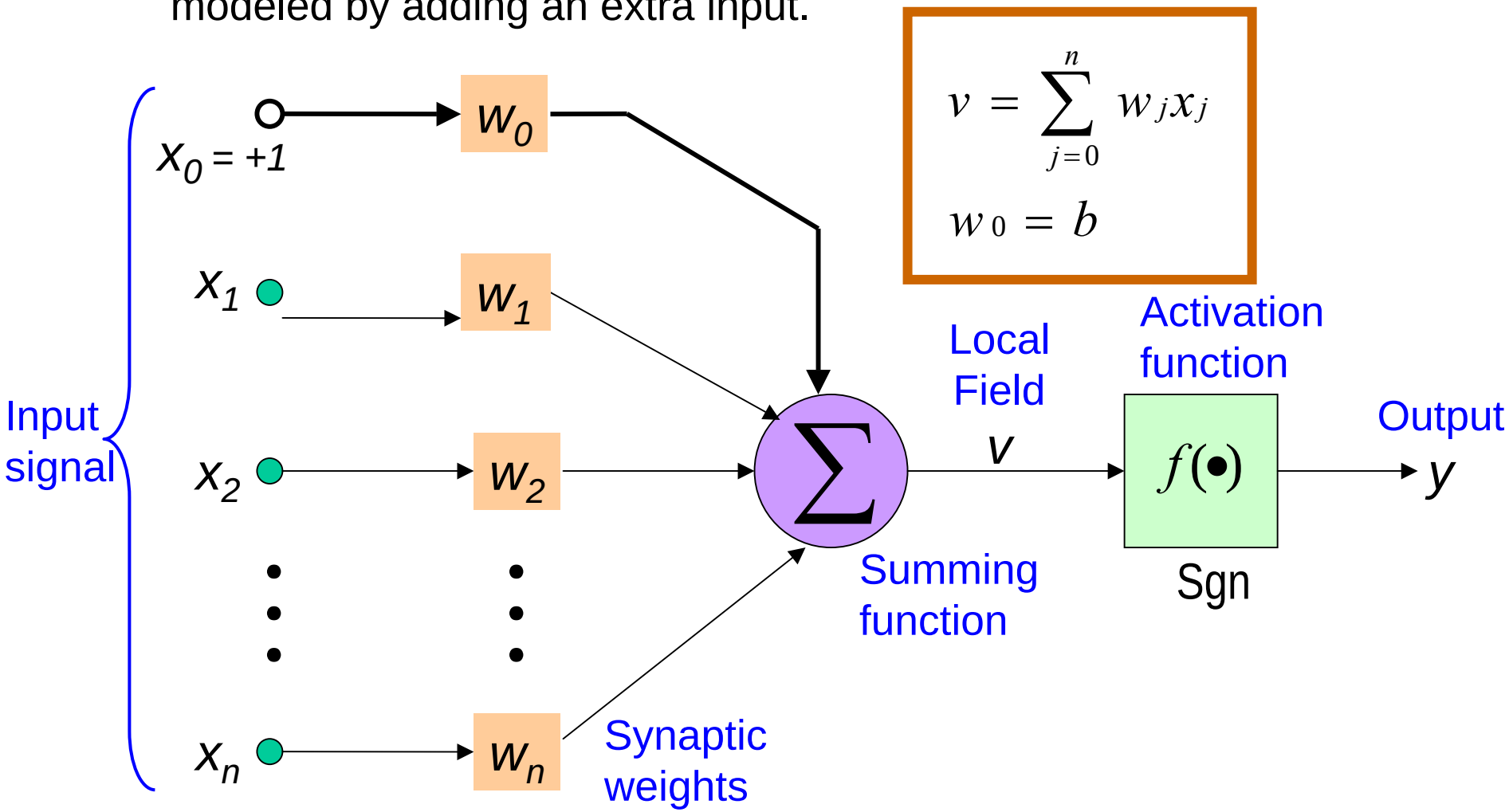


The algorithm requires that the input patterns are linearly separable, which means that there exist linear discriminant function which has zero training error. We assume that this is the case.


Bias
b
Activation function
Output
y
Input signal
x_1
x_2
x_n
w_1
w_2
w_n
$\sum$
$f(\bullet)$
Summing function
Synaptic weights

Bias as extra input

- Bias is an external parameter of the neuron. Can be modeled by adding an extra input.



The Perceptron Algorithm

Repeat

for each training vector pair (x,t)

evaluate the output o when x is the input

if $o \neq t$ then

form a new weight vector w' according

to $w' = w + \eta (t-o) x$

else

do nothing

end if

end for

Until $o = t$ for all training vector pairs

The Perceptron Algorithm

initialize $\mathbf{w}_0 \leftarrow \mathbf{0},\ b_0 \leftarrow 0,\ k \leftarrow 0,\ \eta \in \mathbf{R}^+,\ R \leftarrow \max_i \|\mathbf{x}_i\|$

repeat

error $\longleftarrow$ false

for i=1..l

if $y_i(\langle \mathbf{w}_k, \mathbf{x}_i \rangle + b_k) \le 0$ **then**

$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \eta y_i \mathbf{x}_i$

$b_{k+1} \leftarrow b_k + \eta y_i R^2$

$k \leftarrow k+1$

error $\longleftarrow$ true

end if

end for

until (error==false)

return k,($\boldsymbol{w}_k$,b_k) where k is the number of mistakes

Novikoff theorem

Theorem:

- Suppose that there exists a vector $\mathbf{w}^*, \| \mathbf{w}^* \| = 1$ and a bias term b^* such that the margin on a (non-trivial) data set S is at least γ, i.e.

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \geq \gamma, \quad i = 1, ..., l,$$

then the number of update steps in the perceptron algorithm is at most

$$t^* = \left(\frac{2R}{\gamma} \right)^2,$$

where

$$R = \max_{1 \leq i \leq l} \| x_i \|.$$

Outline

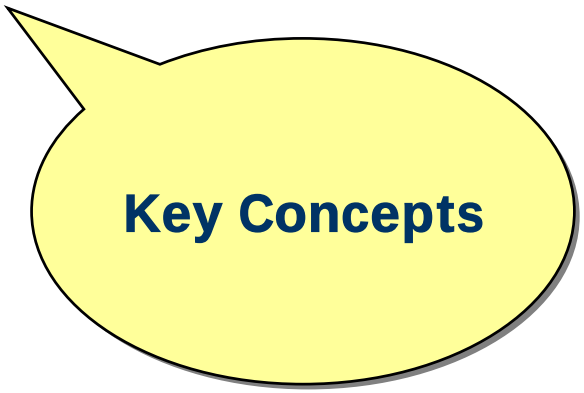
- The Perceptron
- **Preliminaries, Linear classifier & Definition of SVM**
- **Optimization Theory**
- **Non-Linear & Kernel Functions**
- **Implementation**
- **Introduction to NLP applications**

- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVMs introduced by Boser, Guyon, Vapnik in COLT(Computational of Learning Theory)-92
- An important method and active field of all Machine Learning research.
- Special issues of Machine Learning Journal, and Journal of Machine Learning Research.

SVM: A General Definition

- "Support Vector Machines (SVM) are learning systems that use a hypothesis space of **linear functions** in a **high dimensional** feature space, trained with a learning algorithm from **optimisation theory** that implements a **learning bias** derived from statistical learning theory".

(Cristianini & Shawe-Taylor, 2000)



Support Vector Machines: Main Properties & Foundations

- Kernel-induced feature spaces: **high dimensional**
- Learning bias: Generalization bounds for SVMs, e.g., **maximal margin** optimisation. Reduces the danger of overfitting
  - Other margin-based learning algortihms, e.g., AdaBoost **(Schapire 2002; Smola et al., 2000; Rätsch, 2001)**
- Due to Mercer's conditions on the kernels : the optimisation problems are convex $\Rightarrow$ no local minima
- Optimisation theory guides the implementation
- Mainly for **classification** but also for **regression, density estimation, clustering, etc.**
- ...

Preliminaries

- Inner product = dot product = scalar product.

$$\langle \cdot \rangle : \Re^N \to \Re$$

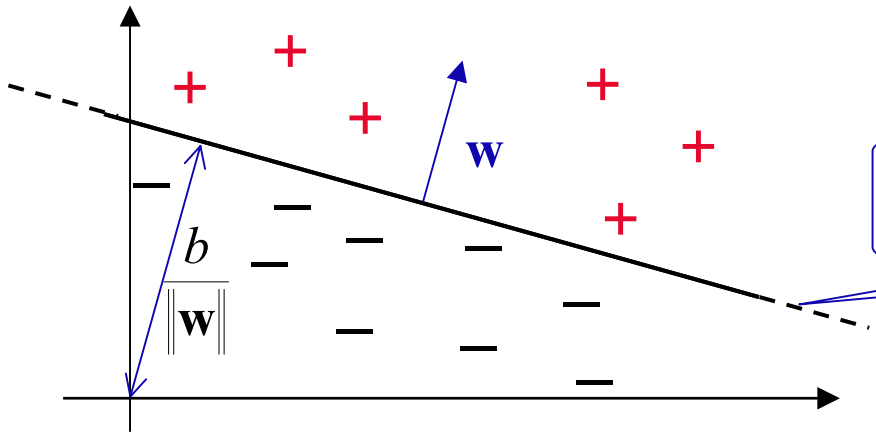
$$\langle \mathbf{x} \cdot \mathbf{z} \rangle = \sum_{i=1}^{N} x_i z_i = x_1 z_1 + x_2 z_2 + \ldots + x_N z_N$$

Linear Classifiers

- **Hyperplanes** in R^n.
- Defined by a **weight vector** (w) and a **threshold** (*b*).
- They induce a **classification** rule:

$$f(\mathbf{x}) = \operatorname{sgn}\left[\sum_{i=1}^{n} w_i x_i + b\right] = \begin{cases} +1 & if \ \sum_{i=1}^{n} w_i x_i + b > 0 \\ -1 & otherwise \end{cases}$$

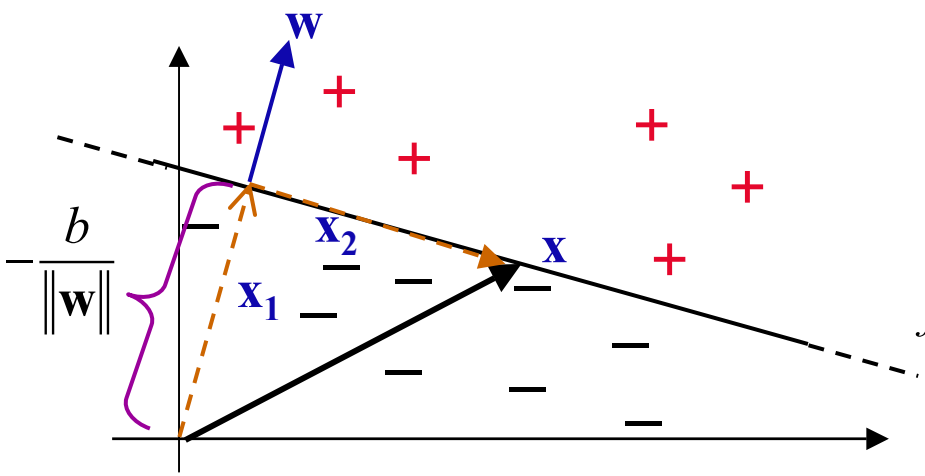
$$\Rightarrow \quad f(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b\right) = \begin{cases} +1 & if \ \langle \mathbf{w} \cdot \mathbf{x} \rangle + b > 0 \\ -1 & otherwise \end{cases}$$



$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + b = \langle \mathbf{w}.\mathbf{x} \rangle + b = 0$$

orthogonal

- **The direction of Hyperplane (w, *b*)** is represented by vector **W;**
- All point **x** on **Hyperplane (w, *b*) is** orthogonal to **(w, *b*)** because
  $<\mathbf{w}.\mathbf{x}> + b = 0$



$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + b = \langle \mathbf{w}.\mathbf{x} \rangle + b = 0$$

$$< \mathbf{w}, \mathbf{x} >= -b, \qquad \text{Let, } \mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$$

$$\Rightarrow \quad < \mathbf{w}, \mathbf{x}_1 > + < \mathbf{w}, \mathbf{x}_2 >= -b$$

$$\Rightarrow \quad < \mathbf{w}, \mathbf{x}_1 >= -b$$

$$\Rightarrow \quad \| \mathbf{x}_1 \| = -\frac{b}{\|\mathbf{w}\|}$$

Margin (I)

- Functional Margin （函数间隔） (γ) of an example ($\mathbf{x_i}$, y_i) w.r.t. a hyperplane ($\mathbf{w}$, b):

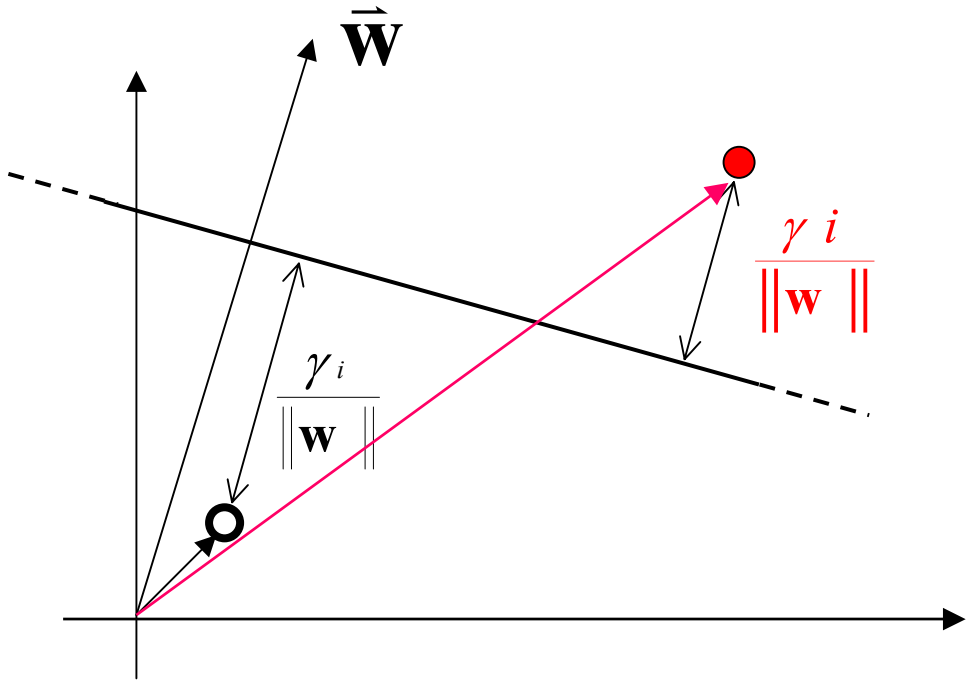
$$\gamma_i = y_i \cdot (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \quad where, \ y_i \in \{-1, \ 1\}$$

- Note that $\gamma_i > 0$ implies correct classification of ($\mathbf{x}_i$, y_i).

- Geometric Margin （几何间隔）:
(normalized hyperplane)

$$\frac{\gamma_i}{\|\mathbf{w}\|}$$

Margin (II): Geometric Interpretation



Distance from a dot $\mathbf{x}'$ to a line (or hyperplane) such as $<\mathbf{w},\mathbf{x}>+b=0$

$$\frac{|<\mathbf{w},\mathbf{x}'>+b|}{\|\mathbf{w}\|} \Rightarrow \frac{y'(<\mathbf{w},\mathbf{x}'>+b)}{\|\mathbf{w}\|} \Rightarrow \frac{\gamma'}{\|\mathbf{w}\|}$$

Margin (III):

- Margin of a training set S w.r.t. a hyperplane($\mathbf{w}$,b):

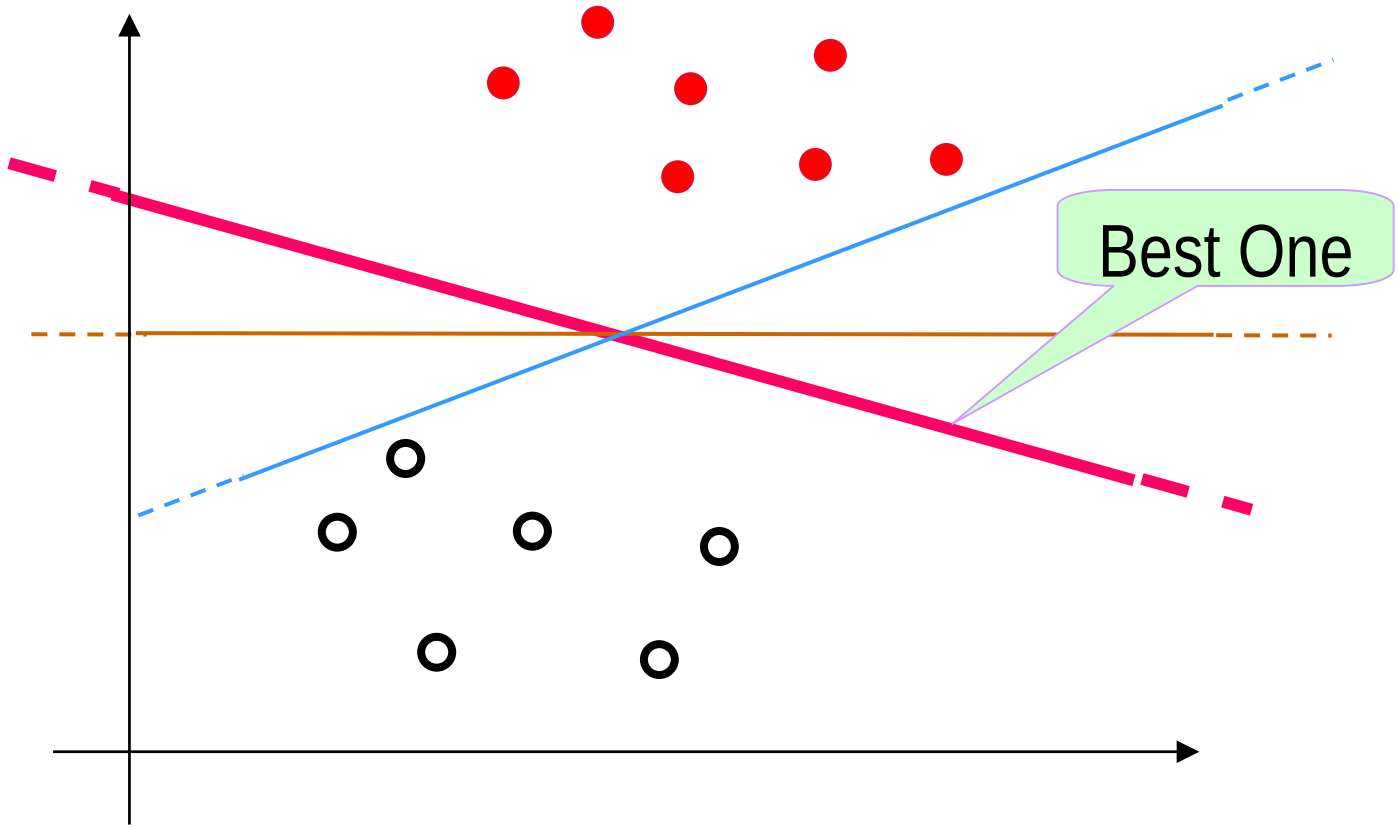
$$\gamma_{S,\mathbf{w},b} = \min_{x_i \in S} \gamma_i$$

- Margin of a training set S :

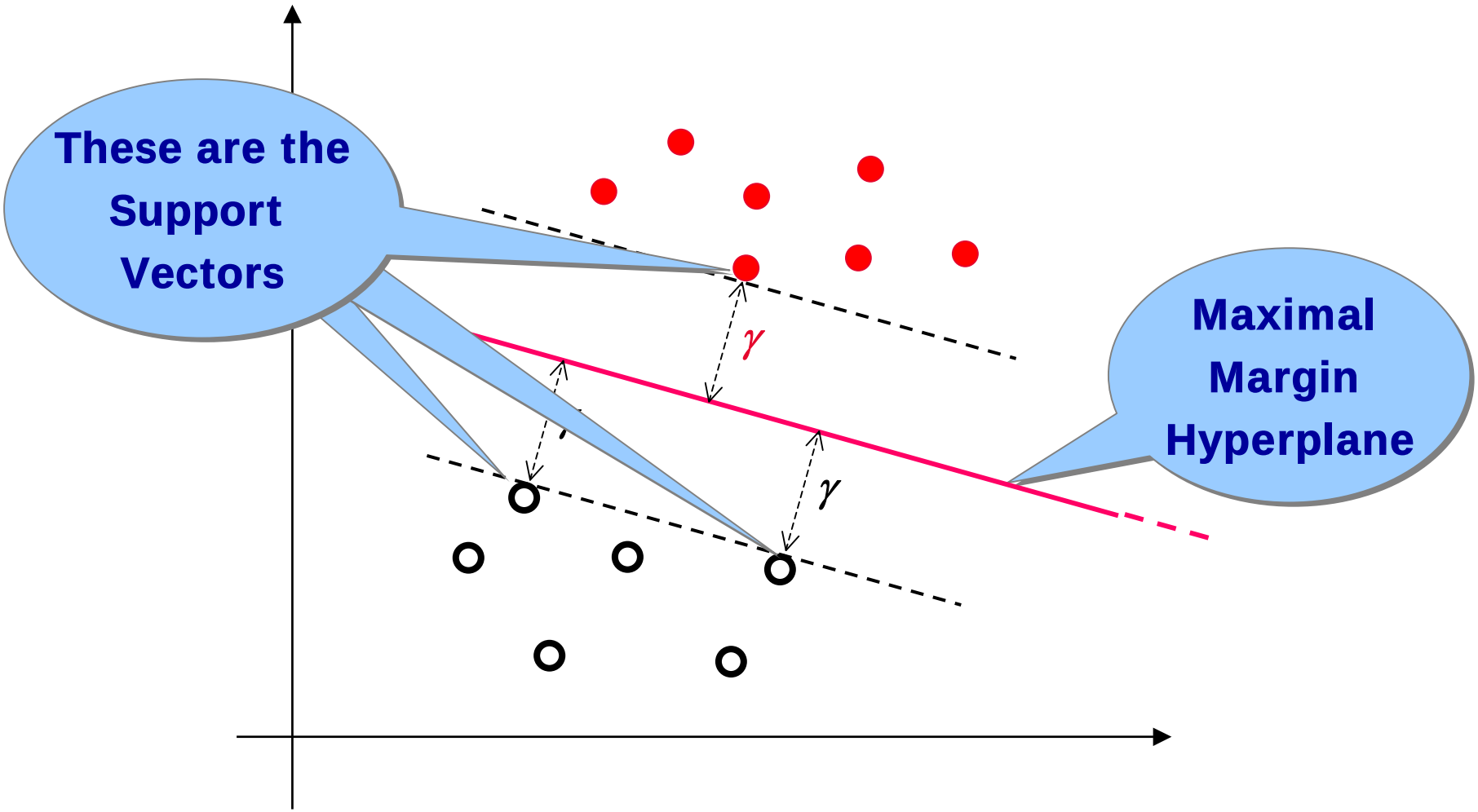
$$\gamma_S = \max_{(\mathbf{w},b)} \gamma_{S,\mathbf{w},b}$$

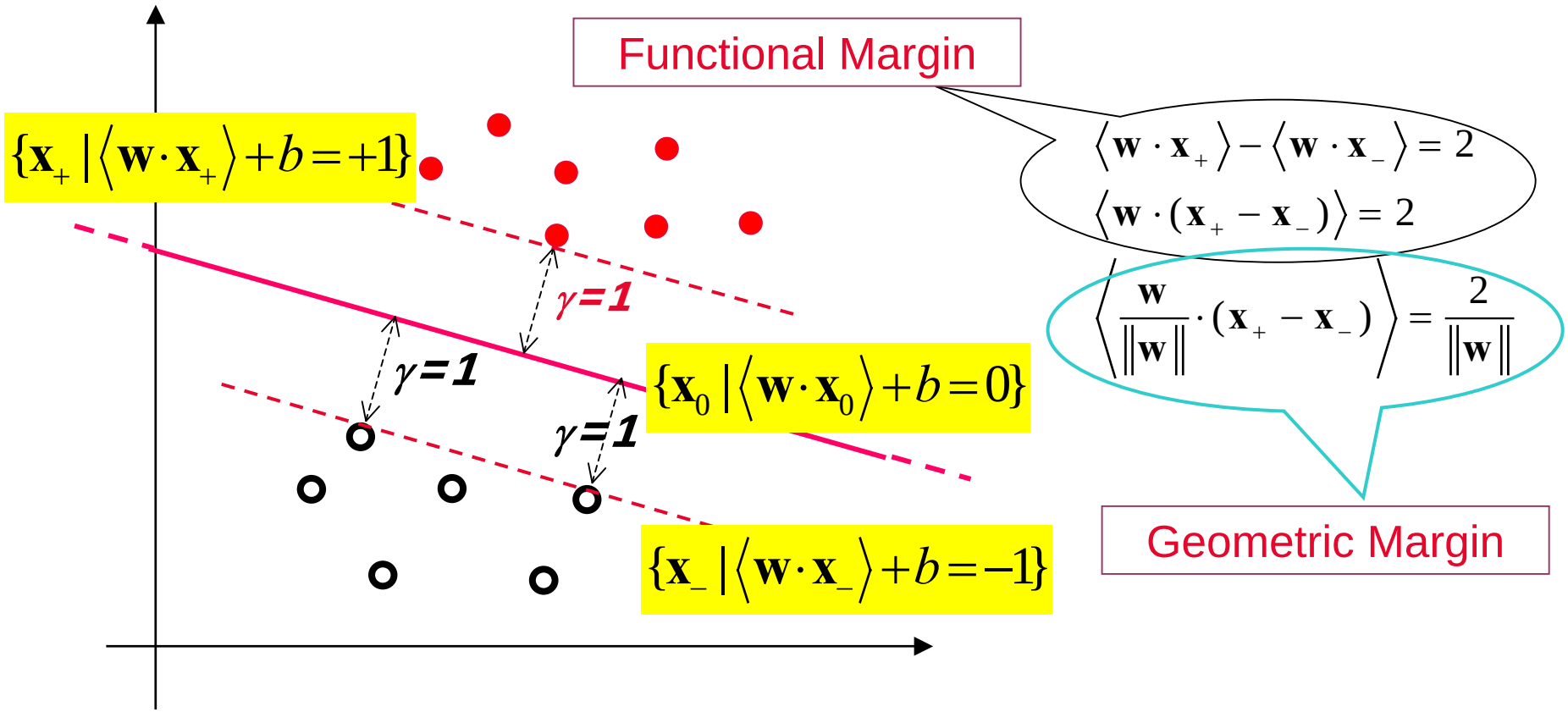
- ... being ($\mathbf{w}$,b) the maximal margin hyperplane.
- in order to minimize the generalization error, we should always pick the model with the maximal margin of the training set.

Optimal Hyperplane: Geometric Intuition



Optimal Hyperplane: Geometric Intuition





Given a hyperplane (**w**,b): $\langle \mathbf{w}\cdot\mathbf{x} \rangle + b = 0$, there is an equivalent hyperplane by rescaling it: ($\lambda\mathbf{w}$, λb), $\lambda \in$ R.

$\langle \lambda\mathbf{w}\cdot\mathbf{x} \rangle + \lambda b = 0$, but functional margin changes by λ.

Let Maximal Margin become 1 by adjusting λ.

Maximal Margin Hyperplane(MMH): how to find

- Note: Geometrical margin remains the same whether the functional margin is rescaled or not.
- Assume (w^*, b^*) is the maximal margin hyperplane.
  The trick: we can fix its functional margin to 1.
  The geometric margin accomplishes:

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

Maximizing Margin ⇔ Minimizing w^* norm

Linear Separable SVMs

- The hyperplane (w,*b*) that solves the optimization problem ...

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle$$

$$\text{subject to} \quad y_i \cdot \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1 \quad , \forall i = 1, \ldots, l$$

Minimizes w norm

⇓

Maximizes geometric Margin

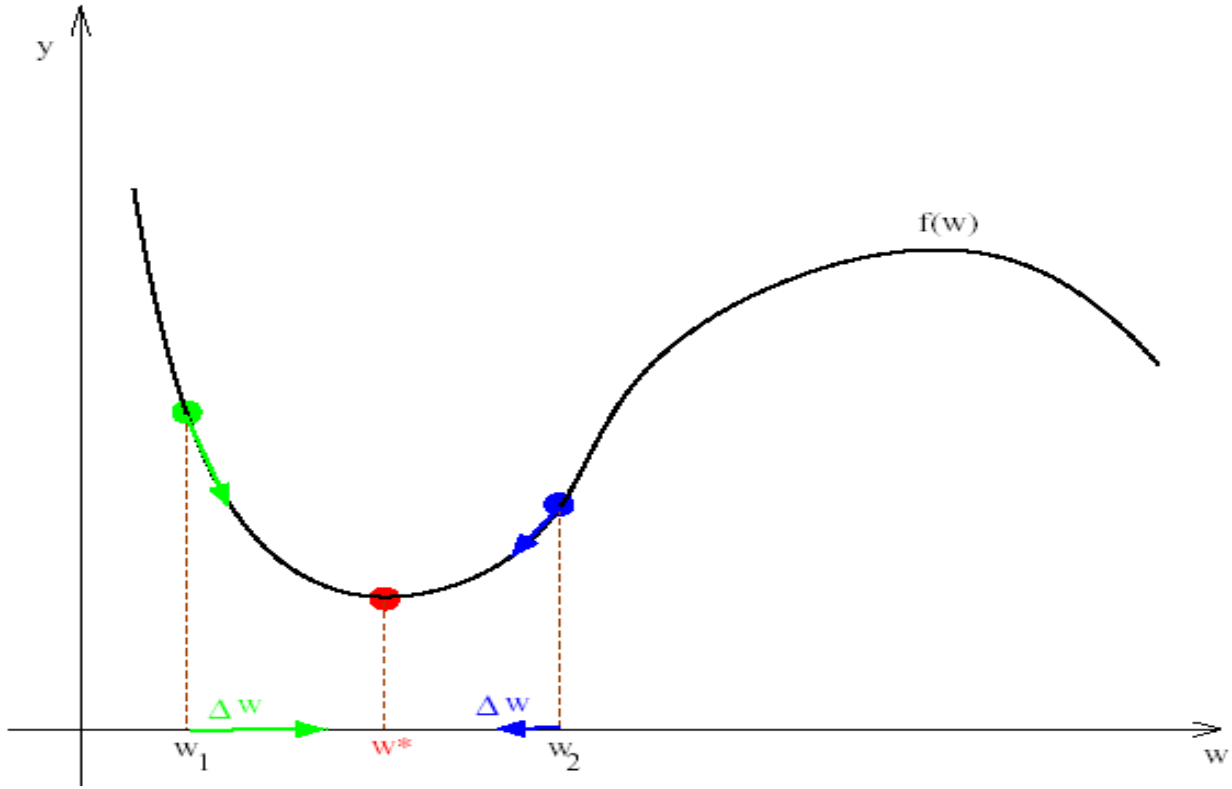
Ensures functional margin is at least 1

Outline

- The Perceptron
- Preliminaries, Linear classifier & Definition of SVM
- **Optimization Theory**
- Non-Linear & Kernel Functions
- Implementation
- Introduction to NLP applications

Optimization Theory - 1D case

- **Go towards maximum :** move in the direction of derivative $+\frac{\partial f}{\partial w}$
- **Go towards minimum : go in the opposite direction** $-\frac{\partial f}{\partial w}$
- **At extremum :** $w^*, f'(w^*)=0$



Optimization Theory - 2D case

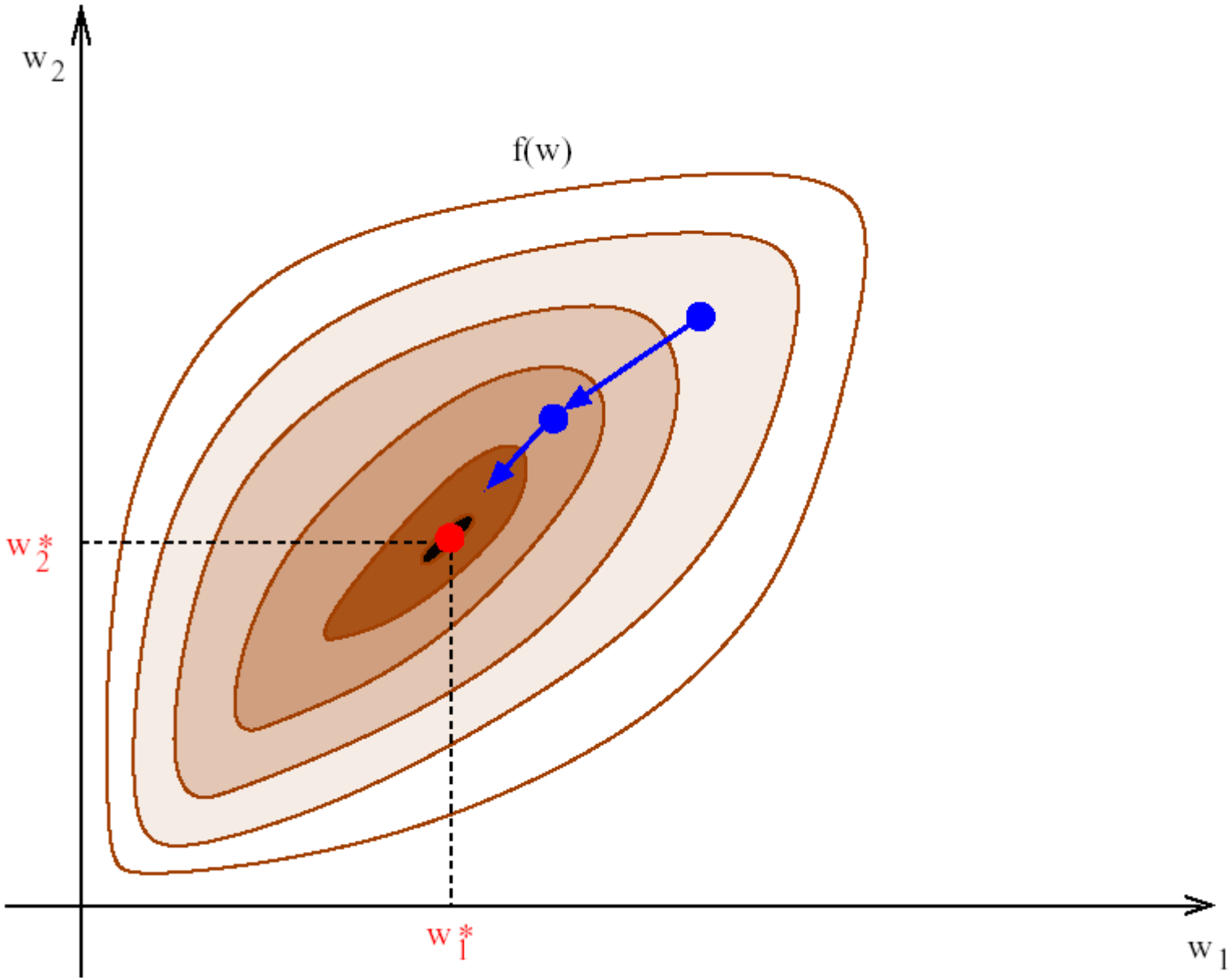
- **Go towards maximum/minimum**

move in the direction/opposite of gradient

$$\pm\left(\frac{\partial f(w)}{\partial w_1},\frac{\partial f(w)}{\partial w_2}\right)$$

- **At extremum w*,**

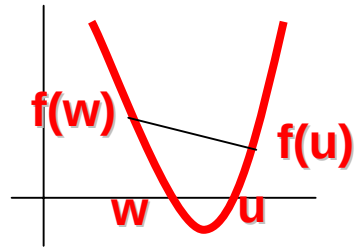
$$\left(\frac{\partial f(w)}{\partial w_1},\frac{\partial f(w)}{\partial w_2}\right)=(0,0)$$


w 2
f(w)
w 2 *
w 1 *
w 1

Convexity

- A real valued function $f(w)$ is called convex for $w \in R^n$ if, $\forall\ w,u \in R^n$, and for any $a \in (0,1)$:

$$f(aw + (1-a)u) \leq a f(w) + (1-a) f(u)$$

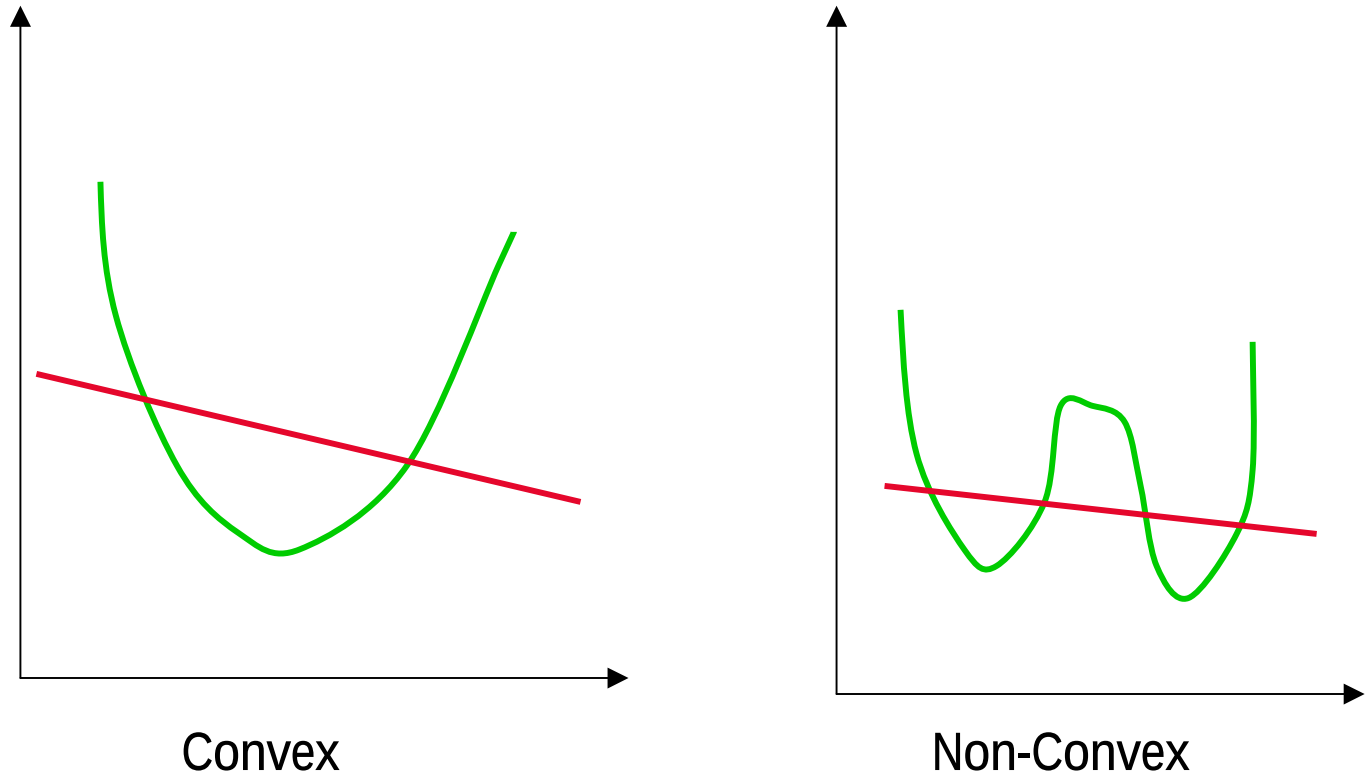


- If a function is convex, any local minimum w^* of the unconstrained optimization problem with objective function f is also a global minimum.
- *Proof*: Take any $u > w^*$ (or $u < w^*$) . By the definition of the local minimum there exists a sufficiently close to 1 that

$$f(w^*) \leq f(aw^* + (1-a)u) \leq a f(w^*) + (1-a) f(u)$$

$$\Rightarrow \quad f(w^*) - a f(w^*) \leq (1-a) f(u)$$

- Where the first inequality is a result of the assumption of local minimum and the second is from the convexity assumption. But that implies (algebra) that $f(w^*) \leq f(u)$
- This is important. It renders optimization problems tractable when the functions involved are convex.



Convexity guarantees a single, global minimum because lower point can be greedily reachable

Solving Optimization Problems

Theorem(Fermat).

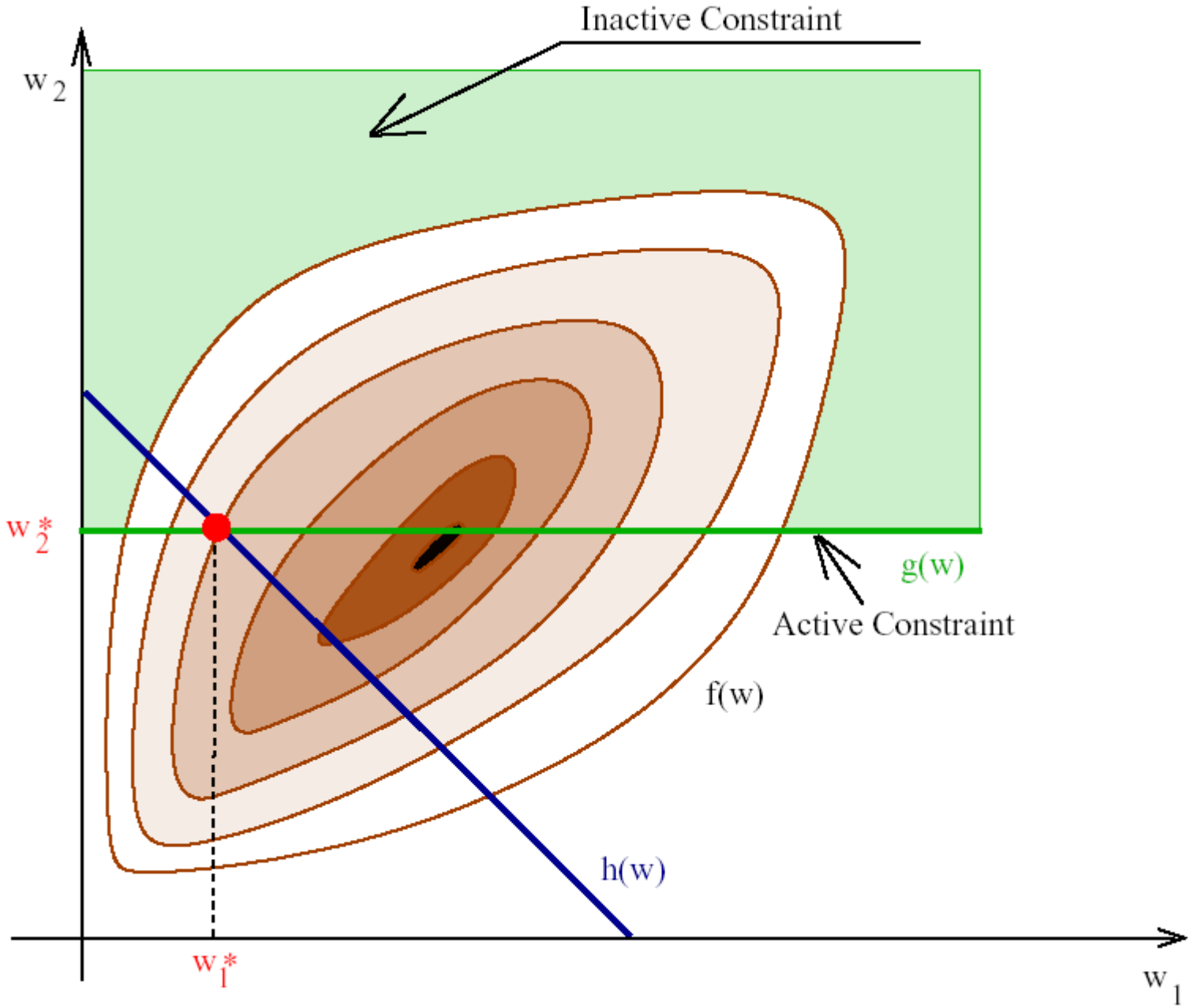
A necessary condition for w^* to be a minimum of *f* , in c^1(一阶连续), is

$$\frac{\partial f(w^*)}{\partial w} = 0$$

This condition, together with convexity, is also a sufficient solution.

Constrained Optimization

- Find minimum of f(w) subject to constraints
- Equality constraints: $h(\mathrm{w}) = 0$
- e.g. $h(\mathrm{w}) = \mathrm{w}_1 + \mathrm{w}_2 - 4 = 0$
- Inequality constraints: $g(\mathrm{w}) \leq 0$
- e.g. $g(\mathrm{w}) = -\mathrm{w}_2 + 3 \leq 0$
- This is actually quite tricky ...


Inactive Constraint
w 2
w 2*
g(w)
Active Constraint
f(w)
h(w)
w 1*
w 1

Constrained Optimization: general form

- The general problem is:

$$\text{Minimize } f(w),\ w \in R^n$$
$$\text{Subject to } g_i(w) \le 0,\ i=1,\ldots k$$
$$h_i(w) = 0,\ i=1,\ldots m$$

- An optimization problem in which the objective function and the constraints are linear is called a linear program.
- If the objective function is quadratic and the constraints linear it is called a quadratic program.

Constrained Optimization Problems

- When the problems are constrained, there is a need to define the Lagrangian function, which incorporates information about both the objective function and the constraints, and which can be used to detect solutions.
- The Lagrangian is defined as the objective function plus a linear combination of the constraints, where the coefficients of the combination are call the Lagrange multipliers.
- Given an optimization problem:

$$\textbf{minimize: } f(\mathrm{w}) \text{ , } \textbf{subject to: } g(\mathrm{w}) \leq 0,\ h(\mathrm{w}) = 0$$

the Lagrangian function is defined as:

$$L(\mathbf{w};\ \alpha,\ \beta) = f(\mathbf{w}) + \alpha\, g(\mathbf{w}) + \beta\, h(\mathbf{w})$$

where the coefficients $\alpha,\ \beta$ are called the Lagrange multipliers.

Constrained Optimization

- Instead of solving

$$\left(\frac{\partial f(w)}{\partial w_1}, \frac{\partial f(w)}{\partial w_2}, \ldots, \frac{\partial f(w)}{\partial w_n} \right) = (0, 0, \ldots, 0)$$

- deal with Lagrangian

$$L(\mathrm{w}; \alpha, \beta) = f(\mathrm{w}) + \alpha g(\mathrm{w}) + \beta h(\mathrm{w})$$

- and solve the dual problem by reasoning about the dual Variables: α, β

Primal and dual problem in Lagrangian

- **Primal problem:**

  minimize $f(\mathbf{w})$ $\quad \mathbf{w} \in R^n$

  subject to $g(\mathbf{w}) \leq 0$, $h(\mathbf{w}) = 0$

- **Dual problem:**

  $\theta(\alpha, \beta) = \inf_{\mathbf{w}} L(\mathbf{w}, \alpha, \beta)$ is minimal value of

  $L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \alpha\, g(\mathbf{w}) + \beta\, h(\mathbf{w})$

w.r.t. w

maximize: $\theta(\alpha, \beta)$

subject to: $\alpha \geq 0$

In general we can have several constraints

- **Primal problem:**

  minimize: $f(\mathrm{w})$

  subject to $g_1(\mathrm{w}) \le 0, g_2(\mathrm{w}) \le 0, \ldots, g_k(\mathrm{w}) \le 0$

  $h_1(\mathrm{w}) = 0, h_2(\mathrm{w}) = 0, \ldots, h_m(\mathrm{w}) = 0$

- **Dual problem:**

  $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$;

  $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$;

  $\theta(\alpha, \beta)$ is minimal value of

  $L(\mathrm{w}, \alpha, \beta) = f(\mathrm{w}) + \Sigma\, \alpha_i\, g_i(\mathrm{w}) + \Sigma\, \beta_j\, h_j(\mathrm{w})$

  w.r.t. w

  maximize: $\theta(\alpha, \beta)$ subject to: $\alpha \ge 0$

Theorem (weak dual theorem)

- If w be a feasible solution to **Primal problem** , and (α, β) a feasible solution to **Dual problem**. Then, $f(\mathbf{w}) \geq \theta(\alpha, \beta)$
- *Proof* : $\theta(\alpha, \beta) = \inf_{\mathbf{u}} L(\mathbf{u}, \alpha, \beta)$

$$\leq L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \alpha\ g(\mathbf{w}) + \beta h(\mathbf{w})$$

$$\leq f(\mathbf{w})$$

Note: $g(\mathbf{w}) \leq \mathbf{0}$ and $h(\mathbf{w})=0$ for feasible solution w, at the same time, $\alpha \geq 0$ for feasible solution (α, β) .

- Corollary 1: $\sup\{\theta(\alpha, \beta): \alpha \geq 0\} \leq \inf\{f(w): g(\mathbf{w}) \leq 0, h(\mathbf{w}) = 0\}$
- Corollary 2: If $f(w^*) = \theta(\alpha^*, \beta^*)$, where, $\alpha^* \geq 0$, $g(\mathbf{w}^*) \leq \mathbf{0}$ and $h(\mathbf{w}^*)=0$, then $\mathbf{w}^*$ and (α^*, β^*) are the solution to primal problem and its dual one, respectively.

Karush-Kuhn-Tucker

minimize: $f(\mathrm{w})$

subject to $g_1(\mathrm{w}) \le 0,\ g_2(\mathrm{w}) \le 0, \ldots,\ g_k(\mathrm{w}) \le 0$

$h_1(\mathrm{w}) = 0,\ h_2(\mathrm{w}) = 0, \ldots,\ h_m(\mathrm{w}) = 0$

on convexity domain $\Omega \subseteq R^n$, where $f \subseteq C^1$, g_i and h_i are affine function, A necessary and sufficient condition for w^* to be a minimum of f is ($L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \alpha\ g(\mathbf{w}) + \beta\, h(\mathbf{w})$):

$$\exists \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, and$$

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \mathbf{w}} = 0, \quad \frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\beta}} = 0$$

$$\begin{cases} \alpha_i^* g_i(\mathbf{w}^*) = 0 \ (either\ \alpha_i^* = 0 \text{ or } g_i(\mathbf{w}^*) = 0) \\ g_i(\mathbf{w}^*) \le 0 \qquad\qquad i = 1, 2, ..., k \\ \alpha_i^* \ge 0; \end{cases}$$

If $g_i(\mathbf{w}^*)$ is inactive, then α_i^* is active, and vice versa.

Again back to SVM

Given a linearly separable training sample

$D = \{(x_1, y_1), \ldots, (x_l, y_l)\} \subseteq (X \times Y)^l$

$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle$

subject to $\quad y_i \cdot \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1, \; or \; 1 - y_i \cdot \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \leq 0; \forall i = 1, \ldots, l$

Note:

Only inequality constraints which are affine in nature

There is a constraint associated with each training point,

Hence there is a dual variable a_i associated with each training point

To solve the problem by Constructing Lagrangian

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\langle\mathbf{w}\cdot\mathbf{w}\rangle-\sum_{i=1}^{l}\alpha_{\mathrm{i}}[y_i(\langle\mathbf{w}\cdot\mathbf{x}_i\rangle+b)\text{-}1]$$

then

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i=1}^{l}\alpha_{\mathrm{i}}y_i\mathbf{x}_i=0,\quad \text{i.e.}\quad \mathbf{w}=\sum_{i=1}^{l}\alpha_{\mathrm{i}}y_i\mathbf{x}_i$$

and

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial\mathrm{b}}=\sum_{i=1}^{l}\alpha_{\mathrm{i}}y_i=0$$

Nature of the solution

- Parameters are expressed as linear combination of training points:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

- By KKT condition, only active constraints lead to non-zero α_i and each α_i corresponds to a training point x_i;
- Only a (usually) small subset of training points will have **non-zero** dual variable α_i and such points are called **support vectors**
- The set of support vectors is denoted be **SV**

$$\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

Dual formulation

Plug $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$ into the Lagrangian,

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{l} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1]$$

$$= \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

$$+ \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \alpha_i y_i b$$

$$= -\frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \sum_{i=1}^{l} \alpha_i$$

$$= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

Linear Separable SVMs
Dual Formulation

Quadratic optimisation problem

$$\text{maximize}_{\boldsymbol{\alpha}} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad and \quad \forall i \; \alpha_i \geq 0$$

Note:

- Non-zero α_i^*'s in the solution $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_l^*)$ correspond to active constraints and determine the support vectors
- α_i^* can be viewed as quantifying the importance of i-th training point (x_i, y_i) for the classifiation task

Optimal Bias

- The bias **b** is determined from the fact that on the decision plane(The maximal margin hyperplane) the functional bias is 1

$$\langle \mathbf{w} \cdot \mathbf{x}_+ \rangle + b = +1$$

$$\langle \mathbf{w} \cdot \mathbf{x}_- \rangle + b = -1$$

$$So,$$

$$b^* = -\frac{1}{2}(\langle \mathbf{w} \cdot \mathbf{x}_+ \rangle + \langle \mathbf{w} \cdot \mathbf{x}_- \rangle)$$

SVM as a classification machine

- Given a new input **x**, its class membership is estimated by

$$Sgn\,[f(\mathbf{x})]$$

$$where,$$

$$f(\mathbf{x}) = f(\mathbf{x}, \mathbf{w}^*) = \left\langle \mathbf{w}^* \cdot \mathbf{x} \right\rangle + b^*$$

$$= f(\mathbf{x}, \alpha^*) = \sum_{i=1}^{l} \alpha_i y_i \left\langle \mathbf{x}_i \cdot \mathbf{x} \right\rangle + b^*$$

$$= \sum_{i \in SV} \alpha_i y_i \left\langle \mathbf{x}_i \cdot \mathbf{x} \right\rangle + b^*$$

Properties of the Dual Form

- Classification rule:

$$\text{sgn}\,[f(\mathbf{x})] = \text{sgn}\left(\sum_{i \in SV} \alpha_i^* y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b^*\right)$$

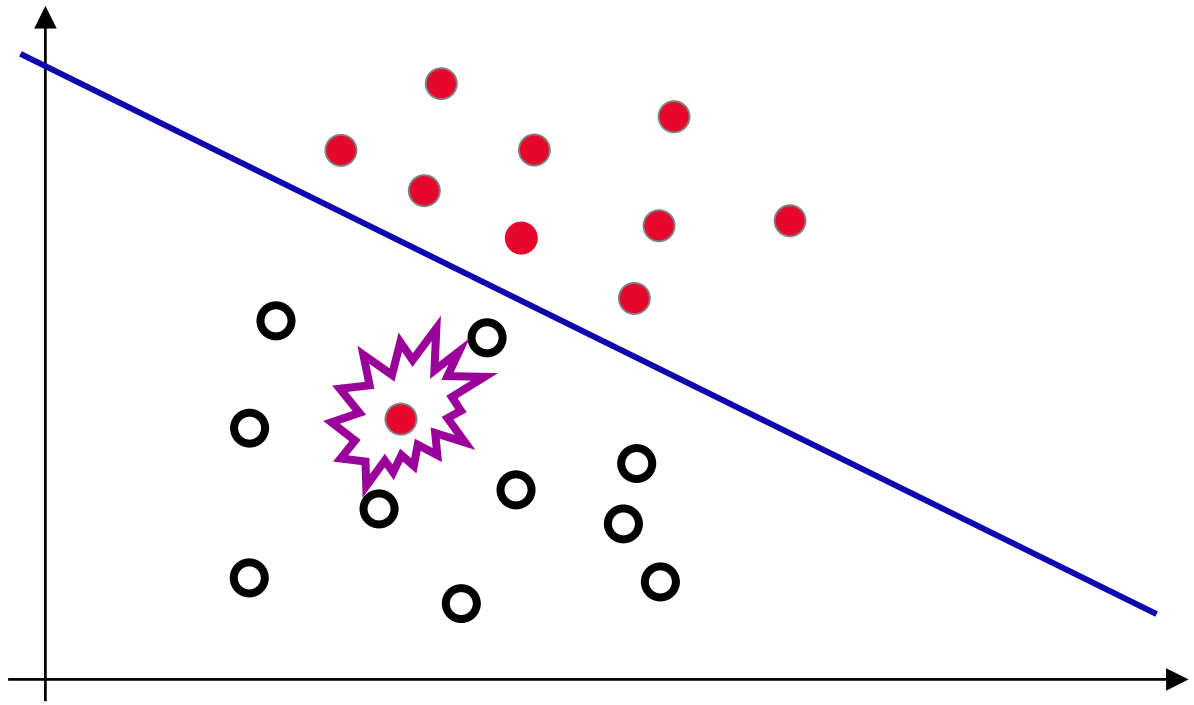
  - Each training example x_i has an associated α_i variable.
    - Describes the influence of the example on the SVM
    - $\alpha_i > 0 \Leftrightarrow x_i$ **is a support vector** ($\alpha_i = 0 \Leftrightarrow$ *other* x_i)
  - Two ways of representing the SVM:
    - Primal: weight vector and threshold (w,*b*)
    - Dual: vector of "influcences" of training set: $\alpha_1, \ldots, \alpha_l$
  - Convex, single solution, solvable in polynomial time.

Outline

- The Perceptron
- Preliminaries, Linear classifier & Definition of SVM
- Optimization Theory
- **Non-Linear & Kernel Functions**
- **Implementation**
- **Introduction to NLP applications**

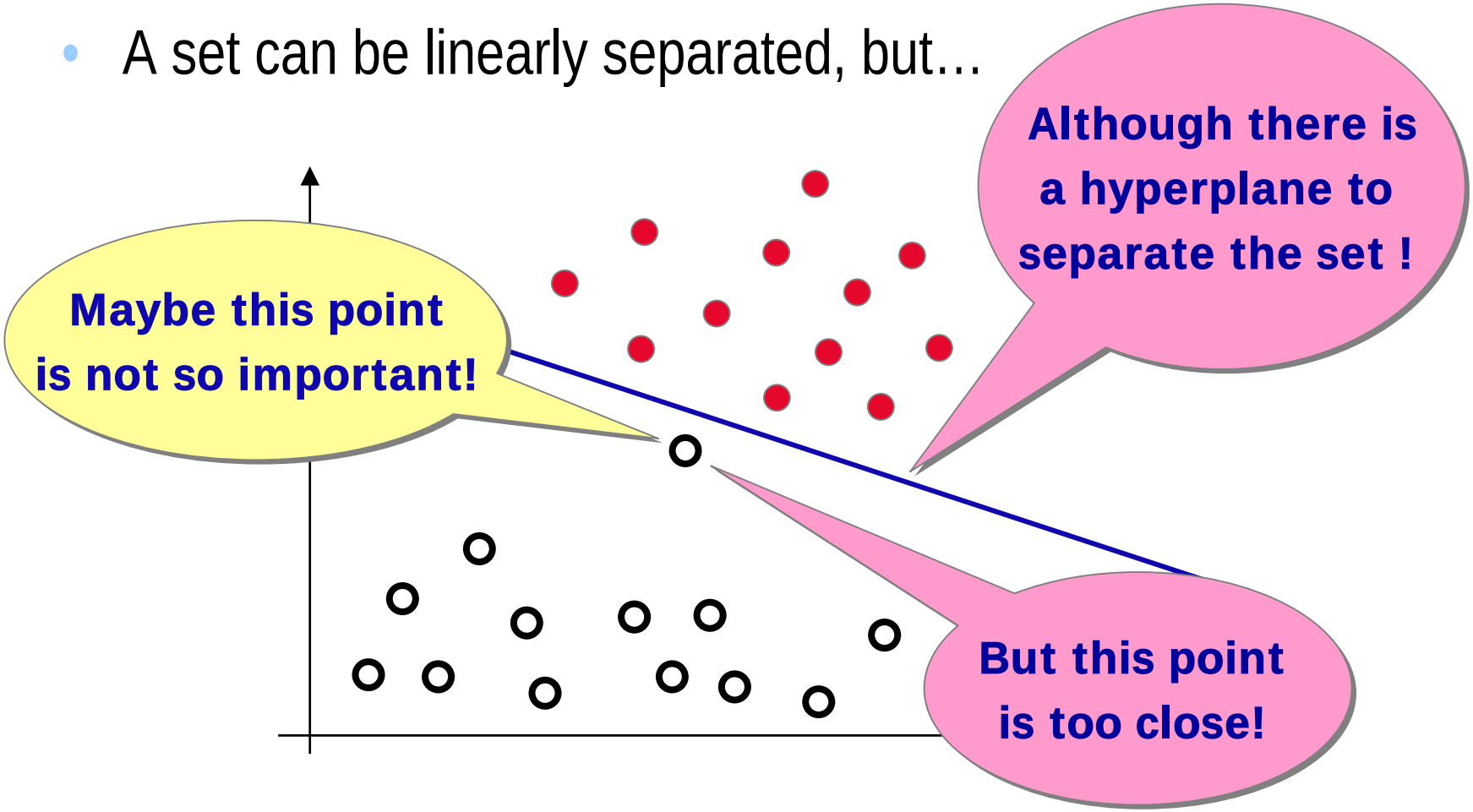
Non-Linear Separable Sets

- Sometimes, data sets are not linearly separable.



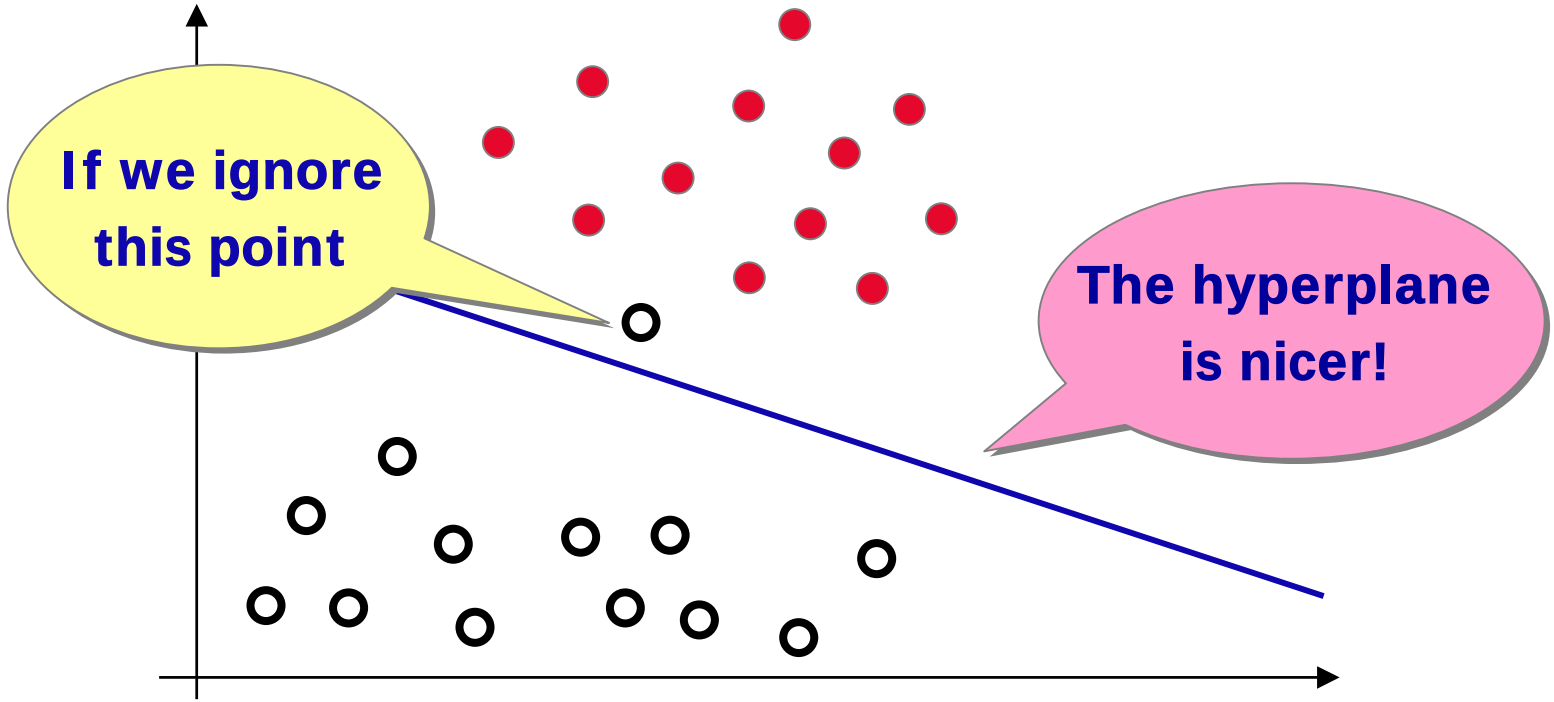
Non-Linear Separable Sets (cont.)

- A set can be linearly separated, but…



Non-Linear Separable Sets (cont.)

- Sometimes, we do not want to separate perfectly.



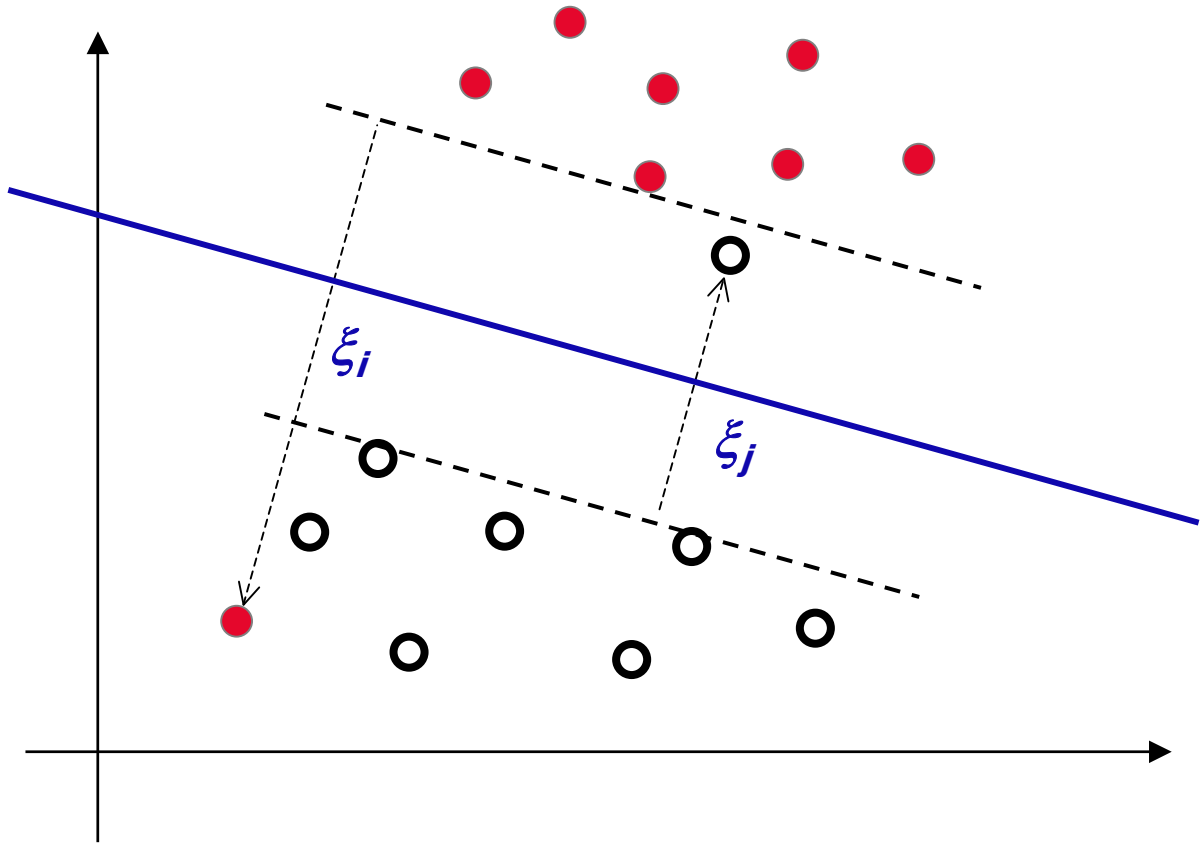
Soft Margin SVMs

- We allow misclassifications of examples.
- Slack variables (ξ_i) are introduced, relaxing the margin constraints to:

$$y_i\left(\left\langle \mathbf{w} \cdot \mathbf{x}_i \right\rangle + b\right) \geq 1 - \xi_i \quad , \forall\, i = 1, \ldots, l$$

$$\xi_i \geq 0 \quad , \forall\, i = 1, \ldots, l$$

Soft Margin SVMs



Soft Margin SVMs (Primal Formulation)

- The optimization problem becomes:

$$\text{minimize}_{\mathbf{w},b,\xi} \quad \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i^2$$

$$\text{subject to} \quad y_i \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i \ , \ \forall i = 1, \ldots, l$$

$$\xi_i \geq 0 \ , \forall i = 1, \ldots, l$$

- C determines the trade-off between margin maximization and training error minimization.

Soft Margin SVMs (Dual Formulation)

$$\text{maximize}_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \qquad \text{and} \qquad \forall i \quad C \geq \alpha_i \geq 0$$

Optimal solution

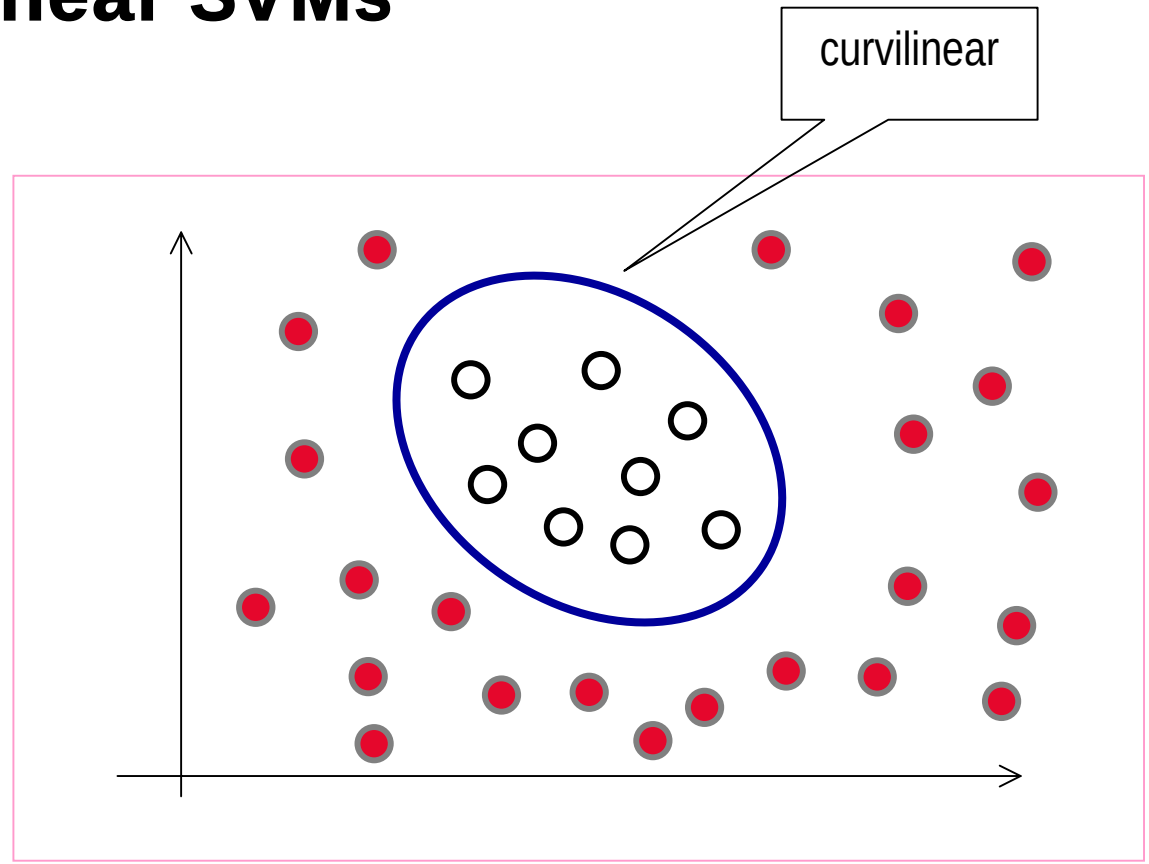
Karush-Kuhn-Tucker conditions. For each example x_i:

$\alpha_i = 0 \quad \Rightarrow y_i f(x_i) \geq 1$ and $\xi_j = 0$ (x_i is not a SV)

$0 < \alpha_i < C \quad \Rightarrow y_i f(x_i) = 1$ and $\xi_i = 0$ (x_i is a SV)

$\alpha_i = C \quad \Rightarrow y_i f(x_i) \leq 1$ and $\xi_l > 0$ (x_i is a SV)

Non-linear SVMs



- How can we separate this data set with SVMs?

Non-linear SVMs

- One solution: creating a net of simple linear classifiers (neurons): a Neural Network

  (problems: local minima; many parameters; heuristics needed to train; etc)

- Other solution: map data into a richer feature space including non-linear features, then use a linear classifier

- Working in high dimensional feature spaces allows to express complex functions, but... some possible problems:
  - Computational problem working with very large vectors
  - The curse of dimensionality, risk of overfitting

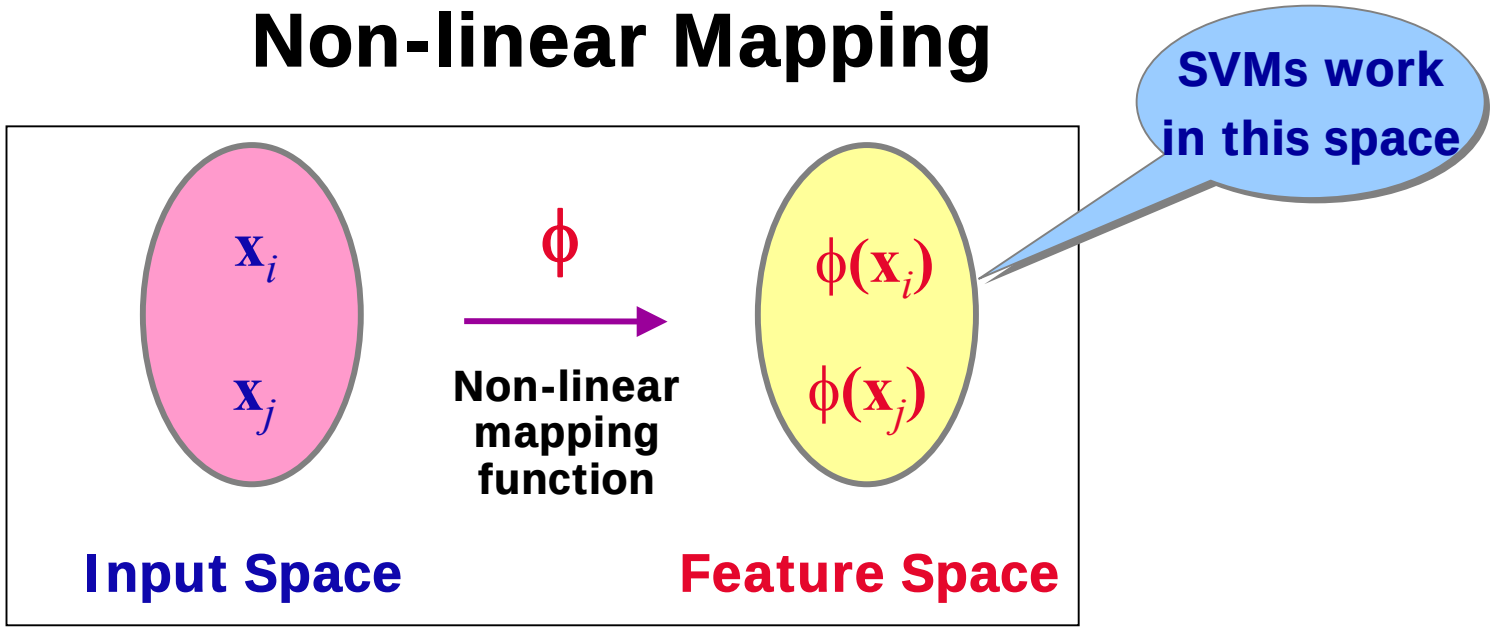
Note an important property

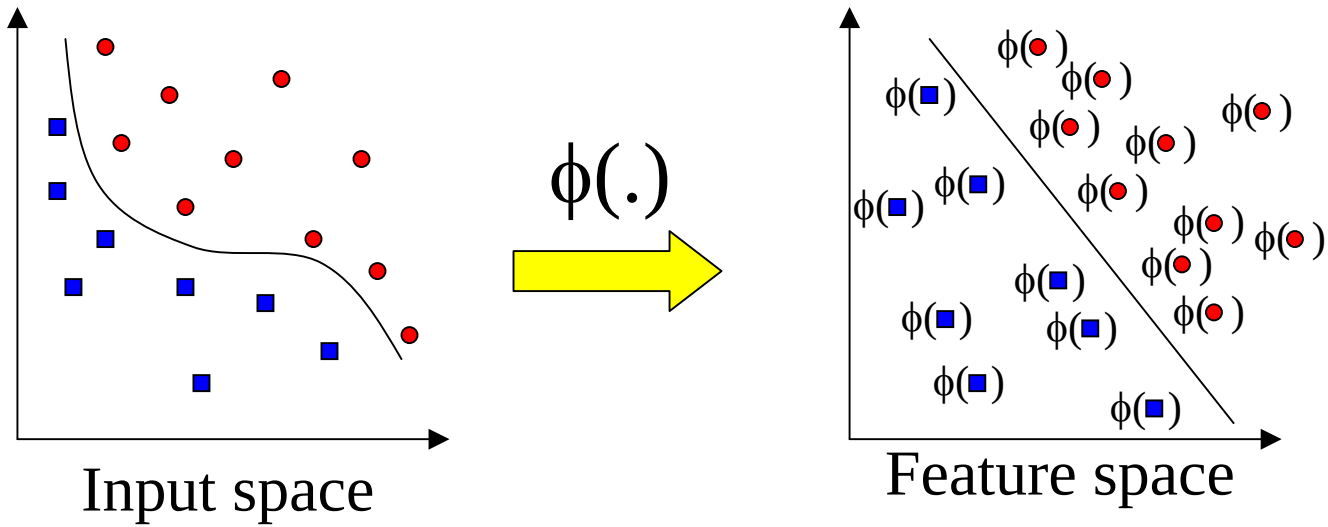
$$\text{maximize} \quad \max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \;\; C \geq \alpha_i \geq 0$$

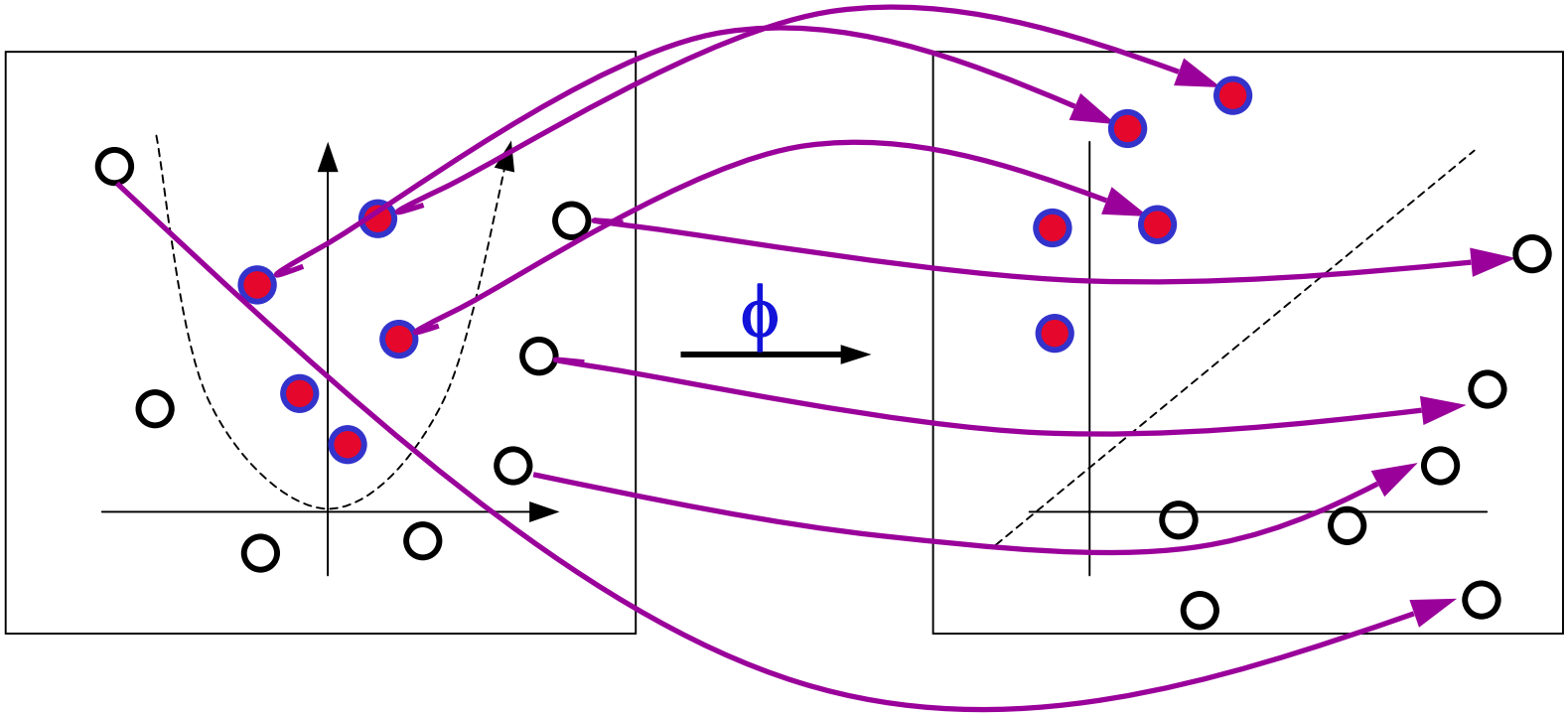
- The inner product is the only operation performed on training examples !
- We can use `more general' dot products !

Non-linear Mapping





Non-linear Mapping



Linear separable with Non-linear Mapping

$$\text{maximize}_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \quad C \geq \alpha_i \geq 0$$

- If there exists a function $K(x_i, x_j)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

$K(x_i, x_j)$ is called Kernel function(Note: $K(x_i, x_j) = K(x_j, x_i)$)

- Given a function K, it is possible to verify that it is a kernel
- The mapping function makes data linearly separable
- No need to represent the Feature Space by $\phi(x)$ explicitly
- The kernel function allows the SVM to work implicitly in the feature space

Kernel Trick

Consider 2-Dim inputs: $\mathbf{x} = (x_1, x_2) \in R^2$

and a function $K(\mathbf{x},\mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^2$

$$K(\mathbf{x},\mathbf{z}) = (x_1 z_1 + x_2 z_2)^2$$

$$= (x_1 z_1)^2 + 2(x_1 z_1 x_2 z_2) + (x_2 z_2)^2$$

$$= \langle (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2) \cdot (z_1 z_1, z_1 z_2, z_2 z_1, z_2 z_2) \rangle = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$$

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \cdot \left(z_1^2, z_2^2, \sqrt{2} z_1 z_2 \right) \right\rangle = \langle \varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) \rangle$$

where $\phi(\mathbf{x}) = (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2)$

$\varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$

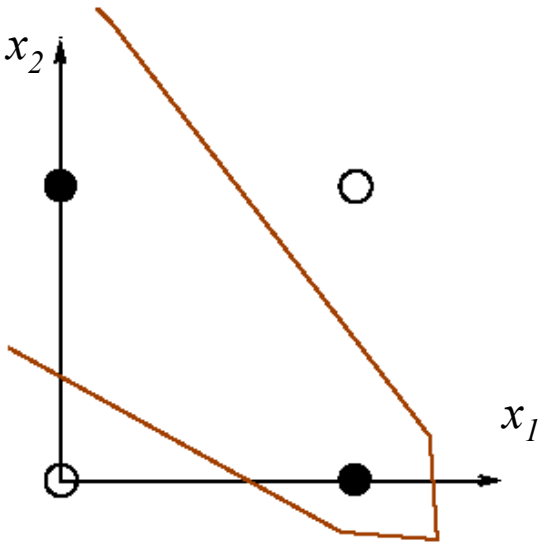
are both vector monomials of degree 2

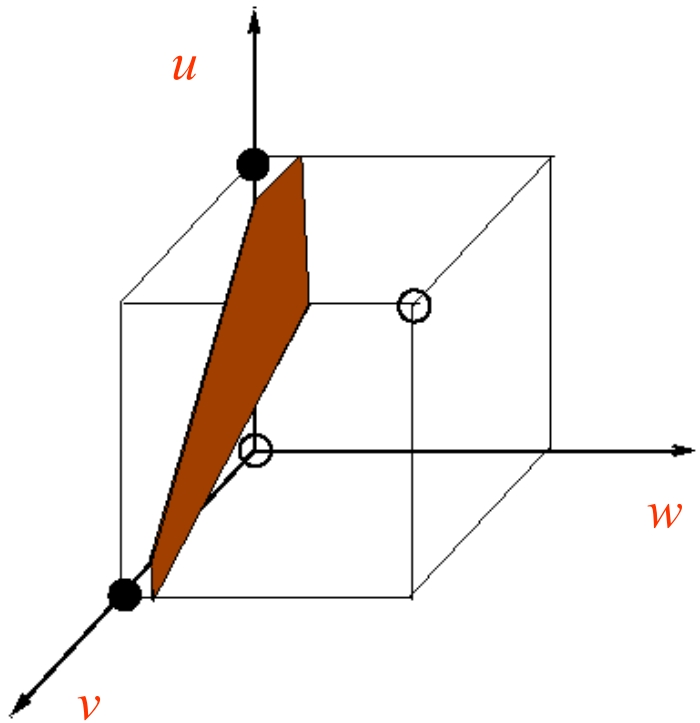
Feature Space

- Function $K(x,z) = \langle x \cdot z \rangle^2$ is not a dot product in the input space R^2.
- but it can be considered a dot product between non-linear images $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ of $\mathbf{x}$, $\mathbf{z}$.
- The images live in a 4-dimensional feature space.
- Non-linear transformation like this can be useful in many applications.
- The dimension changes but the number of examples remain unchanged.
- Having a suitable kernel function $K(x,z) = \langle x \cdot z \rangle^2$, simply plug $K(x,z)$ instead of $\langle x \cdot z \rangle$

Mapping to higher dimensions is good

- Note: $K(x,z) = \langle x \cdot z \rangle^2 = \langle \varphi(x) \cdot \varphi(z) \rangle$
- XOR problem is not a linearly separable problem in R^2. however using φ it is linearly separable in R^3





φ: $(x_1, x_2) \Rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

$u \quad v \quad w$

Kernel Functions

- *One of the curious facts about using a kernel is that we do not need to know the underlying feature map in order to be able to learn in the feature space!* (Cristianini & Shawe-Taylor, 2000)
- But… in practice the kernel function must be efficiently computable

Kernel Function (Mercer's Theorem)

- Any function $K(\mathbf{x},\mathbf{z})$ that creates a symmetric, positive definite matrix $K_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$ is a valid kernel (= an inner product in some space)
- Kernel (Gram) matrix:

$$\begin{pmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \cdots & K(\mathbf{x}_1,\mathbf{x}_l) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_l) \\ \cdots & & & \cdots & \\ \cdots & & & \cdots & \\ K(\mathbf{x}_l,\mathbf{x}_1) & K(\mathbf{x}_l,\mathbf{x}_2) & K(\mathbf{x}_l,\mathbf{x}_3) & \cdots & K(\mathbf{x}_l,\mathbf{x}_l) \end{pmatrix}$$

SVMs with kernels

- Training

$$\text{maximize} \quad \alpha \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \quad C \geq \alpha_i \geq 0$$

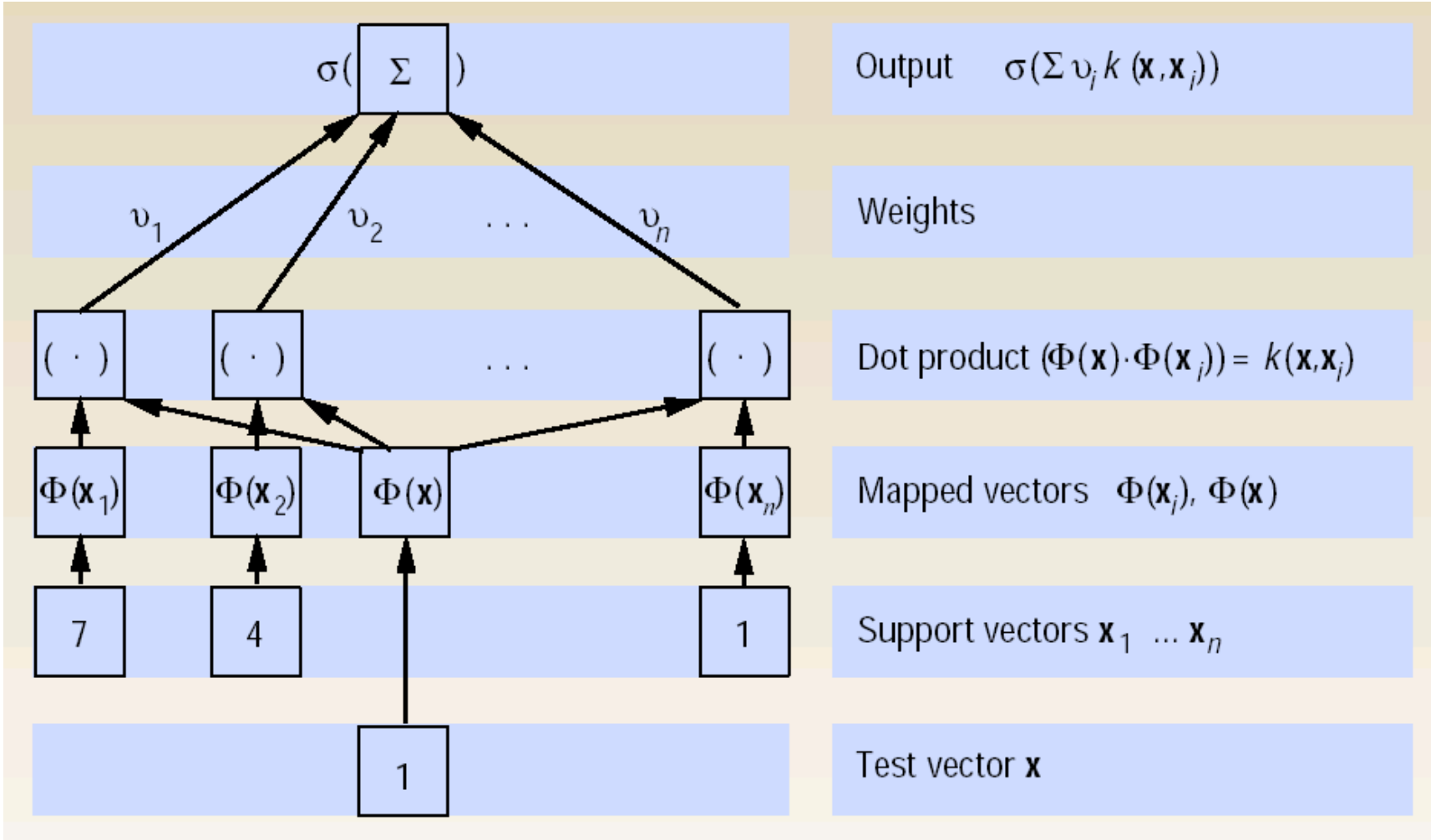
- Classification of x:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

$$= sgn\left(\sum_{i=1}^{l} v_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

$$where, \quad v_i = \alpha_i \cdot y_i \quad \text{stands for weight of} \quad \mathbf{x}_i$$

$$f(\mathbf{x}) = \sigma(\sum_{i=1}^{i=n} v_i K(\mathbf{x}, \mathbf{x}_i)) = \mathrm{sgn}(\sum_{i=1}^{i=n} v_i K(\mathbf{x}, \mathbf{x}_i) + b)$$



Usual kernels

Polynomial:

$$K(\mathbf{x},\mathbf{z}) = \left(\langle \mathbf{x}\cdot\mathbf{z}\rangle + c\right)^d$$

Gaussian Kernel (Radial Basis Function, RBF):

$$K(\mathbf{x},\mathbf{z}) = e^{\left(-\|\mathbf{x}-\mathbf{z}\|^2/2\sigma^2\right)}$$

$$K(\mathbf{x},\mathbf{z}) = \tanh\left(\kappa\langle \mathbf{x}\cdot\mathbf{z}\rangle + \Theta\right)$$

Parameters that the user must choose

where, $\tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

Making Kernels

- The set of kernels is closed under some operations.
- New kernels can be made from valid kernels by allowed operations, e.g. If K and K' are kernels then:
  - $K+K'$ is a kernel
  - cK is a kernel, if $c>0$
  - $aK+bK'$ is a kernel, for $a,b>0$

A bad kernel

- Would be a kernel whose kernel matrix is mostly diagonal: all points orthogonal to each other, no clusters, no structure….

1	0	0	…	0
0	1	0	…	0
		1		
…	…	…	…	…
0	0	0	…	1

- In mapping in a space with too many irrelevant features, kernel matrix becomes diagonal
- Need some **prior knowledge** of target so choose a good kernel

Outline

- The Perceptron
- Preliminaries, Linear classifier & Definition of SVM
- Optimization Theory
- Non-Linear & Kernel Functions
- **Implementation**
- **Introduction and NLP applications**

Training a SVM

- Given a training set with *l* examples:

$$\text{maximize}_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \quad C \geq \alpha_i \geq 0$$

- Give value to *l* α variables, in a quadratic problem
- Consider a matrix of l^2 kernel evaluations: gram/kernel matrix

Training a SVM

- Kernel Matrix

$$\begin{pmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \cdots & K(\mathbf{x}_1,\mathbf{x}_l) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_l) \\ \cdots & & & \cdots & \\ \cdots & & & \cdots & \\ K(\mathbf{x}_l,\mathbf{x}_1) & K(\mathbf{x}_l,\mathbf{x}_2) & K(\mathbf{x}_l,\mathbf{x}_3) & \cdots & K(\mathbf{x}_l,\mathbf{x}_l) \end{pmatrix}$$

- Information “bottleneck”: it contains all the necessary information for the learning algorithm
- It joins information about the data and the kernel
- l^2 can be extremely large in real problems with thousands of examples

Training a SVM

- Traditional Quadratic Programming (QP) methods can be used to solve the quadratic problem.
- Techniques have been developed to cope with high dimensional training sets:
  - Chunking, Decomposition, Random Sampling, etc.
- In general, they address the whole problem by solving many small sub-problems: select/solve/update
- Covergence? KKT conditions
- Programming tricks for speeding-up: shrinking, caching, etc.

Example

- Suppose we have 5 1D data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

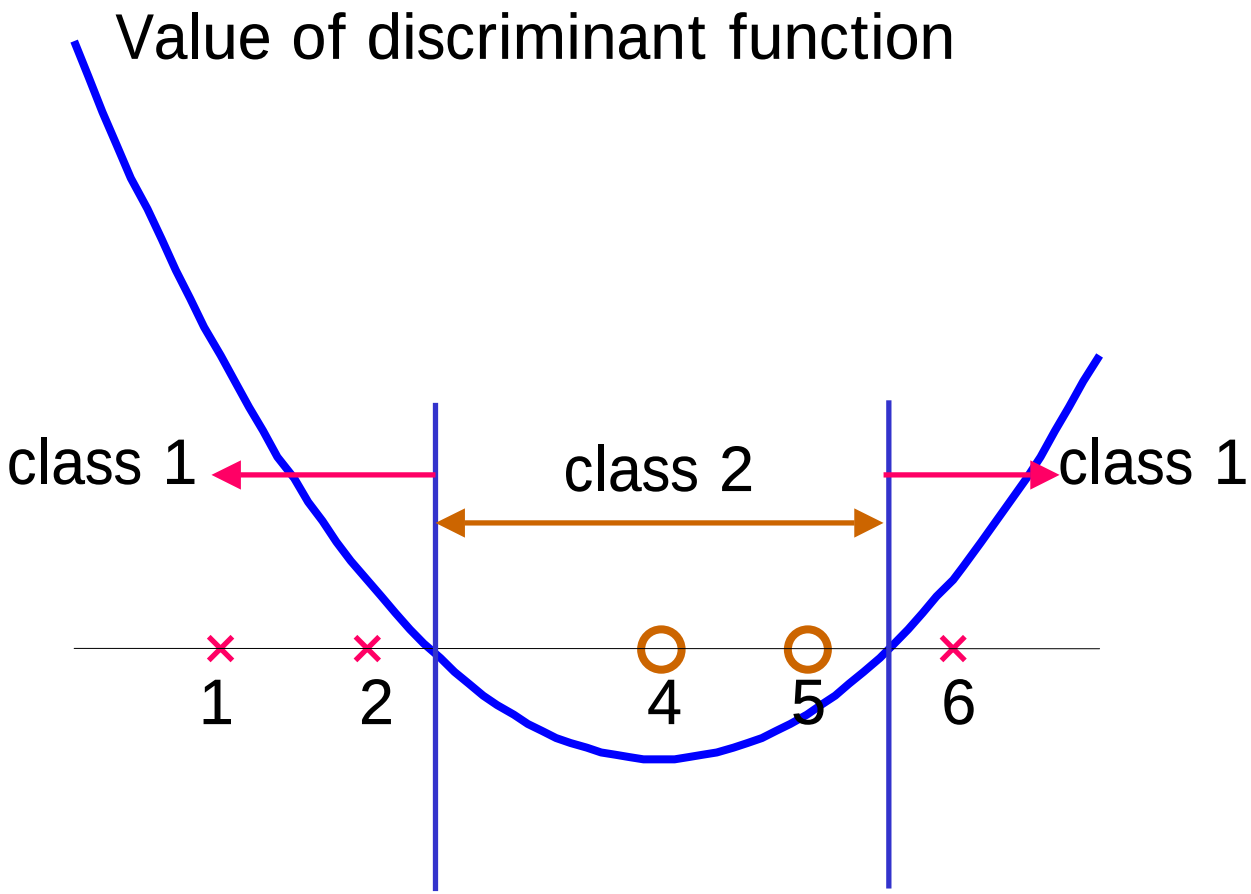
- By using a QP solver, we get
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - Note that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
- The discriminant function is

$$\begin{aligned} & f(y) \\ &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- b is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T\phi(z)+b)=1$ and all give b=9

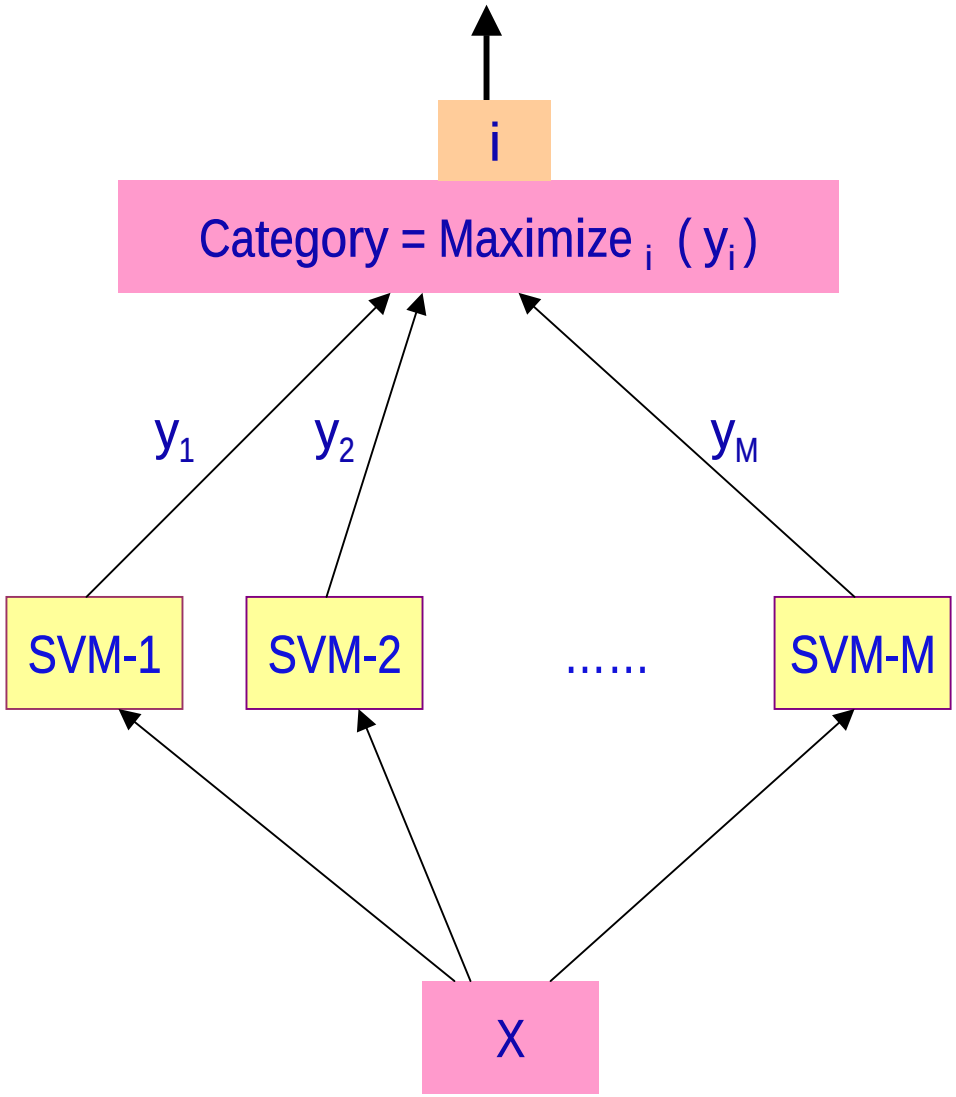
$$\rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Example


Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output with arity 2).
- What can be done for more than 2 categories?
- Answer: with output arity M, learn M SVM's
  - SVM-1 learns "Output==1" vs "Output != 1"
  - SVM-2 learns "Output==2" vs "Output != 2"
  - ...
  - SVM-M learns "Output==M" vs "Output != M"
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.


i
Category = Maximize i (yi)
y1
y2
yM
SVM-1
SVM-2
.....
SVM-M
X

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly**
- Weaknesses
  - Need to choose a “good” kernel function. It is difficult.

Outline

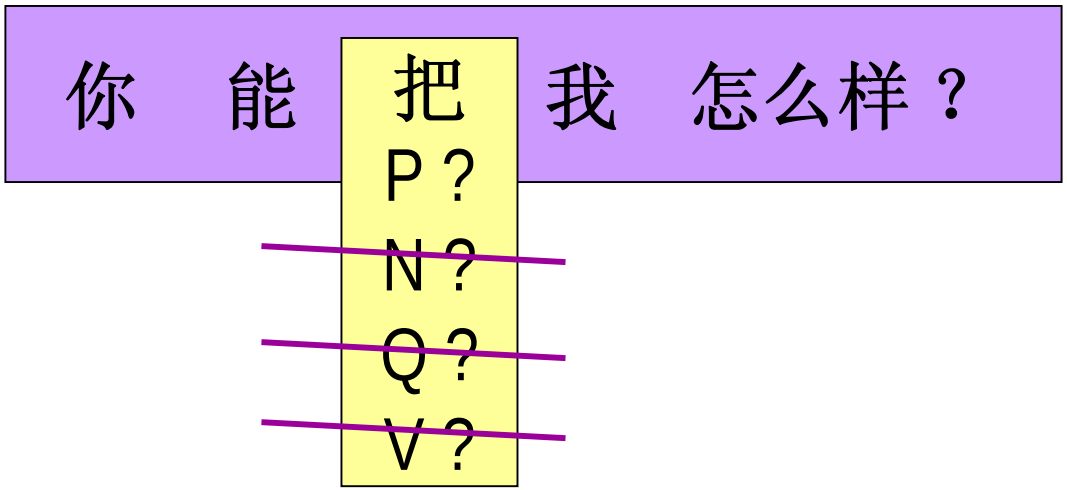
- The Perceptron
- Preliminaries, Linear classifier & Definition of SVM
- Optimization Theory
- Non-Linear & Kernel Functions
- Implementation
- **Introduction to NLP applications**

SVMs in NLP : problems

- Part-of-speech tagging
- Word Sense Disambiguation
- Text Categorization
- …

POS in Chinese: disambiguate

- “把”
  - Preposition: 他 把 书 放回 到 了 原处
  - Noun：他 的 锹 把 断 了
  - Unit (Quantity)：我 想 买 一 把 锁
  - Verb：中医 靠 把 脉 诊断 病情
  - ….



Word Sense Disambiguation

- 动词：“冲”
  - 他 端 着 枪 向 前 冲
  - 冲 茶
  - 冲 胶卷
  - 冲 账
  - 冲 厕所
  - 洪水 冲 走了房屋
  - …

Text Categorization:

- **Text Categorization** (**TC**) is the to classify free text documents into a set of predefined categories or classes;
- **Multiclass** & **multi-label** (there may not be 1 to 1 correspondence between class and document, a document may belong to multiple classes) classification problem;
- Very **high dimensionality**
- Extremely **Sparse** instance vectors
- High Level of Redundancy (**Dense** concept vector)

Example

- We want to classify documents into categories

DOCUMENT	CATEGORY
… win the election …	POLITICS
… win the game …	SPORTS
… see a movie …	OTHER

Document representation

- Training documents are represented as a set of **features** plus a set of associated **labels**
- Which features?
  - Bag of words: each word occurring in a document. Stemming (for some language)and Stop-word list can be used.
  - Linguistic phrases (usually extracted using IE methods)
  - others
- Feature values can be:
  - Binary: appears or not
  - A function of the number of occurrences
  - **TFIDF values**
  - **...**

A simple TF-IDF "kernel"

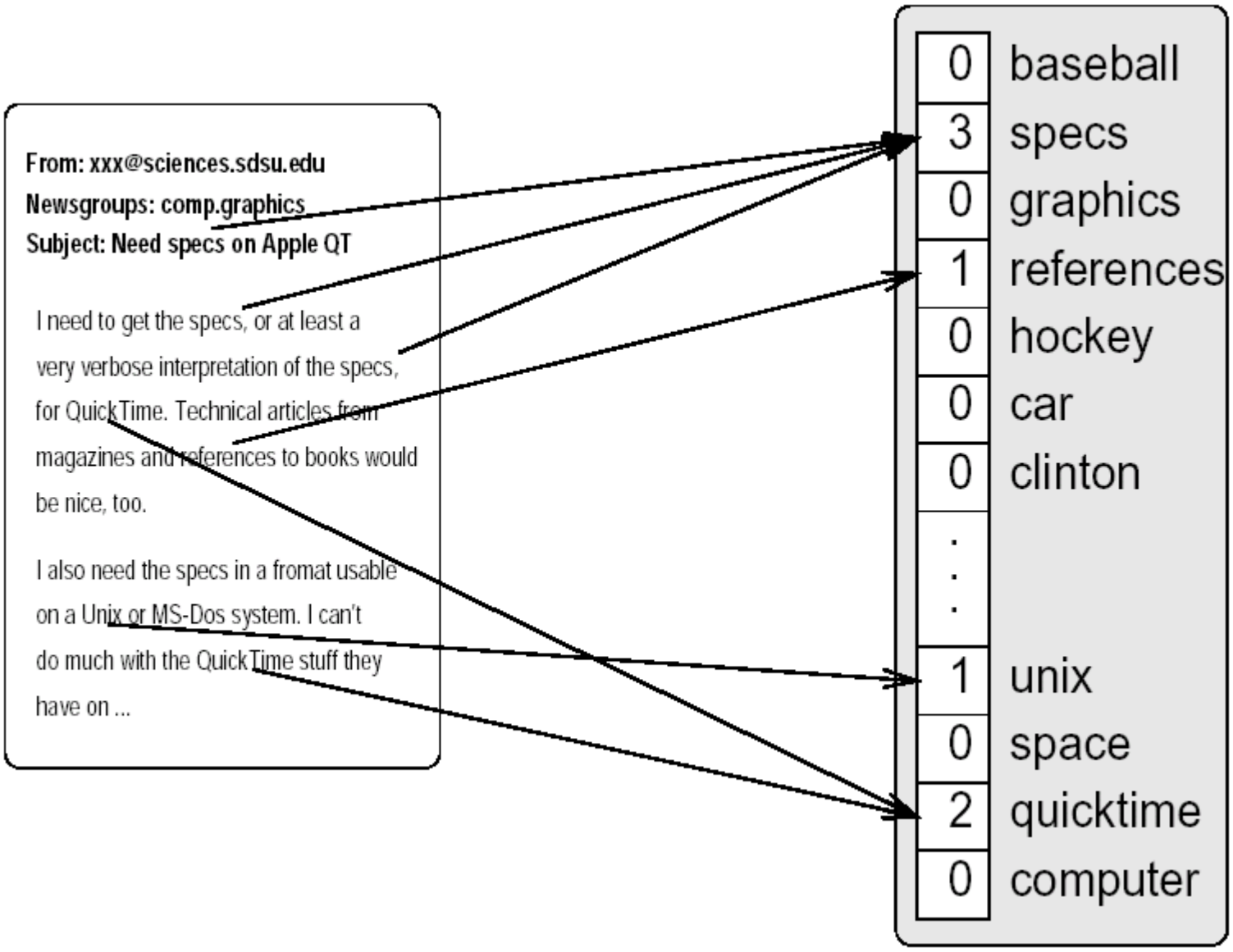
$$tfidf\ (t, x) = \#(t, x) \cdot \log \frac{|T|}{\#_T(t)}$$

$$w_{i,x} = \phi_i(x) = \frac{tfidf\ (t_i, x)}{\sqrt{\sum_{y \in T} (tfidf\ (t_i, y))^2}}$$

- The more often a term t occurs in a document x the more it is representative of its content
- The more documents the term occurs in, the less discriminating it is

Prepare

- Text files are processed to produce a vector of words
- Select 300 words with highest mutual information with each category(remove stopwords)
- A separate classifier is learned for each category.


From: xxx@sciences.sdsu.edu
Newsgroups: comp.graphics
Subject: Need specs on Apple QT
I need to get the specs, or at least a very verbose interpretation of the specs, for QuickTime. Technical articles from magazines and references to books would be nice, too.
I also need the specs in a fromat usable on a Unix or MS-Dos system. I can't do much with the QuickTime stuff they have on ...
0 baseball
3 specs
0 graphics
1 references
0 hockey
0 car
0 clinton
1 unix
0 space
2 quicktime
0 computer

Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Conclusion

- Two key concepts of SVM: maximize the margin and the kernel trick

- Vladimir Vapnik. Statistical Learning Theory. Wiley-Interscience; 1998
- Thorsten Joachims (joachims_01a): *A Statistical Learning Model of Text Classification for Support Vector Machines*
- **www.kernel-machines.org**