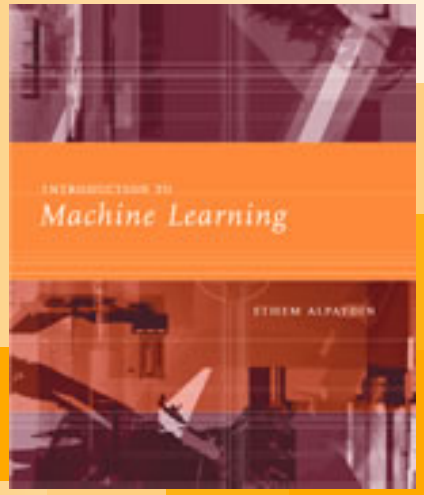



*Lecture Slides for*

INTRODUCTION TO

# *Machine Learning*

ETHEM ALPAYDIN


*alpaydin@boun.edu.tr*
*http://www.cmpe.boun.edu.tr/~ethem/i2ml*

CHAPTER 10:

# *Linear Discrimination*

# Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for $p(\mathbf{x}|C_i)$, use Bayes' rule to calculate $P(C_i|\mathbf{x})$

  *Choose* $C_i$ *if* $g_i(\mathbf{x}) = \log P(C_i|\mathbf{x})$ *is maximum*

- Discriminant-based: Assume a model for the discriminant $g_i(\mathbf{x}|\Phi_i)$; no density estimation
  - Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries

# Linear Discriminant

- Linear discriminant:

$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

- Advantages:
  - Simple: $O(d)$ space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)
  - Optimal when $p(\mathbf{x}|C_i)$ are Gaussian with shared cov matrix; useful when classes are (almost) linearly separable

# Generalized Linear Model

- **Quadratic discriminant:**

$$g_i(\mathbf{x} \mid \mathbf{W}_i, \mathbf{w}_i, w_{i0}) = \mathbf{x}^T \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

- Instead of higher complexity, we can still use a linear classifier if we use higher-order (product) terms.

- Map from ***x*** to ***z*** using nonlinear basis functions and use a linear discriminant in ***z***-space

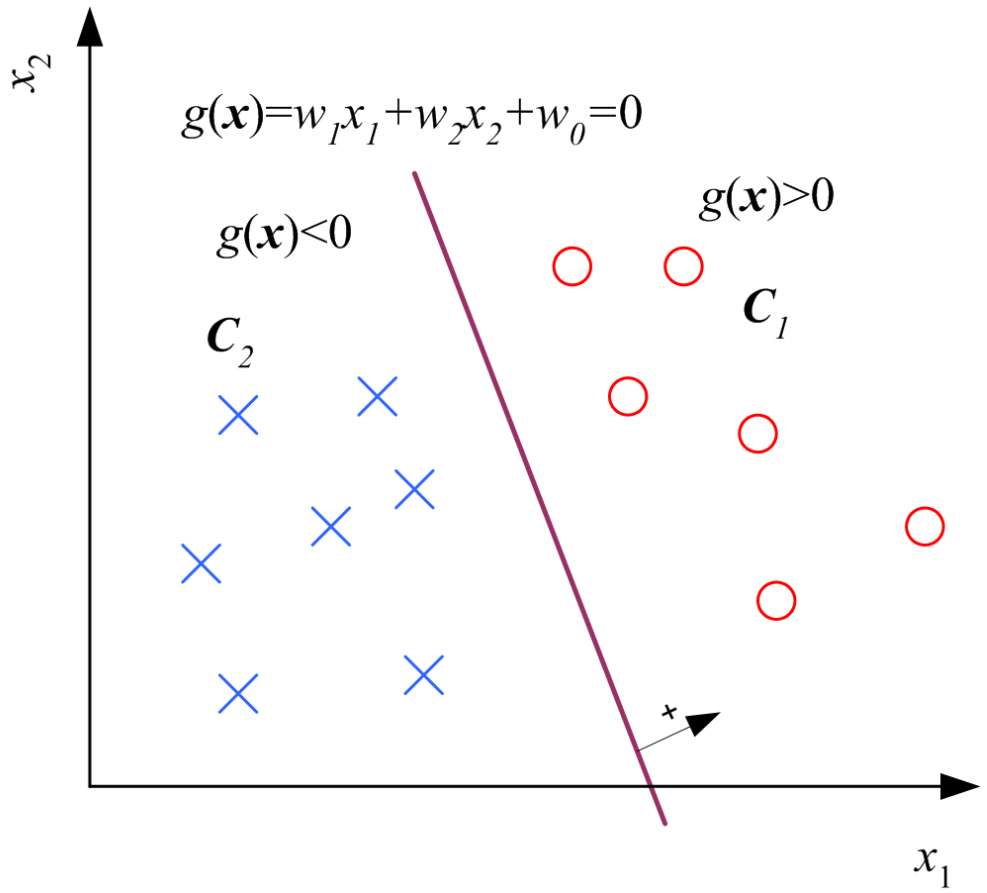
$$z_1 = x_1, \; z_2 = x_2, \; z_3 = x_1^2, \; z_4 = x_2^2, \; z_5 = x_1 x_2$$

- The linear function defined in the **z** space corresponds to a non-linear function in the **x** space.

$$g_i(\mathbf{x}) = \sum_{j=1}^{k} w_{ij} \phi_j(\mathbf{x})$$

# Two Classes

choose $C_1$ if $g_1(x) > g_2(x)$
$C_2$ if $g_2(x) > g_1(x)$




Define:

$$g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$$
$$= \left(\boldsymbol{w}_1^T\boldsymbol{x} + w_{10}\right) - \left(\boldsymbol{w}_2^T\boldsymbol{x} + w_{20}\right)$$
$$= \left(\boldsymbol{w}_1 - \boldsymbol{w}_2\right)^T\boldsymbol{x} + \left(w_{10} - w_{20}\right)$$
$$= \boldsymbol{w}^T\boldsymbol{x} + w_0$$

$$\text{choose} \begin{cases} C_1 & \text{if } g(\boldsymbol{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

# Learning the Discriminants

As we have seen before, when $p(\mathbf{x} \mid C_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, the optimal discriminant is a linear one:

$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}^{-1}\mu_i \quad w_{i0} = -\frac{1}{2}\mu_i^T \boldsymbol{\Sigma}^{-1}\mu_i + \log P(C_i)$$

So, estimate $\mu_i$ and $\Sigma$ from data, and plug into the gi's to find the linear discriminant functions.

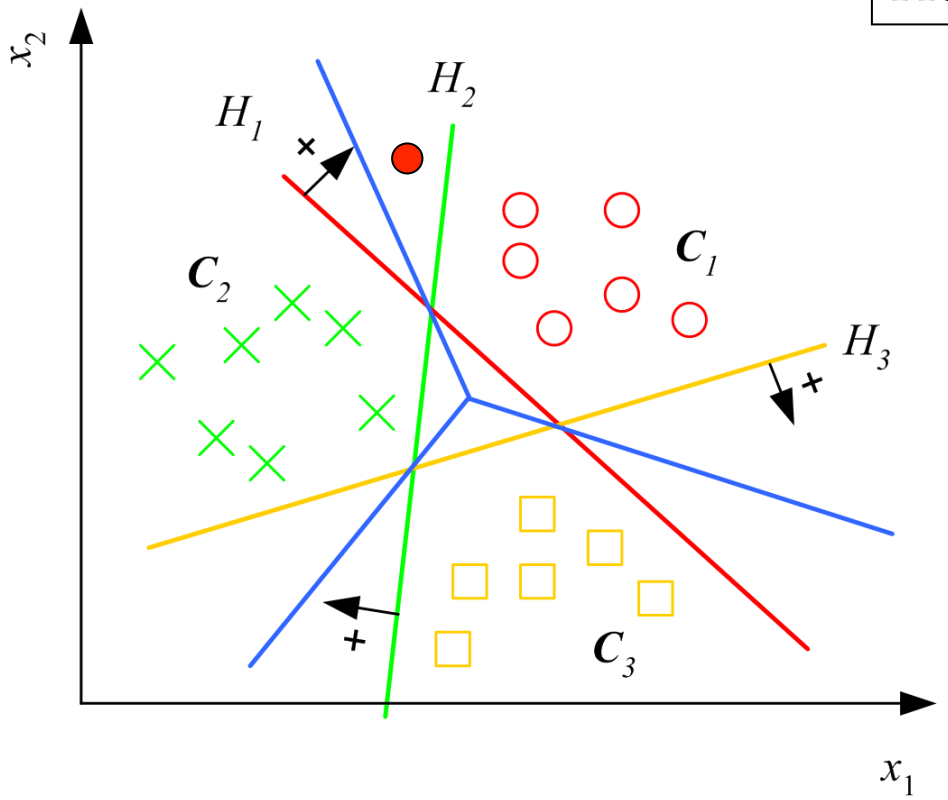
**Of course any way of learning can be used (e.g. perceptron, gradient descent, logistic regression...).**

- When K > 2
  - Combine K two-class problems, each one separating one class from all other classes

# Multiple Classes

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$$

How to train?
How to decide on a test?




Choose $C_i$ if

$$g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x})$$

Why? Any problem?

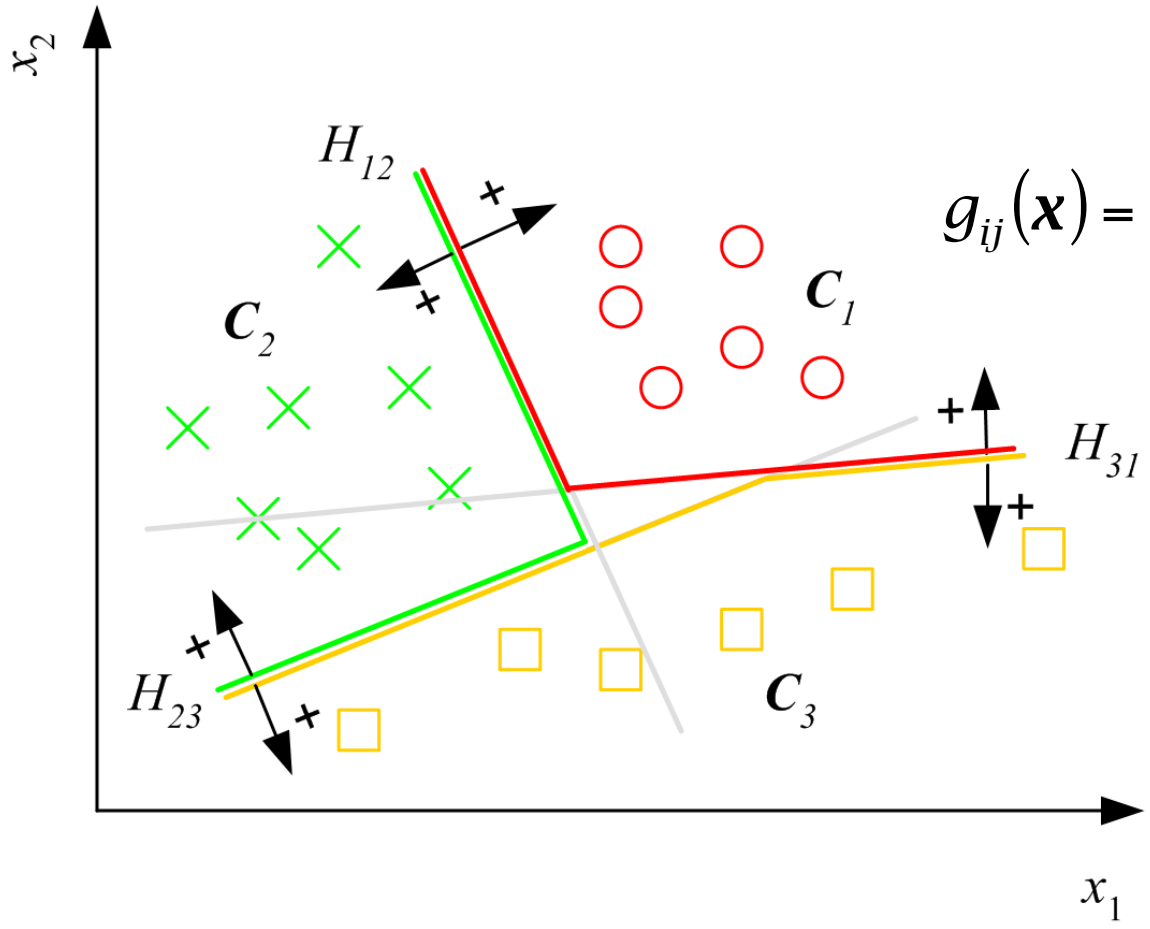
Convex decision regions based on $g_i$s (indicated with blue)
dist is $|g_i(x)|/||wi||$

Assumes that classes are linearly separable:
reject may be used

# Pairwise Separation

If the classes are not linearly separable:

$$g_{ij}\left(\boldsymbol{x} \mid \boldsymbol{w}_{ij}, w_{ij0}\right) = \boldsymbol{w}_{ij}^T \boldsymbol{x} + w_{ij0}$$




$$g_{ij}(\boldsymbol{x}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{if } \boldsymbol{x} \in C_j \\ \text{don't care} & \text{otherwise} \end{cases}$$

choose $C_i$ if $\forall j \neq i, g_{ij}(\boldsymbol{x}) > 0$

Uses k(k-1)/2 linear discriminants

- Pairwise linear separation is much more likely than linear separability
- None of the classes may satisfy the condition
  - Reject
  - Use max

$$\text{choose } C_i \text{ if } \forall j \neq i, g_{ij}(\mathbf{x}) > 0 \quad \rightarrow \quad \text{choose } C_i \text{ maximizing } g_i(\mathbf{x}) = \sum_{j \neq i} g_{ij}(\mathbf{x})$$

# *A Bit of Geometry*

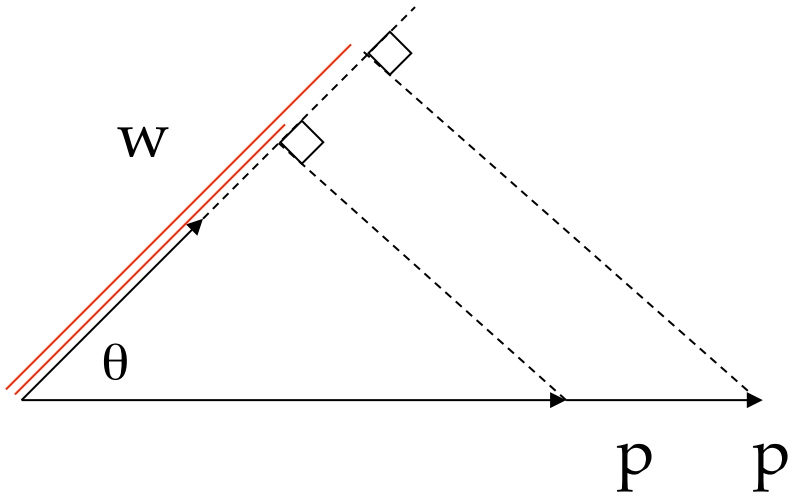
- Dot Product and Projection

- $\langle w,p \rangle = w^T p = ||w||||p||\text{Cos}\theta$

- proj. of p onto w

  $= ||p||\text{Cos}\theta$

  $= w^T.p/||w||$



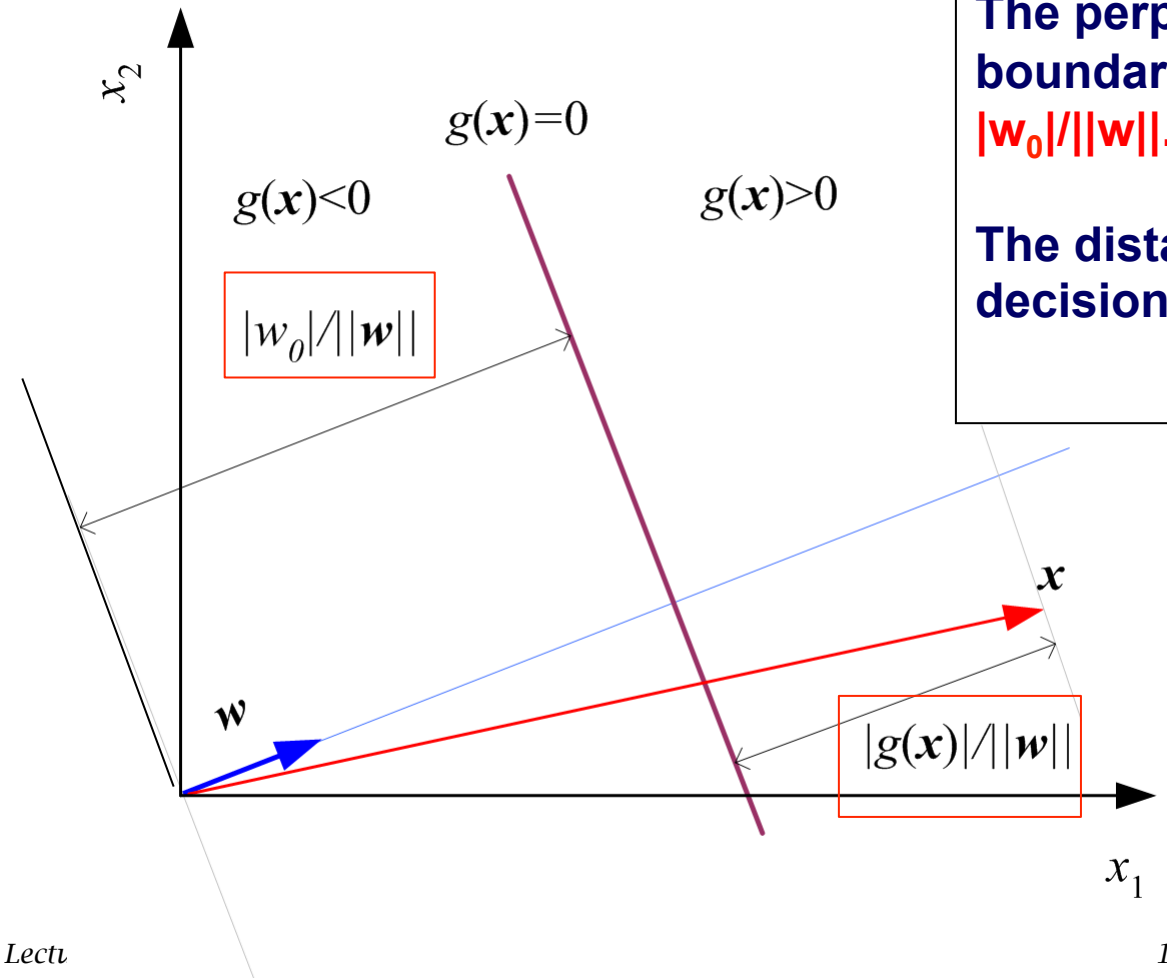
# Geometry

The points **x** on the separating hyperplane have g(**x**) = $\mathbf{w}^T\mathbf{x}+w_0=0$. Hence for the points on the boundary $\mathbf{w}^T\mathbf{x}=-w_0$.

Thus, these points also have the same projection onto the weight vector **w**, namely $\mathbf{w}^T\mathbf{x}/||\mathbf{w}||$ (by definition of projection and dot product). But this is equal to $-w_0/||\mathbf{w}||$. Hence ...

**The perpendicular distance of the boundary to the origin is $|w_0|/||w||$.**

**The distance of any point x to the decision boundary is $|g(x)|/||w||$.**



# *Support Vector Machines*

- Vapnik and Chervonenkis – 1963
- Boser, Guyon and Vapnik – 1992 (kernel trick)
- Cortes and Vapnik – 1995 (soft margin)

- The SVM is a machine learning algorithm which
  - solves classification problems
  - uses a flexible representation of the class boundaries
  - implements automatic complexity control to reduce overfitting
  - has a single global minimum which can be found in polynomial time

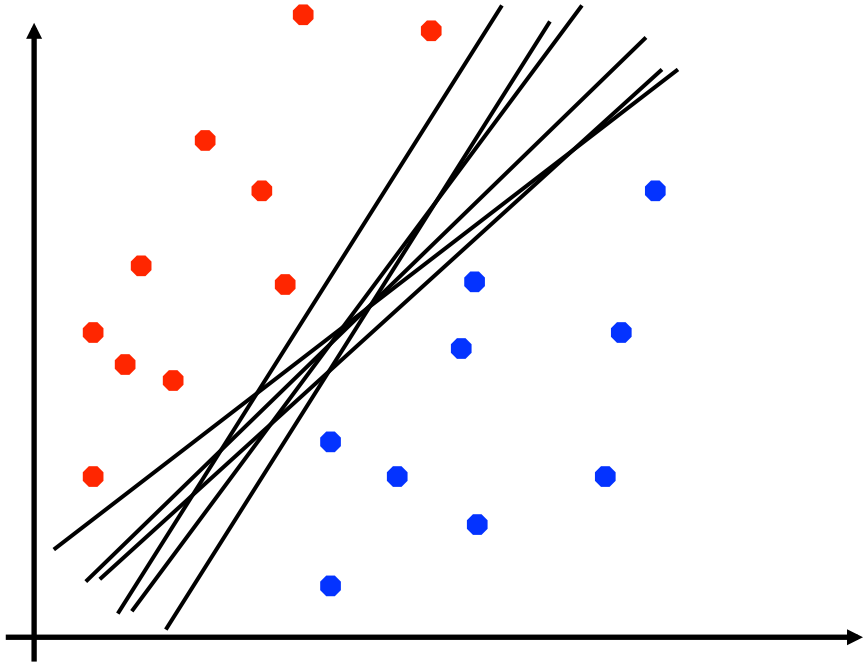
- It is popular because
  - it can be easy to use
  - it often has good generalization performance
  - the same algorithm solves a variety of problems with little tuning

# SVM Concepts

- Convex programming and duality
- Using maximum margin to control complexity
- Representing non-linear boundaries with feature expansion
- The kernel trick for efficient optimization

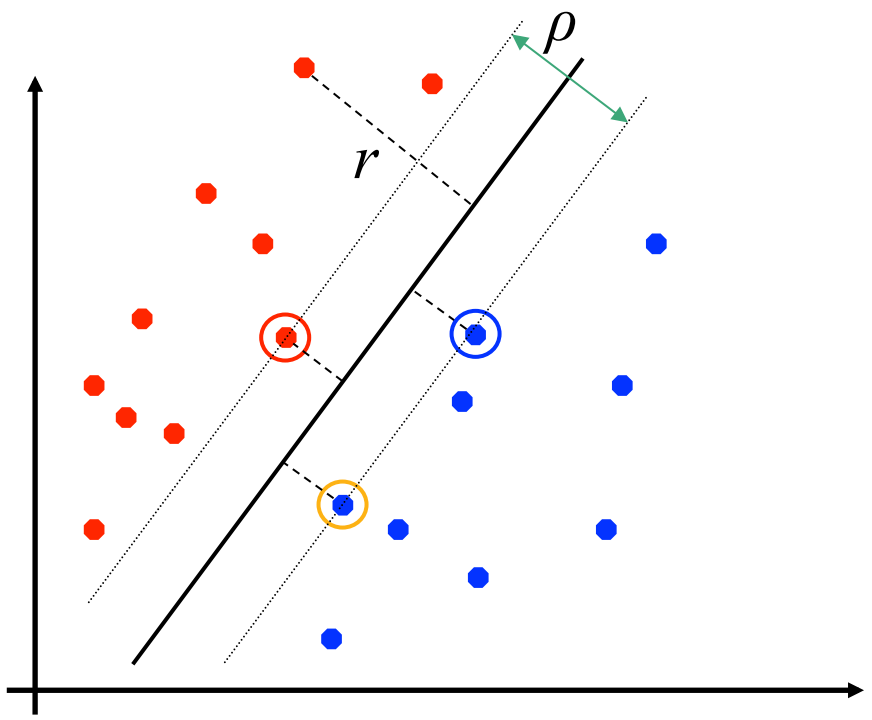
# Linear Separators

- Which of the linear separators is optimal?



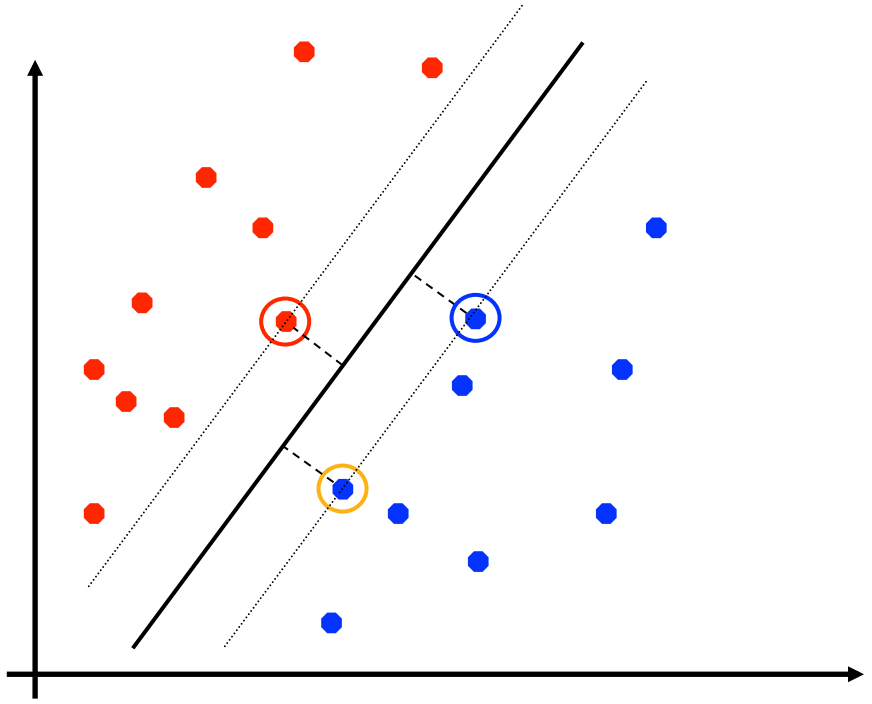
# Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** $\rho$ of the separator is the distance between support vectors from two classes.



# Maximum Margin Classification

- Maximizing the margin is good according to intuition.
- Implies that only support vectors matter; other training examples are ignorable.



# SVM as 2-class Linear Classifier

(Cortes and Vapnik, 1995; Vapnik, 1995)

$$\mathrm{X} = \left\{\mathbf{x}^t, r^t\right\}_t \text{ where } r^t = \begin{cases} +1 & \text{if } \mathbf{x}^t \in C_1 \\ -1 & \text{if } \mathbf{x}^t \in C_2 \end{cases}$$

find $\mathbf{w}$ and $w_0$ such that

$$\mathbf{w}^T\mathbf{x}^t + w_0 \geq +1 \text{ for } r^t = +1$$

$$\mathbf{w}^T\mathbf{x}^t + w_0 \leq -1 \text{ for } r^t = -1$$

**Note the condition >= 1 (not just 0). We can always do this if the classes are linearly separable by rescaling w and w0, without affecting the separating hyperplane:** $\mathbf{w}^T\mathbf{x}^t + w_0 = 0$

Optimal separating hyperplane: Separating hyperplane maximizing the margin

# Optimal Separating Hyperplane

Must satisfy :

$$\mathbf{w}^T\mathbf{x}^t + w_0 \geq +1 \text{ for } r^t = +1$$

$$\mathbf{w}^T\mathbf{x}^t + w_0 \leq -1 \text{ for } r^t = -1$$

which can be rewritten as

$$r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1$$

(Cortes and Vapnik, 1995; Vapnik, 1995)

# Maximizing the Margin

Distance from the discriminant to the closest instances on either side is called the margin

In general this relationship holds (geometry): $d = \frac{|g(x)|}{\|\mathbf{w}\|}$

So, for the support vectors, we have:

$$d = \begin{cases} \frac{1}{\|\mathbf{w}\|} \\ \\ \frac{|-1|}{\|\mathbf{w}\|} \end{cases}$$

$$\rho = 2d = \frac{2}{\|\mathbf{w}\|}$$

To maximize margin, minimize the Euclidian norm of the weight vector w

# *Maximizing the Margin*-Alternate explanation

- Distance from the discriminant to the closest instances on either side is called the margin
- Distance of x to the hyperplane is $\frac{\left|\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right|}{\|\boldsymbol{w}\|}$
- We require that this distance is at least some value $\rho > 0$. $\frac{r^t\left(\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right)}{\|\boldsymbol{w}\|} \geq \rho, \forall t$
- We would like to maximize $\rho$, but we can do so in infinitely many ways by scaling w.
- For a unique sol'n, we fix $\rho\|\boldsymbol{w}\|=1$ and minimize $\|\boldsymbol{w}\|$.

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

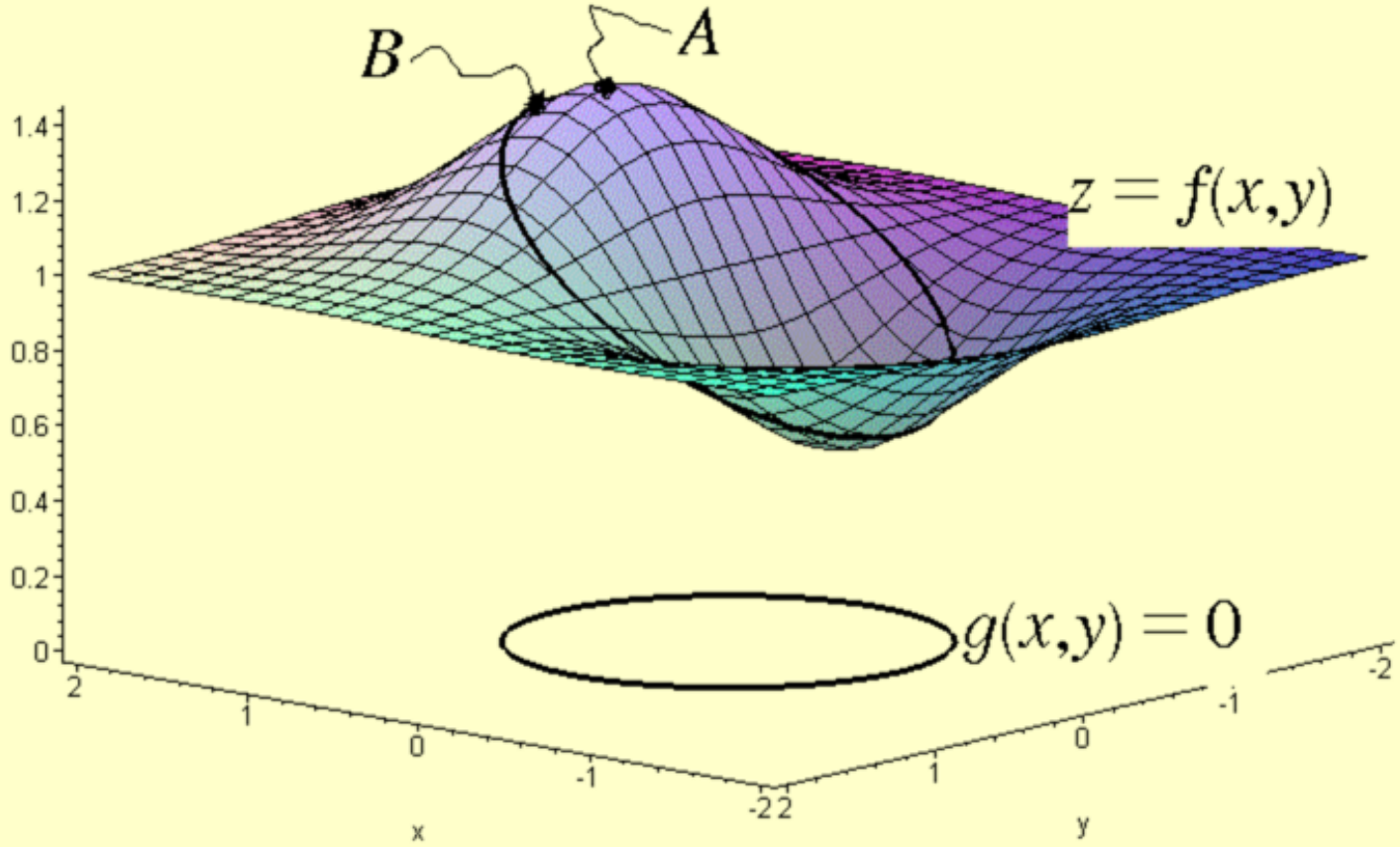
$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N}\alpha^t\left[r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) - 1\right]$$

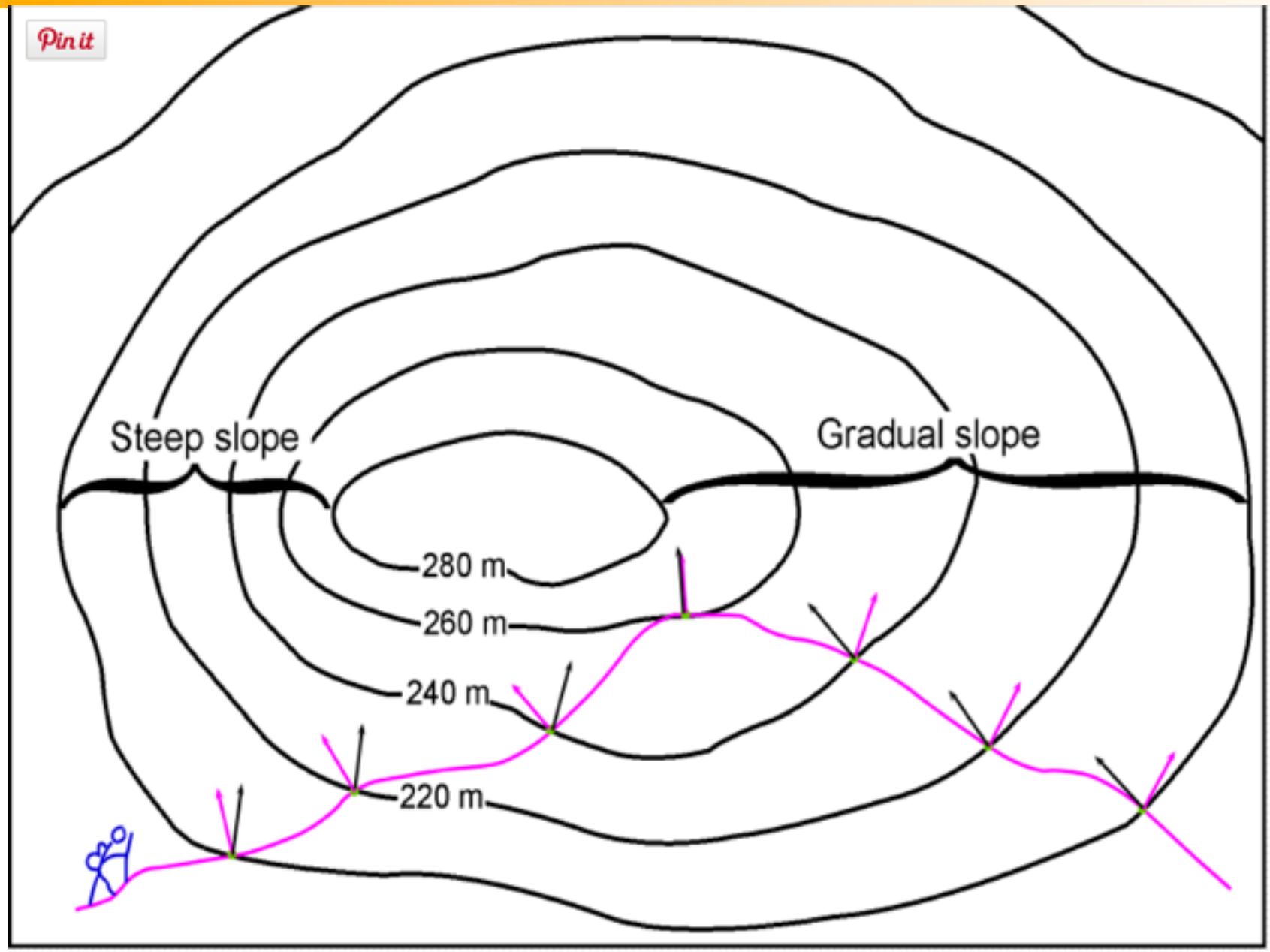
Unconstrained problem using Lagrange multipliers (+ numbers)

The solution, if it exists, is always at a saddle point of the Lagrangian

$L_p$ should be minimized w.r.t **w** and maximized w.r.t $\alpha^t$s

In the figure below we have illustrated an extreme value problem with constraints. The point $A$ is the largest value of the function $z=f(x,y)$ while the point $B$ is the largest value of the function under the *constraint* $g(x,y)=0$.




Pin it
Steep slope
Gradual slope
280 m
260 m
240 m
220 m

The method of Lagrange multipliers allows us to maximize or minimize functions with the constraint that we only consider points on a certain surface. To find critical points of a function $f(x, y, z)$ on a level surface $g(x, y, z) = C$ (or *subject to the constraint* $g(x, y, z) = C$), we must solve the following system of simultaneous equations:

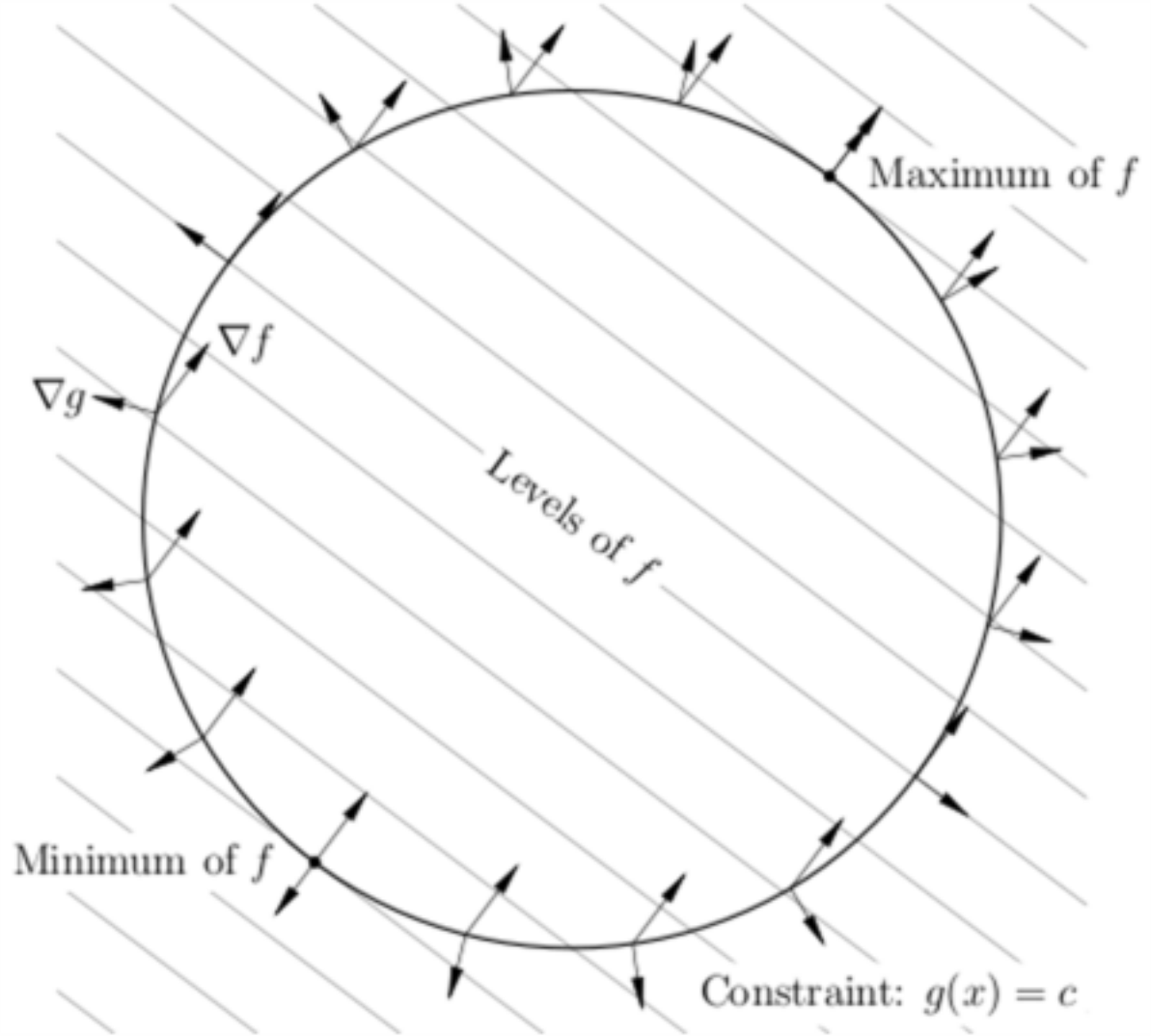
$$\begin{aligned} \nabla f(x, y, z) &= \lambda \nabla g(x, y, z) \\ g(x, y, z) &= C \end{aligned}$$

Remembering that $\nabla f$ and $\nabla g$ are vectors, we can write this as a collection of four equations in the four unknowns $x$, $y$, $z$, and $\lambda$:

$$\begin{aligned} f_x(x, y, z) &= \lambda g_x(x, y, z) \\ f_y(x, y, z) &= \lambda g_y(x, y, z) \\ f_z(x, y, z) &= \lambda g_z(x, y, z) \\ g(x, y, z) &= C \end{aligned}$$

The variable $\lambda$ is a dummy variable called a "Lagrange multiplier"; we only really care about the values of $x$, $y$, and $z$.

The diagram shows a linear function $f(x, y) = ax + by$ subject to a constraint $x^2 + y^2 = c$. Here $\nabla f = (a, b)$ is constant, $\nabla g = (2x, 2y)$, and the constrained extrema of $f$ occur at the points where $(a, b)$ is perpendicular to the circle.



$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) - 1\right]$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) + \sum_{t=1}^{N} \alpha^t$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0$$

Convex quadratic optimization problem can be solved using the dual form where we use these local minima constraints and maximize w.r.t $\alpha^t$s

Problem: maximize

$$f(x,y) = 6x + 8y$$

subject to

$$g(x,y) = x^2 + y^2 - 1 \geq 0$$

Using a Lagrange multiplier $a$,

$$\max_{xy} \min_{a \geq 0} \; f(x,y) + ag(x,y)$$

At optimum,

$$0 = \nabla f(x,y) + a\nabla g(x,y) = \begin{pmatrix} 6 \\ 8 \end{pmatrix} + 2a \begin{pmatrix} x \\ y \end{pmatrix}$$

from: http://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/lagrang/lagrang.html

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[ r^t \left( \mathbf{w}^T \mathbf{x}^t + w_0 \right) - 1 \right]$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t \left( \mathbf{w}^T \mathbf{x}^t + w_0 \right) + \sum_{t=1}^{N} \alpha^t$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0$$

$$L_d = \frac{1}{2}\left( \mathbf{w}^T \mathbf{w} \right) - \mathbf{w}^T \sum_t \alpha^t r^t \mathbf{x}^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t$$

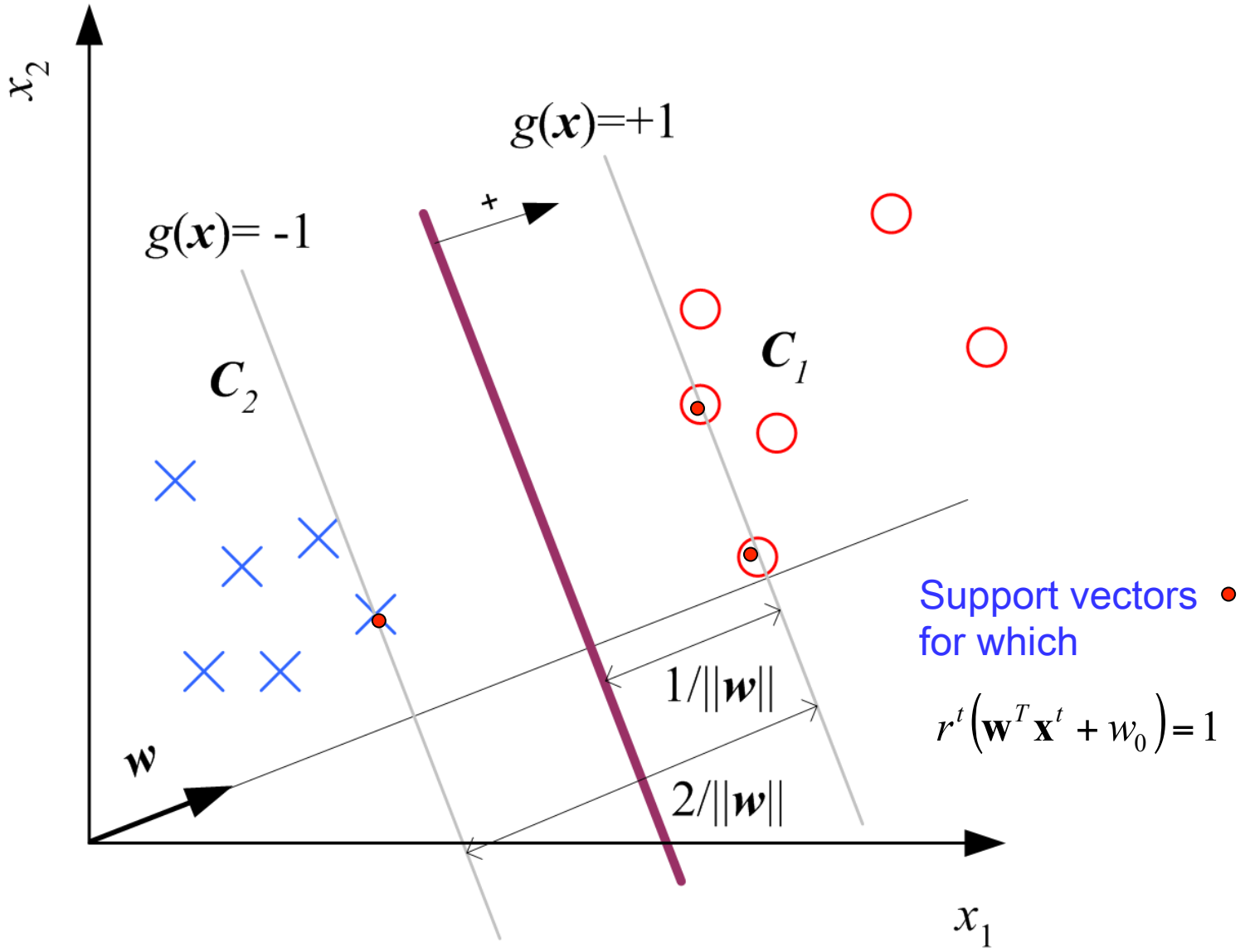
$$= -\frac{1}{2}\left( \mathbf{w}^T \mathbf{w} \right) + \sum_t \alpha^t$$

$$= -\frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s \left( \mathbf{x}^t \right)^T \mathbf{x}^s + \sum_t \alpha^t$$

$$\text{subject to } \sum_t \alpha^t r^t = 0 \text{ and } \alpha^t \geq 0, \forall t$$

- Maximize $L_d$ with respect to $\alpha^t$ only
- Quadratic programming problem
- Thanks to the convexity of the problem, optimal value of $L_p = L_d$

- To every convex program corresponds a dual
- Solving original (primal) is equivalent to solving dual


$x_2$
$g(x)$=+1
$g(x)$= -1
+
$C_2$
$C_1$
Support vectors
for which
$r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) = 1$
$1/||w||$
$2/||w||$
$w$
$x_1$

$$L_d = \frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) - \mathbf{w}^T\sum_t \alpha^t r^t \mathbf{x}^t - w_0\sum_t \alpha^t r^t + \sum_t \alpha^t$$

$$= -\frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) + \sum_t \alpha^t$$

$$= -\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s \left(\mathbf{x}^t\right)^T \mathbf{x}^s + \sum_t \alpha^t$$

Size of the dual depends on N and not on d

- Maximize $L_d$ with respect to $\alpha^t$ only

$$\text{subject to } \sum_t \alpha^t r^t = 0 \text{ and } \alpha^t \geq 0, \forall t$$

- Quadratic programming problem
- Thanks to the convexity of the problem, optimal value of $L_p$ = $L_d$

# Calculating the parameters $\mathbf{w}$ and $w_0$

Note that:

- either the constraint is exactly satisfied (=1)
  (and $\alpha^t$ can be non-zero)
- or the constraint is clearly satisfied (> 1)
  (then $\alpha^t$ must be zero)

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[ r^t \left( \mathbf{w}^T \mathbf{x}^t + w_0 \right) - 1 \right]$$

- Once we solve for $\alpha^t$, *we see that most of them* are 0 and only a small number have $\alpha^t > 0$
  - the corresponding $\mathbf{x}^t$s are called the **support vectors**

# Calculating the parameters $\mathbf{w}$ and $w_0$

Once we have the Lagrange multipliers, we can compute $\mathbf{w}$ and $w_0$:

$$\mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t = \sum_{t \in SV} \alpha^t r^t \mathbf{x}^t$$

where *SV* is the set of the Support Vectors.

$$w_0 = r^t - \mathbf{w}^T \mathrm{x}^t$$

- We make decisions by comparing each query x with only the support vectors

$$y = sign(\mathbf{w}^T\mathbf{x} + w_0) = (\sum_{t \in SV}^{N} \alpha^t r^t \mathbf{x}^t)x + w_0$$

- Choose class C1 if +, C2 if negative

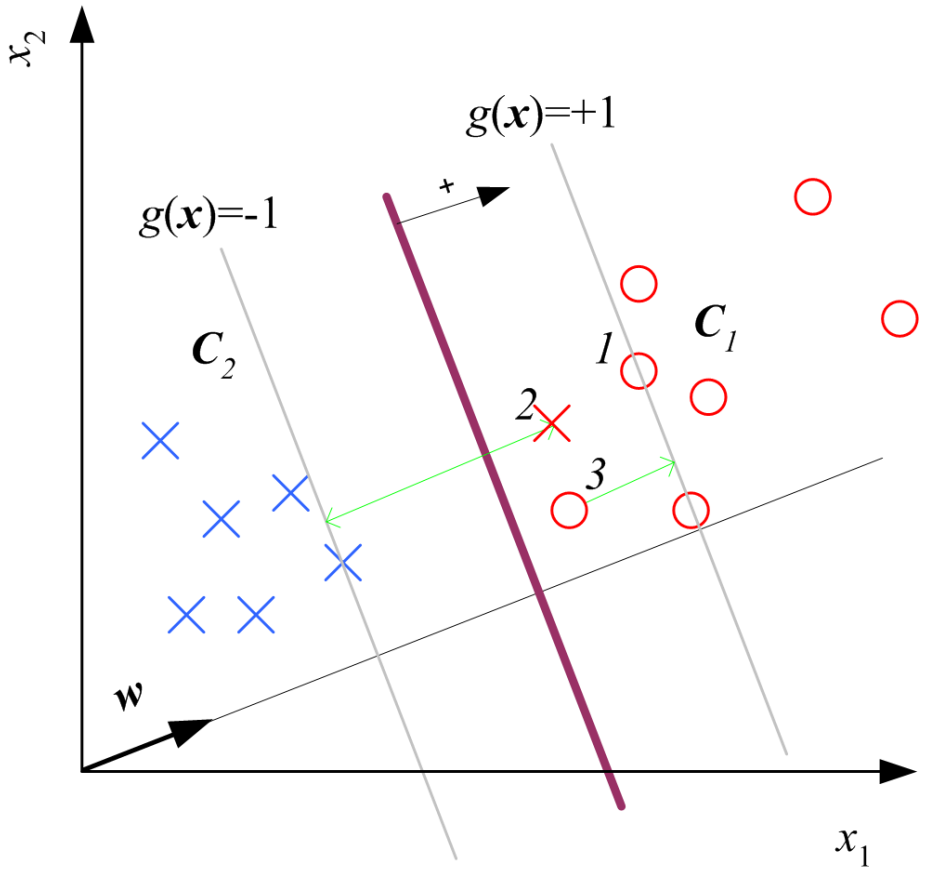
# *Not-Linearly Separable Case*

- The non-separable case cannot find a feasible solution using the previous approach
  - The objective function ($L_D$) grows arbitrarily large.

- Relax the constraints, but only when necessary
  - Introduce a further cost for this

# Soft Margin Hyperplane

- Not linearly separable




$$r^t\left(\mathbf{w}^T x^t + w_0\right) \geq 1 - \xi^t$$

$$\xi^t \geq 0$$

Three cases (shown in fig):

Case 1: $\xi^t = 0$

Case 2: $\xi^t \geq 1$

Case 3: $0 \leq \xi^t < 1$

# Soft Margin Hyperplane

- Define Soft error $\sum_t \xi^t$

Upper bound on the number of training errors

- New primal is

Lagrange multipliers to enforce positivity of $\xi$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \xi^t - \sum_t \alpha^t\left[r^t\left(\mathbf{w}^T x^t + w_0\right) - 1 + \xi^t\right] - \sum_t \mu^t \xi^t$$

- Parameter *C* can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

# Soft Margin Hyperplane

- New dual is the same as the old one

$$L_d = -\frac{1}{2}\sum_t \sum_s \alpha^t \alpha^s r^t r^s \left(\mathbf{x}^t\right)^T \mathbf{x}^s + \sum_t \alpha^t$$

subject to

$$\sum_t \alpha^t r^t = 0 \text{ and } 0 \le \alpha^t \le C, \forall t$$

- As in the separable case, instances that are not support vectors vanish with their $\alpha^t=0$ and the remaining define the boundary.

# *Kernel Functions in SVM*

- We can handle the overfitting problem: even if we have lots of parameters, large margins make simple classifiers
- “All” that is left is efficiency
- Solution: kernel trick

# Kernel Functions

- Instead of trying to fit a non-linear model, we can
  - map the problem to a new space through a non-linear transformation and
  - use a linear model in the new space

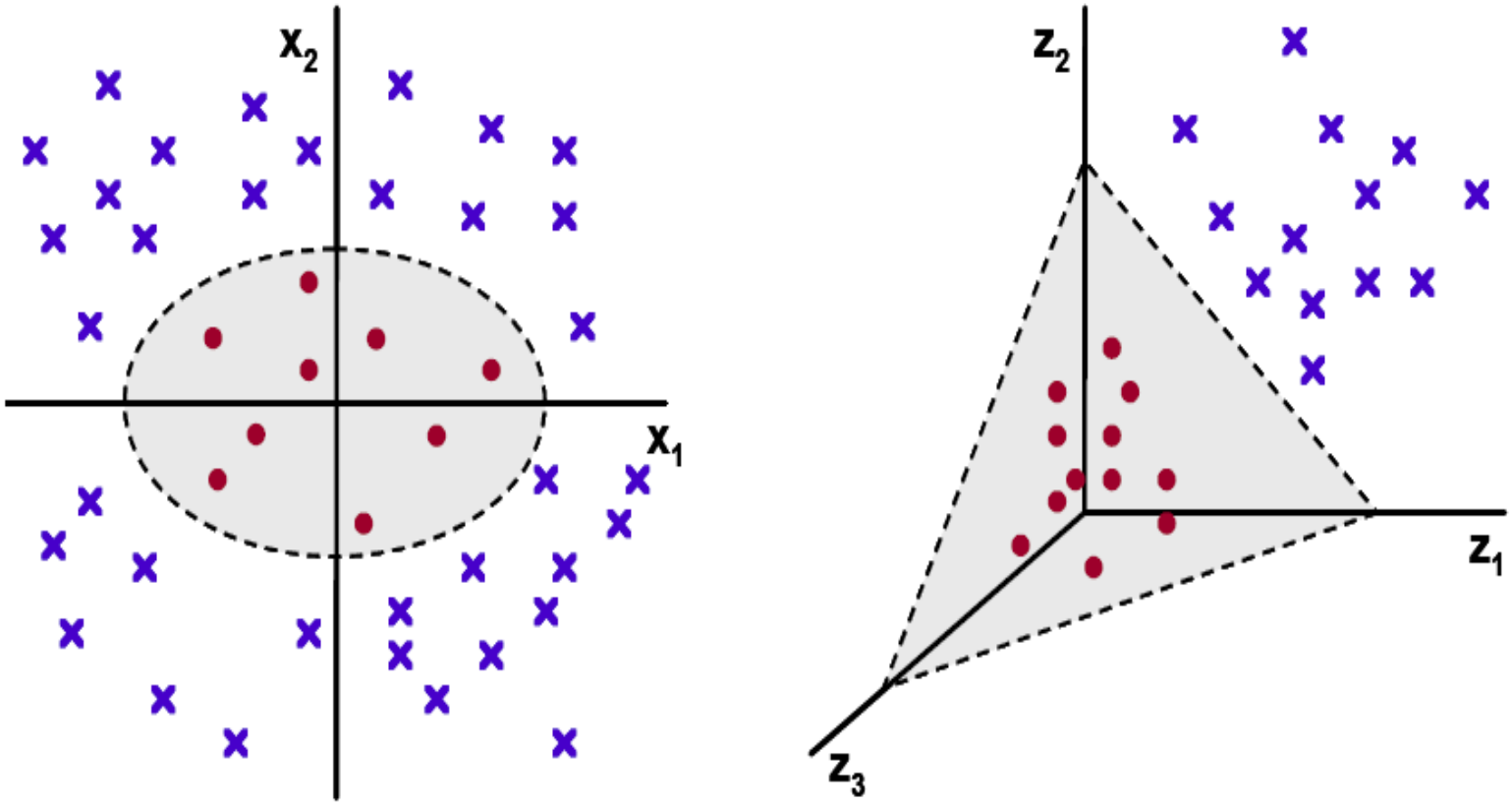
- Say we have the new space calculated by the basis functions $\boldsymbol{z} = \boldsymbol{\varphi}(\boldsymbol{x})$ where $z_j = \phi_j(\boldsymbol{x})$ , j=1,...,k

d-dimensional $\boldsymbol{x}$ space $\longrightarrow$ k-dimensional $\boldsymbol{z}$ space

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(x) \ \phi_2(x) \ ... \ \phi_k(x)]$$

$$\varphi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



# Kernel Functions

$$g(\mathbf{x}) = \sum_{k=1} w_k \varphi_k(\mathbf{x}) + b$$

$$g(\mathbf{x}) = \sum_{k=0} w_k \varphi_k(\mathbf{x})$$

if we assume $\varphi_0(\mathbf{x}) = 1$ for $\forall \mathbf{x}$

# Kernel Machines

- Preprocess input $\boldsymbol{x}$ by basis functions

$$\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x}) \qquad g(\boldsymbol{z}) = \boldsymbol{w}^T\boldsymbol{z}$$

$$g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})$$

- SVM solution: Find Kernel functions K(x,y) such that the inner product of basis functions are replaced by a Kernel function *in the original input space*

$$\boldsymbol{w} = \sum_t \alpha^t r^t \boldsymbol{z}^t = \sum_t \alpha^t r^t \boldsymbol{\varphi}\left(\boldsymbol{x}^t\right)$$

$$g\left(\boldsymbol{x}\right) = \boldsymbol{w}^T\boldsymbol{\varphi}\left(\boldsymbol{x}\right) = \sum_t \alpha^t r^t \boldsymbol{\varphi}\left(\boldsymbol{x}^t\right)^T\boldsymbol{\varphi}\left(\boldsymbol{x}\right)$$

$$g\left(\boldsymbol{x}\right) = \sum_t \alpha^t r^t K\left(\boldsymbol{x}^t, \boldsymbol{x}\right)$$

# Kernel Functions

- Consider polynomials of degree $q$:

$$K(\mathbf{x},\mathbf{y}) = \left(\mathbf{x}^T\mathbf{y}+1\right)^q$$

$$K(\mathbf{x},\mathbf{y}) = \left(\mathbf{x}^T\mathbf{y}+1\right)^2$$

$$= \left(x_1y_1 + x_2y_2 + 1\right)^2$$

$$= 1 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2 + x_1^2y_1^2 + x_2^2y_2^2$$

$$\phi(\mathbf{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2\right]^T$$

(Cherkassky and Mulier, 1998)

$$x = (x_1, x_2);$$

$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

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

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

$$= \langle \phi(x), \phi(z) \rangle$$

www.support-vector.net

# Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
  - Mapping $\Phi$: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is $\mathbf{x}$ itself

- Polynomial of power $p$: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$
  - Mapping $\Phi$: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping $\Phi$: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian)

- Higher-dimensional space still has *intrinsic* dimensionality *d,* but linear separators in it correspond to *non-linear* separators in original space.

- Typically k is much larger than d, and possibly larger than N
  - Using the dual where the complexity depends on N rather than k is advantageous
- We use the soft margin hyperplane
  - If C is too large, too high a penalty for non-separable points (too many support vectors)
  - If C is too small, we may have underfitting
- Decide by cross-validation

# Other Kernel Functions

- Polynomials of degree $q$: $K(\mathbf{x}^t, \mathbf{x}) = (\mathbf{x}^T \mathbf{x}^t)^q$

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = (\boldsymbol{x}^T \boldsymbol{x}^t + 1)^q$$

- Radial-basis functions: $$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{\sigma^2}\right]$$

- Sigmoidal functions such as: $$K(\boldsymbol{x}^t, \boldsymbol{x}) = \tanh(2\boldsymbol{x}^T \boldsymbol{x}^t + 1)$$

# What Functions are Kernels? Advanced

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ can be cumbersome.
- Any function that satisfies some constraints called the Mercer conditions can be a Kernel function - (Cherkassky and Mulier, 1998)

  Every semi-positive definite symmetric function is a kernel

- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$$K=\begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \dots & K(\mathbf{x}_1,\mathbf{x}_n) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_n) \\ & & & & \\ \dots & \dots & \dots & \dots & \dots \\ K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & K(\mathbf{x}_n,\mathbf{x}_3) & \dots & K(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$$

- Informally, kernel methods *implicitly* define the class of possible patterns by introducing a notion of similarity between data
  - Choice of similarity -> Choice of relevant features

- More formally, kernel methods exploit information about the inner products between data items
  - Many standard algorithms can be rewritten so that they only require inner products between data (inputs)
  - Kernel functions = inner products in some feature space (potentially very complex)
  - If kernel given, no need to specify what features of the data are being used
  - Kernel functions make it possible to use infinite dimensions
    - efficiently in time / space

# String kernels

- For example, given two documents, $D_1$ and $D_2$, the number of words appearing in both may form a kernel.

- Define $\phi(D_1)$ as the M-dimensional binary vector where dimension i is 1 if word $w_i$ appears in $D_1$; 0 otherwise.
- Then $\phi(D_1)^T\phi(D_2)$ indicates the number of shared words.

- If we define $K(D_1,D_2)$ as the number of shared words;
  - no need to preselect the M words
  - no need to create the bag-of-words model explicitly
  - M can be as large as we want

# Projecting into Higher Dimensions

- **Naïve application of this concept by simply projecting to a high-dimensional non-linear manifold has two major problems**
  - **Statistical**: operation on high-dimensional spaces is ill-conditioned due to the "curse of dimensionality" and the subsequent risk of overfitting
  - **Computational**: working in high-dimensions requires higher computational power, which poses limits on the size of the problems that can be tackled
- **SVMs bypass these two problems in a robust and efficient manner**
  - First, generalization capabilities in the high-dimensional manifold are ensured by enforcing a **largest margin** classifier
    - Recall that generalization in SVMs is strictly a function of the margin (or the VC dimension), regardless of the dimensionality of the feature space
  - Second, projection onto a high-dimensional manifold is only **implicit**
    - Recall that the SVM solution depends only on the dot product $\langle x_i, x_j \rangle$ between training examples
    - Therefore, operations in high dimensional space $\varphi(x)$ do not have to be performed explicitly if we find a function $K(x_i,x_j)$ such that $K(x_i,x_j)=\langle \varphi(x_i), \varphi(x_j) \rangle$
    - $K(x_1,x_2)$ is called a **kernel** function in SVM terminology

# SVM Applications

- Cortes and Vapnik 1995:
  - Handwritten digit classification
  - 16x16 bitmaps -> 256 dimensions
  - Polynomial kernel where q=3 -> feature space with $10^6$ dimensions
  - No overfitting on a training set of 7300 instances
  - Average of 148 support vectors over different training sets

Expected test error rate:

$Exp_N[P(error)] = Exp_N[\#support\ vectors] / N$

(= 0.02 for the above example)

# SVM history and applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs represent a general methodology for many PR problems: classification,regression, feature extraction, clustering, novelty detection, etc.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of $\alpha_i$'s at a time, e.g. SMO [Platt '99] and [Joachims '99]

# Advantages of SVMs

- There are no problems with local minima, because the solution is a Qaudratic Programming problem with a global minimum.
- The optimal solution can be found in polynomial time
- There are few model parameters to select: the penalty term C, the kernel function and parameters (e.g., spread σ in the case of RBF kernels)
- The final results are stable and repeatable (e.g., no random initial weights)
- The SVM solution is sparse; it only involves the support vectors
- SVMs rely on elegant and principled learning methods
- SVMs provide a method to control complexity independently of dimensionality
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities

# Challenges

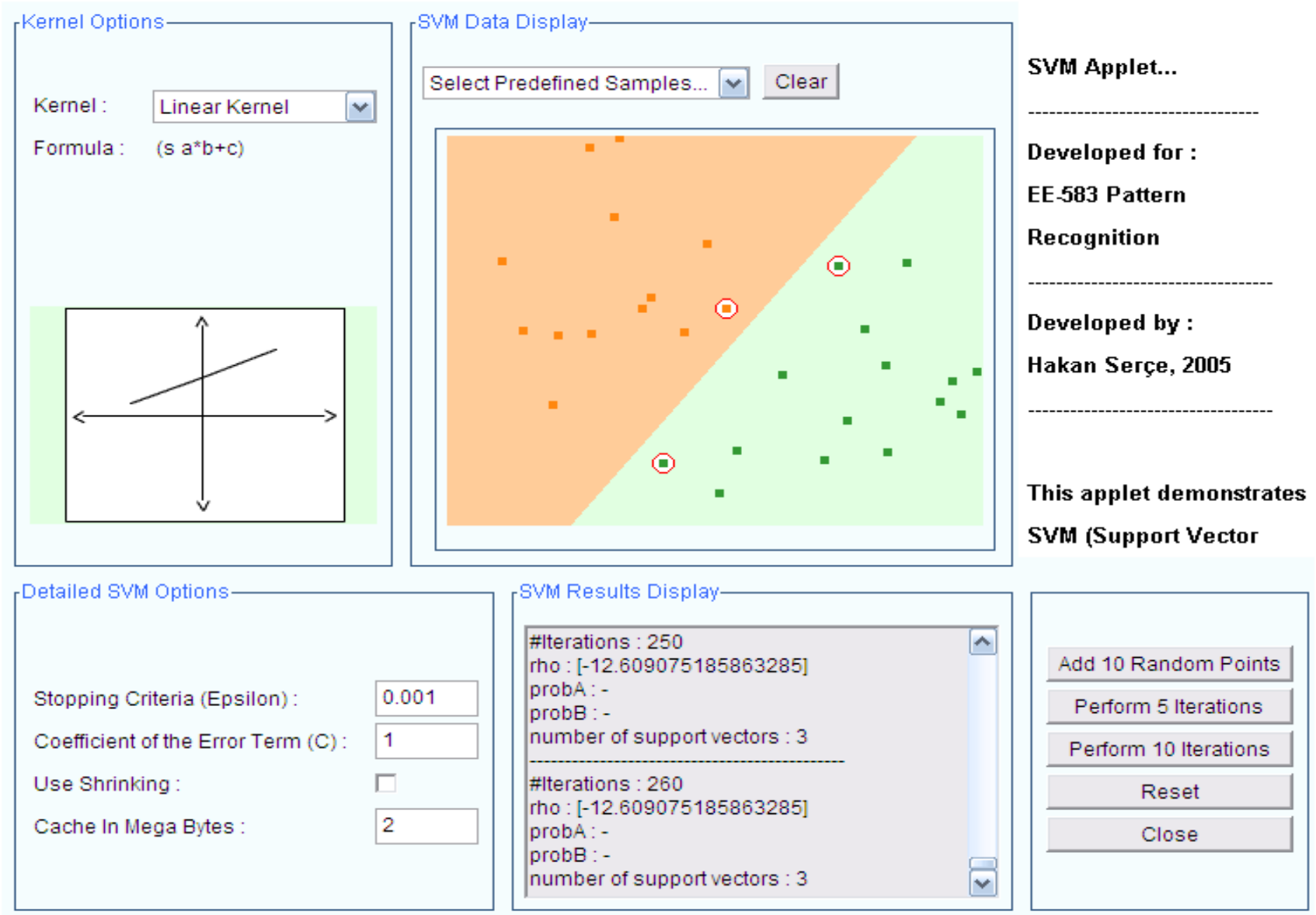
- **Can the kernel functions be selected in a principled manner?**
- **SVMs still require selection of a few parameters, typically through cross-validation**
- **How does one incorporate domain knowledge?**
  - Currently this is performed through the selection of the kernel and the introduction of “artificial” examples
- How interpretable are the results provided by an SVM?
- What is the optimal data representation for SVM? What is the effect of feature weighting? How does an SVM handle categorical or missing features?
- Do SVMs always perform best? Can they beat a hand-crafted solution for a particular problem?
- Do SVMs eliminate the model selection problem?

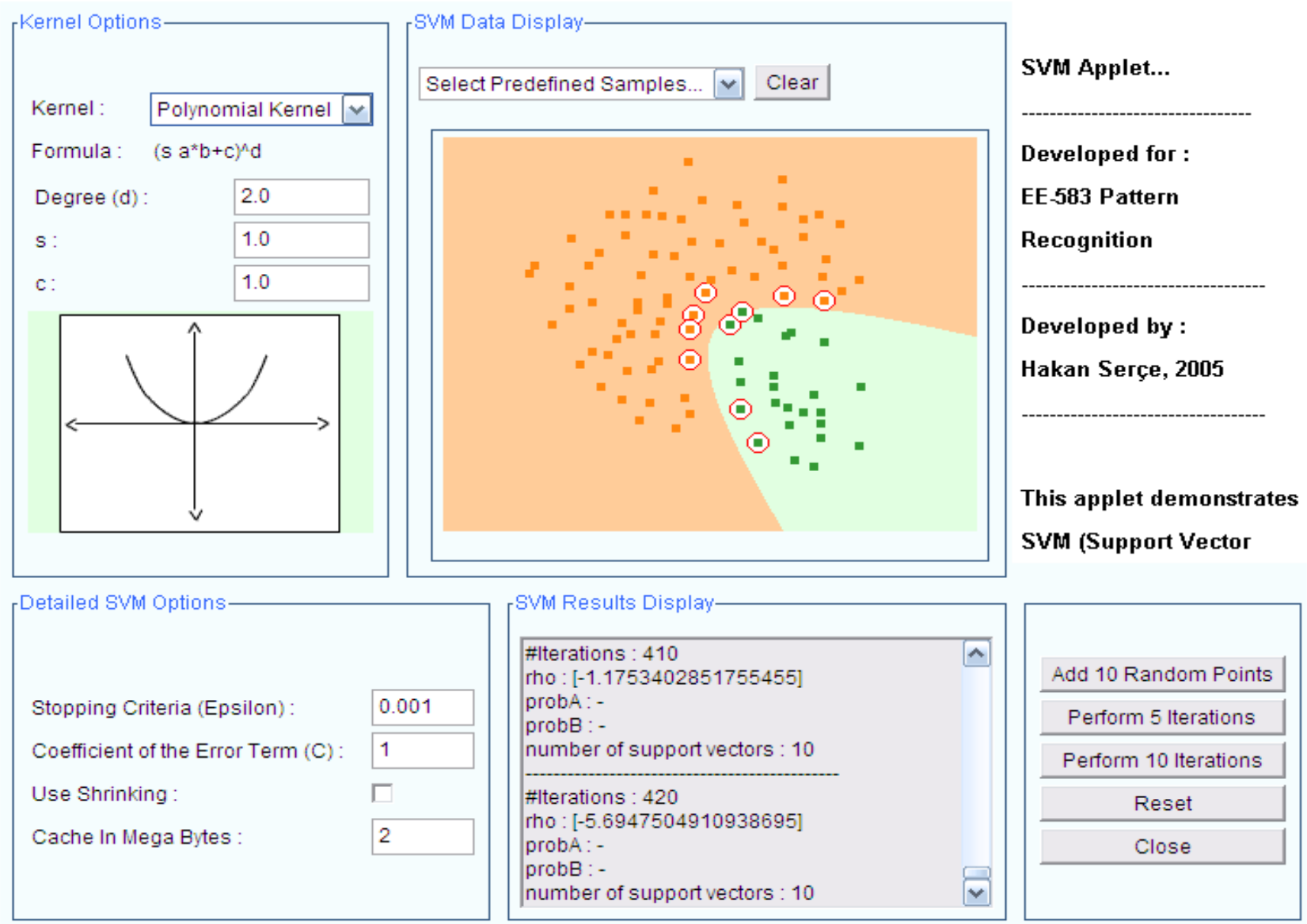
- More explanations or demonstrations can be found at:
  - http://www.support-vector-machines.org/index.html
  - Haykin Chp. 6 pp. 318-339
  - Burges tutorial (under/reading/)
    - Burges, CJC "A Tutorial on Support Vector Machines for Pattern Recognition" *Data Mining and Knowledge Discovery*, Vol 2 No 2, 1998.
  - http://www.dtreg.com/svm.htm

- **Software**
  - **SVM*light***, by Joachims, is one of the most widely used SVM classification and regression package. Distributed as C++ source and binaries for Linux, Windows, Cygwin, and Solaris. Kernels: polynomial, radial basis function, and neural (tanh).

  - **LibSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/** LIBSVM (Library for Support Vector Machines), is developed by Chang and Lin; also widely used. Developed in C++ and Java, it supports also multi-class classification, weighted SVM for unbalanced data, cross-validation and automatic model selection. It has interfaces for Python, R, Splus, MATLAB, Perl, Ruby, and LabVIEW. Kernels: linear, polynomial, radial basis function, and neural (tanh).

- Applet to play with:
  - http://lcn.epfl.ch/tutorial/english/svm/html/index.html
  - http://cs.stanford.edu/people/karpathy/svmjs/demo/


Kernel Options
Kernel :
Linear Kernel
Formula :
(s a*b+c)
SVM Data Display
Select Predefined Samples...
Clear
SVM Applet...
Developed for :
EE-583 Pattern
Recognition
Developed by :
Hakan Serçe, 2005
This applet demonstrates
SVM (Support Vector
Detailed SVM Options
Stopping Criteria (Epsilon) :
0.001
Coefficient of the Error Term (C) :
1
Use Shrinking :
Cache In Mega Bytes :
2
SVM Results Display
#Iterations : 250
rho : [-12.609075185863285]
probA : -
probB : -
number of support vectors : 3
#Iterations : 260
rho : [-12.609075185863285]
probA : -
probB : -
number of support vectors : 3
Add 10 Random Points
Perform 5 Iterations
Perform 10 Iterations
Reset
Close

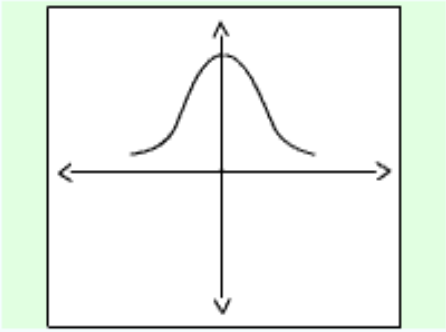

Kernel Options
Kernel :
Polynomial Kernel
Formula :
(s a*b+c)^d
Degree (d) :
2.0
s :
1.0
c :
1.0
SVM Data Display
Select Predefined Samples...
Clear
SVM Applet...
Developed for :
EE-583 Pattern
Recognition
Developed by :
Hakan Serçe, 2005
This applet demonstrates
SVM (Support Vector
Detailed SVM Options
Stopping Criteria (Epsilon) :
0.001
Coefficient of the Error Term (C) :
1
Use Shrinking :
Cache In Mega Bytes :
2
SVM Results Display
#Iterations : 410
rho : [-1.1753402851755455]
probA : -
probB : -
number of support vectors : 10
#Iterations : 420
rho : [-5.6947504910938695]
probA : -
probB : -
number of support vectors : 10
Add 10 Random Points
Perform 5 Iterations
Perform 10 Iterations
Reset
Close

Kernel Options

Kernel : RBF Kernel

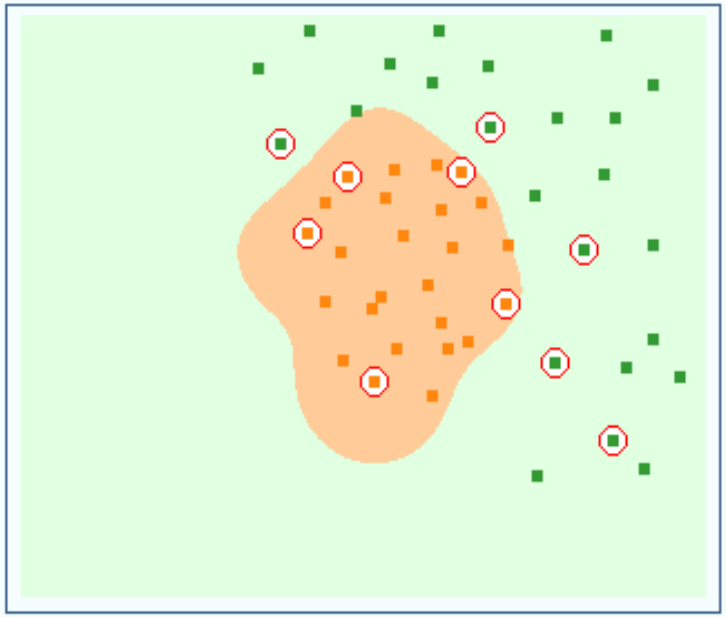
Formula : exp(-gamma ||a-b||^2)

Gamma : 0.0010



SVM Data Display

Select Predefined Samples... Clear



**SVM Applet...**

---

**Developed for :**

**EE-583 Pattern**

**Recognition**

---

**Developed by :**

**Hakan Serçe, 2005**

---

**This applet demonstrates**

**SVM (Support Vector**

Detailed SVM Options

Stopping Criteria (Epsilon) : 0.001

Coefficient of the Error Term (C) : 1

Use Shrinking :

Cache In Mega Bytes : 2

SVM Results Display

---

#Iterations : 5
rho : [0.3013952991115437]
probA : -
probB : -
number of support vectors : 10

Add 10 Random Points

Perform 5 Iterations

Perform 10 Iterations

Reset

Close