Foundations of Machine Learning

Part A: Logistic Regression

Logistic Regression for classification

- Linear Regression:

$$h(x) = \sum_{i=0}^{n} \beta_i x_i = \beta^T X$$

- Logistic Regression for classification:

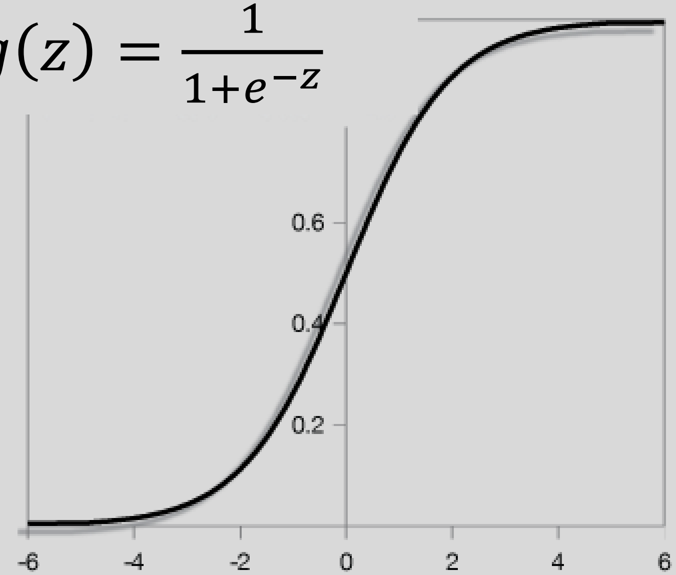
$$h_\beta(x) = \frac{1}{1 + e^{-\beta^T X}} = \mathrm{g}(\beta^T x)$$

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

is called the logistic function or the sigmoid function.

Logistic:

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



Sigmoid function properties

- Bounded between 0 and 1
- $g(z) \rightarrow 1$ as $z \rightarrow \infty$
- $g(z) \rightarrow 0$ as $z \rightarrow -\infty$

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

$$= \frac{1}{(1+e^{-z})^2}.e^{-z}$$

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

$$= g(z)(1-g(z)$$

Logistic Regression

- In logistic regression, we learn the conditional distribution P(y|x)
- Let $p_y(x; \beta)$ be our estimate of P(y|x), where β is a vector of adjustable parameters.
- Assume there are two classes, y = 0 and y = 1 and

$$P(y = 1|x) = h_\beta(x)$$
$$P(y = 0|x) = 1 - h_\beta(x)$$

- Can be written more compactly

$$P(y|x) = h(x)^y(1 - h(x))^{1-y}$$

- We can used the gradient method

Maximize likelihood

$$L(\beta) = p(\vec{y}|X;\beta)$$

$$= \prod_{i=1}^{m} p(y_i|x_i;\beta)$$

$$= \prod_{i=1}^{m} h(x_i\ \ )^{y_i}(1-h(x_i))^{1-y_i}$$

$$l(\beta) = \log\big(L(\beta)\big)$$

$$= \sum_{i=1}^{m} y^i \log h(x^i) + (1-y_i)(\log(1-h(x_i))$$

$$l(\beta) = \sum_{i=1}^{m} y^i \log h(x^i) + (1 - y_i)(\log(1 - h(x_i))$$

- How do we maximize the likelihood? Gradient ascent
  - Updates: $\beta = \beta + \alpha \nabla_\beta l(\beta)$

Assume one training example (x,y), and take derivatives to derive the stochastic gradient ascent rule.

$$\frac{\partial}{\partial \beta_j} l(\beta)$$

$$= \left(\left(y \frac{1}{g(\beta^T(x)}\right) - (1 - y)\frac{1}{1 - g(\beta^T x)}\right)\frac{\partial}{\partial \beta_j} g(\beta^T x)$$

$$= \left(\left(y \frac{1}{g(\beta^T(x)}\right) - (1 - y)\frac{1}{1 - g(\beta^T x)}\right) g(\beta^T x)(1 - g(\beta^T x)\frac{\partial}{\partial \beta_j}\beta^T x$$

$$= (y\left(1 - g(\beta^T x)\right) - (1 - y)g(\beta^T x))x_j$$

$$= (y - h_\beta(x))x_j$$

$$\beta = \beta + \alpha \nabla_\beta l(\beta)$$
$$\beta_j = \beta_j + \alpha\big(y^{(i)} - h_\beta\big(x^i\big)\big)x_j^{(i)}$$

Part B: Introduction to Support Vector Machine

Support Vector Machines

- SVMs have a clever way to prevent overfitting
- They can use many features without requiring too much computation.

Logistic Regression and Confidence

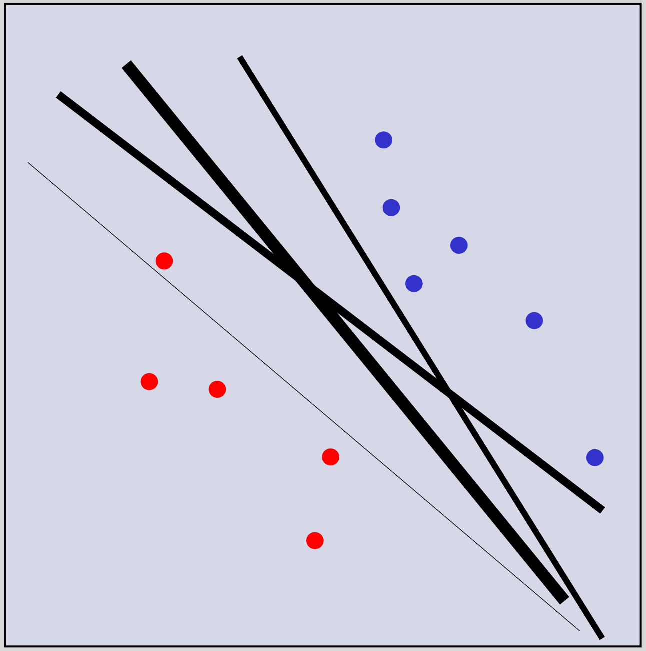
- Logistic Regression:

$$p(y = 1|x) = h_\beta(x) = g(\beta^T x)$$

- Predict 1 on an input x iff $h_\beta(x) \geq 0.5$, equivalently, $\beta^T x \geq 0$
- The larger the value of $h_\beta(x)$, the larger is the probability, and higher the confidence.
- Similarly, confident prediction of $y = 0$ if $\beta^T x \ll 0$
- More confident of prediction from points (instances) located far from the decision surface.

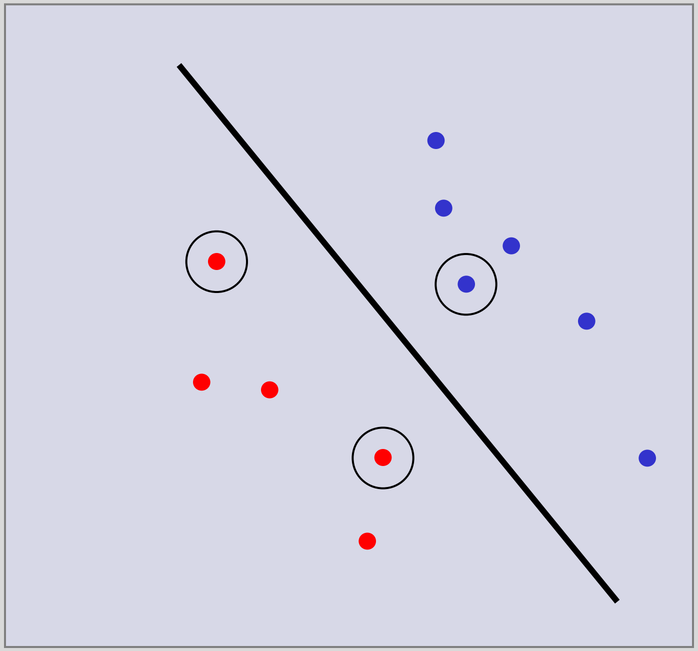
Preventing overfitting with many features

- Suppose a big set of features.
- What is the best separating line to use?
- Bayesian answer:
  - Use all
  - Weight each line by its posterior probability
- Can we approximate the correct answer efficiently?



Support Vectors

- The line that maximizes the minimum margin.
- This maximum-margin separator is determined by a subset of the datapoints.
  - called “support vectors”.
  - we use the support vectors to decide which side of the separator a test case is on.



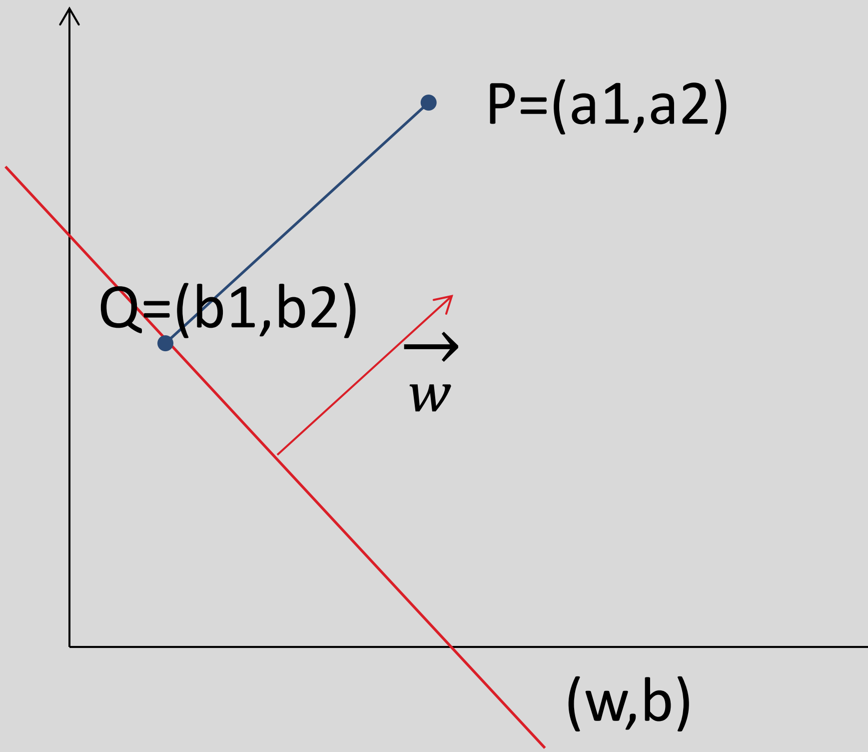
The support vectors are indicated by the circles around them.

Functional Margin

- Functional Margin of a point (x_i, y_i) wrt (w, b)
  - Measured by the distance of a point (x_i, y_i) from the decision boundary (w, b)
    $$\gamma^i = y_i(w^T x_i + b)$$
  - Larger functional margin →more confidence for correct prediction
  - Problem: w and b can be scaled to make this value larger
- Functional Margin of training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ wrt (w, b) is
  $$\gamma = \min_{1 \le i \le m} \gamma^i$$

Geometric Margin

- For a decision surface (w, b)
- the vector orthogonal to it is given by w.
- The unit length orthogonal vector is $\frac{w}{\|w\|}$
- $P = Q + \gamma \frac{w}{\|w\|}$



Geometric Margin

$$P = Q + \gamma \frac{w}{\|w\|}$$

$$(b1, b2) = (a1, a2) - \gamma \frac{w}{\|w\|}$$

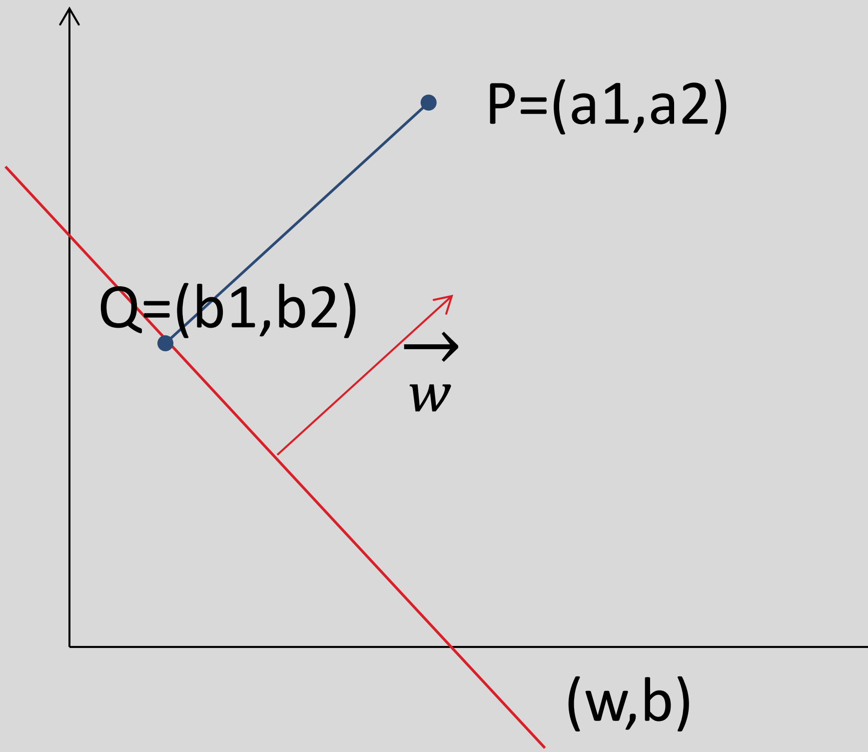
$$\rightarrow \; w^T \left((a1, a2) - \gamma \frac{w}{\|w\|} \right) + b = 0$$

$$\rightarrow \gamma = \frac{w^T (a1, a2) + b}{\|w\|}$$

$$= \frac{w}{\|w\|}^T (a1, a2) + \frac{b}{\|w\|}$$

$$= \frac{w}{\|w\|}^T (a1, a2) + \frac{b}{\|w\|}$$

$$\gamma = y.\left(\frac{w}{\|w\|}^T (a1, a2) + \frac{b}{\|w\|}\right)$$



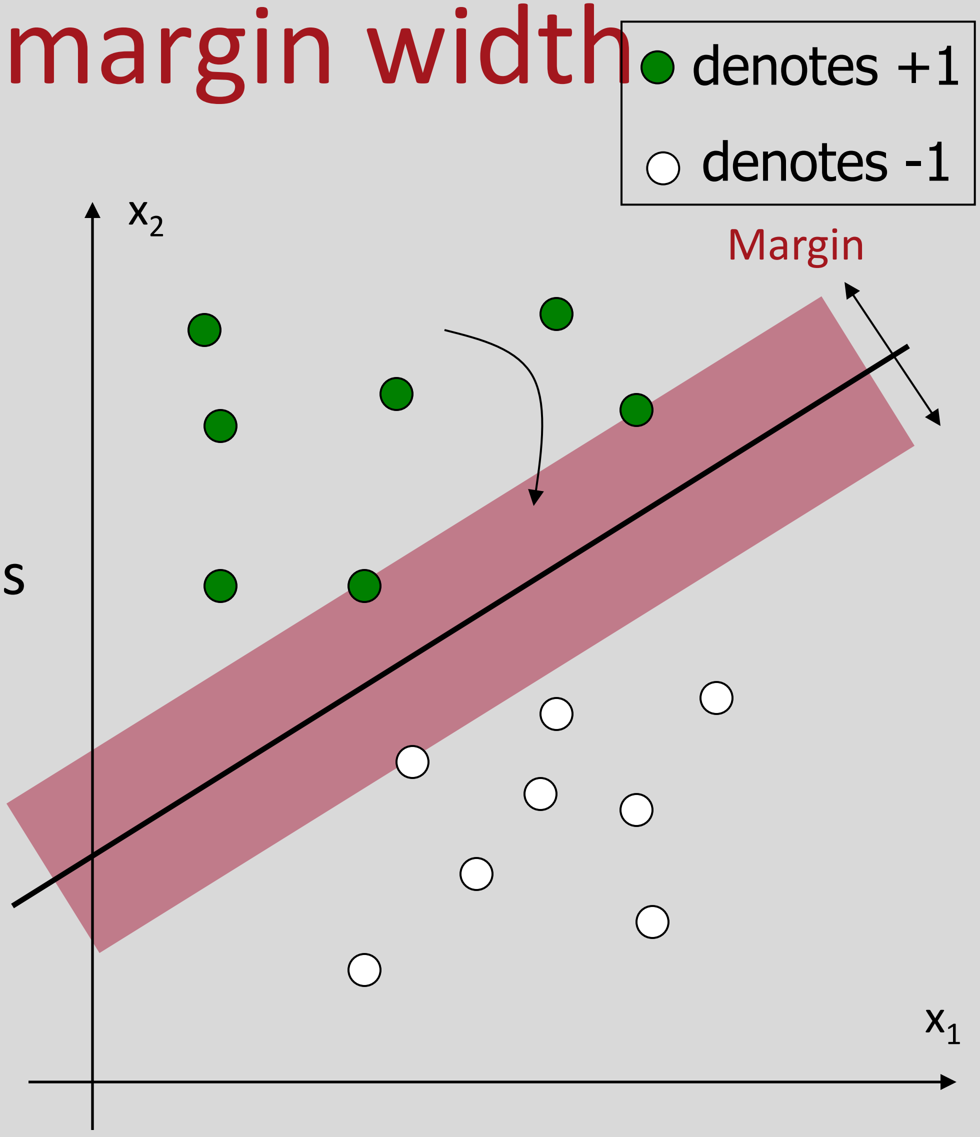
Geometric margin : $\|w\| = 1$

Geometric margin of (w,b) wrt S=$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$

-- smallest of the geometric margins of individual points.

Maximize margin width

- Assume linearly separable training examples.
- The classifier with the maximum margin width is robust to outliners and thus has strong generalization ability



Maximize Margin Width

- Maximize $\frac{\gamma}{||w||}$ subject to
- $y_i(w^T x_i + b) \geq \gamma$ for $i = 1,2, \ldots, m$
- Scale so that $\gamma = 1$
- Maximizing $\frac{1}{||w||}$ is the same as minimizing $||w||^2$
- Minimize $\boldsymbol{w}.\boldsymbol{w}$ subject to the constraints
- for all $(x_i, y_i), i = 1, \ldots, m$:

  $w^T x_i + b \geq 1$ if $y_i = 1$

  $w^T x_i + b \leq -1$ if $y_i = -1$

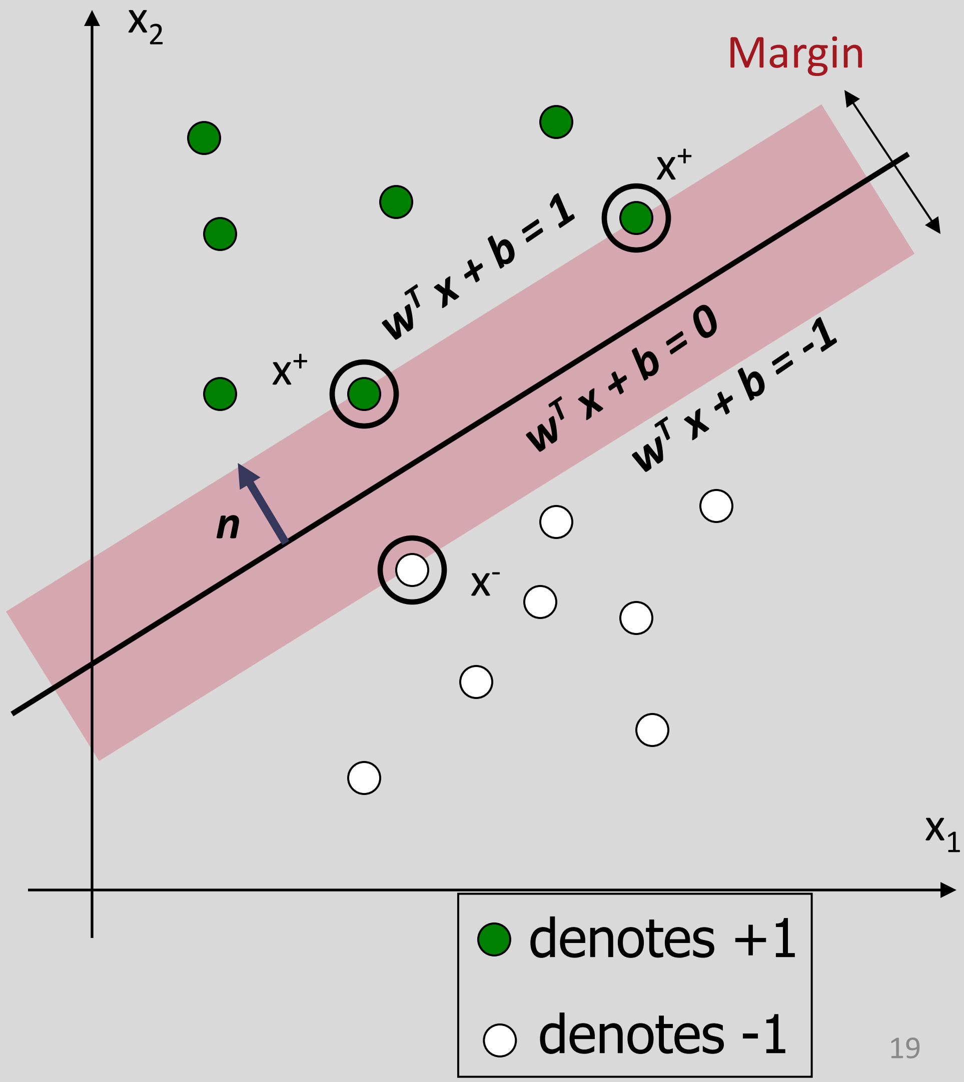
Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

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



Solving the Optimization Problem

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

- Optimization problem with convex quadratic objectives and linear constraints
- Can be solved using QP.
- Lagrange duality to get the optimization problem's dual form,
  - Allow us to use kernels to get optimal margin classifiers to work efficiently in very high dimensional spaces.
  - Allow us to derive an efficient algorithm for solving the above optimization problem that will typically do much better than generic QP software.

Part C: Support Vector Machine: Dual

Lagrangian Duality in brief

The Primal Problem

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \le 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

The generalized Lagrangian:

$$\mathrm{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

the α's ($\alpha_i \geq 0$) and β's are called the Lagrange multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \ge 0} \mathrm{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \ge 0} \mathrm{L}(w, \alpha, \beta)$$

Lagrangian Duality, cont.

The Primal Problem $\boxed{p^* = \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathrm{L}(w,\alpha,\beta)}$

The Dual Problem: $\boxed{d^* = \max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathrm{L}(w,\alpha,\beta)}$

Theorem (weak duality):

$$\boxed{d^* = \max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathrm{L}(w,\alpha,\beta) \ \leq \ \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathrm{L}(w,\alpha,\beta) = p^*}$$

Theorem (strong duality):

Iff there exist a saddle point of $L(w,\alpha,\beta)$, we have $\boxed{d^* = p^*}$

The KKT conditions

If there exists some saddle point of *L*, then it satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial w_i} \mathrm{L}(w, \alpha, \beta) = 0, \quad i = 1, \ldots, k$$

$$\frac{\partial}{\partial \beta_i} \mathrm{L}(w, \alpha, \beta) = 0, \quad i = 1, \ldots, l$$

$$\alpha_i g_i(w) = 0, \quad i = 1, \ldots, m$$

$$g_i(w) \le 0, \quad i = 1, \ldots, m$$

$$\alpha_i \ge 0, \quad i = 1, \ldots, m$$

Theorem: If *w**, *a** and *b** satisfy the KKT condition, then it is also a solution to the primal and the dual problems.

Support Vectors

- Only a few α_i's can be nonzero
- Call the training data points whose α_i's are nonzero the support vectors

$$\boxed{\alpha_i g_i(w) = 0, \quad i = 1, \ldots, m}$$

If $\alpha_i > 0$ then $g(w) = 0$

Solving the Optimization Problem

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Lagrangian Function

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Solving the Optimization Problem

$$\text{minimize } \; L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \qquad \alpha_i \geq 0$$

Minimize wrt w and b for fixed α

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \;\Rightarrow\; \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\mathrm{L}_p(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - b \sum_{i=1}^{m} \alpha_i y_i$$

$$\mathrm{L}_p(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

The Dual problem

Now we have the following dual opt problem:

$$\max_\alpha \mathrm{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

This is a **quadratic programming** problem.

– A global maximum of α_i can always be found.

Support vector machines

- Once we have the Lagrange multipliers $\{\alpha_j\}$ we can reconstruct the parameter vector w as a weighted combination of the training examples:

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

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

- For testing with a new data ***z***
  - Compute

$$\boxed{w^T z + b = \sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b}$$

  and classify ***z*** as class 1 if the sum is positive, and class 2 otherwise

Note: *w* need not be formed explicitly

Solving the Optimization Problem

- The discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in \mathrm{SV}} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- It relies on a *dot product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- Solving the optimization problem involved computing the dot products $\boldsymbol{x}_i^T\boldsymbol{x}_j$ between all pairs of training points
- The optimal $\boldsymbol{w}$ is a linear combination of a small number of data points.

Part D: SVM – Maximum Margin with Noise

Linear SVM formulation

Find **w** and *b* such that

$$\frac{2}{||w||} \text{ is maximized}$$

And for each of the m training points (x_i, y_i),

$$y_i(w.x_i + b) \geq 1$$

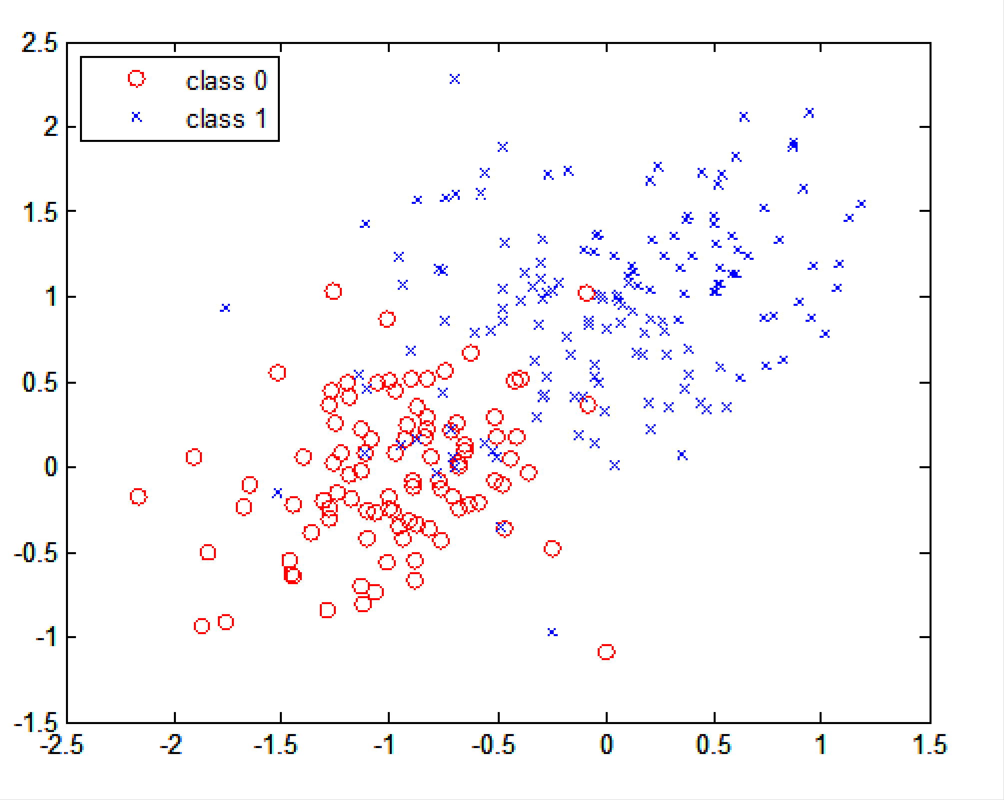
Find **w** and *b* such that

$$||w||^2 = w.w \text{ is minimized}$$

And for each of the m training points (x_i, y_i),

$$y_i(w.x_i + b) \geq 1$$

Limitations of previous SVM formulation



- What if the data is not linearly separable?
- Or noisy data points?

Extend the definition of maximum margin to allow non-separating planes.

How to formulate?

- Minimize $\|w\|^2 = w.w$ and *number of misclassifications*, i.e., minimize
$$w.w + \#(\text{training errors})$$

- No longer QP formulation

Objective to be minimized

- Minimize

  $w.w$
  $+ C(\text{distance of error points to their correct zones})$

- Add slack variables ξ_i

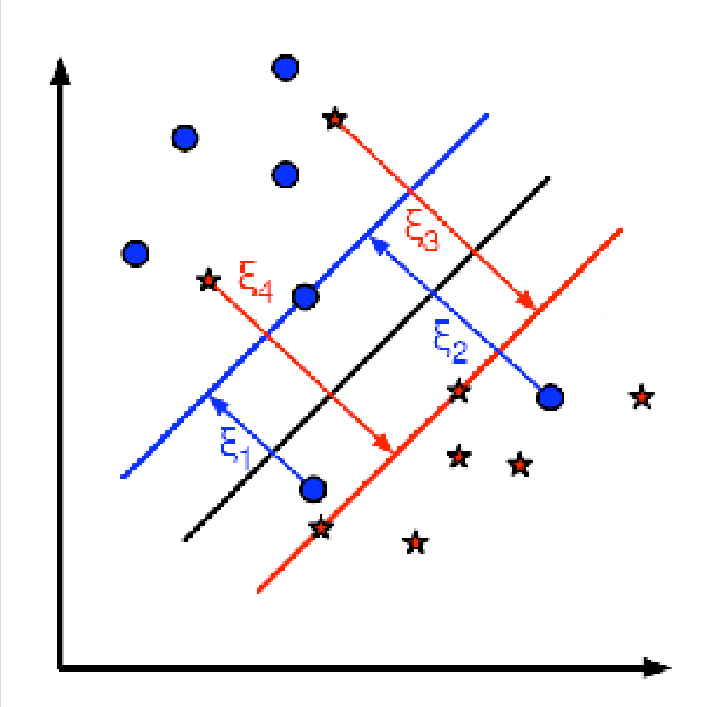
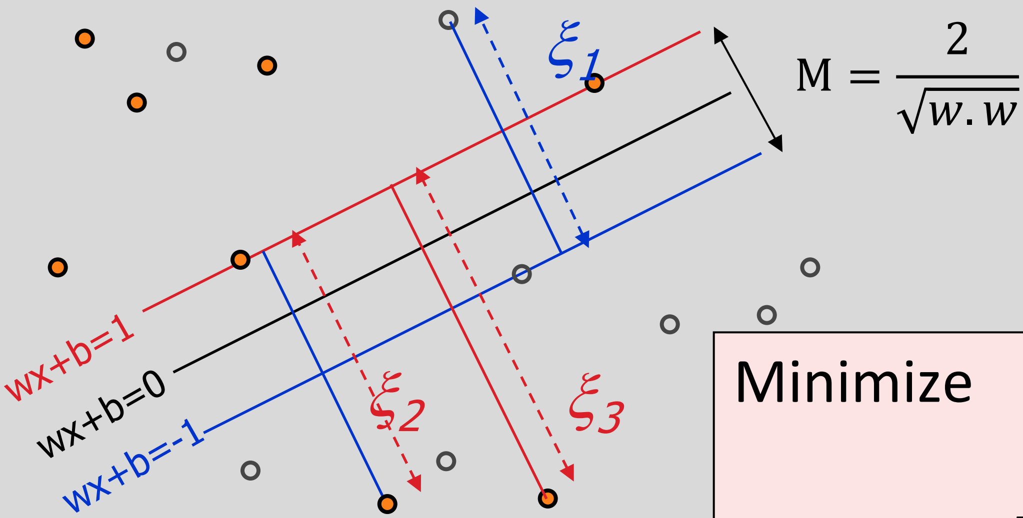


Figure from http://docs.opencv.org/2.4/doc/tutorials/ml/non_linear_svms/

Maximum Margin with Noise



Minimize

$$w.w + C\sum_{k=1}^{m}\xi_k$$

m constraints

$$\begin{bmatrix} w.x_k + b \geq 1 - \xi_k \text{ if } y_k = 1 \\ w.x_k + b \leq -1 + \xi_k \text{ if } y_k = -1 \end{bmatrix}$$

$$\equiv$$

$$\boldsymbol{y_k(w.x_k + b) \geq 1 - \xi_k}, \quad k=1,\ldots,m$$

$$\boldsymbol{\xi_k \geq 0}, \quad k=1,\ldots,m$$

C controls the relative importance of maximizing the margin and fitting the training data.
Controls overfitting.

Lagrangian

$$L(w, b, \xi, \alpha, \beta)$$
$$= \frac{1}{2} w.w + C \sum_{i=1}^{m} \xi_i$$
$$+ \sum_{i=1}^{m} \alpha_i [y_i (x.w + b) - 1 + \xi_i] - \sum_{i=1}^{m} \beta_i \xi_i$$

α_i's and β_i's are Lagrange multipliers (≥ 0).

Dual Formulation

Find $\alpha_1, \alpha_2, \ldots, \alpha_m$ s.t.

$$\max_\alpha \mathrm{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Linear SVM

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

Noise Accounted

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

Solution to Soft Margin Classification

- x_i with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$b = y_k(1 - \xi_k) - \sum_{i=1}^{m} \alpha_i y_i x_i x_k$$

for any k s.t. $\alpha_k > 0$

For classification,

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i x_i . x \ + b$$

(no need to compute w explicitly)

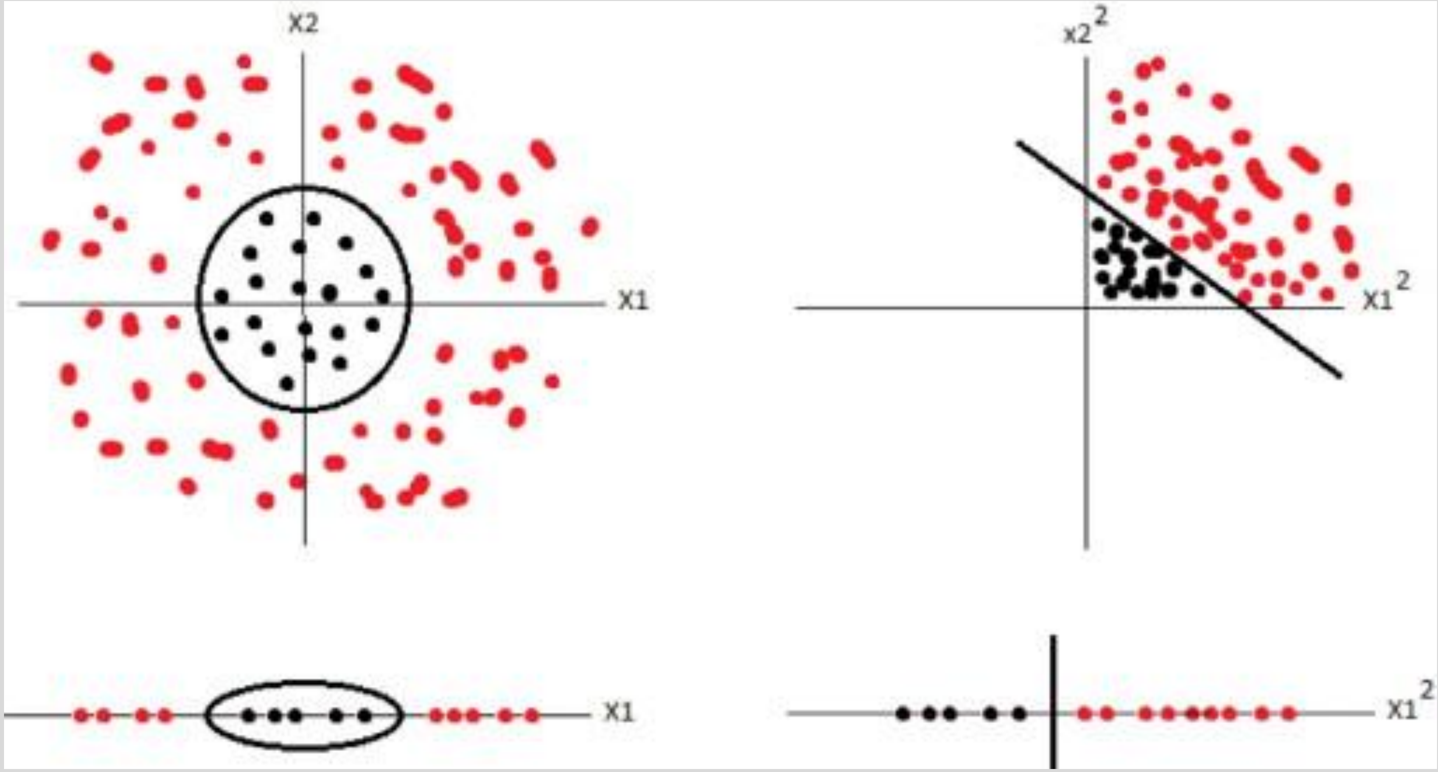
Part E: Nonlinear SVM and Kernel function

Non-linear decision surface

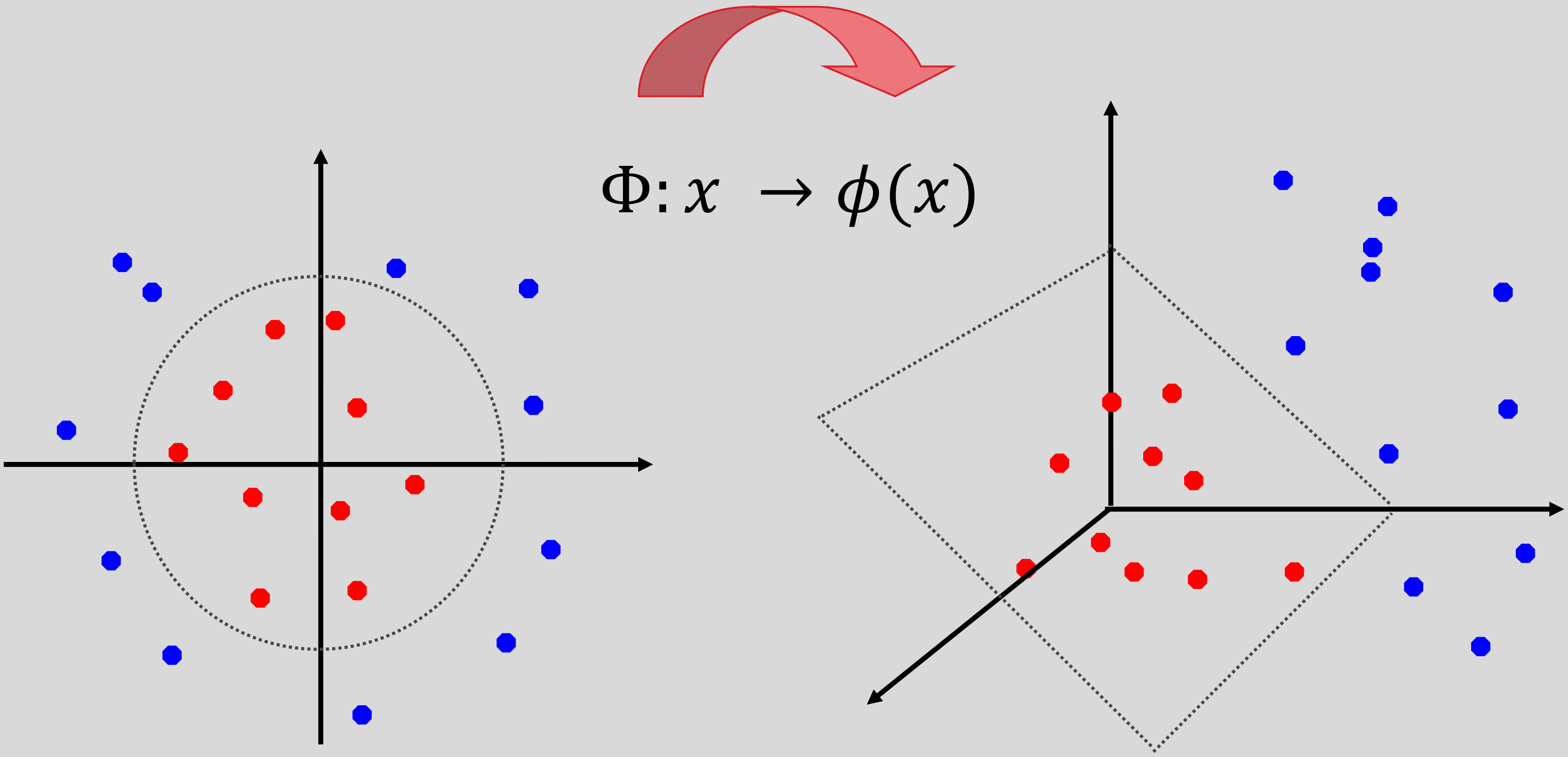
- We saw how to deal with datasets which are linearly separable with noise.
- What if the decision boundary is truly non-linear?
- Idea: Map data to a high dimensional space where it is linearly separable.
  - Using a bigger set of features will make the computation slow?
  - The "kernel" trick to make the computation fast.

Non-linear SVMs: Feature Space

$$\Phi : x \rightarrow \phi(x)$$



Non-linear SVMs: Feature Space



This slide is from *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Kernel

- Original input attributes is mapped to a new set of input features via feature mapping Φ.
- Since the algorithm can be written in terms of the scalar product, we replace $x_a . x_b$ with $\phi(x_a) . \phi(x_b)$
- For certain Φ's there is a simple operation on two vectors in the low-dim space that can be used to compute the scalar product of their two images in the high-dim space

$$K(x_a, x_b) = \phi(x_a) . \phi(x_b)$$

Let the kernel do the work rather than do the scalar product in the high dimensional space.

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- We only use the dot product of feature vectors in both the training and test.
- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(x_a, x_b) = \phi(x_a).\phi(x_b)$$

The kernel trick

$$K(x_a, x_b) = \phi(x_a).\phi(x_b)$$

Often $K(x_a, x_b)$ may be very inexpensive to compute even if $\phi(x_a)$ may be extremely high dimensional.

Kernel Example

2-dimensional vectors $\overline{\boldsymbol{x}} = [x_1 x_2]$

let $\boldsymbol{K(x_i, x_j) = (1 + x_i . x_j)^2}$

We need to show that $K(x_i, x_j) = \phi(x_i).\phi(x_j)$

$K(x_i,x_j) = (1 + x_i x_j)^2,$

$= 1 + x_{i1}^2 x_{j1}^2 + 2\, x_{i1} x_{j1}\, x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1} x_{j1} + 2x_{i2} x_{j2}$

$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1} x_{i2}\ \ x_{i2}^2\ \ \sqrt{2} x_{i1}\ \ \sqrt{2} x_{i2}].[1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1} x_{j2}\ \ x_{j2}^2\ \ \sqrt{2} x_{j1}\ \ \sqrt{2} x_{j2}]$

$= \phi(x_i).\phi(x_j),$

where $\phi(x) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1 x_2\ \ x_2^2\ \ \sqrt{2} x_1\ \ \sqrt{2} x_2]$

Commonly-used kernel functions

- Linear kernel: $K(x_i, x_j) = x_i . x_j$
- Polynomial of power *p*:

$$K(x_i, x_j) = (1 + x_i . x_j)^p$$

- Gaussian (radial-basis function):

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

- Sigmoid

$$K(x_i, x_j) = \tanh(\beta_0 x_i . x_j + \beta_1)$$

In general, functions that satisfy *Mercer's condition* can be kernel functions.

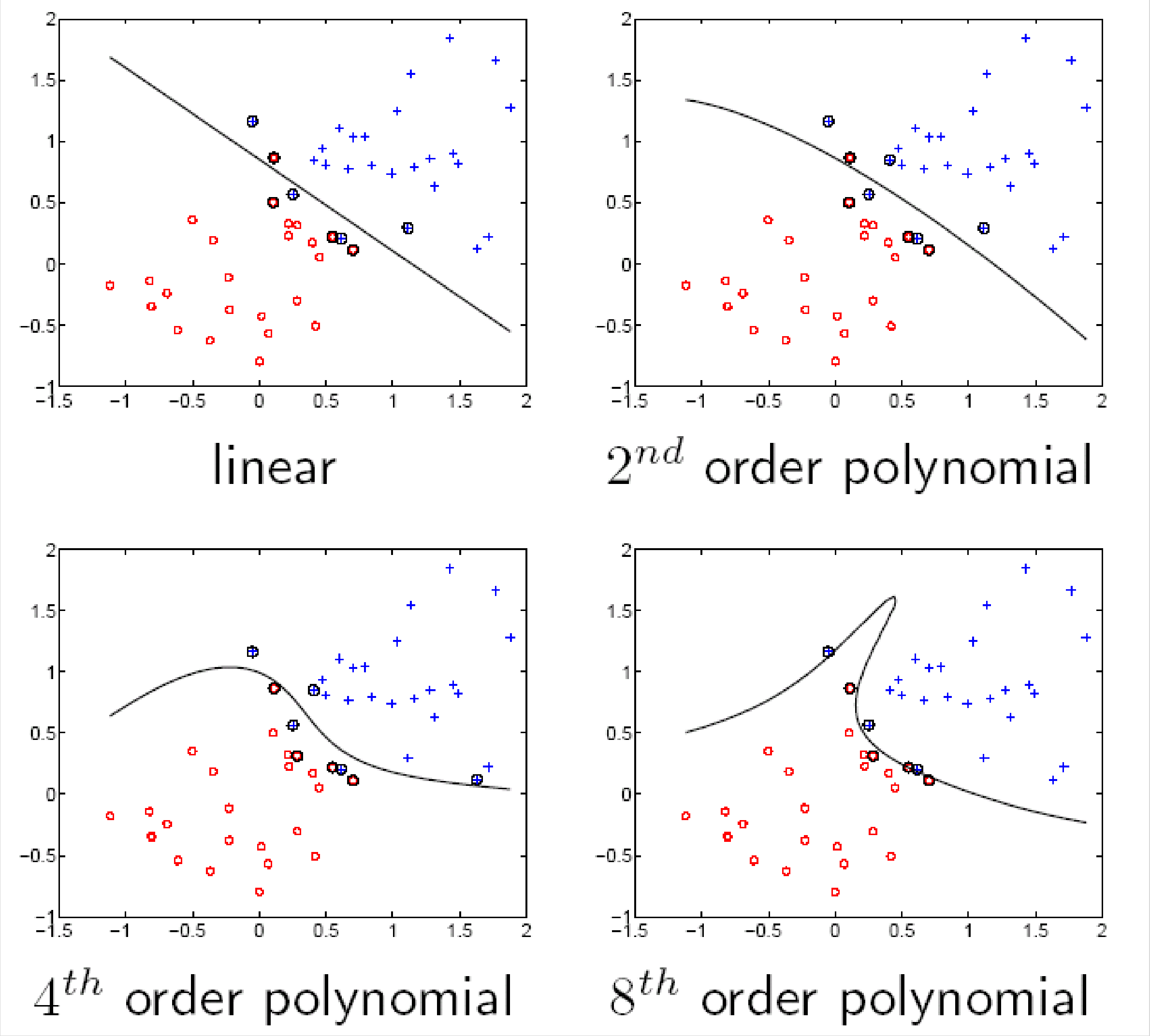
Kernel Functions

- Kernel function can be thought of as a similarity measure between the input objects
- Not all similarity measure can be used as kernel function.
- Mercer's condition states that any positive semi-definite kernel *K*(*x*, *y*), i.e.

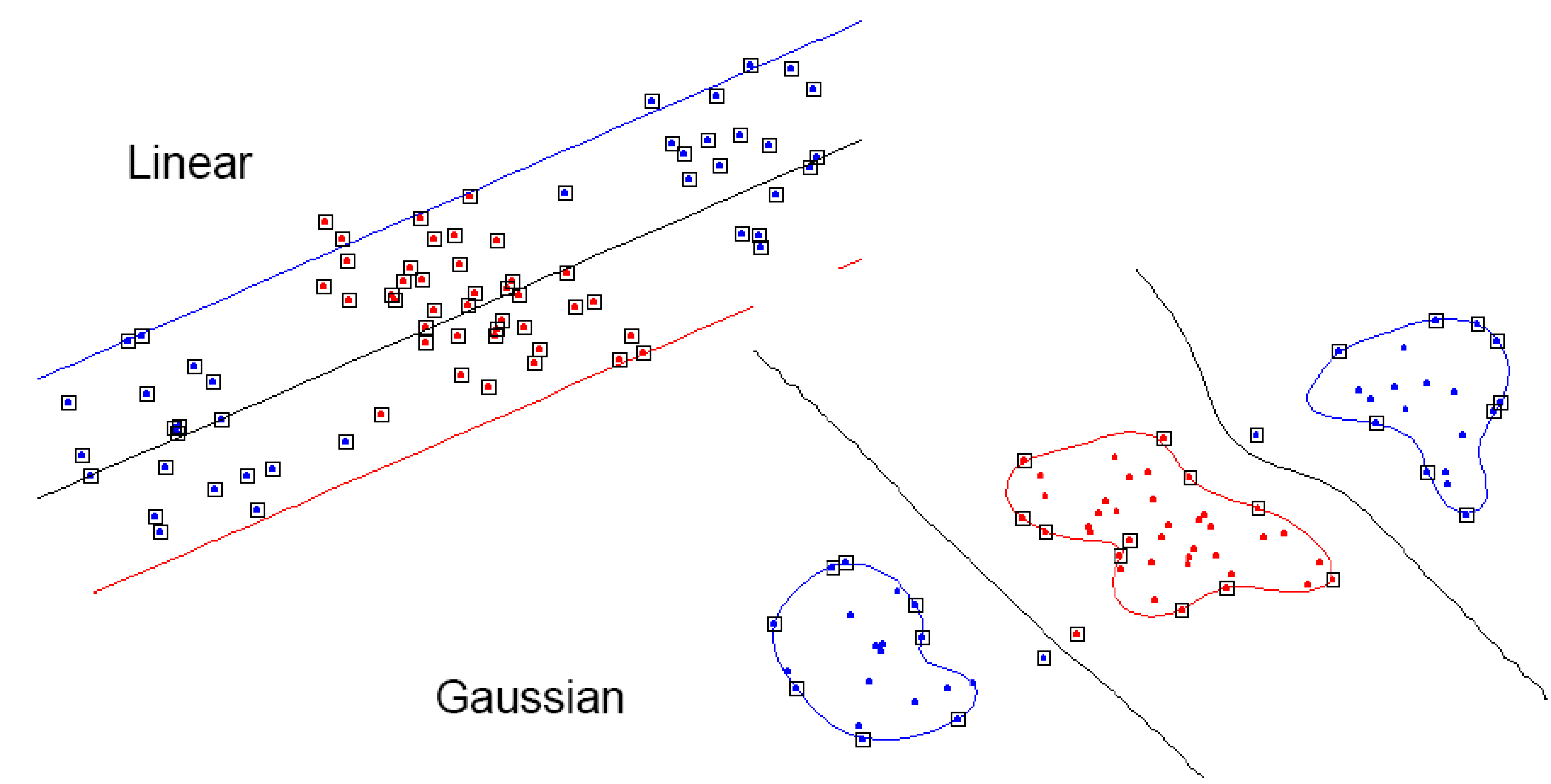
$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

- can be expressed as a dot product in a high dimensional space.

SVM examples



Examples for Non Linear SVMs – Gaussian Kernel



Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

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

$$\text{such that} \quad 0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(x) = \sum_{i \epsilon SV} \alpha_i K(x_i, x_j) + b$$

Performance

- Support Vector Machines work very well in practice.
  - The user must choose the kernel function and its parameters
- They can be expensive in time and space for big datasets
  - The computation of the maximum-margin hyper-plane depends on the *square of the number of training cases.*
  - We need to store all the support vectors.
- The kernel trick can also be used to do PCA in a much higher-dimensional space, thus giving a non-linear version of PCA in the original space.

Multi-class classification

- SVMs can only handle two-class outputs
- Learn N SVM's
  - SVM 1 learns Class1 vs REST
  - SVM 2 learns Class2 vs REST
  - :
  - SVM N learns ClassN vs REST
- 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.

Part F: SVM – Solution to the Dual Problem

The SMO algorithm

The SMO algorithm can efficiently solve the dual problem.
First we discuss Coordinate Ascent.

Coordinate Ascent

- Consider solving the unconstrained optimization problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_n)$$

Loop until convergence: {

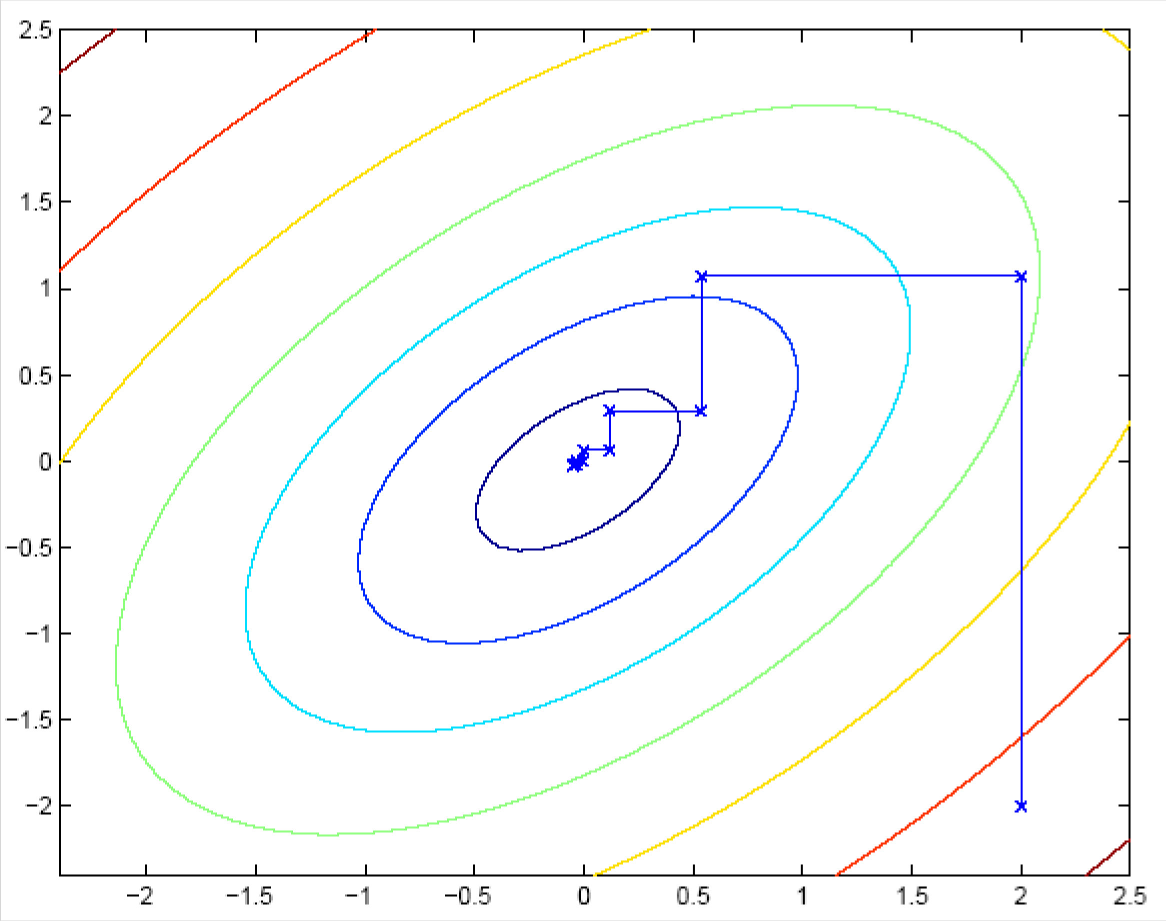
for $i = 1 \; to \; n$ {

$$\alpha_i = arg \max_{\widehat{\alpha_i}} W(\alpha_1, \ldots, \widehat{\alpha_i}, \ldots, \alpha_n);$$

}

}

Coordinate ascent



- Ellipses are the contours of the function.
- At each step, the path is parallel to one of the axes.

Sequential minimal optimization

- Constrained optimization:

$$\max_\alpha \ \mathrm{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i . \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Question: can we do coordinate along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

The SMO algorithm

$$\max_{\alpha} \mathrm{W}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i . \mathbf{x}_j)$$

$$\begin{aligned} \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned}$$

- Choose a set of α_1's satisfying the constraints.
- α_1 is exactly determined by the other α's.
- We have to update at least two of them simultaneously to keep satisfying the constraints.

The SMO algorithm

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Re-optimize W(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i; j$) fixed.

}

- The update to α_i and α_j can be computed very efficiently.

Thank You