
COMP 435 Spring 2010

Lecture 06

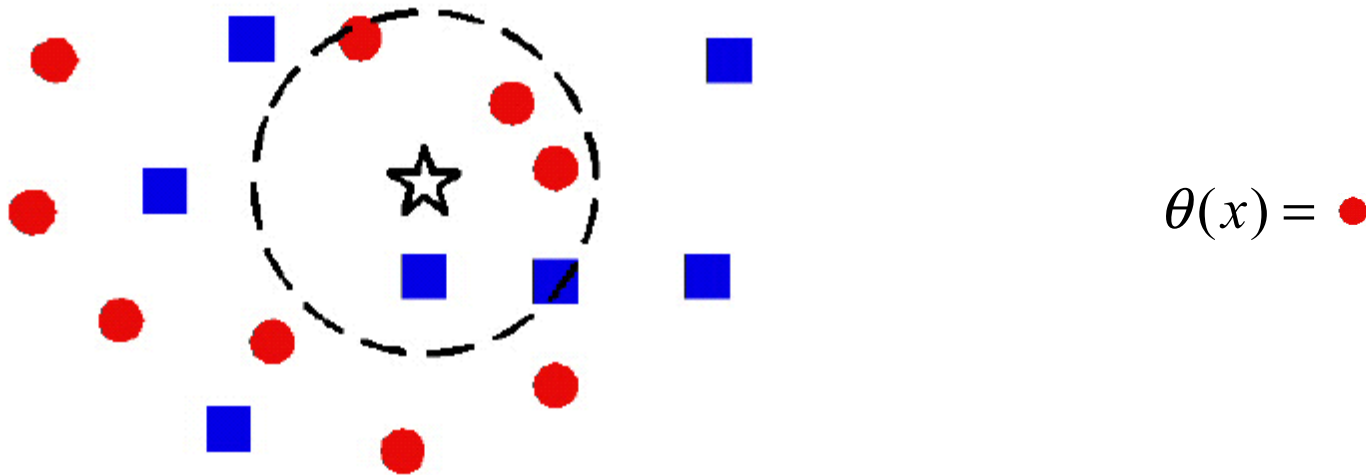
Machine Learning II

Re-CAP Machine Learning

- kNN classifier:

- ▶ The labelled dataset $\mathcal{D}_n = \{(\mathbf{x}_1, \theta_1), \dots, (\mathbf{x}_n, \theta_n)\}$
- ▶ where $\theta_i \in \{\omega_1, \dots, \omega_c\}$ is the class label for \mathbf{x}_i .
- ▶ For an input \mathbf{x} , find its k nearest neighbors
- ▶ The k -nearest-neighbor decision rule

$\mathbf{x} \rightarrow \{majority\ class\ label\ of\ the\ k\ nearest\ neighbors\}$



kNN classifier challenges

- **Large Data Set Size**
 - Complex decision boundaries
 - Access/Storage
- **Solutions:**
 - Editing:
 - » merge cells, simplify decision boundaries
 - Condensing:
 - » keep only data points that can influence decision boundaries
 - Indexing:
 - » Improve access, but not reduce storage size

Bayesian Classifier with Gaussian Distribution Data

- Bayesian Decision:

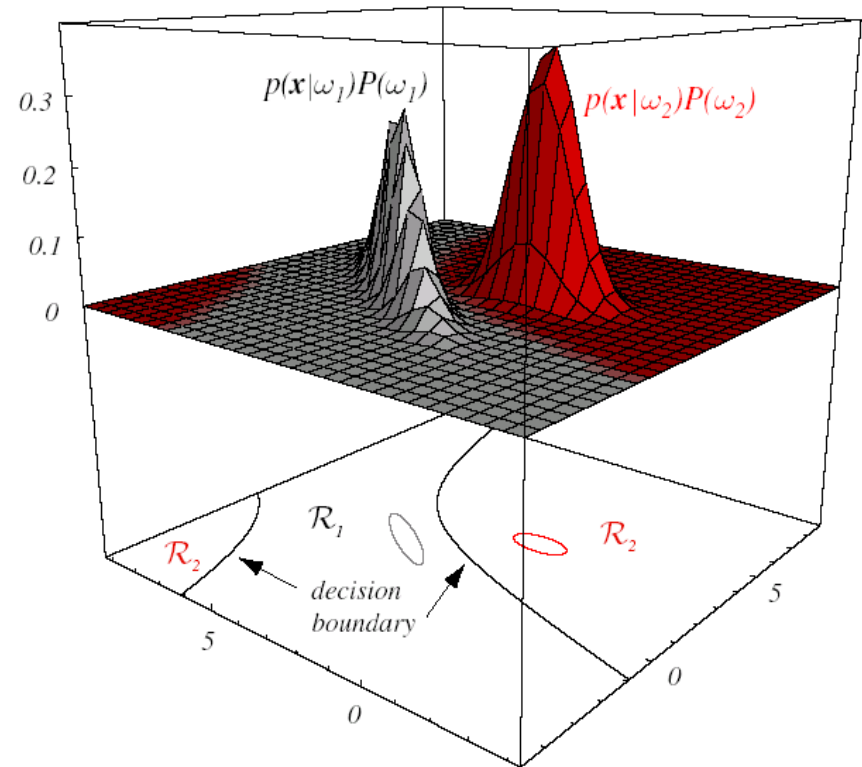
- ▶ $p(\mathbf{x}|\omega_i)$ Likelihood
- ▶ $p(\omega_i)$ Prior
- ▶ $p(\omega_i|\mathbf{x})$ Posterior
- ▶ Bayes Rule

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)}$$

- ▶ In other words

posterior \propto likelihood \times prior

- Apply to Gaussian Data:
 - General case, each class has arbitrary Gaussian distribution : decision boundary is quadratic
 - When have identical Gaussian for each class: decision boundary is linear
- Matlab Implementation:
 - `[rec, err]=classify(x1, x0,y0, method)`
 - Method = 'linear', 'quadratic'
 - Training data: x0, y0
 - Test data: x1



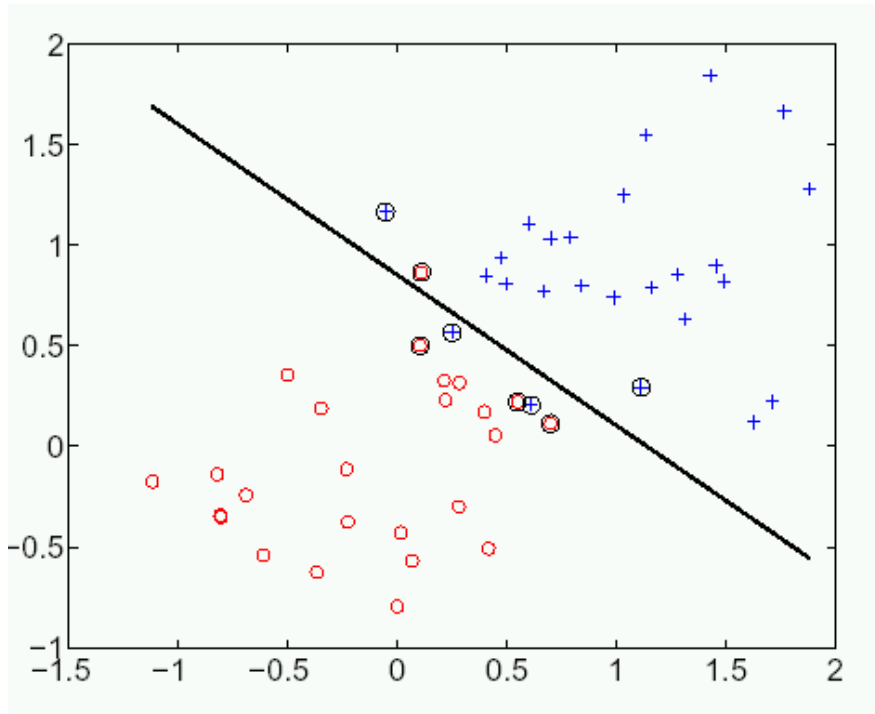
Support Vector Machine Classifier

Outline

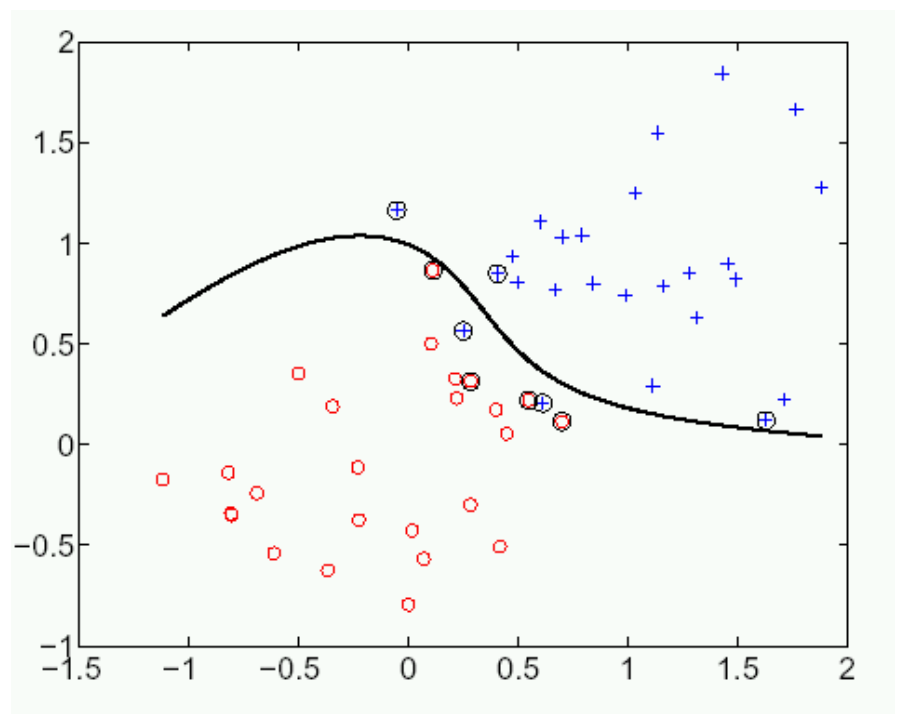
- **What is SVM ?**
- **Why SVM is good ?**
- **Mathematics**
- **Compute SVM in Matlab**

Support Vectors

- Not all training data points are relevant to the decision boundary:
 - Only those “support” the shape of decision boundary are important.
 - Remember the Condensing in kNN classifier.



Linear



Non-Linear

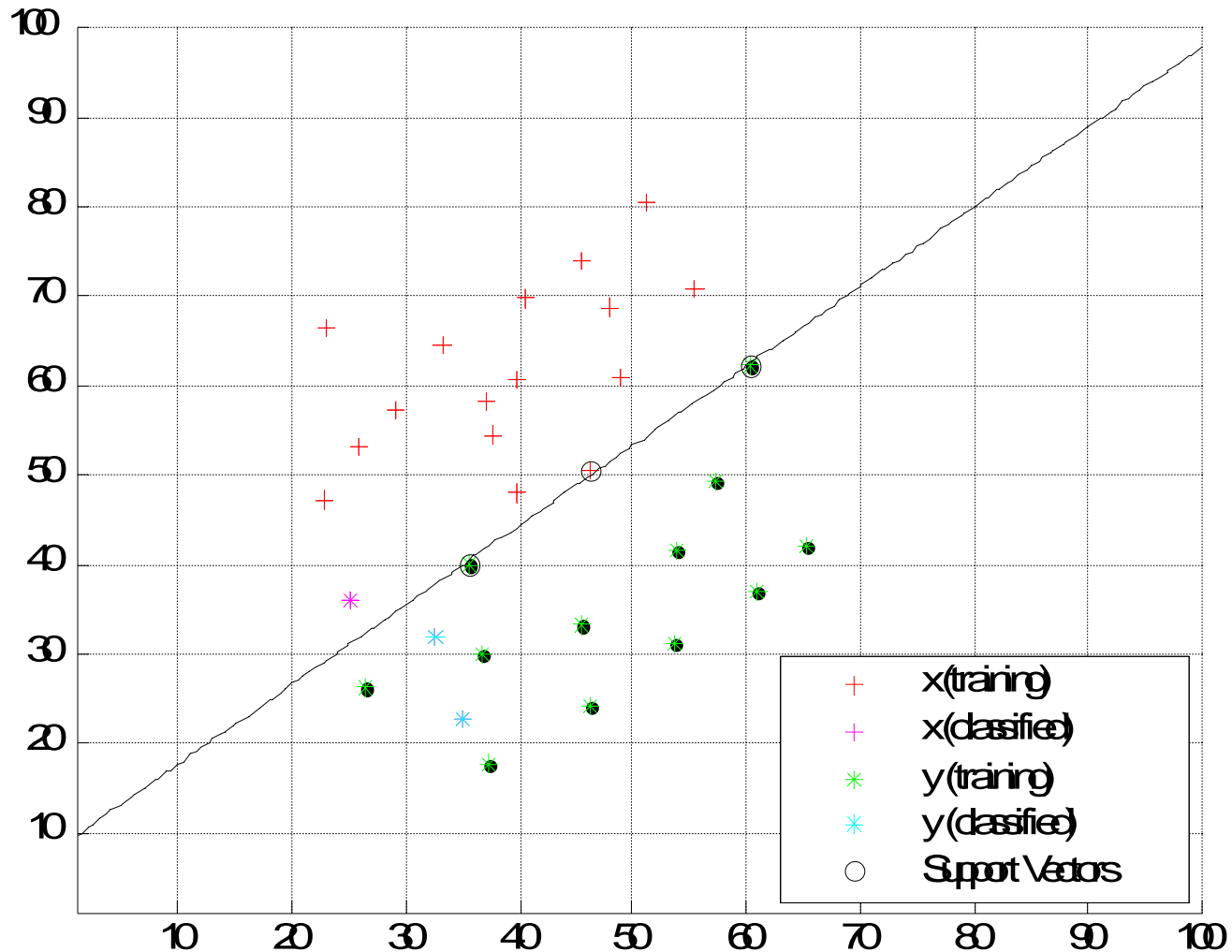
Support Vector Machine

- **SVM**

- Find a decision boundary that maximizes the separation between two classes
- Decision boundary is determined by a small number of training data points called “support vectors”
- Can have non-linear boundary by replacing inner product in the original space with a kernel function.

SVM Example from Matlab

Check out matlab tutorial : `svm_tutorials.m`



Mathematics Behind SVM

- A “good” decision boundary
- VC Complexity
- Gap maximizing objective
- Dual Decomposition solution (Lagrangian Relaxation)
- Kernel Tricks for obtaining non-linear boundaries

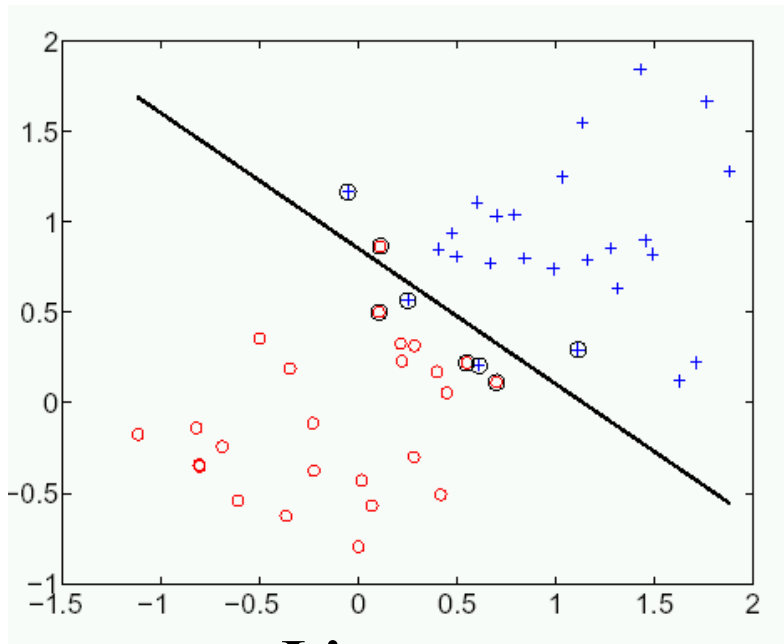
Decision Function

- $\{(x_j, y_j) | j=1..N\}$ are N samples from an i.i.d source with prob $P(x,y)$. $\{x_j\}$ are data/features, for eg, face images $\{y_j\}$ are discrete labels, for eg, {joe,mike,kathy} in classification, and continuous, in regression
- Decision function: $f(X) \rightarrow Y$, gives correct label to input feature X .
- How to find $f(x)$? Minimizing the expected loss function (eg, 0/1 loss, squared):
- $L(f(x), y)$ - loss function. Typically if not correct, 1.

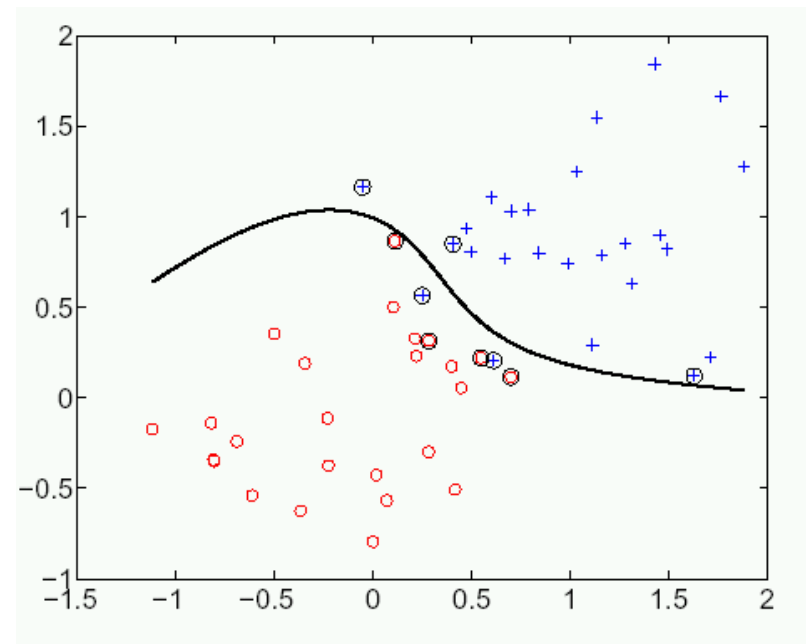
$$f^* = \arg \min_f \underbrace{\int L(f(x), y) dP(x, y)}_{R(f): \text{Expected Risk Function}}$$

A Good Decision Function ?

- Eg, There are a lot of ways to separate two classes, which one is the best ? – minimizing the expected risk, $R(f)$, but...



Linear



Polynomial

Empirical Risk

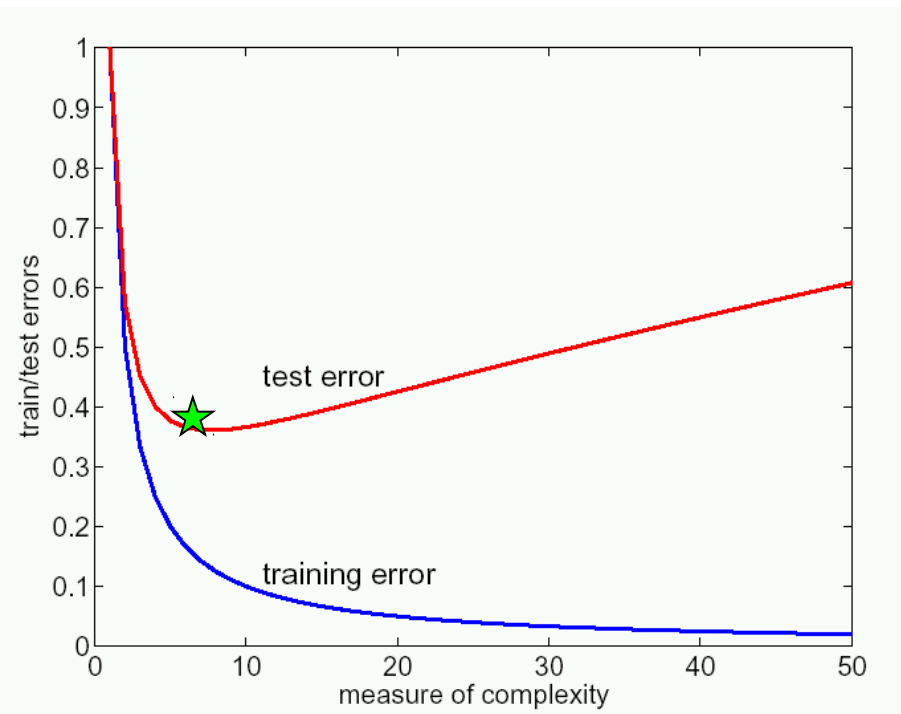
- **Minimizing empirical risk (fitting the known training data):**

$$f^* = \arg \min_f R^E(f) = (1/n) \sum_{j=1}^N L(f(x_j), x_j)$$

- **What is wrong ?**

Empirical Risk

- Minimizing empirical risk may over fitting



$$f^* = \arg \min_f R^E(f):$$

$$R^E(f) = (1/n) \sum_{j=1}^N L(f(x_j), x_j)$$

- **Over fitting** is related to the complexity of $f()$.

Structural Risk

- A good $f()$ is the one minimizing the $R(f)$, not $R^E(f)$.
- This desire is expressed in inequality:

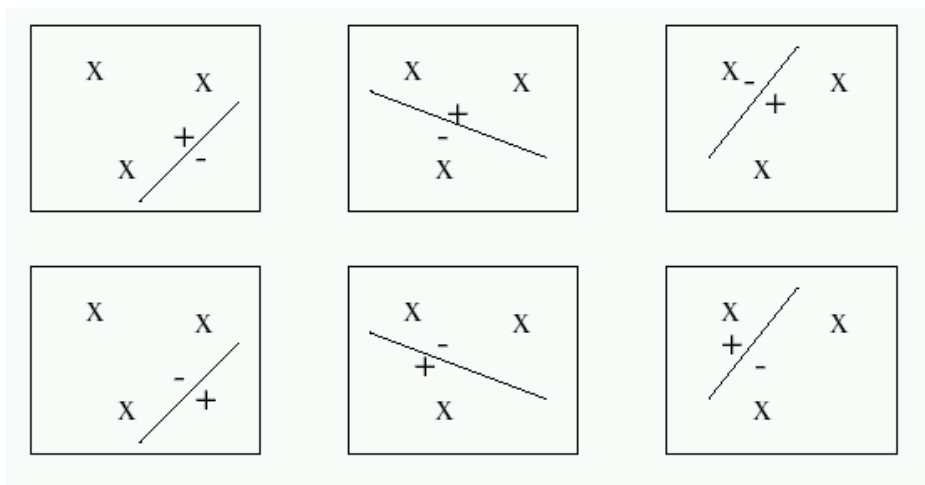
$$R(f) \leq R^E(f) + \underbrace{\sqrt{\frac{h(\ln(2n/h) + 1) - \ln(e/4)}{n}}}_{R^S(f): \text{structural risk}}$$

- Which is true w/ prob $(1-e)$, for $n \gg h$, h is the VC complexity - the discriminating power of a class of decision functions



Vapnick-Chervonenkis (VC) Complexity

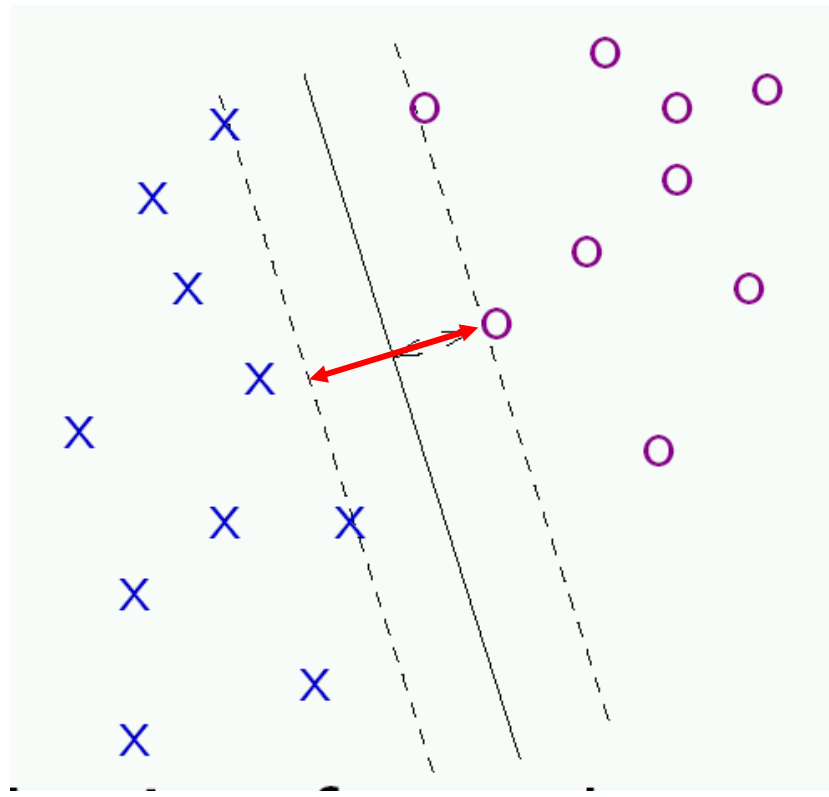
- How to characterize the decision function complexity/power ?
- **VC - Complexity:**
 - The number of points a class of function F can shatter:



Eg, Linear function
can shatter 3 samples
In 2-D space

VC Complexity for Hyperplanes

- Now consider linear decision functions: $F = \{\langle w, x \rangle + b\}$
- To minimize the structural risk is equivalent to maximizing the “gap” along the decision plane:



Support Vector Classifier

- **SVC** : a good linear decision function that maximizes the "Gap":
- A hyperplane: $\langle w, x \rangle + b = 0$
- Canonical form: w, b need to satisfy:

$$\min_i |\langle w, x_i \rangle + b| = 1$$

that is, the distance of the nearest sample to the decision surface should be the inverse of the norm of w .

Support Vector Classifier

- The data: $\{x_i, y_i\}$, where $y_i = +1$, or -1 . $I = 1..N$
- A clean-separable hyperplane:
 $y_i[\langle w, x_i \rangle + b] = 1$, for $I = 1..N$
- Min distances (gap) of $\{x_i\}$ to hyperplane:

$$\begin{aligned}d(w, b) &= \min_{x_i: s.t. y_i = -1} \frac{|\langle w, x_i \rangle + b|}{\|w\|} + \min_{x_i: s.t. y_i = +1} \frac{|\langle w, x_i \rangle + b|}{\|w\|} \\ &= \frac{1}{\|w\|} \left\{ \underbrace{\min_{x_i: s.t. y_i = -1} |\langle w, x_i \rangle + b|}_{=1} + \underbrace{\min_{x_i: s.t. y_i = +1} |\langle w, x_i \rangle + b|}_{=1} \right\} \\ &= \frac{2}{\|w\|}\end{aligned}$$

SVC-Optimization Formulation

Find hyper plane with parameters w and b , that is in canonical form, correctly separating all known points, and maximizes the gap along the decision surface:

$$\max_{w,b} \frac{2}{\|w\|},$$

s.t.

$$y_i(\langle w, x_i \rangle + b) \geq 1, \quad i = 1..N$$

How to solve this non-linear optimization (primal formulaiton) ?

Lagrangian Method

- A classical numerical method, introducing the Lagrangian to relax the constrained problem:

$$L(w, b, \lambda) = (1/2) \|w\|^2 - \lambda_i \sum_{j=1}^N [y_i (\langle w, x_i \rangle + b) - 1]$$

- The solution to the original problem is found at point where (KKT cond.):

$$\left\{ \begin{array}{l} \frac{\partial L(w, b, \lambda)}{\partial w} = 0 \\ \frac{\partial L(w, b, \lambda)}{\partial b} = 0 \\ \lambda_i \geq 0, \quad i = 1..N \\ y_i (\langle w, x_i \rangle + b) - 1 \geq 0, \quad i = 1..N \\ \lambda_i y_i (\langle w, x_i \rangle + b) - 1 = 0, \quad i = 1..N \end{array} \right.$$

Boundary conditions: 

Dual Problem

- Instead of solving the primal formulation, which is difficult, but convex, we solve the (Wolfe) dual problem:

$$\max_{\lambda} \{ \min_{w,b} [L(w, b, \lambda)] \}$$

- That is,

$$\max_{\lambda} L(\lambda), s.t.$$

$$\left\{ \begin{array}{l} \frac{\partial L}{\partial b} = 0, \Rightarrow \sum_{j=1}^N \lambda_j y_j = 0 \\ \frac{\partial L}{\partial w} = 0, \Rightarrow w = \sum_{j=1}^N \lambda_j x_j y_j \end{array} \right\}$$

Dual Problem

- The dual problem:

$$\lambda^* = \arg \max_{\lambda} \left\{ \sum_{i=1}^N \lambda_i - (1/2) \sum_{j=1}^N \sum_{k=1}^N \lambda_j \lambda_k y_j y_k \langle x_j, x_k \rangle \right\}$$

s.t.

$$\left[\begin{array}{l} \frac{\partial L}{\partial b} = 0, \Rightarrow \sum_{j=1}^N \lambda_j y_j = 0 \\ \frac{\partial L}{\partial w} = 0, \Rightarrow w = \sum_{j=1}^N \lambda_j x_j y_j \end{array} \right]$$

- A classical (sparse) QP !

Optimal Solution

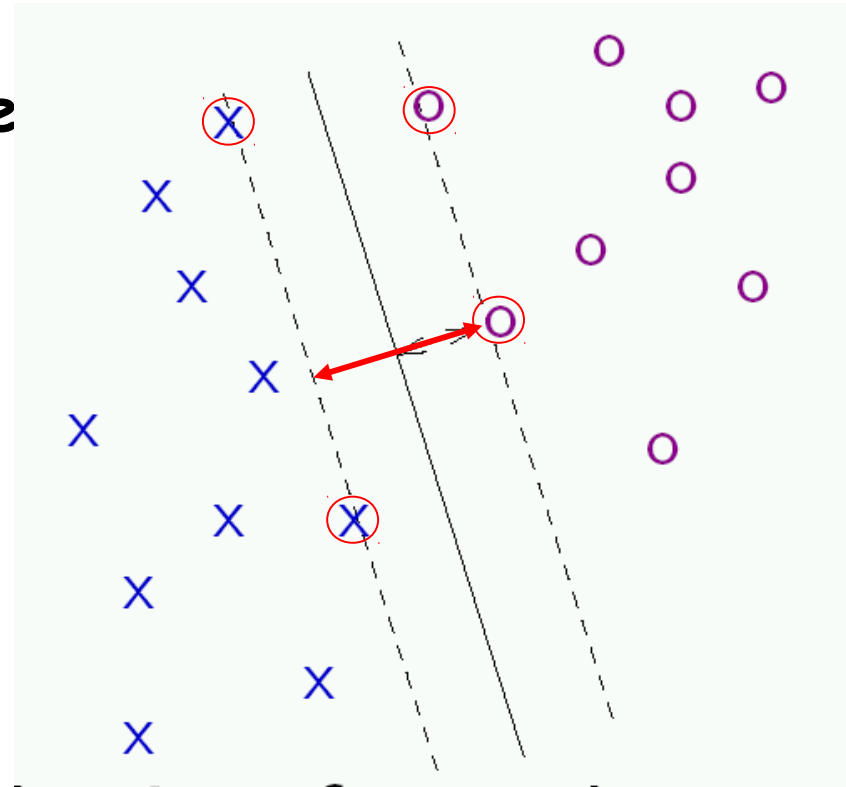
- The optimal hyperplane is give

by:

$$\left\{ \begin{array}{l} w^* = \sum_{j=1}^N \lambda_j^* y_j x_j \\ b^* = -(1/2) \langle w^*, x^+ + x^- \rangle \end{array} \right\}$$

- Support vectors:

$$\{x_k^S \mid \lambda_k > 0\}$$



Observations

- Only boundary points have non-zero Lagrangian multipliers
- The solution is sparse on Lagrangian
- Only boundary points are influencing the decision plane, interior (Lagrangian = 0) points have no influence on the decision boundary.

Compute SVM in Matlab

- **Matlab provides the SVM in the two functions:**
 - `Svmtrain`: train a SVM in `svmStruct`
 - `Svmclassify`; use the SVM to do classification
- **Run Demo:**

Okay, but....

What happens to the non-linear problem ?

Kernel

- A kernel defines a function: $K(x_1, x_2) \rightarrow \mathbb{R}^+$
- If a kernel satisfies conditions in Mercer's theorem:

$$\int K(x_1, x_2)g(x_1)g(x_2)dx_1dx_2 \geq 0, \forall g, s.t. \int g(x)^2 dx < \infty$$

- Kernel induces a mapping from X to Y :

$$y = \Phi(x) : x \in \mathbb{R}^d, y \in \mathbb{R}^q$$

- q could be inf, more detail see RKHS.

Kernel Mapping Examples

- **Linear:**

$$K(x_1, x_2) = x_1^T S x_2, S = A^T A : y = Ax$$

- **Polynomial:**

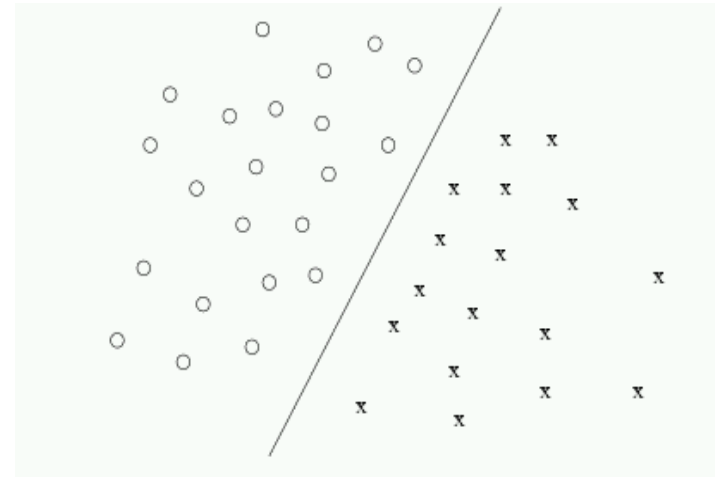
$$K(X_1, X_2) = (X_1^T X_2 + 1)^2 \Rightarrow y = \Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

- *RBF: Y is infinite dimension*

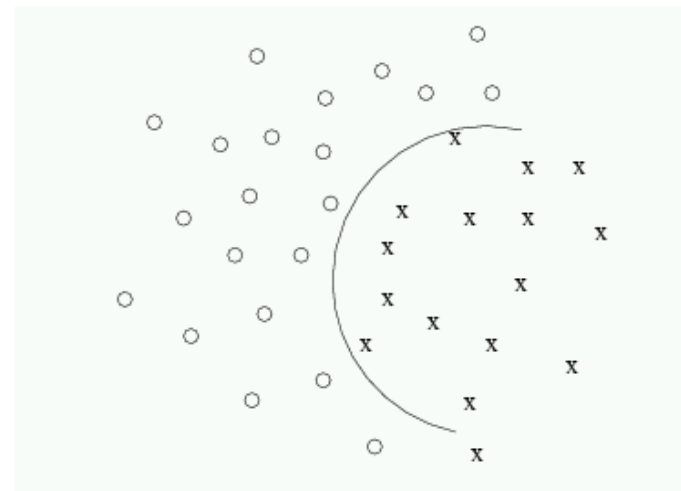
$$K(X_1, X_2) = e^{-\|X_1 - X_2\|^2 / \sigma}$$

Kernel Mapping is good !

- Gives the feature space richer structure and geometry for better classification
- Dimension of Y is typically much higher than X , it's well known that the prob of linear separability grows with feature space dimension.
- How to fit into SVC framework ?



SVC in Y



SVC in X

Support Vector (Kernel) Machine

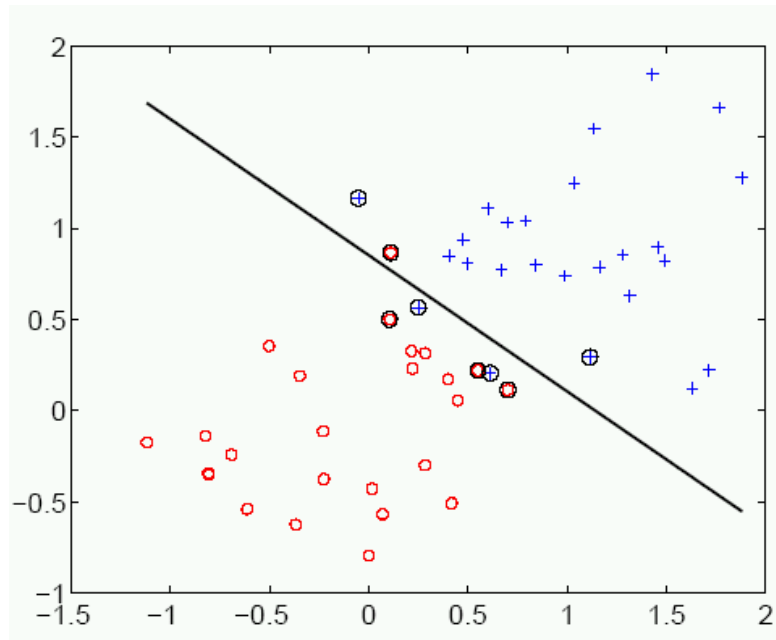
- The optimization problem depends only on inner product evaluation:

$$\lambda^* = \arg \max_{\lambda} \left\{ \sum_{i=1}^N \lambda_i - (1/2) \sum_{j=1}^N \sum_{k=1}^N \lambda_j \lambda_k y_j y_k \langle x_j, x_k \rangle \right\}$$

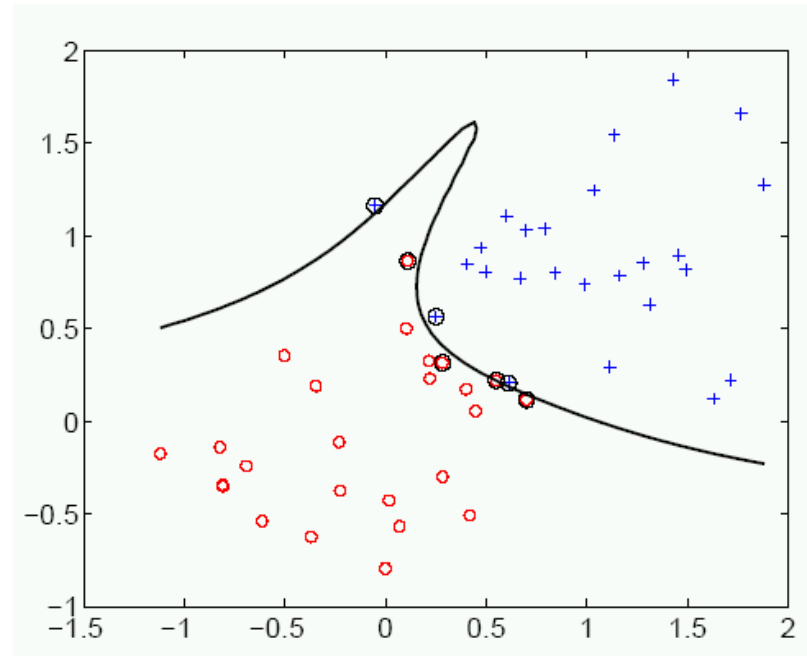
- Substitute the original space inner product with the kernel:

$$\lambda^* = \arg \max_{\lambda} \left\{ \sum_{i=1}^N \lambda_i - (1/2) \sum_{j=1}^N \sum_{k=1}^N \lambda_j \lambda_k y_j y_k K(x_j, x_k) \right\}$$

SVM Examples



Linear Kernel



8th order Polynomial

- Selection of Kernel is still a heuristic process
- Balance between performance and over fitting.

Applications:

- Face Detection (S.Z. Li & A.K. Jain: "Face recognition", Springer 2004, in press)
- Spam email filter
- Many others...

☺Have fun with SVM !

- References:

- “Statistical Learning Theory” , V. Vapnik, John Wiley & Sons, Inc, 1998.
- “A Tutorial on Support Vector Machines for Pattern Recognition”, C. Burges, in Data Mining and Knowledge Discovery, 1998.
- “Kernel Machine Based Learning for Multiple View Face Detection and Pose Estimation”, S.Z. Li et al, ICCV'01
- Many others at: www.kernel-machines.org/