

# Support Vector Machines

Praphul Chandra



# The Dot Product

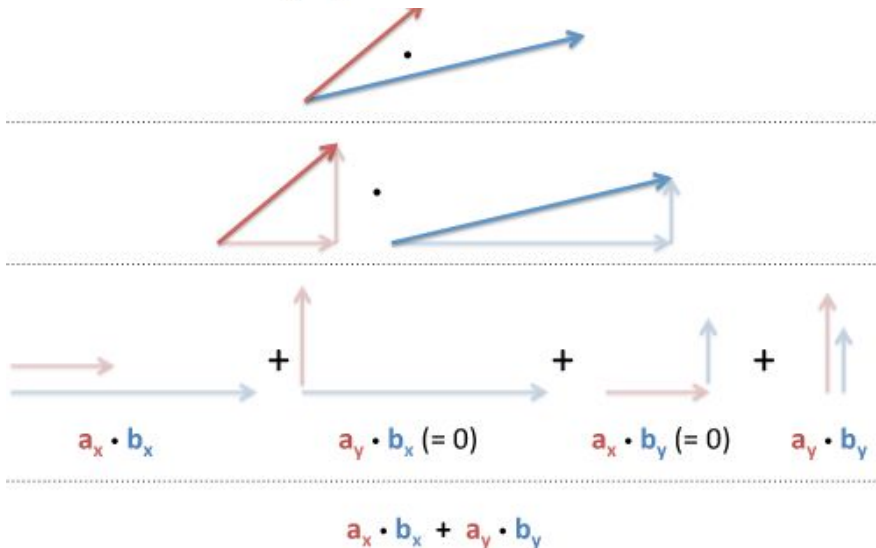
## • Inner Product

- Element wise product of two vectors
- a.k.a. scalar product

$$\begin{pmatrix} 3 \\ -2 \\ 6 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 3 \\ -5 \end{pmatrix} = 3 \times 2 + (-2) \times 3 + 6 \times (-5) = 6 - 6 - 30 = -30.$$

$$\mathbf{w} \in \mathbb{R}^p, \mathbf{x} \in \mathbb{R}^p$$

$$\mathbf{w}^T \mathbf{x} = \sum_{j=1}^p w_j x_j = \langle \mathbf{w}, \mathbf{x} \rangle$$



## • Projection Product

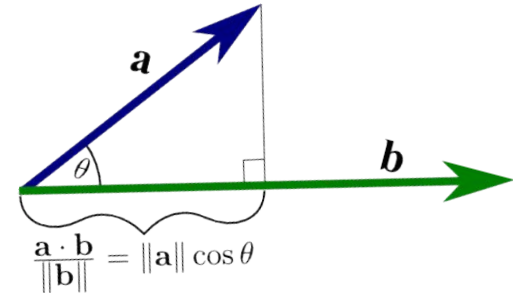
- Vector : Magnitude (Length) & Direction

$$\mathbf{w}^T \mathbf{x} = \|\mathbf{w}\| \|\mathbf{x}\| \cos \theta$$

$$\theta = 90 \Rightarrow \mathbf{w}^T \mathbf{x} = 0$$

$$\theta = 0 \Rightarrow \mathbf{w}^T \mathbf{x} = \|\mathbf{w}\| \|\mathbf{x}\|$$

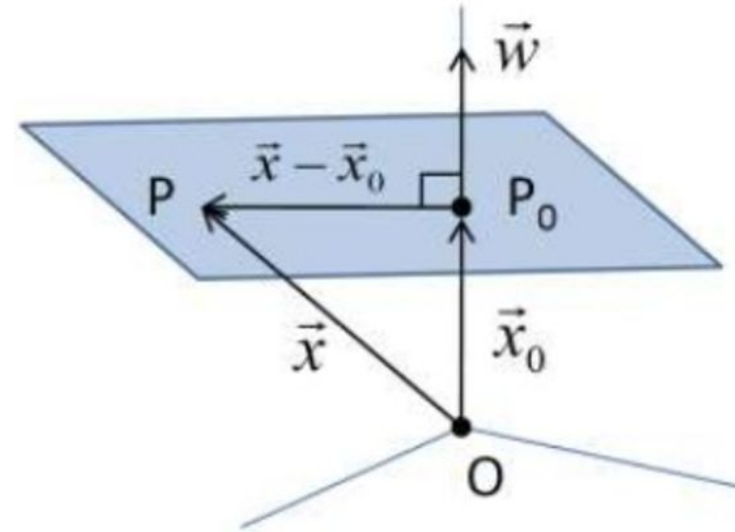
$$\mathbf{w}^T \mathbf{w} = \|\mathbf{w}\|^2$$



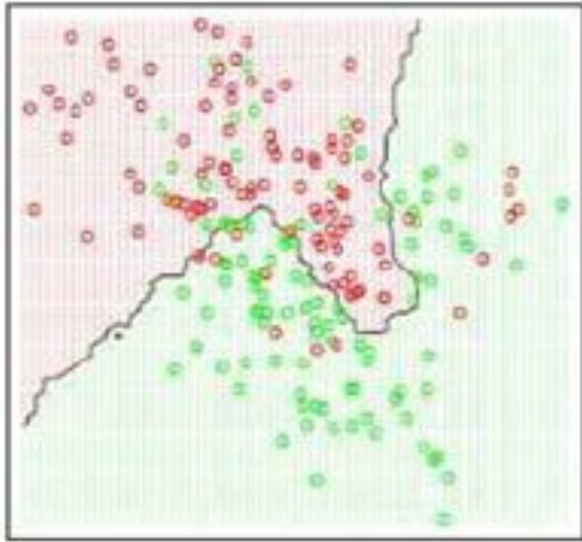
$$x_w = \|\mathbf{x}\| \cos \theta = \frac{\|\mathbf{x}\| \|\mathbf{w}\| \cos \theta}{\|\mathbf{w}\|} = \frac{\mathbf{x}^T \mathbf{w}}{\|\mathbf{w}\|} = \mathbf{x}^T \left( \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) = \mathbf{x}^T \hat{\mathbf{w}}$$

# The Dot Product and the hyperplane

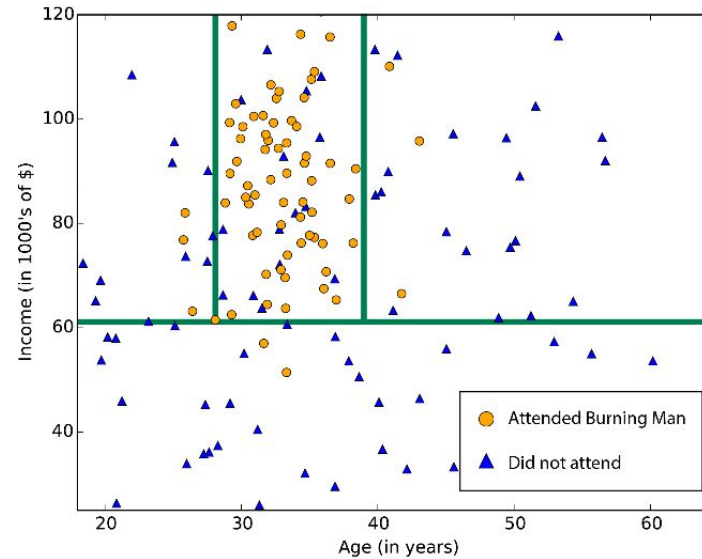
- Equation of a line
  - $ax + by = c$
  - Every point  $(x, y)$  on the line satisfies this
- Equation of a plane
  - $ax + by + cz = d$
  - Every point  $(x, y, z)$  on the plane satisfies this
- Equation of a hyper-plane
  - $w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_px_p = b$
  - $w^T x = b$
- Alternately specify a plane by
  - specifying a point and a vector perpendicular (normal) to the plane at that point
  - Let  $P$  &  $P_0$  be two points on a hyperplane.
  - Let  $x$  &  $x_0$  be two vectors supporting the hyperplane.
  - Consider the vector  $w$  which is orthogonal to the hyperplane at  $x_0$
  - $\rightarrow (x - x_0)$  must lie on the hyperplane  $\rightarrow w$  must be orthogonal to  $(x - x_0)$
  - $\rightarrow w^T(x - x_0) = 0$
  - $\rightarrow w^T x = -w^T x_0$
  - $\rightarrow w^T x = b$



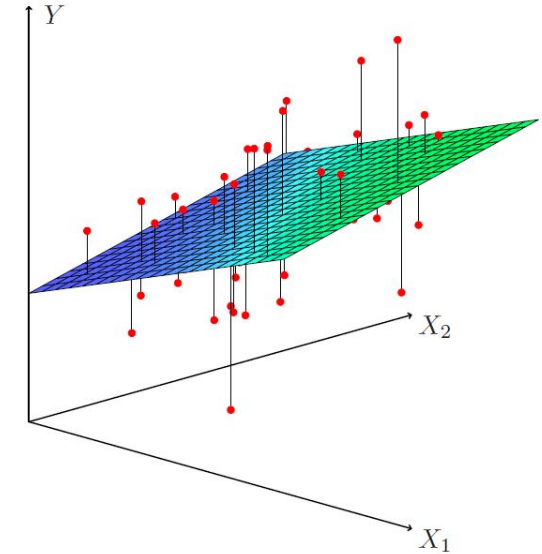
# Classification vs. Regression ( $p=2$ )



knn



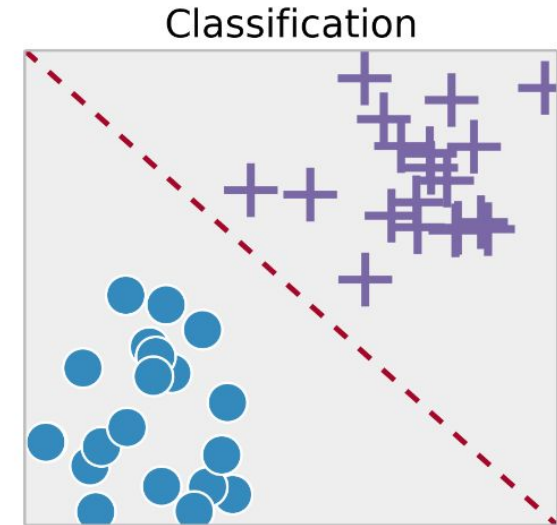
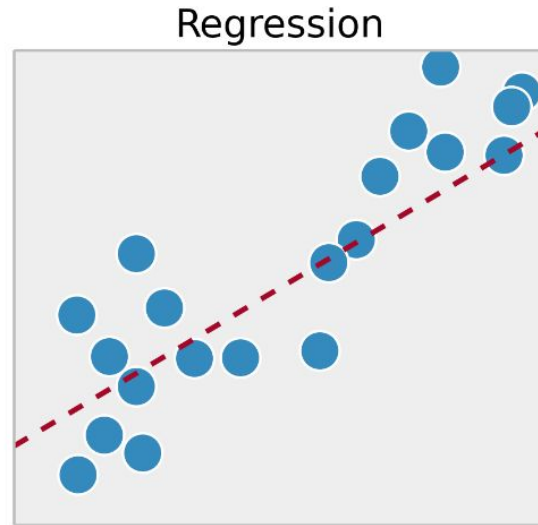
Decision Trees



Linear Regression

- The lines / curves play a different role
  - Regression : approximate the mean value of the dependent variable given independent variables
  - Classification : separating boundary
- Different Algorithms / Model Families result in different curves / shapes
  - Model-Free
  - Locally Linear
  - Globally Linear

# “Linear” Classification?



- Linear Regression

$$\begin{aligned} y_i, b \in \mathbb{R} \quad , \quad \mathbf{x}_i, \mathbf{w} \in \mathbb{R}^p \\ &= b + w_1 x_{i1} + w_2 x_{i2} + \dots + w_p x_{ip} + \epsilon_i \\ &= b + \sum_{j=1}^p w_j x_{ij} + \epsilon_i \\ &= b + \mathbf{w}^T \mathbf{x}_i + \epsilon_i \end{aligned}$$

- Linear Classification ( $y \in \{-1, 1\}^n$ )

- Linear Separating Hyperplane

$$\begin{aligned} y_i \in \{-1, 1\}, b \in \mathbb{R} \quad , \quad \mathbf{x}_i, \mathbf{w} \in \mathbb{R}^p \\ y_i = \text{sign}(b + \mathbf{w}^T \mathbf{x}_i) \end{aligned}$$

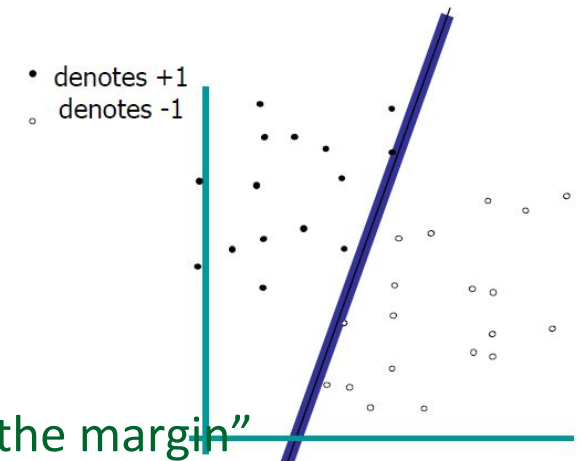
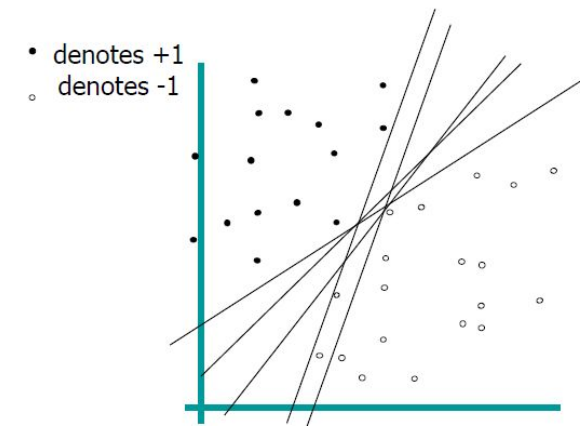


# Maximum Margin Classifier

- Many possible linear separating hyperplanes
  - Which one to choose?
- Idea
  - Margin of a classifier
  - Choose the linear classifier with the largest margin
  - Create the thickest hyper-slab which separates the two classes
- Optimization Criteria
  - Maximize the margin
  - subject to “training observations should lie on the correct side of the margin”
  - and “normalize coefficients” *(so that margins from hyperplanes are comparable)*

$$\min_{\mathbf{w}} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - b - \mathbf{w}^T \mathbf{x}_i)^2$$

Linear Regression



$$\begin{aligned} & : \max_{\mathbf{w}} M \\ \text{s.t. } & y_i(b + \mathbf{w}^T \mathbf{x}_i) > M \quad \forall i, \\ \text{and } & \|\mathbf{w}\| = \sum_{j=1}^p w_j^2 = 1 \end{aligned}$$

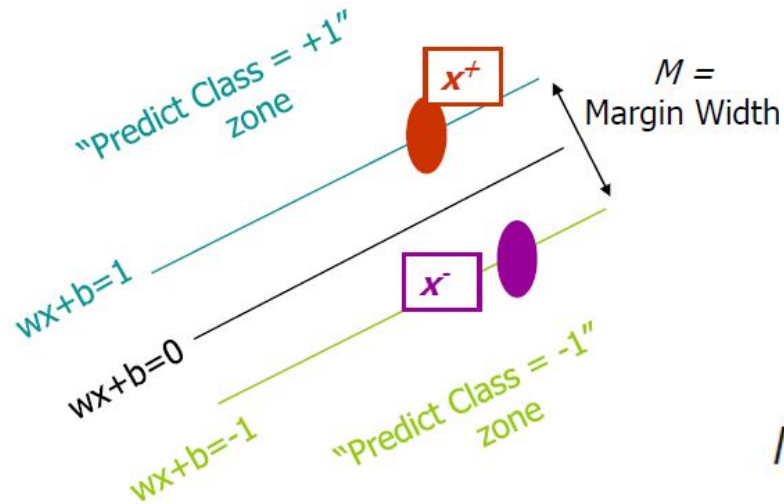
Maximum Margin Classification





# Maximum Margin Classifier : Optimization revisited

- Setup : ( $y_i \in \{-1, 1\}$ )
  - $x^+$ : Nearest positive class training observation closest to the separating hyperplane
  - $x^-$ : Nearest negative class training observation closest to the separating hyperplane



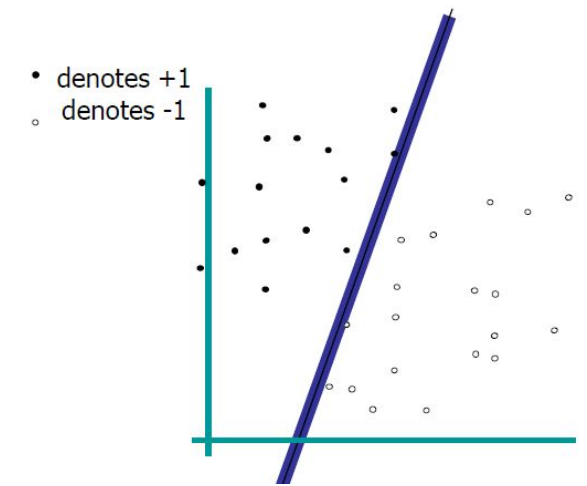
$$b + \mathbf{w}^T \mathbf{x}^+ = +1$$

$$b + \mathbf{w}^T \mathbf{x}^- = -1$$

$$\mathbf{w}^T (\mathbf{x}^+ - \mathbf{x}^-) = 2$$

$$M = \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = \frac{2}{\|\mathbf{w}\|}$$

- Margin
  - Projection of  $(\mathbf{x}^+ - \mathbf{x}^-)$  onto the unit vector normal to the separating hyperplane
- Equivalent Optimization Problem
  - Minimize hyperplane parameters ~ Maximize Margin



$$: \max_{\mathbf{w}} M$$

$$\text{s.t. } y_i(b + \mathbf{w}^T \mathbf{x}_i) > M \quad \forall i,$$

$$\text{and } \|\mathbf{w}\| = \sum_{j=1}^p w_j^2 = 1$$

$$: \min \mathbf{w}^T \mathbf{w}$$

$$\text{s.t. } y_i(b + \mathbf{w}^T \mathbf{x}_i) > 0 \quad \forall i$$



# Support Vector Classifier

$$y_i \in \{-1, 1\}, b \in \mathbb{R} \quad , \quad \mathbf{x}_i, \mathbf{w} \in \mathbb{R}^p$$

$$y_i = \text{sign}(b + \mathbf{w}^T \mathbf{x}_i)$$

- Maximum Margin Classifier

- A change of one observation result in a significant change in the hyperplane
- Maximum Margin classifier has high variance

- Idea

- Add some slack
- Hyper-parameter : Total slack allowed

- Support Vector (a.k.a. Soft Margin) Classifier

- Margin is “soft” i.e. allows some training observations to lie on the wrong side of the margin / hyperplane

$$\begin{aligned} & : \max_{\mathbf{w}} M \\ \text{s.t.} \quad & y_i(b + \mathbf{w}^T \mathbf{x}_i) > M \quad \forall i, \\ \text{and} \quad & \|\mathbf{w}\| = \sum_{j=1}^p w_j^2 = 1 \end{aligned}$$

$$\begin{aligned} & : \max_{\mathbf{w}, \epsilon} M \\ \text{s.t.} \quad & y_i(b + \mathbf{w}^T \mathbf{x}_i) > M(1 - \epsilon_i) \quad \forall i, \\ \text{and} \quad & \sum_{j=1}^n \epsilon_i \leq C \\ \text{and} \quad & \|\mathbf{w}\| = 1 \end{aligned}$$





# Maximum Margin Classifier : Optimization revisited : again

- Yet another (optimization) equivalence
  - Primal - Dual
  - Solving the Dual involves computing only dot products among all training points
  - $\alpha_i \neq 0 \rightarrow \mathbf{x}_i$  is a **support vector**
- The classification function depends only on the dot product of the test observation with **support vectors**.
- Intuition
  - The hyperplane is “supported” by the training data observations which are closest to it
  - The margin depends on how close (near) the support vectors from two classes are to each other

$$\begin{array}{l|l}
 : \max_{\mathbf{w}} M & : \min \mathbf{w}^T \mathbf{w} \\
 \text{s.t. } y_i(b + \mathbf{w}^T \mathbf{x}_i) > M \quad \forall i, & \text{s.t. } y_i(b + \mathbf{w}^T \mathbf{x}_i) > 0 \quad \forall i \\
 \text{and } \|\mathbf{w}\| = \sum_{i=1}^p w_j^2 = 1 &
 \end{array}$$

$$: \max_{\alpha} \left( \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

$$\text{s.t. } \sum_{i=1}^n \alpha_i y_i = 0 \quad \forall i$$

$$\text{and } \alpha_i \geq 0 \quad \forall i$$

Solution to the dual :  $\alpha \in \mathbb{R}^n$  means

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

$$b = y_k - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_k \quad \text{for any } k$$

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

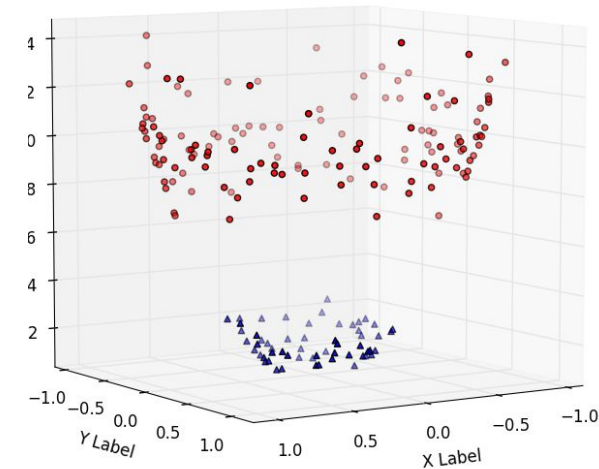
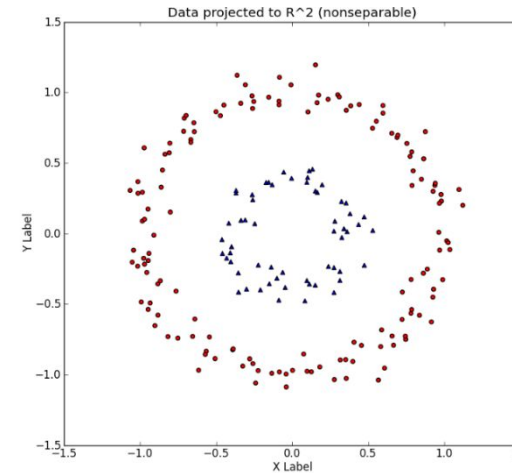
# Support Vector Machine

- Motivation

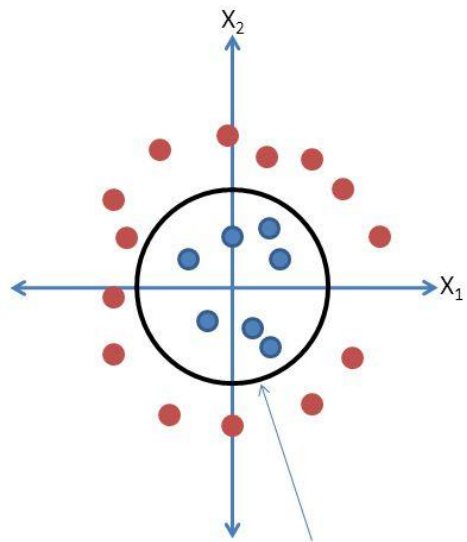
- “Linear” separating hyperplane always possible?
- Even with the slack?

- Intuition

- Data which is separable with a linear hyperplane may not be linearly separable in a **lower** dimension sub-space
- Data which is not separable with a linear hyperplane may be linearly separable in a **higher** dimension space
- Can we increase the dimensionality of the data and then linearly separate it?
- How?
- At what cost?



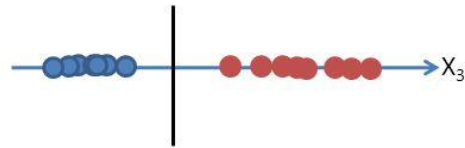
# Achieving Non-Linearity using Dimensionality Expansion



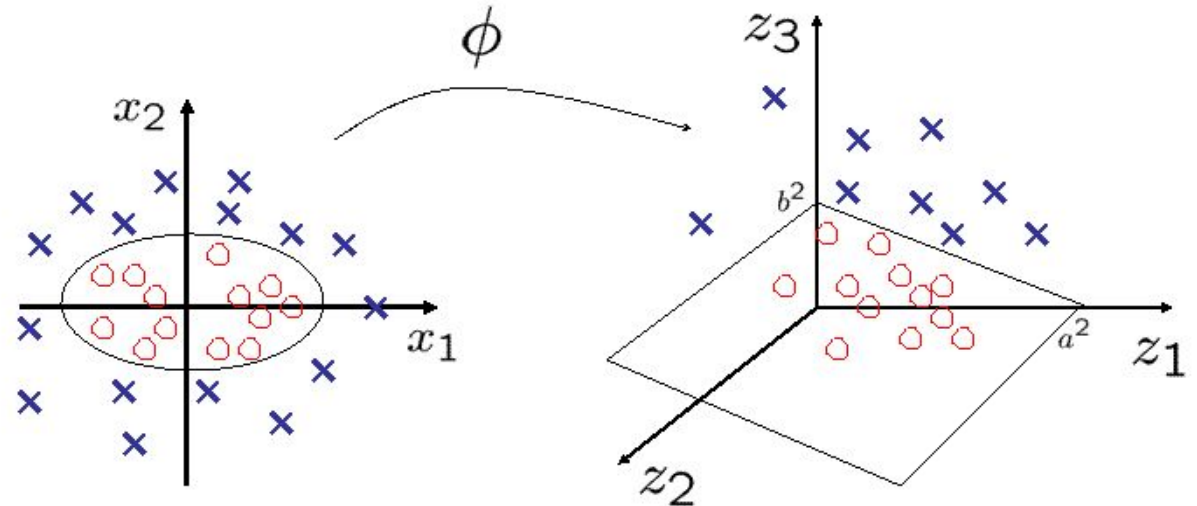
In the original feature space, the linear separator looks like a circle.

The Kernel Trick is to add a new input variable that is computed from the existing ones.

$$\text{Let } X_3 = \sqrt{X_1^2 + X_2^2}$$



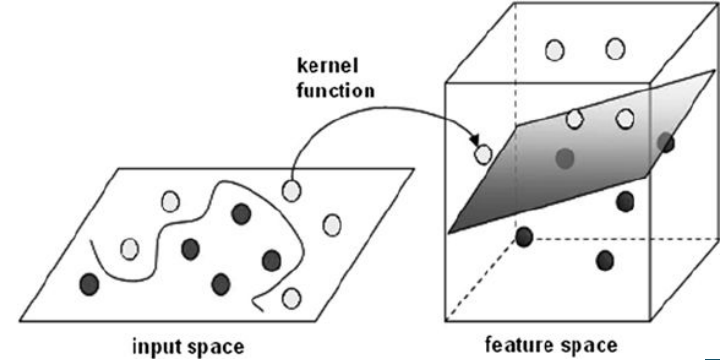
Now there's a linear separator!



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

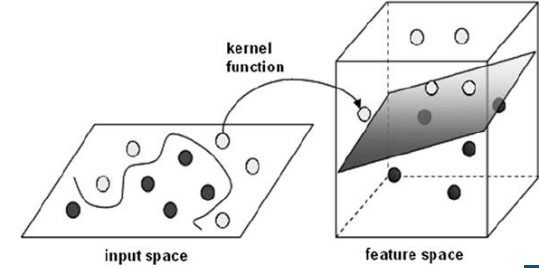
# Dimension Expansion to tackle Non-Linearity



- The curse of dimensionality
  - Exponential increase in volume associated with adding extra dim (e.g. Impact on knn)
  - With a fixed number of training samples, predictive power reduces as the dimensionality increases (Hughes effect)
  - Computational Complexity
  - Dimensionality reduction techniques: Principal Component Analysis
- The boon of dimensionality
  - Data which is not linearly separable in  $m$ -dimensions may be separable in  $m+1$  dimensions
  - Used beyond SVM : Polynomial regression, Basis Transformation
- The beauty of SVM
  - Achieve benefits of dimensionality expansion (linear separability) without paying the computational cost
  - The solution to the optimization problem requires us to calculate ONLY the **dot product** among the support vectors
  - Define a kernel function corresponding to the generalization of the **dot product** (Reduced computational cost)
  - Define a kernel function which captures the proximity of the support vectors (Further Reduced computational cost)



# The Kernel Trick : Dot Product Magic



$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

$\left. \begin{matrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \end{matrix} \right\}$  Constant Term  
 $\left. \begin{matrix} \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \end{matrix} \right\}$  Linear Terms  
 $\left. \begin{matrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \end{matrix} \right\}$  Pure Quadratic Terms  
 $\left. \begin{matrix} \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{matrix} \right\}$  Quadratic Cross-Terms

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

Computational Cost  
 $O(m^2)$  for quadratic expansion  
 $O(m^d)$  in general

Define  $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \bullet \mathbf{b} + 1)^2$

$$\begin{aligned}
 &= (\mathbf{a} \bullet \mathbf{b})^2 + 2\mathbf{a} \bullet \mathbf{b} + 1 \\
 &= \left( \sum_{i=1}^m a_i b_i \right)^2 + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m \sum_{j=1}^m a_i b_i a_j b_j + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m (a_i b_i)^2 + 2 \sum_{i=1}^m \sum_{j=i+1}^m a_i b_i a_j b_j + 2 \sum_{i=1}^m a_i b_i + 1
 \end{aligned}$$

Computational Cost  
 $O(m)$

$$\begin{aligned}
 K_{\text{linear}}(\mathbf{a}, \mathbf{b}) &= \mathbf{a}^T \mathbf{b} \\
 \text{polynomial}(\mathbf{a}, \mathbf{b}) &= (\mathbf{a}^T \mathbf{b} + 1)^d \\
 K_{\text{rbf}}(\mathbf{a}, \mathbf{b}) &= \exp \left( -\frac{\|\mathbf{a} - \mathbf{b}\|^2}{2\sigma^2} \right) \\
 K_{\text{tanh}}(\mathbf{a}, \mathbf{b}) &= \tanh(\kappa \mathbf{a}^T \mathbf{b} - \delta)
 \end{aligned}$$





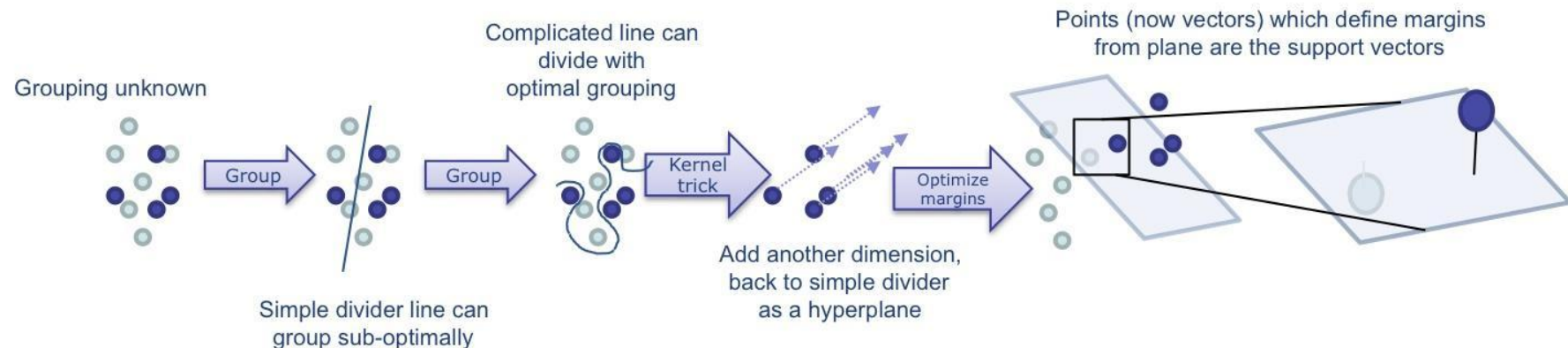
# Support Vector Machine : Summary

- Ideas

- 1) Linear Separating Hyperplane (Add slack to reduce variance)
- 2) Achieve benefits of dimensionality expansion (linear separability) with low computational cost
  - Optimization requires us to calculate ONLY the **dot product** among the support vectors
  - Define a kernel function ~ the generalization of the **dot product** (*Reduced computational cost*)
  - Define a kernel function which captures the proximity of the support vectors (*Further Reduced computational cost*)

- Hyperparameters

- Choice of Kernel : Linear / Polynomial / Radial Basis Function
- Total Slack (softness) allowed



# 3 important extensions from Linear Regression

- Local Regression
  - Fuse nearest-neighbor with linear regression
  - Given a test point, build a regression model with k nearest neighbors as training data
- Penalized Regression
  - Guard against overfitting
  - Minimize the “regularized” sum of least squares
- Basis Expansion
  - Cope with non-linearity
  - Explicit





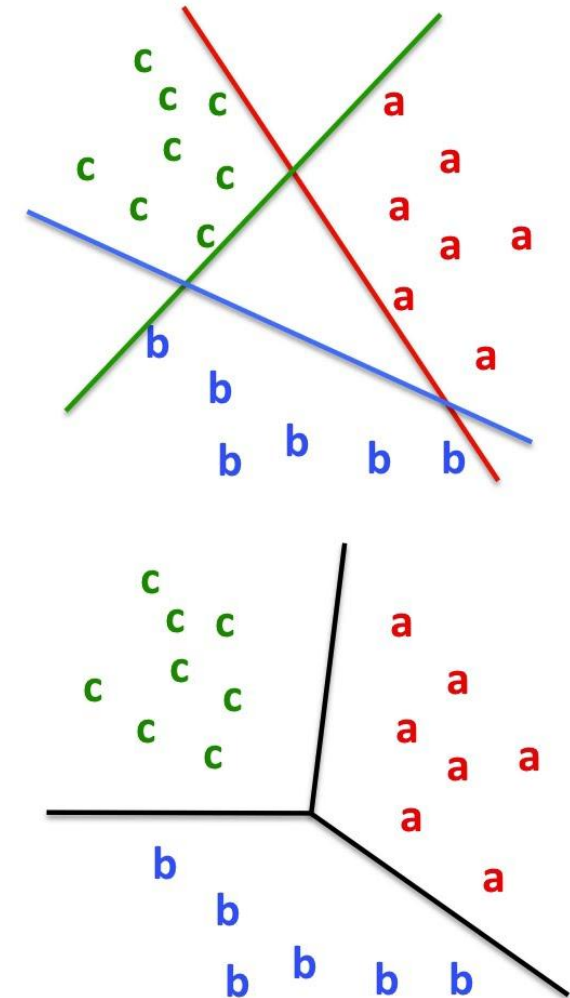
# Using SVMs

- Tuning SVM
  - Feature Engineering: Normalization, Scaling,
  - Hyperparameter: Use Cross Validation to find the best kernel family and kernel parameters
- Advantages
  - Flexible : different kernels try different “non-linear” boundaries (in the native feature space)
  - Exploits sparseness : use the support vectors only for determining the separating hyperplane
  - Can handle large feature spaces efficiently (computational complexity does not depend on  $p$ )
  - Good theoretical guarantees (Maximum margin generalizes better, Convex optimization guaranteed to converge)
- Limitations
  - Sensitive to noise and outliers (Increasing the margin may reduce the accuracy)
  - Doesn't provide a posterior probability
  - Messy Multi-labelled classification ( $m$ -classes)
    - Train  $m$  1-vs-Rest Binary classifiers (But this results in class imbalance – May require fine tuning of cost function)
    - Train Binary classifiers for  $m(m-1)/2$  pairs of classes & classify based on which class receives highest votes (More computation)



# Multi Class Classification as Binary Classification

- One vs. All
  - Train  $m$  Binary classifiers : One classifier per class
  - Base classifiers to produce a real-valued confidence score for its decision (SVM?)
  - a.k.a. One vs. Rest
  - Gotcha: May result in class imbalance
- One vs. One
  - Train  $m(m-1)/2$  binary classifiers : One classifier per pair of classes
  - Classify a new sample based on which class receives highest votes
  - Gotcha: More computation!



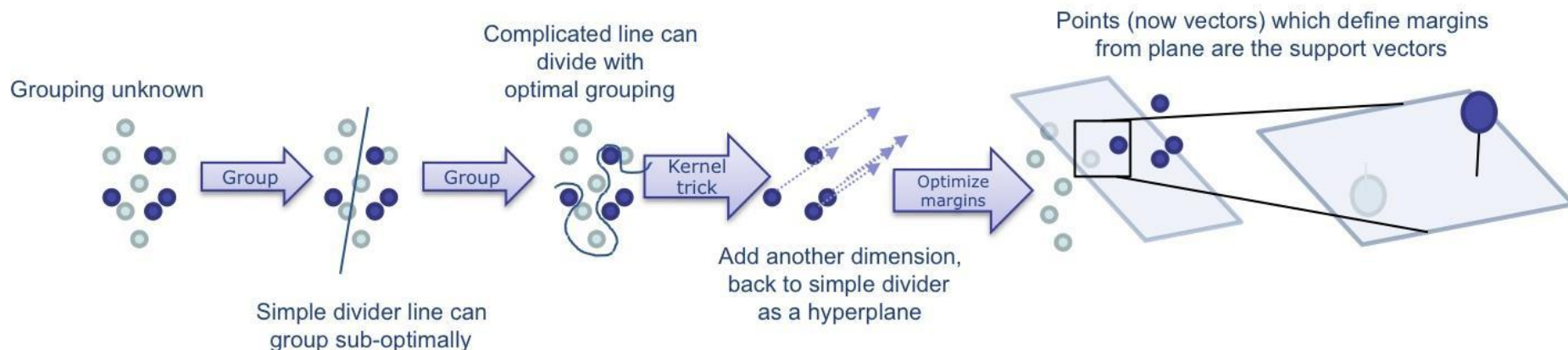
# Support Vector Machine : Summary

- Ideas

- 1) Linear Separating Hyperplane
- 2) Add slack to reduce variance
- 3) Achieve benefits of dimensionality expansion (linear separability) without paying the computational cost

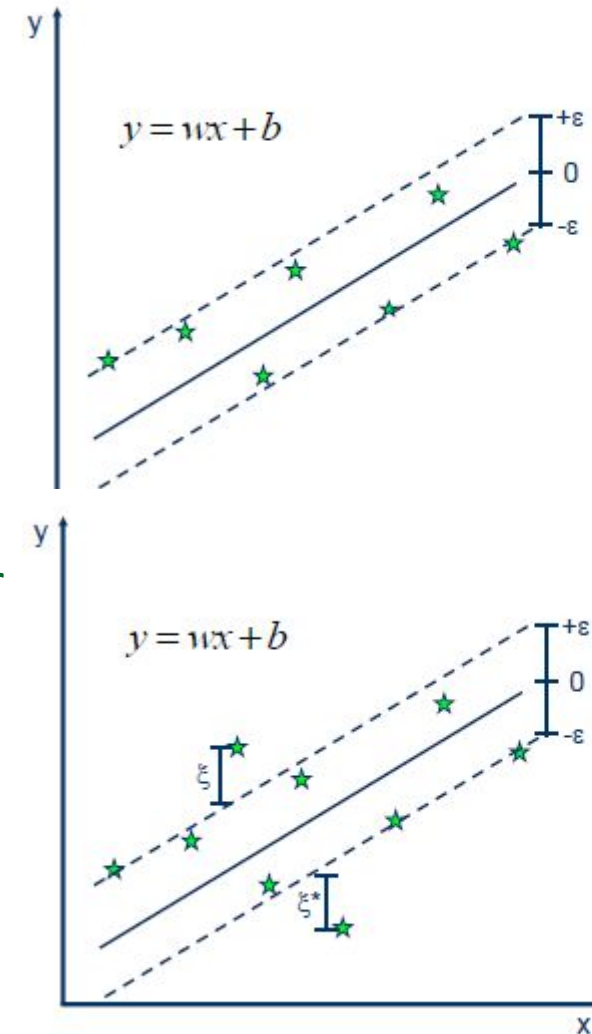
- The solution to the optimization problem requires us to calculate **ONLY** the **dot product** among the support vectors
- Define a kernel function corresponding to the generalization of the **dot product** (*Reduced computational cost*)
- Define a kernel function which captures the proximity of the support vectors (*Further Reduced computational cost*)

- Hy



# Before we finish ... Support Vector Regression

- Regression extension
  - Modify the optimization problem
  - Want the hyperplane close to the “support” vectors
  - Reinterpret Slack
- Exploit the kernel trick
  - Linearity in high dimensions  $\rightarrow$  non-linearity in lower dimensions
  - Without the computational cost



- Solution:
$$\min \frac{1}{2} \|w\|^2$$
- Constraints:
$$y_i - wx_i - b \leq \epsilon$$
$$wx_i + b - y_i \leq \epsilon$$
- Minimize:
$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*)$$
- Constraints:
$$y_i - wx_i - b \leq \epsilon + \xi_i$$
$$wx_i + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

# Q?

Praphul Chandra



## In the lab today

- KNN & SVD for Recommendation Engine (User-Movie Example)
- KNN (Regression)
- Clustering (KMeans)

