Inspire…Educate…Transform.

Support Vector Machines

Dr. Manoj Chinnakotla
Senior Applied Scientist, Microsoft
Adjunct Professor, IIIT Hyderabad

Quick Recap

- Supervised Learning
  - Given training data with true labels, learn a model to predict label
  - So far, we have learnt about
    - Decision Trees
    - Neural Networks
    - Logistic Regression

Web Document Classification

- Very large number of features (words)
- Relatively low number of labelled training samples available
- Highly sparse data
- In such situations, employing Boosted Decision Trees may not be a good idea
  – Usually works well until around 4K features
- In this talk, we will focus on SVMs
  – Known to perform very well even with very low training data
  – Training scales well with large number of features!

Outline of the Talk

- Maximum Margin Classifiers
- Basic Formulation of Linear SVMs
- Dealing with Noisy Data
- Non-linear Decision Boundaries
- Practical Advice on SVMs

CSE 7306c

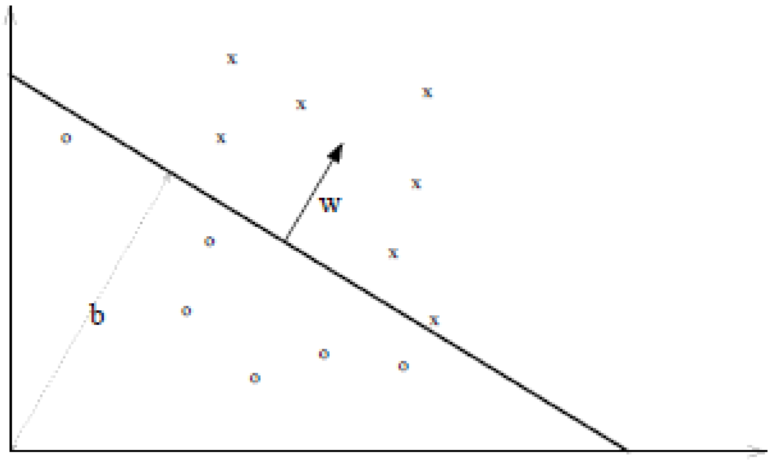
Linear Classifiers

Inner product between vectors

$$\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$$

Hyperplane:

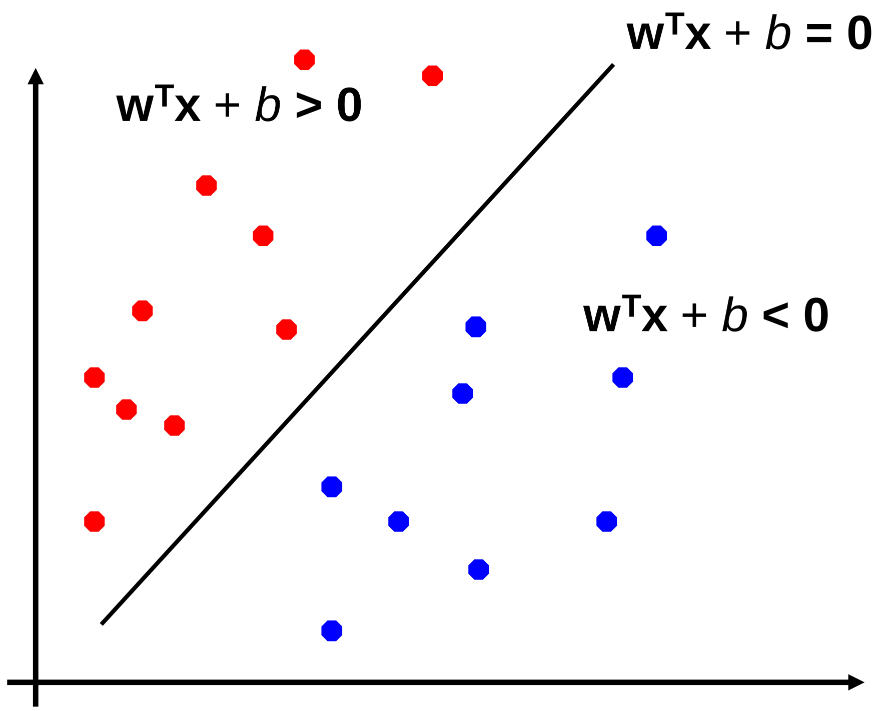
$$\langle w, x \rangle + b = 0$$



CSE 7306c

Linear Classifiers

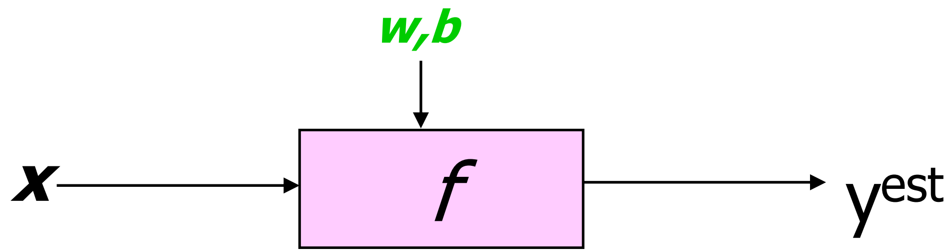
- Binary classification can be viewed as the task of separating classes in feature space:



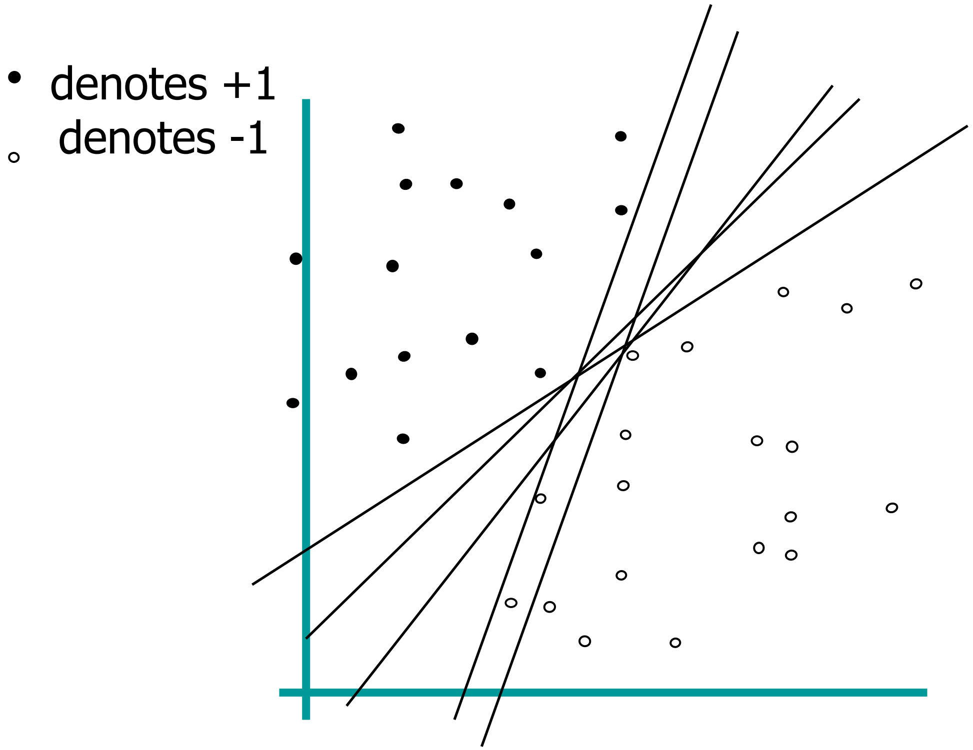
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

CSE 7306c

Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} - b)$$

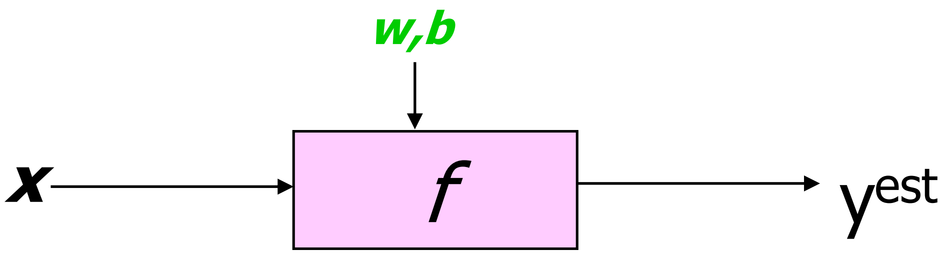


How would you classify this data?

Any of these would be fine..

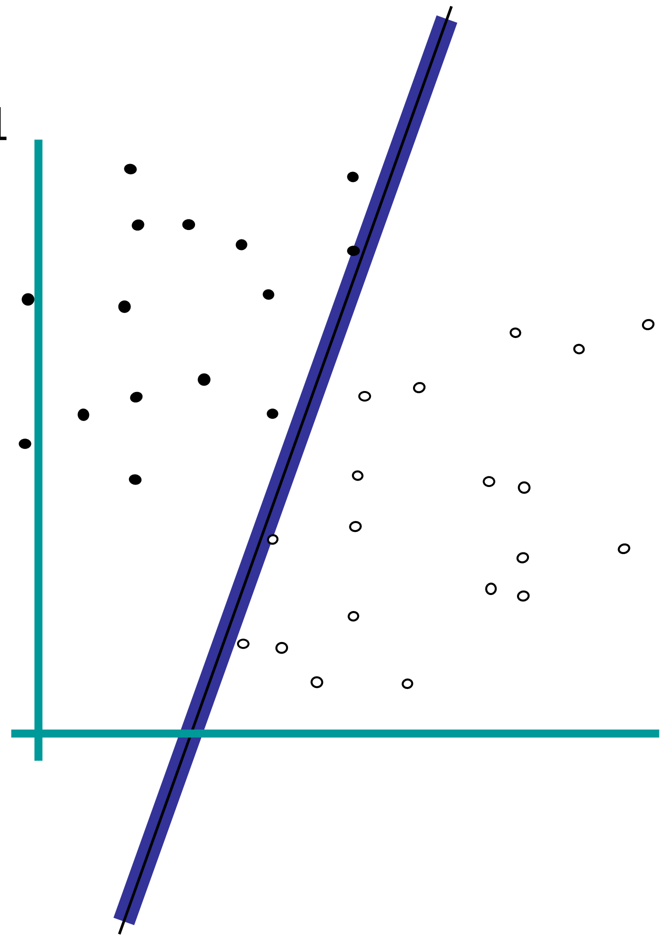
..but which is best?

Margin of a Classifier



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

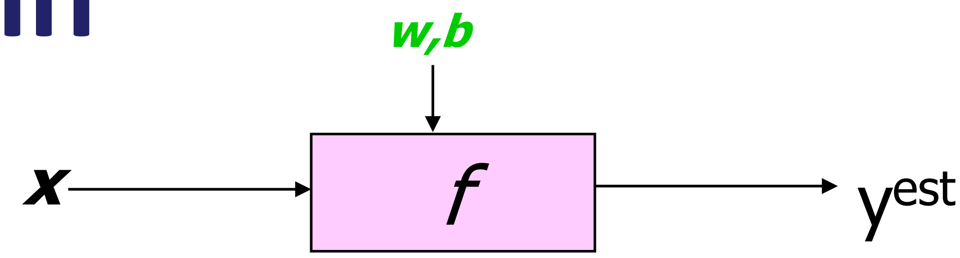
- • denotes +1
- ° denotes -1



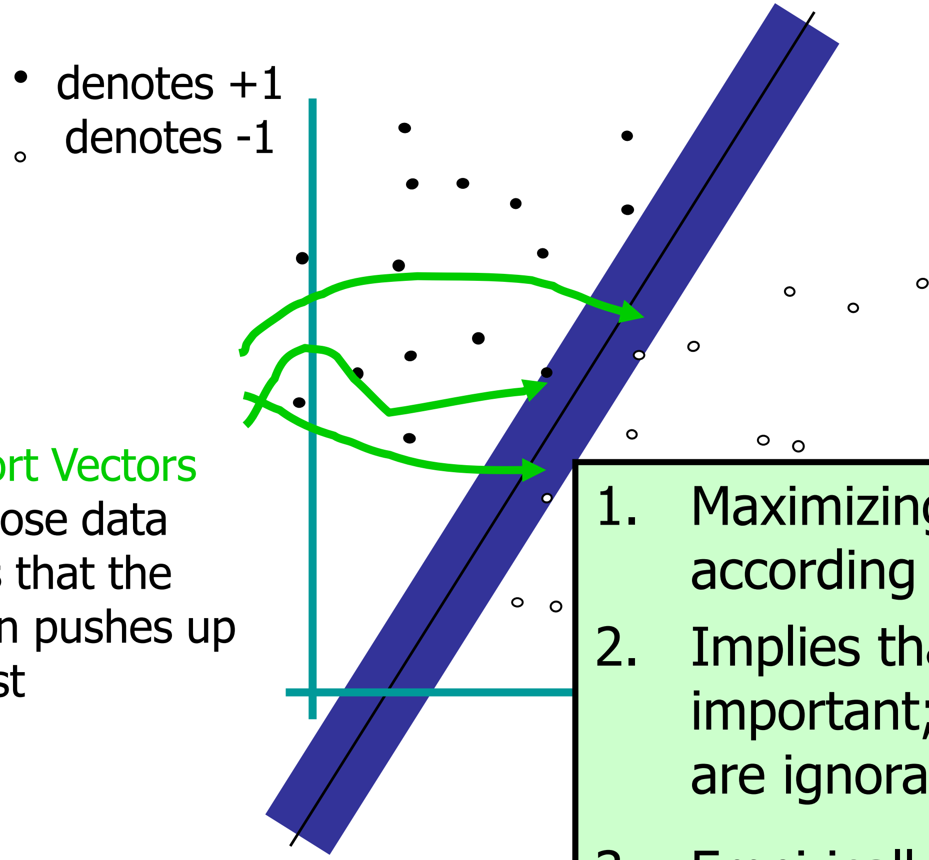
Margin of a Linear Classifier

Width that the boundary could be increased by before hitting a data point.

Maximum Margin



$$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w}\cdot\mathbf{x} - b)$$



Support Vectors are those data points that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.

kind of SVM –

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very well.

Computing the Margin Width

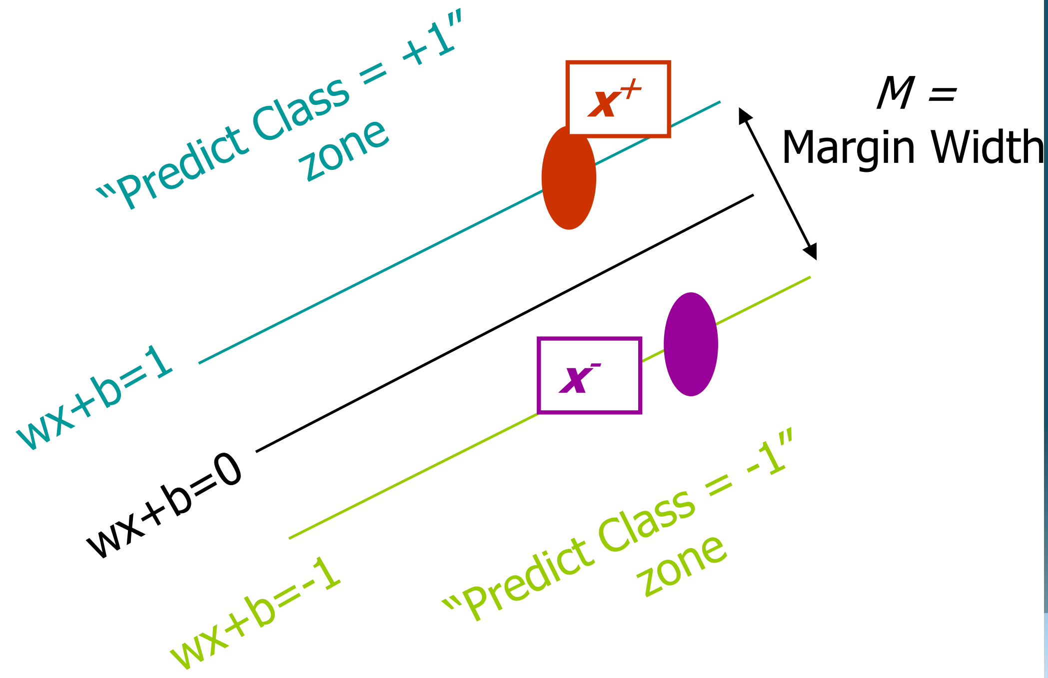
- ***w*** is normal to the separating hyperplane
  - For any two points u, v on the plane
    w .(u – v) = 0

- *M = Projection of (**x**⁺ - **x**⁻) onto unit vector normal to separating plane*

$$M = \frac{w}{||w||}.(x^{+} - x^{-})$$

$$M = \frac{2}{||w||}$$

$$M = \frac{2}{\sqrt{w.w}}$$



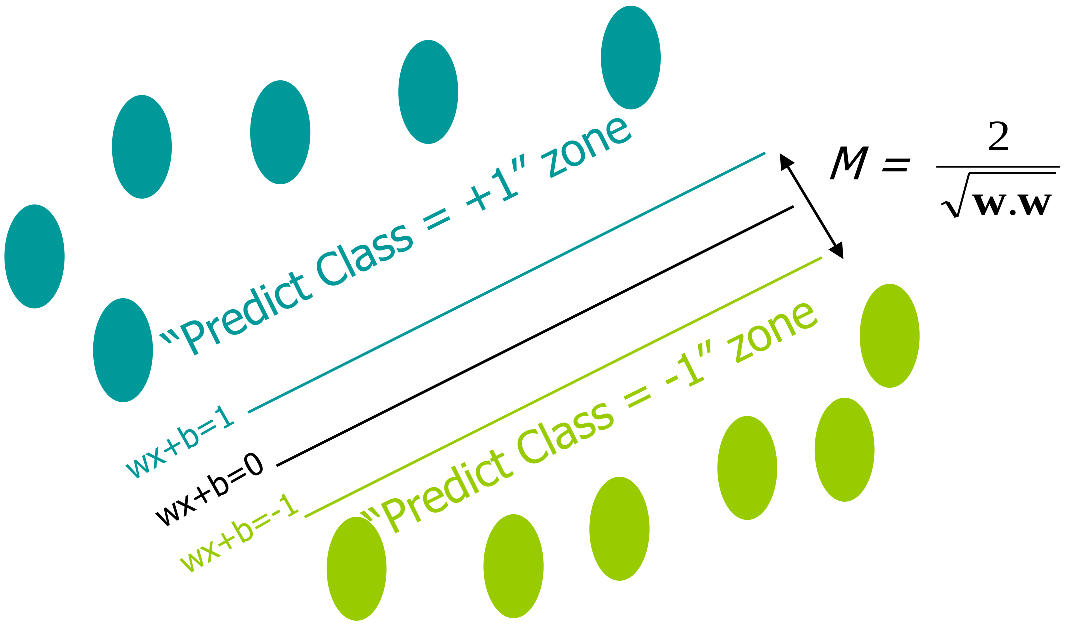
We know that

$\boldsymbol{w} \,.\, \boldsymbol{x}^{+} + b = +1$

$\boldsymbol{w} \,.\, \boldsymbol{x}^{-} + b = -1$

$\boldsymbol{w} \,.\, (\boldsymbol{x}^{+} - \boldsymbol{x}^{-}) = 2$

Learning the Maximum Margin Classifier



Given guess of ***w*** , *b* we can

- Compute whether all data points are in the correct half-planes
- Compute the margin width

Assume *R* datapoints, each $(\mathbf{x}_k, y_k)$ where $y_k = +/- 1$

What should our optimization criterion be?

Minimize ***w.w***

How many constraints will we have?

R

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b >= 1 \text{ if } y_k = 1$

$\mathbf{w} \cdot \mathbf{x}_k + b <= -1 \text{ if } y_k = -1$

Formulation of Linear SVM

Minimize **w.w** — **Quadratic Objective Function**

Subject to — **Linear Constraints**

$$\boldsymbol{w} \cdot \boldsymbol{x}_k + b >= 1 \text{ if } y_k = 1$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_k + b <= -1 \text{ if } y_k = -1$$

Solving the Optimization Problem

Primal Problem

Find **w** and b such that

$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, i=1..n : $y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Dual Problem

Packages are available for efficiently solving the above Quadratic Program (QP) to find the optimal w, b values

Find $\alpha_1 \ldots \alpha_n$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- Given a solution $\alpha_1 \dots \alpha_n$ to the dual problem, solution to the primal is:

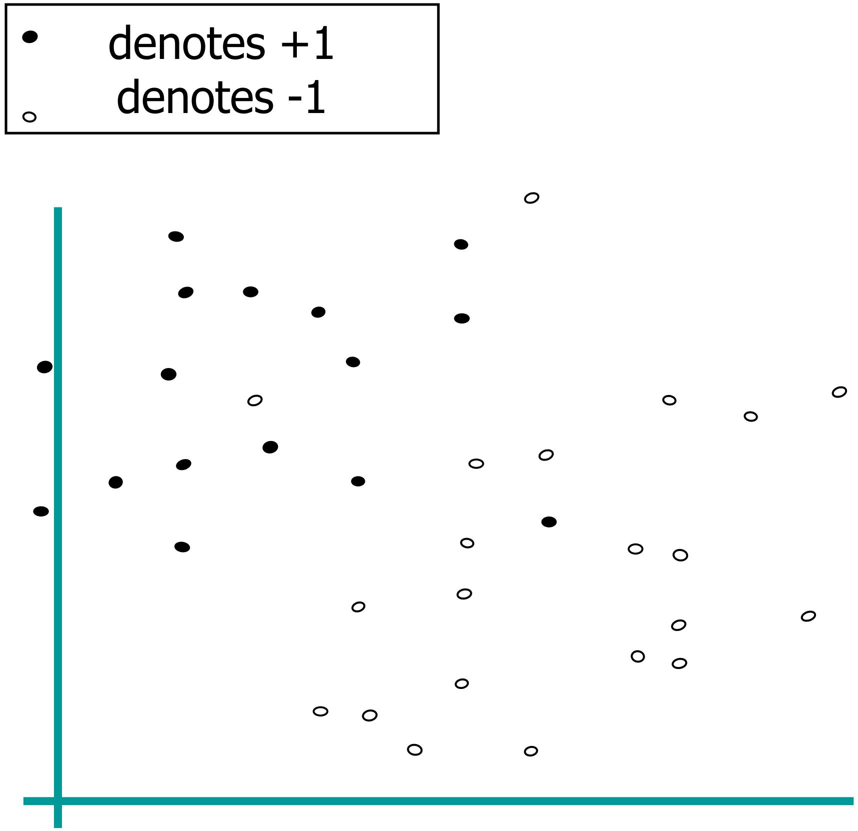
$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function is (note that we don't need **w** explicitly):

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x}_i$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$ between all training points.

Welcome to the real world, Neo! ☺



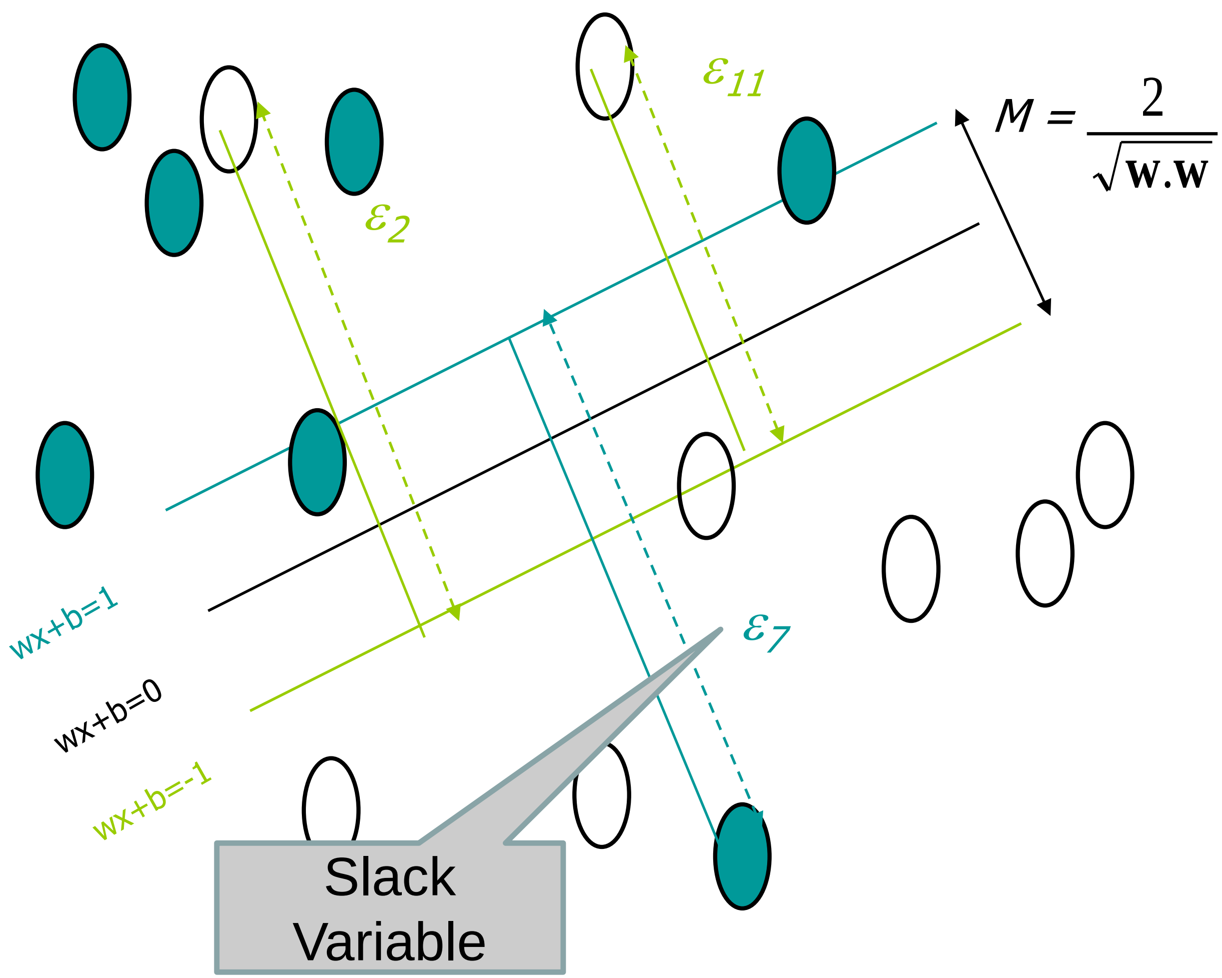
Can we classify this data with the previous formulation?

We can't!

In real world data points are

- Often not linearly separable
- Prone to noise and outliers

Fixing the Formulation to handle Noise



Relaxing Constraints

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k$ if $y_k = 1$

$\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k$ if y_k

Slack Constraints

$\varepsilon_k \geq 0$ for all k

Fixing Objective

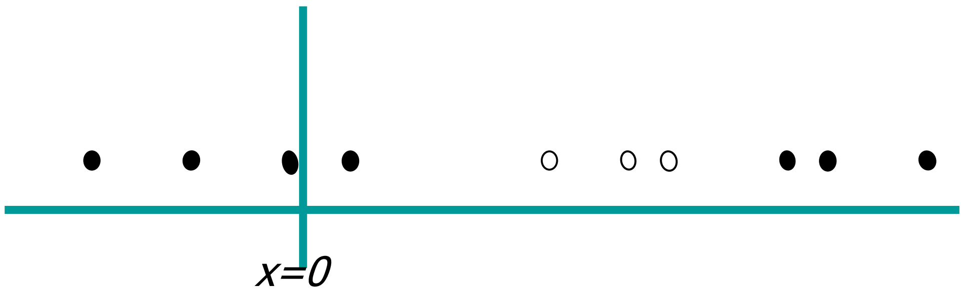
$$Minimize \frac{1}{2}\mathbf{w.w} + C\sum_{k=1}^{R}\varepsilon_k$$

SVM Parameter - C

- Controls training error
- Used to prevent over-fitting
- Lets play with C – Demo Link

Learning Non-Linear Patterns

Yet another challenge to Linear SVM!

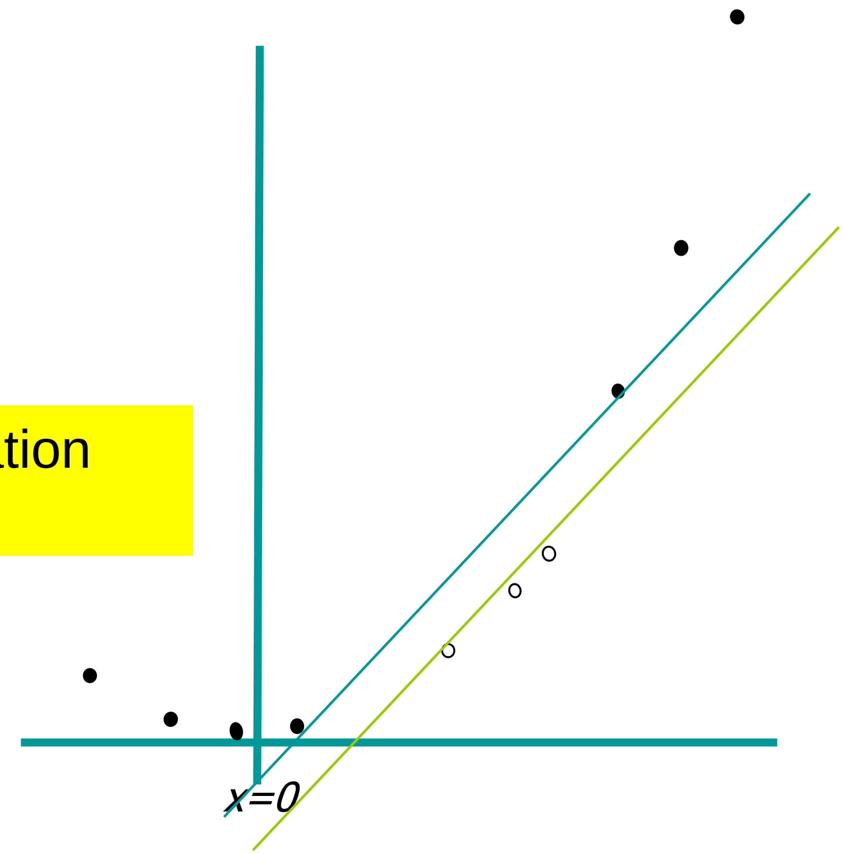


What can be done about this?

Is it time to give up? ☺

Harder 1-dimensional Dataset

Let's map the data points into a higher dimensional space.

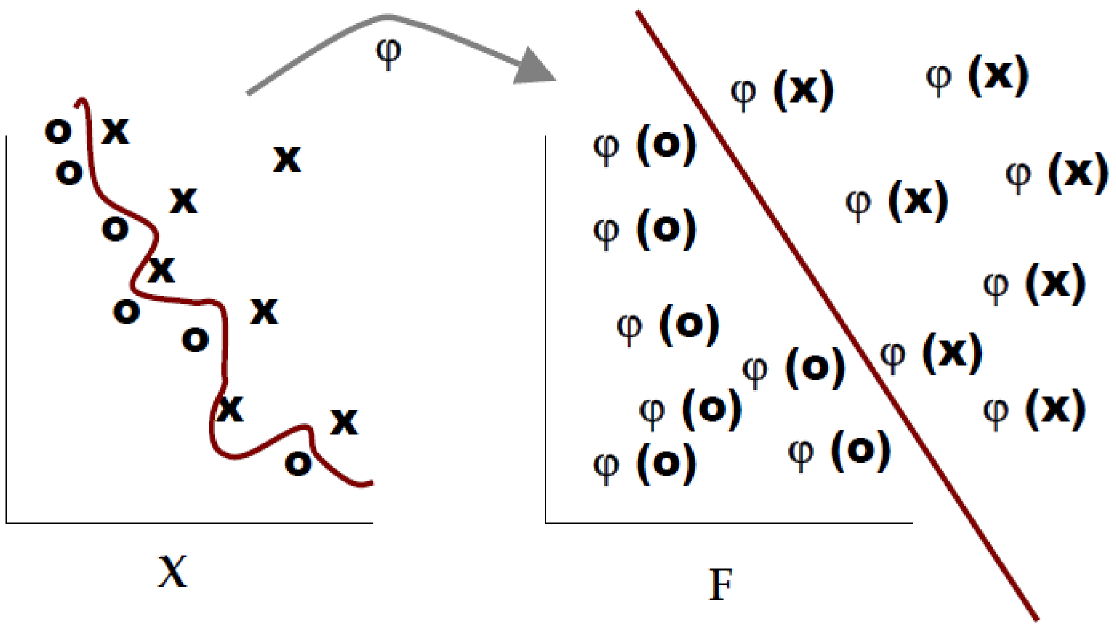


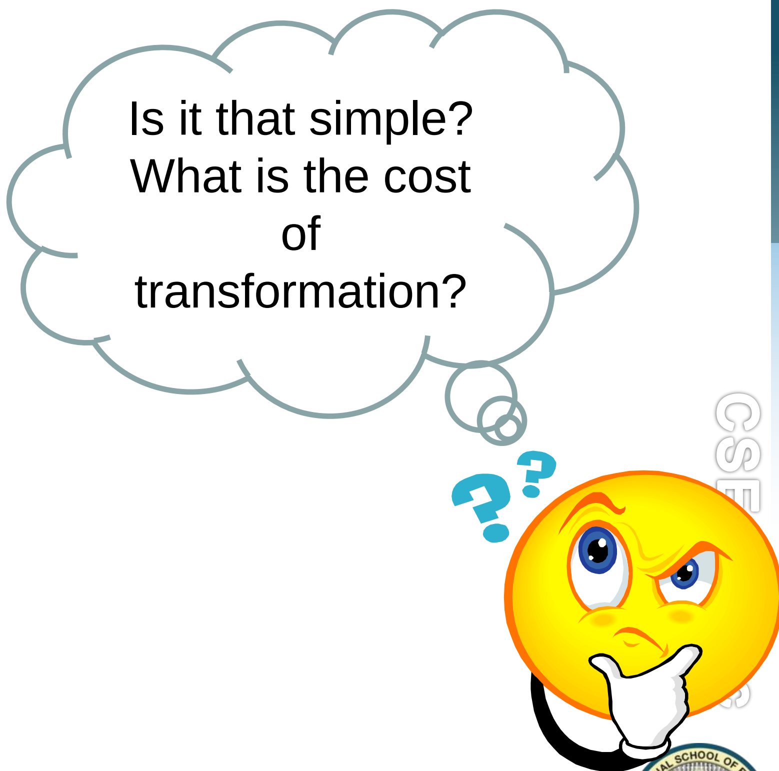
Linear SVM formulation still works! ☺

$$\mathbf{z}_k = (x_k, x_k^2)$$

Non-Linear SVMs: Feature Space Mappings

- Transform the original training samples onto a higher dimensional feature space





Learning Quadratic Functions

$$
\mathbf{\Phi}(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}
\begin{array}{l} \} \text{ Constant Term} \\ \} \text{ Linear Terms} \\ \} \text{ Pure Quadratic Terms} \\ \} \text{ Quadratic Cross-Terms} \end{array}
$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

Number of terms (assuming m input dimensions) =

= (m+1)-Choose-2
= (m+1).(m)/2
= $O(m^2/2)$

In general, for degree ‘d’ polynomials:

(d+m-1)-Choose-d
=$O(m^d)$

The Kernel Trick

$$\mathit{Minimize}\ \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

$\mathbf{w} \,.\, \mathbf{x}_k + b >= 1-\varepsilon_k$ if $y_k = 1$

$\mathbf{w} \,.\, \mathbf{x}_k + b <= -1+\varepsilon_k$ if $y_k = -1$

Original QP

Using optimization theory transformed into Dual Problem

Training examples only occur as dot products of each other and not individually!

Maximize $$\sum_{k=1}^{R}\alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R}\alpha_k\alpha_l y_k y_l < x_k . x_l >$$

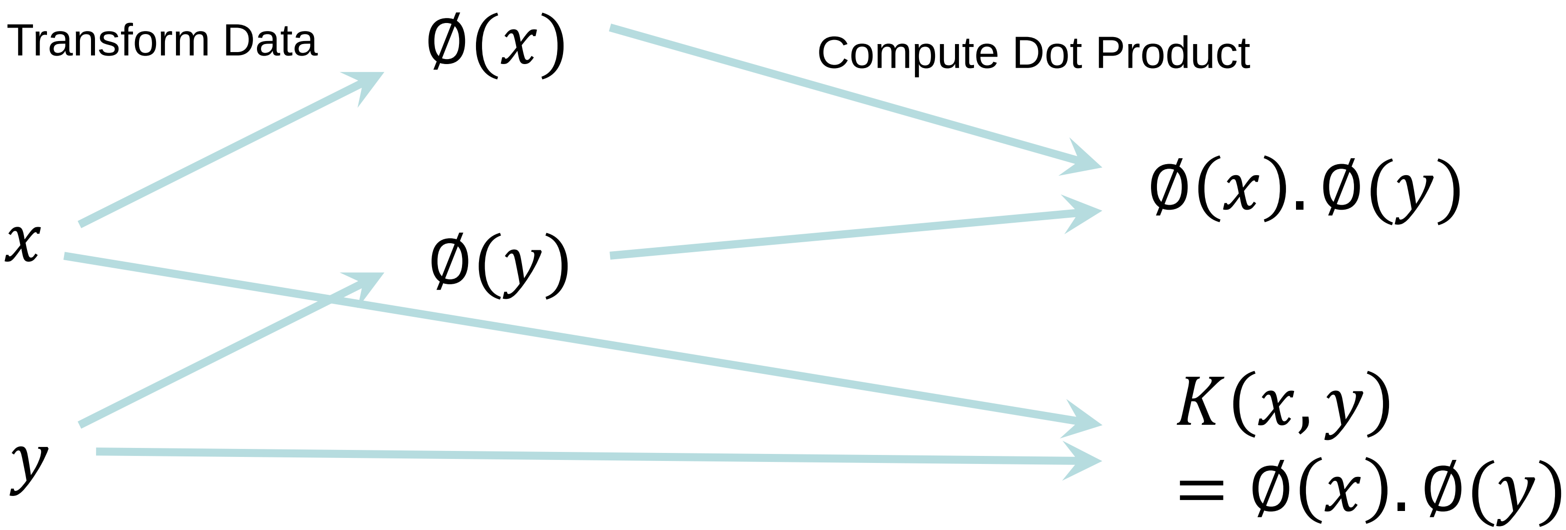
Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k \qquad \sum_{k=1}^{R}\alpha_k y_k = 0$

$$\mathbf{w} = \sum_{k=1}^{R}\alpha_k y_k \mathbf{x}_k$$

$$b = y_K(1-\varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

The Kernel Trick



K is known as the **Kernel Matrix**

Captures the similarity between training instances x and y

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

$$1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

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

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_ib_i\right)^2 + 2\sum_{i=1}^{m} a_ib_i + 1$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ib_ia_jb_j + 2\sum_{i=1}^{m} a_ib_i + 1$$

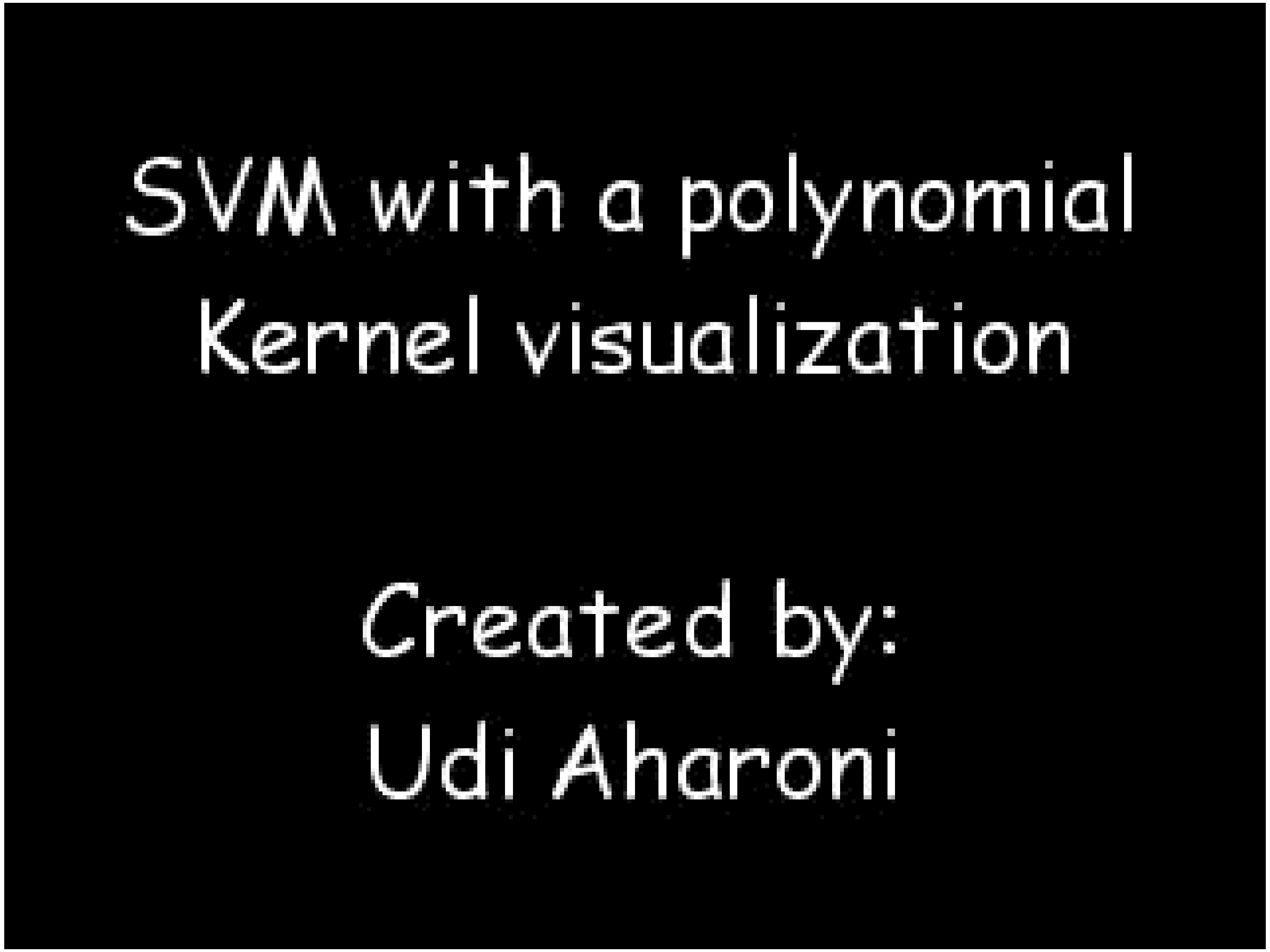
$$= \sum_{i=1}^{m}(a_ib_i)^2 + 2\sum_{i=1}^{m}\sum_{j=i+1}^{m} a_ib_ia_jb_j + 2\sum_{i=1}^{m} a_ib_i + 1$$

Both are same!

And K(a,b) is only O(m) to compute!

CSE 7306c

Polynomial Kernel Visualization



CSE 7306c

SVM Kernel Functions

- Polynomial Kernel

$$K(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}.\boldsymbol{b} + 1)^d$$

- Radial Basis Kernel

$$K(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$$

- Hyperbolic Tangent

$$K(\mathbf{a}, \mathbf{b}) = \tanh(\kappa \mathbf{a}.\mathbf{b} - \delta)$$

Project onto infinite dimensional space

SVM Parameter Tuning

- Is very important
- C, Kernel parameters
- Example:

$$-\sigma \ in \ \exp\left(\frac{(\boldsymbol{a}-\boldsymbol{b})^2}{2\sigma^2}\right) \text{ RBF Kernel}$$

- How to select them?

Tuning Steps

- Map categorical features to numerical values
- Re-scale features so that features with high values don't dominate

	height	sex
x_1	150	F
x_2	180	M
x_3	185	M

$$y_1 = 0, y_2 = 1, y_3 = 1$$

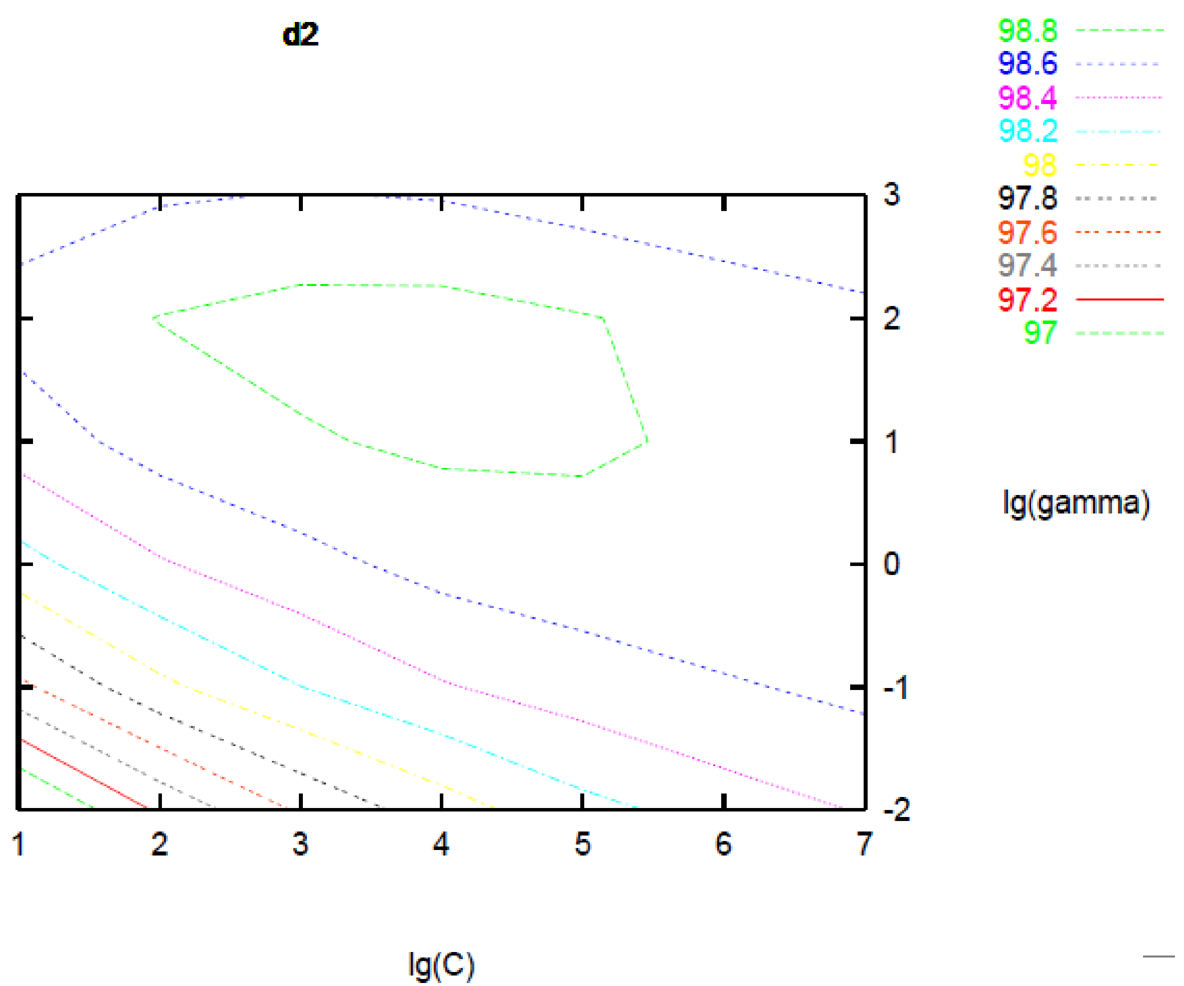
$$height = \frac{height - 150}{185 - 150}$$

	height	sex_m	sex_f
x1	0	0	1
x2	0.857143	1	0
x3	1	1	0

Tuning Steps (Contd..)

- Pre-process (scaling, numerical mapping etc.) training data
- Pick up RBF Kernel
- Use Cross-Validation to find the best C and σ parameter values
- Use the best C, σ to train on the entire training set
- Test

Tuning SVMs (Contd..)



Handling Class Imbalance

- Many practical applications have class imbalance
  - Detecting Fraud Transactions
  - Cancer Detection
- Default SVM formulation will learn highly skewed hyperplanes
- However, we want generalization errors to be equally distributed between positive and negative class

$$\min_{w,b} \frac{1}{2}||w||_2^2 + \frac{C}{N_+} \sum_{j\,:\,y_j=+1} \xi_j + \frac{C}{N_-} \sum_{j\,:\,y_j=-1} \xi_j$$

SVMs – Advantages

- Flexibility in choosing the similarity function (by choosing the kernel)
- Exploits the sparseness of solution when dealing with large datasets
  - Only support vectors specify the hyperplane irrespective of size of dataset
- Can handle large feature sets efficiently
  - Complexity doesn't depend on the dimensionality of feature space
- Good theoretical guarantees
  - Maximum margin generalizes better
  - Convex optimization problem which is guaranteed to converge

SVMs – Shortcomings

- Sensitive to noise and outliers
  - Some noisy or outlier training samples can significantly alter the hyperplane and hence the performance
- SVM doesn’t provide a *posterior probability* which is required for many applications
- Formulation is not clean for multi-labeled classification problems
  - For ‘m’ classes, train ‘m’ One vs. Rest Binary Classifiers
    - Input can be assigned to multiple classes
    - Results in class imbalance
  - Train m (m-1)/2 different binary classifiers on pairs of classes
    - Classify test points based on which class receives highest votes
    - For large ‘m’ requires significantly larger training time

Summary

- Support Vector Machines (SVMs) work very well in practice for a large class of classification problems
- SVMs work on the principle of learning a maximum margin hyperplane which results in good generalization
- The basic linear SVM formulation could be extended to handle noisy and non-separable data
- The Kernel Trick could be used to learn complex non-linear patterns
- For better performance, one has to tune the SVM parameters such as C, kernel parameters using validation set

References

- John Shawe-Taylor and Nello Cristianini. 2000. *Support Vector Machines and other Kernel-Based Learning Methods.* Cambridge University Press.
- Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition
- S. T. Dumais. 1998. Using SVMs for Text Categorization, IEEE Intelligent Systems, 13(4)
- Tong Zhang, Frank J. Oles. 2001. Text Categorization Based on Regularized Linear Classification Methods. Information Retrieval 4(1): 5-31
- Trevor Hastie, Robert Tibshirani and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer-Verlag, New York.
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang. 2003. A Loss Function Analysis for Classification Methods in Text Categorization. ICML 2003: 472-479.
- Tie-Yan Liu, Yiming Yang, Hao Wan, et al. 2005. Support Vector Machines Classification with Very Large Scale Taxonomy, SIGKDD Explorations, 7(1): 36-43.

International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

Web: http://www.insofe.edu.in

Facebook: https://www.facebook.com/insofe

Twitter: https://twitter.com/Insofeedu

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE

LinkedIn: http://www.linkedin.com/company/international-school-of-engineering

This presentation may contain references to findings of various reports available in the public domain. INSOFE makes no representation as to their accuracy or that the organization subscribes to those findings.