Support Vector Machine II

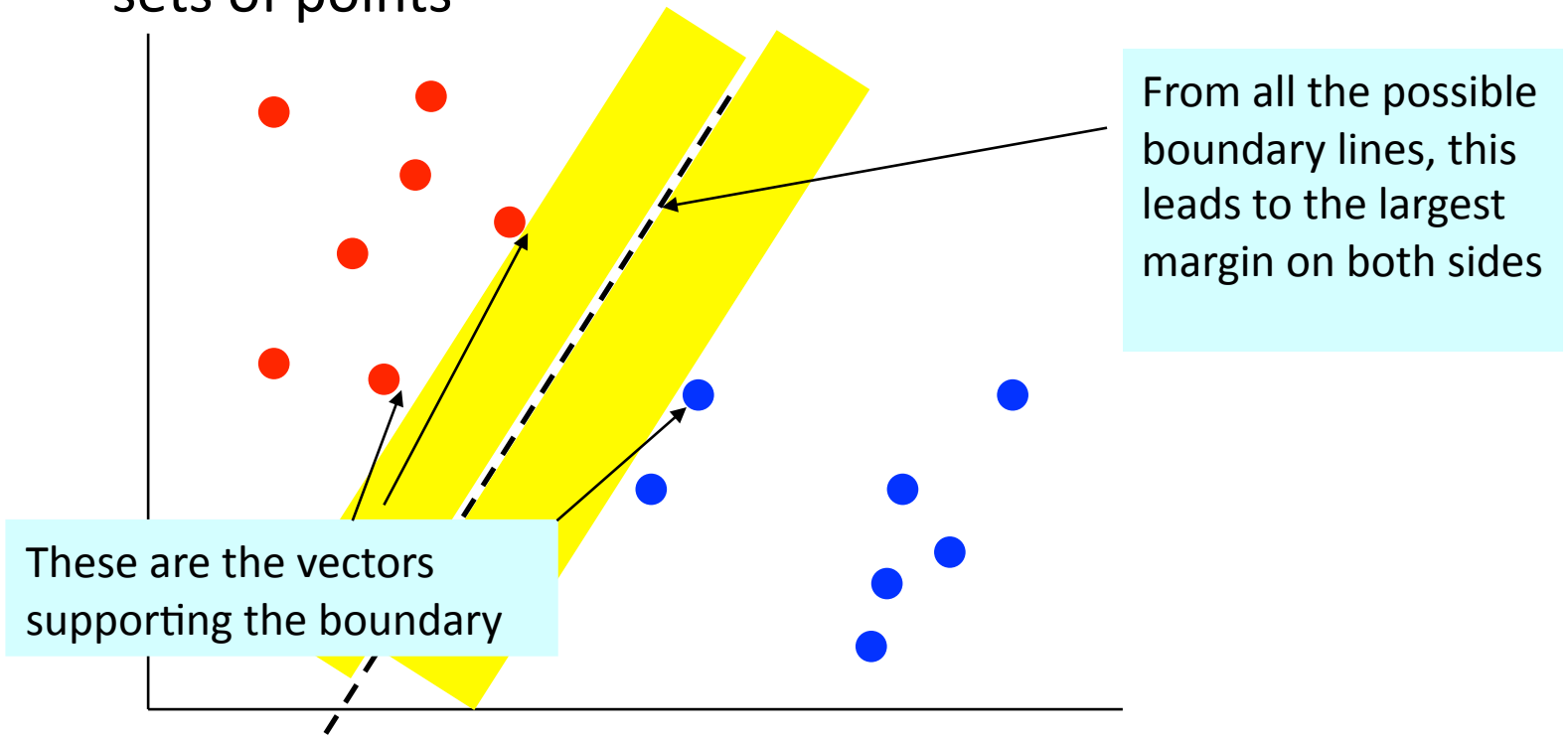
Machine Learning 10-601B

Seyoung Kim

Many of these slides are derived fromTom Mitchell, Ziv Bar-Joseph. Thanks!

Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from both sets of points



Support Vector Machines

Two optimization problems: For the separable and non separable cases

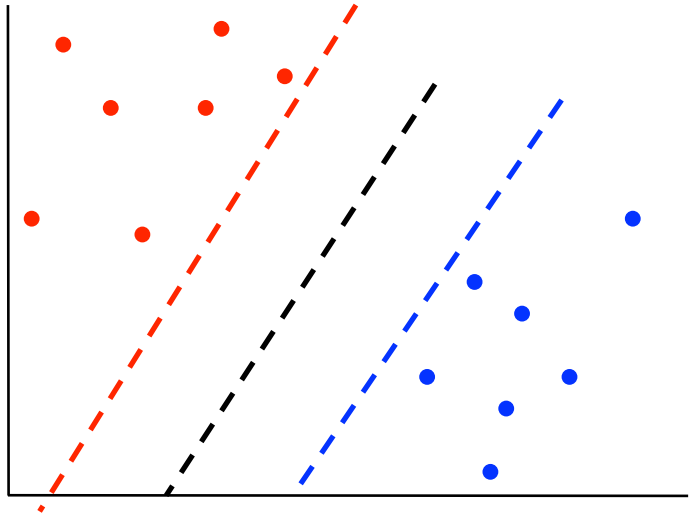
$$\min_w \frac{w^T w}{2}$$

For all x in class + 1

$$w^T x+b \geq 1$$

For all x in class - 1

$$w^T x+b \leq -1$$



$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

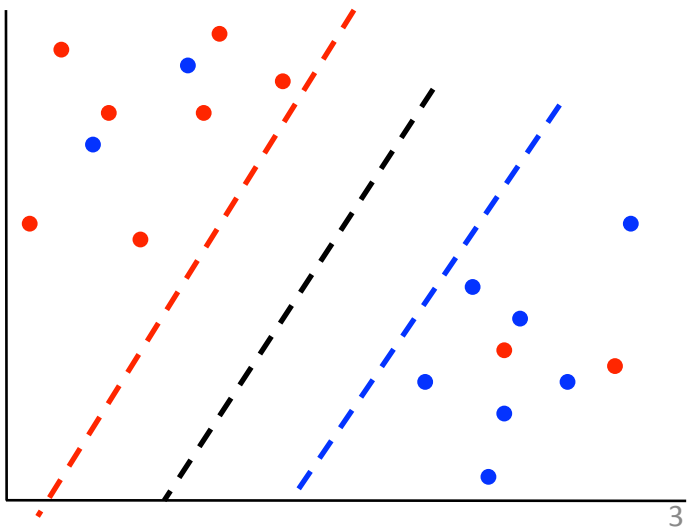
$$w^T x+b \geq 1- \varepsilon_i$$

For all x_i in class - 1

$$w^T x+b \leq -1+ \varepsilon_i$$

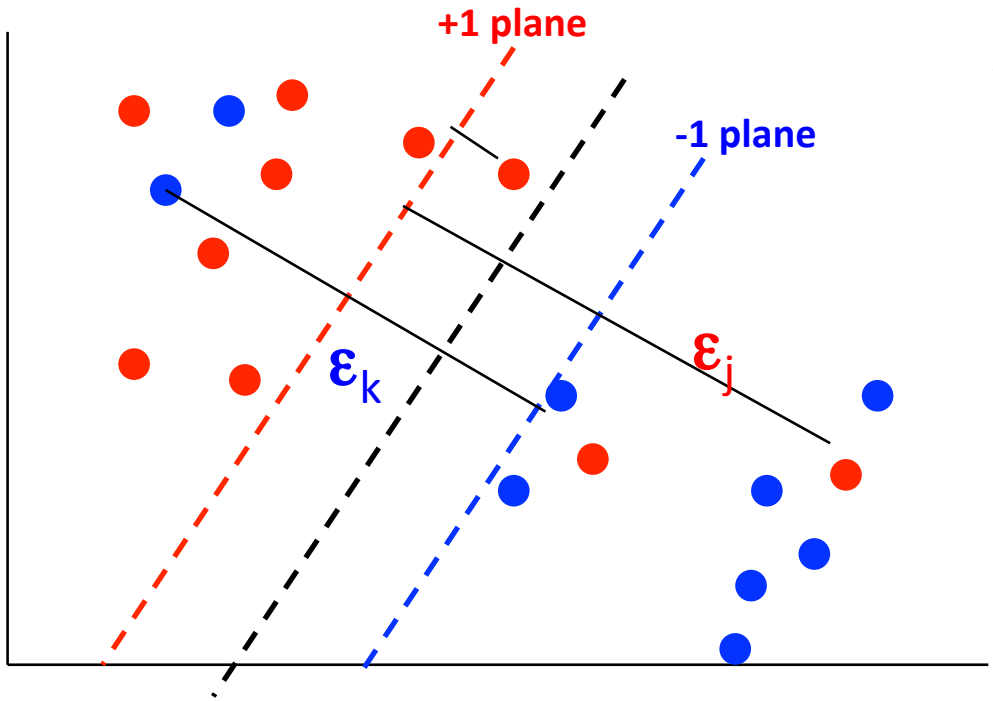
For all i

$$\varepsilon_i \geq 0$$



Non linearly separable case

• Instead of minimizing the number of misclassified points we can minimize the *distance* between these points and their correct plane



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$$w^T x + b \geq 1 - \varepsilon_i$$

For all x_i in class - 1

$$w^T x + b \leq -1 + \varepsilon_i$$

Wait. Are we missing something?

Support Vector Machines

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x+b \geq 1$

For all x in class - 1

$w^T x+b \leq -1$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x+b \geq 1- \varepsilon_i$

For all x_i in class - 1

$w^T x+b \leq -1+ \varepsilon_i$

For all i

$\varepsilon_i \geq 0$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

An alternative (dual) representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use Lagrange multipliers to encode it as part of our minimization problem

Min $(w^T w)/2$

For all x in class +1

$w^T x+b \geq 1$

For all x in class -1

$w^T x+b \leq -1$

Why? ⇓

Min $(w^T w)/2$

$(w^T x_i+b)y_i \geq 1$

An alternative (dual) representation of the SVM QP

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

- We will start with the linearly separable case
- Instead of encoding the correct classification rule a constraint we will use Lagrange multipliers to encode it as part of our minimization problem

Recall that Lagrange multipliers can be applied to turn the following problem:

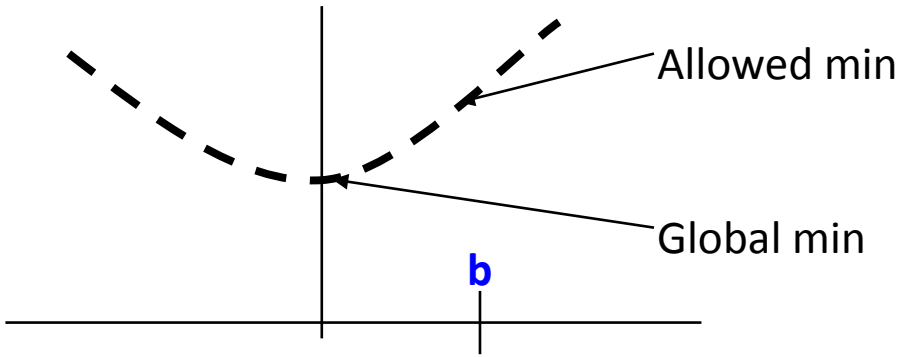
$\min_x x^2$

s.t. $x \geq b$

To

$\min_x \max_\alpha x^2 - \alpha(x-b)$

s.t. $\alpha \geq 0$



Lagrange multiplier for SVMs

Dual formulation

$$\min_{w,b} \max_{\alpha} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

w: primal parameters

α_i's: dual parameters

Original (primal) formulation

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

Lagrange multiplier for SVMs

Dual formulation

$$\min_{w,b} \max_{\alpha} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

Original (primal) formulation

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

Using this new formulation we can derive w and b by taking the derivative w.r.t. w leading to:

$$w = \sum_i \alpha_i x_i y_i, \ \ where \ \ \alpha_i \geq 0$$

taking the derivative w.r.t. b we get:

$$\sum_i \alpha_i y_i = 0$$

Lagrange multiplier for SVMs

Dual formulation

$$\min_{w,b} \max_{\alpha} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

Original (primal) formulation

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

Using this new formulation we can derive w and b by taking the derivative w.r.t. w leading to:

$$w = \sum_i \alpha_i x_i y_i, \ \ where \ \ \alpha_i \geq 0$$

taking the derivative w.r.t. b we get:

$$\sum_i \alpha_i y_i = 0$$

Substituting w into our target function and using the additional constraint we get:

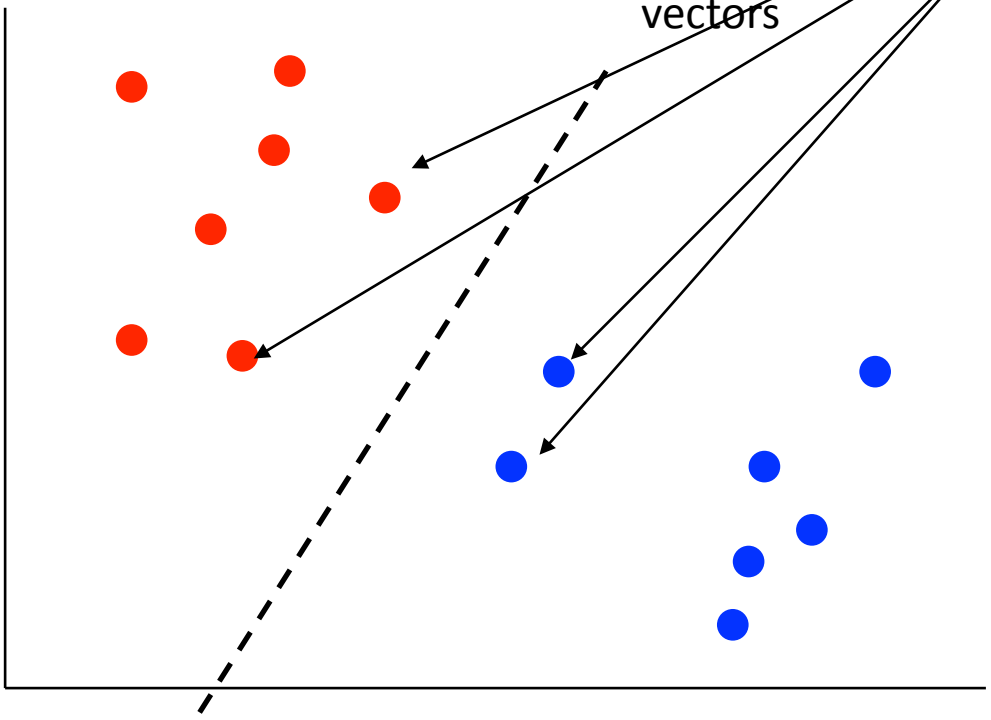
Dual formulation

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i}^{\mathbf{T}} \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dual SVM - interpretation



$$w = \sum_i \alpha_i x_i y_i$$

For α's that are not 0

Computational Cost

- During training, the computational costs for solving primal vs. dual problems are

Primal problem:

$\min (w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

m parameters

Dual problem:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_j}^T \mathbf{x_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

n parameters

- The cost of QP solver depends on #variables
- Often, n < m, where n = #samples, m = #input features
  ->Solving dual is often more efficient
- Even when n > m, working with dual allows you to use kernels!

Computational Cost

- During testing, the computational costs using primal vs. dual representations are

Using primal variables:

$$y_{new} = \text{sign}(w^T x_{new} + b)$$

m operation

Using dual variables:

$$y_{new} = sign(\sum_i \alpha_i y_i \mathbf{x_i}^T \mathbf{x_{new}} + b)$$

Dot product with all training samples?

mr operations where *r* is the number of support vectors (α_i>0)

If one uses dual parameters to make predictions, the prediction depends only on the support vectors, but this is not explicitly represented in the primal

Dual formulation for non linearly separable case

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x+b \geq 1- \varepsilon_i$

For all x_i in class - 1

$w^T x+b \leq -1+ \varepsilon_i$

For all i

$\varepsilon_l \geq 0$

$\Rightarrow$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

$(w^T x_i+b)y_i \geq 1- \varepsilon_l$

$\varepsilon_l \geq 0$

Dual formulation for non linearly separable case

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i} \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$C > \alpha_i \geq 0 \qquad \forall i$$

The only difference is that the α_i's are now bounded

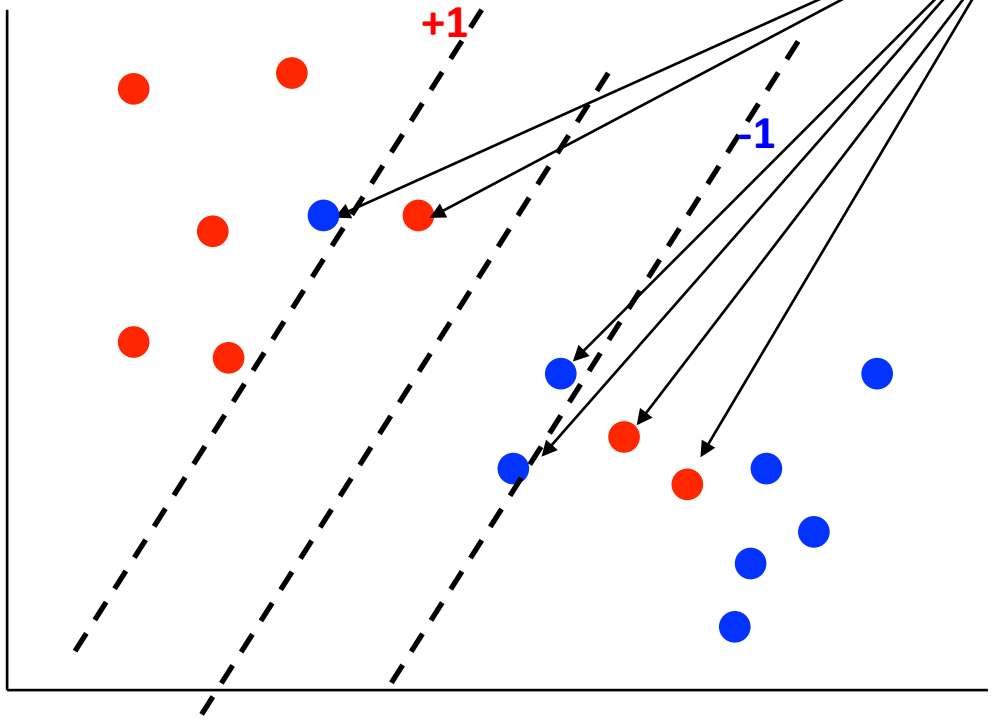
To evaluate a new sample x_j we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_j + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \mathbf{x_i} \mathbf{x_j} + b$$

Dual SVM – Interpretation for Non-linearly Separable Case

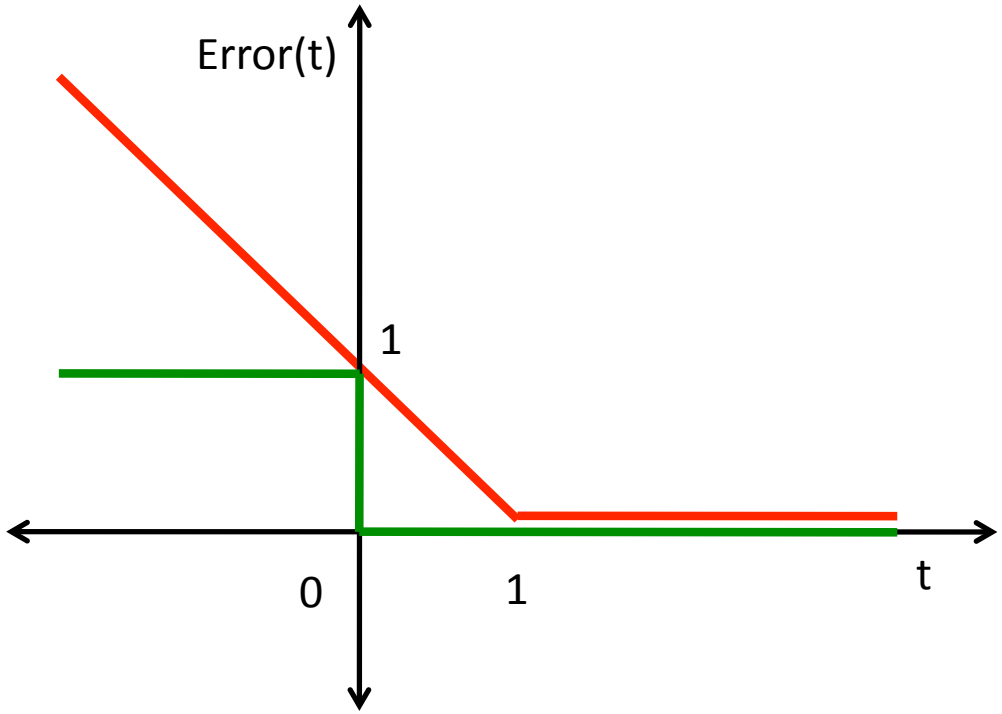
Support vectors: data points in the wrong side of margin

$$w = \sum_i \alpha_i x_i y_i$$



For α's that are not 0

Error Function for SVM



t > 0 for both positive and negative training samples if classified correctly

Let $t = (w^T x_i + b) y_i$

Ideal classifier:

$$\text{Error}(t) = \begin{cases} 0 & \text{if } t > 0 \\ 1 & \text{if } t < 0 \end{cases}$$

SVM

$$\text{Error}(t) = [1 - t]_+$$

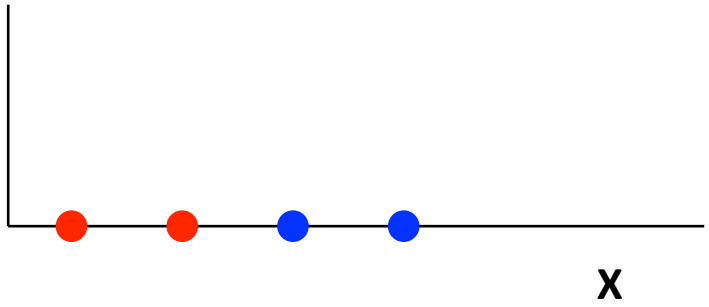
$[\ ]_+$ denotes positive part

Hinge Loss

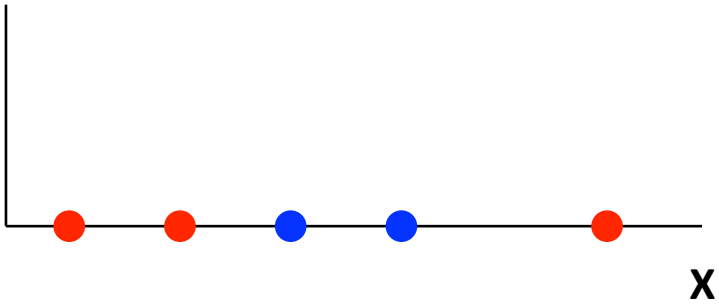
FROM LINEAR TO NON-LINEAR DECISION BOUNDARY

Classifying in 1-d

Can an SVM correctly classify this data?

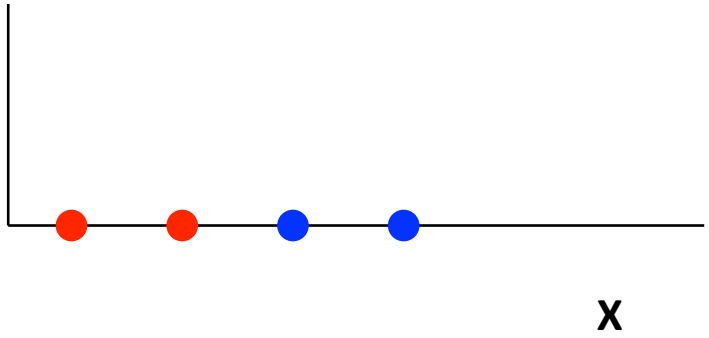


What about this?

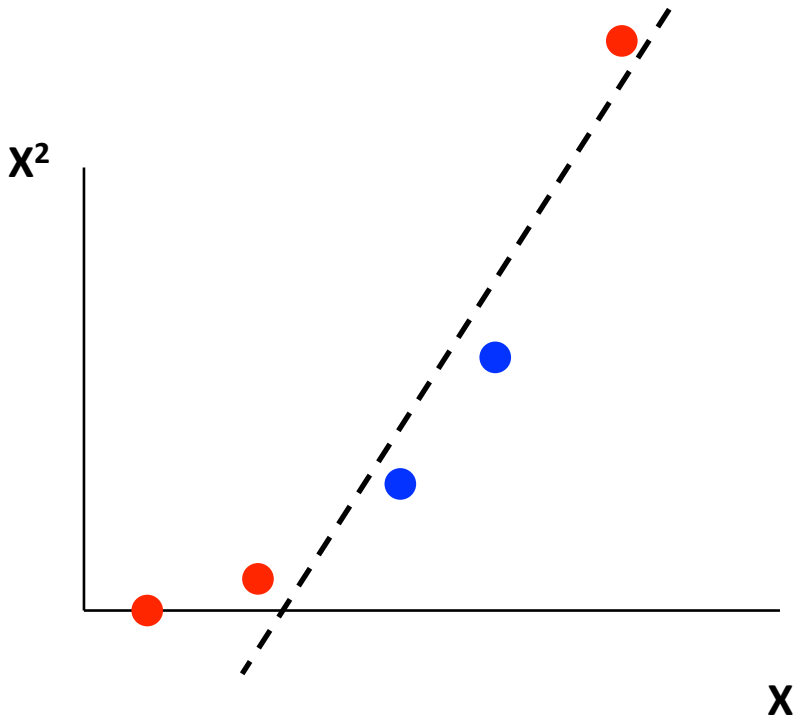


Classifying in 1-d

Can an SVM correctly classify this data?

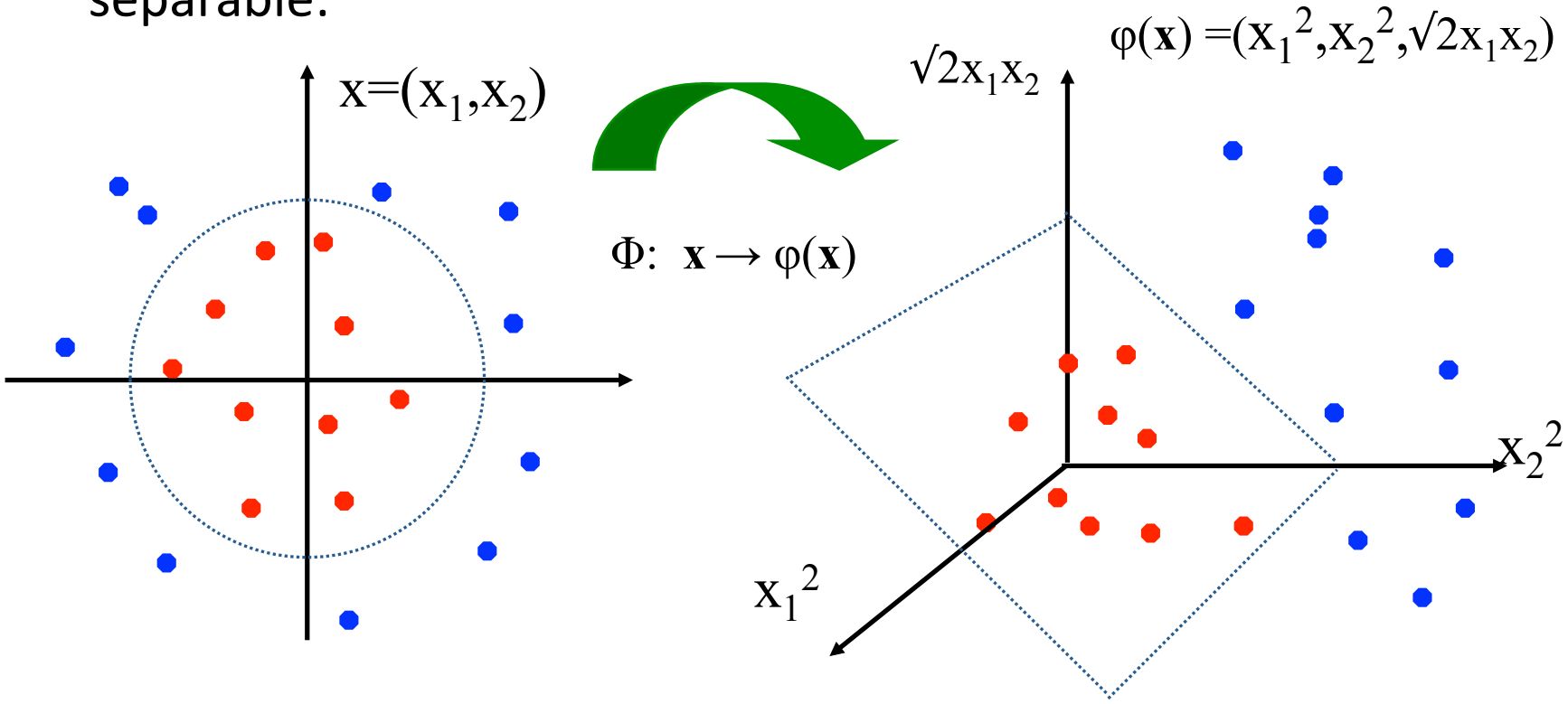


And now?



Non-linear SVMs: 2D

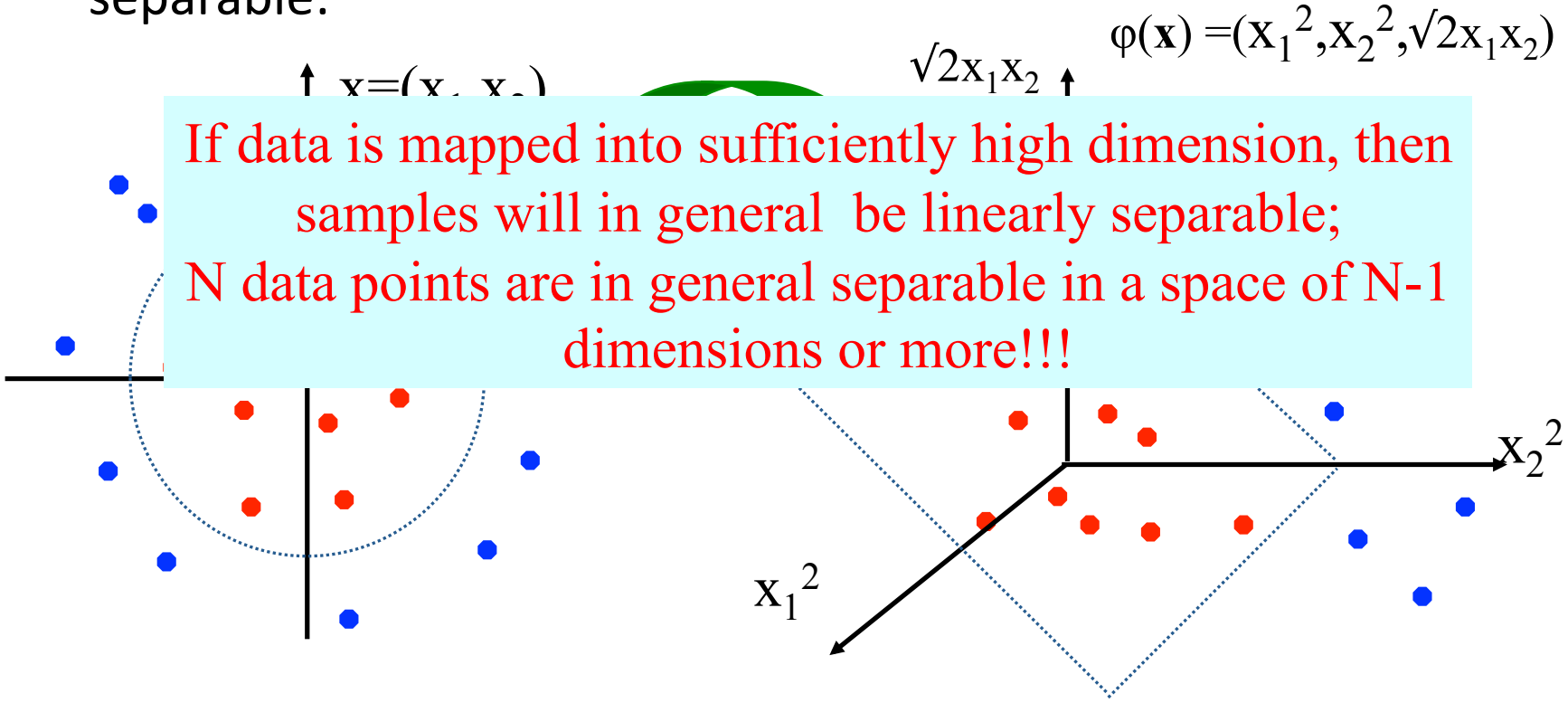
- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$) where the training set is separable:



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

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$) where the training set is separable:

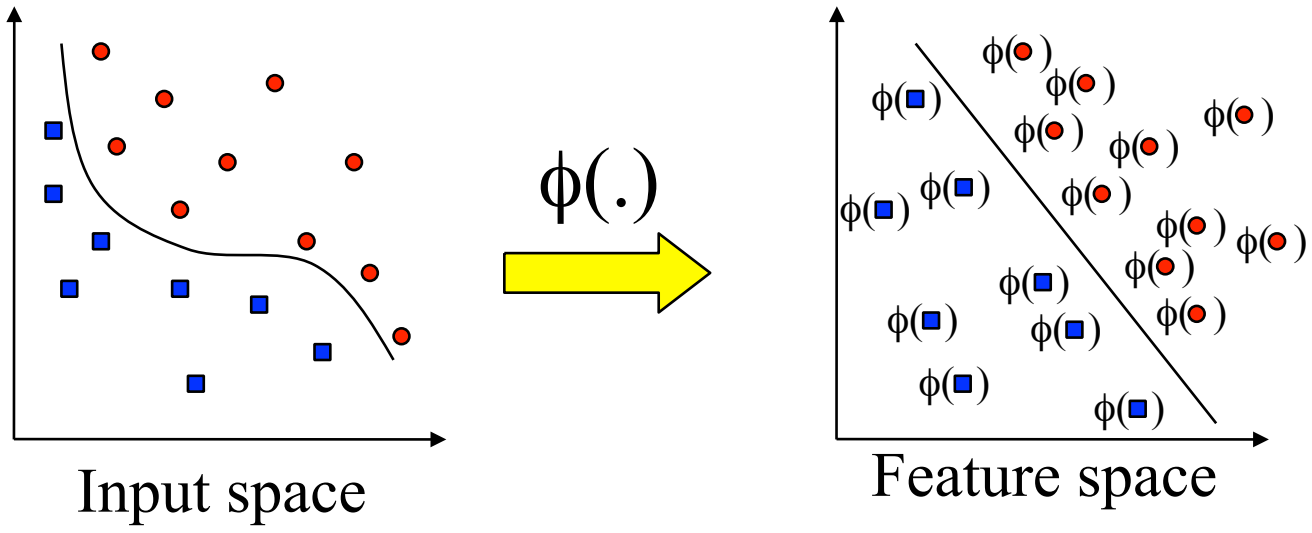


If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

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

Transformation of Inputs

- Possible problems
  - High computation burden due to high-dimensionality
  - Many more parameters
- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation
  - Dual formulation only assigns parameters to samples, not features



Quadratic kernels

- While working in higher dimensions is beneficial, it also increases our running time because of the dot product computation

$$\max_\alpha \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x_i}) \Phi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

- However, there is a neat trick we can use

- consider all quadratic terms for x_1, x_2 … x_m

m is the number of features in each vector

The √2 term will become clear in the next slide

$$\Phi(x) = \begin{matrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \end{matrix}$$

m+1 linear terms

$$\begin{matrix} x_1^2 \\ \vdots \\ x_m^2 \end{matrix}$$

m quadratic terms

$$\begin{matrix} \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{matrix}$$

m(m-1)/2 pairwise terms

Dot product for quadratic kernels

How many operations do we need for the dot product?

$$\Phi(x)\Phi(z) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \sqrt{2}z_1 \\ \vdots \\ \sqrt{2}z_m \\ z_1^2 \\ \vdots \\ z_m^2 \\ \sqrt{2}z_1z_2 \\ \vdots \\ \sqrt{2}z_{m-1}z_m \end{bmatrix} = \sum_i 2x_iz_i + \sum_i x_i^2z_i^2 + \sum_i \sum_{j=i+1} 2x_ix_jz_iz_j + 1$$

m m m(m-1)/2 **=~ m^2**

The kernel trick

How many operations do we need for the dot product?

$$\Phi(x)\Phi(z) = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

m m m(m-1)/2 **=~ m^2**

However, we can obtain dramatic savings by noting that

$$\Phi(x)\Phi(z) = k(x,z) = (x^T z + 1)^2 = (x^T z)^2 + 2(x^T z) + 1$$

$$= (\sum_i x_i z_i)^2 + \sum_i 2x_i z_i + 1$$

$$= \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

We only need m operations!

Note that to evaluate a new sample we are also using dot products so we save there as well

Kernel SVM

Our dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{\Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})}$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

To evaluate a new sample x_j we need to compute:

$$w^T x_j + b = \sum_i \alpha_i y_i \boxed{\Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})} + b$$

⇓ Using kernels

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{k(\mathbf{x_i}, \mathbf{x_j})}$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

⇓

$$w^T x_j + b = \sum_i \alpha_i y_i \boxed{k(\mathbf{x_i}, \mathbf{x_j})} + b$$

Other kernels

• The kernel trick works for higher order polynomials as well.

• For example, a polynomial of degree 4 can be computed using $(x^T z+1)^4$ and, for a polynomial of degree d $(x^T z+1)^d$

• Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function
-Radial-Basis-style Kernel Function:

$$K(x,z) = \exp\left(-\frac{(x-z)^2}{2\sigma^2}\right)$$

- Neural-net-style Kernel Function:

$$K(x,z) = \tanh(\kappa x.z - \delta)$$

Why do SVMs work?

• If we are using huge feature spaces (with kernels) how come we are not overfitting the data?

- Number of parameters remains the same (and most are set to 0)

- While we have a lot of input values, at the end we only care about the support vectors and these are usually a small group of samples

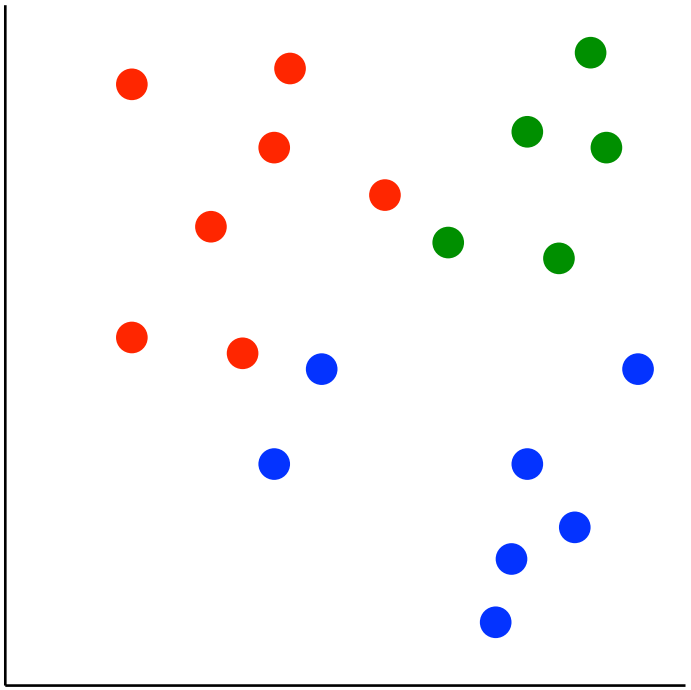
- The minimization (or the maximizing of the margin) function acts as a sort of regularization term leading to reduced overfitting

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Multi-class classification with SVMs

What if we have data from more than two classes?



- Most common solution: One vs. all

- create a classifier for each class against all other data

- for a new point use all classifiers and compare the margin for all selected classes

Note that this is not necessarily valid since this is not what we trained the SVM for, but often works well in practice

Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→Lots of very successful applications!!!

Important points

- Maximum margin principle
- Target function for SVMs
- Linearly separable and non separable cases
- Dual formulation of SVMs
- Support vectors of SVMs
- Kernel trick and computational complexity