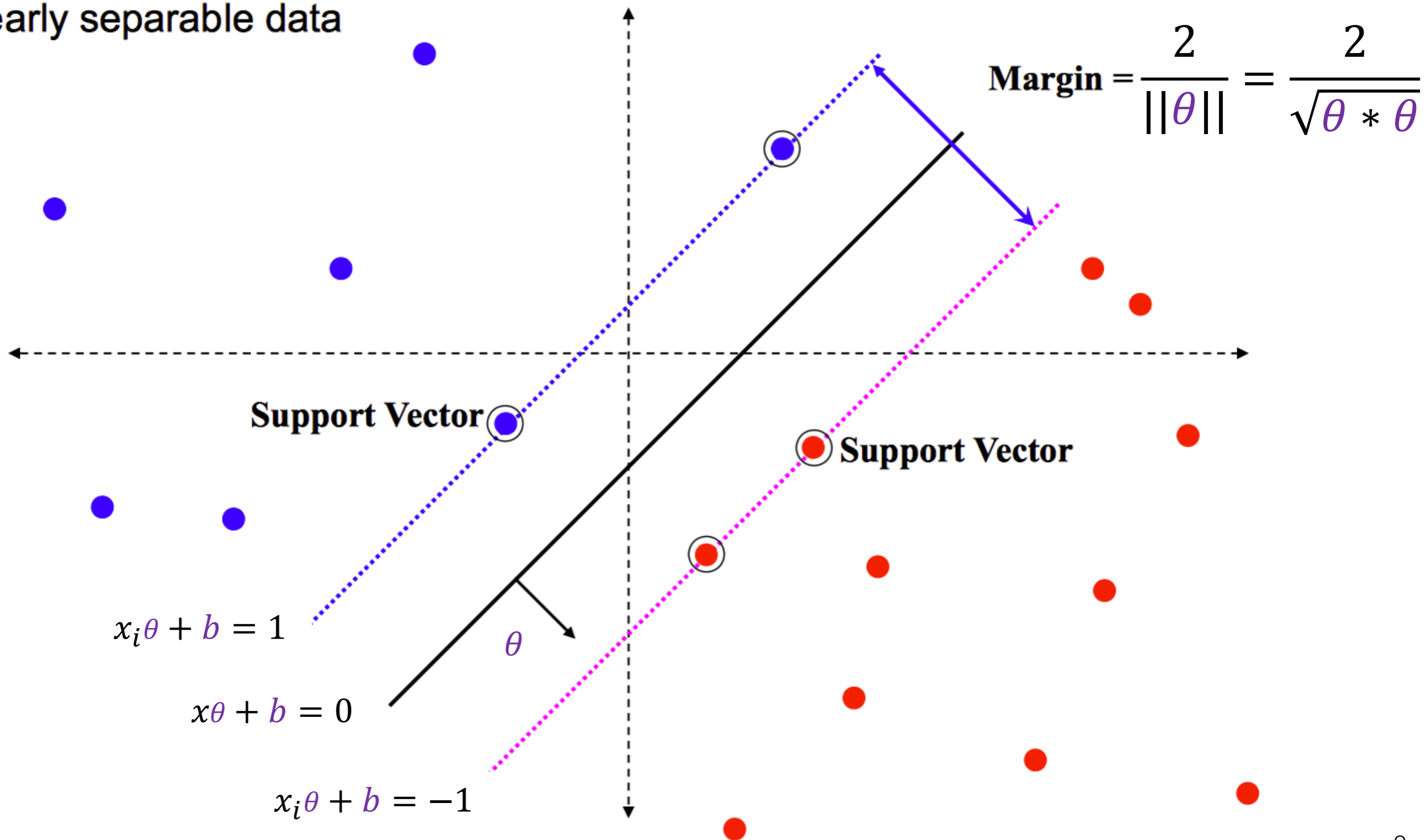


Kernel SVM

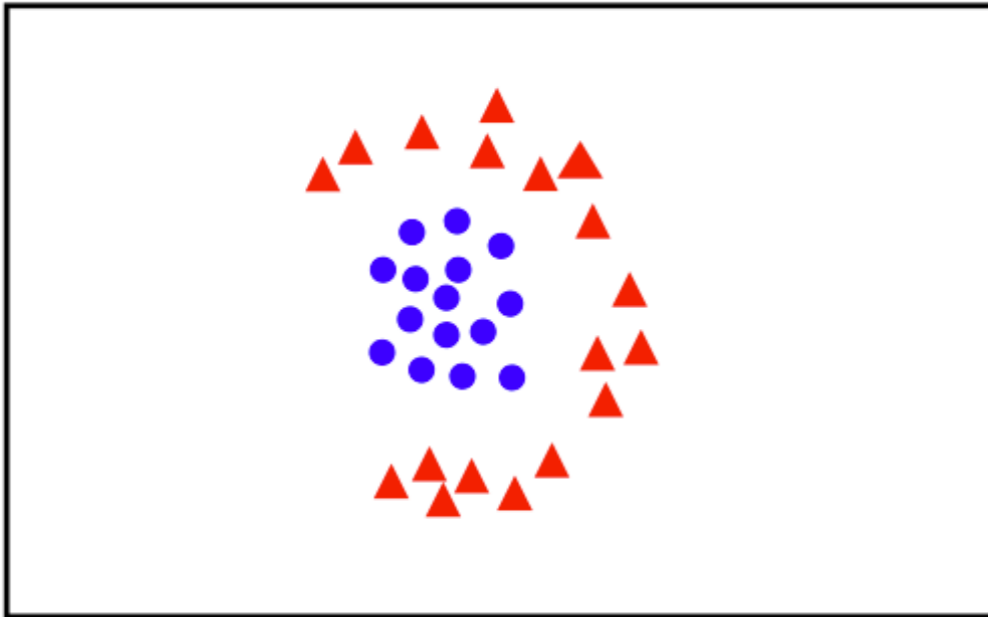
Nakul Gopalan
Georgia Tech

Geometric Interpretation

linearly separable data



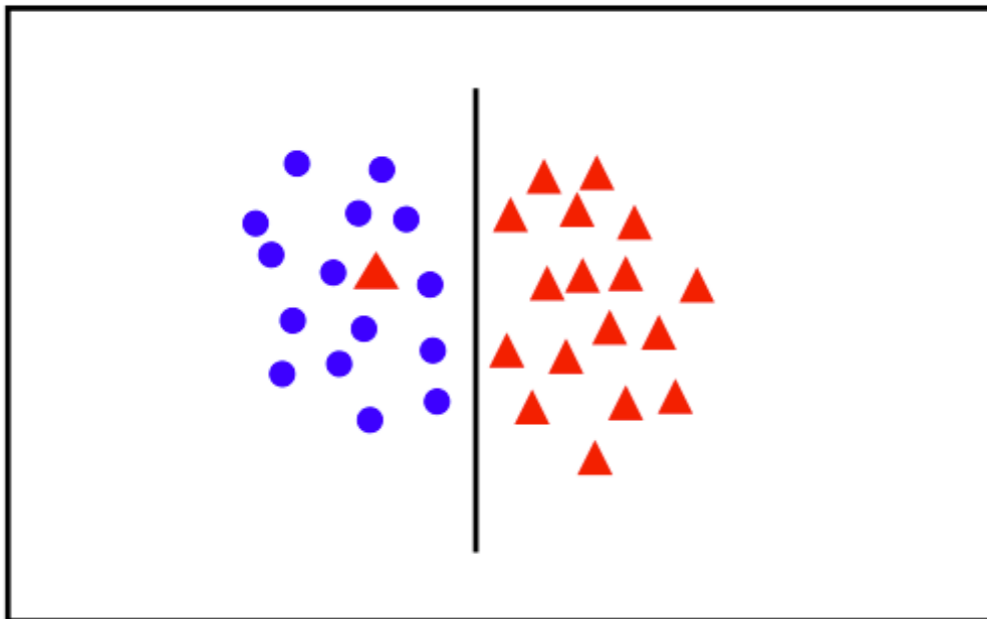
Handling Data that are Not Linearly Separable



- linear classifier not appropriate

??

Kernel trick

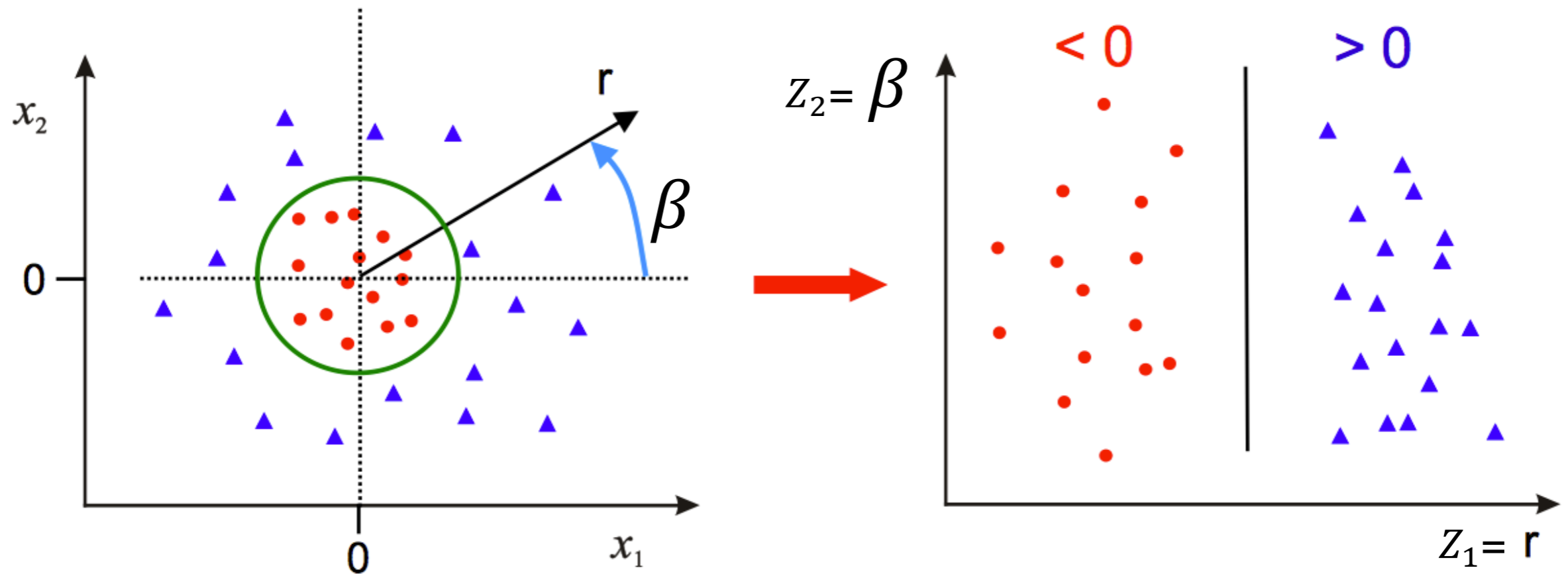


- introduce slack variables

Soft Margin SVM

(allowing ourselves to make errors)

Idea 1: Use Polar Coordinates to go to z space

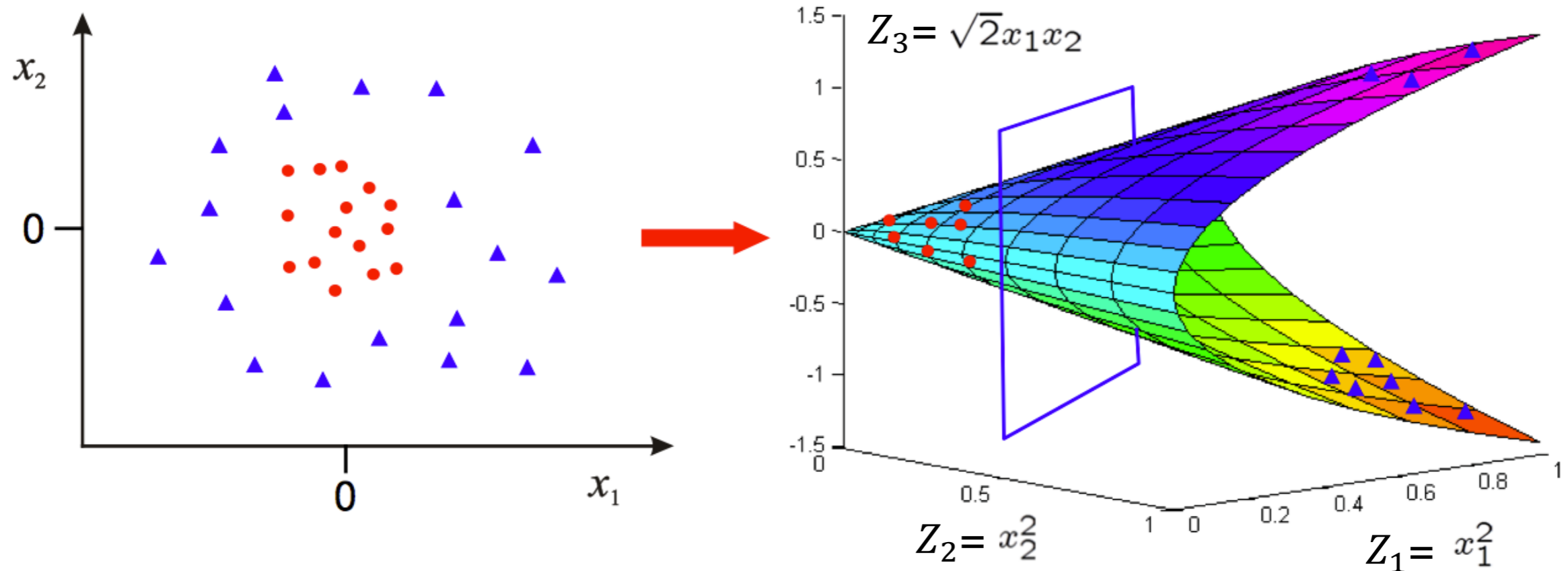


- Data **is** linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \beta \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

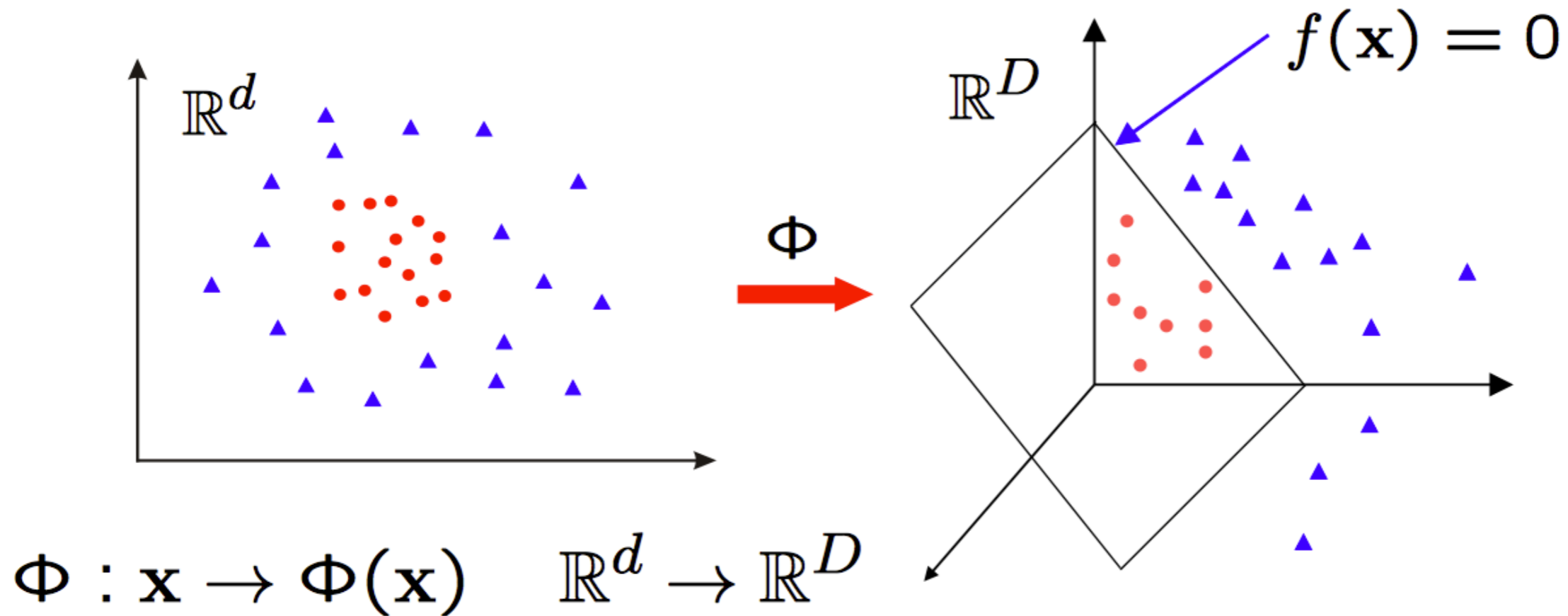
Idea 1: Map Data to Higher Dimension \mathbb{Z} space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



- Data **is** linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

SVM in a Transformed Feature Space



Learn classifier linear in \mathbf{w} for \mathbb{R}^D :

$$f(x) = \phi(x)\theta + \theta_0 = z\theta + \theta_0$$

$\Phi(\mathbf{x})$ is a feature map

Kernel trick – what do we need from \mathbb{Z} space

$$l(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \mathbf{z}_i \mathbf{z}_j^T \xrightarrow{\text{Inner products}}$$

Constraints: $\alpha_i \geq 0$ for $i = 1, \dots, N$ and $\sum_{i=1}^N \alpha_i y_i = 0$

We already have this:

$$\begin{bmatrix} y_1 y_1 \mathbf{z}_1 \mathbf{z}_1^T & y_1 y_2 \mathbf{z}_1 \mathbf{z}_2^T & \dots & y_1 y_N \mathbf{z}_1 \mathbf{z}_N^T \\ y_2 y_1 \mathbf{z}_2 \mathbf{z}_1^T & y_2 y_2 \mathbf{z}_2 \mathbf{z}_2^T & \dots & y_2 y_N \mathbf{z}_2 \mathbf{z}_N^T \\ \dots & \dots & \dots & \dots \\ y_N y_1 \mathbf{z}_N \mathbf{z}_1^T & y_N y_2 \mathbf{z}_N \mathbf{z}_2^T & \dots & y_N y_N \mathbf{z}_N \mathbf{z}_N^T \end{bmatrix}$$

Same result as hard SVM:

Solve α_i using quadratic programming and predict a test data point \mathbf{z} in \mathbb{Z} space \rightarrow

$$\text{sign}(\mathbf{z}\theta + b) = \sum_{\mathbf{z}_i \text{ in } SV} \alpha_i y_i \mathbf{z}_i \mathbf{z} + b$$

Generalized inner product

Given **two points** x and x' , we need $z'z^T$ $\begin{bmatrix} yyK(x, x) & yy'K(x, x') \\ y'yK(x', x) & y'y'K(x', x') \end{bmatrix}$

Let $z'z^T = K(x, x')$ **The kernel** inner product of x and x'

Example:

$x = (x_1, x_2) \rightarrow 2nd - order \Phi$ here x has two dimensions

$z = \Phi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)$ How many dimensions z has?

$$K(x, x') = z'z^T = 1 + x_1x_1' + x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2 + x_1x_1'x_2x_2'$$

We can also calculate: $K(x, x)$ $K(x', x)$ $K(x', x')$

The trick

Can we compute $K(x, x')$ **without** transforming x and x' ?

Example:

Datapoint 1 in x space $x = (1, x_1, x_2)$

Datapoint 2 in x space $x' = (1, x_1', x_2')$

Datapoint 1 and 2 have 2 dimensions in x space and 1 is for the biased term

Datapoint 1 in z space $\phi(x) = z = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$

Datapoint 2 in z space $\phi(x') = z' = (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2')$

Datapoint 1 and 2 have 5 dimensions in z space and 1 is for the biased term

We need to calculate the dot product for the kernel

$$K(x, x') = \phi(x') \phi(x)^T = z' z^T$$

$$z = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

$$z' = (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2')$$

$$K(x, x') = 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'$$

$$K(x, x') = (1 + x_1 x_1' + x_2 x_2')^2$$

$$x = (1, x_1, x_2) \quad x' = (1, x_1', x_2')$$

$$K(x, x') = (x' x^T)^2 \quad \text{Homogeneous kernel}$$

Example: Let's say we have two datapoints

We need to build the inner product matrix to optimize SVM parameters.

$$l(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j z_i z_j^T = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

$$X = \begin{array}{cc} \text{weight} & \text{Height} \\ \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix} & Y = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix} \end{array}$$

$$K(x_i, x_j) = \begin{bmatrix} y y K(x, x) & y y' K(x, x') \\ y' y K(x', x) & y' y' K(x', x') \end{bmatrix}$$

The polynomial kernel

$\mathcal{X} = \mathcal{R}^d$ and $\Phi: \mathcal{X} \rightarrow \mathbb{Z}$ is polynomial of order Q

The “equivalent kernel = $K(x, x') = (1 + x^T x')^Q$ Inhomogeneous kernel

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q$$

Does it matter if Q is 2 or 1000?

What will happen if we have $d = 10$ and $Q = 100$ and we want to compute the inner product explicitly?

We need to calculate the inner product of two big huge ugly vectors

We only need \mathbb{Z} space to exist

if $K(x, x')$ is an inner product in some space \mathbb{Z} , we are doing good

Example:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2) \quad \text{Radial basis kernel}$$

First thing first, this is a function of x and x'

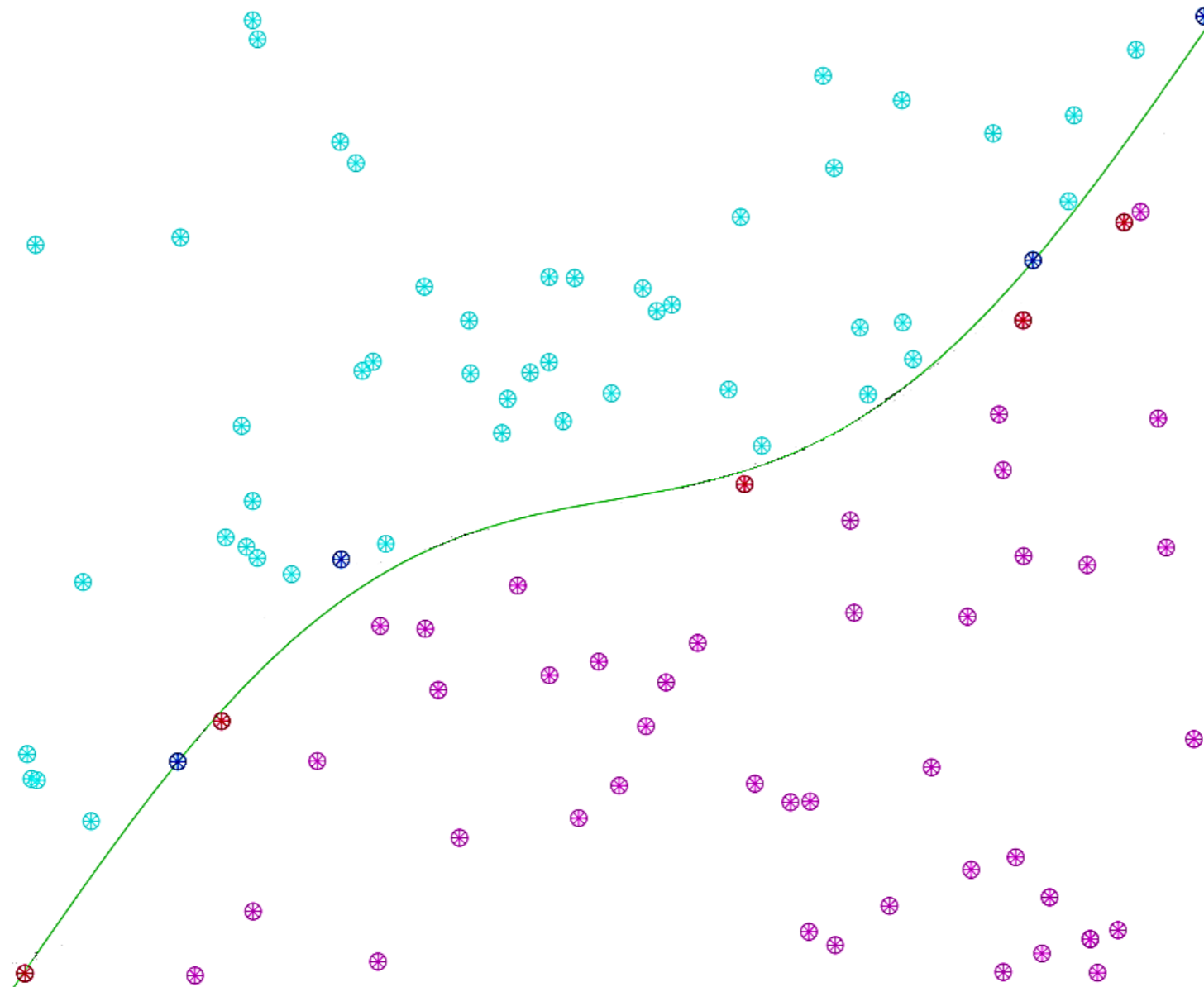
This function will take us to infinite-dimensional $\mathbb{Z} \rightarrow$ **CONGRATULATIONS**

$$\text{For } d \text{ and } \gamma = 1 \Rightarrow K(x, x') = \exp(-(x - x')^2)$$

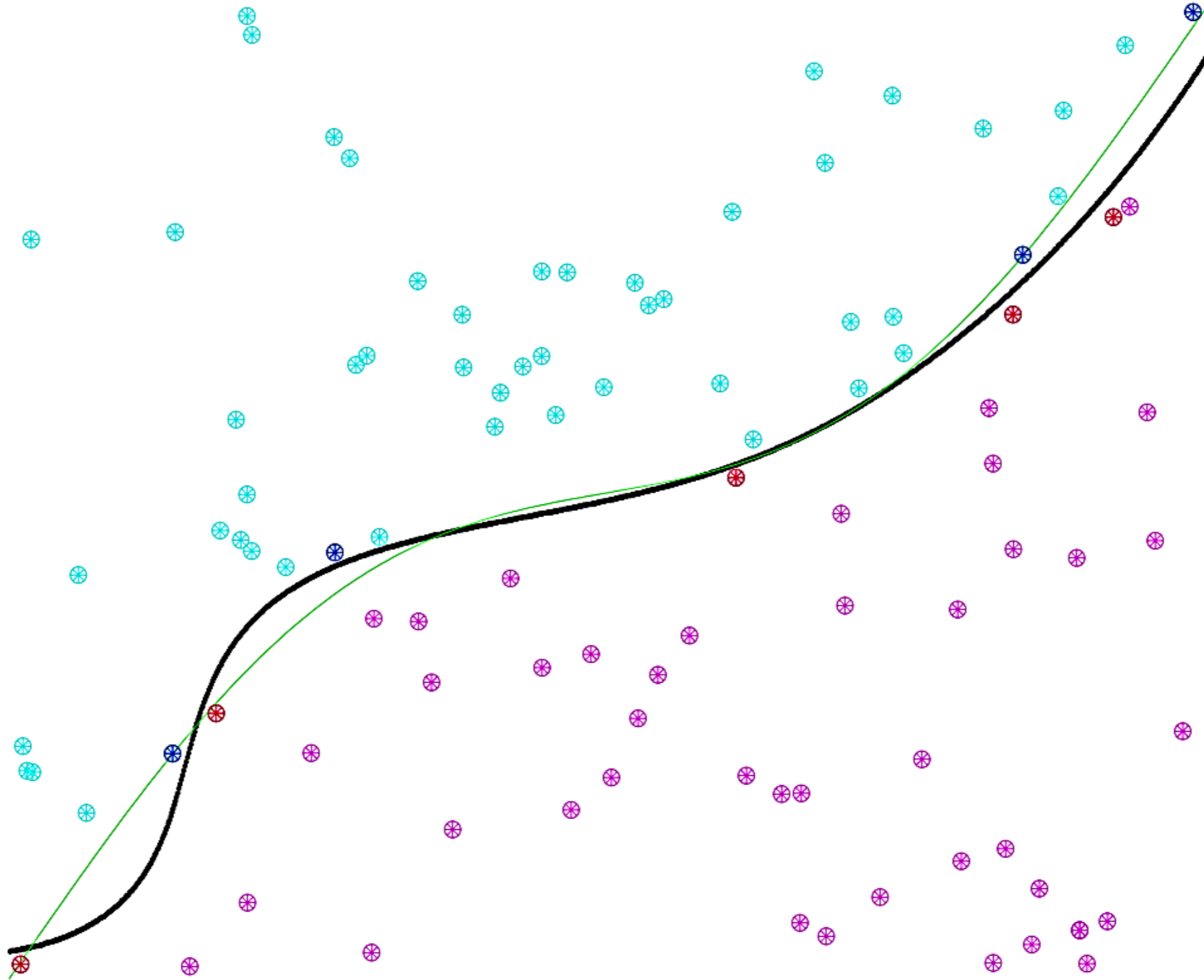
$$= \exp(-x^2) \exp(-x'^2) \underbrace{\sum_{k=0}^{\infty} \frac{2^k x^k x'^k}{k!}}_{\text{exp Taylor expansion for } \exp(2xx')}$$

Radial basis kernel in action

Slightly non-linearly separable case for 100 datapoints:



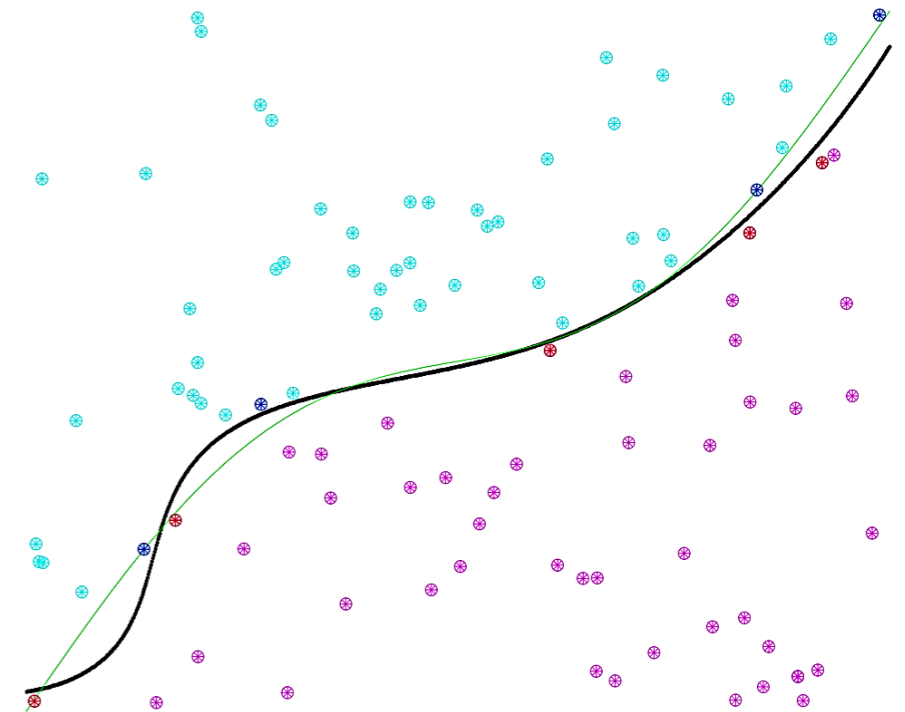
Transforming χ
into ∞ -dimensional \mathbb{Z} space



Generalization

Are we killing the generalization by going to infinite-dimension? (overfitting)

I am going to answer this with a question.
In this, example how many support vectors, we have?



What will happen if we have many support vectors?

The decision boundary line (plane) will be super wiggly → overfitting alarm

$$\mathbf{E}[E_{out}] \leq \frac{\mathbf{E}[\textit{Number of support vectors}]}{N - 1}$$

N is number of datapoints

Kernel formulation of SVM

Remember quadratic programming?

$$\begin{bmatrix} y_1 y_1 x_1 x_1^T & y_1 y_2 x_1 x_2^T & \dots & y_1 y_N x_1 x_N^T \\ y_2 y_1 x_2 x_1^T & y_2 y_2 x_2 x_2^T & \dots & y_2 y_N x_2 x_N^T \\ \dots & \dots & \dots & \dots \\ y_N y_1 x_N x_1^T & y_N y_2 x_N x_2^T & \dots & y_N y_N x_N x_N^T \end{bmatrix}$$

Quadratic coefficients

In \mathbb{Z} space, the only thing you need:

$$\begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \dots & \dots & \dots & \dots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix}$$

Final stage:

$$g(x) = \text{sign}(z\theta + b) \quad \text{in terms of } K(-, -)$$

$$\text{where } \theta = \sum_{z_i \text{ in SV}} \alpha_i y_i z_i \rightarrow \quad g(x) = \text{sign}\left(\sum_{z_i \text{ in SV}} \alpha_i y_i z_i z + b\right)$$

$$g(x) = \text{sign}\left(\sum_{\alpha_i > 0} \alpha_i y_i K(x_i, x) + b\right)$$

$$\text{and } b: \quad b = y_j - \sum_{\alpha_i > 0, \alpha_j > 0} \alpha_i y_j K(x_i, x_j)$$

for any SV x_i and x_j

How do we know that the kernel is valid?

Does a z exist?

For a given $K(x, x')$ → We can check the validity

Three approaches:

1. By construction (Polynomial one)
2. Math properties (Mercer's condition)
3. Not worrying about validity of a kernel

Design your kernel

$K(x, x')$ is valid *iff*

1. It is symmetric $\rightarrow K(x, x') = K(x', x)$

2. The matrix \rightarrow

$$\begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \dots & \dots & \dots & \dots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix}$$

is positive-semi definite

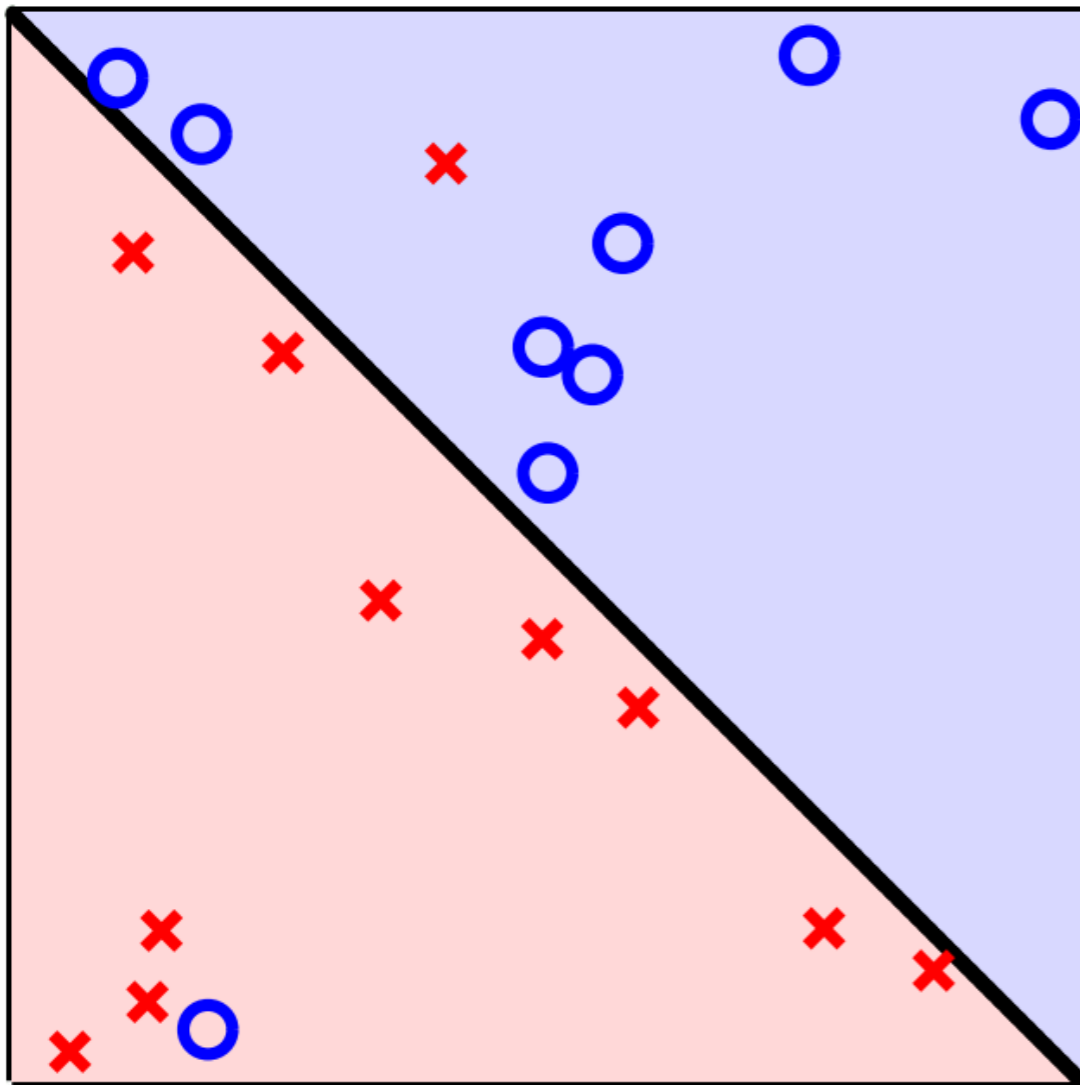
For any x_1, \dots, x_N (Mercer's condition)

Common Kernels

- **Linear** kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}'^T \mathbf{x}$
- **Polynomial** kernels $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}'^T \mathbf{x})^d$ for any $d > 0$
 - Contains all polynomial terms up to degree d
- **Gaussian** kernels $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2)$ for $\sigma > 0$
 - Infinite dimensional feature space

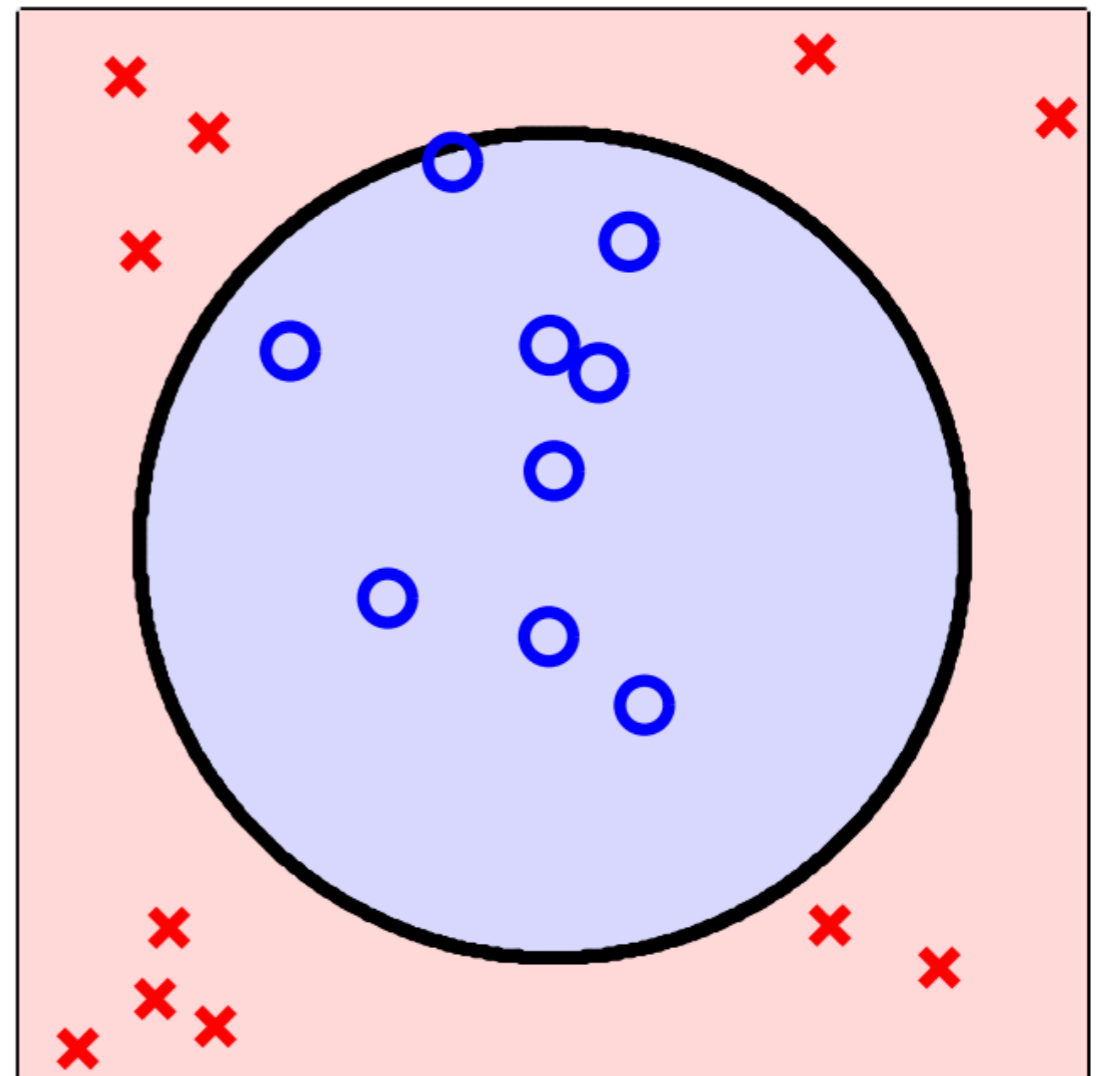
Soft SVM – Two types of non-separable

slightly:



Soft SVM will deal with this

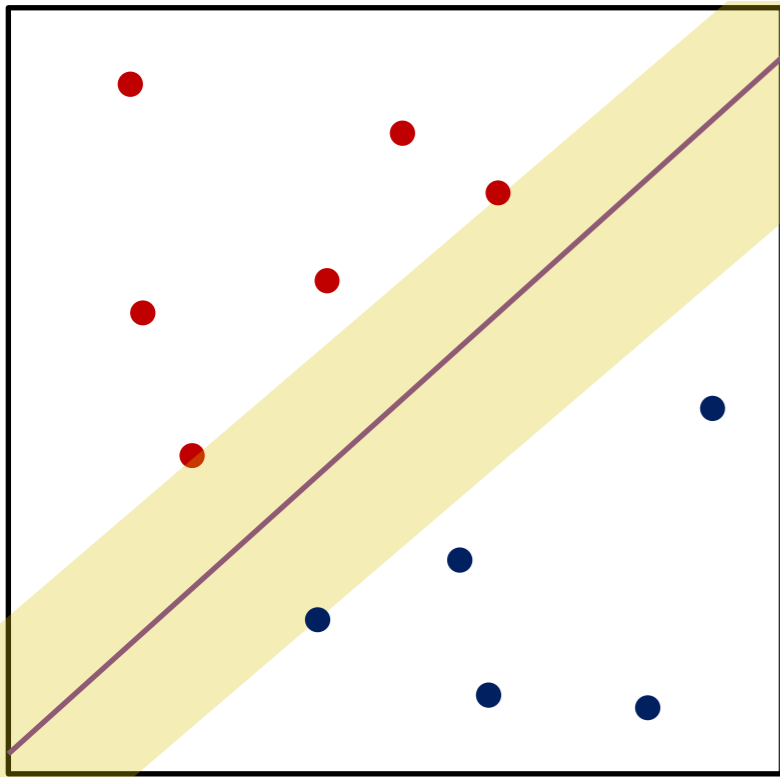
seriously:



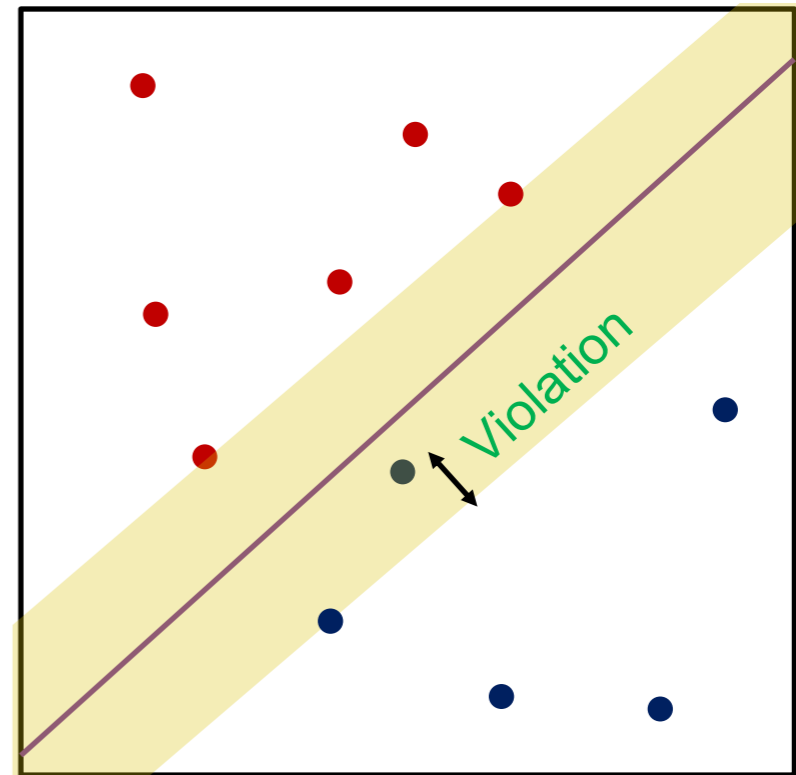
Kernel will deal with this

Error measure

Non-violated case:



Margin violation:



if $y_i(x_i\theta + b) > 1 \Rightarrow$ Non SV

Let's introduce a slack variable: $y_i(x_i\theta + b) \geq 1 - \xi_i$ $\xi_i \geq 0$

$$\text{Total violation} = \sum_{i=1}^N \xi_i$$

The new optimization

Minimize $\frac{1}{2} \theta \theta^T + C \sum_{i=1}^N \xi_i$

C will define the relative importance of the first or second term

C = 0 is equal to Hard SVM

s. t. $y_i(x_i \theta + b) \geq 1 - \xi_i$ for $i = 1, \dots, N$

and $\xi_i \geq 0$ for $i = 1, \dots, N$

$$\theta \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^N$$

The Lagrange formulation

Hard svm: Minimize $\frac{1}{2}\theta\theta^T$ s.t. $y_i(x_i\theta + b) \geq 1$

$$\mathcal{L}(\theta, b, \alpha) = \frac{1}{2}\theta\theta^T - \sum_{i=1}^N \alpha_i (y_i(x_i\theta + b) - 1)$$

Soft svm: Minimize $\frac{1}{2}\theta\theta^T + C \sum_{i=1}^N \xi_i$ s.t. $y_i(x_i\theta + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

$$\mathcal{L}(\theta, b, \xi, \alpha) = \frac{1}{2}\theta\theta^T + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i(x_i\theta + b) - 1 + \xi_i) - \sum_{i=1}^N \beta_i \xi_i$$

PLEASE do not scare, terms will be dropping fast

$$\mathcal{L}(\theta, b, \xi, \alpha) = \frac{1}{2} \theta \theta^T + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^N \beta_i \xi_i$$

Minimize w.r.t $\theta, b, \text{ and } \xi$ and maximize w.r.t $\alpha_i \geq 0 \text{ and } \beta_i \geq 0$

KKT condition for inequality constraints

Let's do the minimization:

$$\nabla_{\theta} \mathcal{L}(\theta, b, \xi, \alpha) = \theta - \sum_{i=1}^N \alpha_i y_i x_i = 0$$

If we substitute β_i up there, the whole formulation will get back to hard svm

$$\nabla_b \mathcal{L}(\theta, b, \xi, \alpha) = - \sum_{i=1}^N \alpha_i y_i = 0$$

$$\nabla_{\xi} \mathcal{L}(\theta, b, \xi, \alpha) = C - \alpha_i - \beta_i$$

The solution

$$\beta_i = C - \alpha_i$$

$$\beta_i \geq 0 \quad \rightarrow \quad C - \alpha_i \geq 0 \rightarrow 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, N$$

$$\text{Maximize } \mathcal{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j x_i x_j^T \quad \text{w.r.t } \alpha$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, N \quad \text{and} \quad \sum_{i=1}^N \alpha_i y_i = 0$$

$$\Rightarrow \theta = \sum_{i=1}^N \alpha_i y_i x_i \quad \text{will minimize} \quad \frac{1}{2} \theta \theta^T + C \sum_{i=1}^N \xi_i$$

Type of support vectors

We call the three points as **margin** support vectors

$$0 < \alpha_i < C$$

$$\beta_i = C - \alpha_i$$

$$y_i(x_i\theta + b) = 1 \Rightarrow \beta_i > 0 \Rightarrow \xi_i = 0 \text{ (KKT condition)}$$

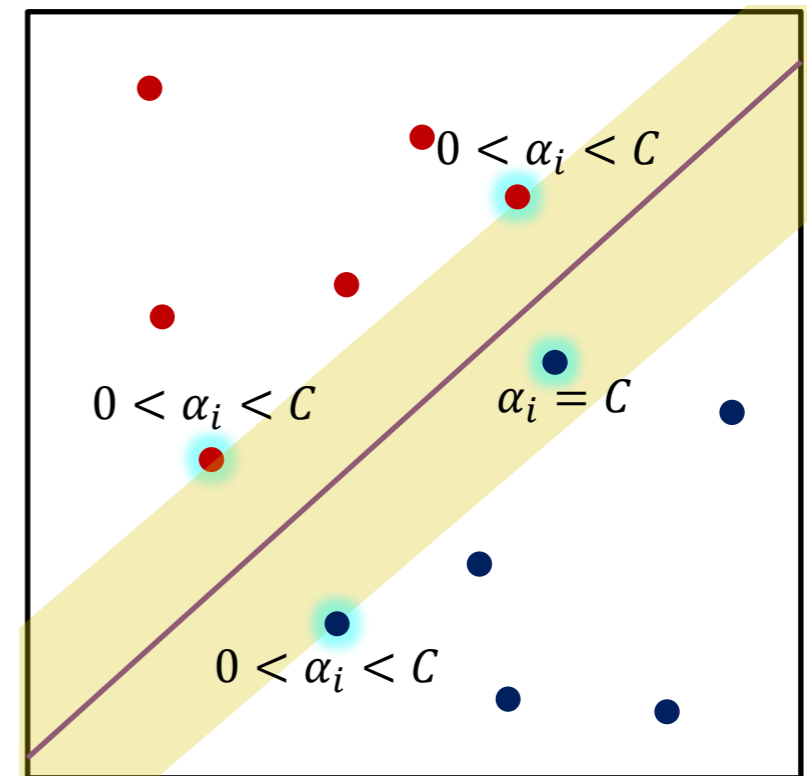
non-margin support vectors ($\alpha_i = C$)

$$\beta_i = 0 \Rightarrow \xi_i > 0 \text{ (KKT condition)}$$

$$y_i(x_i\theta + b) > 1 - \xi_i \text{ if } \xi_i > 0$$

$$y_i(x_i\theta + b) < 1$$

Any violating points become support vectors



$$\alpha_i = 0 \Rightarrow y_i(x_i\theta + b) > 1$$

Non SV

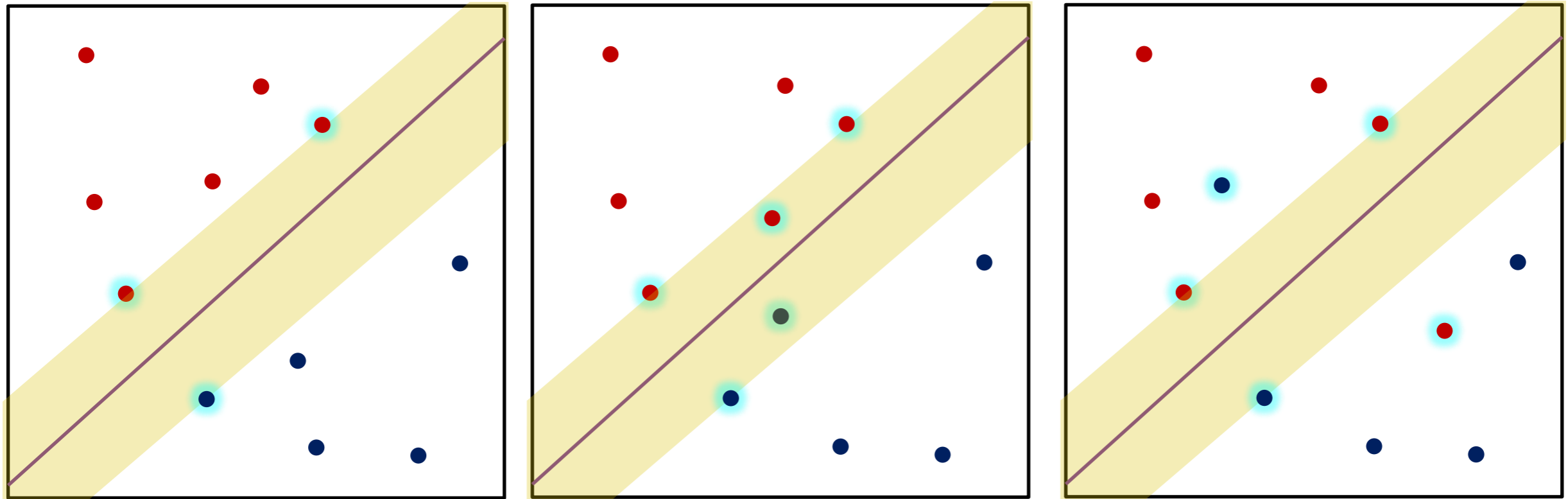
$$\alpha_i = C \Rightarrow y_i(x_i\theta + b) < 1$$

SV on the wrong side

$$0 < \alpha_i < C \Rightarrow y_i(x_i\theta + b) = 1$$

SV on the margin

How to choose C?



violating points become
support vectors

How to define the hyper-parameter C:

Cross Validation

Primal and Dual Forms of SVM

Primal version of classifier:

$$f(x_{test}) = x_{test}\theta + \theta_0$$

Dual version of classifier:

$$f(x_{test}) = \sum_{x_i \text{ in } SV} \alpha_i y_i x_i x_{test}^T + b$$

Kernel SVM: Summary

- Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space
- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space
- Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere – they are not tied to the SVM formalism