

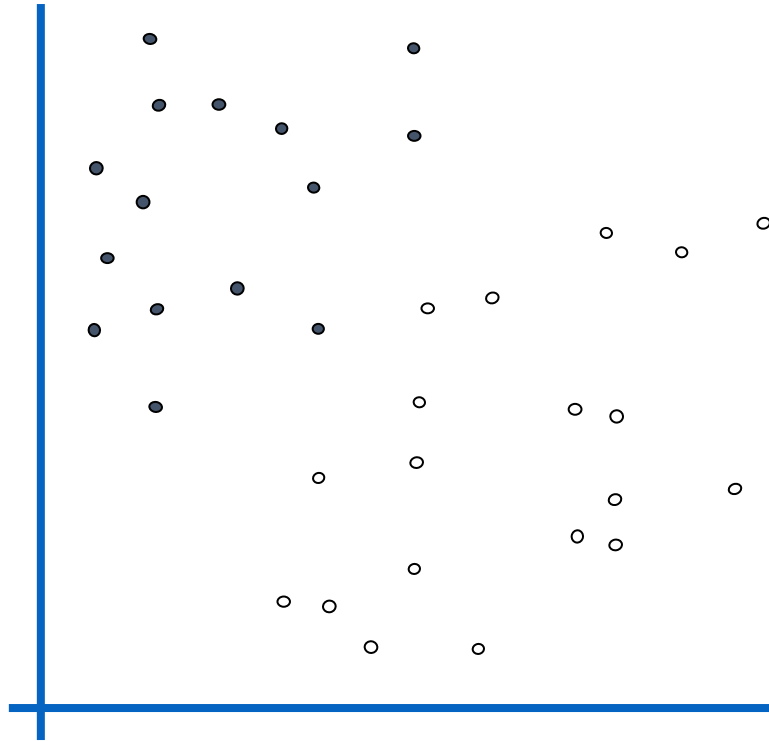
Support Vector Machines (SVM)

Support Vector Machine (SVM)

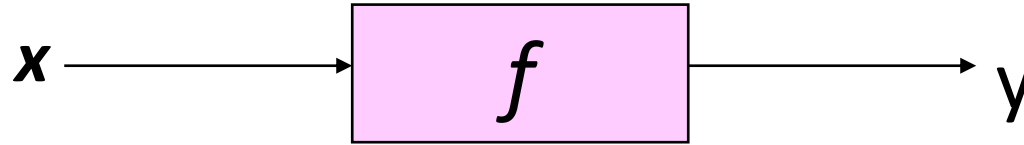
- **Support vector machines (SVM)** are supervised learning models that analyze data used for classification and regression analysis.
- They analyse the large amount of data to identify patterns from them.
- SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes.
- SVM models the situation creates a *feature space*, which is a finite-dimensional vector space, each dimension of which represents a in the data. For e.g., in the context of sentiment analysis of documents, each feature is a particular relevant word.
- It achieves this by creating a linear partition of the feature space into two categories.
- Based on the features of the new unseen datapoint (e.g., documents), SVM assigns the new datapoint to one of the two binary groups, "above" or "below" the separation plane. This will lead to a categorising (e.g. positive, r negative sentiment group).

Linear Classification

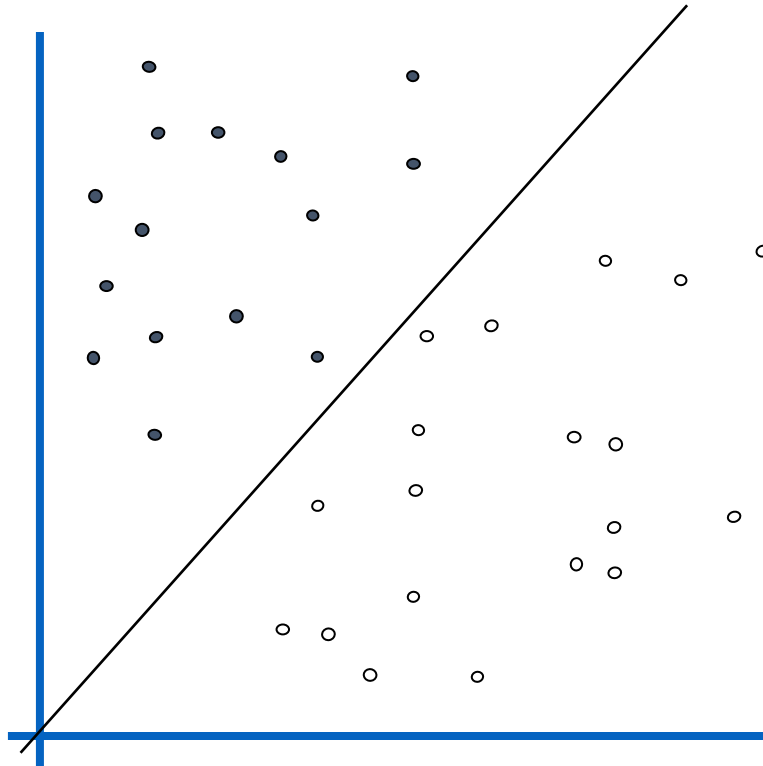
- Class1
- Class2



Linear Classification



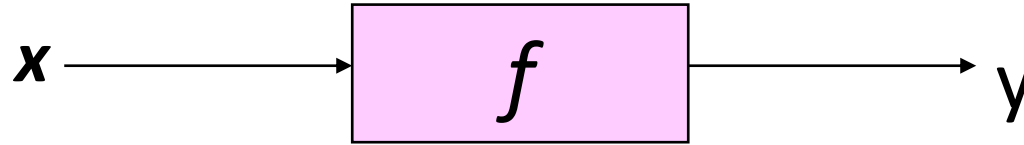
- Class1
- Class2



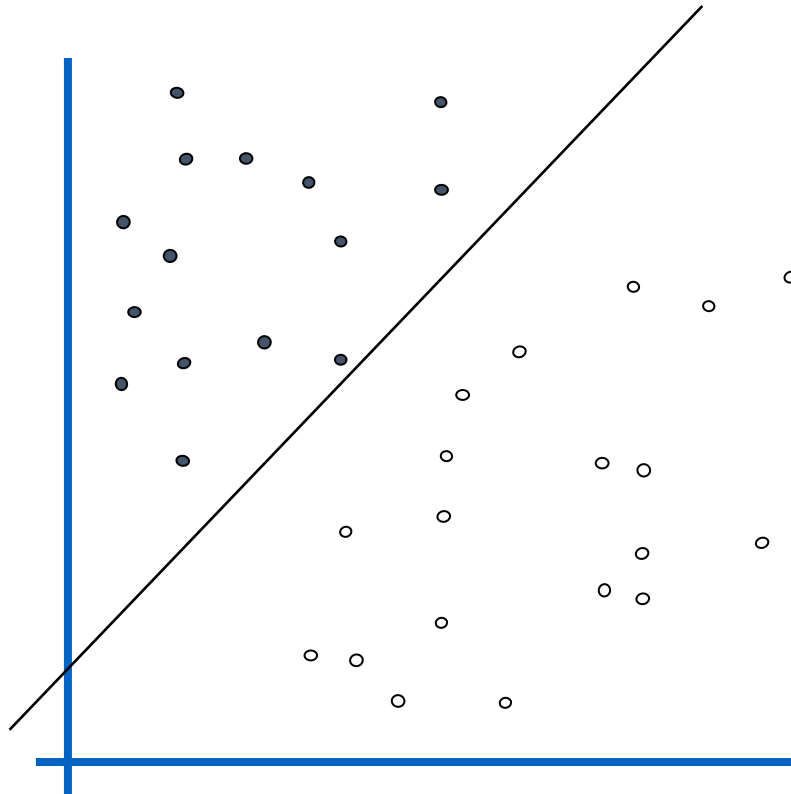
$$f(x, w, b) = w^T x + b$$

w^T = is the weight vector
 x = The input feature vector.
 b = is the bias term

Linear Classification



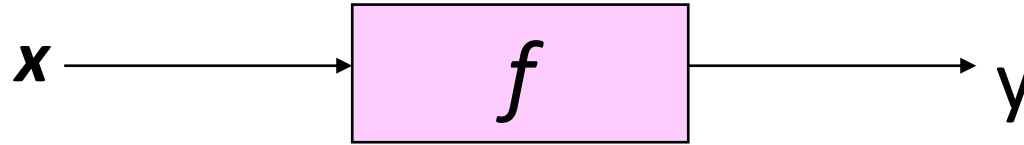
- Class1
- Class2



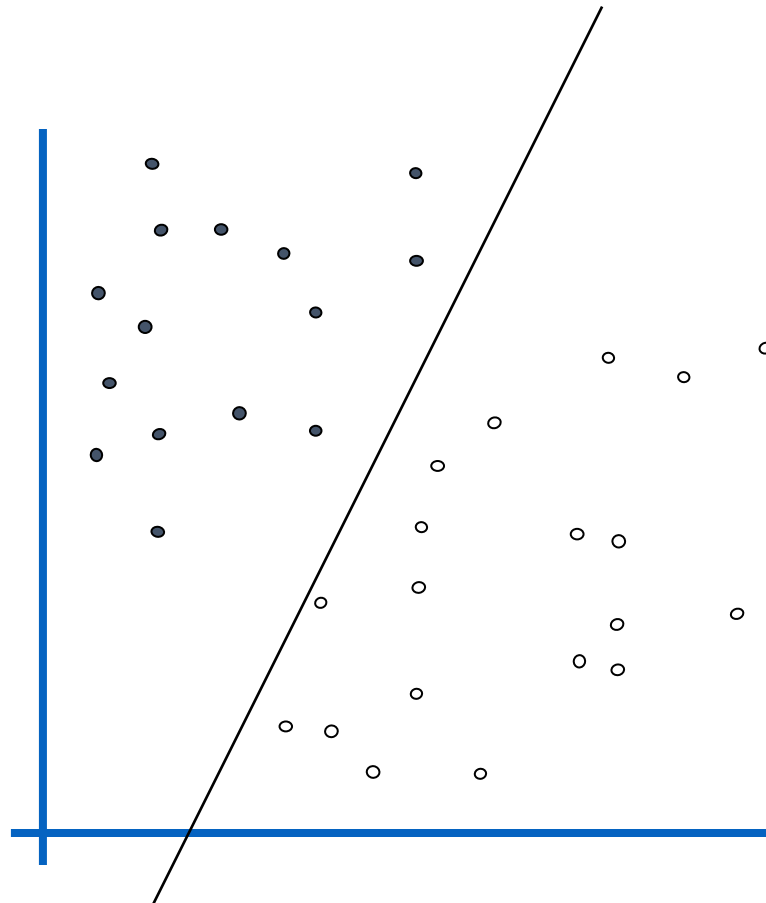
$$f(x, w, b) = w^T x + b$$

w^T = is the weight vector
 x = The input feature vector.
 b = is the bias term

Linear Classification



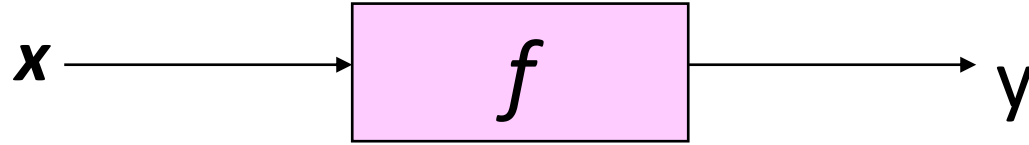
- Class1
- Class2



$$f(x, w, b) = w^T x + b$$

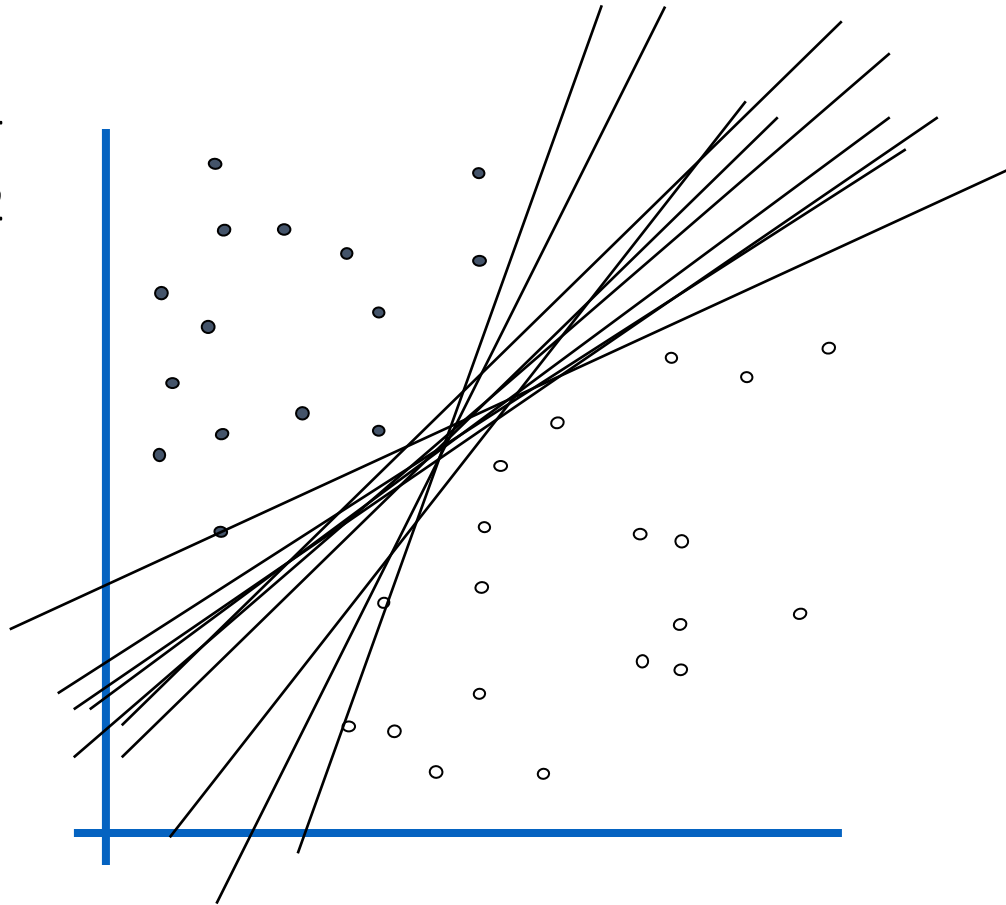
w^T = is the weight vector
 x = The input feature vector.
 b = is the bias term

Linear Classification



- Class1
- Class2

Lots of possible solutions, but which one is the optimal.

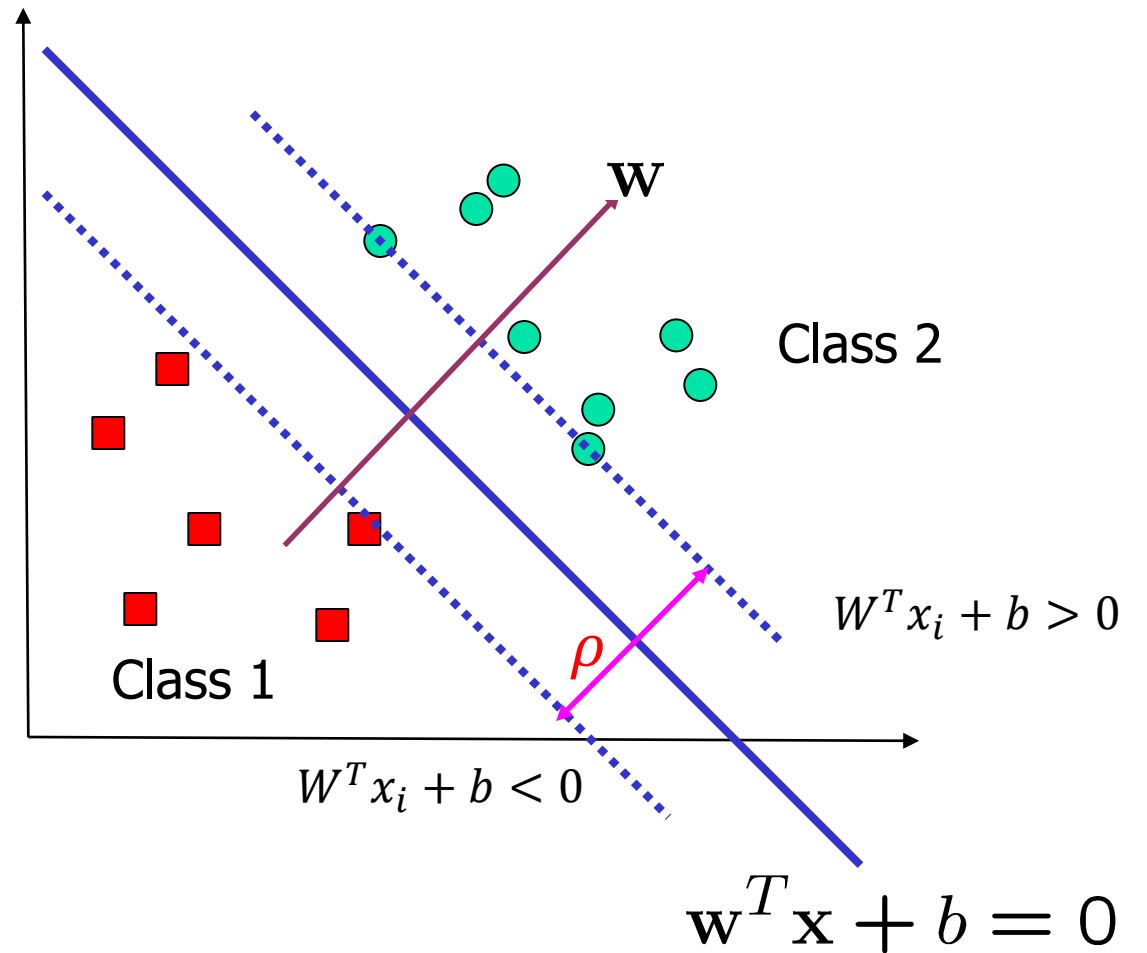


SVM Hyperplane

- This linear equation in 3 dimensions represents a plane. Whereas if I increase the dimension
- where the dimensionality of the feature vector is more than 3, this linear equation actually represents what is called an hyperplane.
- w is a vector, which is perpendicular to that hyperplane. Therefore, w represents the orientation of the hyperplane.
- This vector w represents the orientation of the hyperplane.
- b represents the position of the hyperplane in the d dimensional space, where d is the number of features in the data.

Linear SVM Mathematically

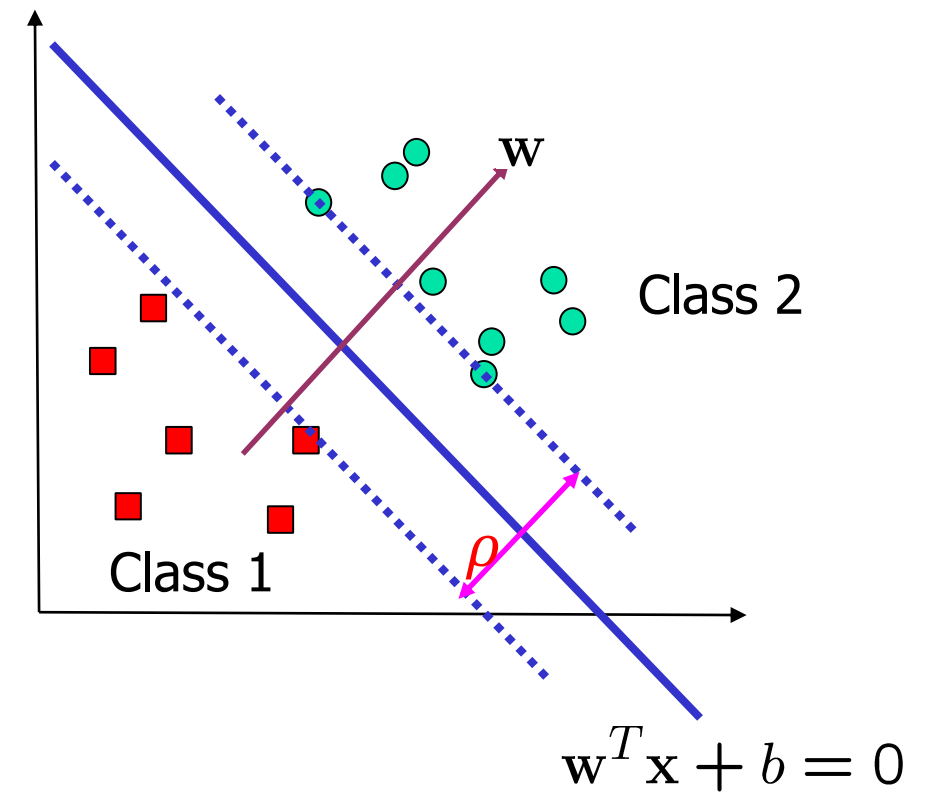
- Support Vector Machine (SVM) finds an optimal solution:
 - Maximizes the distance (ρ) between the hyperplane and the “difficult points” close to decision boundary
 - One intuition: if there are no points near the decision hyperplane, then there are no very uncertain classification decisions.



If $w^T x + b > 0 \rightarrow x \in C_1$, and if $w^T x + b < 0 \rightarrow x \in C_2$

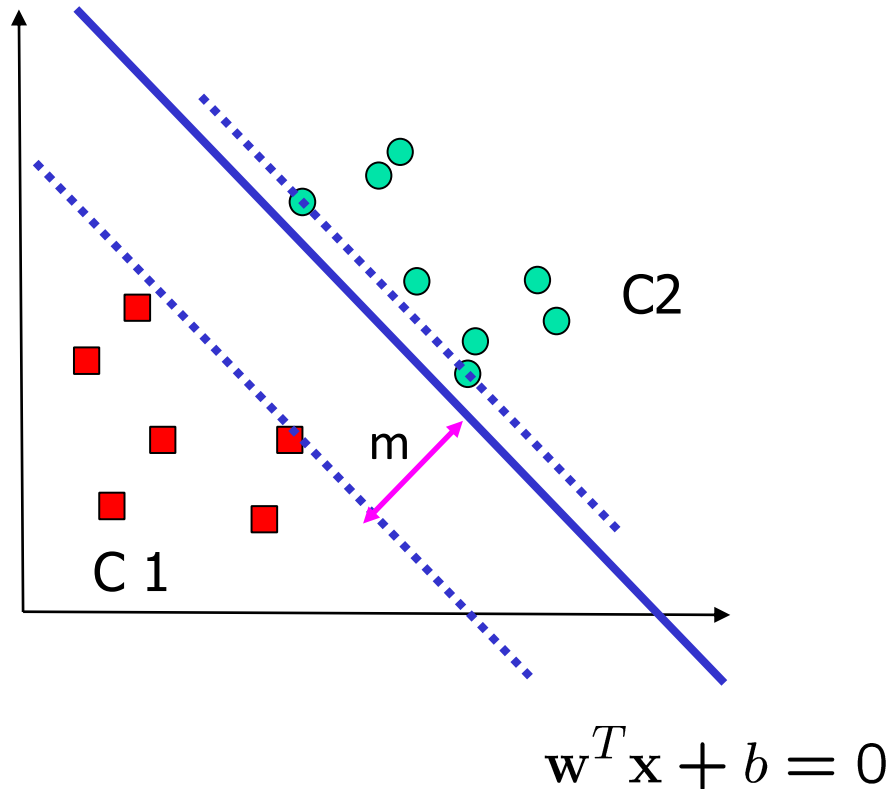
Linear SVM Mathematically

- What is the best value of W^T , and b ?, and How to find these values?
 - Start with initial values of W^T and b .
 - During the training process, SVM will use the training dataset to modify the values of W^T and b iteratively.
 - Assume x_i is belong to C_i , during the training, the orientation and the position of the hyperplane is changed to classify this training data point correctly.



Linear SVM Mathematically

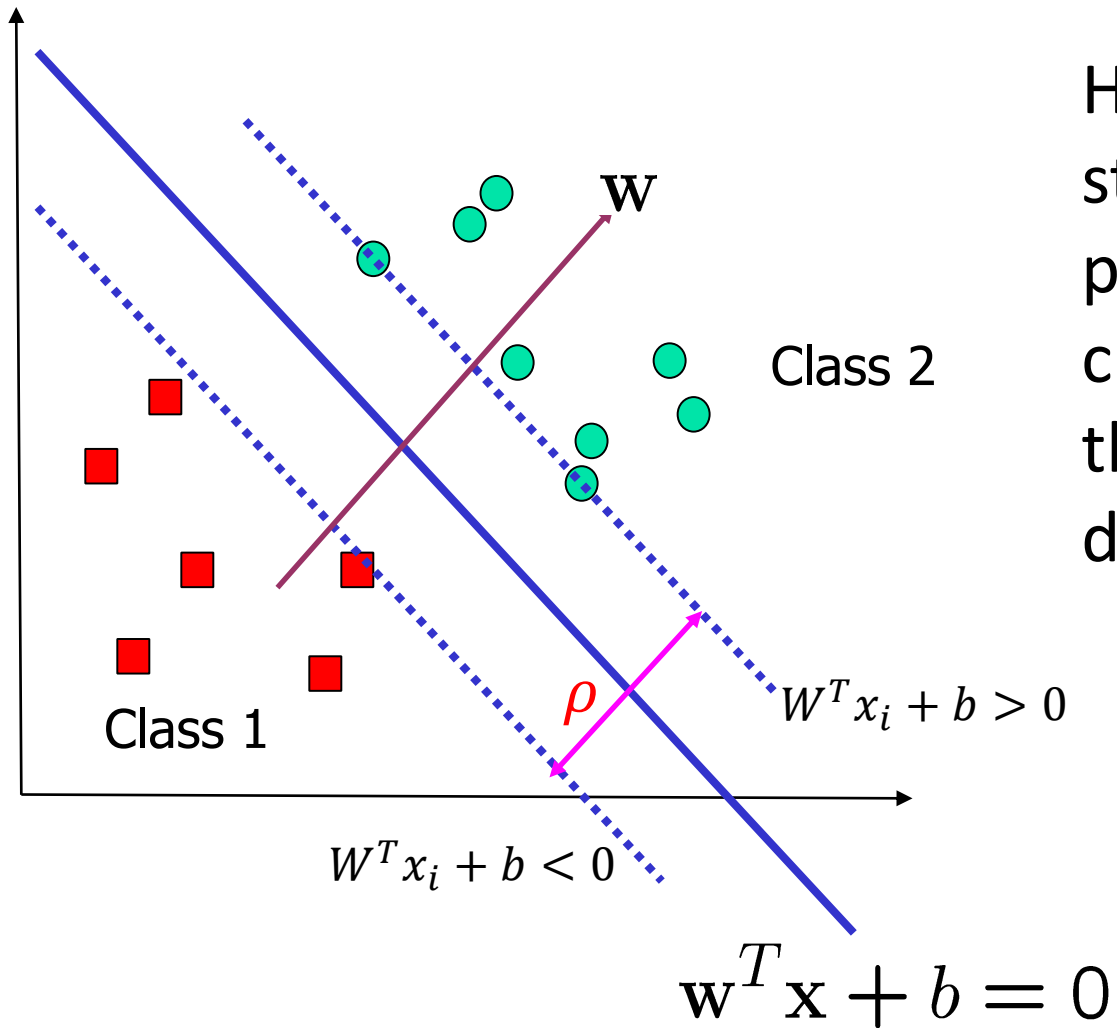
What is the desirable position and orientation of the hyperplane?



Example 1

- A classifier of this nature is not desirable!
- It gives a large bias in favour of class C1, whereas it puts a penalty against class C2.
- The classifier gives the entire space m to class C1.
- The desirable position is somewhere between the two classes. So the training data vectors of both class C1 and C2 are equally apart from decision hyperplane.

Linear SVM Mathematically



Having large boundary will give more stability to the classifier. So no that point cross the boundary and miss classified because of any disturbance that is caused by noisy training datapoints.

Linear SVM Mathematically

for x_i

$$W^T x_i + b > 0 \rightarrow x_i \in C_1$$

$$W^T x_i + b < 0 \rightarrow x_i \in C_2$$

x_i, y_i where $y_i = \pm 1$

$$y_i(W^T x_i + b) > 0$$

For unknown datapoint vector P

we have to find the sign of the results of $W^T \cdot p + b$

The distance between point p and the hyperplane is :

$$\frac{W^T \cdot p + b}{\|W^T\|}$$

Linear SVM Mathematically

Let us make some modification on the classifier:

$$W^T x_i + b > \gamma \rightarrow x_i \in C_1$$

$$\frac{W^T \cdot x_i + b}{\|W^T\|} > \gamma \Rightarrow W^T x_i + b > \gamma \cdot \|W^T\| = 1$$

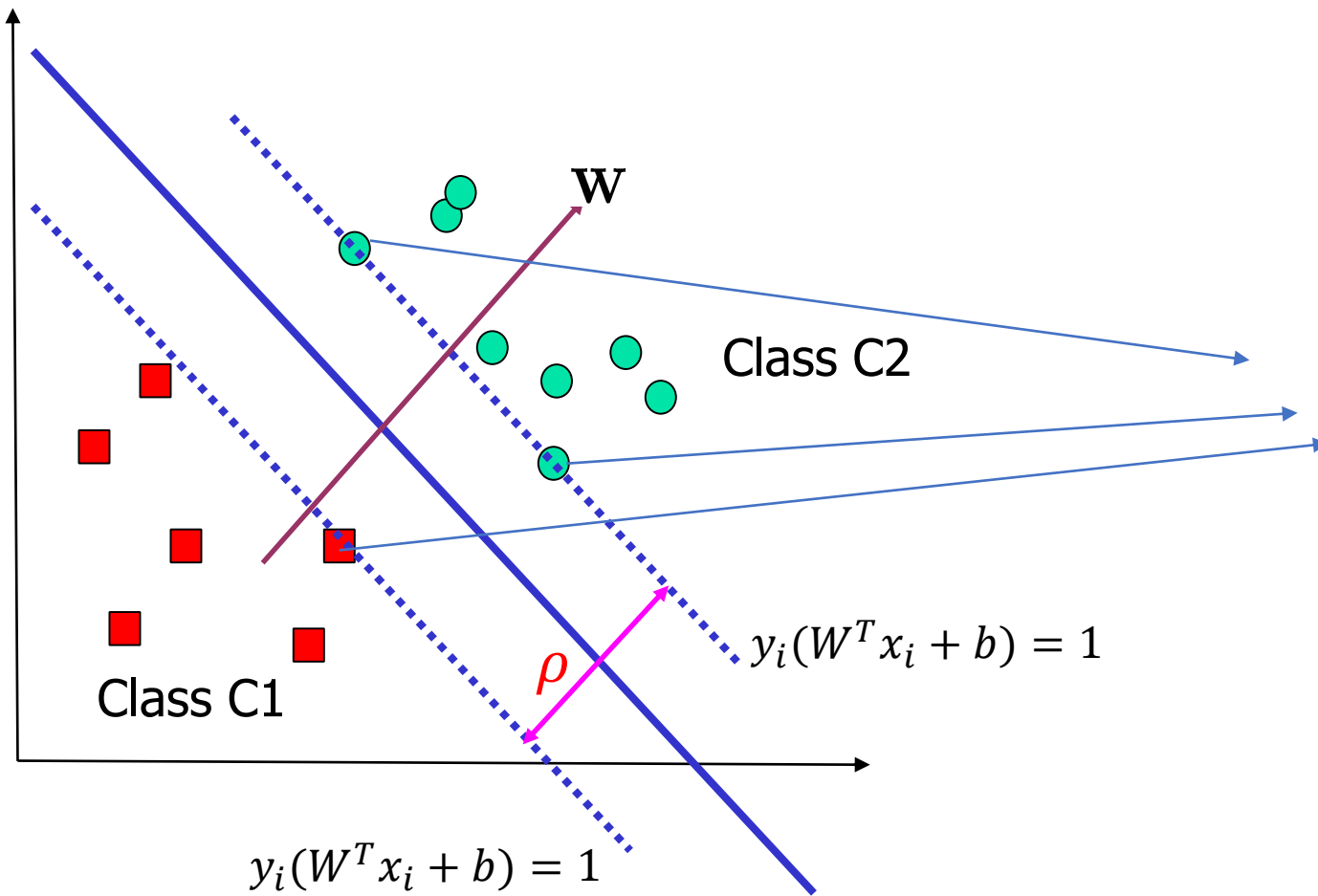
$$W^T x_i + b \geq 1 \text{ if } x_i \in C_1$$

$$W^T x_i + b \leq -1 \text{ if } x_i \in C_2$$

$$y_i(W^T x_i + b) > 1$$

$$y_i(W^T x_i + b) = 1 \text{ if } x_i \text{ is a support vector}$$

Linear SVM Mathematically



Support Vectors
are those
datapoints that
the margin
pushes up
against

Linear SVM Mathematically

- The margin has to be as maximum as possible.
- This can be done based on this equation $\frac{W^T \cdot x_i + b}{\|W\|} > \gamma$, by minimising of the mod of $\|W^T\|$ and maximising the value the bias b
- This is the simplest kind of SVM (Called an LSVM)

Learning the Maximum Margin Classifier

$\phi(W) = W^T W = \frac{1}{2}(W.W) \rightarrow$ we want to minimise $\phi(W)$

With the constraint $y_i(W^T x_i + b) = 1$

This constrained optimisation problem to we have to solve using Lagrangian multiplier.

$$L(W, b) = \frac{1}{2}(W.W) - \sum \alpha_i (y_i(W^T x_i + b) - 1)$$

α_i is Lagrangian multiplier

Learning the Maximum Margin Classifier

- Calculate the derivative of L with respect to b .

$$L(W, b)$$

$$\frac{\partial L}{\partial b}$$

$$\frac{\partial L}{\partial b}$$

There exist algorithms for finding such constrained optimization problem

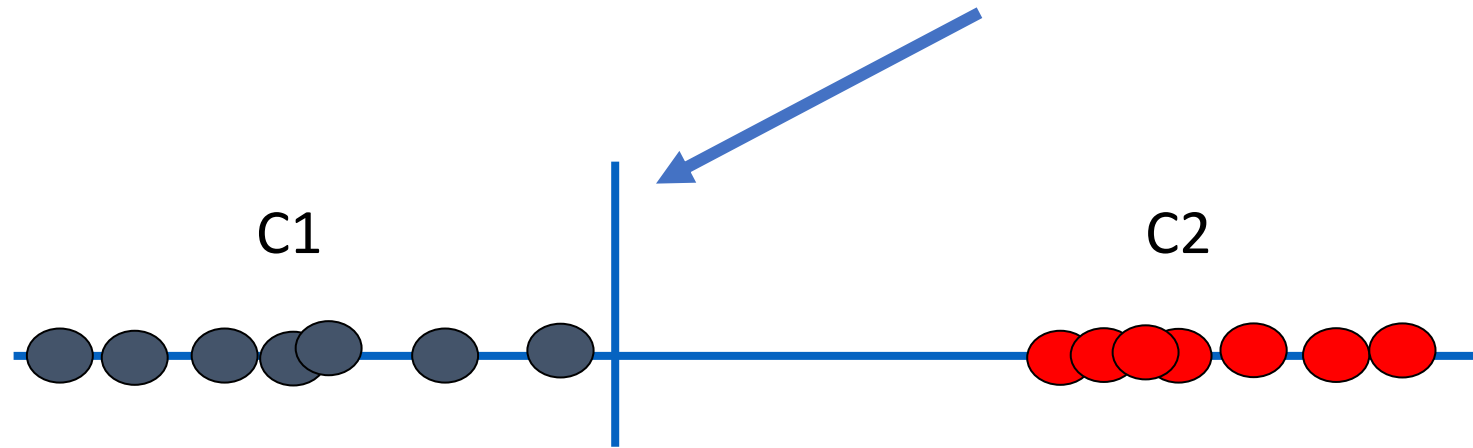
(But they are very fiddly...you probably don't want to write one yourself)

$$\sum \alpha_i$$

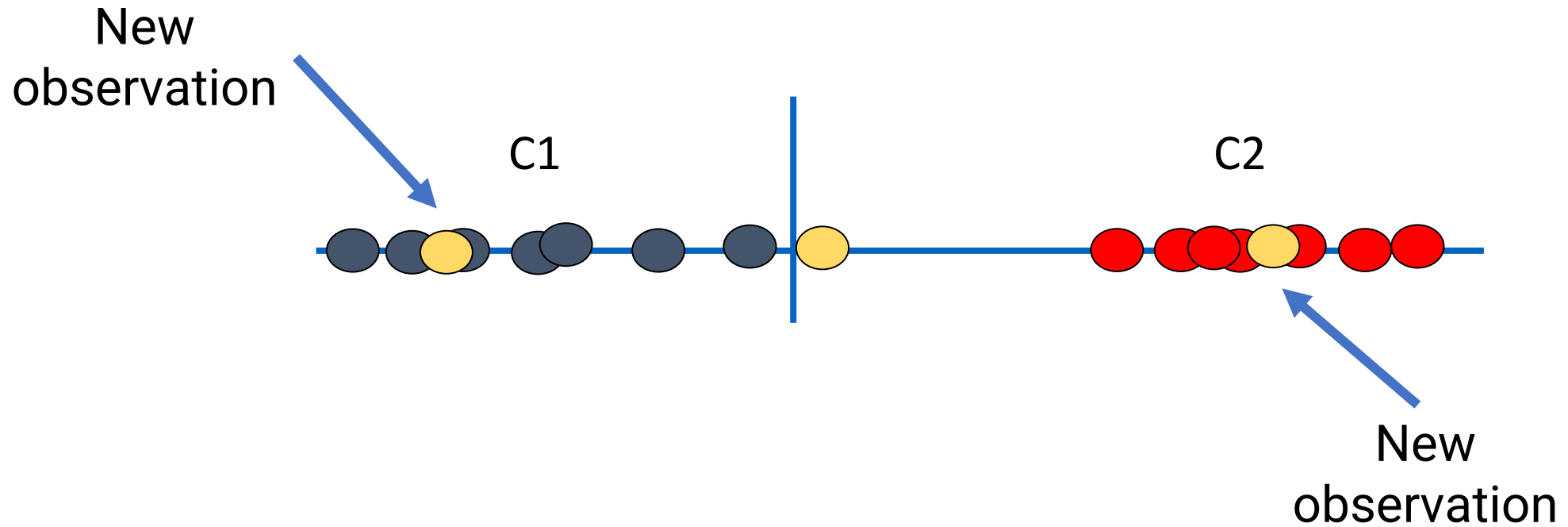
$$0$$

One dimension data set

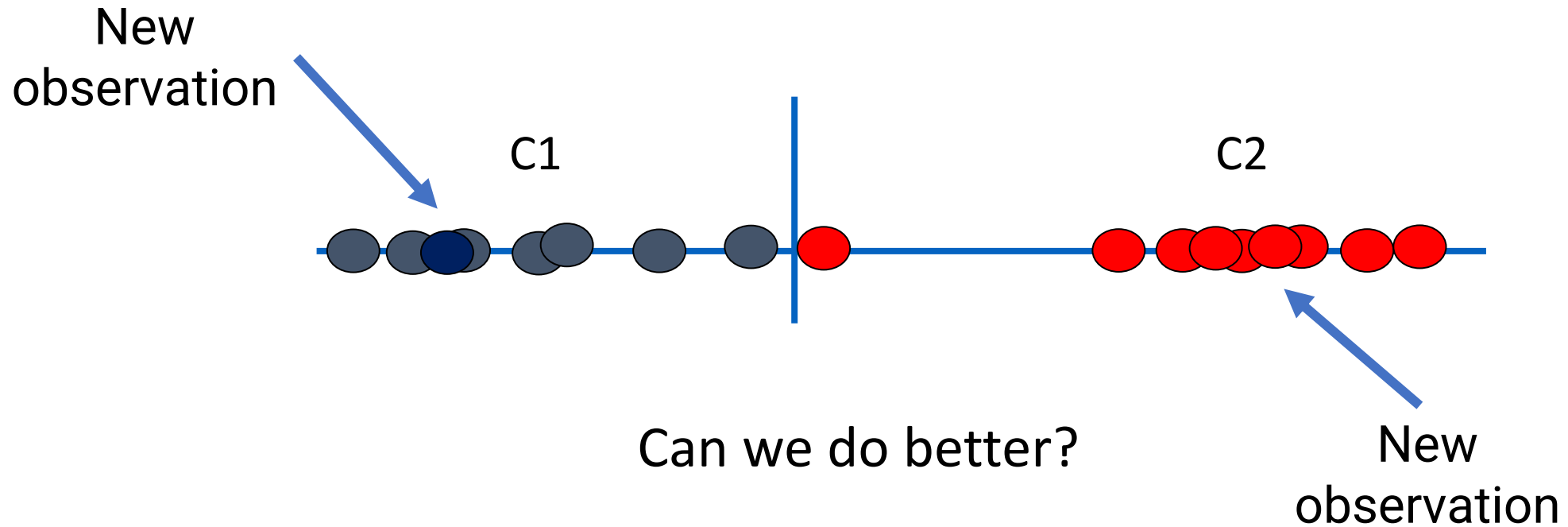
Based on the training data, we can pick a threshold



One dimension data set

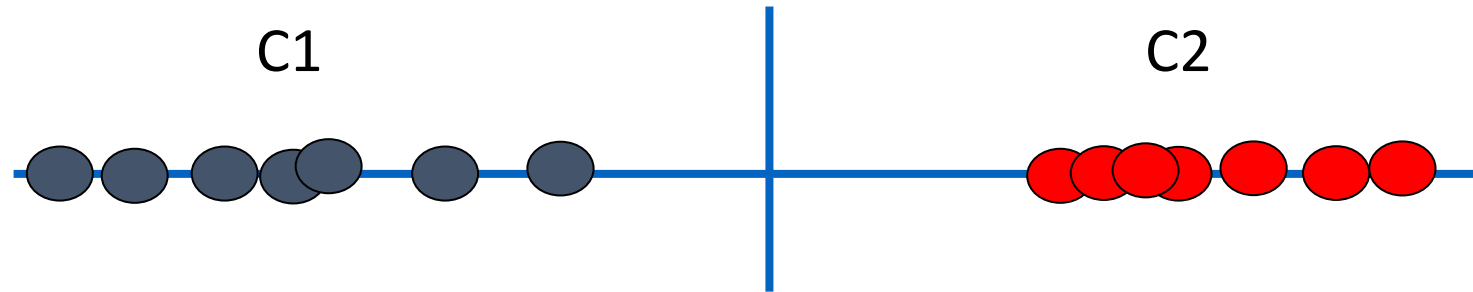


One dimension data set



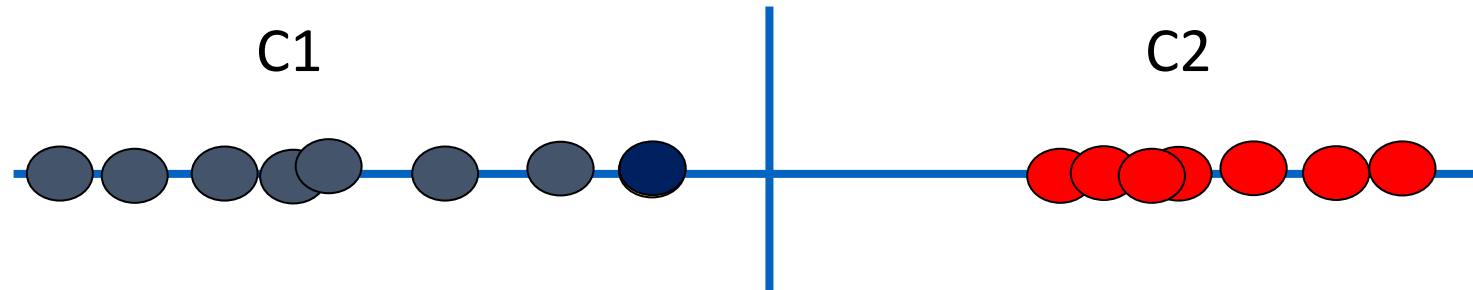
One dimension data set

We can focus on the edges of each class and place the boundary in the middle



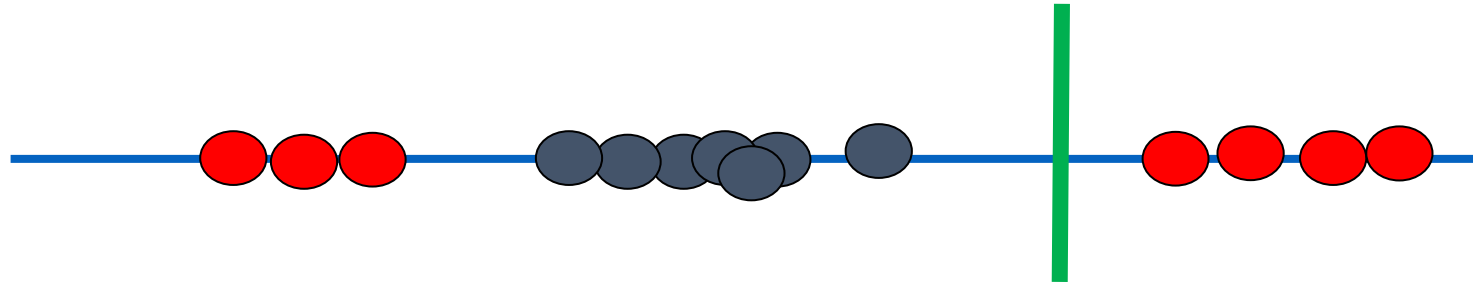
One dimension data set

We can focus on the edges of each class and place the boundary in the middle

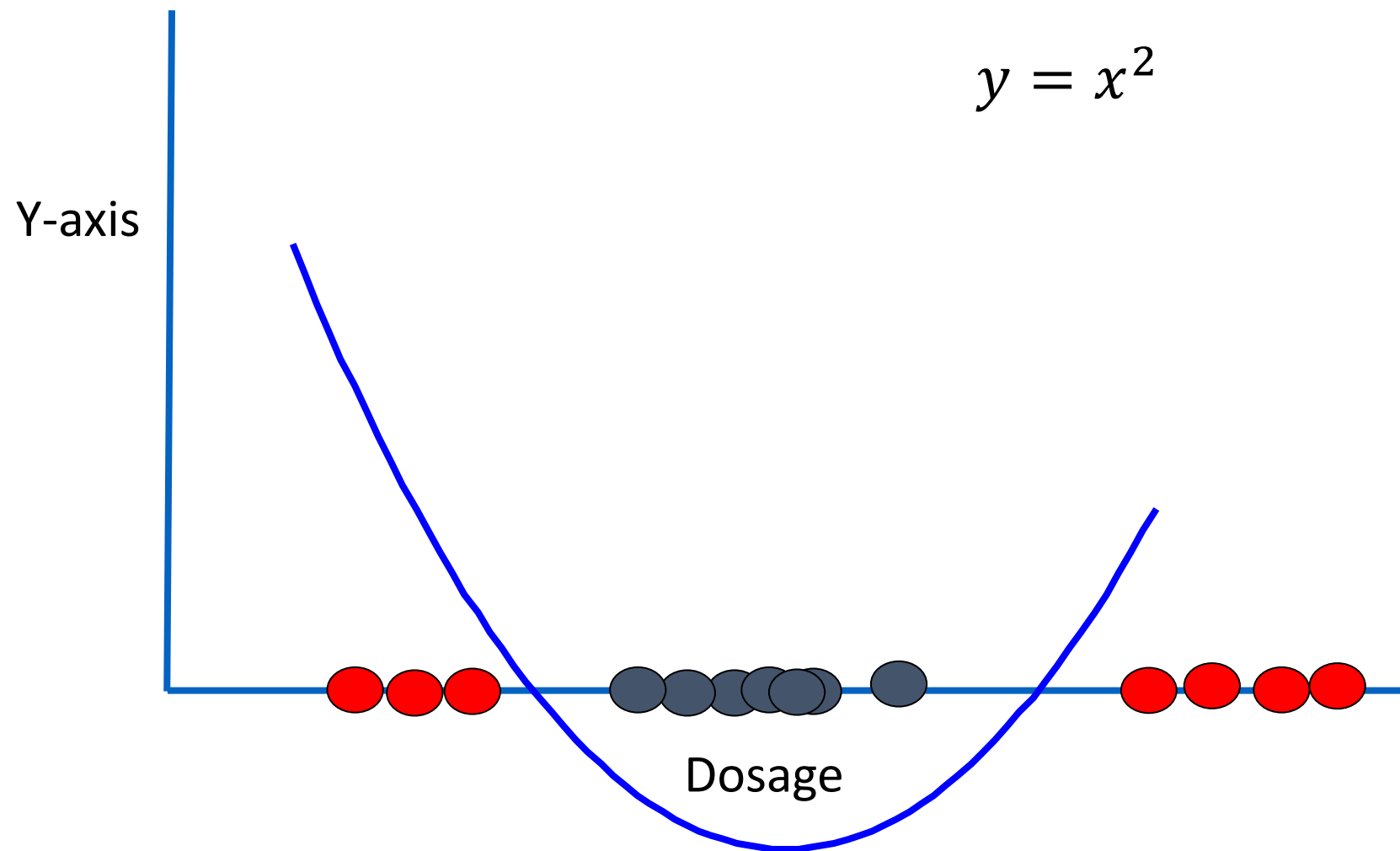


One dimension data set

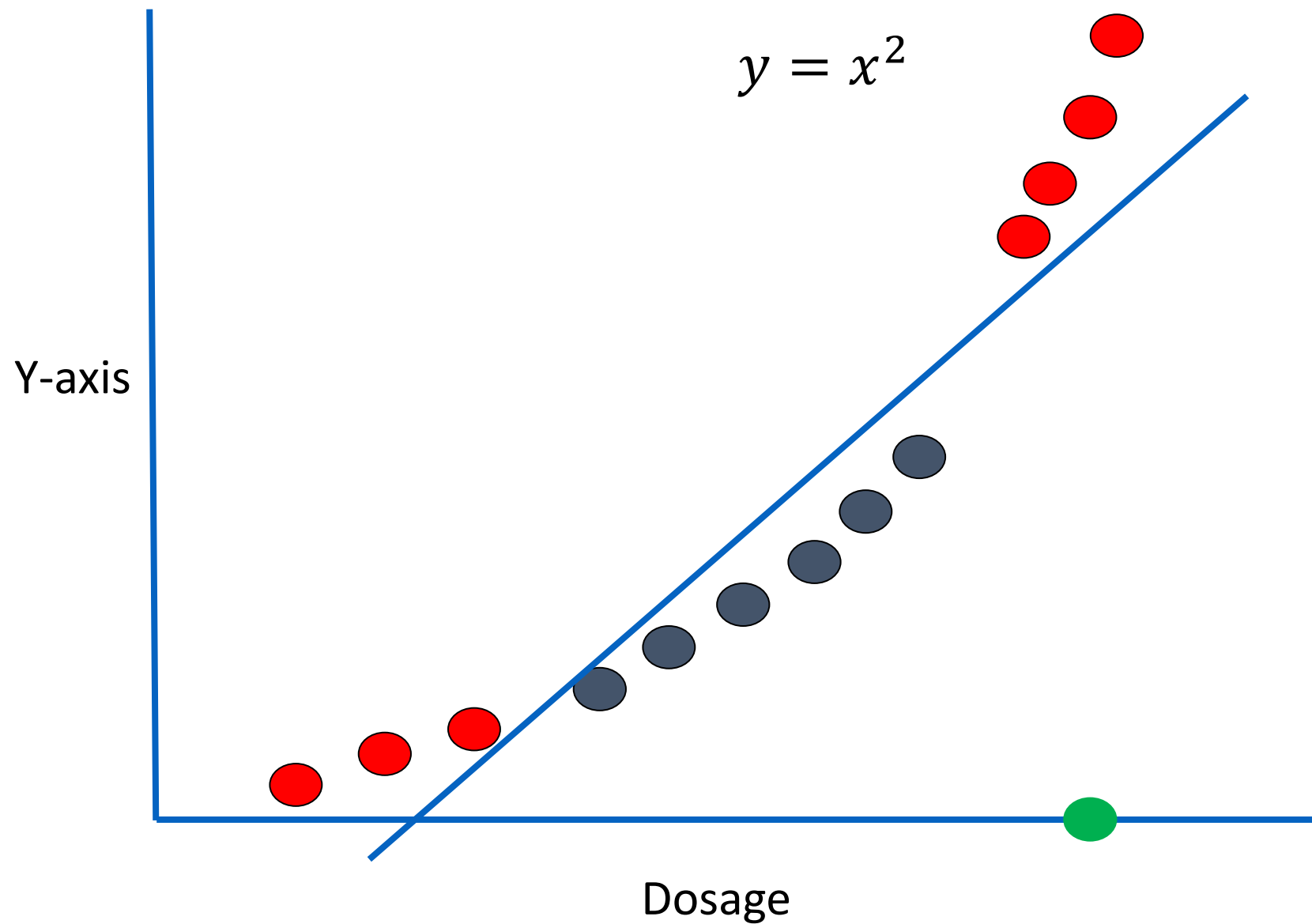
● c2
● c1



Linear SVM can't handle this problem



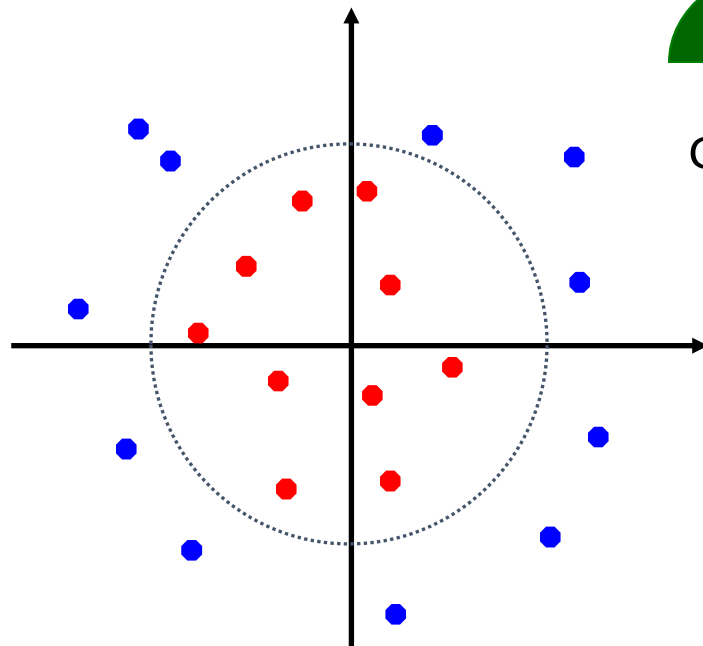
Non-linear SVMs



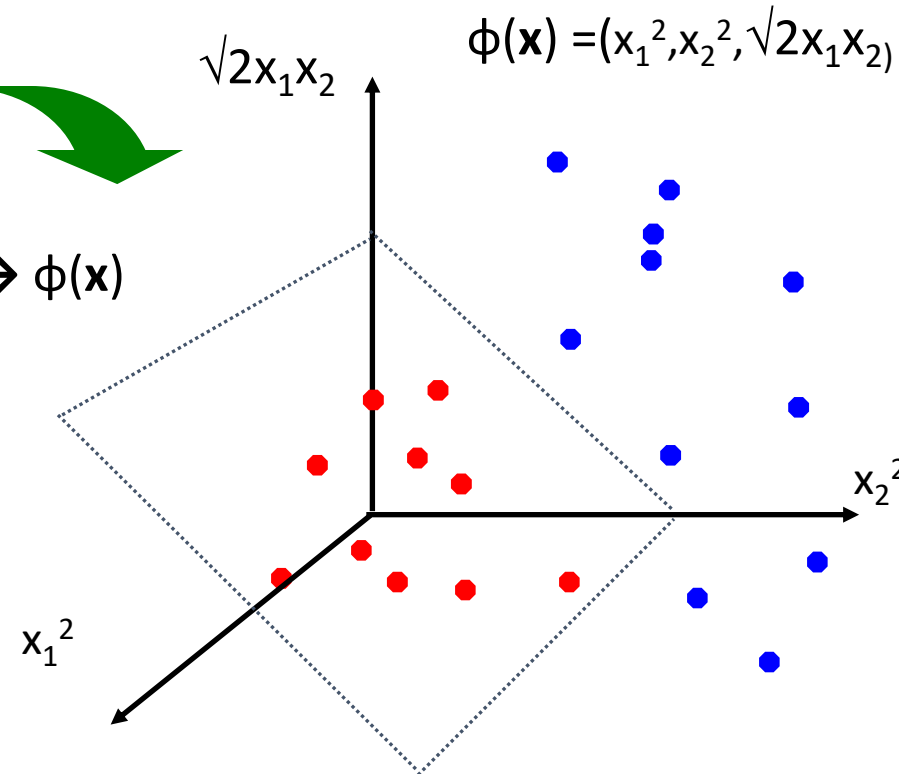
Non-linear SVMs (The “Kernel Trick”)

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

$$\mathbf{x} = (x_1, x_2)$$



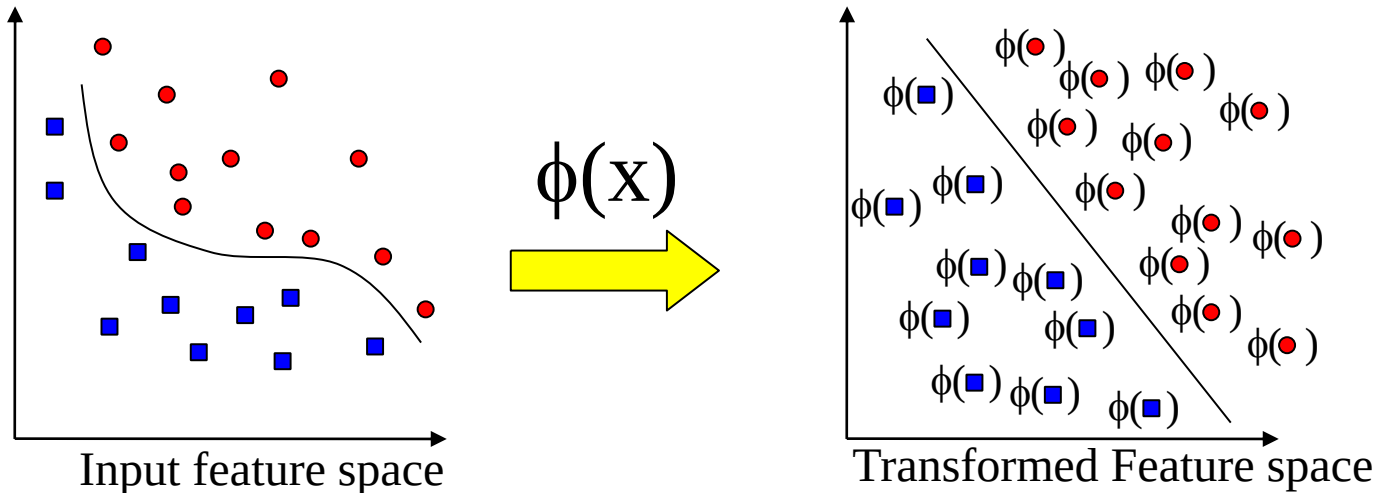
$$\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$$



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

Transformation to Higher dimensional Space

- Possible problem of the transformation
 - High computation burden due to high-dimensionality and hard to get a good estimate
- SVM solves these two issues simultaneously
 - “Kernel tricks” for efficient computation
 - Minimize $\|\mathbf{w}\|^2$ can lead to a “good” classifier



The “Kernel Trick”

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$$

- A *kernel function* is a function that is equivalent to an inner product in some feature space.
- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} = \\ &= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}] = \\ &= \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j), \quad \text{where } \boldsymbol{\phi}(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2] \end{aligned}$$

- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\phi}(\mathbf{x})$ explicitly).

Examples of Kernel Functions

- Polynomial kernel with degree d

- $K(x_i, x_j) = (x_i \times x_j + r)^d$,

x_i, x_j are two pair data points, r = Polynomial coefficient, and d is the Polynomial degree. (r , and d are determined using cross validation)

- Radial basis function kernel with width σ

$$K(x_i, x_j) = e^{\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}$$

- Research on different kernel functions in different applications is very active

Non-linear SVMs Mathematically

- Dual problem formulation:

Find $\alpha_1 \dots \alpha_n$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and

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

(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

$$f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

- Optimization techniques for finding α_i 's remain the same!

Example Transformation

- Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

- Define the kernel function $K(\mathbf{x}, \mathbf{y})$ as

$$\begin{aligned} \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- The inner product $\phi(.)\phi(.)$ can be computed by K without going through the map $\phi(.)$ explicitly!!!

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\begin{aligned} \max. \quad W(\boldsymbol{\alpha}) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to } C &\geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

With kernel
function

$$\begin{aligned} \max. \quad W(\boldsymbol{\alpha}) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to } C &\geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

Example

- Suppose we have 5 1D data points
 - $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$
 $y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1$
- We use the polynomial kernel of degree 2
 - $K(x,y) = (xy+1)^2$
 - C is set to 100
- We first find α_i ($i=1, \dots, 5$) by

$$\begin{aligned} \max. \quad & \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2 \\ \text{subject to} \quad & 100 \geq \alpha_i \geq 0, \sum_{i=1}^5 \alpha_i y_i = 0 \end{aligned}$$

Example

- By using a QP solver, we get
 $\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$
 - Verify (at home) that the constraints are indeed satisfied
 - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$

- The discriminant function is

$$\begin{aligned} f(y) &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- b is recovered by solving $f(2)=1$ or by $f(5)=-1$ or by $f(6)=1$, as x_2, x_4, x_5 lie on and all give $b=9$

$$y_i(\mathbf{w}^T \phi(z) + b) = 1$$

➡ $f(y) = 0.6667x^2 - 5.333x + 9$

Choosing the Kernel Function

- Probably the trickiest part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, ...)
- There is even research to estimate the kernel matrix from available information
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a Binary classification problem).
- Extend to output arity N , **learn N SVM's**
 - SVM 1 learns “Output==1” vs “Output != 1”
 - SVM 2 learns “Output==2” vs “Output != 2”
 - $\vdots$
 - SVM N learns “Output== N ” vs “Output != N ”
- SVM can also be extended to compute any real value functions.

Summary

- Weaknesses
 - Training (and Testing) is quite slow compared to other learning algorithms
 - Because of Constrained Quadratic Programming
 - Essentially a binary classifier
 - However, there are some tricks to evade this.
 - Very sensitive to noise
 - A few off-data points can completely throw off the algorithm
 - Biggest Drawback: The choice of Kernel function.
 - There is no “set-in-stone” theory for choosing a kernel function for any given problem (still in research...)
 - Once a kernel function is chosen, there is only ONE modifiable parameter.

Summary

- Strengths
 - Training is relatively easy
 - SVM doesn't suffer from "curse of dimensionality".
 - Less prone to overfitting
 - Simple, easy to understand geometric interpretation.
 - No large networks to mess around with.

References

- The VC/SRM/SVM Bible:

Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998