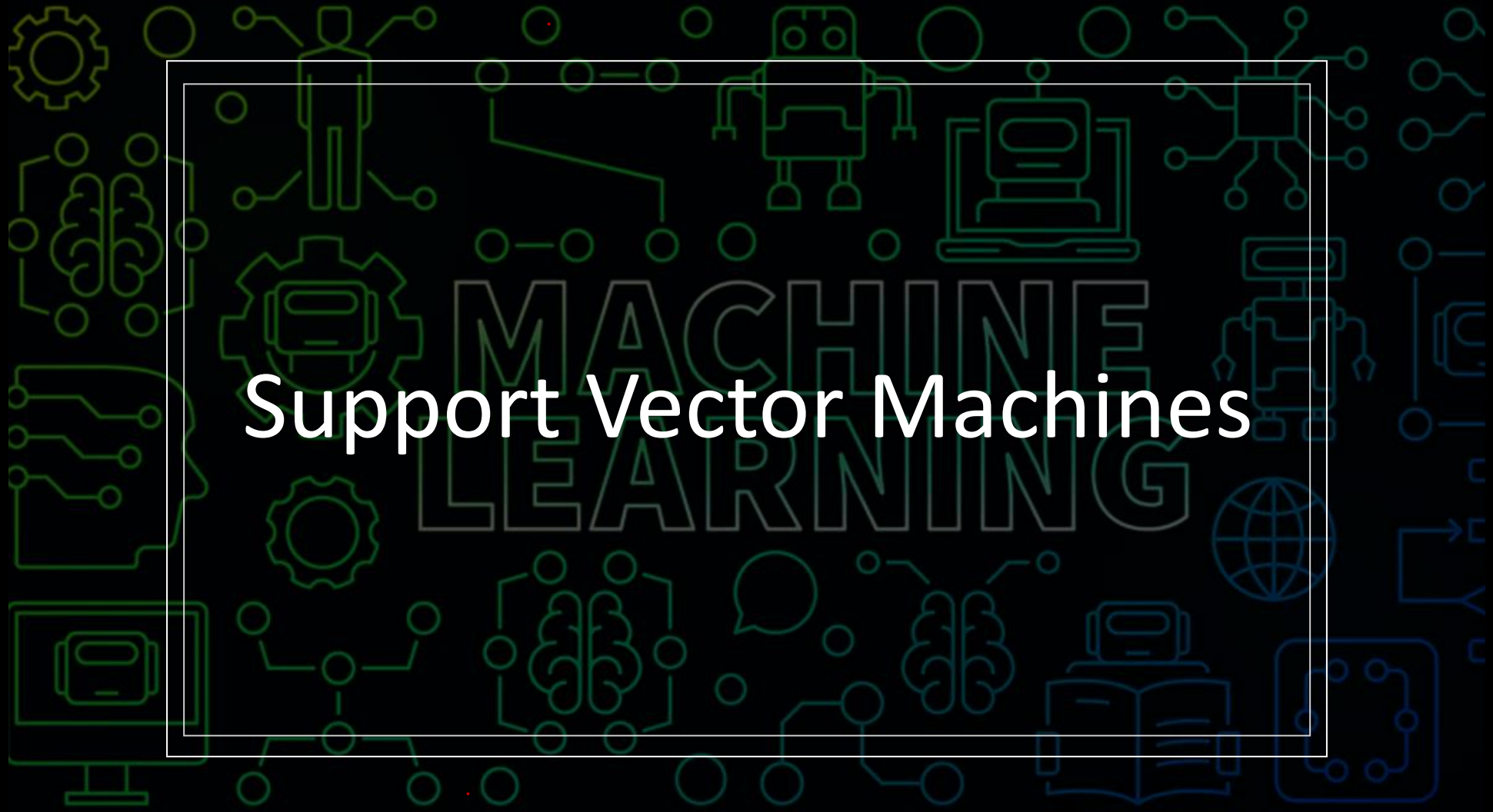
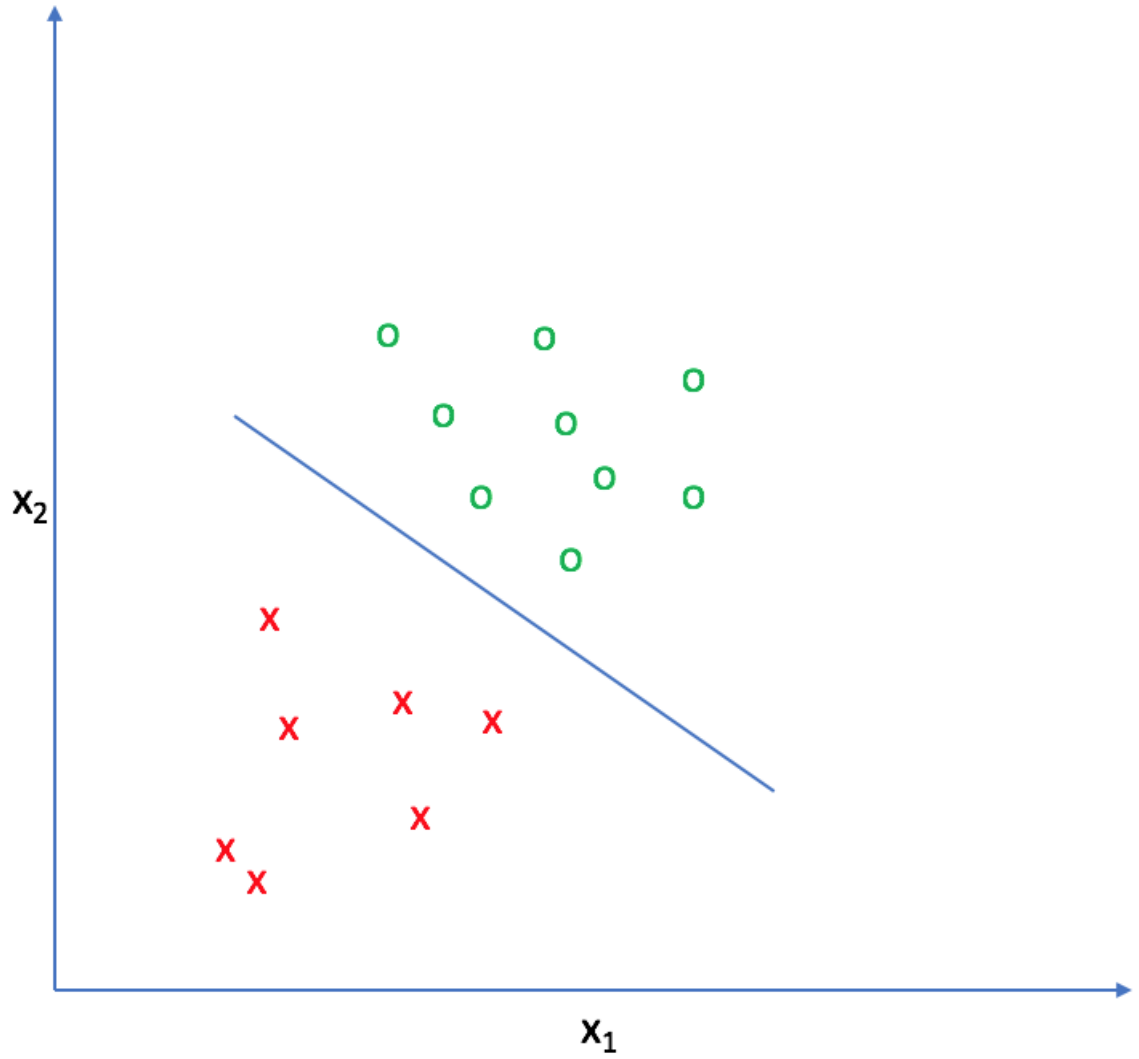
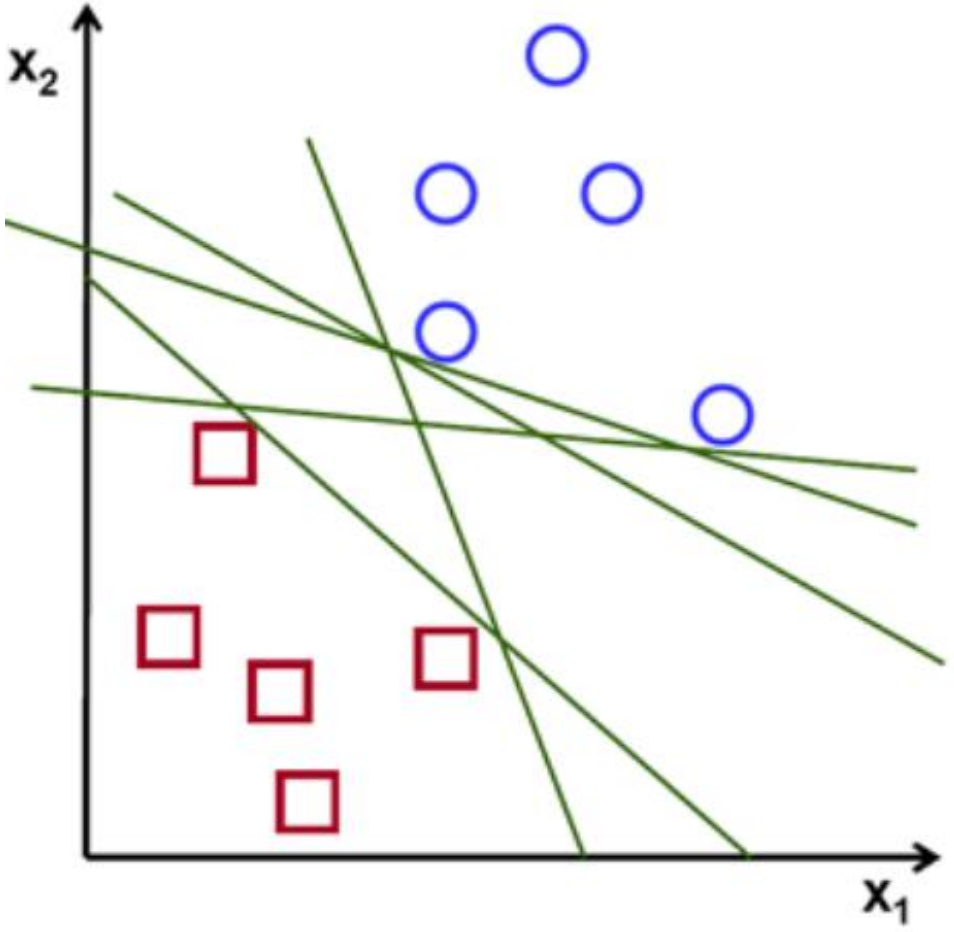


Support Vector Machines

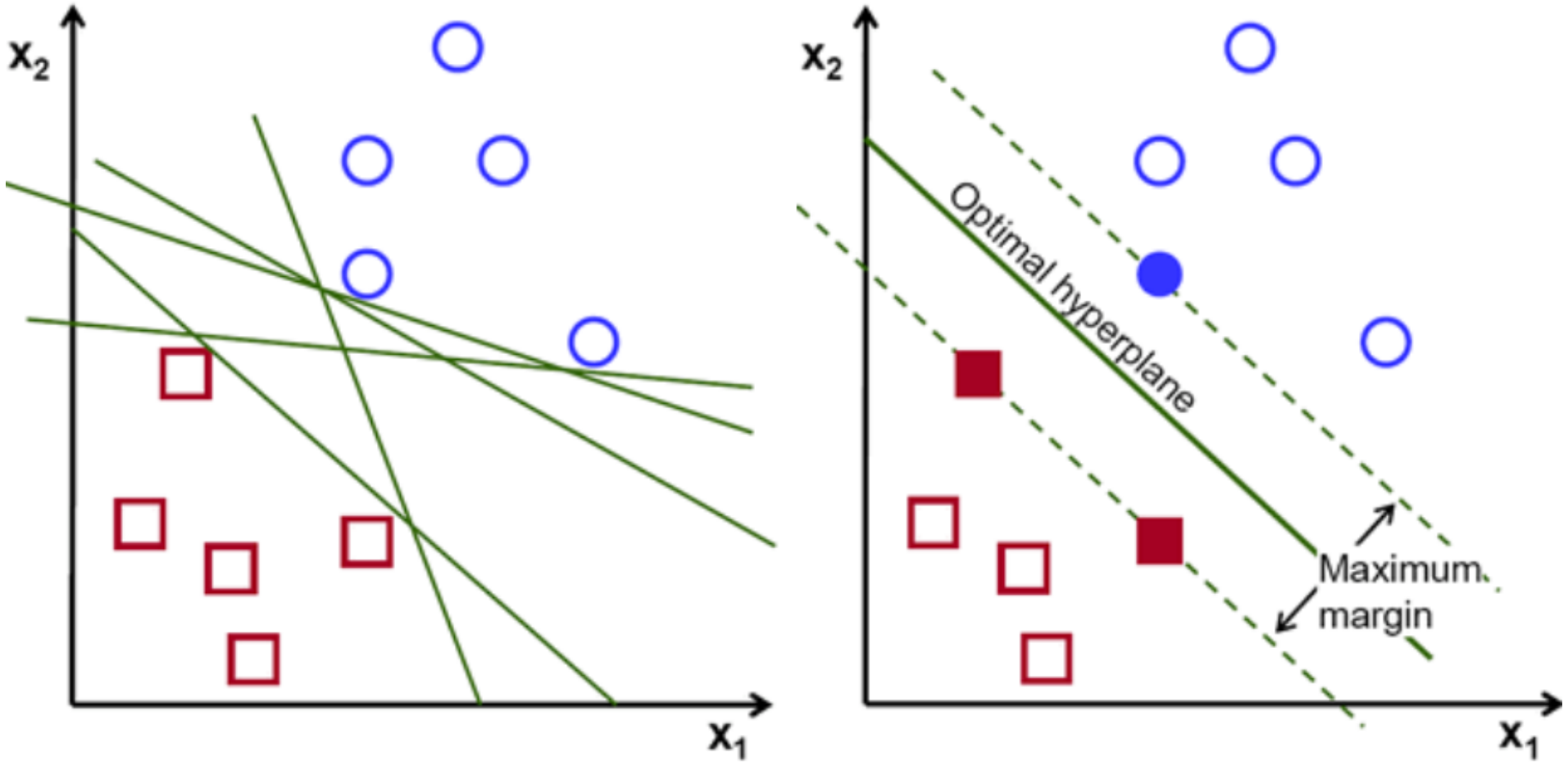
Perceptron and Logistic Regression



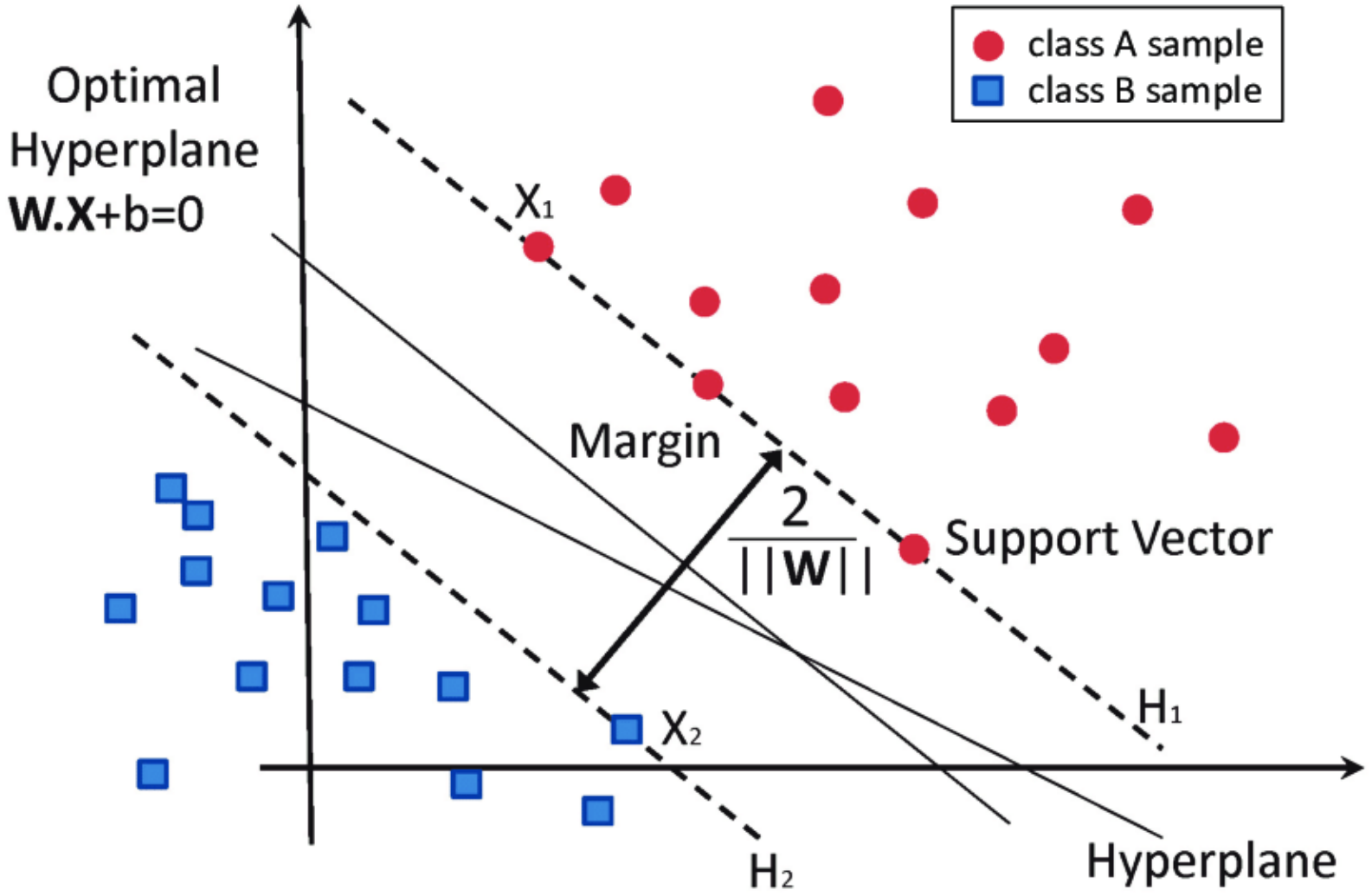
Margins



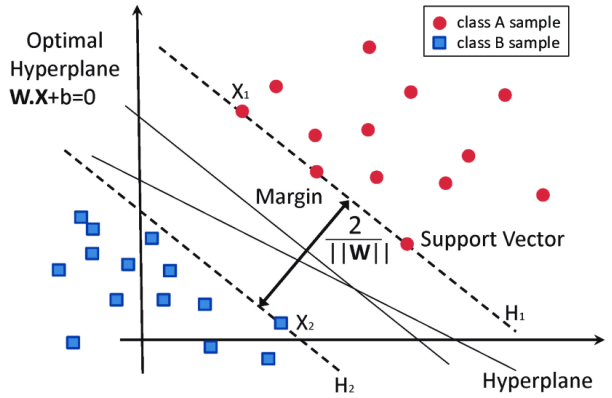
Maximum Margin



SVM Classification



Terms



- **Hyper Plane**: In SVM this is basically the separation line between the data classes. Although in SVR we are going to define it as the line that will help us predict the continuous value or target value
- **Boundary line**: In SVM there are two lines other than Hyper Plane which creates a margin . The support vectors can be on the Boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.
- **Support vectors**: This are the data points which are closest to the boundary. The distance of the points is minimum or least.

SVM Model

- Given $\left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)} \right), \left(\boldsymbol{x}^{(2)}, y^{(2)} \right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)} \right) \right\}$

  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$

$$g(z) = z$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

How to maximize the Margin?

How to maximize the Margin?

- Its very simple… use **Hinge Loss**

For an intended output $t = \pm 1$ and a classifier score y, the hinge loss of the prediction y is defined as

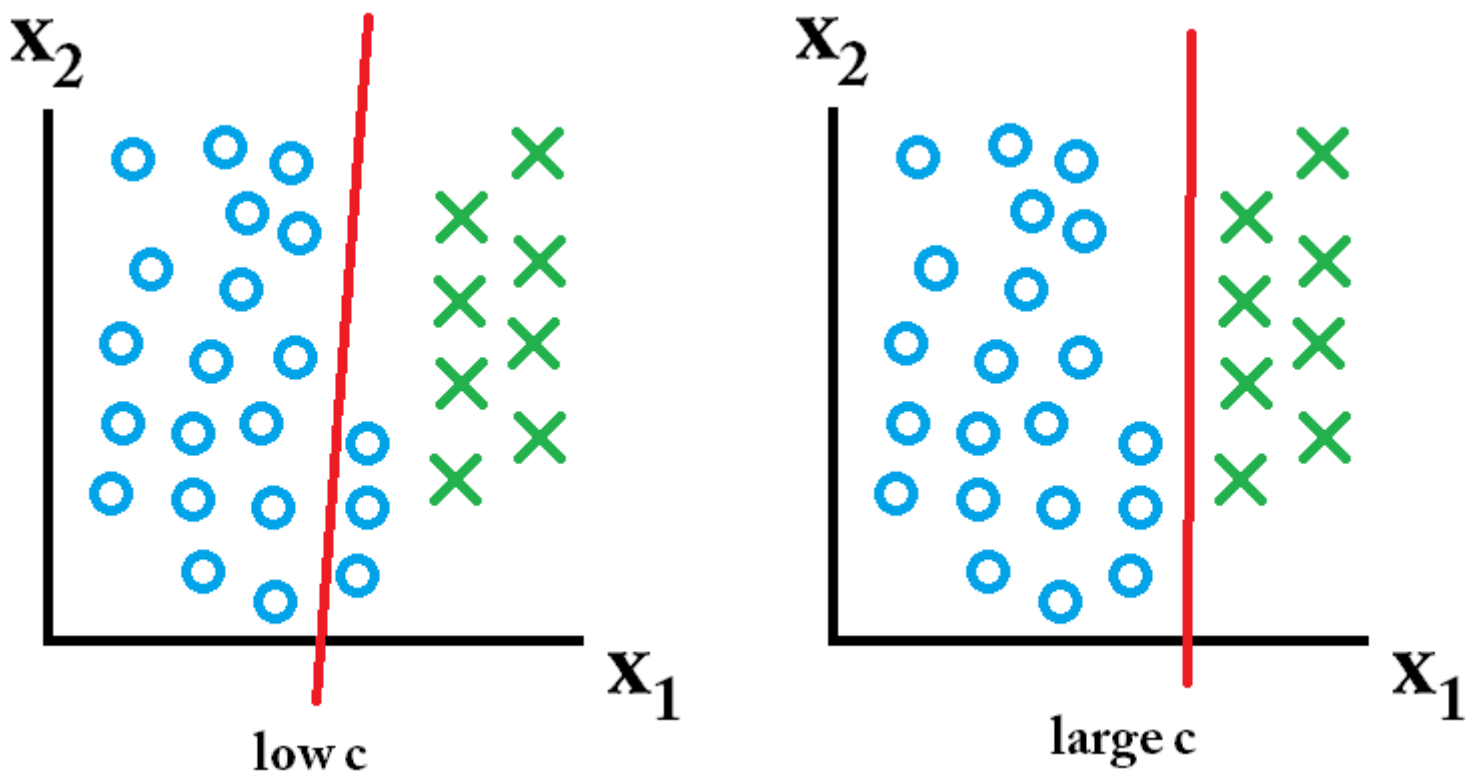
$$\ell(y) = \max(0, 1 - t \cdot y)$$

How to maximize the Margin?

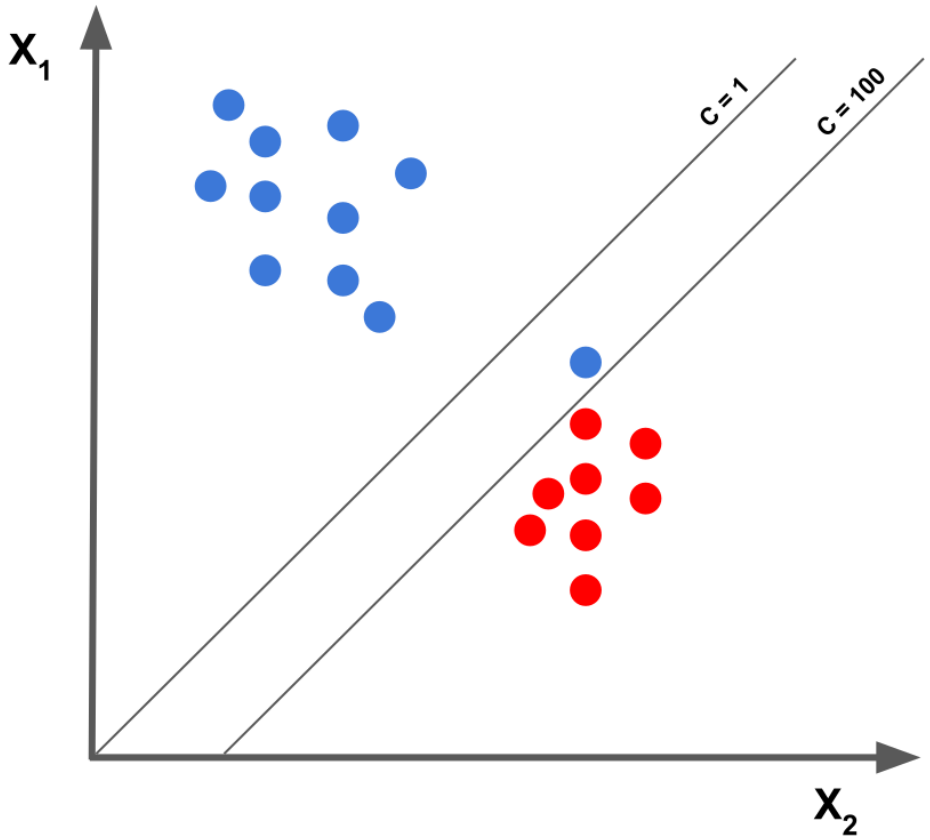
- Its very simple… add Δ to **Hinge Loss**

- L(y) = max(0,1-t*y+Δ)
- Let's say Δ is 1 then L(y) = max(0,2-t*y)
- If t=1 then y should be 2 or if t=-1 then y should be -2.
- It means the model has to learn the maximum margin to give larger y in order to compensate with the Δ. Otherwise loss will increase.

C Parameter



Larger C vs Smaller C



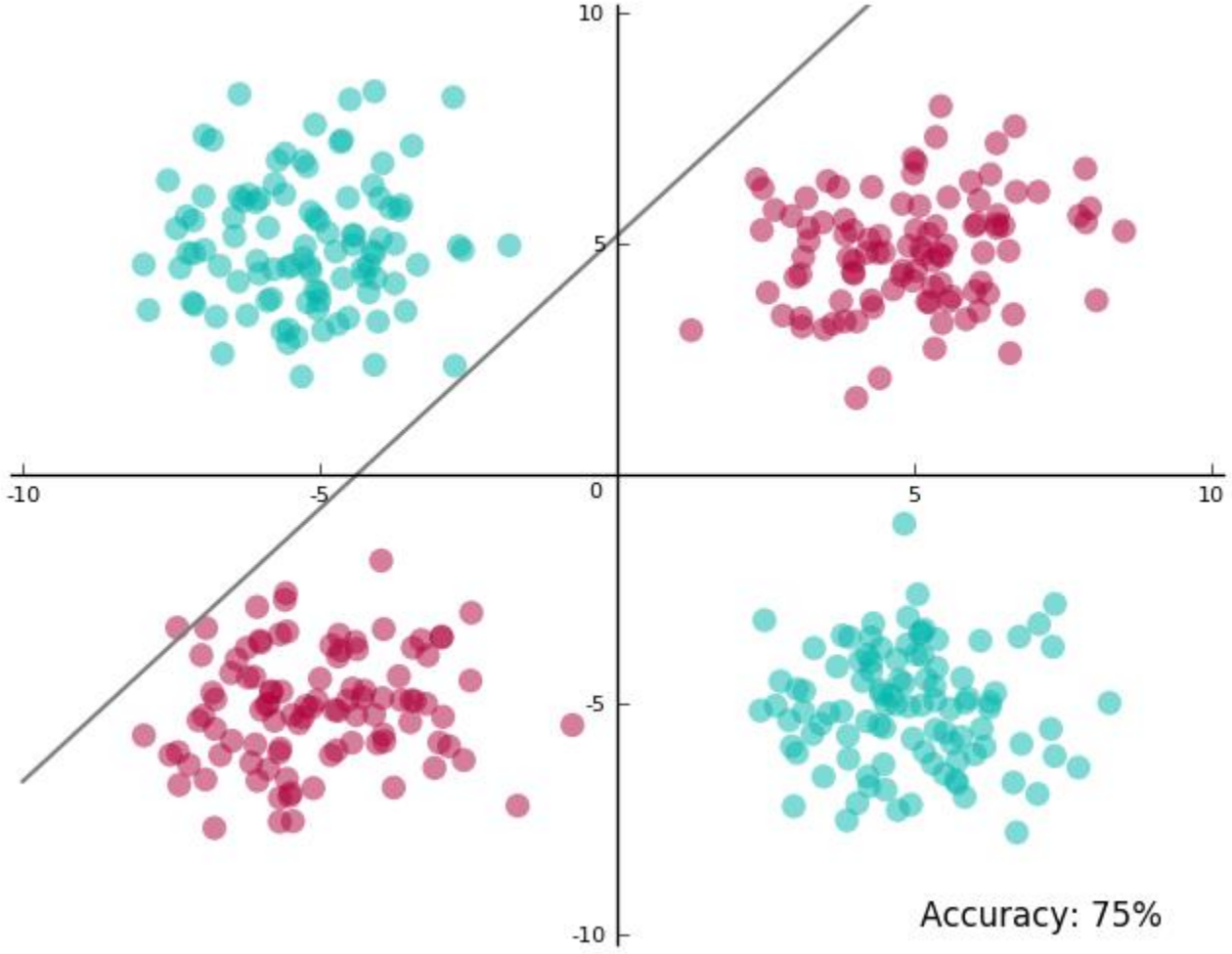
How gradients are computed?

$$\sum_i \max\{0, 1 - y_i w \cdot x\}$$

To find w^* take derivative of the total hinge loss . Gradient of each component is:

$$\frac{\partial l_{hinge}}{\partial w} = \begin{cases} 0 & y_i w \cdot x \geq 1 \\ -y_i x & y_i w \cdot x < 1 \end{cases}$$

How to deal with Non-Linear Data



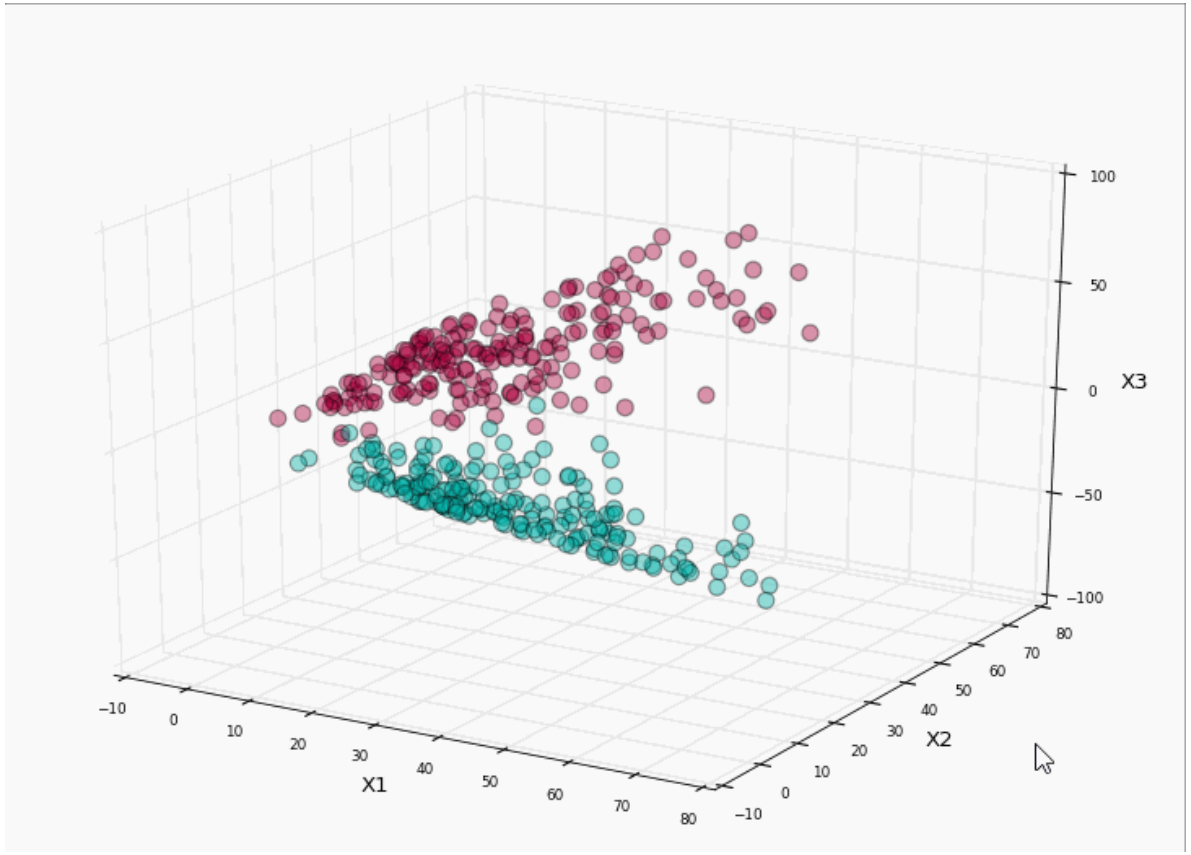
Solution

Train a non-linear model?
or
Make the data linear?

How to deal with Non-Linear Data

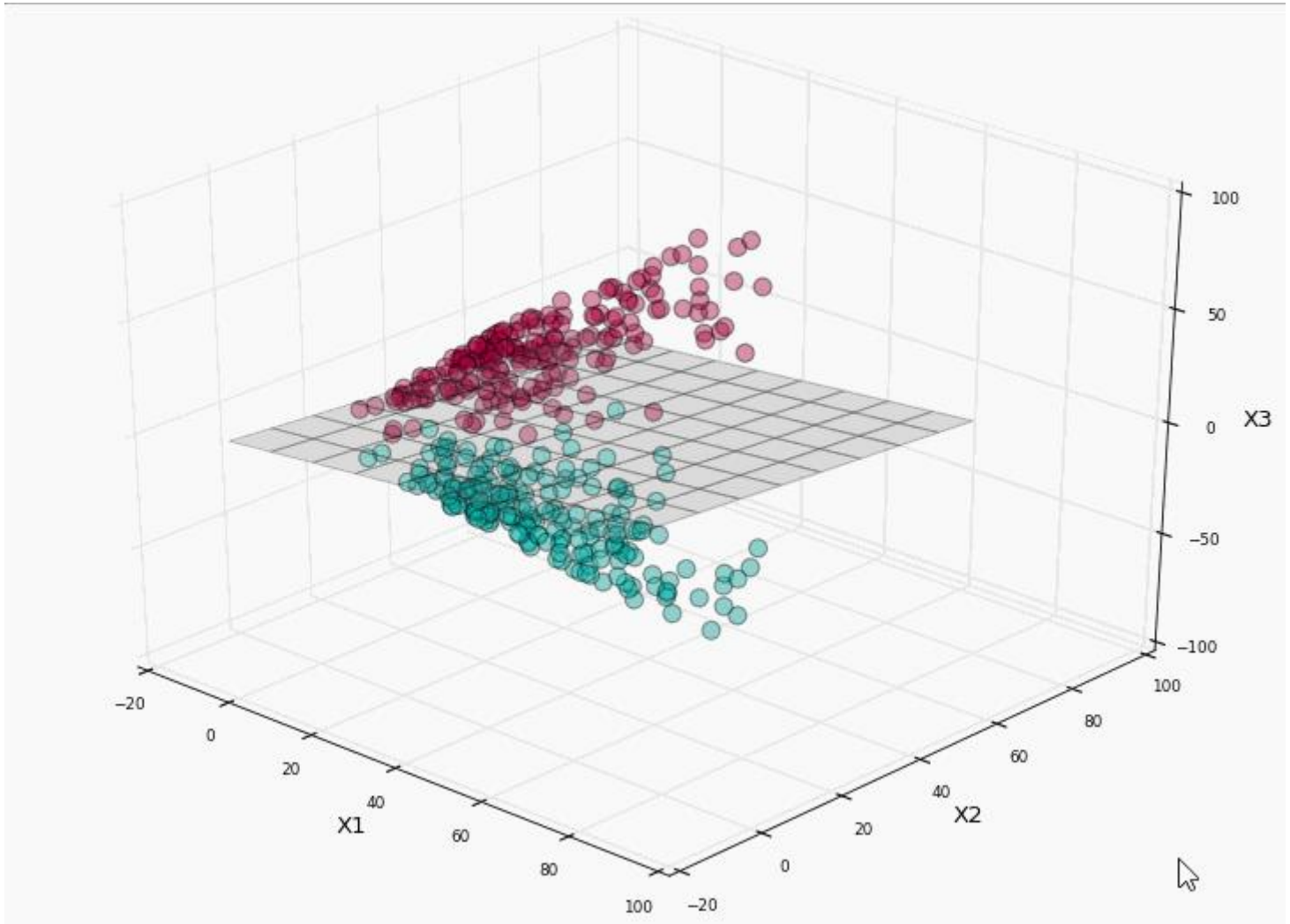
- Start with the dataset, and project it into a three-dimensional space where the new coordinates are

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$



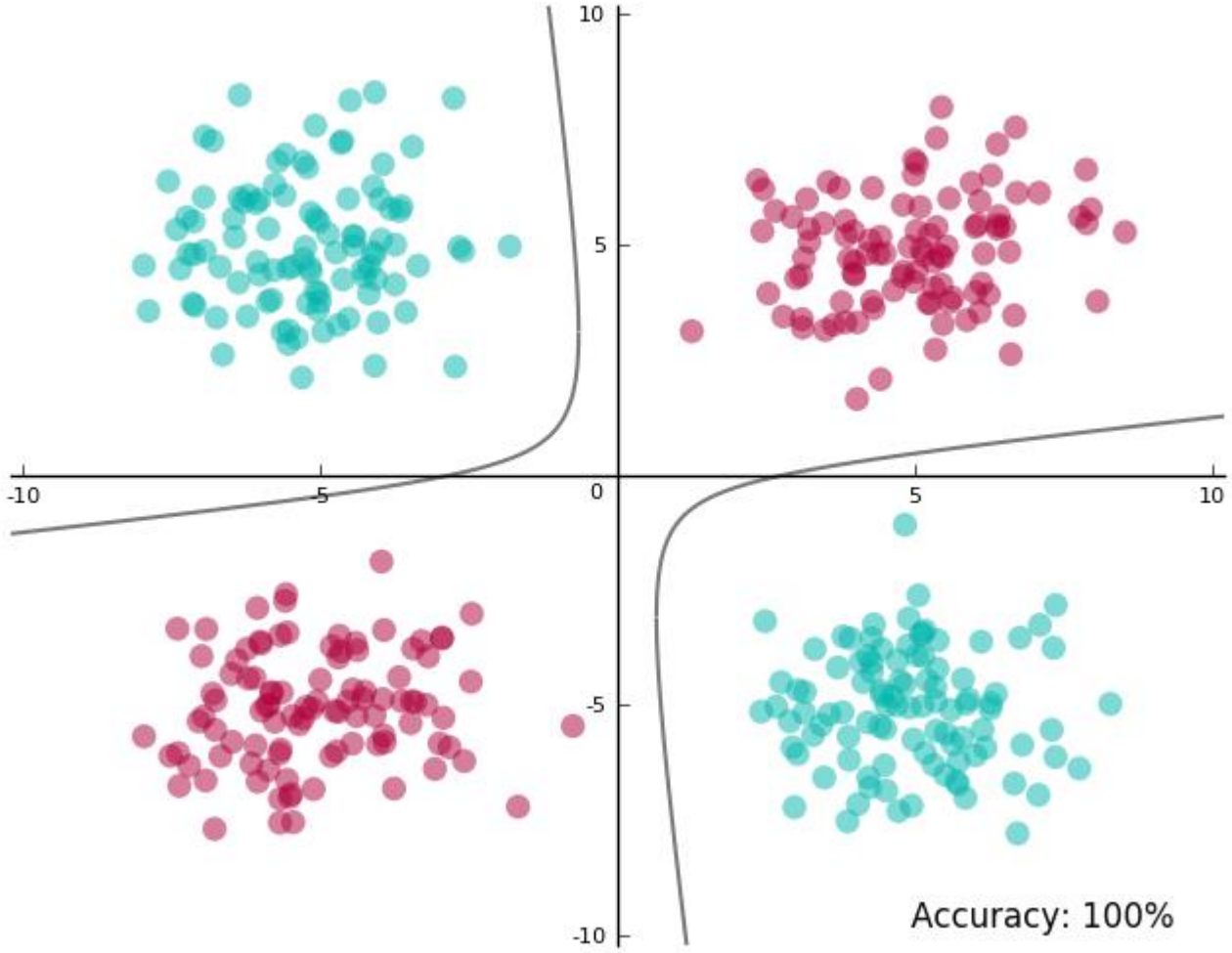
How to deal with Non-Linear Data

- Run SVM on higher dimensional data



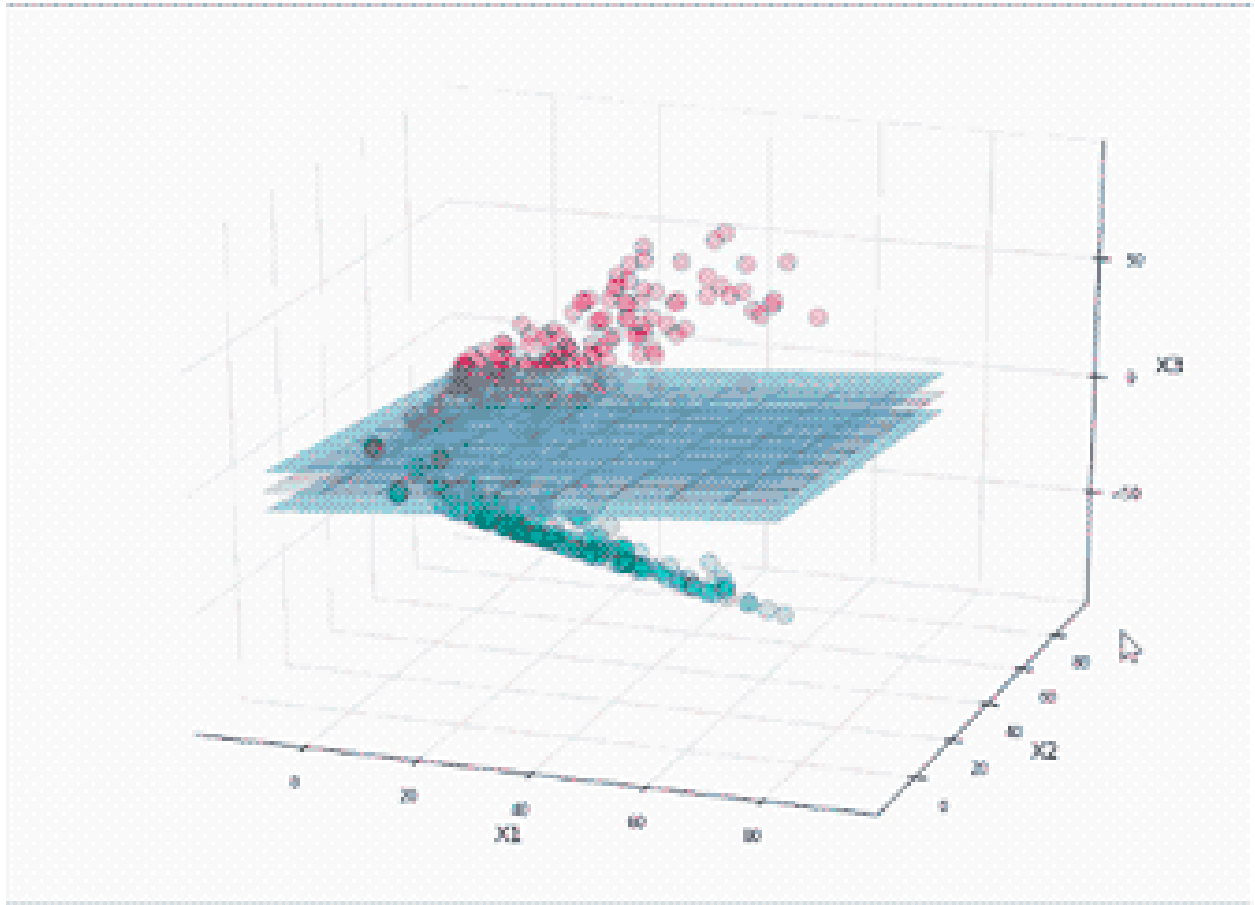
How to deal with Non-Linear Data

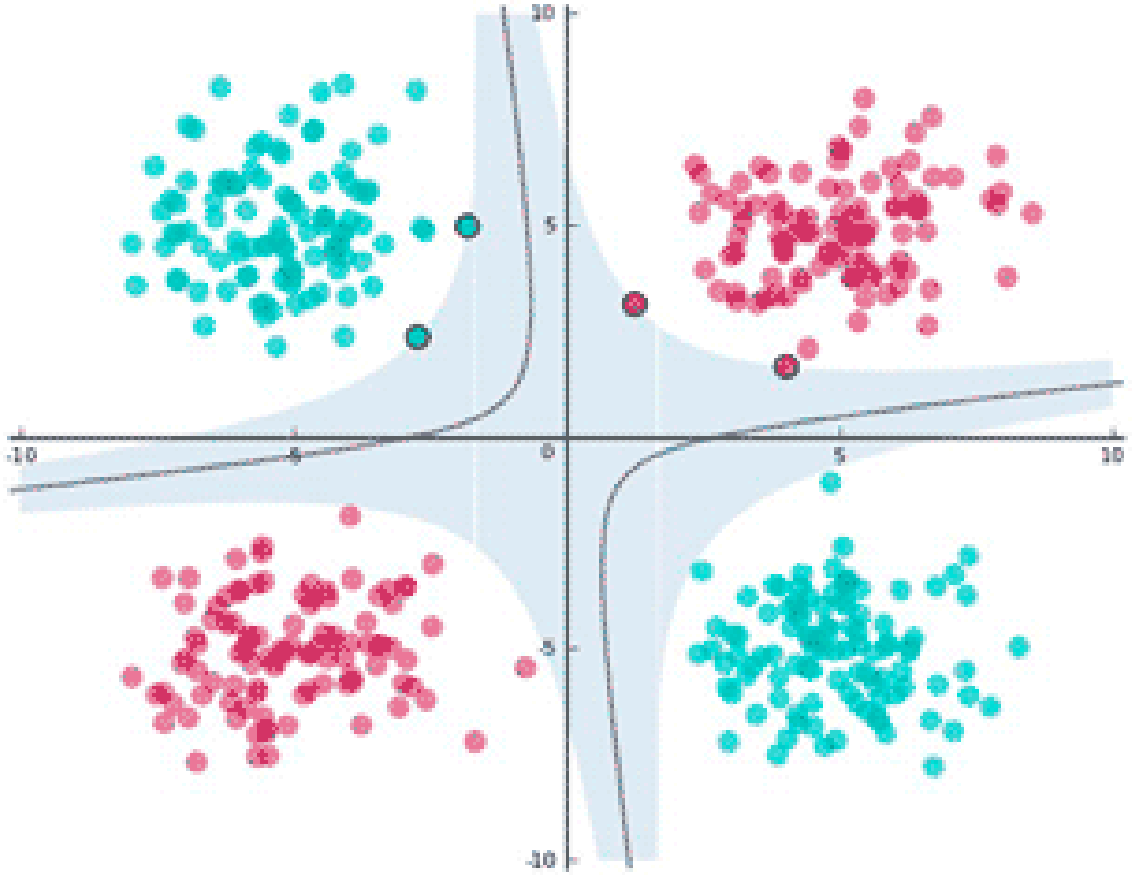
- Separation Boundary on 2D



How to deal with Non-Linear Data

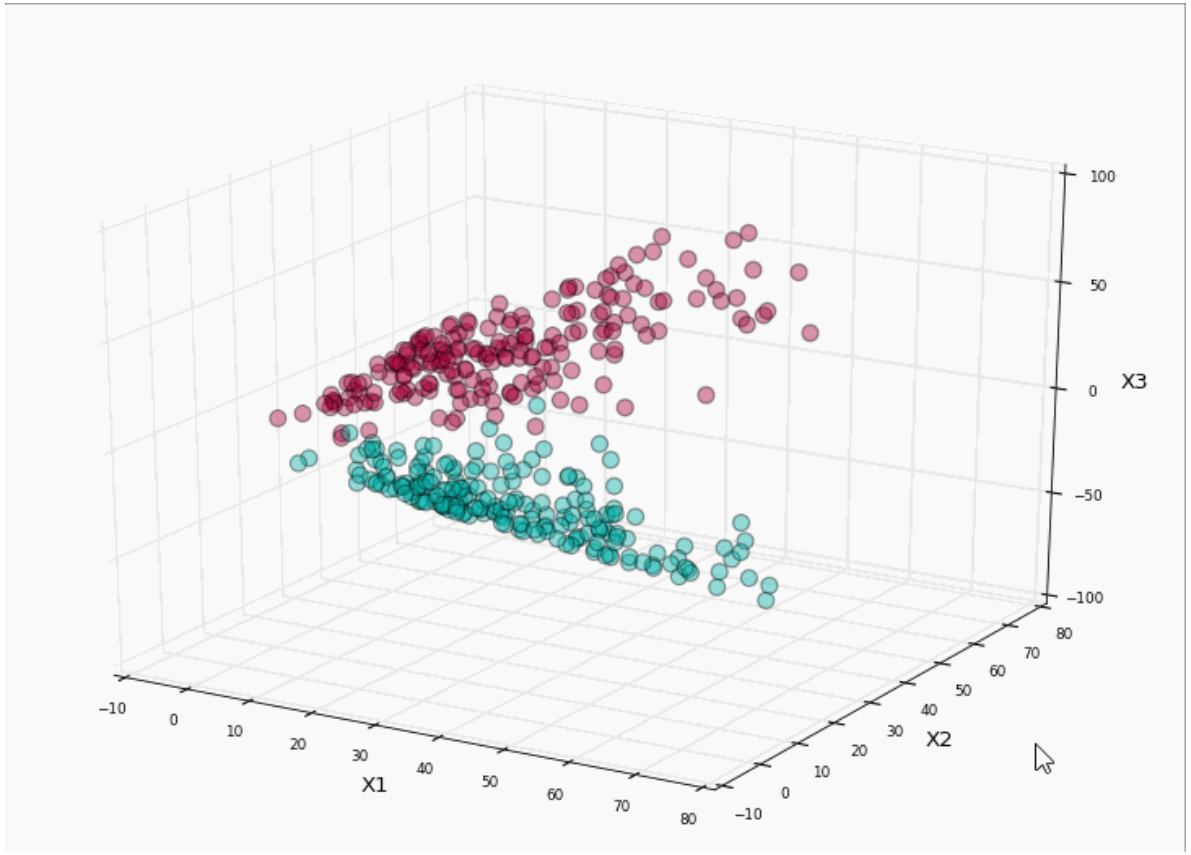
- With maximum margin





How to choose transformation as well as degree of polynomials

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$



Kernel Tricks

- Projection of data from lower dimension to higher dimension is achieved by a method called Kernel Trick.
- For *p*-dimensional vectors *i* and *j* where the first subscript on a dimension identifies the point and the second indicates the dimension number:

$$\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip})$$

$$\vec{x_j} = (x_{j1}, x_{j2}, \ldots, x_{jp})$$

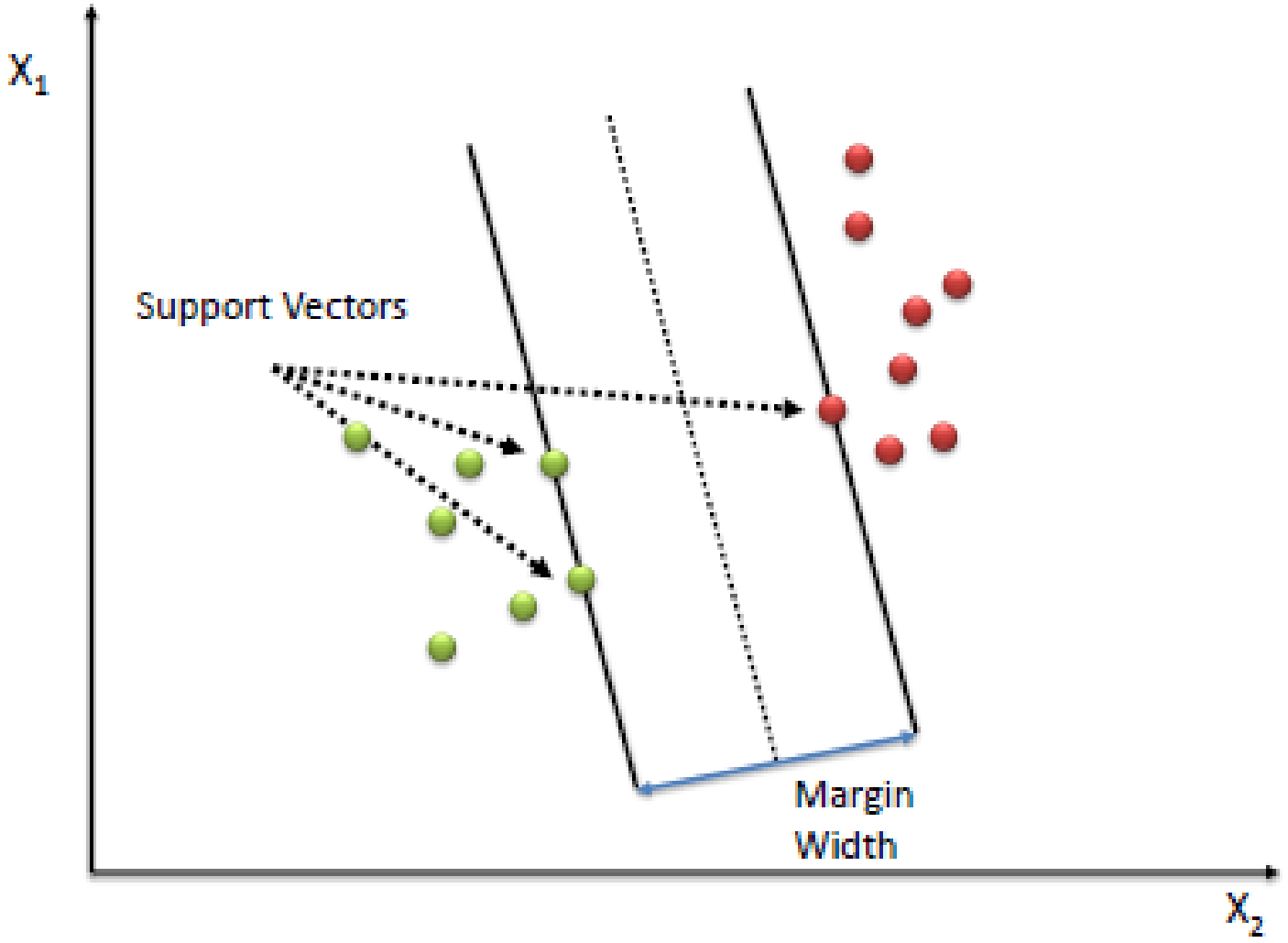
- The dot product is defined as:

$$\vec{x_i} \cdot \vec{x_j} = x_{i1}x_{j1} + x_{i2}x_{j2} + \ldots + x_{ip}x_{jp}$$

How Kernel Tricks is applied?

Image Id	No. of Exudates	Area of Largest Span	Largest Spot		yellowness
			Major Axis	Minor Axis	
Image001	12	1061	45.97	29.82	0.55
Image002	11	413	25.92	20.60	0.44
Image003	19	880	44.03	25.96	0.62
Image004	10	530	28.57	24.51	0.15
Image005	4	536	28.23	24.96	0.44
Image006	13	338	25.80	17.29	0.50
Image007	18	14809	169.96	113.87	0.59
Image008	8	5997	152.68	53.95	0.59
Image009	9	1967	64.20	40.08	0.16
Image010	10	4161	99.38	54.83	0.62
Image011	28	4023	86.23	61.83	0.64
Image012	21	630	53.66	15.68	0.62
Image013	8	1748	57.13	39.38	0.67
Image014	15	1079	62.78	22.55	0.53
Image015	12	383	27.64	18.43	0.54
Image016	20	1175	53.81	28.20	0.57
Image017	10	694	36.29	25.36	0.61
Image018	24	1601	71.12	29.01	0.64
Image019	4	535	28.47	24.81	0.61
Image020	11	626	35.54	23.97	0.57

Why Kernel Tricks works?



Use of Support Vectors

SVC in Sklearn

Attributes:

support_ : *ndarray of shape (n_SV,)*
Indices of support vectors.

support_vectors_ : *ndarray of shape (n_SV, n_features)*
Support vectors.

n_support_ : *ndarray of shape (n_class,), dtype=int32*
Number of support vectors for each class.

dual_coef_ : *ndarray of shape (n_class-1, n_SV)*
Dual coefficients of the support vector in the decision function (see Mathematical formulation), multiplied by their targets. For multiclass, coefficient for all 1-vs-1 classifiers. The layout of the coefficients in the multiclass case is somewhat non-trivial. See the multi-class section of the User Guide for details.

coef_ : *ndarray of shape (n_class * (n_class-1) / 2, n_features)*
Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

Kernel Functions (Famous ones)

- Linear Kernel: Just dot product
- Polynomial kernel

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

- Gaussian kernel

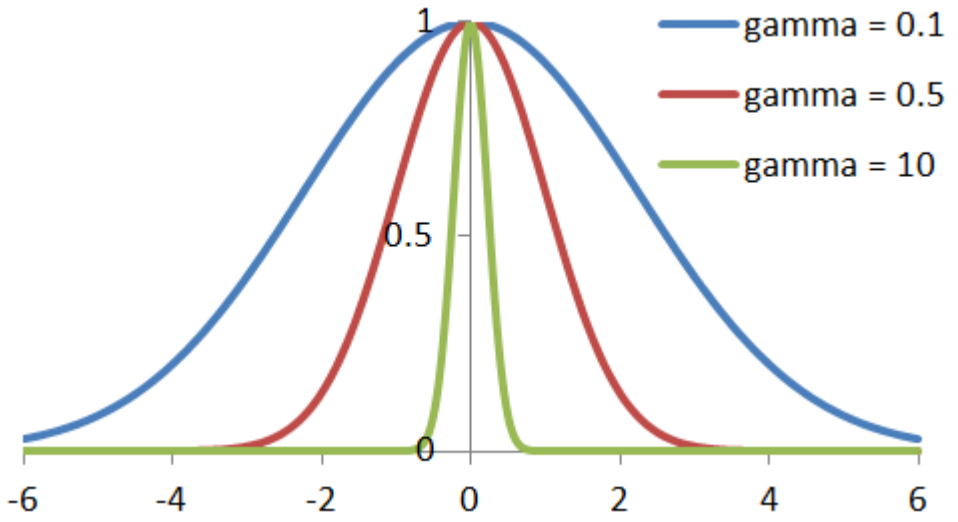
$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

- Gaussian radial basis function (RBF)

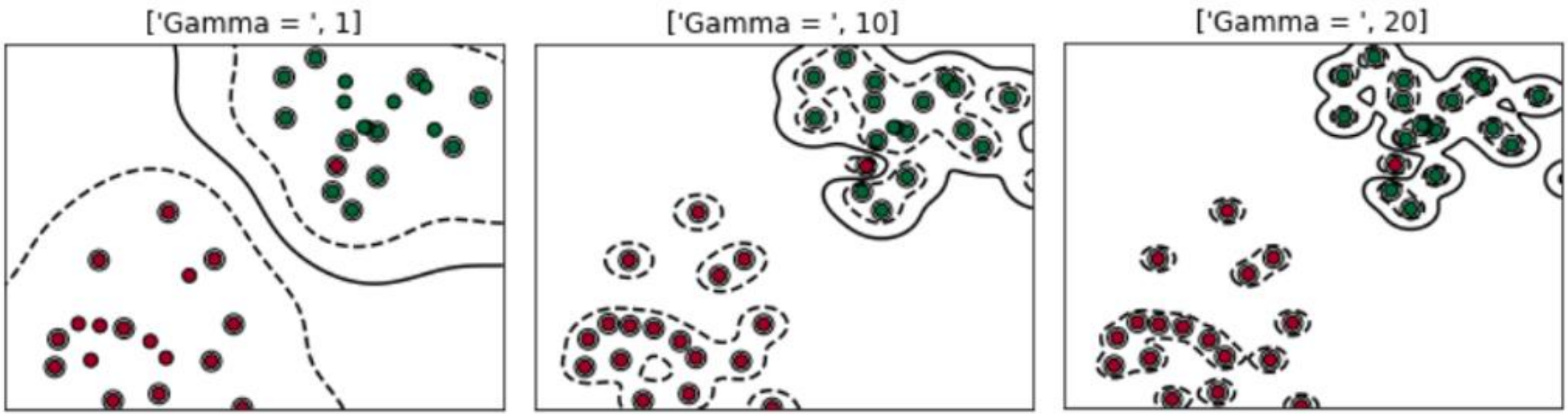
$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

Gamma Parameter

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

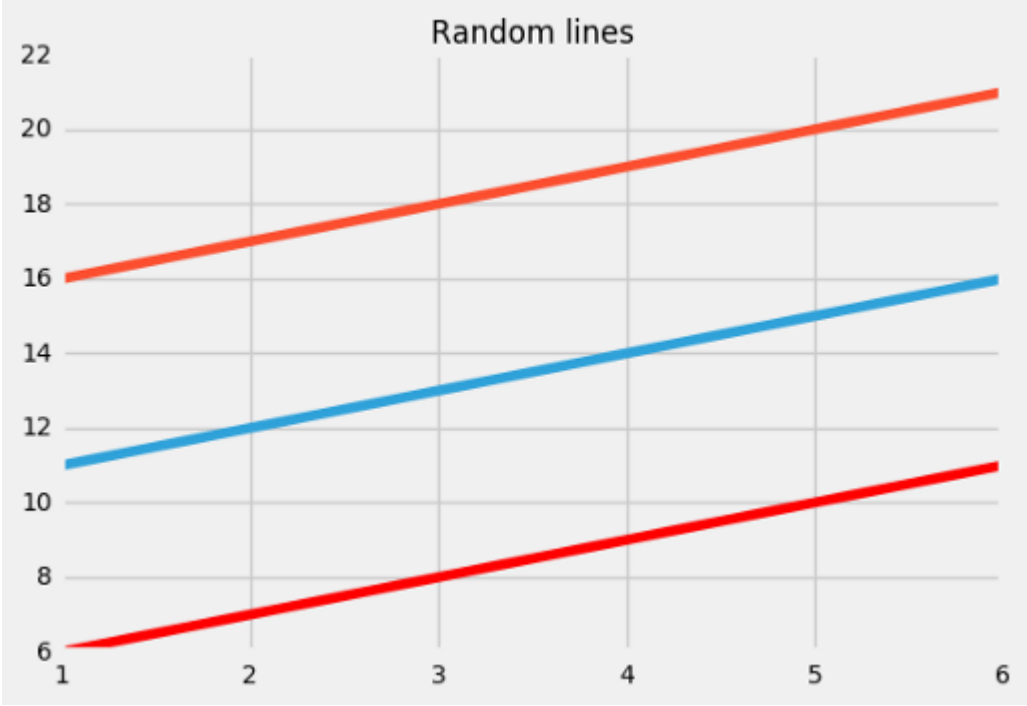


Influence of Gamma



What happens in SVR

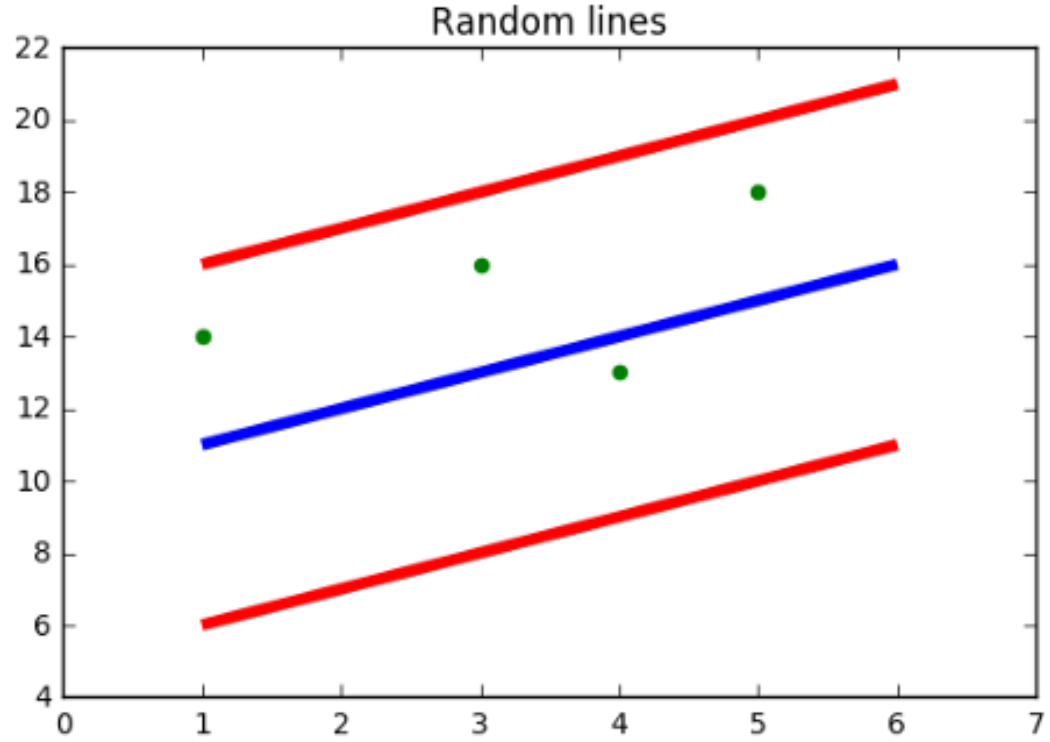
- In simple regression we try to minimise the error rate. While in SVR we try to fit the error within a certain threshold.



Blue line: Hyper Plane; Red Line: Boundary Line

What happens in SVR

- Our objective when we are moving on with SVR is to basically consider the points that are within the boundary line. Our best fit line is the line hyperplane that has maximum number of points.



Blue line: Hyper Plane; Red Line: Boundary Line

Objective

- Decide a decision boundary at ‘e’ distance from the original hyper plane such that data points closest to the hyper plane or the support vectors are within that boundary line

- Thus the decision boundary is our Margin of tolerance that is We are going to take only those points who are within this boundary.
- Or in simple terms that we are going to take only those points which have least error rate. Thus giving us a better fitting model.