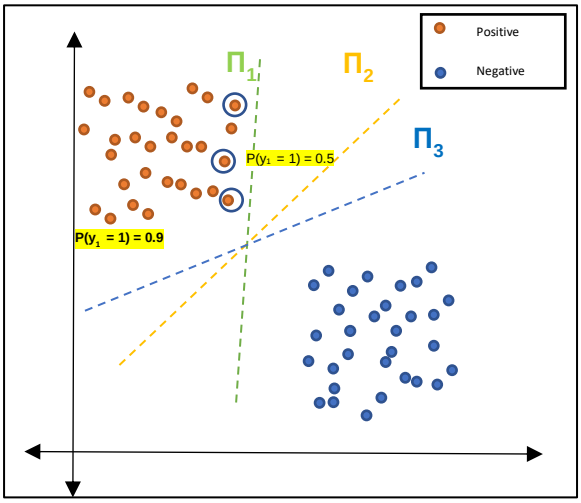
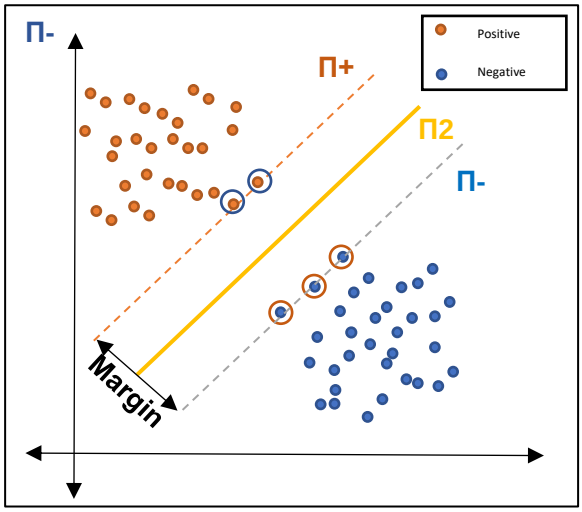
- **Support Vector Machines (SVM)**

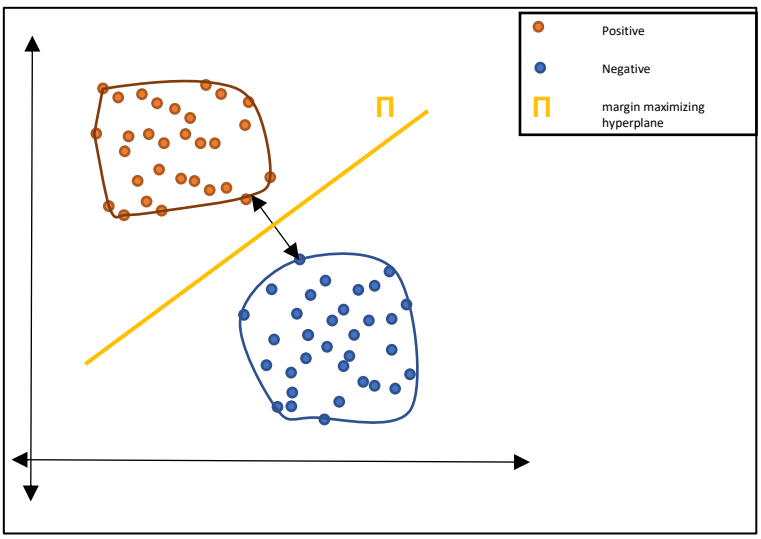


- SVM is a powerful and versatile ML model, capable of performing linear or nonlinear classification, regression, and outlier detection. SVMs are particularly well suited for classification of complex small- or medium-sized datasets.
- In the example1, we are having three planes which separates the points well but as we know the points which are closer to hyperplane, there will be probability of 0.5 that it belongs to either class. The points which are far away from hyperplane there will be more probable of belonging to either class
- The key idea of SVM is to fitting the widest possible street between the classes



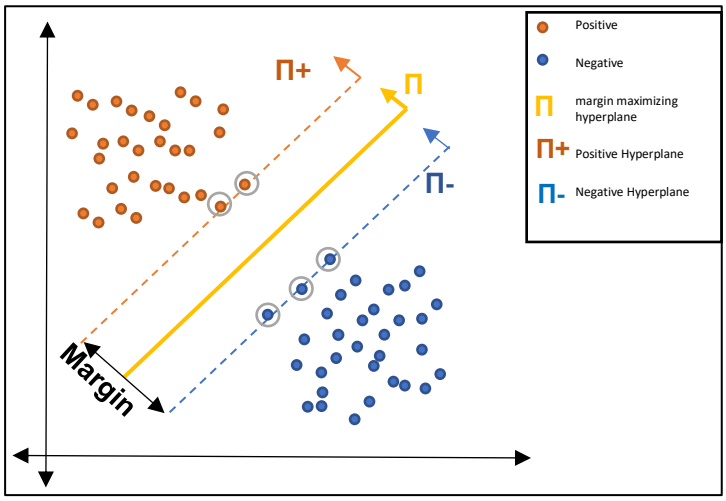
- In the example2, **Π2** hyperplane separates classes as widely as possible as compared to **Π1** **&** **Π3**. Such hyperplane is called as margin maximising hyperplane
- In this plane **Π2** we have two parallel planes such that **Π+** is the plane which touches the positive point first and **Π-** which touches negative point first.
- The distance between **Π+** and **Π-** is called as margin. If this margin is maximised then we can say our points are farther away from plane
- If the margin is high, the chances of misclassification will automatically decrease and the generalization accuracy will improve i.e., the model will give good results on unseen data

- **Support Vector**
  - Support vector is a vector consisting of points through which **Π+** and **Π-** will pass through
- **Alternative geometric intuition of SVM -Convex hull**



- Given a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it such that all points will lie inside the shape or on edge of shape
- For given set of points, we build a convex hull for both classes
- Then we find a shortest line connecting these two convex hulls
- Then we bisect the line. This line will be the margin maximising hyperplane

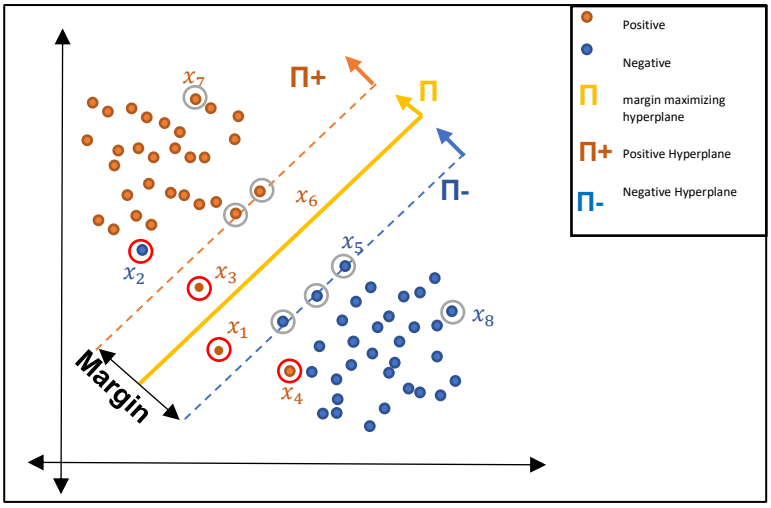
- **Mathematical derivation**



  - Let, $\pi: W^T.X + b = 0$
  - $\pi+: W^T.X + b = 1$ and $\pi-: W^T.X + b = -1$
  - $||2\ planes|| = \frac{difference\ of\ vector\ components}{||w||} = \frac{W^T.X + b - 1 - (W^T.X + b + 1)}{||W||} = \frac{|-2|}{||W||}$
  - $margin(d) = \frac{2}{||W||_2}$
  - We want to find, $w^*, b^* = argmax_{w,b}$ $y_i * W^T.X + b > 1\ for\ all\ x_i$

y_i	Support vector	$W^T x_i + b$	$y_i(W^T x_i + b)$
+1	No	>1	>1
+1	Yes	1	1
-1	No	<-1	>1
-1	Yes	-1	1

  - But practically it's not possible that all of same class points will lie on either side and there will be no points between Π+ and Π- hence the above formula is called as hard margin SVM.



x_i	y_i	$W^T x_i + b$	$y_i(W^T x_i + b)$	Zeta (ζ_i)	Remark
x_1	+1	-0.5	-0.5	1-(1.5)	Misclassified
x_2	-1	1.6	-1.6	1-(2.6)	Misclassified
x_3	+1	0.5	0.5	1-(0.5)	Correct
x_4	+1	-1.5	-1.5	1-(2.5)	Misclassified
x_5	-1	-1	1	0	Correct
x_6	+1	1	1	0	Correct
x_7	+1	1.9	1.9	0	Correct
x_8	-1	-1.9	1.9	0	Correct

  - For every point x_i We are creating a variable ζ_i such that

ζ_i	$y_i(W^T x_i + b)$	Interpretation
0	≥1	Correctly Classified
>0	<1	Distance of a point from correct hyperplane in incorrect direction

  - Updating the equation with ζ_i

$$w^*, b^* = argmax_{w,b} \frac{2}{||w||} = argmin_{w,b} \frac{||w||}{2}$$

$$w^*, b^* = argmin_{w,b} \frac{||w||}{2} + C.\frac{1}{n}\sum_{i=1}^{n} \zeta_i \ \mid \{y_i W^T x_i \geq 1 - \zeta_i \ \ for\ all\ points\ and\ \zeta_i \ \ \geq 0\}$$

$$= regularizer + hyperparameter * Loss$$

$$\frac{1}{n}\sum_{i=1}^{n} \zeta_i \ = average\ distance\ of\ misclassified\ points$$

  - Above equation is called as soft margin SVM.
  - C is a hyperparameter which is always positive.
  - As C increases, tendency to make mistakes decreases, overfitting on training data, high variance.
  - As C decreases underfitting occurs, high bias.

- **Why we take values +1 and -1 for Support vector planes**

$$\boxed{\pi+: W^T.X + b = 1}$$
$$\boxed{\pi-: W^T.X + b = -1}$$

  - W is perpendicular plane to π of any length
  - The objective of optimization was to maximize the margin between positive & negative hyperplanes with constraints.
  - 1st approach-
    - Let,

$$\boxed{\pi+: W^T.X + b = k}$$
$$\boxed{\pi-: W^T.X + b = -k}$$
$$\boxed{margin(d) = \frac{2k}{||W||_2}}$$

    - our ultimate goal was to maximise the margin, Ultimately the w & b that will maximise this margin will be same for any k. e.g., if k = 5 following equation will be

$$\boxed{w^*, b^* = argmax_{w,b} \frac{10}{||w||}}$$

    - It does not matter we can take any $+k$ and $-k$ and also want both the support vector planes be equidistant from the margin maximizing hyper plane. +1 and -1 are chosen for mathematical convenience.
  - 2nd approach-
    - Let,

$$\boxed{\pi: W^T.X + b = k}$$

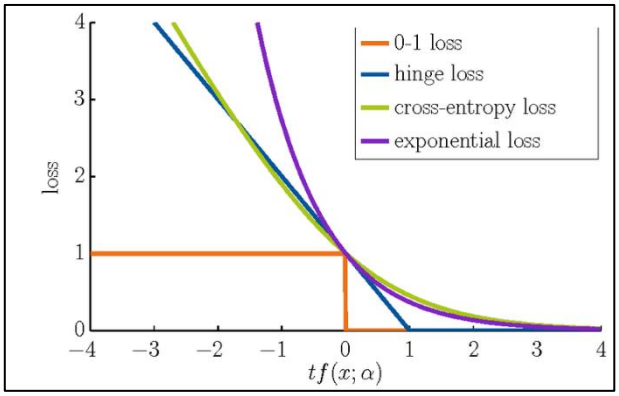
    - now we divide every term in the equation by 'k'.

$$\boxed{\frac{W^T.X}{k} + \frac{b}{k} = 1}$$
$$\boxed{W' + b' = 1}$$

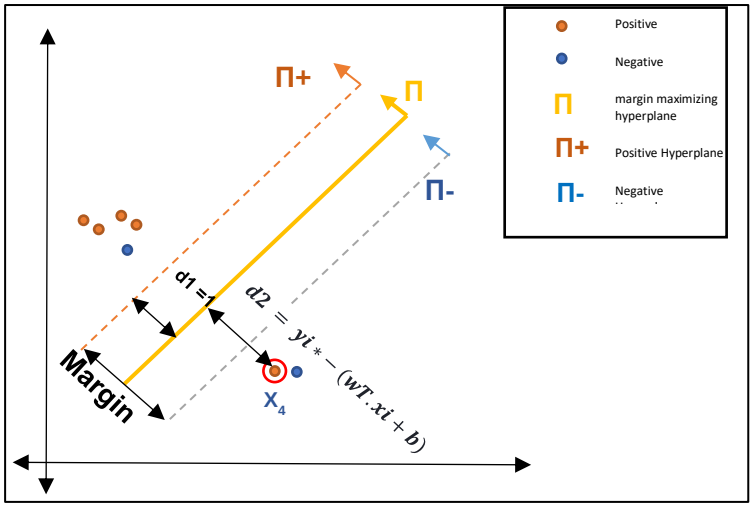
    - We can now consider w/k as a variable w' and b/k as say b'. So, now the RHS can be considered to be 1. Doing this is perfectly fine because we do not have any conditions or presumptions regarding values of w and b. They can be any value.

- **Loss function (Hinge Loss) based interpretation**

Algorithm	Loss Function
Logistic Regression	Logistic Loss + Reg.
Linear Regression	Linear Loss + Reg.
Support Vector Machines	Hinge Loss +Reg.



- $\boxed{\mathbb{Z}_i = y_i(W^T x_i + b)}$
- 1 if $\mathbb{Z}_i < 0$ i.e., incorrectly classified points
- 0 if $\mathbb{Z}_i > 0$ i.e., correctly classified points
- $\boxed{\mathbb{Z}_i = y_i.W^T x_i}$
- If $\mathbb{Z}_i \geq 1$; Hinge -loss = 0
- If $\mathbb{Z}_i < 1$; Hinge -loss = 1 - $\mathbb{Z}_i$
- Ultimately, hinge-loss is $\boxed{max\ (0,\ 1 - \mathbb{Z}_i)}$



- In above equation, total distance of the point (xi, yi) which is a misclassified positive point from the hyperplane 'pi+' is d1+d2.
  1. If 'w' is a unit vector, then distance between (xi, yi) and the hyperplane = wT. xi
  2. If 'w' is not a unit vector, then distance between (xi, yi) and the hyperplane = (wT. xi)/||w||

$$\zeta_i = \text{total distance of misclassified point from correct plane} = 1 - (y_i . W^T x_i + b)$$

$\zeta_i = 1 - \mathbb{Z}_i$ when xi is misclassified

$\zeta_i = \mathbb{Z}_i = 0$ when xi is correctly classified

- This signifies that hinge-loss is $max(0,\ 1 - \mathbb{Z}_i) = \zeta_i$
- Soft margin SVM which allows error and tried to minimize them

$$(\boldsymbol{W}^*, \boldsymbol{B}^*) = argmin_{w,b} \frac{||w||}{2} + C.\frac{1}{n}\sum_{i=1}^{n}\zeta_i \ | \{y_i W^T x_i \geq 1 - \zeta_i \ \ for\ all\ points\ and\ \zeta_i \ \geq 0\}$$
$$= regularizer + hyperparameter * Loss$$

- If value of C increases, model tends to overfit and If value of C decreases, model is underfit
- Loss Min of SVM

$$(W^*, B^*) = \sum_{i=1}^{n} max(0, 1 - (y_i . W^T x_i + b)) + \lambda||W||^2$$
$$= Loss + hyperparameter * regularizer$$

- If value of λ increases, model tends to underfit and If value of λ decreases, model is overfit

- **Dual form of SVM formulation**
  - The following soft margin SVM is primal Form of SVM

$$(W^*, B^*) = argmin_{w,b} \frac{||w||}{2} + C.\frac{1}{n}\sum_{i=1}^{n}\zeta_i \ | \{y_i W^T x_i \geq 1 - \zeta_i \ \ \forall\ for\ all\ points\ and\ \zeta_i \ \geq 0\}$$
$$= regularizer + hyperparameter * Loss$$

  - The above equation will give us $f(x_q) = W^T . X + b$ for a query point
  - The alternative method is dual form of SVM which uses Lagrange's multiplier to solve the constraints optimization problem.

$$max_{\alpha_i} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i.\alpha_j.y_i.y_j.(x_i^T x_j) \ | \alpha_i \geq 0 \ \& \sum_{j=1}^{n}\alpha_i.y_i = 0$$

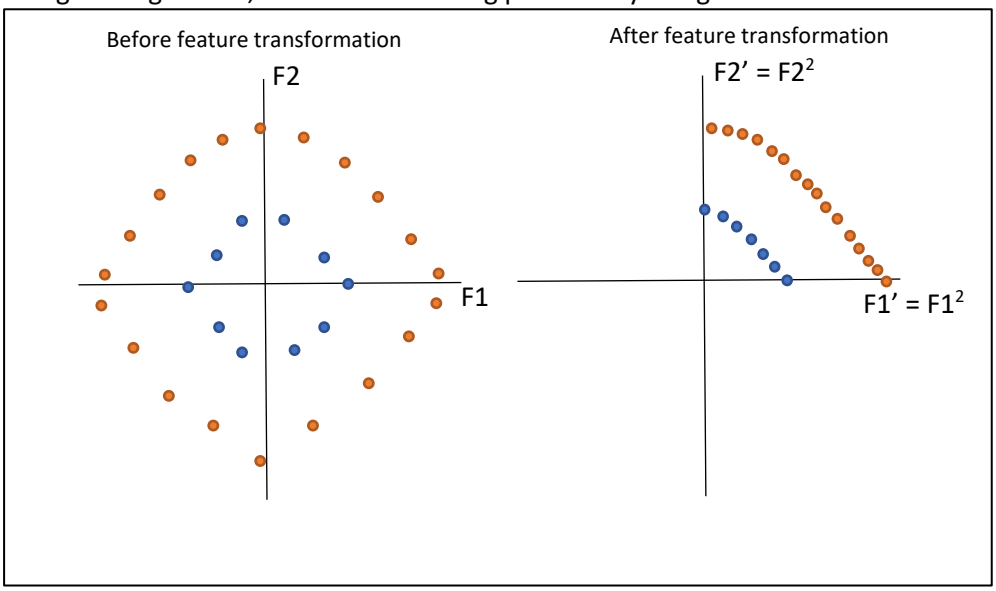
    - For every x_i there is a corresponding α_i
    - x_i is present in form of product $x_i^T x_j$.
    - x_i and x_j are two data points in Training Data (x_i and x_j are Normalized vectors).
    - Dual form will find $f(x_q) = \sum_{i=1}^{n}\alpha_i.y_i.(x_i^T x_q) + b$
    - $\alpha_i > 0$ for support vectors & $\alpha_i = 0$ for non-support vectors. Which implies only points that will matter for finding $f(x_q)$ are support vectors
    - $x_i^T x_j = x_i . x_j = cosine\ sim(x_i, x_j)$

- This similarity can be considered as any similarity
- One class of similarity function is kernels so above similarity equation can be rewritten as $K(x_i, x_j)$ Where K is a kernel function.

$$K(x_i, x_j) = exp\left(-\gamma \left|\left|x_i - x_j\right|\right|^2\right)$$

- Kernel function makes it easier to use dual form as compared to the primal form of SVM as we can use similarity
- If we keep $(x_i^T x_j)$ term as is, it's called as linear SVM
- If we replace $(x_i^T x_j)$ term with $K(x_i, x_j)$, it's called as kernel SVM
- In linear SVM, we are trying to find a margin maximising hyperplane in the space of $x_i's$
- In logistic regression, we are trying to find hyperplane giving min logistic loss in the space of $x_i's$
- Kernelization makes SVM handle non-linearly separable data
- For a given query point x_q
  a) We compute the similarity of x_q with each of support vectors x_i (only support vectors have α_i>0 and hence only support vectors matter) using $K(x_i, x_q)$
  b) multiply the similarity between x_i and x_q obtained from $K(x_i, x_q)$ with the corresponding class label of each support vector x_i and α_i i.e., $y_i.\alpha_i$
  c) We find the sum of each value obtained in the above step(b).
  d) The sign of the result obtained in step(c) determine the class of Xq.

- **Polynomial Kernel**
  - In logistic regression, we solved following problem by using feature transform



  - Given 2 points (x_1, x_2), it's polynomial kernel will be

$$K(x_1, x_2) = (\, x_1^T x_2 + C\,)^d$$

  - When d=2, K will give us Quadratic kernel

$$K(x_1, x_2) = (\, 1 + \, x_1^T x_2\,)^2$$

Let, $\begin{bmatrix} x_1 = [x_{11} & x_{12}] \\ x_2 = [x_{21} & x_{22}] \end{bmatrix}$

$$K(x_1, x_2) = (1 + \, x_{11}x_{21} + x_{12}x_{22}\,)^2$$

$$K(x_1, x_2) = 1 + \, {x_{11}}^2{x_{21}}^2 + {x_{12}}^2{x_{22}}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}$$

$$\begin{bmatrix} Let, x_1' = \left[1, {x_{11}}^2, {x_{12}}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}\right] \\ x_2' = \left[1, {x_{21}}^2, {x_{22}}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22}\right] \end{bmatrix}$$

$$K(x_1, x_2) = {x_1'}^T x_2'$$

- Here, we can say that internally kernelization is transforming the datapoints into x_1, x_2 to x_1', x_2' where dimensionality is increases from d to d'. typically, d'>d

$$kernelization: \ d \xrightarrow{FT} d' | \ d' > d$$

- According to mercer's theorem, the d' after kernelization will be linearly separable even if d is not separable

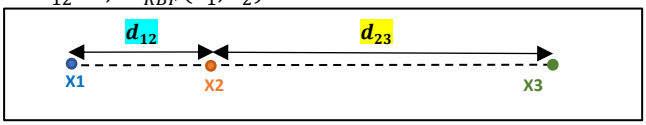
- **Radial basis function kernel**
  - It's the most popular and most generic kernel in SVM.

$$K_{RBF}(x_1, x_2) = exp\left(\frac{-||x_1 - x_2||^2}{2\sigma^2}\right)$$

$$||x_1 - x_2||^2 = d_{12} = distance\ between\ x_1\ and\ x_2$$
$$\sigma = hyperparameter$$

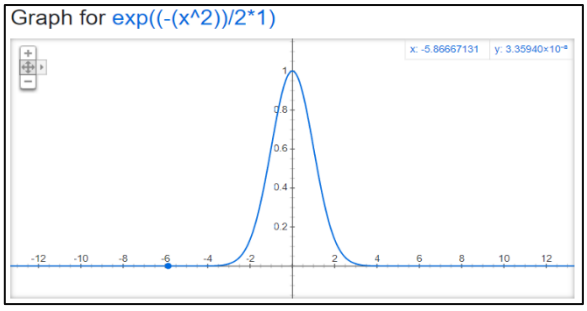
$$K_{RBF}(x_1, x_2) = exp\left(\frac{-{d_{12}}^2}{2\sigma^2}\right)$$

  - As $d_{12} \uparrow$; $K_{RBF}(x_1, x_2) \downarrow$

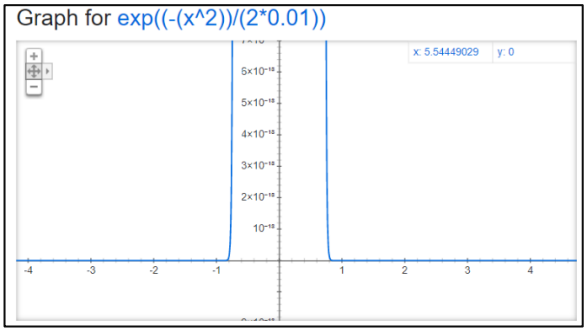


  - $as\ d_{12} < d_{23} : K_{RBF}(x_1, x_2) > K_{RBF}(x_2, x_3)$
  - which is ultimately saying as the two points are similar i.e., the distance between them is less., the corresponding kernel value is more
  - As distance increases, K (x1, x2) decreases and reaches 0
  - As sigma reduces, the tolerance to distances reduces, i.e., data points need to be very near to have similarity values non-zero

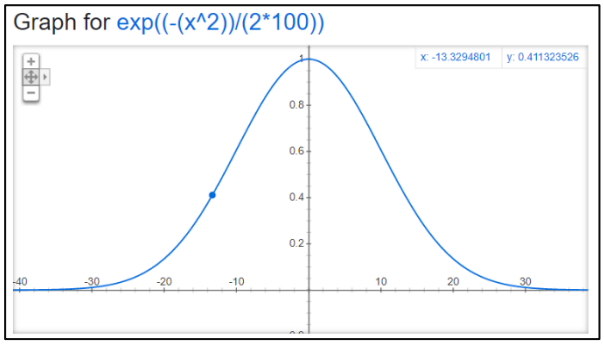
- Impact of σ on K



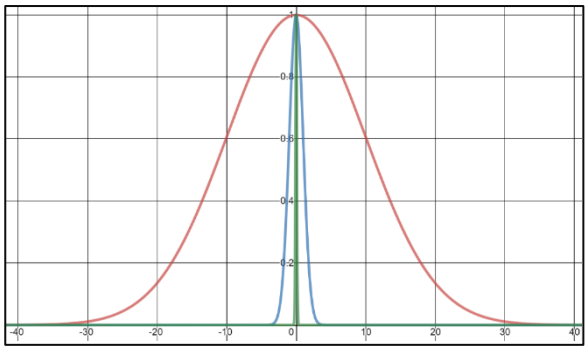
  - Case-1{ $\sigma = 1$ }
    - if d = 0, k =1
    - at d > 4, k = 0
    - As $d_{12} \uparrow$; $K \downarrow$ exponentially
    - To summarize, as the distance is increased and similarity decreased, k tends to be 0



  - Case-2{ $\sigma = 0.1$ }
    - if d >1, k =0
    - even at 0<d<1, the k falls very sharply. The graph is very peaked.



- Case-3{ $\sigma = 10$ }
  - if d >30, k =0



- In the diagram we are comparing graphs corresponding to $\sigma = 0.1, \sigma = 1, \sigma = 10$
- as σ increases more point will be similar to a given point
- This is similar to KNN such that

$$\boxed{as\ \sigma_{RBF} \uparrow = K_{KNN} \uparrow}$$

- In case of KNN the time complexity was O(nD), in RBF it's reduced as we only have to find support vectors
- There are some domain specific kernels like string kernel, genome kernel, etc.

- **Train and run time complexities**
  - Training Time ~ $O(n^2)$ for kernel SVM.
  - If d < n train time complexity is $O(nd^2)$
  - Run Time complexity = O(kd) where k = no of support vectors s.t. $1 \leq k \leq n$
- **nu-SVM: control errors and support vectors**
  - nu-SVM is a SVM model where nu is hyperparameter where $0 \leq nu \leq 1$
  - nu can be interpreted as
    - nu $\geq$ fraction of errors
    - nu $\leq$ fraction of support vectors
    - e.g., if nu = 0.01 and n = 100000
      - percentage of errors $\leq$ 1%
      - no of misclassified points $\leq$ 1000
      - no of Support vector $\geq$ 1000
- **SVM Regression**

$$\boxed{\begin{gathered}(\boldsymbol{W}^*, \boldsymbol{B}^*) = \ argmin_{w,b}\ \frac{||w||}{2} \\ such\ that\ for\ all\ i\ \ |y_i - (\hat{y}_i)| \leq \epsilon \\ \epsilon > 0\end{gathered}}$$

  - ϵ is a hyperparameter,
    - if it is low, the errors are less and chances of overfitting increases
    - if it is high, the errors are more and chances of underfitting increases