KNN, Naive Bayes, Support Vector Machines with Text Data

EDVANCER EDUVENTURES

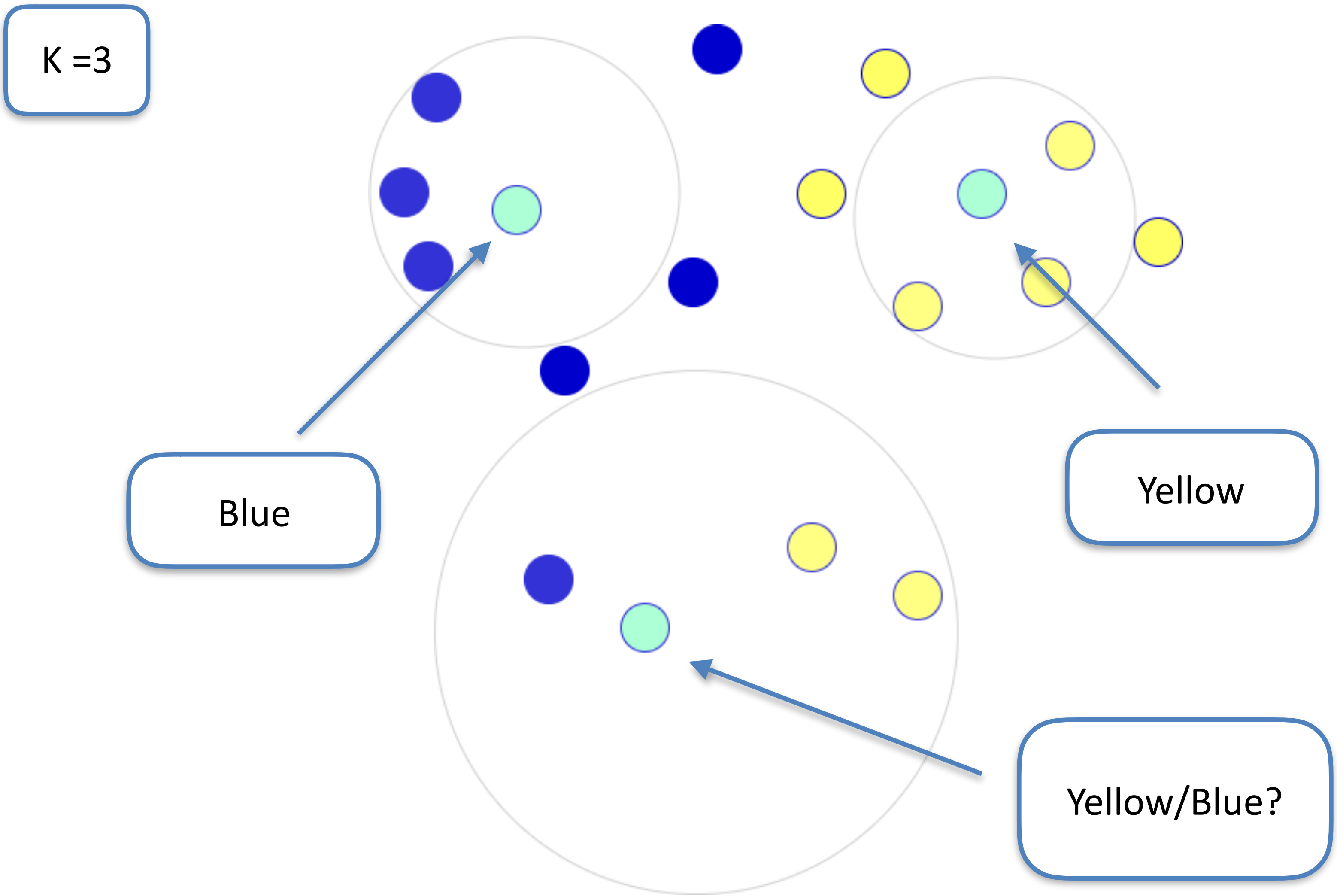
Agenda

Discussion Flow

- K-Nearest Neighbours
- Naive Bayes
- Support Vector Machines
- Implementation in Scikit-Learn (with features from text data)

K-Nearest Neighbours

KNN

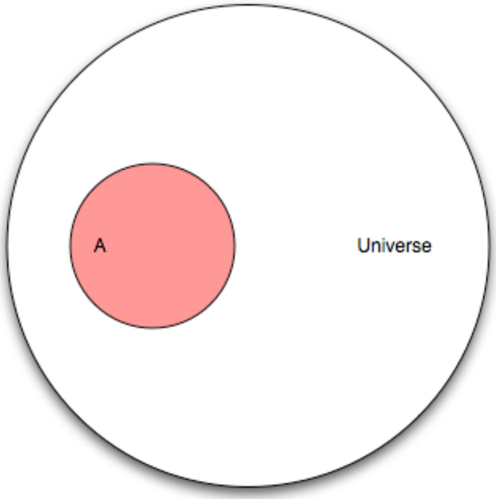


Notes on KNN

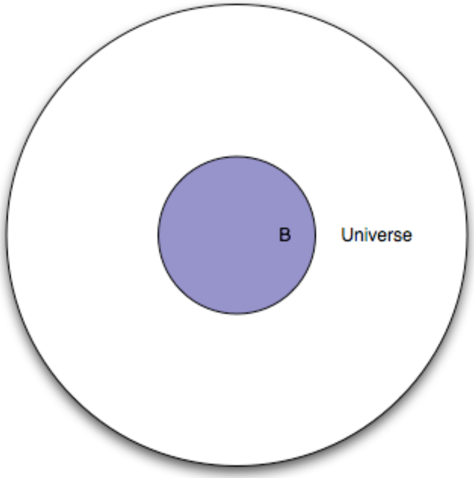
- Since the algorithm is distance based , consider normalising distances
- Votes can be weighted in various ways , weighing by distance is most popular
- Low number of K, captures heavily localised patterns and can lead to overfit
- Very high value of K, captures broad level patterns and can miss out on complex patterns
- There is no equation/output for the model , training data itself is the model
- Standalone KNN is not a very powerful algorithm

Naive Bayes

Bayes Theorem



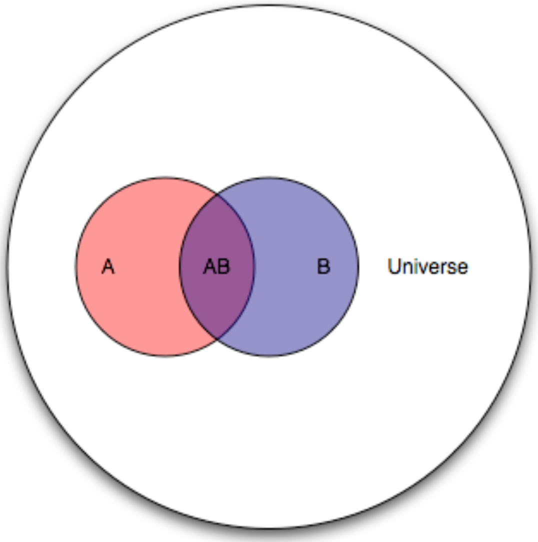
$$P(A) = \frac{|A|}{|U|}$$



$$P(B) = \frac{|B|}{|U|}$$

Reference : https://oscarbonilla.com/2009/05/visualizing-bayes-theorem/

contd..



$$P(AB) = \frac{|AB|}{|U|}$$

$$P(A|B) = \frac{|AB|}{|B|}$$

$$P(A|B) = \frac{\frac{|AB|}{|U|}}{\frac{|B|}{|U|}}$$

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Reference : https://oscarbonilla.com/2009/05/visualizing-bayes-theorem/

Bayes theorem statement

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Extension of Bayes

$$P(A|B_1 \cap B_2 \cap B_3 \cap ..) = \frac{P(B_1 \cap B_2 \cap B_3 \cap ..|A) * P(A)}{P(B_1 \cap B_2 \cap B_3 \cap ..)}$$

Naive Bayes

$$P(A|B_1 \cap B_2 \cap B_3 \cap ..) = \frac{P(B_1|A) * P(B_2|A) * P(B_3|A) .... * P(A)}{P(B_1 \cap B_2 \cap B_3 \cap ..)}$$

Example

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other Fruit	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Reference : https://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification

Contd..

```
P(Banana)       = 0.5 (500/1000)
P(Orange)       = 0.3
P(Other Fruit) = 0.2
```

```
p(Long)   = 0.5
P(Sweet)  = 0.65
P(Yellow) = 0.8
```

```
P(Long|Banana) = 0.8
P(Long|Orange) = 0  [Oranges are never long in all the fruit we have seen.]
 ....

P(Yellow|Other Fruit)     =  50/200 = 0.25
P(Not Yellow|Other Fruit) = 0.75
```

Reference : https://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification

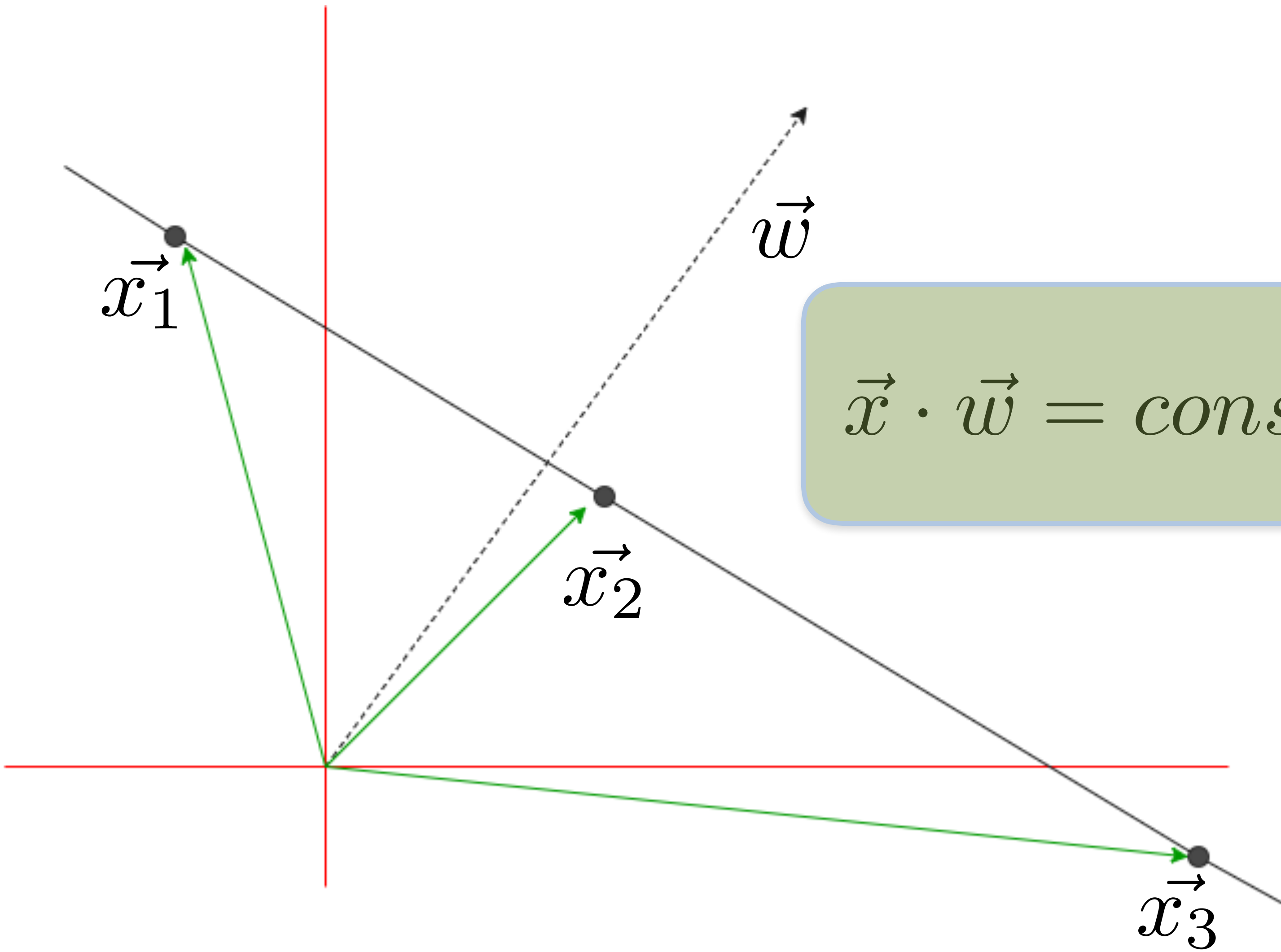
Contd..

```
P(Banana|Long, Sweet and Yellow)
      P(Long|Banana) * P(Sweet|Banana) * P(Yellow|Banana) * P(banana)
    = _______________________________________________________________
                      P(Long) * P(Sweet) * P(Yellow)

    = 0.8 * 0.7 * 0.9 * 0.5 / P(evidence)

    = 0.252 / P(evidence)


P(Orange|Long, Sweet and Yellow) = 0


P(Other Fruit|Long, Sweet and Yellow)
      P(Long|Other fruit) * P(Sweet|Other fruit) * P(Yellow|Other fruit) * P(Other Fruit)
    = ____________________________________________________________________________________
                                          P(evidence)

    = (100/200 * 150/200 * 50/200 * 200/1000) / P(evidence)

    = 0.01875 / P(evidence)
```

Reference : https://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification

Support Vector Machines

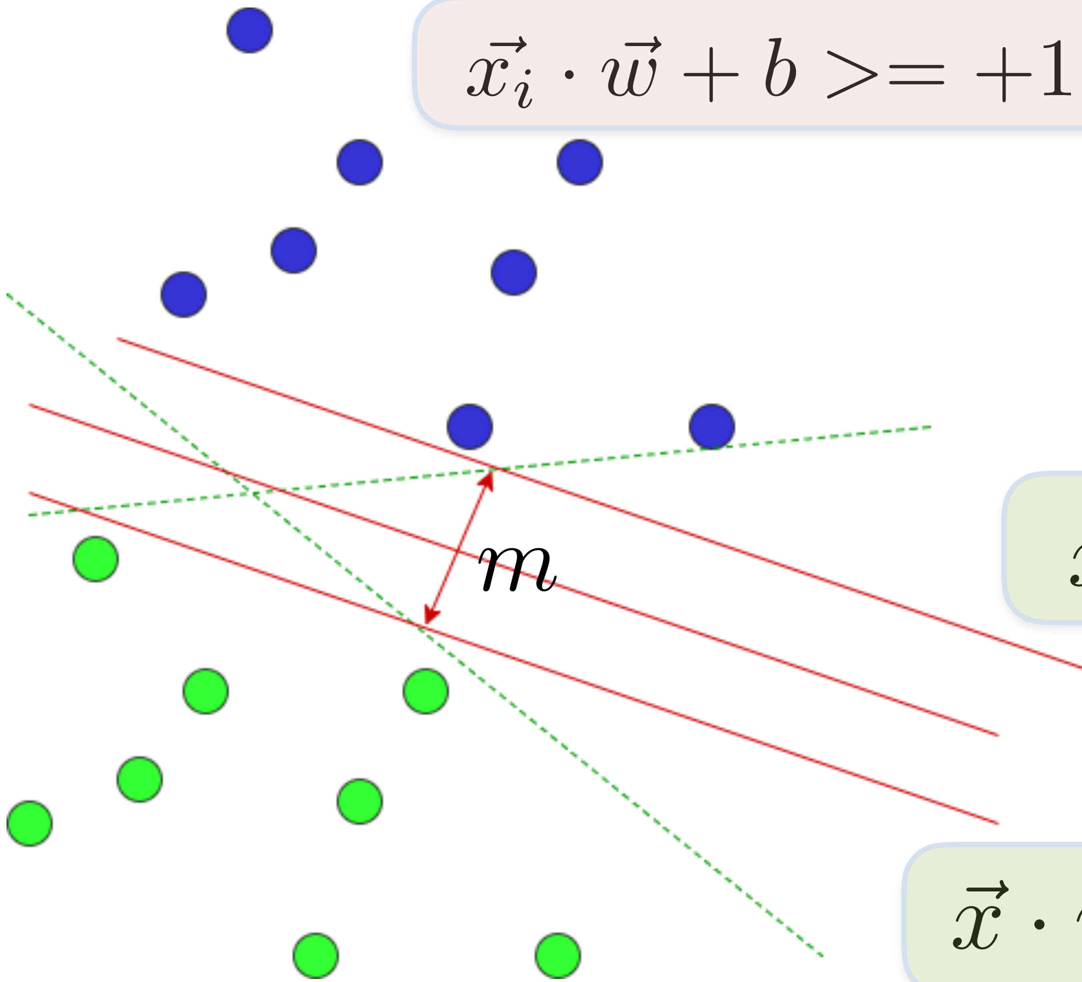
Vector Equation of A line



$$\vec{x} \cdot \vec{w} = constant$$

Idea behind svm

$$\vec{x_i} \cdot \vec{w} + b >= +1 \quad \forall x_i \mid y_i = +1$$



$$\vec{x} \cdot \vec{w} + b = +1$$

$$\vec{x} \cdot \vec{w} + b = -1$$

$$\vec{x_i} \cdot \vec{w} + b <= -1 \quad \forall x_i \mid y_i = -1$$

Objective Function & Constraints

$$m = \frac{2}{||\vec{w}||}$$

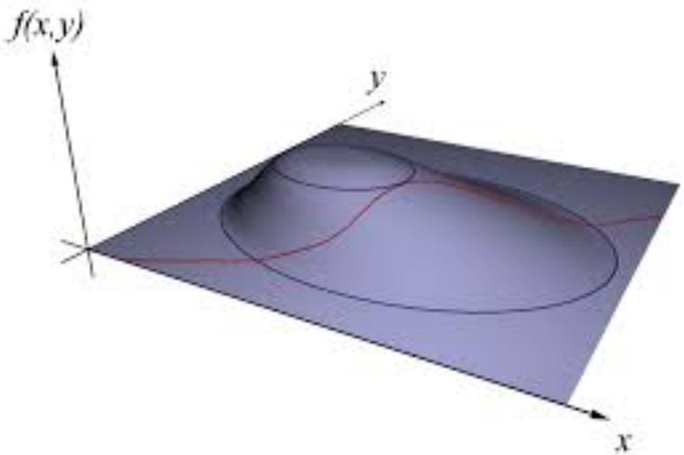
Objective Function

$$\frac{1}{2}||\vec{w}||^2$$

Constraints

$$y_i * [\vec{x_i} \cdot \vec{w} + b] >= +1 \quad \forall x_i$$

Optimisation with constraints : Lagrange's Multiplier



$$\nabla f(x, y, z) = \lambda \nabla g(x, y, z)$$
$$g(x, y, z) = k$$

Example

$$find\ values\ of\ x\ and\ y\ for\ which\ (5x-3y)\ takes\ its\ max/min\ value\ under\ constraints\ x^2+y^2=136$$

New Objective Function

$$5x-3y-\lambda*(x^2+y^2-136)$$

$$5=2\lambda x$$
$$-3=2\lambda y$$
$$x^2+y^2=136$$

$$\lambda^2=\frac{1}{16} \quad \Rightarrow \quad \lambda=\pm\frac{1}{4}$$

If $\lambda=-\frac{1}{4}$ we get,

$x=-10$ $\quad y=6$

and if $\lambda=\frac{1}{4}$ we get,

$x=10$ $\quad y=-6$

In context of svm

New Objective Function

$$\frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{n} \alpha_i * [y_i * (\vec{x_i} \cdot \vec{w} + b - 1)]$$

$$\vec{w} = \sum_{i=1}^{n} \alpha_i * \vec{x_i} * y_i$$

$$\sum_{i=1}^{n} \alpha_i * y_i = 0$$

Dual

$$max \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\alpha^T H \alpha \right]$$

$$s.t.\ \alpha_i \geq 0\ \forall_i\ and\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

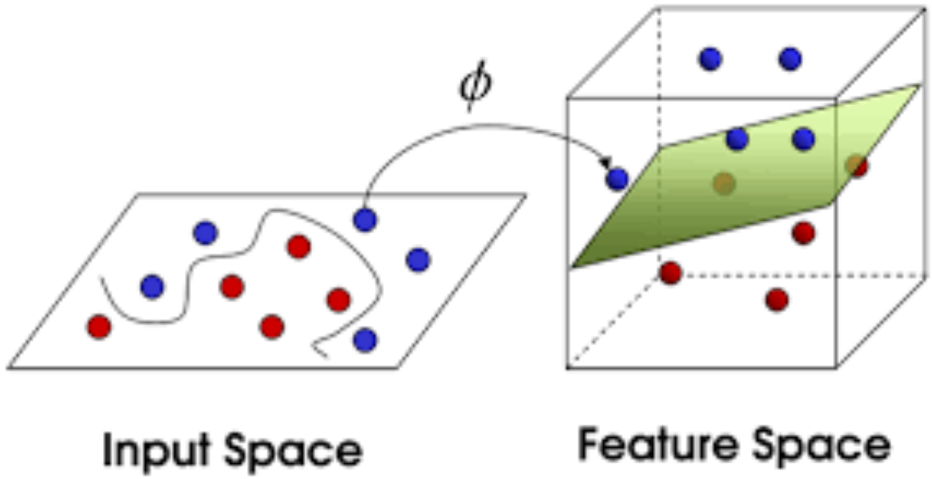
$$where\ H_{ij} = y_i y_j \vec{x_i} \cdot \vec{x_j}$$

Overlap Case

$$\frac{1}{2}||\vec{w}||^2 + C * \sum_{i=1}^{n} \epsilon_i - \sum_{i=1}^{n} \alpha_i * [y_i * (\vec{x_i} \cdot \vec{w} + b) - 1 + \epsilon_i] - \sum_{i=1}^{n} \mu_i \epsilon_i$$

for complete mathematical discussion : refer to svm paper uploaded in LMS

Nonlinear Separation Line and Kernel Function



Kernel Functions

RBF

$$k(\vec{x_i}, \vec{x_j}) = \exp^{-\left(\frac{||\vec{x_i} - \vec{x_j}||^2}{2\sigma^2}\right)}$$

Polynomial

$$k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + a)^b$$

Sigmoid

$$k(\vec{x_i}, \vec{x_j}) = tanh(a\vec{x_i} \cdot \vec{x_j} - b)$$

SVM and probabilities

- There is no concept of probabilities in svm
- Separation is based on distance from the separating plane
- Normalised distance can be used as ad-hoc probabilities
- you'll need to use specific options in sklearn implementation to forcefully obtain probabilities
- Multiclass svm , simply uses multiple binary svm (One vs All) internally

Lets see it in action in Python