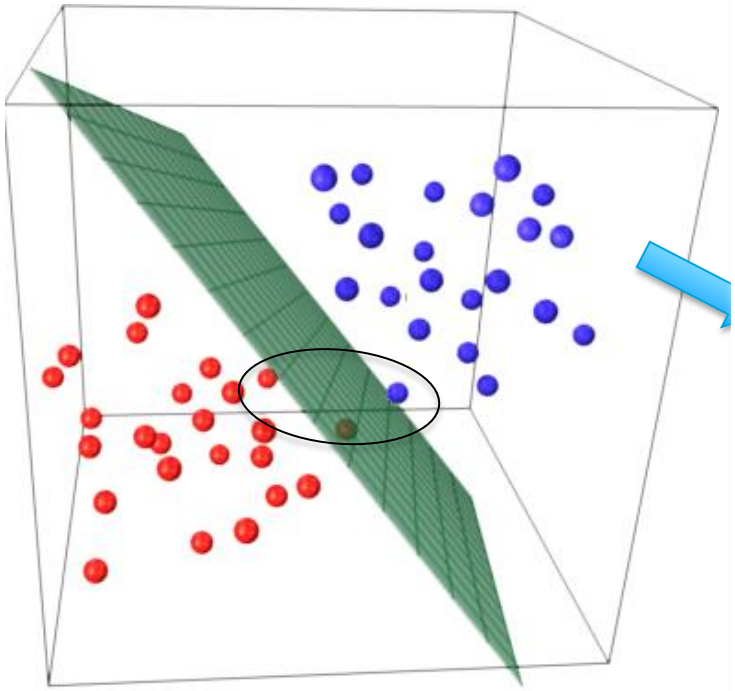
Support Vector Machines

Support Vector Machines

1. Known as maximum-margin hyperplane, find that linear model with max margin. Unlike the liner classifiers, objective is not minimizing sum of squared errors but finding a line/plane that separates two or more groups with maximum margins

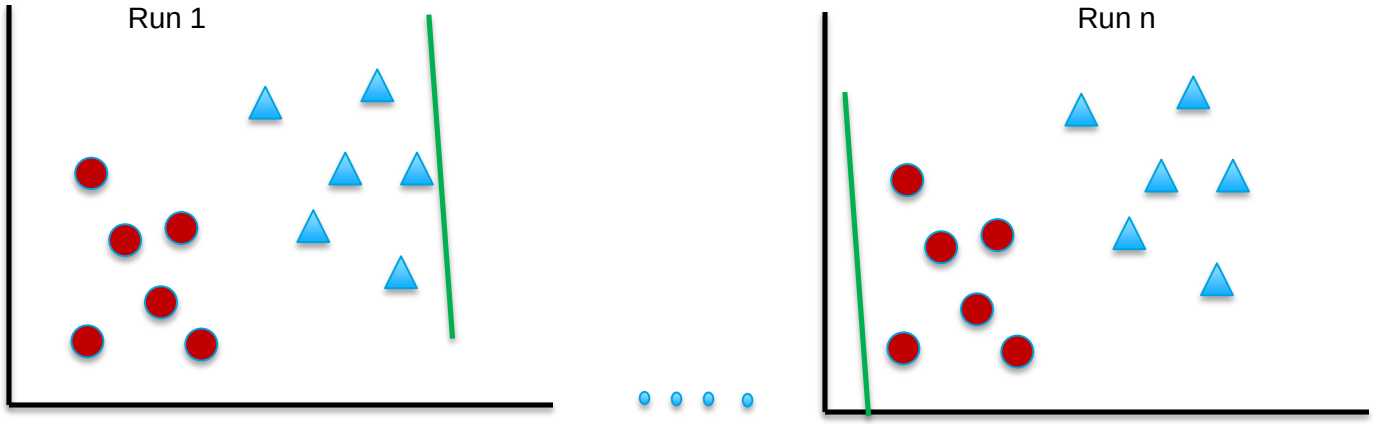


http://stackoverflow.com/questions/9480605/what-is-the-relation-between-the-number-of-support-vectors-and-training-data-and

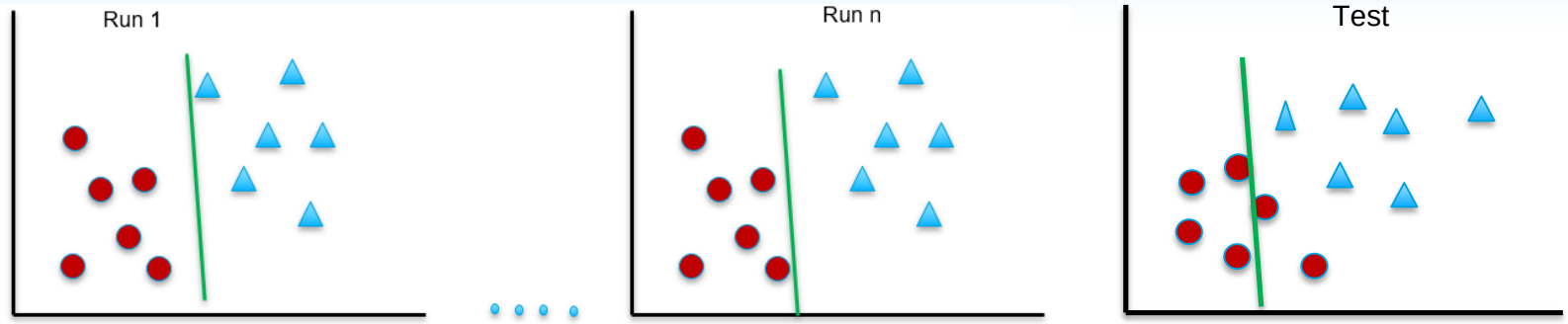
Perceptron

Perceptron Learning Algorithm –

1. Select random sample from training set as input. Draw the first random line (green) such that blue triangles lie above it and red circles ones below
2. If classification is correct, do nothing. But first time many blue triangles are on wrong side!
3. If classification is incorrect, modify the weight vector w and shift the green line
4. Repeat this procedure until the entire training set is classified correctly
5. Howsoever times we run this algorithm, it will find a surface which separates the two classes



Supervised Machine Learning



6. Convergence theorem guarantees that when the classes are linearly separable in the training set, perceptron will find that surface which separates the two classes correctly

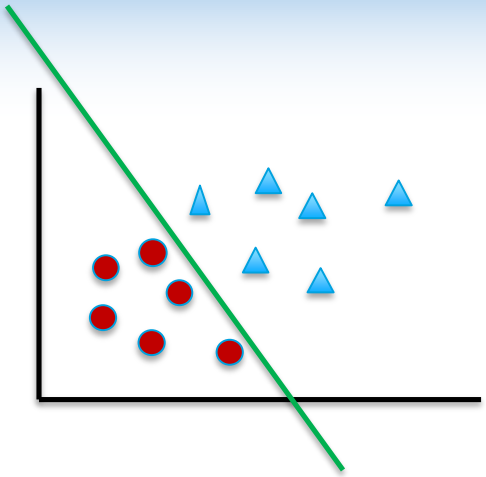
7. However, what we need is a perceptron which will correctly classify in test set. Not just training set

8. The perceptron algorithm does not guarantee it will be able to separate the two classes correctly even when the classes are linearly separable

9. Why? Because it does not look for an optimal plane. It stops the moment it finds the separator plane (a.k.a dichotomizer).

10. Since the planes, are passing very close to the data points in the training set, it may not perform well in test set where the distribution of the data will be different

Supervised Machine Learning



11. Formally the properties of data is

    Given D = { (xi , yi) / xi E R^n , yi E {-1 , +1}} for i = 1 to m Note “E” means “belongs to”

    Given xi, yi in a data set D, xi being defined with real values in “n” dimensions, corresponding yi (label) is assigned value -1 for data belonging to one class and +1 to data belonging to\ the other class

12. With the formal definition of the data, the objective is to find a plane which will segregate the two classes such that all -1 lie on one side and all +1 lie on the other and keep maximum margins on both sides (to ensure it performs equally well in unseen data)

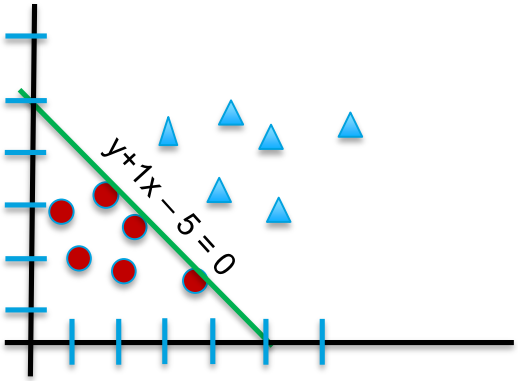
13. The line makes all blue data points +1 and red ones -1

14. Substitute (2,2) i.e. red in the equation (2 + 1*2 -5) returns – number

15. Similarly substitute 5,3 (blue) , (5 + 1*3 – 5) returns positive value.

16. All on right side are + and left –ve. A correct classification but the line is passing too close to data points!

17. There are infinite lines(each corresponding to a small change in the slope) possible that can cleanly separate the two classes of data points

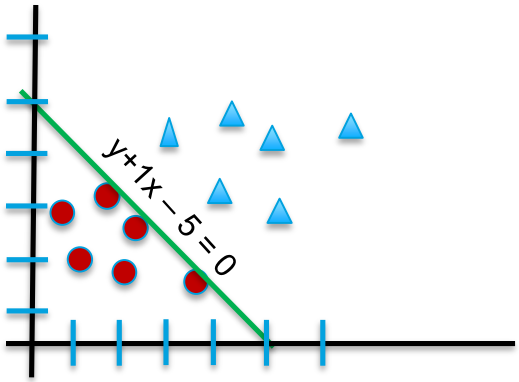


18. The problem is how do we find out the best / optimal plane from the infinite!

19. We need a way to compare the different planes. We have already seen that when we substitute the coordinate values of a datapoint in the equation, we get a positive or negative number.

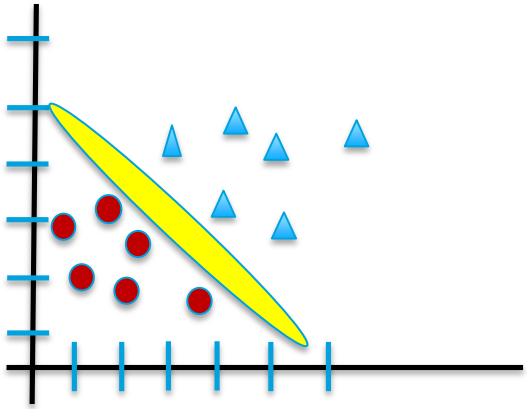
20. When we substitute the coordinates of the points into the equation of a line such as (y + 1x – 5 = 0), we get back a positive or negative number. In the pic below the farthest blue and red points are (7,6) and (1, 2) with returned value of 8 and -2 respectively

21. All data points will return a number whose magnitude will be greater than zero except when the point is on the line. Thus we can treat the returned number as a distance of the point from the line



22. Often line or a plane is represented as y = mX + b. This representation is useful when 'y' is dependent on (influenced by) the value of X and we wish to represent the relation between 'y' and 'X' i.e the pattern in form of a equation

    a) 'y' is the dependent variable whose value depends on the values of 'X', 'm' and 'b'
    b) 'm' is called a slope / coefficient or a collection of slopes / coefficients. (we will call it slope)
    c) 'b' is the intercept (value of 'y' when all coefficients are zero)
    d) 'X' represents the independent variables that impact value of 'y'



23. In classification, we do not have any dependent variable with continuous value that we need to predict. Instead we have classes with data points scattered in the feature space. So, where is the pattern?

24. We started by assuming the classes are linearly separable! That means the space separating the data points from the two classes has a linear shape, a linear pattern (yellow space). It is that pattern we are trying to capture in form of a linear equation. Thus the axis in the feature space are all independent variables.

25. By adjusting their slopes we are trying to find a line that represents the separating space.

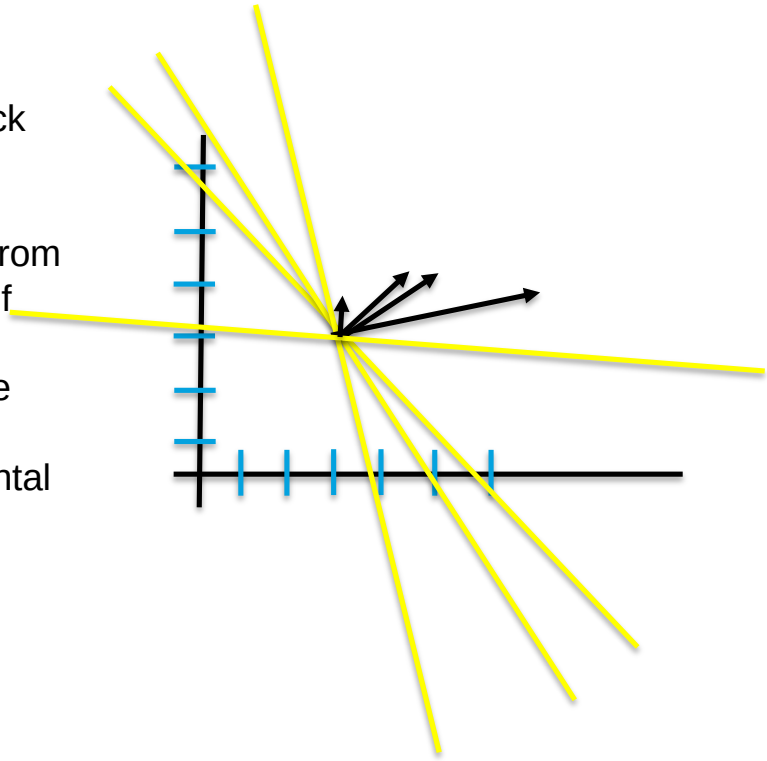
26. The slope is also represented by a normal vector (black arrow which is always perpendicular to the line)

27. The normal vector is usually represented as a vector from the origins, here it is shown on the respective line itself

28. In the animation the slope of the yellow line is negative

29. As the line becomes more and more parallel to horizontal axis, the magnitude of the normal vector becomes 0

30. As the line becomes parallel to the vertical axis, the magnitude of the normal vector becomes infinite



Supervised Machine Learning

31. Every data point can also be considered as a vector, a data vector

32. Data vector (blue and purple arrows) originates from the origin as does the slope vector w (black) of the line (1y+1x-5=0)

33. When we substitute value of the data point (4,6) into the equation of the line, we are doing a multiplication a.k.a dot product that results in a number … 1*6 + (1*4) – 5 = 5.

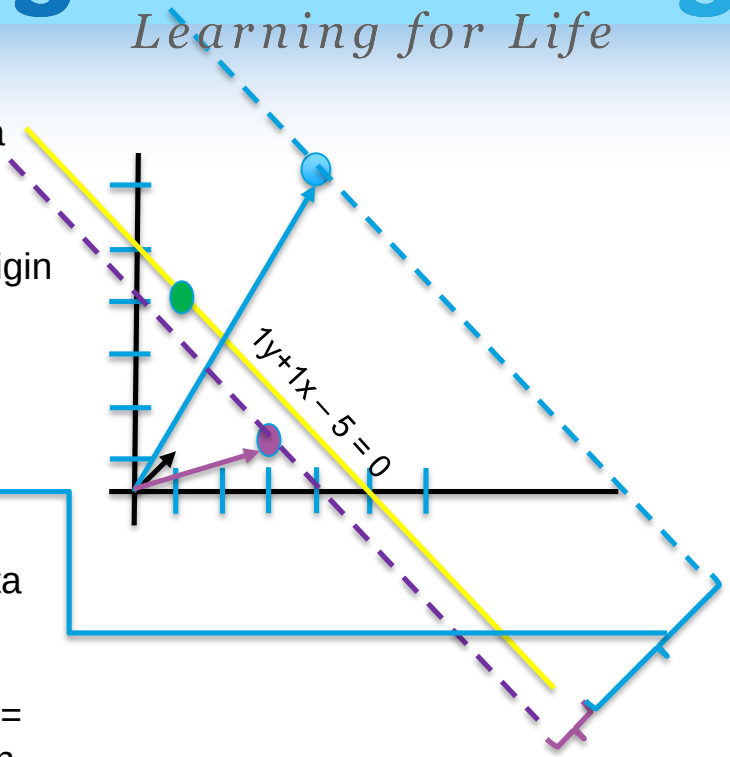
34. Dot products are calculated using matrix algebra where data vector and slope vector are stored in separate matrices

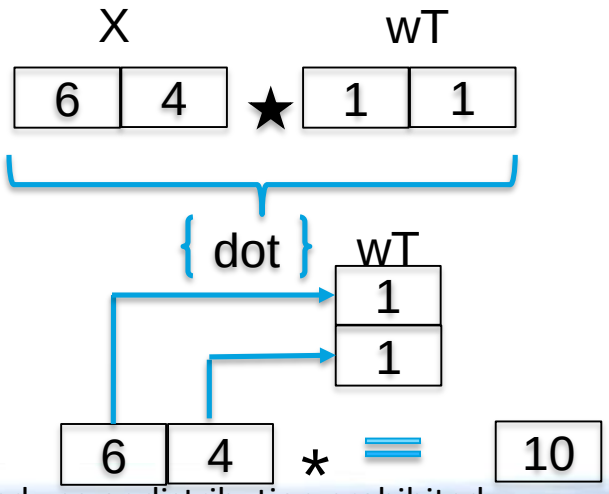
35. The dot product is shown as wT * X. Here X = (4,6) and w = (1,1), a row. wT (is transpose of w matrix which is a column

36. Thus the entire operation becomes wT * X – 5 = 5 for blue data vector. For the purple data vector (3,1), wT * X – 5 = 1*1 + 1*3 – 5 = -2

37. For data points lying exactly on the line, e.g. (1, 4), the dot product will return 0

38. dot product uses slope vector to rescale the data vector and gives the vertical distance between a parallel line through the scaled point and the given line





Supervised Machine Learning

greatlearning
Learning for Life

In [35]:
```
%matplotlib inline
import numpy as np
import pandas as pd

df1 = pd.DataFrame(np.random.randint(1,10, size=(2,2)), columns=['A', 'B'])

df1
```

Out[35]:

	A	B
0	5	9
1	2	8

In [38]:
```
df2 = pd.DataFrame(np.random.randint(1, 4, size=(1,2)), columns=['A', 'B'])
```

In [39]:
```
df2  # Assume df2 is the slope coefficients or  w of some line
```

Out[39]:

	A	B
0	2	1

In [41]:
```
df2.T
```

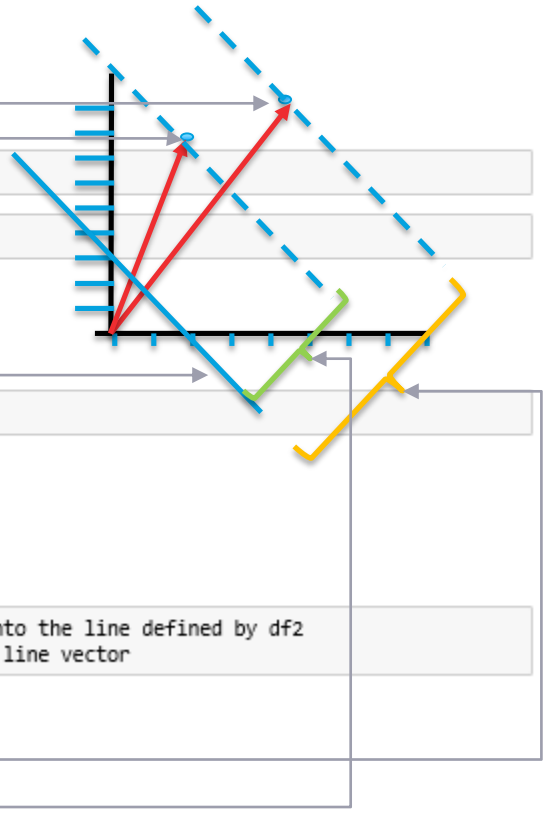
Out[41]:

	0
A	2
B	1

In [40]:
```
df1.dot(df2.T)   # This operation is substituting values  of each data point into the line defined by df2
                 # you can think of it as projection of the data vector on the line vector
```

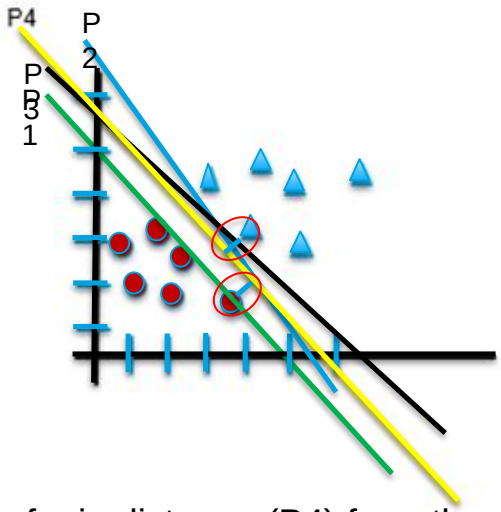
Out[40]:

	0
0	19
1	12



Note: the numbers are not truly distance of the point from the line. They cannot be represented geometrically. We only take them as functional margin i.e. margin from a functional calculation

39. Now we have a way of comparing the planes. Planes can be arranged in descending order of the distance from the nearest point (min distance). **Note:** a data point with -5 is physically farther away from line than say a point with -2. We wish to pick -2 as min distance not -5. Hence we go for magnitude.



Plane	Ranked on descending order of distance from nearest point
P4	D4
P2	D3
P3	D2
P1	D1

40. The plane with max of min distance (P4) from the nearest point is the most optimal plane. This plane is likely to perform relatively better than other planes
    a) The distance between nearest data point and a plane is called margin of the plane which is the least of **yi(yi -mXi +b)** where i ranges from 1 to n (number of data points)

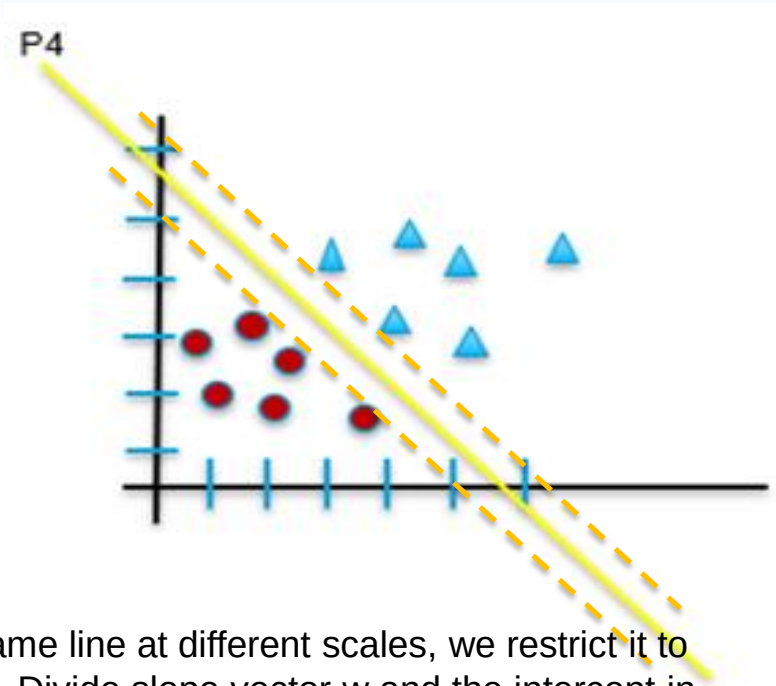
    b) The equation of line is multiplied by label value (yi) to ensure right plane is selected when going for magnitude based distance. This is called **functional margin** of the plane due to the point (Xi, yi) which is closest.

41. The challenge is, we can have infinite planes and millions of points to evaluate each on… Brute force method may take age of the universe to find the most optimal plane

Supervised Machine Learning

greatlearning
Learning for Life

42. Given the optimal plane is maximally separated from the data points from the two classes, we can look at the situation as shown



43. The objective thus is to find a plane which has maximum functional margins. Margin is a no-man's land. No data points should be inside the margin (hard margins).

44. However, while comparing the planes, we know that line (1*y+(1* X -5) = 0) is same as line (10*y + 10*X – 50 = 0) is the same as (100*y + 100*X – 500 = 0)… i.e. all lines can be represented at different scales. The algorithm should not waste time comparing a plane to itself!

45. To prevent the algorithm from unnecessarily evaluating same line at different scales, we restrict it to evaluate only that plane whose slope vector is unit length. Divide slope vector w and the intercept in the equation by the magnitude of the slope vector

46. Magnitude of a slope vector w is represented as ||w|| (magnitude a.k.a norm of the vector). ||w|| is derived using Euclidian formula Sqrt(W1^2 + W2^2) where W1 and W2 are the length of the vector projection on dimension x1 and x2. Suppose we take 10*y + 10*X – 50 = 0. ||w|| = sqrt(10^2 + 10^2) = 10.

47. Thus 10*y + 10*X -50 = 0 ➔ 10/||w|| *y + 10/||w||*X - 50 / ||10|| = 0 ➔ 1*y + 1*X – 5 = 0 (unit vector line)

Supervised Machine Learning

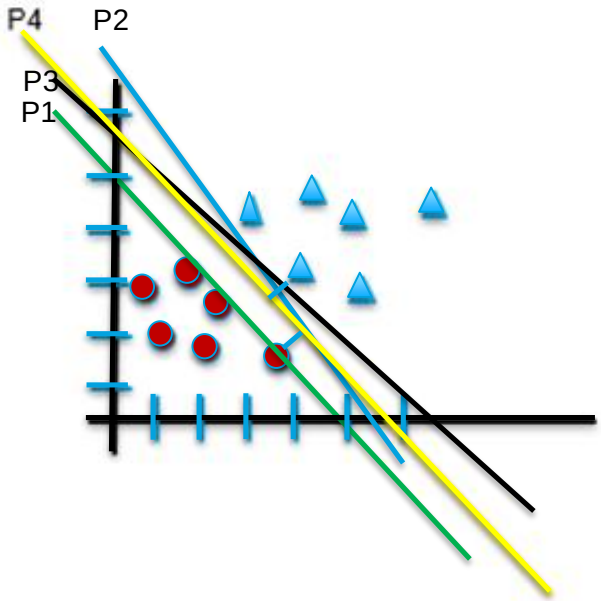
greatlearning
Learning for Life

48. Thus the expression for the functional margin of a point becomes yi(yi***w**/||w|| + Xi***w**/||w|| - b/||w||)

49. This is called **geometric margin** of a point as it can be derived geometrically. GM = F / ||w|| (F is functional margin)

50. The objective now is to find a plane in terms of w and b which has maximum geometric margin given the data points

51. To find the w and b we need to max geometric margin M with respect to w and b with the constraint that mi (distance of point i from the plane) for each data point is greater than or equal to M



$$\underset{\mathbf{w},b}{\text{maximize}} \quad M$$
$$\text{subject to} \quad \gamma_i \geq M, \; i = 1, \ldots, m$$

52. The constraint will help find the point of least distance for the plane defined by w and b

53. Since GM = FM / ||w||, for a given w, the constraint is fi > F where fi is functional margin of each point and F contains the least distance

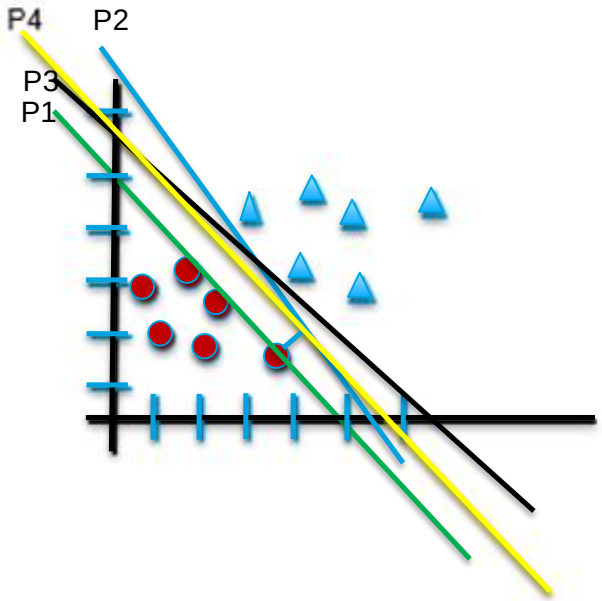
54. In simple terms, this is like a for next loop for I = 1 to n where if fi < F, replace F with fi else next i. This will pocket the least fi eventually before the loop is over.

55. Let this least fi stored in F be 1 unit.

56. Thus, the objective now is to maximize 1/||w|| with respect to w, b (across all possible planes) with the constraint that fi > 1, for all data points i = 1 to n

57. This is equivalent to minimize ||w|| with respect to w, b subject to yi(w.xi) + b >= 1 for i = 1 to n

58. Let the first line be P2 (w2, b2), min distance be 0.5. Let this be 1 unit. The next best plane will be the one for which the functional margin of all data points will be greater or equal to 0.5. **Note** we do not want a plane whose functional margin is less than 0.5… such plane will not be better than P2. We want a plane which is better than P2



59. In the next loop, suppose we find P1 (w1,b1). We find that at least one red data point is much close than P2 GM. Thus we stop evaluating this plane and move to next one

60. Instead of finding the least margin in P1 and then comparing the min distance of P1 with min distance of P2, compare the functional margins of every data point from P1 to min functional margin of P2

61. Replace GM with min functional margin of P1 and let us treat this as 1

62. Repeat the steps 59 - 61 till we find the plane with max margins

63. The objective function can now be represented as min ½(||w||^2 with respect to (w,b) subject to the condition that functional margin of all data points i =1 to n, yi(w.xi) + b >=1 for subsequent planes

64. The factors of ½ and square term are included to make later calculations easier

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} && \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} && y_i(\mathbf{w}\cdot\mathbf{x}_i)+b \geq 1, \quad i=1,\ldots,m \end{aligned}$$

65. This is convex quadratic optimization problem. Easier to solve than the original constraints

66. Here we introduce Lagrangian function which combines the two problems into one unconstrained problem

$$\mathcal{L}(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\alpha_i\left[y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1\right]$$

67. A new term (Lagrangian multiplier) represented by alpha is introduced for each data point. The multiplier can be considered as a weight associated with each point

68. To solve the Lagrangian expression we need to minimize with respect to “w,b” and maximize with respect to alpha

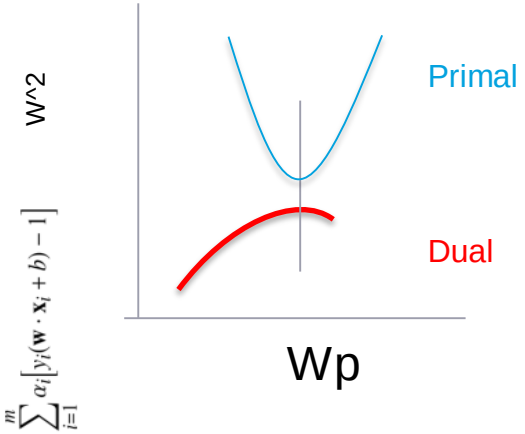
69. **Wolfe Dual Problem** – The lagrangian has m inequality constraints where m is number of data points in the data set. Such problem is typically solved **using it's dual form**

70. The duality principle says that this problem can be viewed in two perspectives. First is the primal, a minimization in this case and the other one is maximization which is dual in this case

71. The max of the dual will always be less than the min of the primal. It sets a lower bound to the solution of the primal

72. In the case of SVM, the minimization problem (primal problem) is convex (quadratic and hence a global minima guaranteed) and the primal problem is affine (a linear equation)

73. Given that, **Slater's** conditions says that the maximum of dual will be equal to minimum of primal i.e. both solutions will be the same (Strong Dual case)

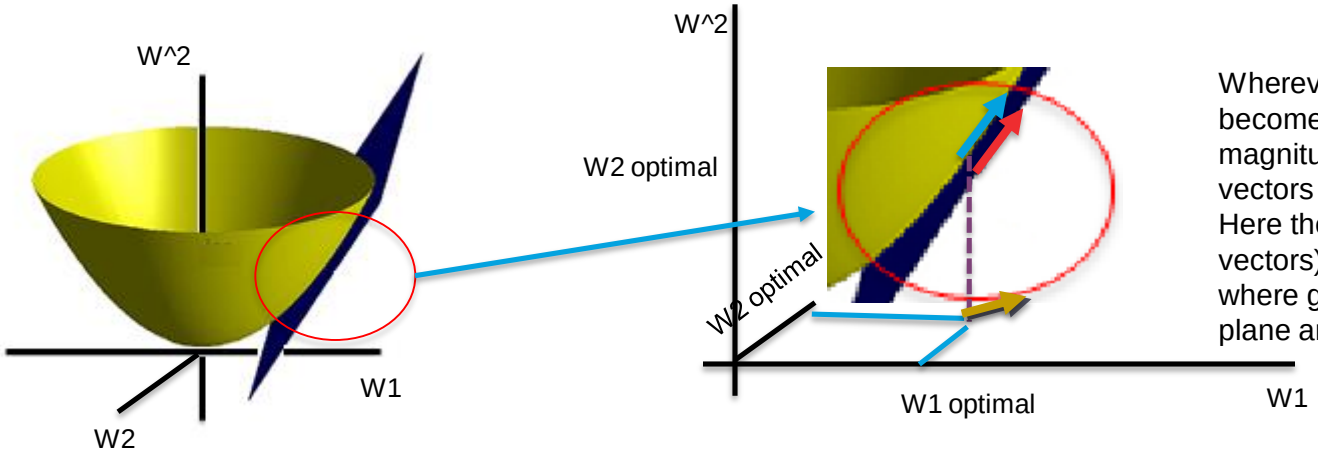


Lagrange Method for Function Optimization

1. When faced with a situation where we need to find the local minima/maxima of a function subject too constraints (i.e. restrict the region in MS within which to look for), Guiseppe Lodovico Lagrangia a.k.a Joseph Loui Lagrange, invented a strategy of finding such local maxima and minima in a non-parameterized way. This method is called Lagrange Multiplier

2. When we try to solve an optimization problem of the form – minimize f(x) given x subject to condition g(x) = 0, Lagrange noticed that the minimum is found at that value of x where the gradient point in the same direction for both f(x) and g(x)

3. Gradient is a vector i.e. combination of direction and magnitude. In our minimization and constraint function it can be shown as



Wherever the gradient vectors become same in direction and magnitude. Technically gradient vectors is shown on input space. Here the blue and red arrow (slope vectors) are showing the point where gradient is same for both plane and the convex function

SVM Mechanism

4. Lagrange noticed that the min of first is found when gradient vector points in same direction as gradient of g. In other words

$$\nabla f(\mathbf{x}) = \alpha \nabla g(\mathbf{x})$$

5. Note, the gradient vector is shown as golden and black arrows on the base of the weight space. Both are pointing is same direction

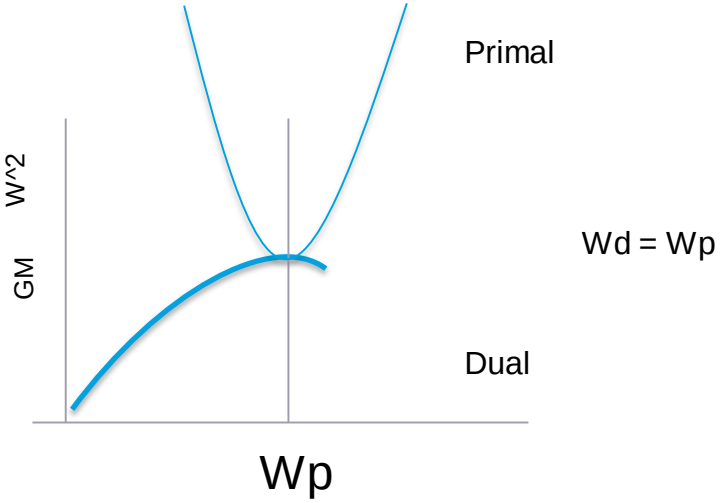
6. Therefore we have to solve the problem $\nabla f(\mathbf{x}) - \alpha \nabla g(\mathbf{x}) = 0$

7. The gradient of the two functions could be as shown in next slide

SVM Mechanism

8. In the case of SVM, the minimization problem (primal problem) is convex (quadratic and hence a global minima guaranteed) and the primal problem is affine (a linear equation)

9. Given that, **Slater's** conditions says that the maximum of dual will be equal to minimum of primal i.e. both solutions will be the same (Strong Dual case)



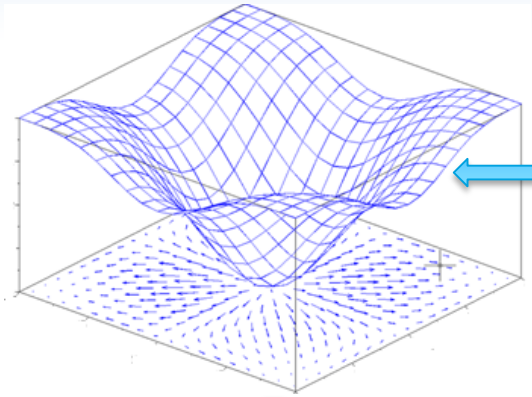
To solve the primal, we use the familiar partial differential based approach on the Lagrangian

Note: The minima of the primal in the solution may not be global minima

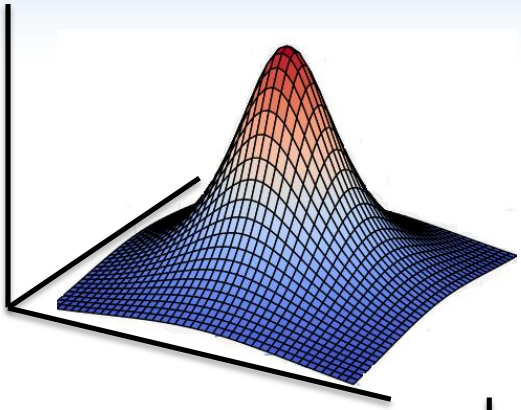
Supervised Machine Learning

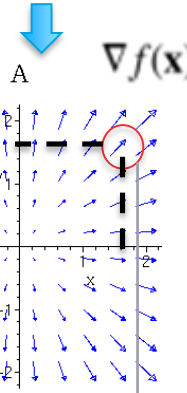
greatlearning
Learning for Life

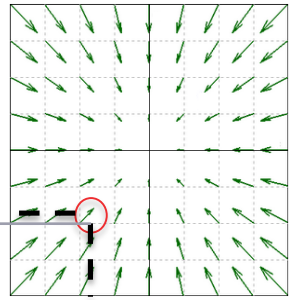
SVM Mechanism

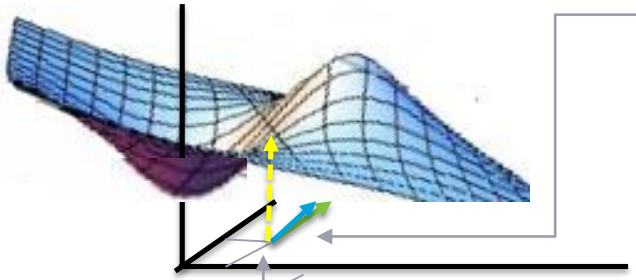


$$\frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right]$$









Combination of w and b where gradient of both minima and maxima point in same direction
The Lagrange variable alpha

SVM Mechanism

10. To solve the primal, we use the familiar partial differential based approach on the Lagrangian. The optimization problem takes a form -

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

This expression becomes important for us to understand another concept called Kernel tricks used by SVM!

SVM Mechanism

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$
$$= \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

Lagrangian

The Lagrangian primal problem is:

$$\min_{\mathbf{w},b} \max_{\alpha} \quad \mathcal{L}(\mathbf{w}, b, \alpha)$$
$$\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m$$

The dual form

Solving the minimization problem involves taking the partial derivatives of $\mathcal{L}$ with respect to $\mathbf{w}$ and b.

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

1

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

Partial derivative of the lagrangian

From the first equation, we find that:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Let us substitute $\mathbf{w}$ by this value into $\mathcal{L}$:

$$W(\alpha, b) = \frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j\right) - \sum_{i=1}^{m} \alpha_i \left[y_i \left((\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_i + b\right) - 1\right]$$
$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m} \alpha_i y_i \left((\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_i + b\right) + \sum_{i=1}^{m} \alpha_i$$
$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i$$
$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b\sum_{i=1}^{m} \alpha_i y_i$$

We note that $\frac{\partial \mathcal{L}}{\partial b} = 0$ implies that $\sum_{i=1}^{m} \alpha_i y_i = 0$. As a result, the last term is equal to zero, and we can write:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

This is the **Wolfe dual Lagrangian function**.

The optimization problem is now called the **Wolfe dual problem**:

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$
$$\text{subject to} \quad \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

Support Vector Machines Soft Margins

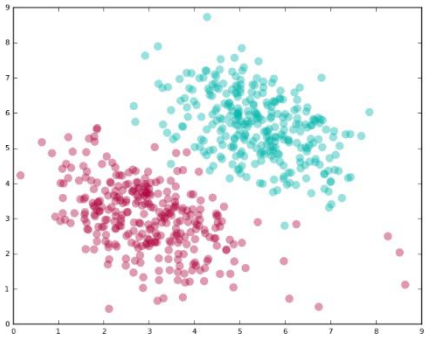
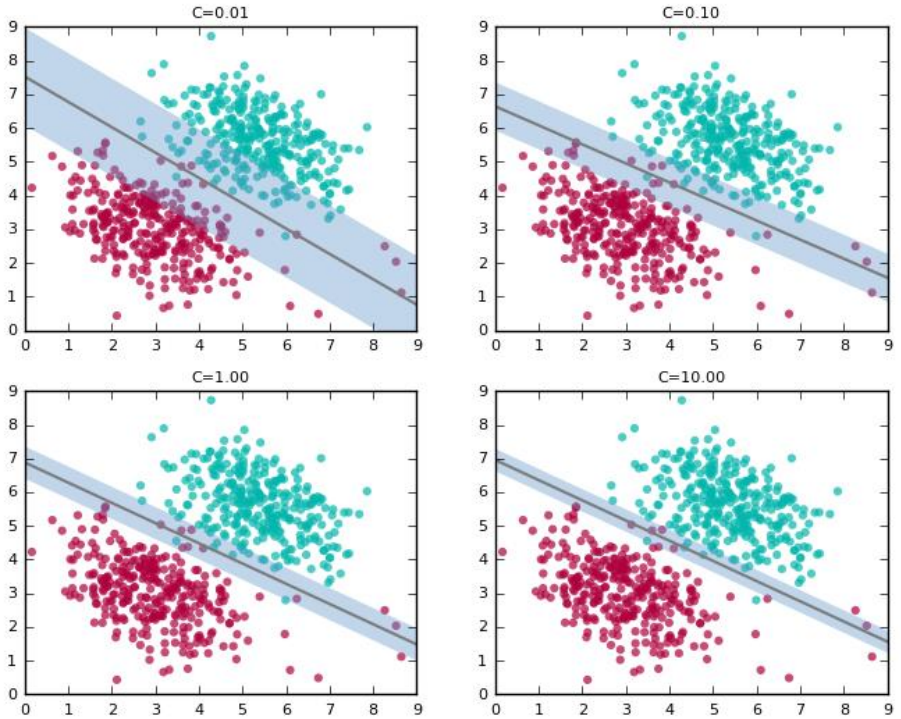


Image Source : https://dzone.com/articles/support-vector-machines-tutorial



1. Data in real world is typically not linearly separable.
2. There will always be instances that a linear classifier can't get right
3. SVM provides a complexity parameter, a tradeoff between: wide margin with errors or a tight margin with minimal errors. As C increases, margins become tighter

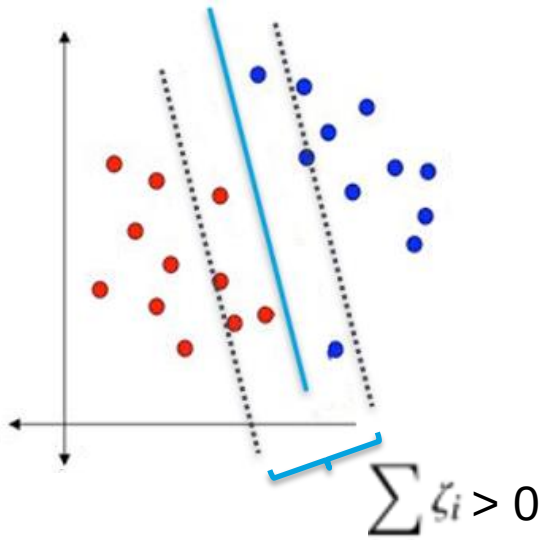
SVM For linearly separable data (Soft Margins)

The optimization problem is changed to

$$\underset{\mathbf{w},b,\zeta}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m}\zeta_i$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \zeta_i \quad \text{for any } i = 1,\ldots,m$$

A slack variable is introduced which allows data points to be on the wrong side of the margins



The slack for a data point is magnitude of the violation from respective margins.

Slack for data points on the correct side of the margins = 0

Objective is to minimize overall sum of slack across all data points

Support Vector Machines Soft Margins

To give user a control over the slack variable while building a model, another term 'C' is introduced and the optimization looks like

$$\begin{aligned} \underset{\mathbf{w},b,\zeta}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \zeta_i \\ & \zeta_i \geq 0 \quad \text{for any } i = 1, \ldots, m \end{aligned}$$

'C' helps us decide how important the slack is. C=0 means no constraint , When 'C' is high , stricter the margins

Image Source :https://mubaris.com/posts/svm/

Support Vector Machines Linearly Non Separable Data

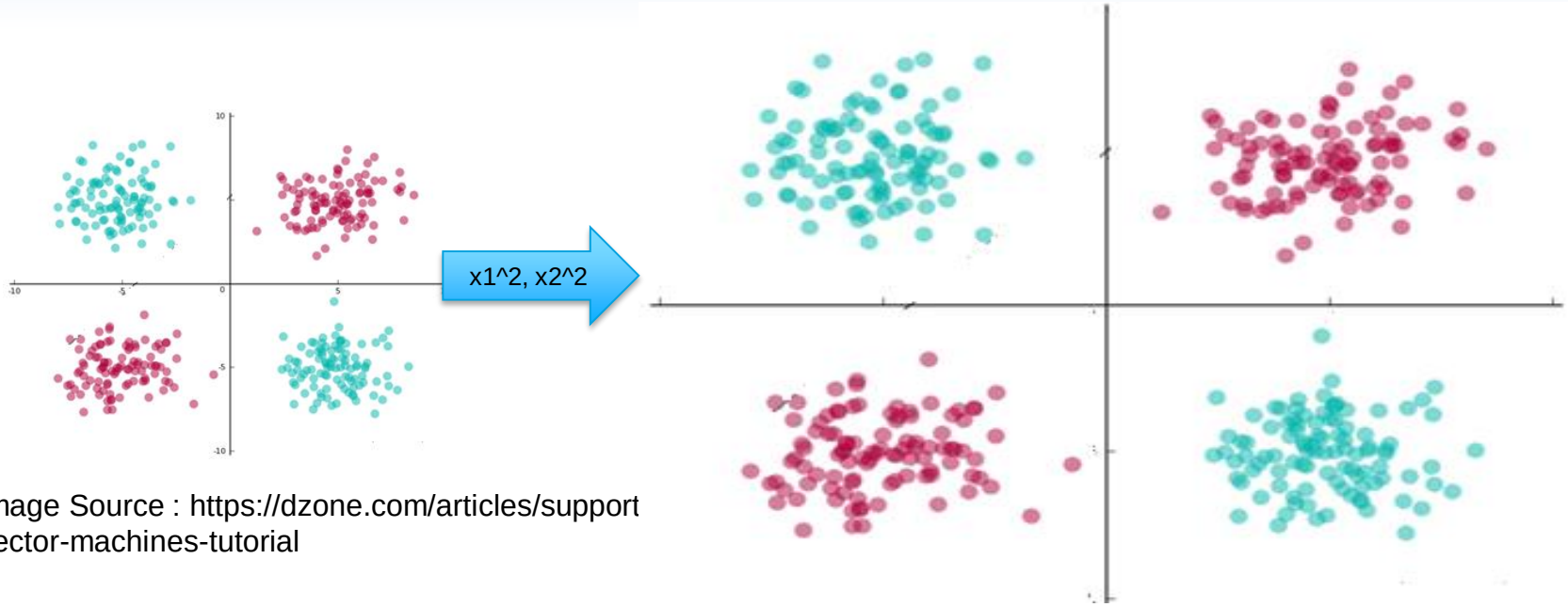


Image Source : https://dzone.com/articles/support vector-machines-tutorial

1. When data is not linearly separable, SVM uses kernel trick to make it linearly separable
2. This concept is based on **Cover's theorem** "given a set of training data that is not linearly separable, with high probability it can be transformed into a linearly separable training set by projecting it into a higher-dimensional space via some non-linear transformation"
3. In the pic above, replace x1 with x1^2, x2 with x2^2 and create a third dimension x3 = sqrt(2x1x2)

greatlearning
Learning for Life

Support Vector Machines Linearly Non Separable Data

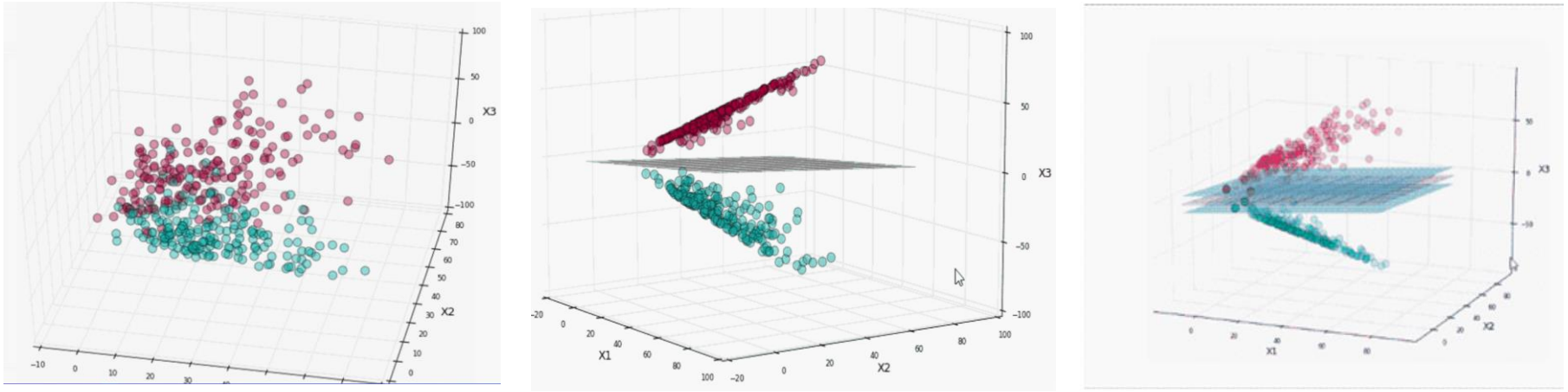


Image Source : https://dzone.com/articles/support-vector-machines-tutorial

1. Using kernel tricks the data points are project to higher dimensional space
2. The data points become relatively more easily separable in higher dimension space
3. SVM can now be drawn between the data sets with a given complexity

SVM Kernel tricks for higher dimensional analysis

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\text{subject to} \quad \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m$$

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

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \text{ for any } i = 1, \ldots, m$$

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

Machine Learning (Support Vector Machines - Kernels)

1. Kernels is a way of computing dot product of two vectors (how much they influence each other and net result) in some higher dimensional space.

2. That is why Kernels are also known as generalized dot products
3. Cover's theorem shows that the chances of linearly non-separable data sets becoming linearly separable increase in higher dimensions
4. Apply the transformations and use Kernel functions to find the dot products to solve the SVM constrained optimization

Support Vector Machines Kernel Functions

1. SVM libraries come packaged with some standard kernel functions such as polynomial, radial basis function (RBF), and Sigmoid

2. For degree-*d* polynomials, the polynomial kernel looks like $K(x, y) = (x^\mathsf{T} y + c)^d$ where x and y are input vectors in lower dimension space, c is a user specified constant (usually 1). K denotes inner product of x,y in higher dimension space

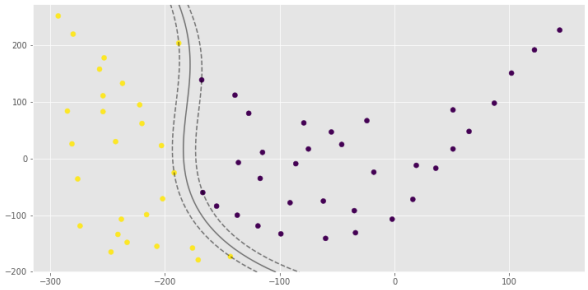


Image Source :https://mubaris.com/posts/svm/

Support Vector Machines Kernel Functions

1. RBF (Radial Basis Function) kernel on two samples x and x' is represented as -

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

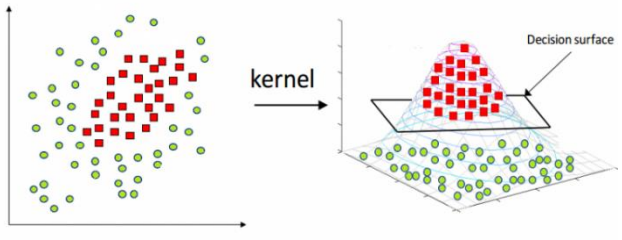


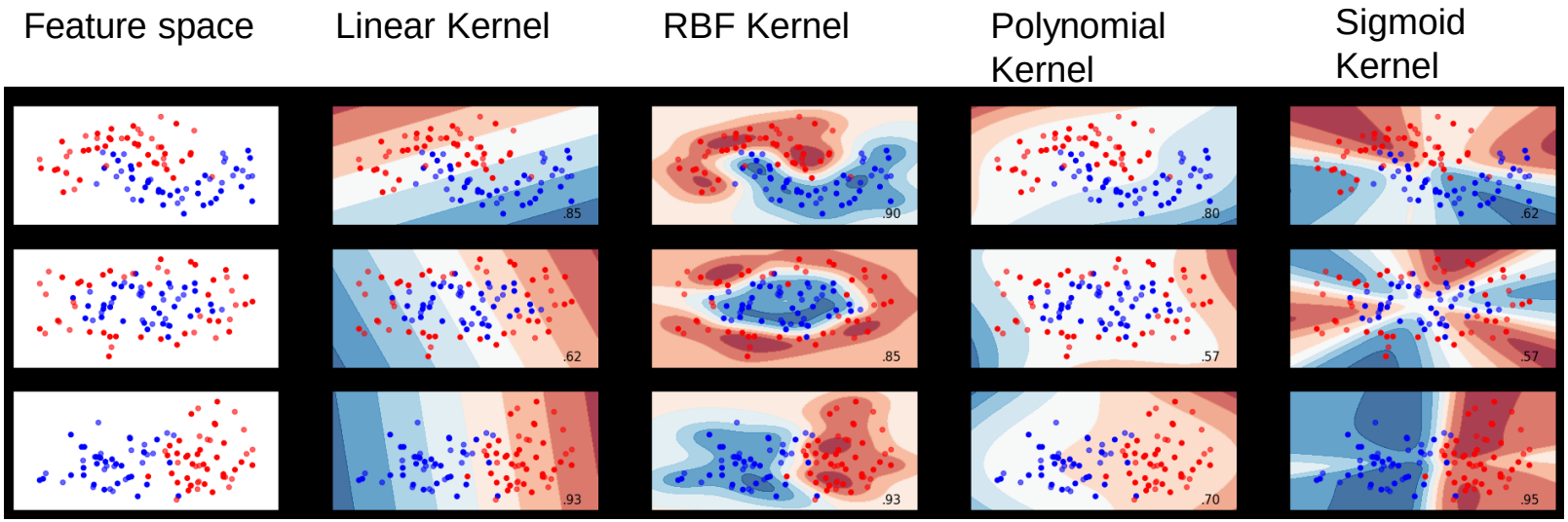
Image Source :https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/

2. It ranges from 0 when distance between x and x' increases (e^-infinity becomes 0) and becomes 1 when x = x' because x – x' = 0 and anything raised to 0 is 1

Image Source :https://mubaris.com/posts/svm/

Support Vector Machines Kernel Functions

5. Sigmoid Kernel looks like **K(X,Y)=tanh(γ·X(transpose)Y+r)**
6. Linear Kernel are of the form that represents linear equation



Source: https://gist.github.com/WittmannF/60680723ed8dd0cb993051a7448f7805

Machine Learning (Support Vector Machines)

Strengths	Weakness
Very stable as it depends on the support vectors only. Not influenced by any other data point including outliers	Computationally intensive
Can be adapted to classification or numeric prediction problems	Prone to over fitting training data
Capable of modelling relatively more complex patterns than nearly any algorithm	Assumes linear relation between dependent and independent variables
Makes no assumptions about underlying data sets	Generally treated as a blackbox model

greatlearning
Learning for Life

Ensemble Learning – **OCR Support Vector Machine**

Lab- 11 Handwritten character recognition

Description – Sample data is available at local file system as Letterdata.csv

The dataset is described at
https://archive.ics.uci.edu/ml/datasets/letter+recognition

Sol: OCR-SVM.ipynb