BITS Pilani
Pilani Campus

Machine Learning

AIML CLZG565

Instance-based Learning & Support Vector Machine

Raja vadhana P
Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement

- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: Slides of Prof. Chetana, Prof.Seetha, Prof.Sugata, Prof.Vimal, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes, Tom Mitchell, Andrew Ng and many others who made their course materials freely available online

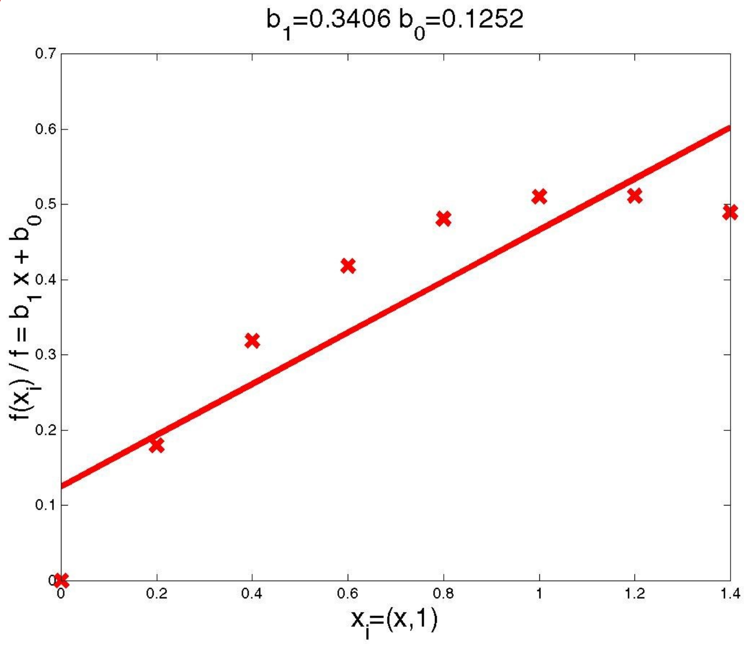
Course Plan

M1	Introduction & Mathematical Preliminaries
M2	Machine Learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

innovate achieve lead

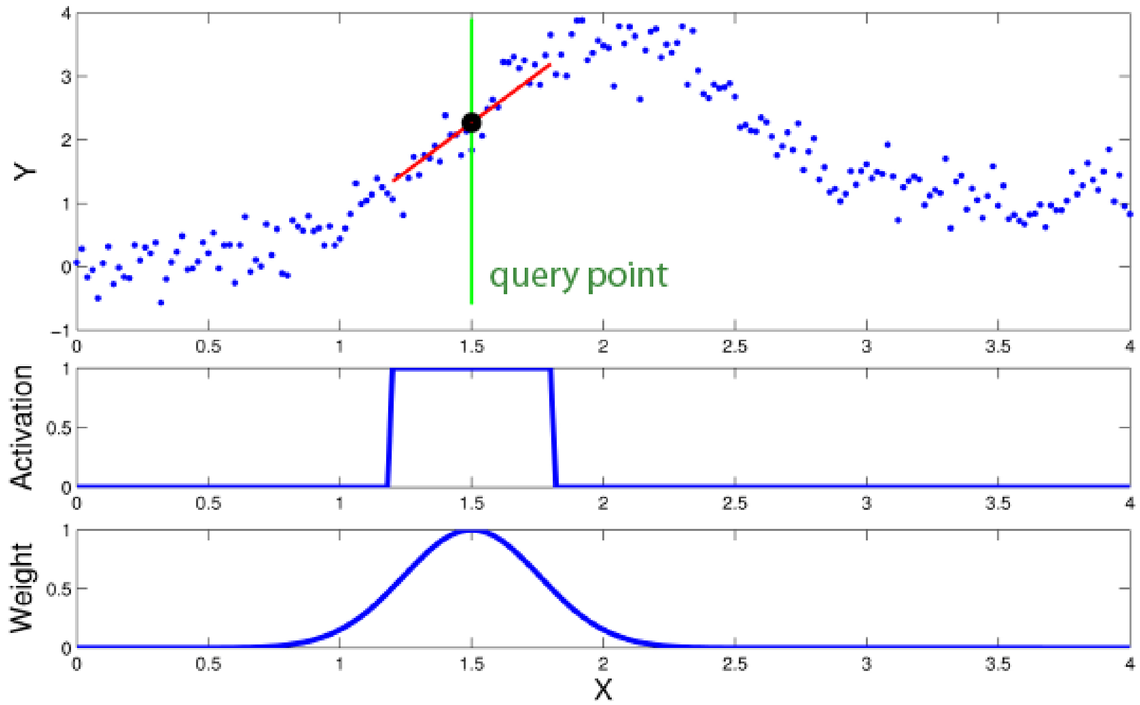
Instance Based Learning

Locally Weighted Regression



Locally Weighted Regression

- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function
- ***Residual*** is the error in approximating the target function. $\hat{f}(x) - f(x)$
- ***Kernel function*** is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function ***K*** such that $w_i = K(d(x_i, x_q))$



Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

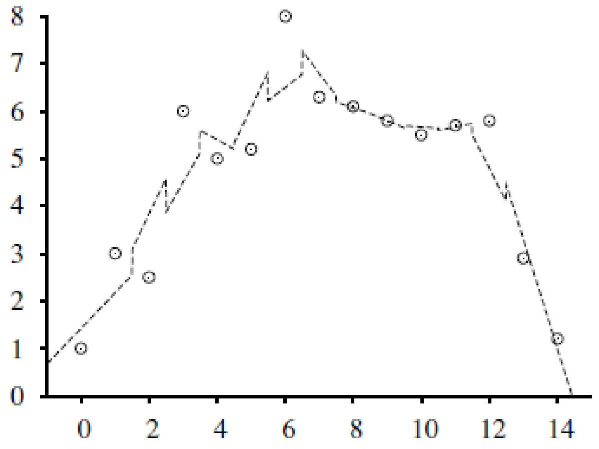
Locally Weighted Regression

- The nearest-neighbor approaches can be thought of as approximating the target function f ***(x)*** at the single query point $\boldsymbol{x} = \boldsymbol{x_q}$

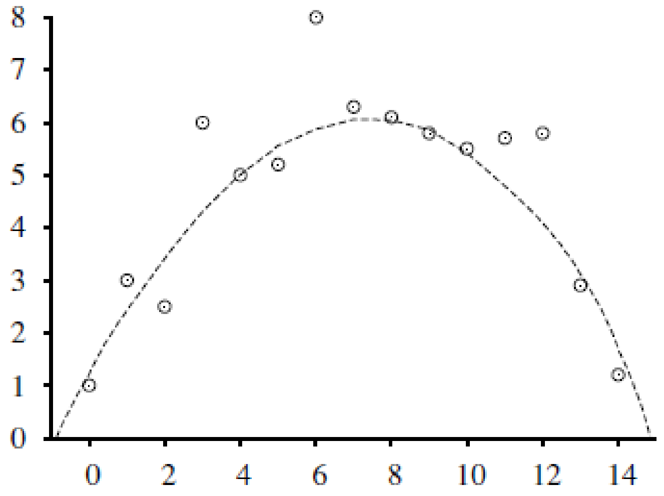
- Locally weighted regression is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding $\boldsymbol{x_q}$

- Approximate the target function in the neighborhood surrounding x, using a linear function, a quadratic function, a multilayer neural network, or some other functional form.

3-nearest-neighbors linear regression



Locally weighted regression with a quadratic kernel



Locally weighted linear regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

- methods like **gradient descent** can be used to calculate the coefficients $w_0, w_1, \ldots, w_n$ to minimize the error in fitting such linear functions

- ANNs require a global approximation to the target function
- here, just a local approximation is needed

$\Rightarrow$ the error function has to be redefined

Locally weighted linear regression

- possibilities to redefine the error criterion E
  1. Minimize the squared error over just the k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neigbors}} (f(x) - \hat{f}(x))^2$$

  2. Minimize the squared error over the entire set D, while weighting the error of each training example by some decreasing function K of its distance from x_q

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

  3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

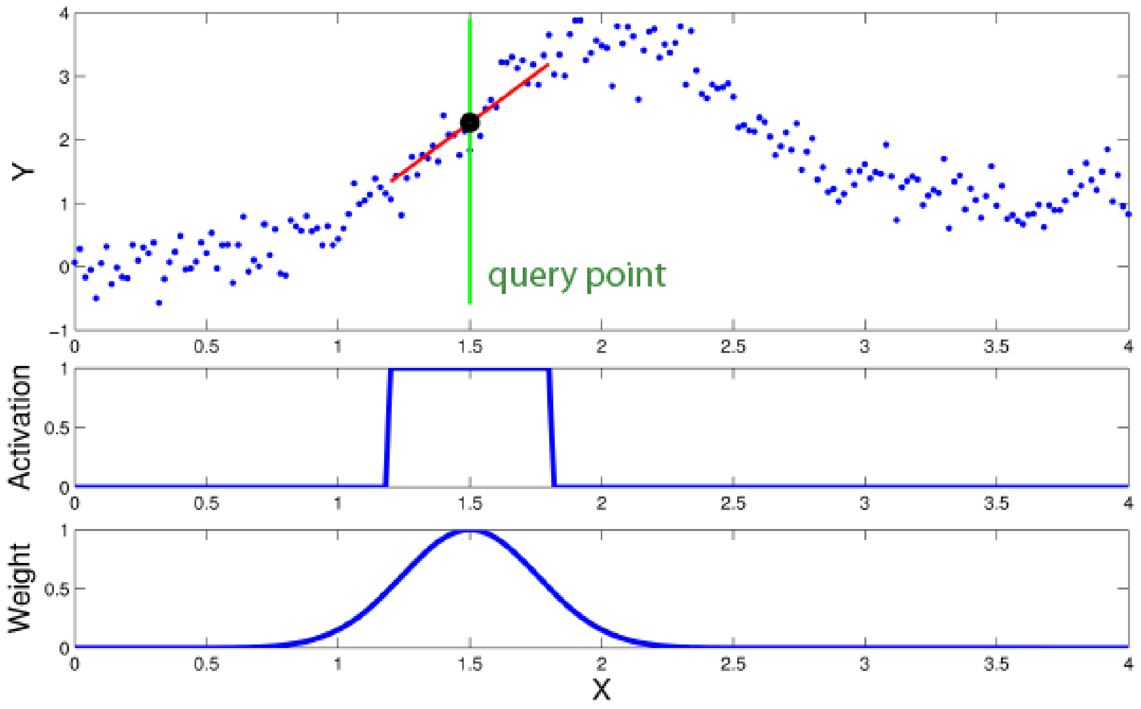
Locally weighted linear regression

- For a given query point $\mathbf{x}_q$ we solve the following weighted regression problem using gradient descent training rule:

$$\Delta w_j = \eta \sum_{x \in \, k \ nearest \ nbrs \ of \ x_q} K(d(x_q, x)) \, (f(x) - \hat{f}(x)) \, a_j(x)$$

- Note that we need to solve a new regression problem for *every* query point—that's what it means to be *local*.
- In ordinary linear regression, we solved the regression problem once, globally, and then used the same h_w for any query point.

Example



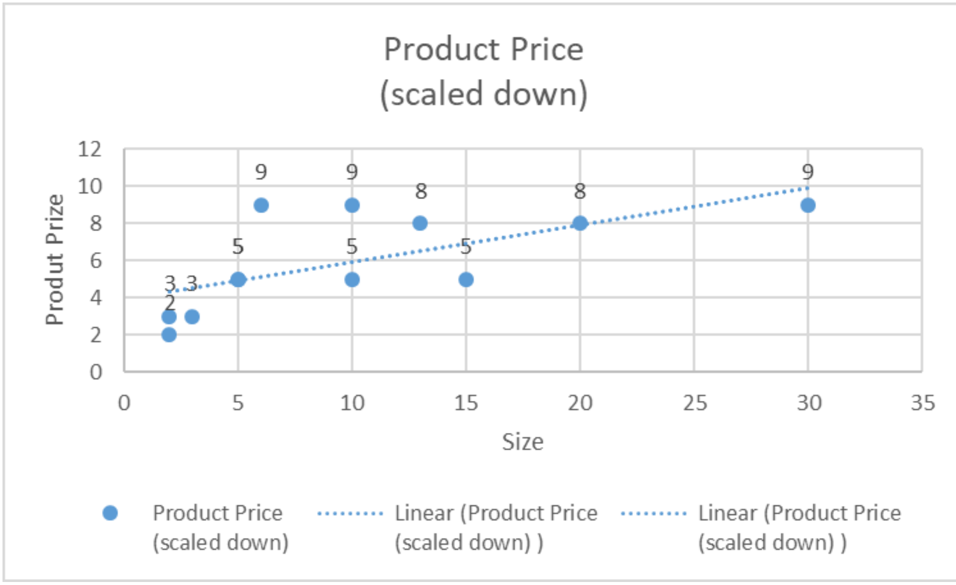
- The upper graphic the set of data points (x,y) (blue dots), query point (green line), local linear model (red line) and prediction (black dot).
- The graphic in the middle shows the activation area of the model.
- The corresponding weighting kernel (receptive field) is shown in the bottom graphic.

Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

Exercise

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(- d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8\,x$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

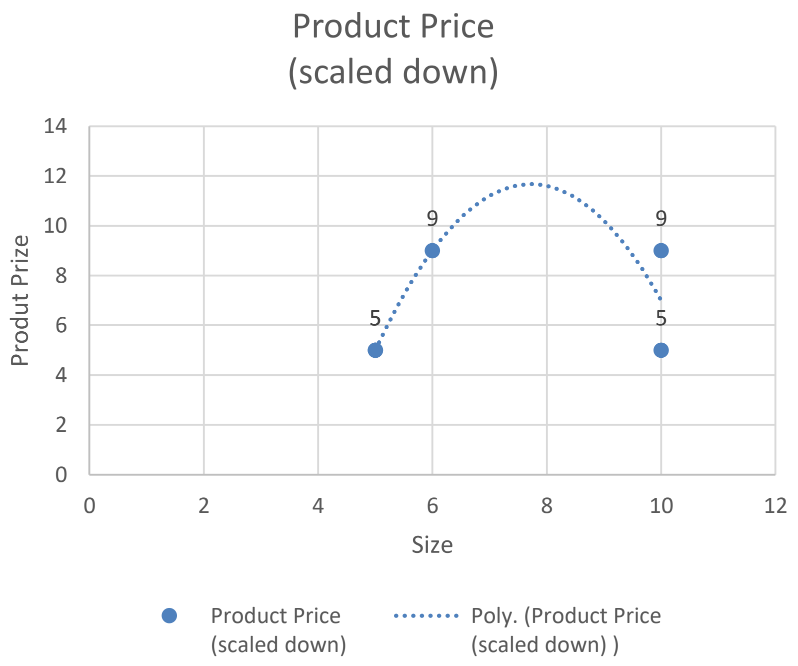
Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8



Exercise

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(-d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8x$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8



Exercise

Use either of below formula :

$$W_{j-new} = W_{j-old} + \alpha * \sum_{i=1}^{K\ Neighbors} K(d(xq,xi))(\ f(x_i) - f\hat{}(x_i)\ )a_j(x_i)$$

$$W_{j-new} = W_{j-old} - \alpha * \sum_{i=1}^{K\ Neighbors} K(d(xq,xi))(f\hat{}(x_i) - f(x_i))a_j(x_i)$$

Size	Product Price (scaled down)	d(xq, xi)	$K(d(x_q,x_i))$
10	5	3	0.01
5	5	2	0.14
2	2	5	0
3	3	4	0
5	5	2	0.14
15	5	8	0
30	9	23	0
20	8	13	0
6	9	1	0.61
2	3	5	0
10	9	3	0.01
13	**8**	6	0

Apply gradient descent where the gradient is weighed by the kernel output.
Answer :
Updated weights after 1st iteration
w0 = 3.43, w1= 0.43

Price Prediction	
6.459515	After 1st iteration of GD
9.1	Before GD

When to Consider Nearest Neighbors

- Suitable for Low dimensional datasets
- Local information can yield highly adaptive behavior
- Lots of training data (distance-weighted KNN)
    - Training is very fast
- Learn complex target functions
    - Do not lose information
- Noisy training data (distance-weighted KNN)
    - by taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples.

Nearest-Neighbor Classifiers: Challenges

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Expensive, Slow at query time
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)

Exercise for Practice – As requested by the students

Size	Rating (1-10)	Product Price (scaled down)
10	5	5
5	2	5
2	7	2
3	6	3
5	4	5
15	8	5
30	4	9
20	8	8
6	10	9
2	2	3
10	4	9
13	6	8

1. Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 12 if 7-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = 1/\ d(x_q,x_i)^2$. Use the linear regression of the form $y = 2 + 1.5\ x$ and the learning rate of 0.6 to apply the gradient descent and update the weights at the end of first iteration. Use Manhattan Distance for the distance calculation.
2. Try to scale all the features with min-max of [0-5] and redo the modelling under previous question. Observe to Interpret the influence of the scaling
3. Find the MSE of the K-NNs from the results of both above parts separately & state your observations
4. If the Product with features (13,6) and (30,4) are outliers then interpret the effect of the same in the K-NN modelling, before and after removing them with numerical justication.

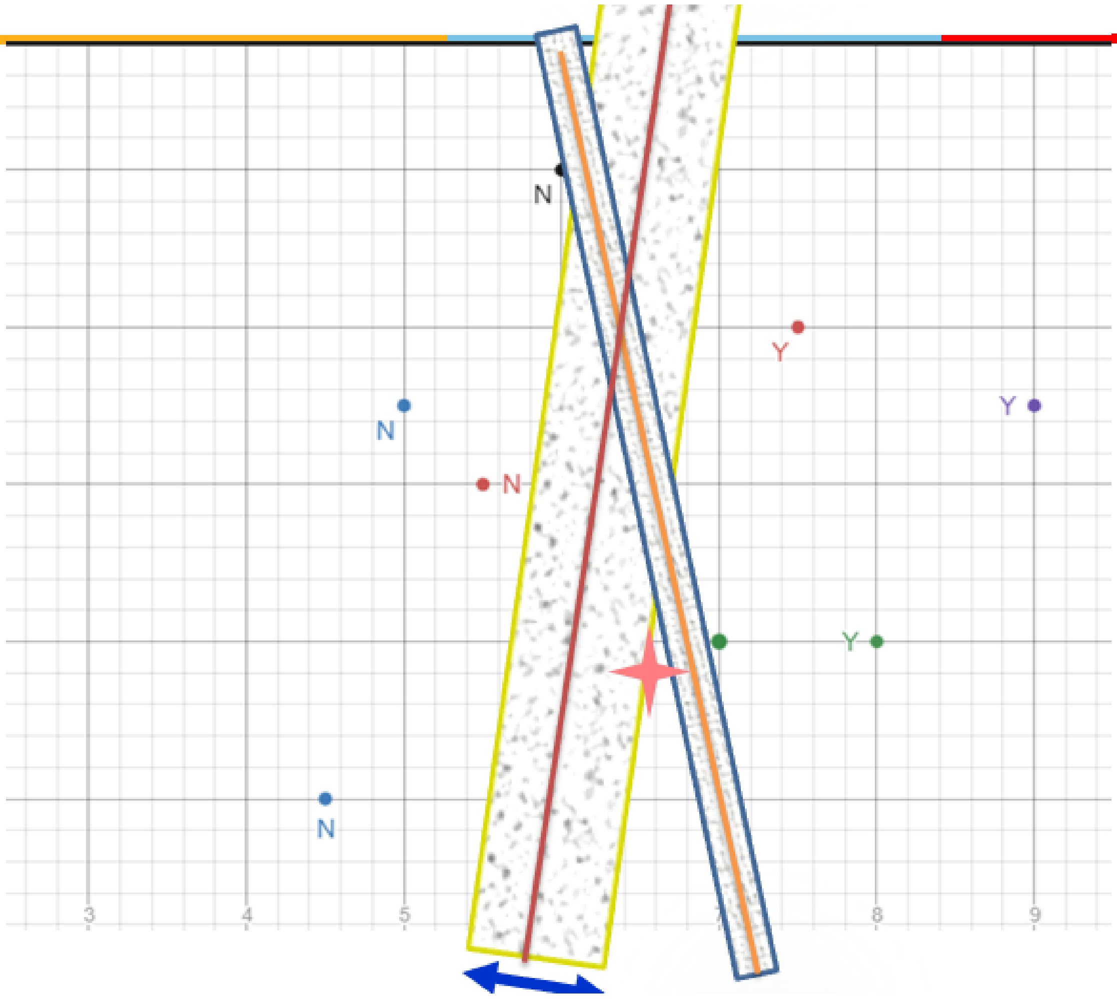
Lab : Practise Problems

- **Tune KNN**. Try different values of *k* to see if you can improve the performance of the algorithm.
- **Use weighted kNN** and check the performance
- **Regression**. Apply it to a regression predictive modeling problem (e.g. predict a numerical value)
- **More Distance Measures**. Implement other distance measures such as Hamming distance, Manhattan distance and Minkowski distance.
- **Data Preparation**. Distance measures are strongly affected by the scale of the input data. Experiment with standardization and other data preparation methods in order to improve results.
- **Dealing with categorical attributes**

innovate achieve lead

Notion of SVM Classifier

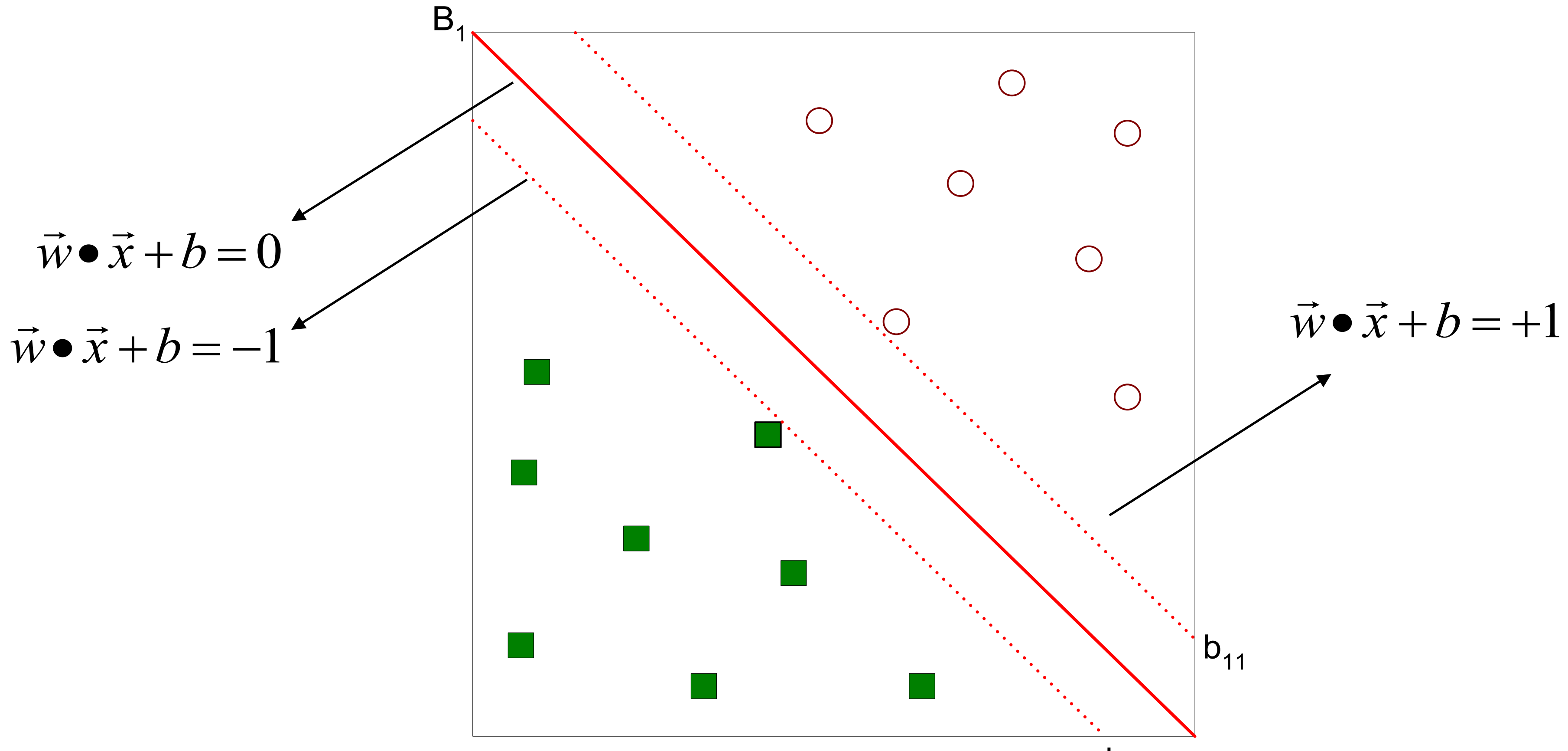
Linear SVM



Maximum margin hyperplane in SVM

CGPA	IQ	Job Offered
5.5	100	No
5	105	No
8	90	Yes
9	105	Yes
6	120	No
7.5	110	Yes
4.5	80	No
7	90	Yes
8.5	95	????
6	130	????

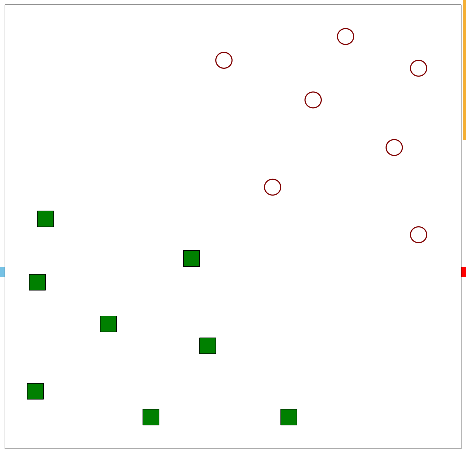
Support Vector Machine

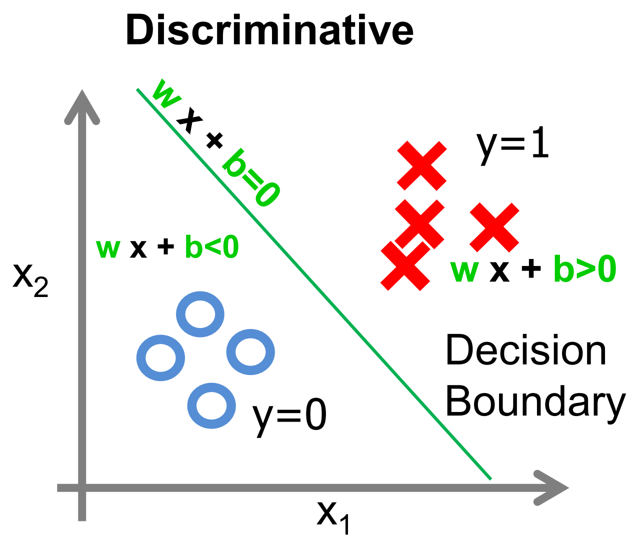

B_1
$\vec{w} \bullet \vec{x} + b = 0$
$\vec{w} \bullet \vec{x} + b = -1$
$\vec{w} \bullet \vec{x} + b = +1$
b_{11}
b_{12}

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

Support Vector Machine





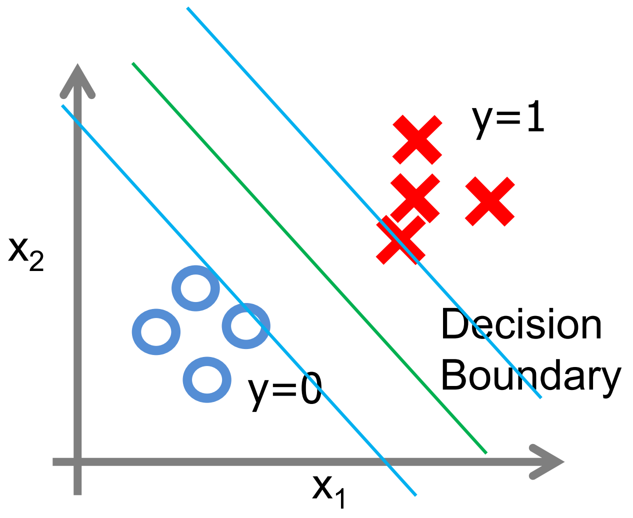
Requirement

- $W^T X \gg 0$ for $y = 1$
- $W^T X \ll 0$ for $y = -1$

$$\theta_0 + \sum_i \theta_i x_i \geq 0$$

$$\theta_0 + \sum_i \theta_i x_i < 0$$

Support Vector Machine



$$\theta_0 + \sum_i \theta_i x_i \geq +1$$

$$\theta_0 + \sum_i \theta_i x_i \leq -1$$

•SVM technique has its roots in statistical learning theory (Vlamidir Vapnik, 1992).

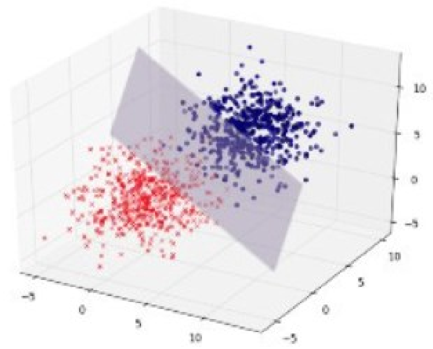
•As a task of classification, it searches for optimal hyperplane(i.e., decision boundary) separating the tuples of one class from another.

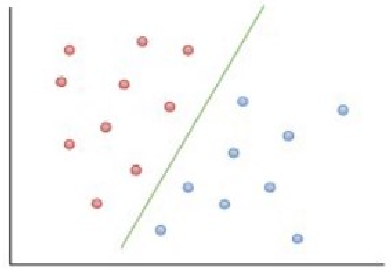
Decision Boundary & Margin

$$\mathbf{w}^T\mathbf{x} = 0 \qquad y = ax + b$$

Hyperplane

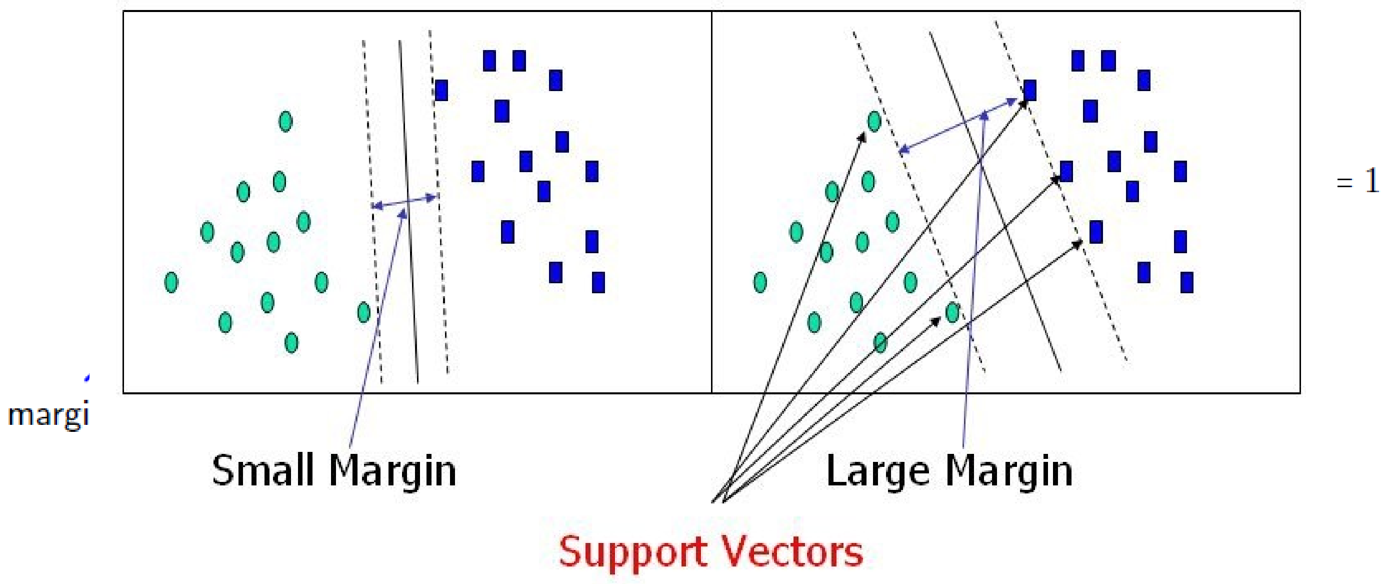
Line





- The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points
- The location of this boundary is determined by a subset of the data points, known as **support vectors**
- Training or learning phase searches for the hyperplane with maximum margin. Such a hyperplane is called maximum margin hyperplane
- Support vectors if removed, would alter the position of the dividing hyperplane

Decision Boundary & Margin



- The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points
- The location of this boundary is determined by a subset of the data points, known as **support vectors**
- Training or learning phase searches for the hyperplane with maximum margin. Such a hyperplane is called maximum margin hyperplane
- Support vectors if removed, would alter the position of the dividing hyperplane

Support Vector Machine

$w \cdot x^+ + b = +1$
$w \cdot x^- + b = -1$

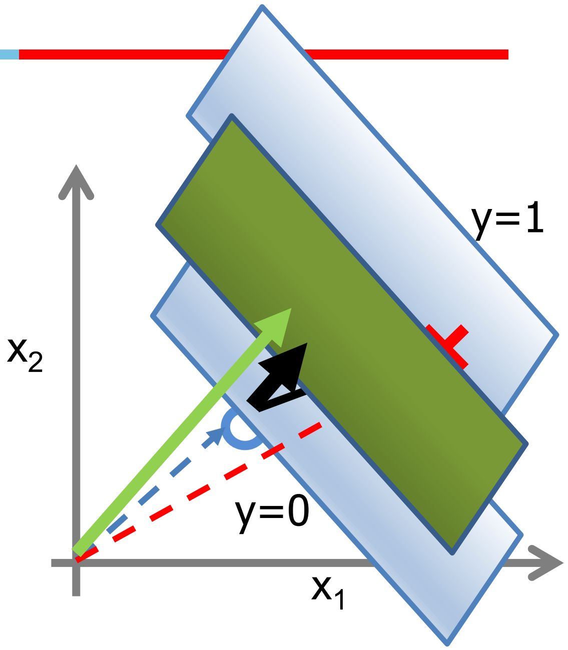
Margin width

$$= (x^+ - x^-) \cdot \frac{w}{||w||}$$

$$= \frac{w \cdot x^+ - w \cdot x^-}{||w||}$$

$$= (1-b) - (-1-b) / ||w||$$

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



There are no datapoints between H_1 and H_2

Support Vector Machine

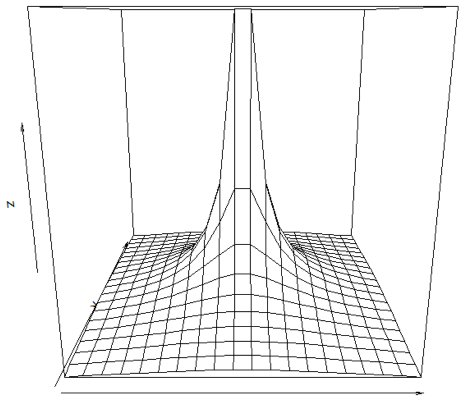
1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data points:

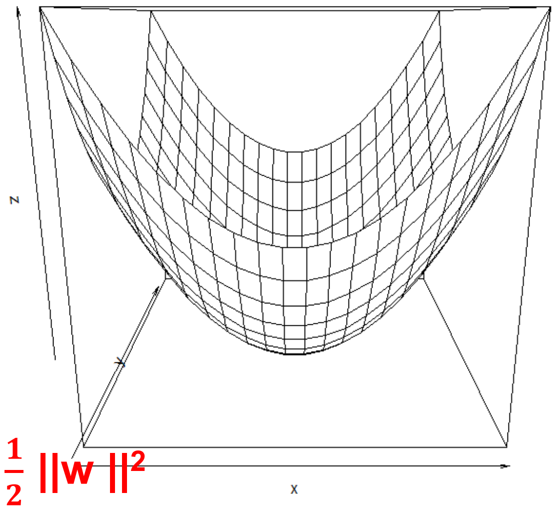
$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

The solution involves constructing a unconstrained problem where a Lagrange multiplier is associated with every constraint in the primary problem: Tip : Use KKT condition!

Quadratic optimization problem:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1$





Maximize margin: r = $\frac{2}{||w||}$ → **Equivalent to minimize: $\frac{1}{2}$ ||w ||^2**

[*Math Basics Solving constrained Optimization Problem by Lagrange Multipliers*]

Support Vector Machine

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

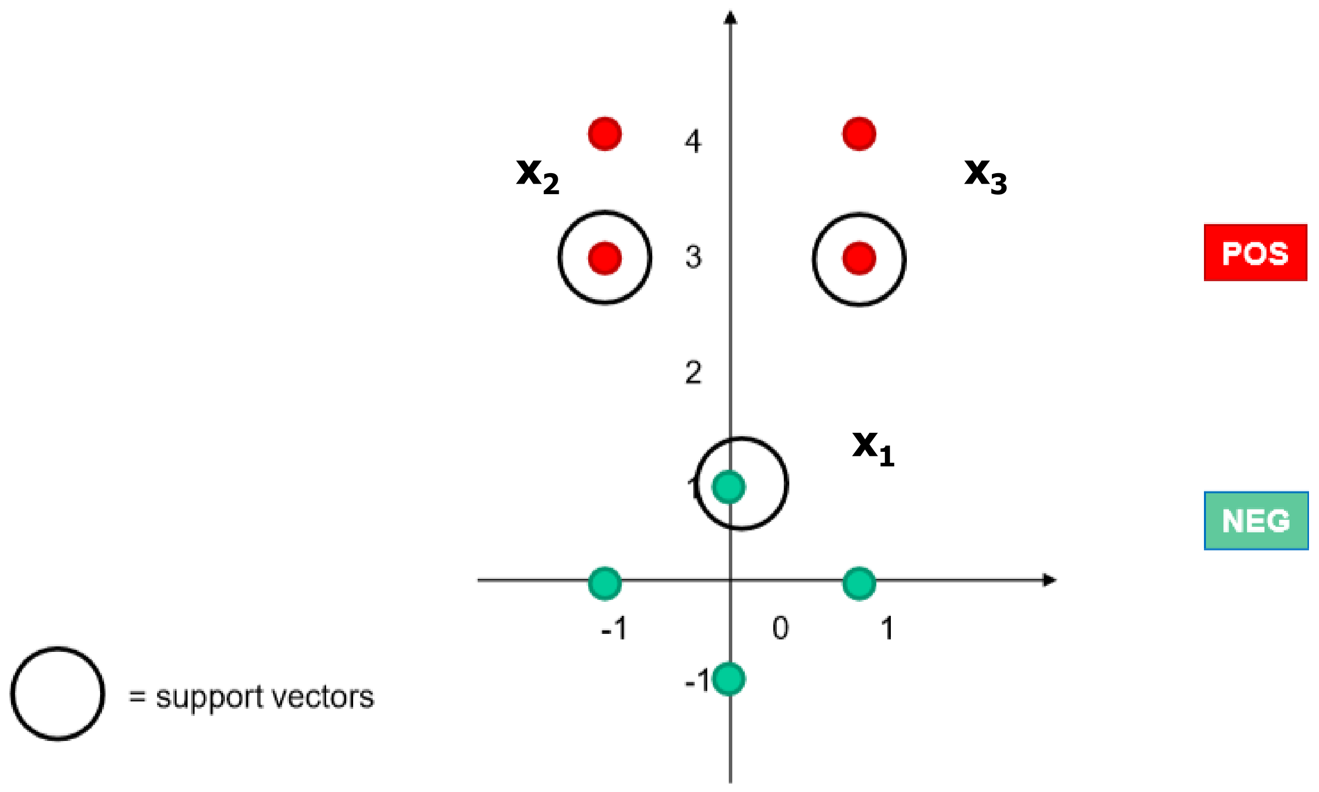
- Classification function:

$$f(x) = \text{sign}\,(\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$
$$= \text{sign}\left(\sum_i \alpha_i y_i \boxed{\mathbf{x}_i \cdot \mathbf{x}} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

Example



Example adapted from Dan Ventura

Solving for α : Problem Type – 1 : Linear SVM

- We know that for the support vectors, f(x) = 1 or -1 exactly
- Add a 1 in the feature representation for the bias

- The support vectors have coordinates and labels:
  - **x_1 = [0 1 1], y_1 = -1**
  - **x_2 = [-1 3 1], y_2 = +1**
  - **x_3 = [1 3 1], y_3 = +1**

- Thus we can form the following system of linear equations:

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$\boxed{f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b}$$

I Select the support vectors:

$x_1 = [0\ 1\ 1]$, $y_1 = -1$

$x_2 = [-1\ 3\ 1]$, $y_2 = +1$

$x_3 = [1\ 3\ 1]$, $y_3 = +1$

$\alpha_i [-1\ (\mathbf{w}\ .\ \mathbf{x_i} + b)] = -1$

$\alpha_i [+1\ (\mathbf{w}\ .\ \mathbf{x_i} + b)] = 1$

II Substitute in Lagrangian function: $L(w, b, \alpha_i) = \Sigma\ \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j\ xi\ .\ xj)$

$L(w, b, \alpha_i) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 y_1 y_1 x_1 . x_1 + \alpha_2 \alpha_2 y_2 y_2 x_2 . x_2 + \alpha_3 \alpha_3 y_3 y_3 x_3 . x_3 + 2 \alpha_1 \alpha_2 y_1 y_2 x_1 . x_2 + 2 \alpha_1 \alpha_3 y_1 y_3 x_1 . x_3 + 2 \alpha_2 \alpha_3 y_2 y_3 x_2 . x_3)$

$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 x_1 . x_1 + \alpha_2 \alpha_2 x_2 . x_2 + \alpha_3 \alpha_3\ x_3 . x_3 - 2 \alpha_1 \alpha_2\ x_1 . x_2 - 2 \alpha_1 \alpha_3\ x_1 . x_3 + 2 \alpha_2 \alpha_3 x_2 . x_3)$

$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 [0\ 1\ 1] . [0\ 1\ 1] + \alpha_2 \alpha_2 [-1\ 3\ 1] . [-1\ 3\ 1] + \alpha_3 \alpha_3\ [1\ 3\ 1].[1\ 3\ 1] - 2\ \alpha_1 \alpha_2 [0\ 1\ 1].[-1\ 3\ 1] - 2\ \alpha_1 \alpha_3 [0\ 1\ 1].[1\ 3\ 1] + 2\ \alpha_2 \alpha_3 [-1\ 3\ 1].[1\ 3\ 1])$

III Find the Unconstrained Optimization Function:

$L(w, b, \alpha_i) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (2\alpha_1 \alpha_1 + 11\ \alpha_2 \alpha_2 + 11\ \alpha_3 \alpha_3 - 8\ \alpha_1 \alpha_2 - 8\ \alpha_1 \alpha_3 + 18\ \alpha_2 \alpha_3)$

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$\boxed{f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b}$$

IV Gradient of the Lagrangian:

$\alpha_i\,[-1\ (\mathbf{w}\ .\ \mathbf{x}_i + b)] = -1$

$\alpha_i\,[+1\ (\mathbf{w}\ .\ \mathbf{x}_i + b)] = 1$

$L(w, b, \alpha_i) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(2\alpha_1\alpha_1 + 11\,\alpha_2\alpha_2 + 11\,\alpha_3\alpha_3 - 8\,\alpha_1\alpha_2 - 8\,\alpha_1\alpha_3 + 18\,\alpha_2\alpha_3)$

$\frac{\partial L}{\partial \alpha_1} = 0 \rightarrow -2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$

$\frac{\partial L}{\partial \alpha_2} = 0 \rightarrow -4\alpha_1 + 11\alpha_2 + 9\alpha_3 = +1$

$\frac{\partial L}{\partial \alpha_3} = 0 \rightarrow -4\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1$

V Solve the simultaneous linear equation and find the lagrange multiplier:

$(\alpha_1, \alpha_2, \alpha_3) = (3.5, 0.75, 0.75)$

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$
$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

VI Substitute the Lagrange multiplier and obtain the weight's:

$\alpha_i [-1\ (\mathbf{w} \,.\, \mathbf{x}_i + b)] = -1$
$\alpha_i [+1\ (\mathbf{w} \,.\, \mathbf{x}_i + b)] = 1$

$W = -\alpha_1\ [0\ 1\ 1] + \alpha_2\ [-1\ 3\ 1] + \alpha_3 [1\ 3\ 1]$
$= -3.5\ [0\ 1\ 1] + 0.75\ [-1\ 3\ 1] + 0.75\ [1\ 3\ 1]$
$= [0\ 1\ -2]$

VII Find the bias with help of any one of the support vectors:

Note the Bias is found above as a part of weight vector!!

VIII Construct the equation of the LSVM hyperplane:

$W X + b = 0$
$Y - 2 = 0$
$Y = 2$

IX Optionally find the width of the margin: $\frac{2}{||W||}$ H.W

Solving for α

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

- System of linear equations:

α1 y1 dot(x1, x1) + α2 y2 dot(x1, x2) + α3 y3 dot(x1, x3) = y1

α1 y1 dot(x2, x1) + α2 y2 dot(x2, x2) + α3 y3 dot(x2, x3) = y2

α1 y1 dot(x3, x1) + α2 y2 dot(x3, x2) + α3 y3 dot(x3, x3) = y3

-2 * α1 + 4 * α2 + 4 * α3 = -1

-4 * α1 + 11 * α2 + 9 * α3 = +1

-4 * α1 + 9 * α2 + 11 * α3 = +1

$$\alpha_i[-1\,(\mathbf{w}\,.\,\mathbf{x}_i + b)] = -1$$
$$\alpha_i[+1\,(\mathbf{w}\,.\,\mathbf{x}_i + b)] = 1$$

- Solution: α1 = 3.5, α2 = 0.75, α3 = 0.75

Solving for w, b; plotting boundary

- We know **$w = \Sigma\, \alpha_i\, y_i\, x_i$** **i.e** **$w = \alpha_1\, y_1\, x_1 + \ldots + \alpha_N\, y_N\, x_N$**

  where N = No of SVs

- Thus w = -3.5 * [0 1 1] + 0.75 [-1 3 1] + 0.75 [1 3 1] = [0 1 -2]
- Separating out weights and bias, we have: **w = [0 1] and b = -2**

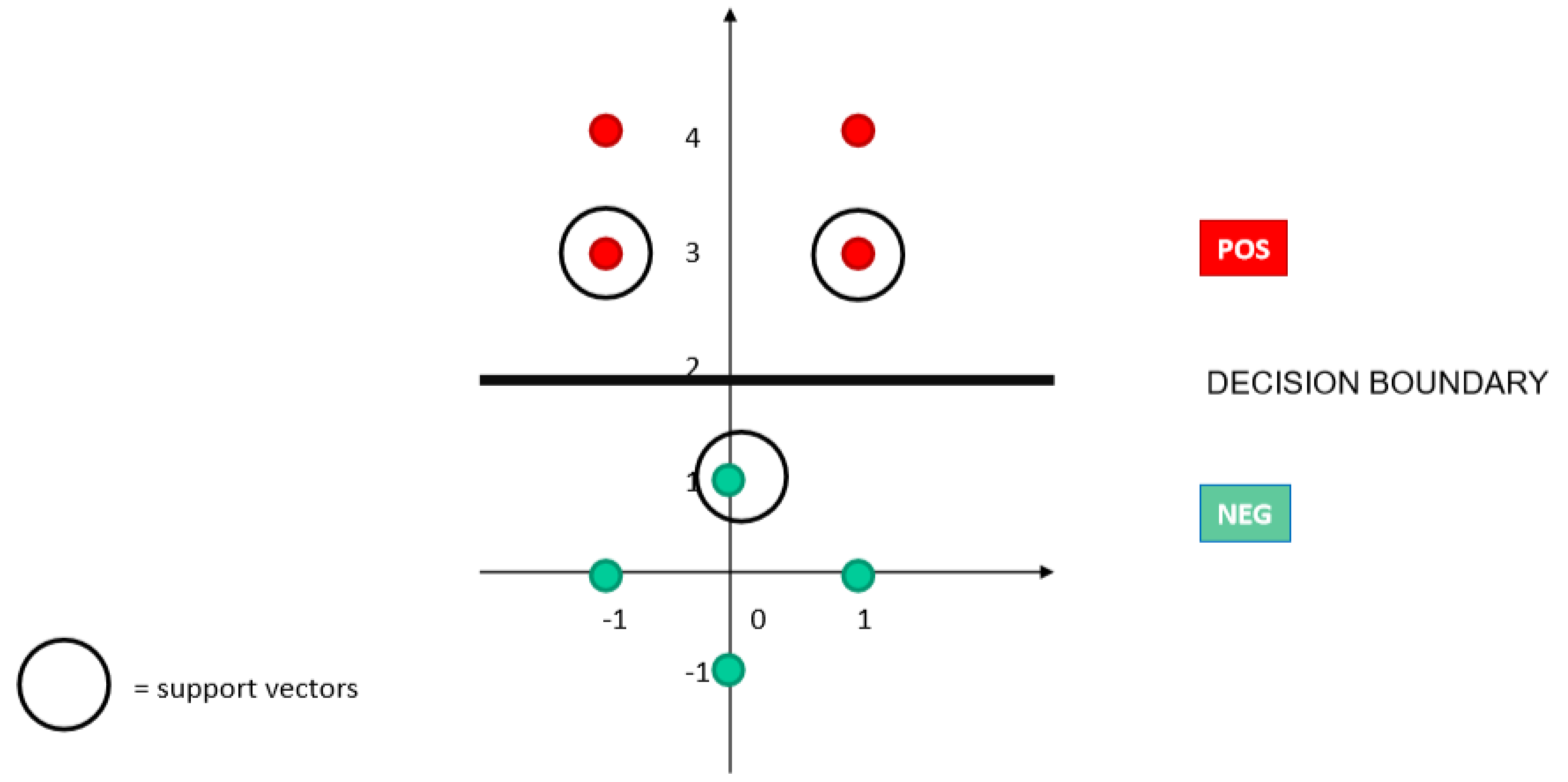
  a=0, c=1

<u>Boundary:</u>

- For SVMs, we used this eq for a line: ax + cy + b = 0 where w = [a c]
- Thus ax + b = -cy ➔ y = (-a/c) x + (-b/c)
- Thus y-intercept is (-b/c) = -(-2)/1 = 2
- The decision boundary is perpendicular to w and it has slope

  =(-a/c) = -0/1 = 0

Decision boundary



Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$
$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

X Predict the class for unknown data:

$$\boxed{f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b}$$

α_i [-1 ($\mathbf{w}$. $\mathbf{x}_i$ + b)] = -1
α_i [+1 ($\mathbf{w}$. $\mathbf{x}_i$ + b)] = 1

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

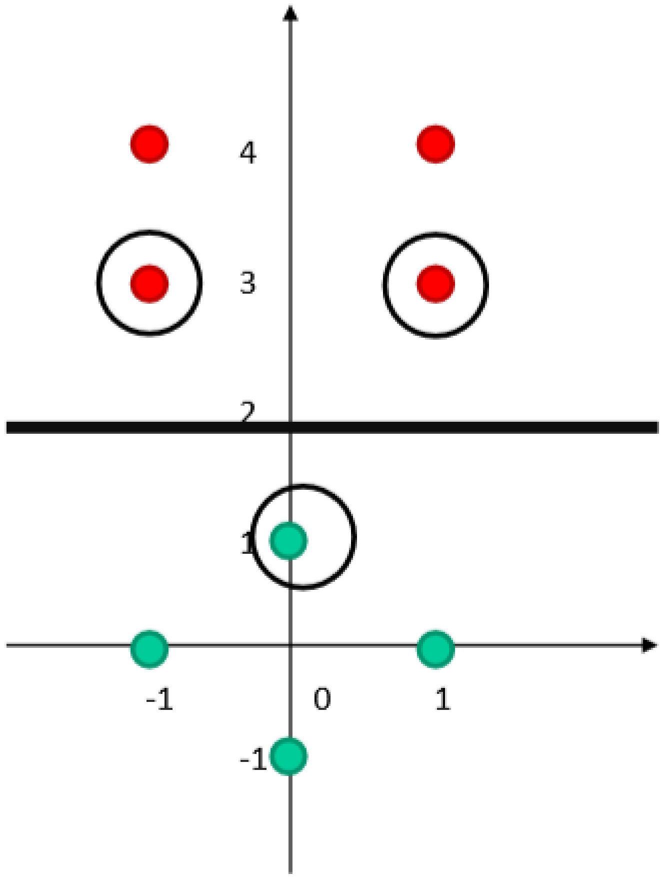
  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)
- Classification function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + \text{b})$$
$$= \text{sign}\left(\sum_i \alpha_i y_i \boxed{\mathbf{x}_i \cdot \mathbf{x}} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

Linear SVM



🟥 POS

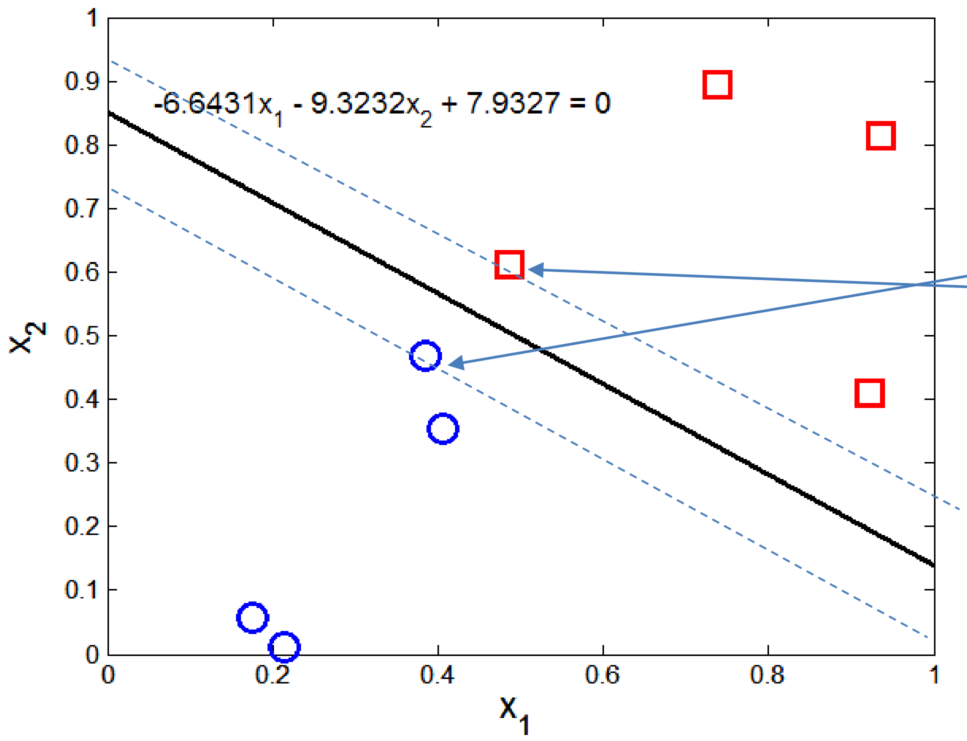
DECISION BOUNDARY

🟩 NEG

○ = support vectors

Support Vectors

$$L(\mathbf{w}, b, \lambda_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma \lambda_i [y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1]$$

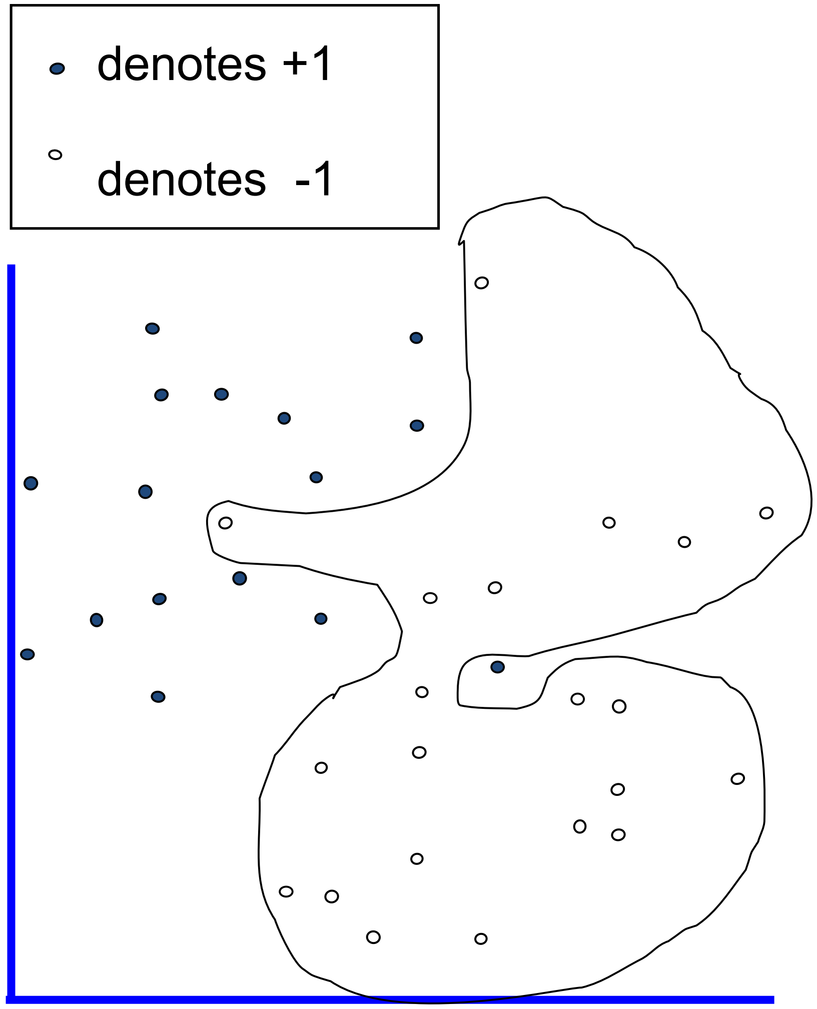


Support vectors

x1	x2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

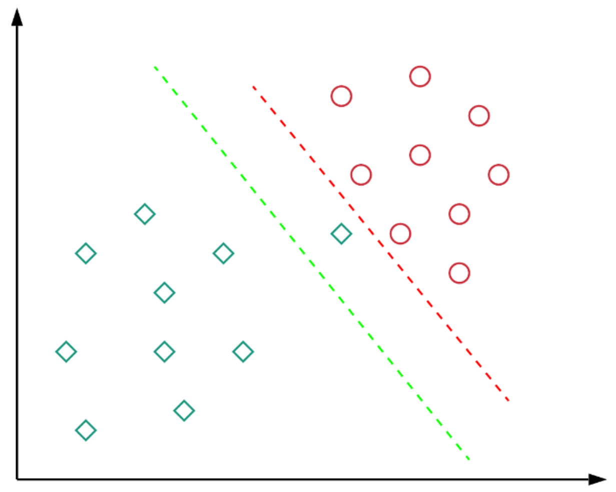
Dataset with noise

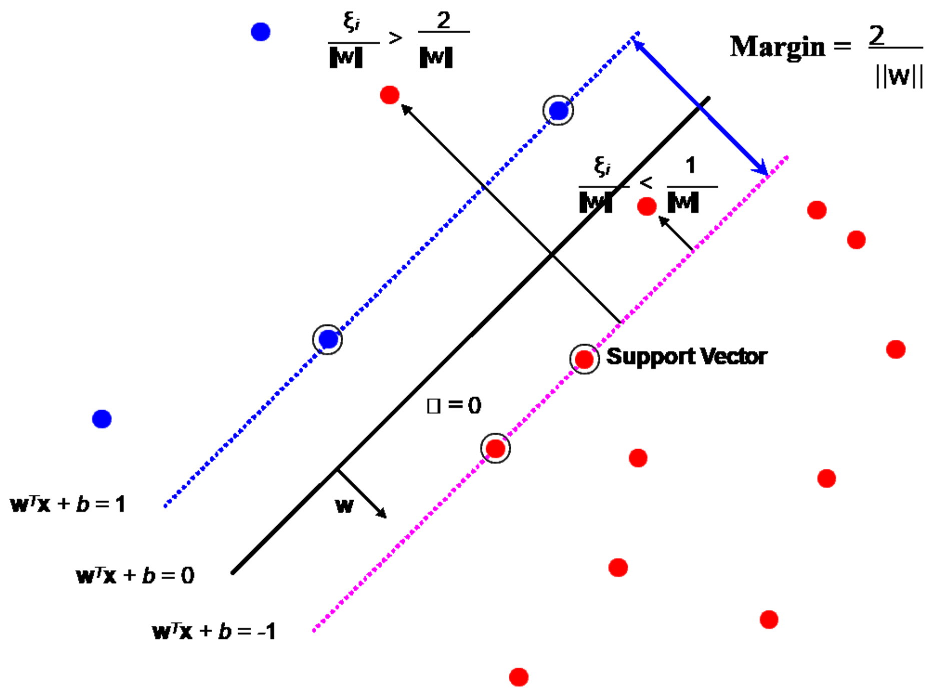


- Hard Margin: So far we require all data points be classified correctly
  - No training error
- What if the training set is noisy?

Motivation : Soft margin

- Almost all real-world applications have data that is linearly inseparable.
- When data *is* linearly separable, choosing perfect decision boundary leads to overfitting





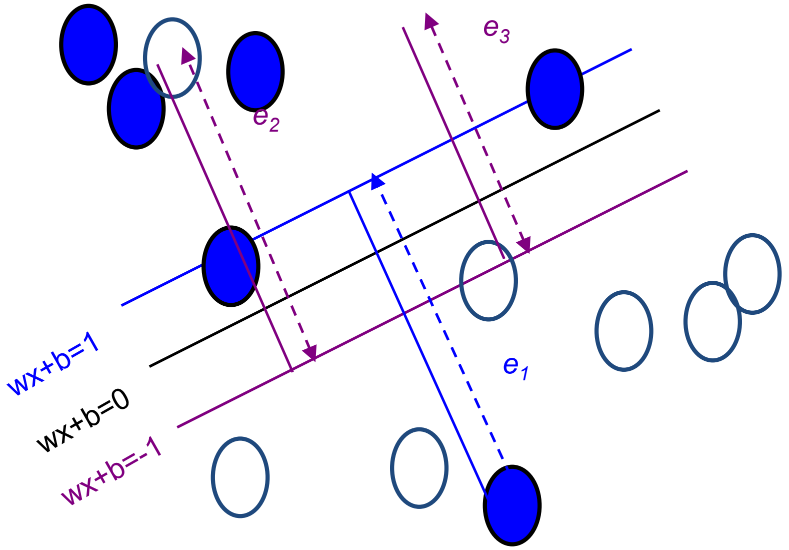
$\frac{\xi_i}{\|w\|} > \frac{2}{\|w\|}$
Margin = $\frac{2}{\|w\|}$
$\frac{\xi_i}{\|w\|} < \frac{1}{\|w\|}$
Support Vector
$\xi = 0$
$w^T x + b = 1$
$w^T x + b = 0$
$w^T x + b = -1$
w

Introduce "slack" variables

- **$\xi_i \geq 0$**

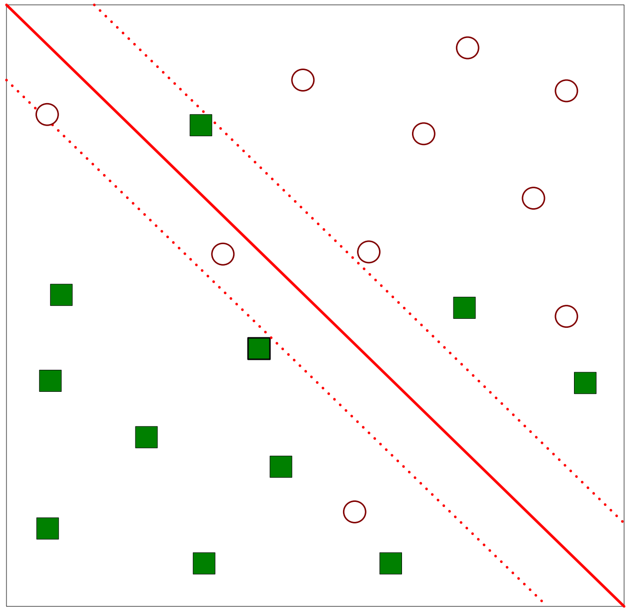
- for $0 < \xi \leq 1$: point is between margin and correct side of hyper-plane. This is a <u>margin violation</u>
- for $\xi > 1$ point is misclassified

Misclassified point



Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples. It can be regarded as a measure of confidence

Soft Margin



$$\underbrace{\min_{w}}_{\text{The } w \text{ that minimizes...}} \underbrace{\frac{1}{2}\|w\|^2}_{\text{Maximize margin}} + \underbrace{\overset{\text{Misclassification cost}}{C} \sum_{i=1}^{\overset{\text{\# data samples}}{N}} \overset{\text{Slack variable}}{\xi_i}}_{\text{Minimize misclassification}}$$

$$\text{subject to} \quad y_i w^T x_i \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

- **Hard Margin:**

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i(\mathbf{w}^T\mathbf{x_i} + \mathrm{b}) \geq 1$

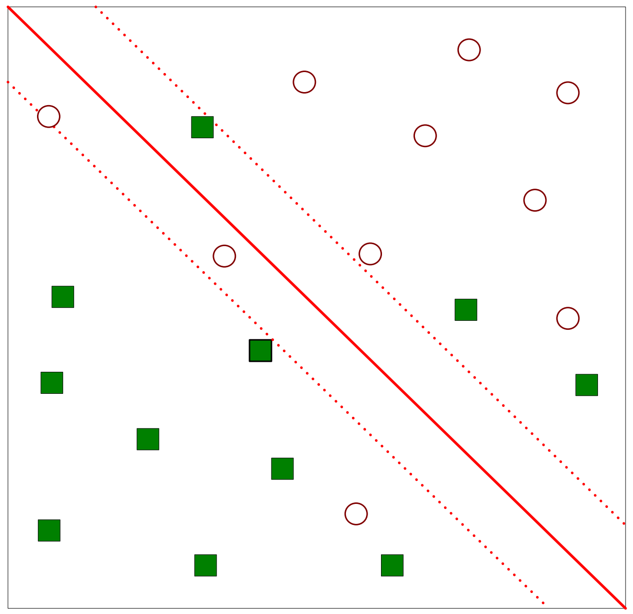
- **Soft Margin incorporating slack variables:**

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

Soft Margin



Noisy Training Set

Solution :

Slack variables ξ

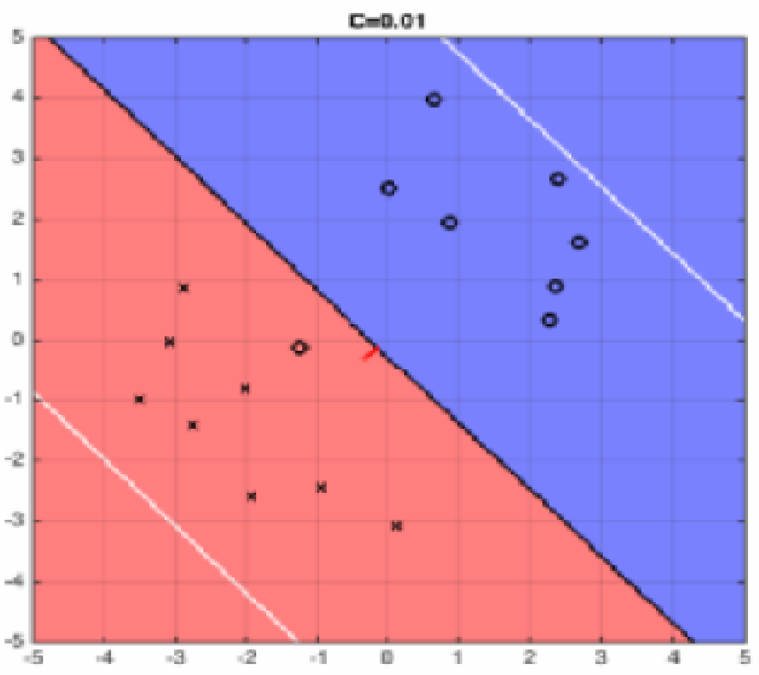
Regularization C

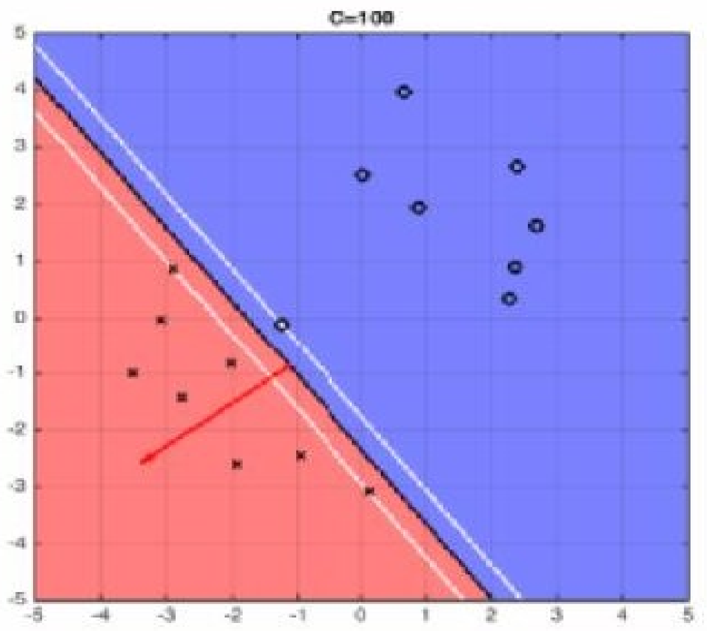
When C is large, larger slacks penalize the objective function of SVM’s more than when C is small

For Large values of C, the optimization will choose a smaller-margin hyperplane

Effect of Margin size v/s misclassification cost

Effect of C





Small value of C will cause the optimizer to look for a larger-margin (small penalties) separating hyperplane, even if that hyperplane misclassifies more points.

For large values of C, the optimization will choose a smaller-margin (large penalties) hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
C=infinity ->hard margin SVM

SVM Problem - Summary

SVM: Optimization

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

$$\text{Subject to:} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

SVM: Training

Input: (X,y), C

Output: alpha for support vectors, b

Hyper parameter : C

SVM: Classification

$$sign(\mathbf{w}^T\mathbf{x} + b)$$
$$= sign(\sum_i \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b)$$

C parameter tells the SVM optimization how much you want to avoid misclassifying each training example and *C* can be viewed as a way to control overfitting

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - Only support vectors are used to specify the separating hyperplane
  - Therefore SVM also called sparse kernel machine.
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution

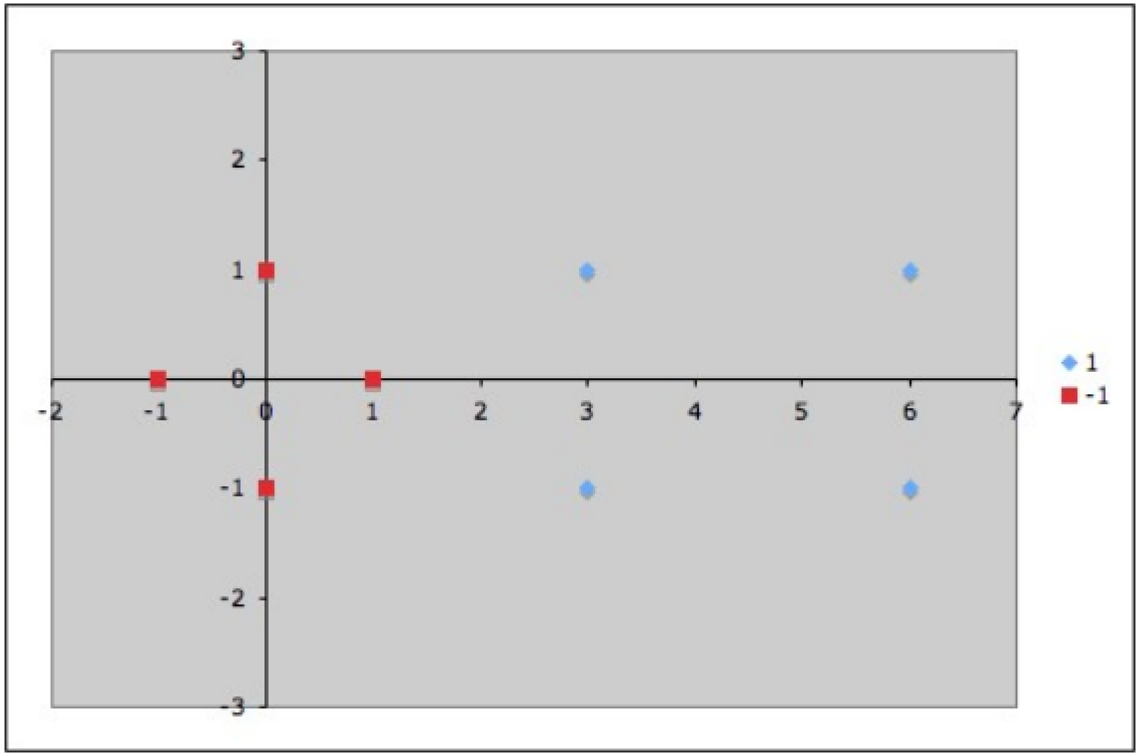
Problem for Practice

1. Assume among the given training instances, Positive & Negative training examples , (0,1), (0,-1) , (3,1), (3,-1) are the support vectors. Construct the Lagrangian and find the equation of the hyperplane P1 that best seperates the classes.

2. Using the results find the training accuracy of the classifier. Use all the training instance for this case to construct the confusion matrix.

3. Instead to the above case , If the support vectors are (1,0), (3,1) and (3,-1) instead, find the equation of the hyperplane P2

4. Find the width of the margin for both P1 & P2 case.
5. Predict the class of the test data (1.75, 3) for the both the P1 & P2 case



References for SVM

- **Text categorization with Support Vector Machines: learning with many relevant features** - T. Joachims, ECML
- **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges
- http://www.cs.utexas.edu/users/mooney/cs391L/
- https://www.coursera.org/learn/machine-learning/home/week/7
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://data-flair.training/blogs/svm-kernel-functions/
- MIT 6.034 Artificial Intelligence, Fall 2010
- https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior
- https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796
- https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be
- https://www.svm-tutorial.com/

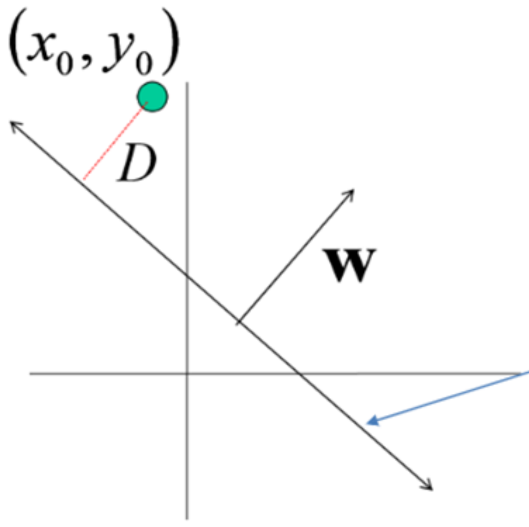
Refresher from Basic Mathematics

(from MFML course)

- Distance between a point and plane
- Vector Distance between two points
- Optimization Problem – Significance Primal Dual
- Non-Linear Optimization Problem

Additional Reference

Math Basics



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

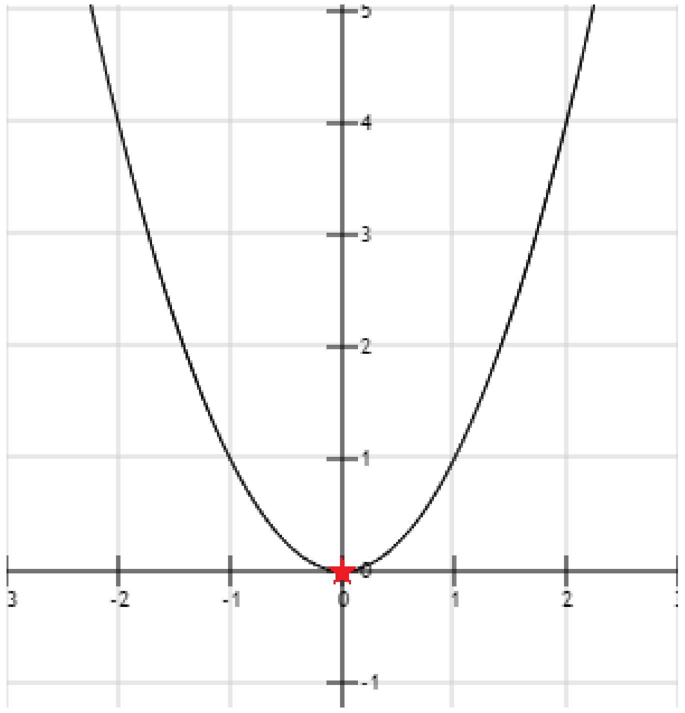
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^{\mathrm{T}}\mathbf{x} + b|}{\|\mathbf{w}\|}$$

distance from point to line

Unconstrained Optimization

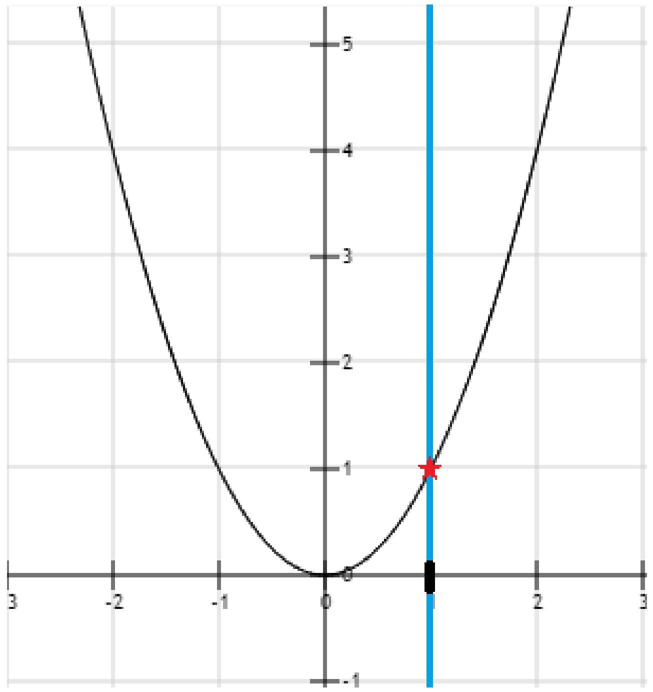
Minimize x^2



Constrained Optimization – Equality Constraint

Minimize x^2

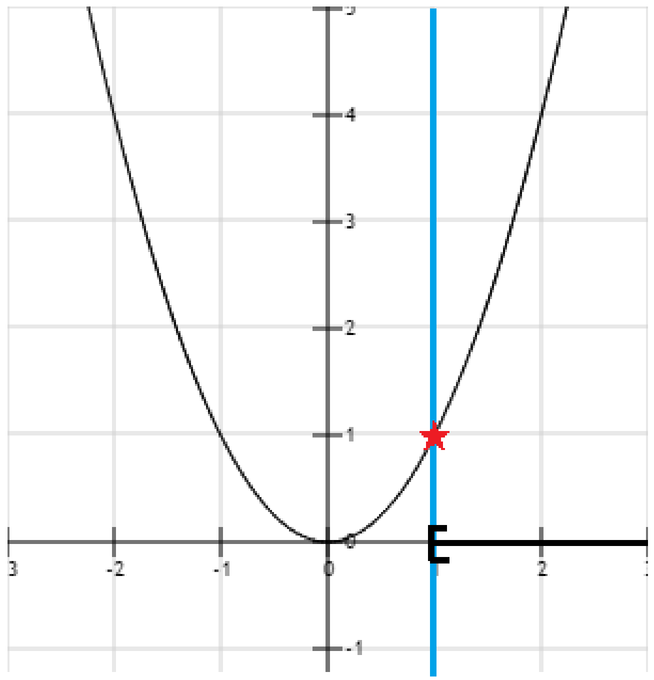
Subject to $x = 1$



Constrained Optimization – Inequality Constraint

Minimize x^2

Subject to $x >= 1$



Optimization Problem

- Optimization problem is typically written:

$$\text{Minimize } f(x)$$

$$\text{subject to}$$

$$g_i(x) = 0, \quad i=1,\dots,p$$

$$h_i(x) <= 0, \quad i=1,\dots,m$$

- f(x) is called the objective function
- By changing x (the optimization variable) we wish to find a value $x*$ for which f(x) is at its minimum.
- p functions of g_i define equality constraints and
- m functions h_i define inequality constraints.
- The value we find MUST respect these constraints!

Constrained Optimization

- We can also have mix equality and inequality constraints together.
- Only restriction is that if we use contradictory constraints, we can end up with a problem which does not have a feasible set

  Minimize x^2

  Subject to

  $x = 1$

  $x < 0$

Impossible for x to be equal 1 and less than zero at the same

Lagrange Multiplier

- **How do we find the solution to an optimization problem with constraints?**
- Constrained maximization (minimization) problem is rewritten as a Lagrange function.
- Lagrange function is a strategy for finding the local maxima and minima of a function subject to equality constraints
- It uses Lagrange multipliers

Lagrange Multiplier - Example

Maximize

$$f(x,y) = x^2 y$$

Subject to

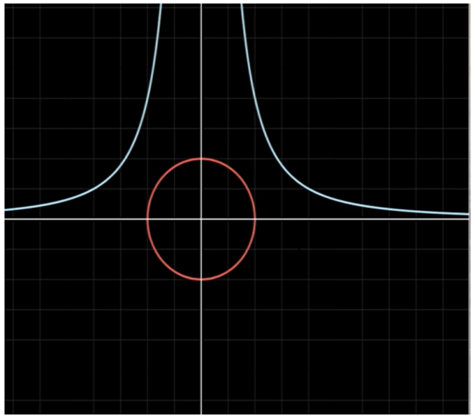
$$g(x, y) : x^2 + y^2 = 1$$

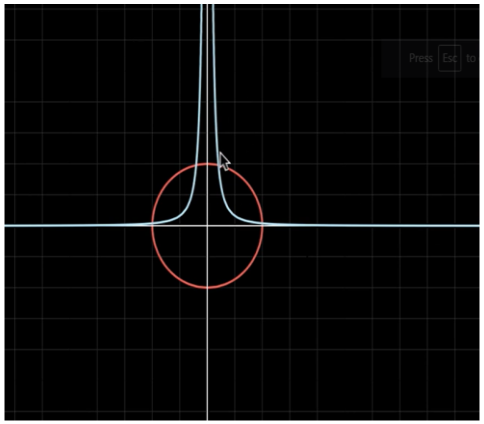
$f(x,y) = K$

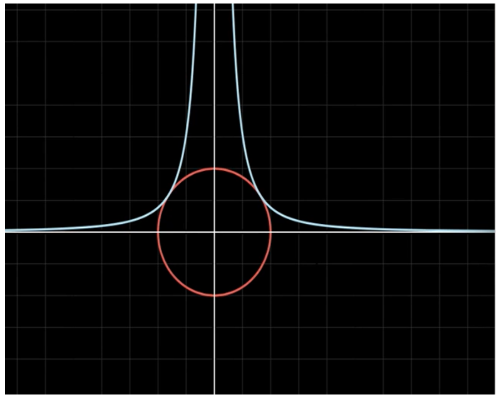
Larger constant K
constant K

Smaller

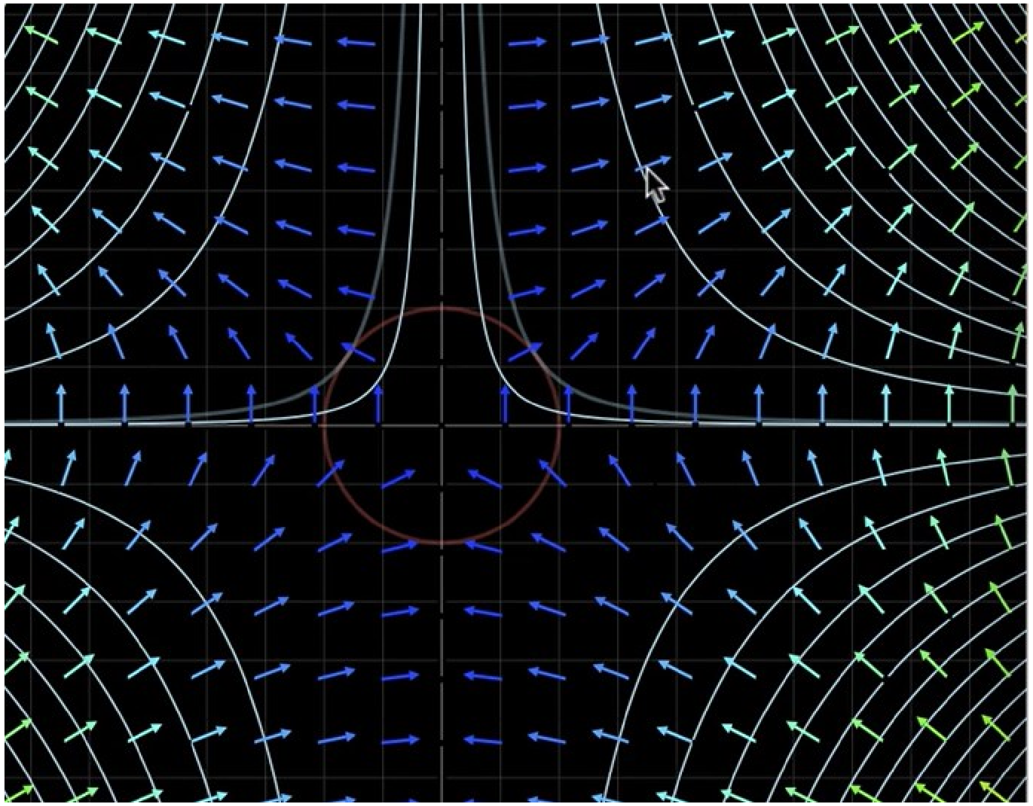
Maximize the function for some value of K







Lagrange Multiplier - Example



- Gradients are perpendicular to various contour lines(w.r.t to different values of function) of function f
- Function is not changing the value along the points on any the contour line

Constrained to Unconstrained Optimization - Lagrange Multiplier

- Maximum of f(x,y) under constraint g(x, y) is obtained when their gradients point to same direction
(when they are tangent to each other).
- Introduce a Lagrange multiplier λ for the equality constraint
- Mathematically,

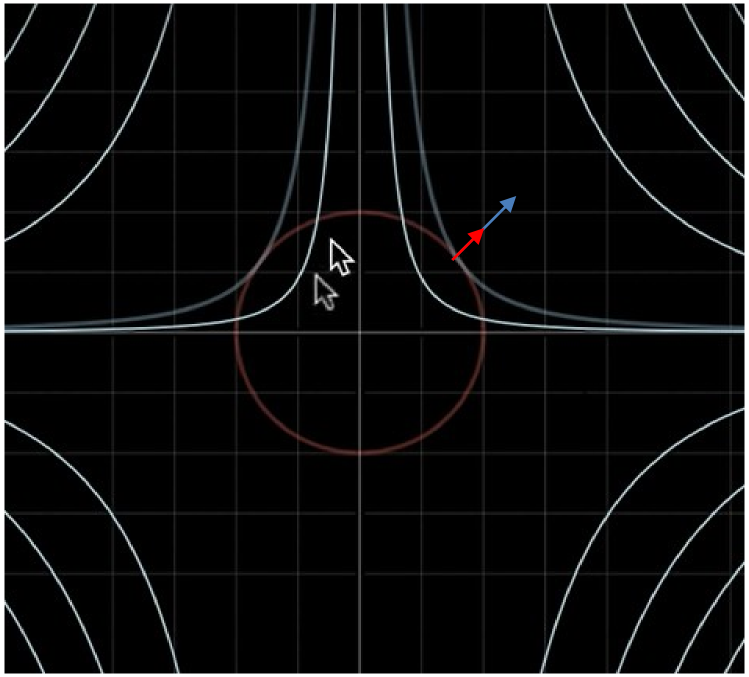
  $\nabla f(x,y) = \lambda \nabla g(x,y)$

  Equivalently:

  $L(x,y,\lambda) = f(x,y) - \lambda . g(x,y) - C$

  Optimize L using gradients

  $\nabla L(x,y,\lambda)=0$



Summary - Lagrange multiplier method

1. Construct the Lagrangian function by introducing one multiplier per constraint

2. Get the gradient ∇L of the of the Lagrangian

3. Solve for $\nabla L(x,y, \lambda)=0$

Example

$$\max_{x,y} xy \text{ subject to } x + y = 6$$

- Introduce a Lagrange multiplier λ for constraint
- Construct the Lagrangian

$$L(x, y) = xy - \lambda(x + y - 6)$$

-

- Stationary points

$$\frac{\partial L(x,y)}{\partial \lambda} = x + y - 6 = 0$$

$$\left.\begin{aligned} \frac{\partial L(x,y)}{\partial x} &= y - \lambda = 0 \\ \frac{\partial L(x,y)}{\partial y} &= x - \lambda = 0 \end{aligned}\right\} \Rightarrow x = y = \lambda$$

$$\Rightarrow x = y = 3$$

x and y values remain same even if you take +λ or -λ for equality constraint

2 x = 6
x = y = 3
λ = 3

Karush–Kuhn–Tucker (KKT) theorem

- KKT approach to nonlinear programming (quadratic) generalizes the method of Lagrange multipliers, which allows only equality constraints.

- KKT allows inequality constraints

Karush–Kuhn–Tucker (KKT) theorem

- Start with

  max f(x) subject to

$$g_i(x) = 0 \text{ and } h_j(x) \geq 0 \text{ for all } i, j$$

- f is the objective function, g_i (i=1 ….m) are the equality constraint functions and h_j (j=1…. *l*) are the inequality constraint functions

- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Take gradient and set to 0 – but satisfy other conditions also.

KKT conditions

- Make the Lagrangian function

$$\boxed{\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)}$$

- Necessary conditions to have a minimum are

Stationarity $\longrightarrow$ $\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$

primal feasibility condition $\longrightarrow$ $g_i(x^*) = 0 \text{ for all } i$

primal feasibility condition $\longrightarrow$ $h_j(x^*) \geq 0 \text{ for all } j$

dual feasibility condition $\longrightarrow$ $\mu_j \geq 0 \text{ for all } j$

complementary slackness condition $\longrightarrow$ $\mu_j^* h_j(x^*) = 0 \text{ for all } j$