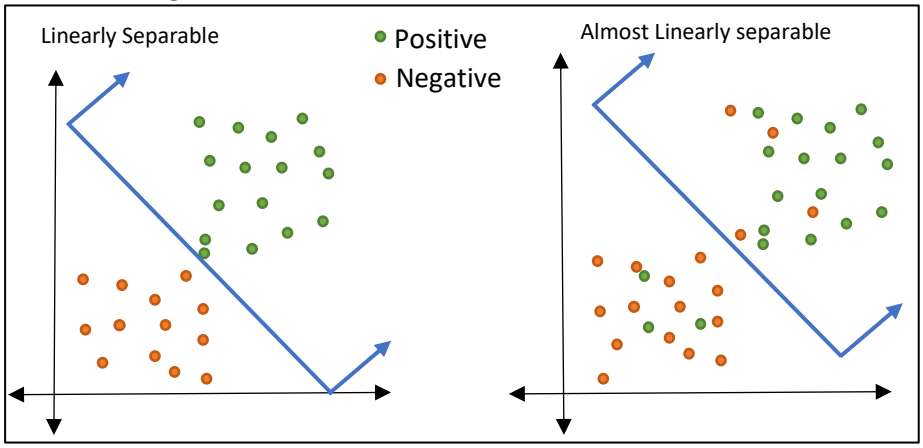
- **Geometric Intuition-Linear Regression**



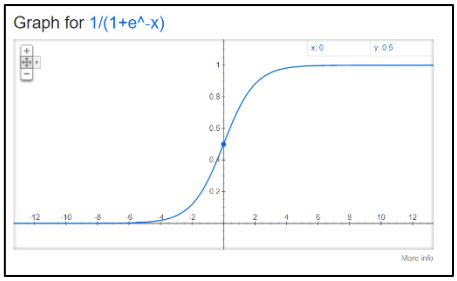
  - When a data is separated by a line it's called as linear regression.
  - If data can be separated into two parts by a line, it is called as linearly separable data. If it has some errors and can't be linearly separable is called as almost linearly separable data
  - Assumption of linear regression is that the data is perfectly linearly separable. In other words, the decision surface will be plane or line.
  - The main task in linear regression is to find out line or plane which will separate the data linearly.
    - Equation of a Plane $y = W^T x + B$ W is normal to plane & B is intercept
    - Distance of a point from plane $d_i = \frac{W^T x_i + B}{||w||}$ if B =0 & w is a unit vector $d_i = W^T x_i$
  - For geometric interpretation, +1 is positive class & -1 is negative class
  - When points are Plotted, one side we will have positive class & on other side will be negative class
    - $d_i = W^T x_i > 0 \rightarrow x_i \; belongs \; to \; positive \; class$
    - $d_j = W^T x_j < 0 \rightarrow x_j \; belongs \; to \; negative \; class$

x_i	Actual (y_i)	Predicted (y_i')	D = $W^T x$	SD = $y_i' * W^T x$	Validation
x_1	+1	+1	D > 0	SD > 0	Correctly Classified
x_2	-1	-1	D < 0	SD > 0	Correctly Classified
x_3	+1	-1	D > 0	SD < 0	Misclassified
x_4	-1	+1	D < 0	SD < 0	Misclassified

  - In linear regression, the main task is to find optimal plane W which gives maximum correctly classified points. Argmax means the index at which value will be maximum. Argmin is index at minimum value.

$$W^* = argmax_W \rightarrow \left[\sum_{i=1}^{n} y_i . W^T . x_i\right]$$

- **Logistic Regression**
  - Logistic Regression is a classification technique. Here, instead of a straight line, we fit a S shaped curve, called Sigmoid, to observations



  - The sigmoid function squashes the output from $[-\infty, +\infty]$ to $[0, 1]$. This fits the goal of classification.
  - By computing the sigmoid function of X (weighted sum of the input features), we get a probability of an observation belonging to one of the two categories.
  - Sigmoid is easily differentiable

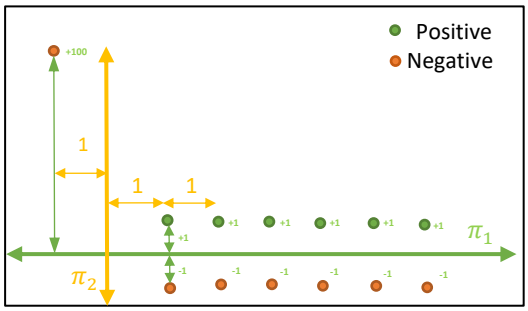
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  - the main task will be to find optimal W out of multiple W's which gives maximum correct distances. The distances function will be squashed through sigmoid function

$$W^* = argmax_W \rightarrow \left[\sum_{i=1}^{n} \sigma(y_i . W^T . x_i)\right] = argmax_W \rightarrow \left[\sum_{i=1}^{n} \frac{1}{1 + e^{-y_i . W^T x_i}}\right]$$

  - In the W^*formula, we will call $y_i . W^T x_i$ this term as Signed distance $\mathbb{Z}_i$
  - Due to squashing phenomenon, the effect of outlier on hyperplane selection reduces heavily.

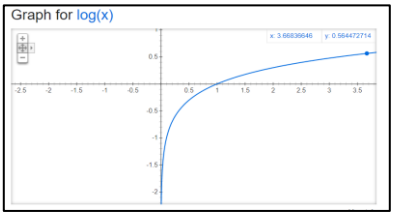
- We have some points belong to positive & some to negative class. We have got two hyperplanes π_1 and π_2 we will select one that gives maximum correctly classified points. The values next to point shows distance from respective plane

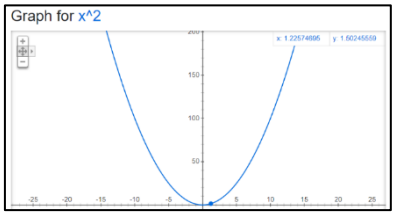


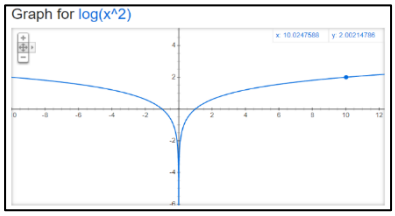
Pts	Class	D(Π1)	D(Π2)	Z(Π1)	Z(Π2)
1	1	1	1	1	1
2	1	1	2	1	2
3	1	1	3	1	3
4	1	1	4	1	4
5	1	1	5	1	5
6	-1	-1	1	1	-1
7	-1	-1	2	1	-2
8	-1	-1	3	1	-3
9	-1	-1	4	1	-4
10	-1	-1	5	1	-5
11	-1	100	-1	-100	1
TOTAL				-90	1

  - In the above example, Sum of signed distance of points from plane π_1 is -90 & from π_2 is 1.
  - As π2 is giving us better results, we will choose it as a classifier.
  - But intuitively, π2 is a bad classifier as no of correctly classified points are less than π1.
  - This is all happening due to single outlier point. Hence the output needs to be squashed that even if distance is greater after some threshold value, the function should output the threshold itself.
  - Thus, we use sigmoid function.

- **Mathematical formulation of Objective function**
  - To simplify above equation further, we will use a monotonic increasing function. It means always increasing or remains constant, and never decreasing.







    - $\log x$ is a monotonically increasing function
    - When $f(x) = x^2$,minima lie at $x = 0$
    - If $g(x) = \log x$ then for $g(f(x)) = \log x^2$,minima lie at $x = 0$
    - $\therefore argmin_x\ x^2 = argmin_x \log x^2$
    - x^2 is not a monotonically increasing function but $\log x$ is a monotonically increasing function.
    - Note-Generally, SGD optimization is used to solve the LR optimization problem and to minimize the loss. But the limitation of the Gradient Descent optimizer is it can be only used for convex function optimization. Sigmoid is NOT a convex function but Log of the sigmoid is a convex function
  - We can conclude that if $g(x)$ is a monotonically increasing function then

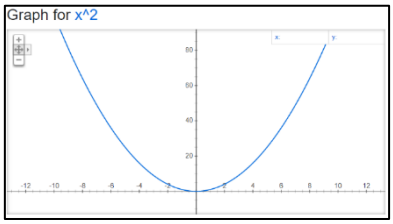
$$\begin{aligned} argmin_x\ f(x) &= argmin_x\ g(f(x)) \\ argmax_x\ f(x) &= argmax_x\ g(f(x)) \end{aligned}$$

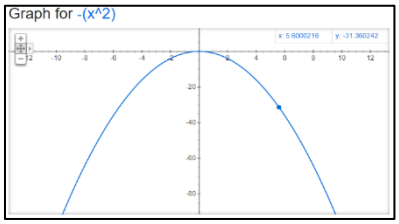
  - As sigmoid is monotonically increasing function

$$argmax_x\ \sigma(x) = argmax_x \log \sigma(x)$$

$$W^* = argmax_W \rightarrow \left[\sum_{i=1}^{n} \frac{1}{1+e^{-y_i.W^T x_i}}\right] = argmax_W \rightarrow \left[\sum_{i=1}^{n} \log \frac{1}{1+e^{-y_i.W^T x_i}}\right]$$

$$W^* = argmax_W \rightarrow \left[\sum_{i=1}^{n} -\log\left(1+e^{-y_i.W^T x_i}\right)\right] ; \log \frac{1}{x} = -\log(x)$$





  - From Above graph, $argmax_x\ f(x) = argmin_x - f(x)$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i.W^T x_i}\right)\right]$$

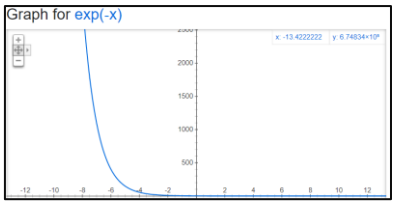
- **Weight Vector (Feature Importance)**
  - The Weight vector is the optimal weight vector of size $d = n_{features}$ normal to a hyper plane that separates data points into classes where positive data points are in the direction of W

$$W^* = w_i \in \mathbb{R}^d$$

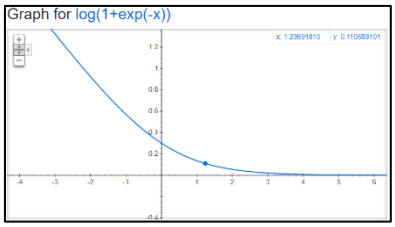
  - If w_i corresponding to feature f_i for a given query point is positive, then probability of the point belonging to positive class increases and vice versa.
  - For a query point if it's f_i increases and $w_i > 0$ then there are more chances that point belongs positive class
    - Case 1: If $w_i > 0$; $x_{qi} \uparrow \Rightarrow w_i x_{qi} \uparrow \Rightarrow \sigma(w_i x_{qi}) \uparrow \Rightarrow P(y_q = 1) \uparrow$
    - Case 1: If $w_i < 0$; $x_{qi} \downarrow \Rightarrow w_i x_{qi} \downarrow \Rightarrow \sigma(w_i x_{qi}) \downarrow \Rightarrow P(y_q = -1) \uparrow$
- **L2 (Ridge) Regularization: Overfitting and Underfitting**

Let's say $\mathbb{Z}_i = y_i.W^T x_i$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log(1 + \exp(-\mathbb{Z}_i))\right]$$



  - From the above graph, we can say that $e^{-x} \geq 0$. So, $1 + \exp(-\mathbb{Z}_i) \geq 0$
    - As $\log(1) \geq 0$
    - $\log(1 + exp(-\mathbb{Z}_i) \geq 0$
    - $[\sum_{i=1}^{n} \log(1 + exp(-\mathbb{Z}_i))] \geq 0$



  - From above graph, function minima will be 0 when $\mathbb{Z}_i$ is positive
  - If $Z_i > 0$ AND $\mathbb{Z}_i \rightarrow +\infty$, then $\frac{1}{\mathbb{Z}_i} = \frac{1}{\infty} = 0$

    $\exp 0 = 0$; $\log(1 + 0) = 0$ for all datapoints(i)
  - For all datapoints(i), $\mathbb{Z}_i = y_i.W^T x_i$ should be positive & tends to $+\infty$
    - x_i, y_i are the datapoint pairs which are constants and can't be changed.
    - $\therefore W^T \rightarrow +\infty$
    - Also, when $\mathbb{Z}_i > 0$ is positive, datapoint is correctly classified.
  - Hence, if we pick W such that,
    - All points are correctly classified
    - $\mathbb{Z}_i \rightarrow +\infty$ for all datapoints(i)
  - $W^T \rightarrow +\infty$ means $w_i \rightarrow \pm\infty$ i.e. weights are becoming very large
  - By increasing weight vector, we can get good result on train data. We may classify all points correctly but some of the points will include outliers too.
  - This Is overfitting as the same result may not be reproduced for test data
  - While constructing optimization problem we have missed one key aspect which is W is normal to hyperplane.
  - To avoid this overfitting problem, we have to add a regularization term $\lambda W^T W$ to above formula. First part is called as 'Loss Term'

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i.W^T x_i}\right)\right] + \lambda W^T W$$

$$W^T W = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} * [W_1 \quad W_2 \quad W_3] = {W_1}^2 + {W_2}^2 + {W_3}^2 = \sum_{j=1}^{d} W_j^2 = ||W||_2^2$$

$$W^* = argmin_{\ W} \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i . W^T x_i}\right) \right] + \lambda ||W||_2^2$$

- With the above formula, the tug of war between loss & regularization term will start such that if we get a W where $\mathbb{Z}_i \rightarrow +\infty$ the loss term will be 0 but regularization term will be very large
  When $W^T \rightarrow +\infty$; $W^* = [0] + \infty$
  When $W^T \rightarrow 0$; $W^* = [\infty] + 0$
- λ is hyperparameter in logistic regression
  - If $\lambda = 0$ & $W^T \rightarrow +\infty$, the regularization term will become 0 and ultimately loss term will be ∞ the model is overfitting.
  - If $\lambda = large$, the influence of loss term will be reduced and model is left with regularization term which doesn't have any reference of training data, the model is underfitting

- **L1(Lasso) regularization and sparsity**
  - In L1 regularization, we keep everything as it is but we replace L2 norm with L1 norm

$$W^* = argmin_{\ W} \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i . W^T x_i}\right) \right] + \lambda ||W||_1$$

  - It serves the same purpose as L2 regularization but has an additional advantage of sparsity
  - It means that the weight vector is a sparse & has many of the values 0 i.e., less important features
  - By using L1 norm, it also proven fact that, for less important features corresponding weight vector i.e., W_i becomes 0 while as it becomes a very small value if L2 regularization is used.
  - By using elastic net, we use both L1 & L2 norms for logistic regression

$$W^* = argmin_{\ W} \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i . W^T x_i}\right) \right] + \lambda ||W||_1 + \lambda ||W||_2^2$$

- **Probabilistic derivation of Logistic regression**
  - Assumption:
    - Y is Bernoulli random distribution $(y_k \in 0,1)$
    - $X = (X_1, X_2, X_3, \dots, X_d)$ where feature X_i is continuous random variable
    - For each X_i, $P(X_i | y = y_k)$ is a gaussian distribution with $N(\mu_{ik}, \sigma_i)$
    - Two features X_i & X_j are conditionally independent given Y
    - So logistic regression can be derived as,

$$Logistic\ Regression: Gaussian\ Naïve\ Bayes + Bernoulli$$

    Read further
  - The probabilistic derivation is as follows where the loss term will be derived as follows $(y_i \in 0,1)$

$$W^* = argmin_{\ W} \rightarrow \left[\sum_{i=1}^{n} -y_i \log p_i - (1 - y_i) \log(1 - p_i) \right] + + \lambda ||W||_1 + \lambda ||W||_2^2$$

$$p_i = \sigma(w^T x_i) = \frac{1}{1 + e^{-W^T x_i}}$$

  - The geometric derivation$(y_i \in -1,1)$

$$W^* = argmin_{\ W} \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i . W^T x_i}\right) \right] + \lambda ||W||_1 + \lambda ||W||_2^2$$

  - The loss term in probabilistic derivation & geometric derivation are same. For this calculation, we will not consider the regularizer part as that is same for both equations
  - **Case1-** $y_i = positive$
  - Geometric $y_i = +1$

$$W^* = argmin_{\ W} \rightarrow \left[\sum_{i=1}^{n} \log\left(1 + e^{-1 . W^T x_i}\right) \right]$$

  - Probabilistic $y_i = +1$
  - Here, only first term remains as $1 - y_i = 0$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -1 \log p_i - (1-1)\log(1-p_i)\right]$$

$$p_i = \sigma(w^T x_i) = \frac{1}{1+e^{-W^T x_i}}$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -1 \log \frac{1}{1+e^{-W^T x_i}}\right]$$

$$-\log x = \log \frac{1}{x}$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log\left(1+e^{-W^T x_i}\right)\right]$$

This is same as geometric derivation

- **Case1-** $y_i = negative$
- Geometric $y_i = -1$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log\left(1+e^{+1.W^T x_i}\right)\right]$$

- Probabilistic $y_i = 0$
- Here, only second term remains as $y_i = 0$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -0 \log p_i - (1-0)\log(1-p_i)\right]$$

$$p_i = \sigma(w^T x_i) = \frac{1}{1+e^{-W^T x_i}}$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -1 \log\left(1 - \frac{1}{1+e^{-W^T x_i}}\right)\right]$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -1 \log\left(\frac{1+e^{-W^T x_i}-1}{1+e^{-W^T x_i}}\right)\right]$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -\log\left(\frac{e^{-W^T x_i}}{1+e^{-.W^T x_i}}\right)\right]$$

$$-\log x = \log \frac{1}{x}$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} \log\left(\frac{1+e^{-W^T x_i}}{e^{-W^T x_i}}\right)\right]$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -\log\left(\frac{1}{e^{-W^T x_i}} + \frac{e^{-W^T x_i}}{e^{-W^T x_i}}\right)\right]$$

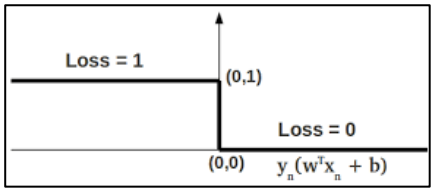
$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -\log\left(1 + \frac{1}{e^{-W^T x_i}}\right)\right]$$

$$\frac{1}{e^{-x}} = e^x$$

$$W^* = argmin_W \rightarrow \left[\sum_{i=1}^{n} -\log\left(1+e^{W^T x_i}\right)\right]$$

- **Loss minimization interpretation of Logistic Regression (0-1 Loss Function)**
  - We want loss function which outputs +1 for incorrectly classified points & 0 for correctly classified points.
  - Such loss function is 0-1 loss function which provides output as follows
    - If signed distance $\mathbb{Z}_i = y_i.W^T x_i > 0$, the point is correctly classified
    - If signed distance $\mathbb{Z}_i = y_i.W^T x_i < 0$, the point is incorrectly classified

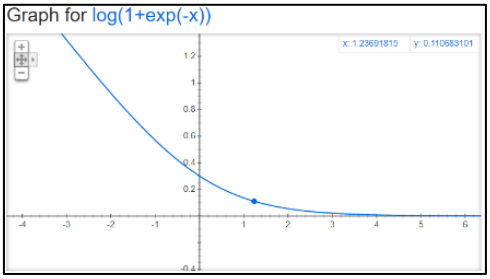
$$0\text{-}1\ Loss(\mathbb{Z}_i) = \begin{matrix} 1 \text{ if } \mathbb{Z}_i = y_i.W^T x_i < 0 \\ 0 \text{ if } \mathbb{Z}_i = y_i.W^T x_i > 0 \end{matrix}$$



- Our main goal is to find a hyperplane which gives minimum incorrectly classified points.

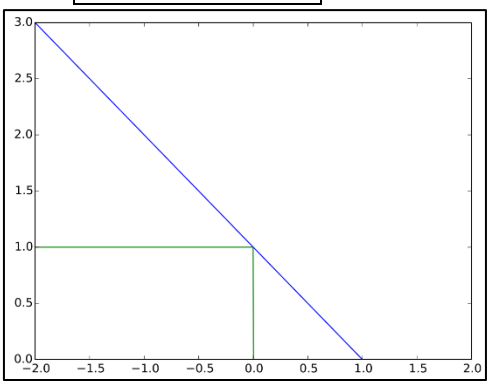
$$W^* = \underset{W}{argmin} \rightarrow \left[\sum_{i=1}^{n} \text{0-1 } Loss(x_i, y_i, w)\right]$$

- To solve any optimization problem, we have to use differentiation. Any function $f(x)$ is differentiable when $f(x)$ is continuous.
- In the above 0-1 loss function interpretation, there is a discontinuity i.e., there is no clarity at $\mathbb{Z}_i = 0$. Hence, it's not differentiable.
- We have to use approximation of this function to overcome above problem
- logistic loss is one such approximation of 0-1 loss i.e. $f(\mathbb{Z}_i) = \log(1 + \exp(-\mathbb{Z}_i))$ we interpret that,
  - When $Z_i = y_i . W^T x_i > 0$, when point is correctly classified the $f(\mathbb{Z}_i) \uparrow$
  - When $\mathbb{Z}_i = y_i . W^T x_i < 0$, when point is incorrectly classified points. the $f(\mathbb{Z}_i) \rightarrow \infty$
  - When we apply log loss minimization, we get logistic regression.



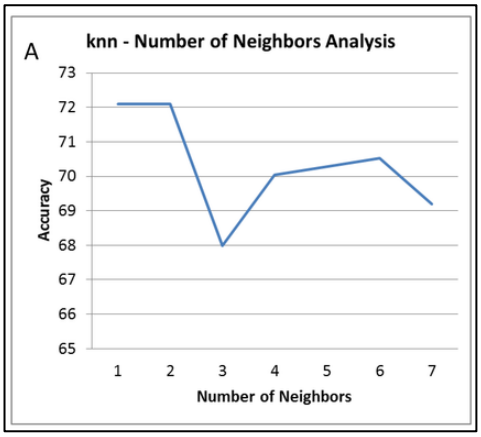
- Similar to log loss function we can have many other functions. Another such most studied function is hinge loss

$$l(\mathbb{Z}_i) = \max(0, 1 - \mathbb{Z}_i)$$



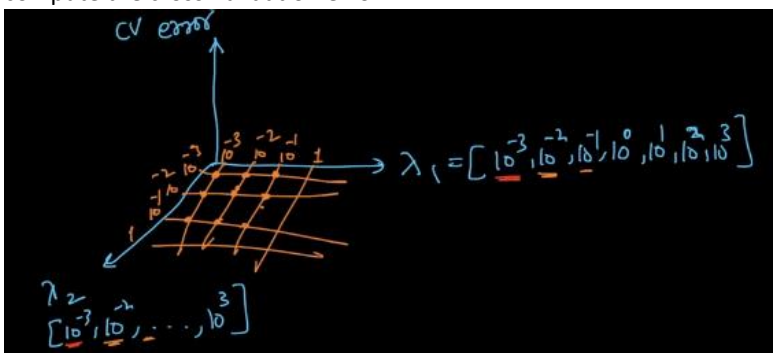
  - When $Z_i = y_i . W^T x_i > 0$, when point is correctly classified, after $\mathbb{Z}_i = 1$ the $l(\mathbb{Z}_i) = 0$
  - When $\mathbb{Z}_i = y_i . W^T x_i < 0$, when point is incorrectly classified points the $l(\mathbb{Z}_i) \uparrow$
  - When we apply hinge loss, we get Support vector machines
  - When we apply squared loss, we get linear regression. For exponential loss we get Ada Boost.
- Loss function is basically the function which is minimized and whose minima is tried to be achieved. it is different for different algorithms. We can think of it as a metric against which the model is evaluated.
  - For e.g., in logistic regression we want a hyperplane which gives max correct classification. E further do some modifications to equation of plane & arrive at loss function which we minimize.

- **hyperparameter Search: Grid search and random search**
  - λ will be the hyperparameter in logistic regression similar to k & α in KNN & Naïve Bayes respectively
    - If $\lambda = 0$, the regularization term will become 0 and ultimately loss term be ∞ which is actually saying the model is overfit
    - If $\lambda = large$, the influence of loss term will be reduced and model is left with regularization term which doesn't have any reference of training data , this will imply that the model is underfit
    - $\lambda \in \mathbb{R}$



- In case of KNN we calculated KNN by plotting graph between accuracy/Error and no of neighbours
- We found the optimal K by find lowest error or highest accuracy
- In case of KNN , Hyperparameter K was Discrete random variable.
- We plot similar graph like this in logistic regression hyperparameter λ
- The problem is λ will be a continuous random variable.
- Here we will search for optimal hyper parameter in following set of values $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$

- For each of this value of λ Compute cross validation error and select the best hyper parameter value from the plot
- If we have two hyperparameters λ_1 , λ_2 we will plot grid between λ_1 , λ_2 with $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ for each of pairs we will compute the cross-validation error



- Let's say we have m possibilities for k number of λ we need to compute the model m^k times
- In case of random search, we will pick any random values between a range like { 10^{-4}, 10^{4} } And find the corresponding CV Error in turn provides best hyper parameter choices that optimizes the algorithm

- **Column Standardization (Mean centring and scaling)**
  - In logistic regression , since we are using distances to predict the class variable and weight of each feature will impact the distance.
  - So, we have to perform standardization on data

$$x_{ij'} = \frac{x_{ij} - \pi_j}{\sigma_j} = \frac{mean\ centering}{scaling}$$

- **Feature importance and Model interpretability**
  - Each feature will have a corresponding weight to it.

f_1	f_2	.	f_j	.	f_d
w_1	w_2	.	w_j	.	w_d

  - We have d features and through optimization we determine W vector of d dimensionality, each element in W vector corresponds to respective features
  - If we assume each feature is independent then $abs|w_j|$ increases, $W^T x_i$ also increases as

$$W^T x_i = \sum_{j=1}^{d} w_j . x_{qj}$$

    - This implies if $w_j > 0$ & $large, P(y_q = +1) \uparrow$
    - If $w_j < 0$ & $large, P(y_q = -1) \uparrow$
  - We can determine the important features by w_j
    - E.g., we have to predict male (+1) or female (-1) and we have a feature called hair length
    - As majority of females have long hairs, the value of feature weight will be large negative, $P(y_q = -1)$ is more
    - Similarly, if we have a feature height, the value of feature weight will be medium negative, $P(y_q = +1)$ is more
  - This also gives us model interpretability such that if we are saying query point belongs to particular class, we can give reasoning of the weight values of specific feature
  - If the points are present near the hyperplane then the class label prediction is given with low confidence because the class label might change when the hyperplane gets slightly changed/rotated.

- Hence as the points x_i are same, in order to get larger magnitude values for $W^T x_i$, we need to depend on the values of 'W'. For that reason, we expect majority of the weight components to have w_j values to be large.
- Top weighted features are considered as most important features for model

- **Collinearity of features**
  - Feature importance interpretation is done from weight vectors assuming independence. If there is co-linearity then we cannot interpret feature importance from weight vector.
  - Two features are collinear if a feature can be expressed as a function of another feature.

$$\boxed{\begin{matrix} f_i = \alpha f_j + \beta \\ weight = \alpha * height + \beta \end{matrix}}$$

  - Multi collinear feature is a feature that can be expressed as a function of multiple features

$$\boxed{\begin{matrix} f_1 = \alpha f_2 + \alpha f_3 + \alpha f_4 + \alpha f_5 \\ f_1, f_2, f_3, f_4, f_5 \ are \ multicolinear \end{matrix}}$$

  - Weight vectors are not much useful if features are colinear

F	W	Xq
F1	1	Xq1
F2	2	Xq2
F3	3	Xq3

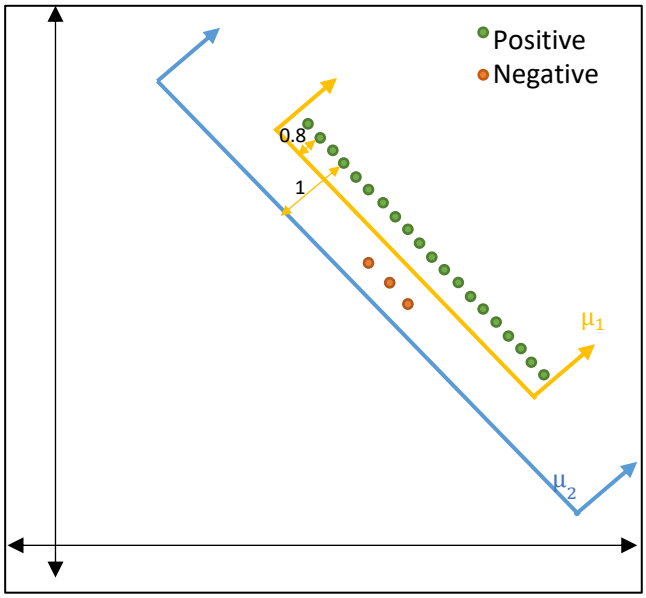
$$\boxed{W^{*T}.X_q = x_{q1} + 2x_{q2} + 3x_{q3}}$$

  - If f1 & f2 are colinear such that $f2 = 1.5 * f1$

$$\boxed{W^{*T}.X_q = x_{q1} + 2 * 1.5 * x_{q1} + 3x_{q3} = 4 * x_{q1} + 3 * x_{q3}}$$

  - Now, the weight vector was originally $\boxed{W^* = 1 \quad 2 \quad 3}$ now due to collinearity it has become $\boxed{\widetilde{W} = 4 \quad 0 \quad 3}$
  - This implies both $\boxed{W^* = 1 \quad 2 \quad 3}$ $\boxed{\widetilde{W} = 4 \quad 0 \quad 3}$ will give same results
  - To conclude if features are colinear, then weight vector can change arbitrarily
  - **Perturbation test to check the presence of collinearity (assignment 8D)**
    - Step 1: Find the weight vectors of the feature by fitting a logistic regression model. $W1$
    - Step 2: Add a random noise e to the dataset. Then again find the weight vectors for the features of the noise added dataset. Say. $W2 = W1 + e$
    - Step 3: Compute the difference between $W1$ and $W2$ vectors.
    - Step4: If the $W1$ and weight $W2$ differs much then the features are said to be multicollinear.
    - Then in this case weight vectors can't be used for feature importance. So, we should go with some other techniques such as forward feature selection.

- **Logistic Regression with Imbalanced data: A Geometric View**



  - In the above example we are having an imbalanced dataset such that It has 20 positives & 3 negatives
  - For Plane μ_1, the signed distance of each point from plane is 0.8

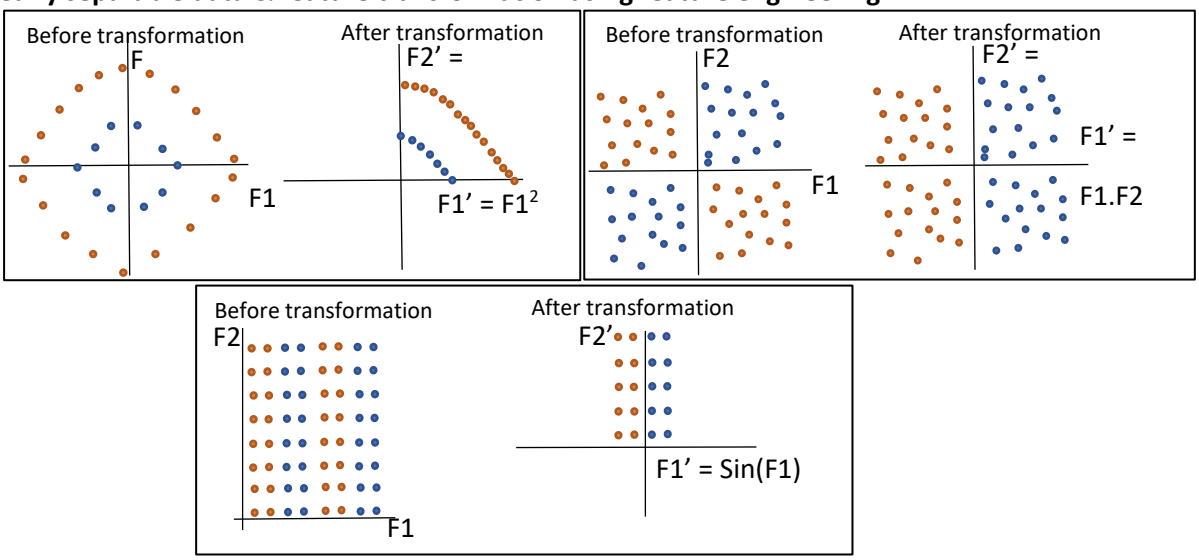
$$W = \sum_{i=1}^{n} y_i . W^T . x_i$$

for μ_1 ,$20 * 0.8 + 3 * 0.8 = 18.4$

for μ_2 ,$20 * 1.0 + 3 * 0.0 = 20.0$

- Here μ_2 is better as compared to μ_1 for classification
- When dataset is imbalanced dataset having more values in positive class, our hyperplane will move as away as possible from positive data points in order to classify them correctly but will classify negative data points incorrectly**.**

- **Train & Run Time Space & Time Complexity**
  - Training Logistic Regression: Solving the optimization problem using Stochastic Gradient Descent;
  - Train time complexity: O(nd)
  - Run time: Space: O(d) Time: O(d)
  - Logistic Regression is applicable for low latency applications; popular algorithm for internet companies, memory efficient at run time.
  - If dimensionality is large, L1 regularization inducing Sparsity can be applied to reduce run time complexity; As the hyper parameter increases Sparsity also increases; this achieves both a suitable bias variance trade off and the low latency requirements;
  - As λ increases, bias increases, latency decreases and Sparsity increases; but the model is just a working model not an optimized model as regularization and Sparsity are induced;
- **Summary**
  - Decision surface: Linear Hyperplane
  - Assumption: data is linearly separable or almost linearly separable;
  - We also have good interpretability
  - **Imbalanced data**: Up sampling or down sampling has to be applied.
    with imbalanced data we will have incorrect hyper planes that can optimize the loss function where the hyper plane can make all the data points lie on one side optimizing the loss function for majority class;
  - **Outliers**: due to sigmoid function impact of outlier is less.
  - We can compute W from D_{train} through training and generate distances through$W^T x_i$
    the points that have large distances are outliers can be removed from D_{train} . the model is retrained on the outlier free dataset to get final Weight vector
  - Missing values: mean, median or model imputation;
  - Multi class classification: One Vs Rest and Max entropy/SoftMax/Multinomial LR models
  - Input similarity matrix: Extension of LR, Kernel LR (SVM) can be trained on similarity matrix;
  - Best case: Data is linearly separable, low latency, faster to train; high dimensionality (higher chance that data is linearly separable and we can use L1 regularization)
  - Worst case: Non-linear data;
- **Non-linearly separable data & feature transformation using feature engineering**



  - Typical Transforms for real valued features
    - **Polynomial Features-** F1*F2, $F1^2$, $F2^2$, $F1^3$, $F2^3$, $F1^2$.F2
    - **Trigonometric Features-** Sin(F1), Sin(F2), Sin(F1). Cos(F2), $Sin(F1)^2$
    - **Boolean Features-** OR, AND, XOR
    - **Mathematical functions-** Log (), exp ()