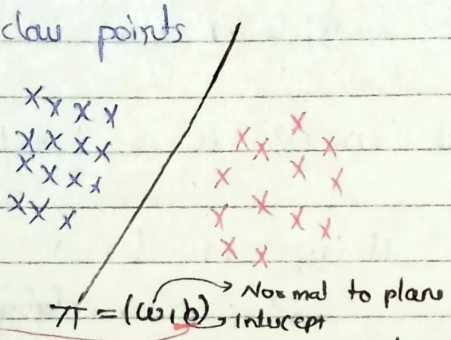
6H- Logistic Regression

→ Geometric Intuition of Logistic Regression.

- Don't go with name regression, it is a classification algo.
- It is a simple, elegent classification algorithm.
- Logistic regression can be interpreted/derived from 3 ways
  - (i) Geometry
  - (ii) Probability
  - (iii) Loss-func

We will stad with geometric intuition, then take a brief look on probabilistic intuition and then loss-func

Imagine we have 2 class points



If my data is linearly seperable or almost linearly seperable

Assumption in LR :- classes are almost/perfectly linearly seperable.

We have data set $D_n := \{+ve, -ve\}$

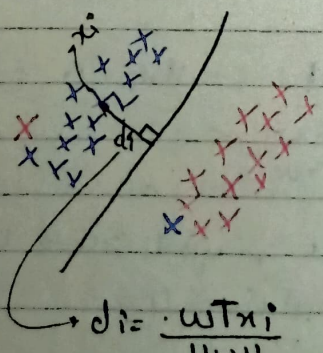
task is to find the plane $\pi(w, b)$ that separates the classes.

→ If assumption is wrong, then LR will not work.

Note: $y_i = +1$ for +ve class

$= -1$ for -ve class

It is not zero as previous methods.



$$d_i = \frac{w^T x_i}{\|w\|}$$

let w is unit vector the $\|w\| = 1$

$\therefore$ $d_i = w^T x_i$ → positive point

$d_i = w^T x_i > 0$ (x_i is in same side of w)

let a point from -ve class x_j, and d

$d_j = w^T x_j < 0$ (x_j is in opposite dir of w)

classifier:

if $w^T x_i > 0$:

~~else~~ $y_i = +1$

else

$y_i = -1$

* decision surface in LR is a Line/plane.

Case1: $y_i w^T x_i$: If $y_i = +1$ (+ve)

$w^T x_i > 0$ → classifier says it is positive point

> 0 & $y_i = +1$

then plane is correctly classifying the point.

Case2: $y_i = -1$, $w^T x_i < 0$ ⇒ LR concluding that x_i is -ve point

$y_i * w^T x_i > 0$

-ve -ve

Hence for both +ve & -ve point if $y_i w^T x_i > 0$ then model is correctly classifying the point x_i

Case 3: $y_i = +1$, $w^T x_i < 0$ ⇒ LR concluding that x_i is -ve

$y_i \; w^T x_i < 0$

+ve -ve

Case 4: $y_i = -1$, $w^T x_i > 0$ ⇒ LR concluding that x_i is +ve

$y_i * w^T x_i < 0$

-ve +ve

$\therefore$ For both +ve & -ve point if $y_i w^T x_i < 0$ then model is incorrectly classifying the point x_i

A good classifier make min no. of wrong classification.
We need $y_i w^T x_i > 0$

~~find a plane~~ -

But we have n data points, we want to maximize.

$$\underset{w}{\text{maximize}} \sum_{i=1}^{n} y_i w^T x_i$$

(y_i, x_i) → they are fixed bcoz they come from training data set

(w) → This is the only thing that can be change.

Hence we want w (↳ plane) which maximize the $\sum_{i=1}^{n} y_i w^T x_i$

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} y_i w^T x_i$$

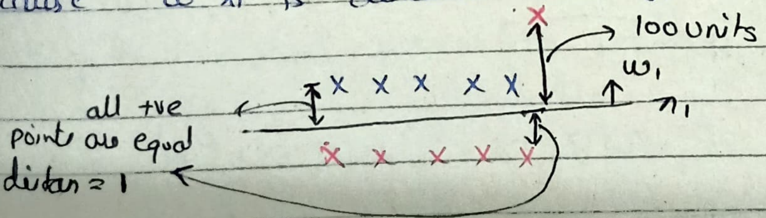
→ our final problem, althoug we modify it later.

★ Squashing & sigmoid function:-

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} y_i w^T x_i$$ // optimization problem.

Let's understand this function:

Let $y_i w^T x_i$ is a signed distance, signed distance because $w^T x_i$ is distance and y_i is sign $\{+1, -1\}$

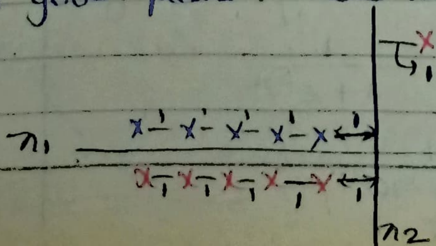


Case 1: π_1 is my seperator

$\sum y_i w^T x_i$:- $(1+1+1+1+1+1+1+1+1+1 - 100)$ (distance of correctly classified ↑, -100 ↑ misclassification)

$= \underline{-90}$

Here we get -90, due to an outlier point.
But π_1 seems a good plane. Now imagine another plane in above data point.



$\sum y_i w^T x_i$ ⇒

⇒ $(1+1+1+1+1+1 -1-1-1-1-1)$

$= \underline{1}$

Here we get +1 , so we might wrongly conclude that π_2 is a better separator that π_1. It is all due to outlier which is favouring π_2 instead of π_1. One single outlier is impacting my model.

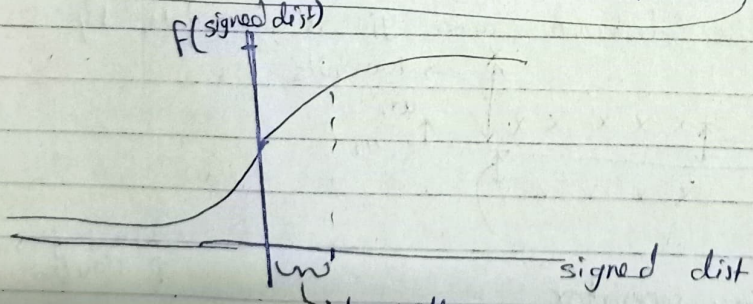
Hence maximizing sum of signed distance formulation is prone to outlier. We need to modify it. We do it using squashing.

→ Squashing: Instead of using signed distance,

- If signed dist is small: use it as is
- If signed dist is large: make it a smaller value.

In our previous example: distance from π_1, all the point have small distance i.e. 1, but only outlier have large distance of 100. we need to somehow decrease that.

We need a function that takes the signed ~~dist~~ distance value and ~~do~~ works as above 2 points.

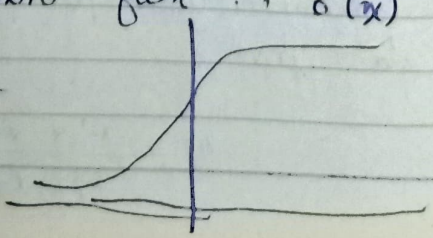


for smaller values func^n should use them as it is hence increase as dist increase, for larger value it should flatten or not increase so much.

One such func is sigmoid func^n : $\sigma(x)$

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

For max value of $\sigma(n) = 1$
min val is $\pi_i = -\infty$
$\sigma(0) = 0.5$



Now our formula becom

$$w^* = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \sigma(y_i w^T x_i)$$

There are various other fuction that can do our task but sigmoid is choosen because

① It have nice probabilistic interpretation

$\sigma(y_i w^T x_i)$ → returns probability of $y_i = 1$

$\sigma(0) = 0.5$

→ distance from plane is zero i.e point exactly lies on the plane hence in that case probability of point belonging to +ve or -ve class is equal.

$$w^* = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \sigma(y_i w^T x_i)$$

$$= \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \frac{1}{1+e^{(-y_i w^T x_i)}}$$

② It is easy to differentiate. To actually solve our optimization problem we need a func that can easily differentiable.

→ optimization problem :-

Lets try to solve our optimization problem

$$w^* = \operatorname{argmax} \sum_{i=1}^{n} \sigma(y_i w^T x_i)$$

But Before that we need to understand monotonic func^n.

monotonic func^n :- A func $g(x)$ is called monotonic if

if $x_1 > x_2$ then $g(x_1) > g(x_2)$

$g(x)$ is monotonic increasing func^n

We can use monotonic func^n to solve optimization problem.

Ex :- $x^* = \underset{x}{\operatorname{argmin}} (x^2)$ (we want x st x^2 is minimize)

$f(x) = x^2$

let $g(x) = \log(x)$ $x' = \underset{x}{\operatorname{argmin}}\, g(f(x))$

$$x^* = \underset{x}{\text{argmin}}\ g(f(x))$$

$$= \underset{x}{\text{argmin}}\ \log(x^2)$$

claim: $x^* = x$ because $g(x)$ is monotonic func^n.

Now lets go back to our optimization problem:

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} \frac{1}{1+\exp(-y_i w^T x_i)}$$

$g(x) = \log(x)$: monotonic func^n

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} \log\left(\sigma(y_i w^T x_i)\right)$$

$$= \underset{w}{\text{argmax}} \sum_{i=1}^{n} \log\left(\frac{1}{1+e^{-(y_i w^T x_i)}}\right)$$

$$= \underset{w}{\text{argmax}} \sum_{i=1}^{n} -\log\left(1+\exp(-y_i w^T x_i)\right) \quad \left\{\because \log\left(\frac{1}{w}\right) = -\log w\right\}$$

we know that $\underset{x}{\text{argmax}}\ f(x) = \underset{x}{\text{argmin}}\ (-f(x))$

Hence

$$\boxed{w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n} \log\left(1+\exp(-y_i w^T x_i)\right)}$$

This is the optimization problem of LR $\quad y_i \in \{1, -1\}$ $\quad \not\in \{1, 0\}$

Always remember that in LR $y_i \in \{1, -1\}$.

→ weight-vector:- The optimal w^* we get is also called weight-vector because it is a d-dim vector where d → # of features.

w^T x

$1 \times d$ $\quad d \times 1$

It is called weight vector because it assign weights to each feature $w = \langle w_1\ w_2\ w_3 \ldots w_d \rangle$

$x_i = \langle f_1\ f_2\ f_3 \ldots f_d \rangle$

Interpretation of w :-

Case ① If $w = \langle w_1\ w_2 \dots w_i \dots w_d \rangle$

↳ If w_i is +ve and we have a $x_q \langle x_{q1}, x_{q2} \dots x_{qi} \dots x_{qd} \rangle$

If w_i is +ve and $x_{qi} \uparrow$ then $w_i x_{qi}$ increases.

Then $\sum w_i x_{qi} \uparrow$

Then $\sigma(w^T x_q) \uparrow$

Then proba of $y_q = +1 \uparrow$

Case ② If w_i = -ve,

as $x_{qi} \uparrow$ $w_i x_{qi} \downarrow \rightarrow \sum w_i x_{qi} \downarrow \rightarrow \sigma(w^T x_{qi}) \downarrow$

$P(y_q = -1) \uparrow \leftarrow P(y_q = +1) \downarrow$

→ L2 regularization : overfitting vs Underfitting

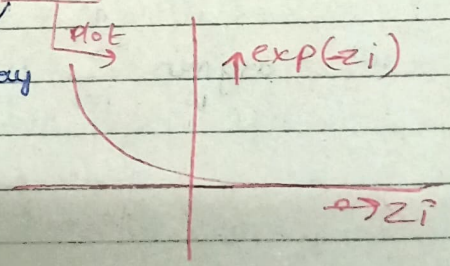
$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i w^T x_i)\right)$$

Let $z_i = y_i w^T x_i$

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log\left(1 + \exp(-z_i)\right)$$

This is always +ve

$\exp(-z_i) > 0 \quad (\forall z_i)$



$\log(1 + \exp(-z_i))$ (>0, >1)

$\therefore \log(1 + \exp(-z_i)) \geq 0$

$\therefore \sum \log(1 + \exp(-z_i))$ (≥ 0) Then whole sum ≥ 0

Our optimization problem was to get w that minimize the sum

We just showed that $\sum \log(1 + \exp(-z_i)) \geq 0$ ∴ min value it can take $= 0$.

So, when does $\sum_{i=1}^{n} \log(1+\exp(-z_i))$ becomes 0.

As $z_i \to \infty$, $\exp(-z_i) \to 0$

$\therefore \log(1+\exp(-z_i)) \to 0$

$\therefore \sum_{i=1}^{n} \log(1+\exp(-z_i)) \to 0$

Hence when $z_i \to \infty, \forall i$ then $\sum_{i=1}^{n} \log(1+\exp(-z_i)) \to 0$.

$\downarrow$

$y_i w^T x_i$ → modify w s.t each $z_i \to \infty$.

Fixed

If we pick w s.t

(a) all training points are correctly classified

(b) $z_i \to +\infty \ \forall i$.

Then that is Best w

But what if some of the points are outlier. We still manage to correctly classify them and that is nothing but overfitting.

To get rid of this problem, we modify the formula using regularization.

$$w^* = \arg\min_w \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i)) + \lambda w^T w$$

$\lambda w^T w \downarrow$

$\lambda \|w\|_2^2$ ← L_2 Norm

loss term

Regularization term.

If $w \to +\infty$ or $-\infty$

loss term ↓ or becomes 0

but Regularization term ↑

Hence it makes a balance.

It avoid w's to become $+\infty$ or $-\infty$ so that it is not overfit.

λ is a hyper parameter, it is not given we need to calculate it.

If $\lambda = 0$ then there is no regularization term hence it cause overfit

If λ = very large then regularization term dominates and hence we are not using training data anymore it is underfit.
Hence, we need to get a balanced λ.

L_1 regularization and sparsity.

Now, our optimization formula becomes.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i)) + \lambda \|w\|^2$$

logistic loss — L_2 regularizer.

In L_1 regularization, instead of L_2 Norm of w we use L_1 - norm of w.

L_1 norm of any vector $v = \|v\|_1 = \sum_{i=1}^{d} |w_i|$

Now our optimization formula becomes.

$$w^* = \underset{w}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \log(1+\exp(-z)) + \lambda \|w\|_1 \right)$$

L_1 regularizer have a major advantage:- sparsity.

'Sparsity':- If $w = \langle w_1, w_2, \ldots, w_d \rangle$
solution to LR is said to be sparse if many w_i's are zero.

If we use L_1 reg in LR, all the unimportant features $\underline{\text{become}}$ w_i zero.

* L_1 creates sparsity in w, as compared to L_2.

✱ Probabilistic interpretation of Logistic regression.

Till now we have derived LR using Geometry. Now we will look at probability method to derive LR.

We saw Naive Bayes ① if features are real valued, we assumed that conditional probability of feature follow Gaussian distribution

② the class label $y_i = +1$ or 0 was assumed ✱ Bernoulli R.V

with above 2 assumption we can derive LR.

$\therefore$ LR = GNB + Bernoulli class label

$P(x_i/y_i)$ $\quad$ $y_i \sim$ Bernoulli

$$w^* = \arg\min_w \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i) + \text{regularization}$$

where $p_i = \sigma(w^T x_i)$

$\in \{+1, 0\}$

✱ Loss minimization interpretation of LR :-

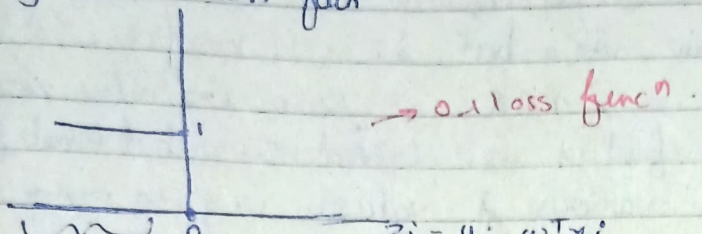
Till now we have learned geometry & probability derivation of LR. Now we will learn loss minimization derivation. Forget about all maths till now, let's imagine how we will design a classification func^n.

Let we assign $+1$ ✱ or incorrect classification and 0 for correct classificatin. Then we ✱ have to minimize

$w^* = \arg\min_w$ (number of incorrectly classified points)

Let we have a func^n which give $+1$ for incorrect classi & 0 for correct classification.

plot of ideal loss fun'



→ 0-1 loss func^n.

1 for incorrect cla, when z_i = +ve we have 0 & for correct clas

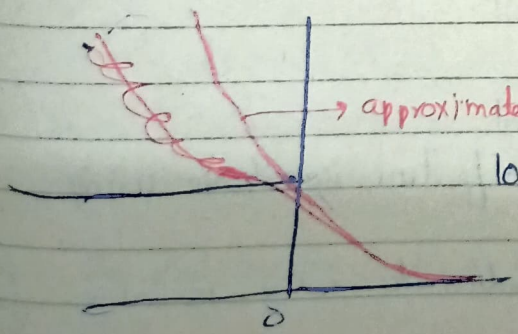
such a loss fun^n is called 0-1 loss func'.

$$w^* = \arg\min_w \sum_{i=1}^{n} 0\text{-}1\,loss(x_i, y_i, w)$$

$$0\text{-}1\,loss(z_i) = \begin{cases} 1 & \text{if } z_i < 0 \\ 0 & \text{if } z_i > 0 \end{cases}$$

One problem with 0-1 loss() fun^n is that it is not convex, and hence it is not differentiable. But to solve an optimizat. problem we need a differentiable fun^n.

So, sol^n is, to approximate the 0-1 loss() fun^n.



→ approximated 0-1 loss() it is also known as logistic loss.

· logistic loss() is approximation of 0-1 loss.
And when we use logistic_loss() as an approximation of 0-1 loss() we get LR
There is another approximation fun^n called Hinge loss(). And when we use Hinge loss as loss fun() we get SVM.
When we use exp loss we get · Ada Boost.
When we use sq-loss we get Linear Regression.

★ Hyper parameter search / optimization

How to determine the best λ.

problem with λ is, $\lambda \in R$ it have infinite possibilities.

(i) One technique to find λ is Grid search (Brute force)

- Try with various λ values in given range.

(ii) Random search :- Randomly pick value in given interval.

★ Column / feature standardisation.

In Logistic regression, it is compulsory to perform column standardisation. Because we are calculating distance from plane. We know that if two feature have different scale that may impact the distance.

★ Feature importance & model interpretability.

let see how we can calculate the importance of feature in LR.

	f_1	f_2	f_3	...	f_j	...	f_d
$w \rightarrow$	w_1	w_2	w_3	...	w_j	...	w_d

assume: all features are independent.

If assumption is true, we can get feature importance using w.

If $|w_i|$ is large, its contribution to $w^T x$ is large and hence its importance is more.

★ Collinearity (or) Multicollinearity :-

As we have already discussed that we can use feature weights to tell feature importance only when features are independent. But why this so. Lets understand it. But before, we need to understand collinearity.

If we have 2 feature f_i & f_j
s.t $f_i = \alpha f_j + \beta$, then f_i & f_j are collinear.

And collinearity across various feature is called multi collinearity

ex:- $f_1 = \alpha + \alpha_2 f_2 + \alpha_3 f_3 + \alpha_4 f_4$

Then f_1, f_2, f_3, f_4 are said to be multi-collinear.

Q: Why does $|w_i|$ not be useful as feature imp. if feature are collinear.

$D := \langle x_i, y_i \rangle_{i=1}^{n}$

$w = \langle 1, 2, 3 \rangle$; $x_q = \langle x_{q_1}, x_{q_2}, x_{q_3} \rangle$

$w^T x_q = x_{q_1} + 2x_{q_2} + 3x_{q_3}$

If $f_2 = 1.5 f_1$ (f_1 & f_2 are collinear)

$w^T x_q = x_{q_1} + 2 * 1.5 * x_{q_1} + 3x_{q_3}$

$= 4x_{q_1} + 3x_{q_3}$

$\langle 4, 0, 3 \rangle$ we get a totally new w.

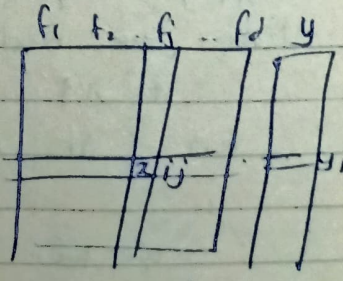
$w_1 \; w_2 \; w_3$

Hence w is changing i.e. for 2 different w we getting same $w^T x_q$ hence $|w_i|$ cannot be used to get feature imp if feature are not independent.

Now, the next question is, how do we know if feature are colinear or not.

One technique is perturbation technique

take a x_{ij} and add small noise term; i.e. perturbing the column value. Before adding noise compute w. after perturbation again train the model we calculate w', if w & w' are differ drastically then we can conclude that our features are colinear and here we cannot use w to calculate feature importance.



if we cannot use $|w_i|$ to calculate feature importance we can use forward feature selection.

★ Train & Test: space time complexity.

Till now, we haven't seen how to train LR. To train LR we need to solve ours:

$$w^* = \underset{w}{\operatorname{argmin}} \left(\text{Logistic loss} + \text{Regularization} \right)$$

To solve this we use a technique called stocastic Gradient distance (SGD)

Train Time : $O(nd)$ n = # of data point, d = dimen / feature.

→ Runtime (Testing)

Time $O(d)$ → we only need to multiply $w^T x_q$ ($1 \times d \cdot d \times 1$)

Space :- $O(d)$ we only need w i.e. weight vector

∴ We can see that run time & space complexity is low, we can use LR on Low-latency system.

∴ LR is V.V good for low-latency application

what if d is large say $d = 1000$.

$w^T x_q$:- 1000 multiplication

1000 addition.

In that case we use L1 regularization which apply sparsity (w_i corresponding to less imp feature = 0)

Hence we need to choose good λ,

If $\lambda \uparrow$, Latency ↓

But If $\lambda \uparrow$ Bias ↑ (underfit)

∴ We have tradeoffs $\lambda \uparrow$, Bias ↑, Latency ↓

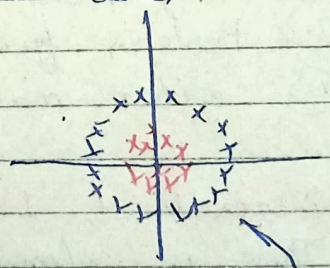
Cases:-

→ Decision surface :- Linear / hyperplane.
→ assumption :- data is linearly separable or almost nearly sep.
→ feature imp & interpretability :- $|w_i|$ if features are not collinear.
→ Imbalanced datasets :- upsampling & down sampling
→ outliers :- less impact on model ~~so~~ due to use of sigmoid.
→ Missing values :- used standard imputation technique.
→ Multi-class :- one vs Rest model.
→ Similarity matrix :- If similarity or dis tan matrix is given there is a extension of LR is there which can use similar matrix → Kernel LR.
→ Best & Worst case :- Best: (i) Data is almost linearly separable
(ii) if we have low latency requirement.

→ Large dimensionality :- if d is large LR works good, because if d is large, the chance of linear-separability increases but at same time computation time increases.

Non-linearly separable data.

What if we have non-linearly separable data.
Can we use LR on this data.
We can't directly apply LR on this but we can make some changes to apply LR



We can do feature transformation / feature engineering FE

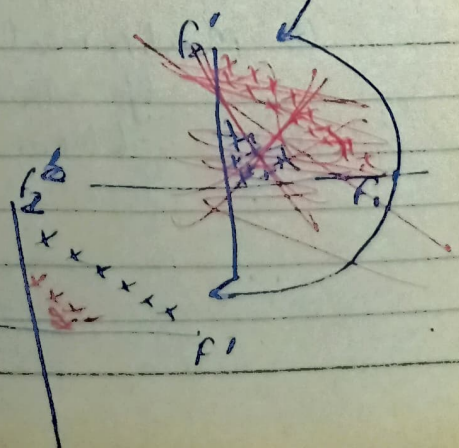
$f_1' = f_1^2$
$f_2' = f_2^2$

$x < x_1, x_2 >$
$\downarrow$
$x' < x_1', x_2' >$

Now,

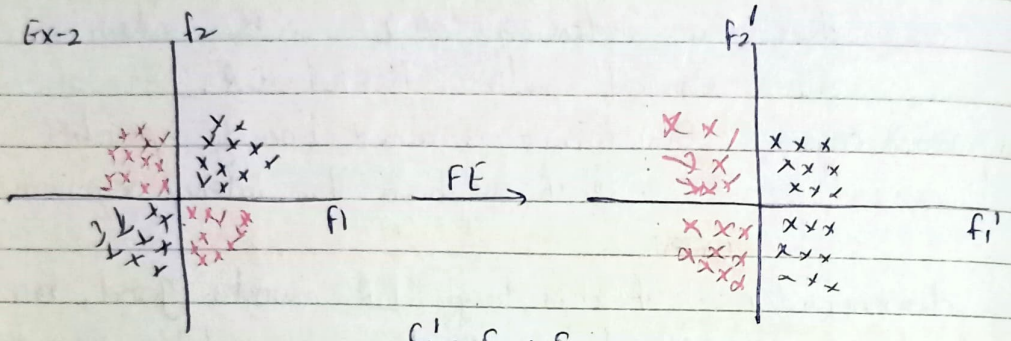
$x_1' = x_1^2$
$x_2' = x_2^2$

Now we can apply LR.



Now the question is, how do you know which transform to apply.
That we know using experience and practice.

The most important aspects of applied ML/AI
(i) Feature engineering
(ii) Bias - variance
(iii) Data analysis & Visualisation



$f_1' = f_1 * f_2$
$f_2' = f_2$

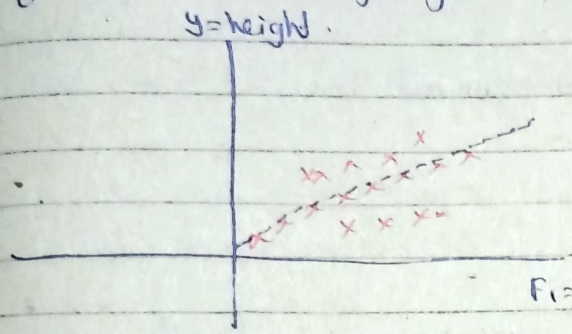
→ Typical transformation for real valued feature
(i) polynomial : $f_1 * f_2$, $\{f_1^2, f_2^2\}$, $\{f_1^3, f_2^3\}$ ...
(ii) Trigonometric : $\sin(f_i)$, $\cos(f_i)$ ...
(iii) boolean function.
(iii) other mathematical : $\log(f_i)$ $\exp(f_i)$

★ Generalized Linear models.
There is extension of LR known as Generalized Linear model.
LR :- Gaussian NB + Bernoulli class ← prob interpretable
(i) Multinomial LR :- Instead of Bernoulli distribution of class we have multinomial distribution
(ii) Linear regression :- $P(y|x) \sim N(\mu, \sigma^2)$

CH:- Linear Regression

* Linear Regression:-

Let's understand the Geometry behind Linear Regression :- suppose we want to predict height of a person given some feature like : weight, gender, ethnicity, hair color :-



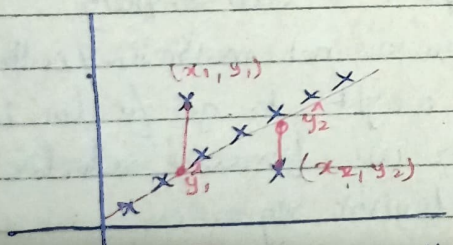
In Linear Regression we try to find a line/plane that fits the give data

height = $w_1 * weight + w_0$ // equation of line

$y = mx + c$

objective is to find w_1 & w_0.

Similarly in higher dimension instead of a line, we have plane, and we need to predict the parameter of plane.



error = $y_i - \hat{y}_i$

we need to minimize the error.

$$(w^*, w_0^*) = \underset{w, w_0}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

w^* → vector, w_0^* → scalar (intercept)

$$= \operatorname{argmin} \sum_{i=1}^{n} (y_i - (w^T x_i + x_0))^2$$

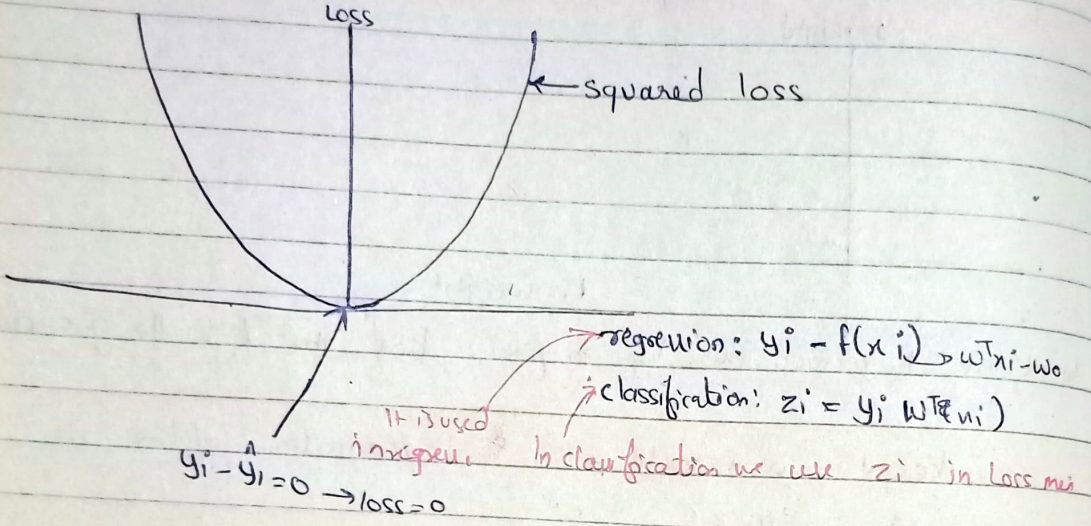
$$\boxed{(w^*, w_0^*) = \underset{w, w_0}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - (w^T x_i + x_0))^2}$$

Just like Logistic regression, here also we need to apply Regularization term

$$(w^*, w_0^*) = \underset{w, w_0}{\operatorname{argmin}} \sum_{i=1}^{n} \{ y_i - (w^T x_i + w_0) \}^2 + \lambda \|w\|_2^2$$

This can also be L_1 regularization term.

This is the geometric interpretation.
Now look at loss-minimization derivation.



★ Cases :- Most of the Linear regression and logistic regression are same except for some.

1. Imbalanced data :- upsampling & down sampling
2. Feature Imp & interpr :- If feature are not multicollin/collin use feature weight to get feature importance otherwise use forward selection
3. Feature Eng / ↪ same as logistic regression.
4. Regularization :- same " " "
5. Outliers :- In logistic reg we used sigmoid fun", in linear reg. If we have outlier then error will be very big for that point ∴ hence Linear Regression is very prone to outliers :-

So, follow these steps.

① . D Train → w^*, w_0
- find points very far away from π
- remove these points as outliers
- again train model with rest of points
- iterate in same way.