Machine Learning

by ~~freecodecamp~~.org

What is machine learning?

Subdomain of computer science that focuses on algorithms which help a computer learn from data with explicit programming.

without the programmer being there telling the computer exactly what to do.

Difference between AI, ML & Data Science

AI	ML	DS
Area of computer science, where the goal is to enable computers and machines to perform human-like tasks and simulate human behaviours.	Subset of AI that tries to solve a specific problem and make predictions using data.	A field that attempt to find patterns and draw insights from data (might use ML)

All fields overlap! All may use ML!

Types of machine learning

Supervised learning uses labeled inputs (meaning the input has a corresponding output label) to train models and learn outputs. For example, pictures of animals. For

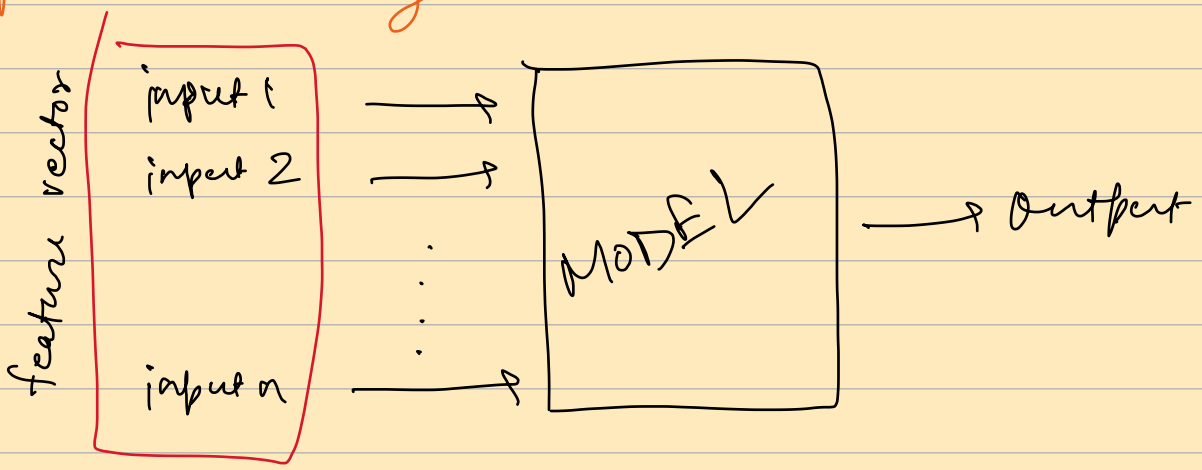
Unit 1 learn about

Computer those are just pixels with colors but in supervised learning these inputs have a label associated with it like cat, dog and this is the output that we might want the computer to be able to predict.

Unsupervised learning uses unlabled data to learn about patterns in data. For example, pictures of animals without label and computer is not able to tell cat, dog but able to cluster similar images into one.

Reinforcement learning agent learning in interactive environment based on rewards and penalties.

Supervised learning



Types of features:-

Qualitative categorical data (finite number of categories and groups). For example: gender, location, nation. No inherent order, like can't say male 1 or femal 2. That's why it is called nominal data. The way we want to feed the data to the computer is by using one-hot encoding.

One-hot encoding is a way to turn words or categories into numbers so computers can use them. It works like

for each possible category, you make a column with a "yes" or "no" value (1 or 0). For example, if you have animals: cat, dog and bird, you create three columns - one for each animal.

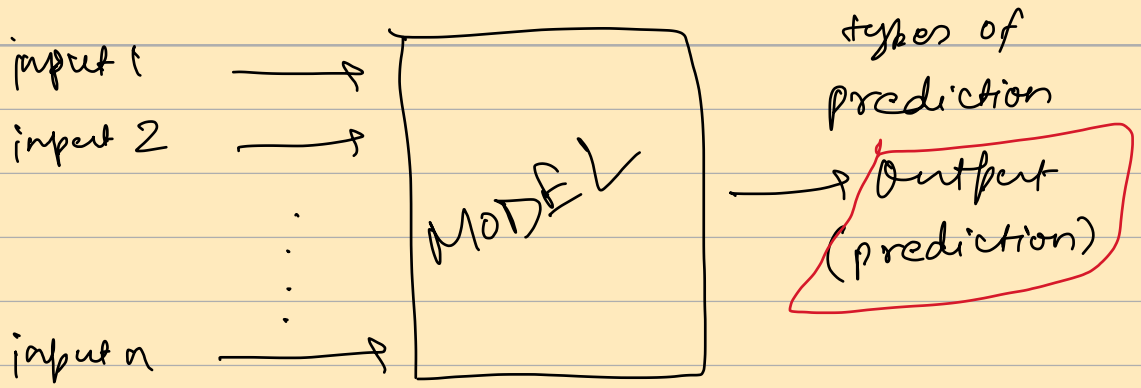
- If your data is "cat" you write 1 in the "cat" column and 0 in others.
- If it's dog you put 1 in the "dog" column and 0 for others

cat	dog	bird
1	0	0
0	1	0
0	0	1

This lets computers understand the categories without thinking one is bigger or smaller than the others.

Inherent order also called Ordinal Data because they have some sort of inherent order. like a toddler is a lot closer to being a baby than to an adult. Or good is closer to great than it is to bad. For these type of data set we can mark with numbers from 1 to 5.

Quantitative numerical valued data (could be discrete or continuous). For example length, temperature or how many eggs in the basket.

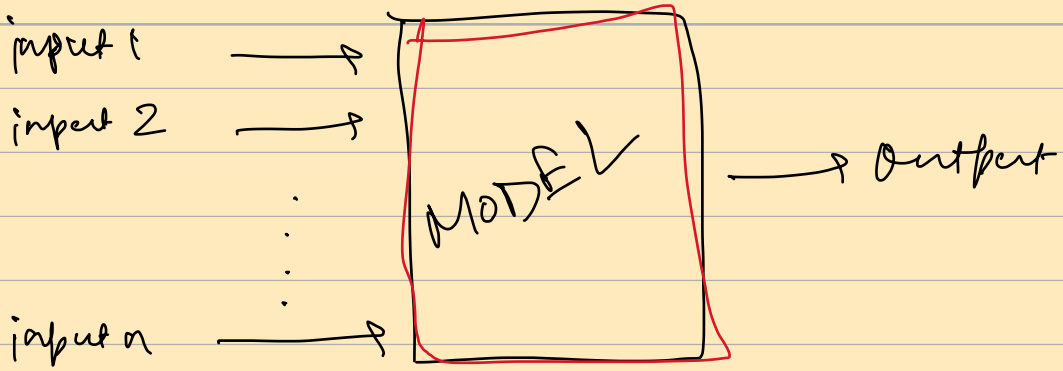


In supervised learning there are some different tasks

Classification predict discrete classes, for example hot dog, pizza, ice cream. This is something known as multi class classification. There is also binary classification like you might have hot dog or not hot dog.

Binary Classification	Multiclass Classification
Positive / Negative	Cat / dog / dolphin
Cat / dog	Orange / apple / pear
Spam / Not spam	Plant species

Regression predict continuous values. Instead of categories we come up with numbers. For example price of a stock tomorrow, the temperature or what is the price of the house.



How do we make the model learn? How can we tell wheather or not it's learning?
Let's take a data Quantitive feature because all of them are on a scale.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Age	Outcome
6	148	72	35	0	33.6	50	1
1	85	66	29	0	26.6	31	0
8	183	64	0	0	23.3	32	1
1	89	66	23	94	28.1	21	0
0	137	40	35	168	43.1	33	1
5	116	74	0	0	25.6	30	0
3	78	50	32	88	31.0	26	1
10	115	0	0	0	35.3	29	0
2	197	70	45	543	30.5	53	1
8	125	96	0	0	0.0	54	1
4	110	92	0	0	37.6	30	0
10	168	74	0	0	38.0	34	1

each row = different sample in the data

This is what we would call a feature vector

Target of that feature vector

Each column = different feature

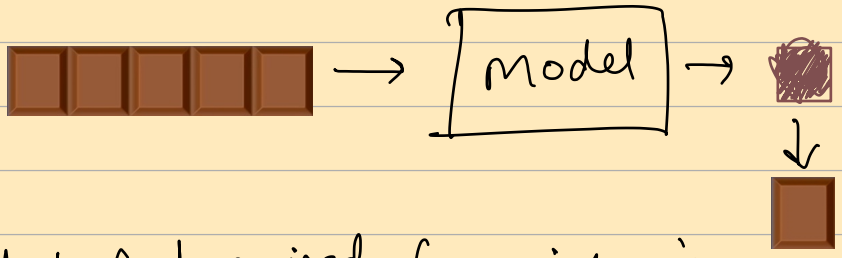
Expect this one, it's the output label

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Age	Outcome
6	148	72	35	0	33.6	50	1
1	85	66	29	0	26.6	31	0
8	183	64	0	0	23.3	32	1
1	89	66	23	94	28.1	21	0
0	137	40	35	168	43.1	33	1
5	116	74	0	0	25.6	30	0
3	78	50	32	88	31.0	26	1
10	115	0	0	0	35.3	29	0
2	197	70	45	543	30.5	53	1
8	125	96	0	0	0.0	54	1
4	110	92	0	0	37.6	30	0
10	168	74	0	0	38.0	34	1

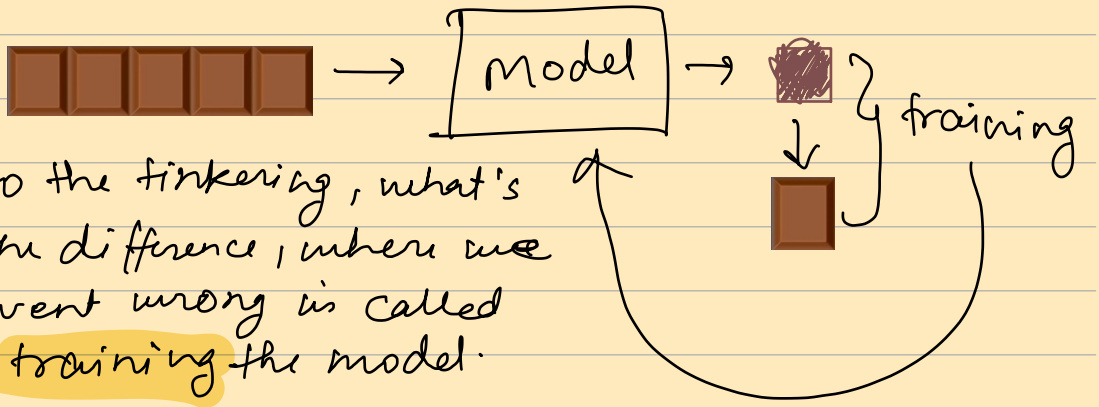
Features matrix, X

Labels, targets vector, Y

Each row of this will be fed into our model. And our model will make some sort of prediction. We compare that prediction to the actual value of y that we have in our label data set.



This is what supervised learning is, we compare that prediction to the truth and we can go back and we can adjust some things. So that the next iteration we get closer to what the true value is.

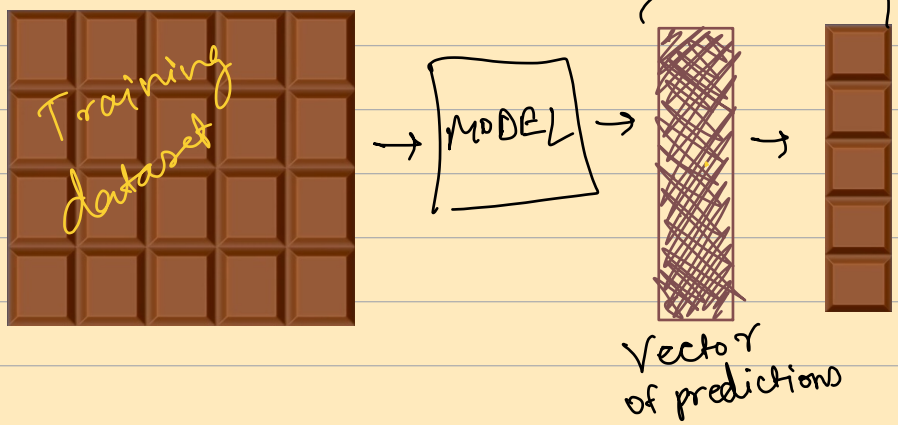


So the tinkering, what's the difference, when we went wrong is called training the model.

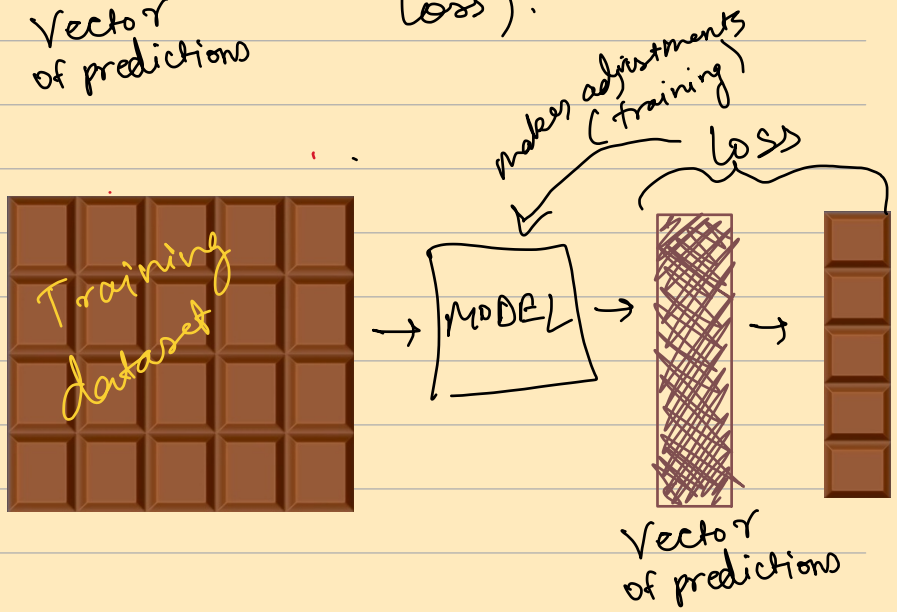
Not required to put the whole data into once into the model, because how we will know how model will work with new data.

How do we get our model to access that? so we actually break up our whole data set that we have into three different types of data set

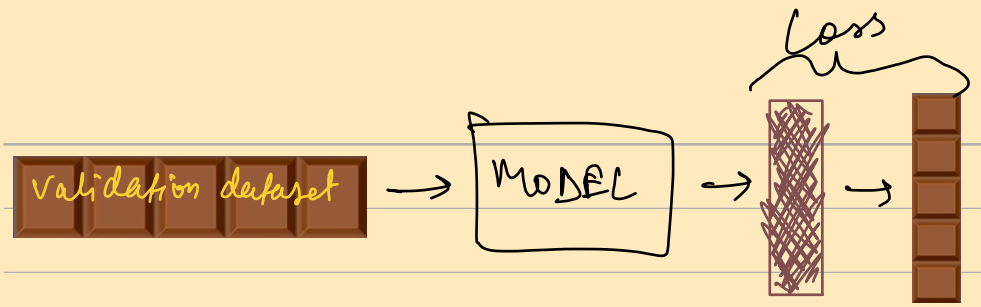
We feed our training dataset into our model and from the vector of predictions we figure out what's the difference between our prediction and true values (known as loss).



And then we make bunch of adjustments and that's what we call training the model.

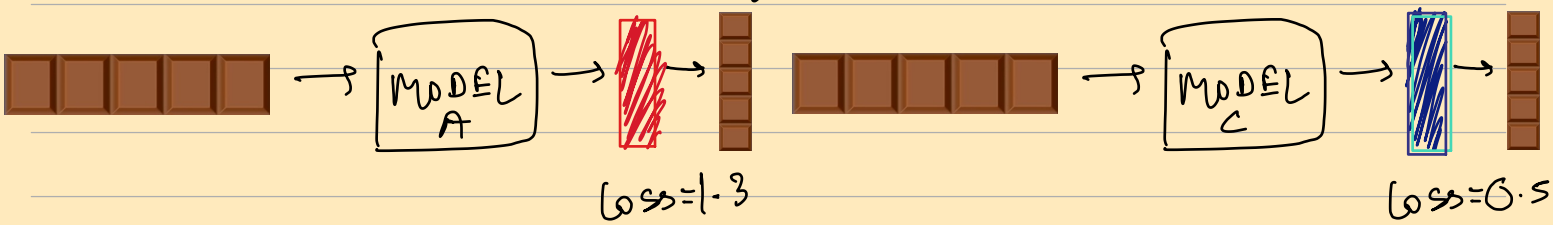


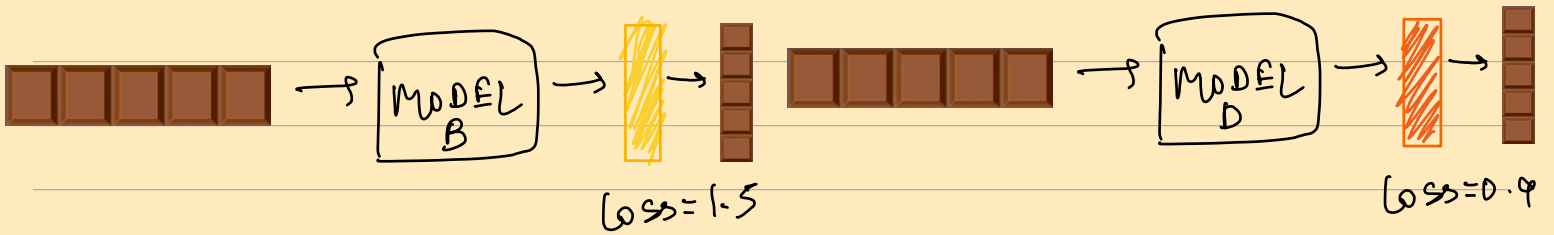
Once we make bunch of adjustments we can put our validation set through this model. And the validation set is kind of used as a reality check during or training to ensure the model can handle unseen data still

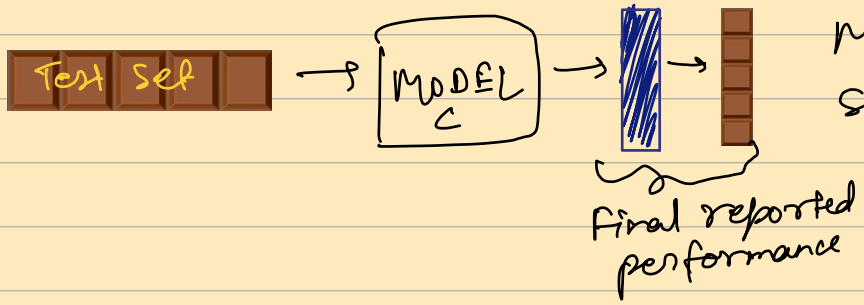


we don't have that training step. The loss with validation dataset is never gets fed back into the model.

In the below, out of the 4 model types Model C has the least loss

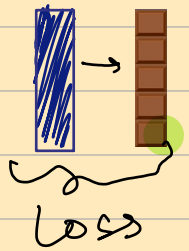






Model C performs the best since it has the least loss.

Test set is used as a final check to see how generalizable that chosen model is. And this data set is the data that the model has not seen at any time during the training process.



Loss is the difference between your prediction and the actual label.

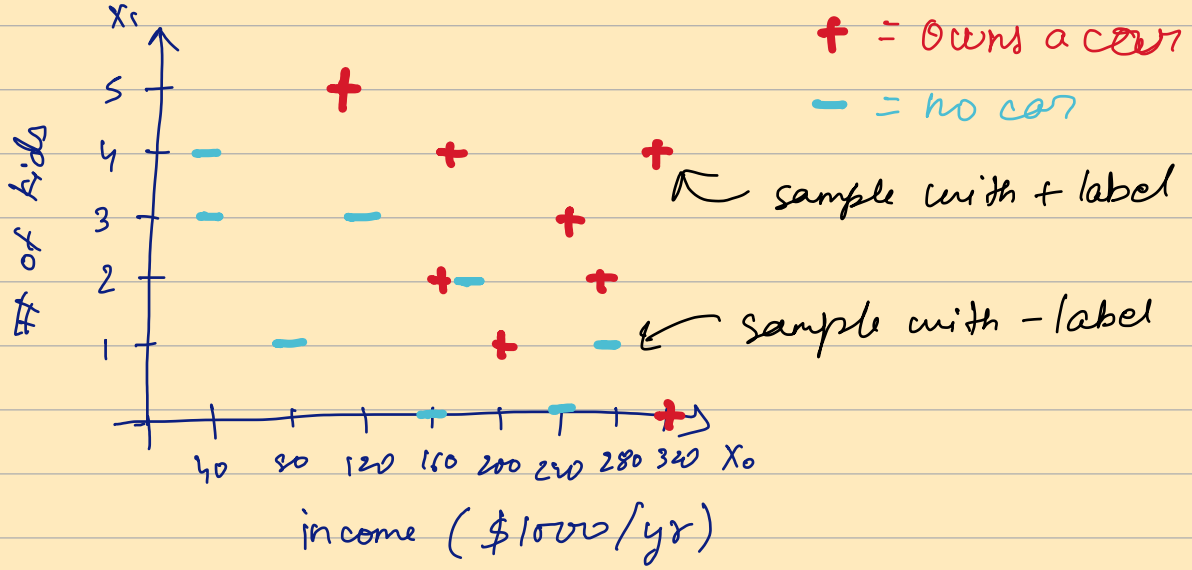
Loss decreases as the performance gets better. Loss is a measure of performance, just like Accuracy

Predictions	Actual
Apple	Apple
Orange	Orange
Apple	Orange
Apple	Apple

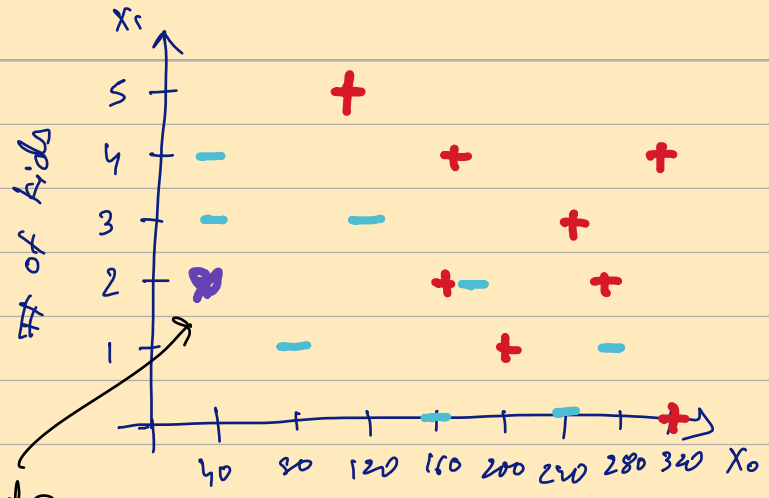
Accuracy of the model is 3/4 or 75%!

Different types of models

K-nearest neighbours



The above data shows if in a labeled inputs with + and -
But what if we add a new point

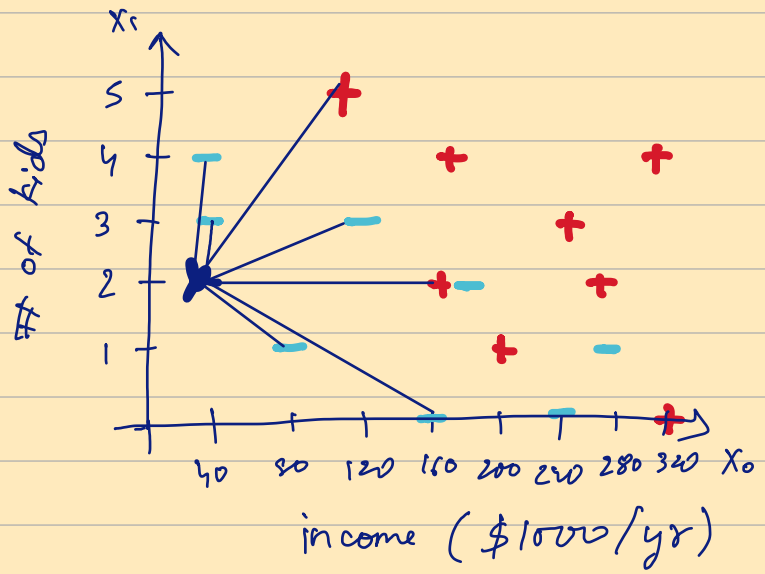


what do we think this would be?

Logically it seems they wouldn't have a car because that matches the pattern of everybody else around them.

That's the concept of this nearest neighbour is that you look what's around you. And then take the label of the majority that's around.

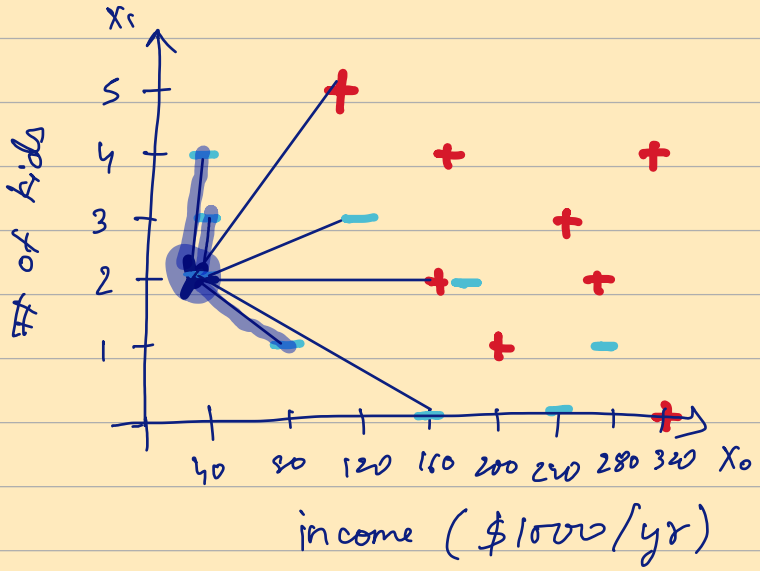
First we define the distance function → and in 2-d plots like above we have distance function known as Euclidean distance. It is a straight line distance between the other plots like below



formula:-

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

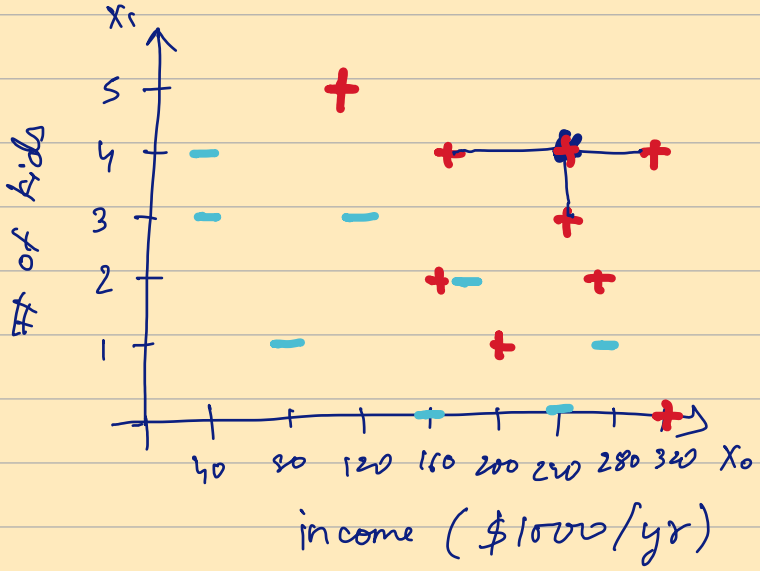
The "K" in the K-nearest neighbour algorithm, tells us how many neighbours do we use in order to judge what the label is. Usually K is 3, or 5 depending how large the dataset is.



For this one we will use K as 3

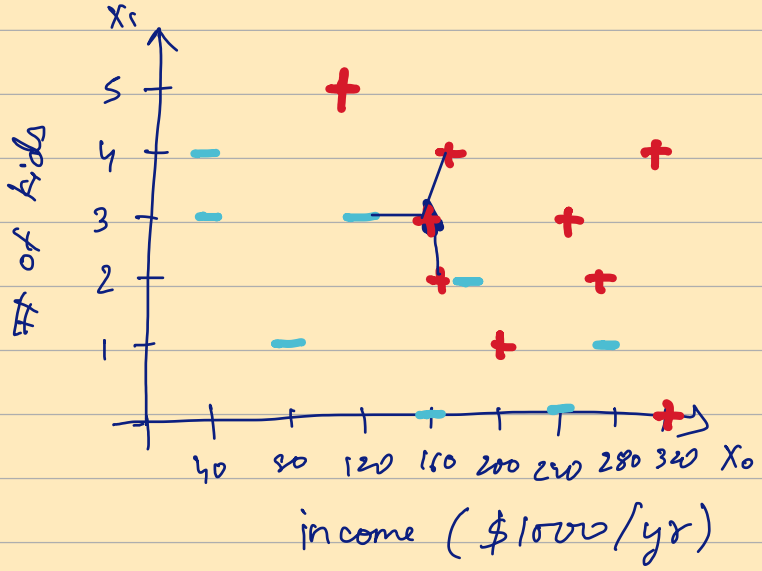
Since most of the nearest neighbor for is - so the prediction for this point is going to be -

Another example



for this it will be +

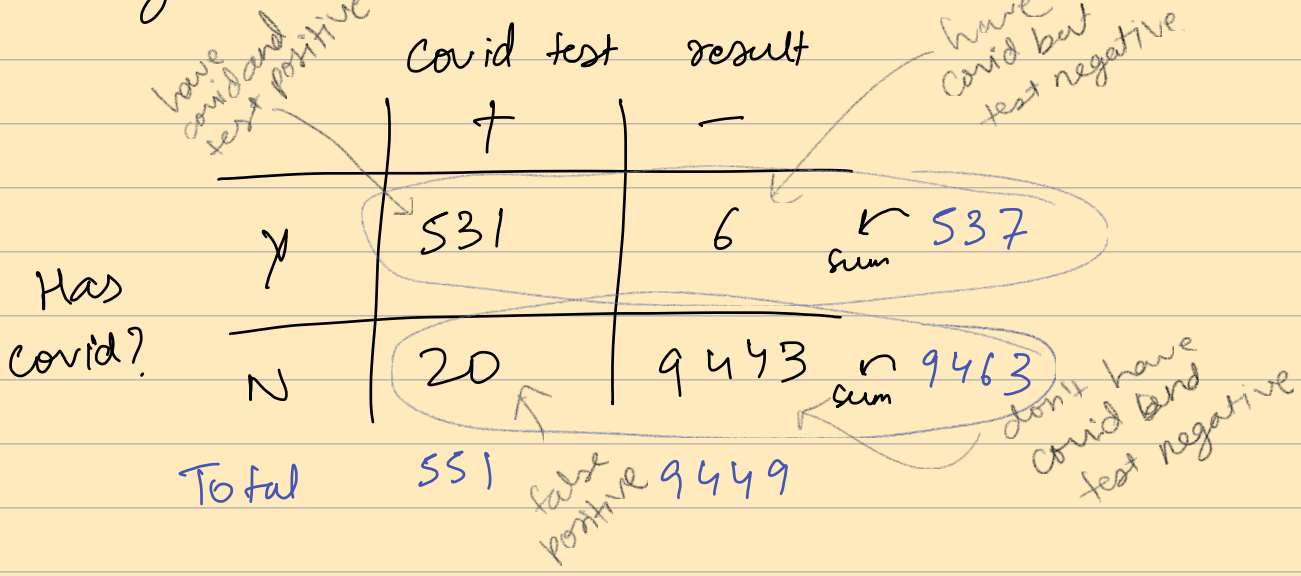
One more



Since the closest 3 are 2 + and 1 − so the prediction for the point will be + based on the majority.

Naive Bayes

For this need to understand conditional probability and bayes rule.



what is probability of having covid given a positive test?

$$P(\text{Covid} \mid (+)\text{test}) = 531/551 = 96.4\%$$

(| this line means given that)

Bayes Rule :-

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

↑ condition

What is the probability of some event A happening given that B happened

For example :-

$P(\text{false positive}) = 0.05$ $\quad P(+ \mid \text{no disease})$

$P(\text{false negative}) = 0.01$ $\quad P(- \mid \text{disease})$

$P(\text{disease}) = 0.1$

	+	−
disease	0.99	0.01
no disease	0.05	0.95

$$P(\text{disease} \mid \oplus \text{test}) = \frac{P(+ \mid \text{disease}) \cdot P(\text{disease})}{P(+)}$$

$P(+) \rightarrow P(+ \mid \text{disease}) \times P(\text{disease}) + P(+ \mid \text{no disease}) \times P(\text{no disease})$

$$= \frac{0.99 \times 0.1}{0.99 \times 0.1 + 0.05 \times 0.9}$$

$$= 0.6875 \text{ or } 68.75\%$$

We can expand the rule and apply to classification

$$P(C_k \mid x) = \frac{P(x \mid C_k) \times P(C_k)}{P(x)}$$

C_k ← category; $P(x \mid C_k)$: likelihood; $P(C_k)$: prior; $P(C_k \mid x)$: posterior; x ← feature vector; $P(x)$: evidence

what is the probability it's actualy from this class, given all the evidence that we see that we see the x's.

posterior what is the probability of some class C_k. so by C_k it means different categories of class like C_1 will be cat, C_2 can be dog call the way to C_k

likelihood given that assume that this class is C_k, assume that this is a category. what is the likelihood of seeing x, all these different features from that category.

prior in the entire population of things, what are the probabilities of this class in general. Like if I have in my entire dataset what is the chance that this image is a cat? How many cats do I have?

evidence just the chance of seeing the evidence or data, no matter what the cause or class is.

This is the rule of Naive Bayes -

$$P(C_k \mid x_1, x_2, \cdots, x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

This says what is the probability that we are in some class C_k given that this as in x_1 is my first input, x_2 is my second input at the last x_n is my n^{th} input. For example, let's say our classification is, do we play soccer today or not? So our x's are is it windy? How much rain is there? and what day of the week is it? So let's say it's not windy, it's raining but it's wednesday. Do we play soccer or not?

$$P(C_k \mid x_1, x_2 \cdots x_n) = \frac{P(x_1, x_2, \cdots x_n \mid C_k) \times P(C_k)}{P(x_1, x_2, \cdots, x_n)}$$

$\propto$ sign means proportional to

$$\frac{P(x_1, x_2, \cdots x_n \mid C_k) \times P(C_k)}{P(x_1, x_2, \cdots, x_n)}$$

the denominator over here has no impact on the class so this is going to be constant. for all of the different classes.

So we can write the above as

$$P(C_k \mid x_1, x_2, \dots, x_n) \propto P(x_1, x_2, \dots x_n \mid C_k) \times P(C_k)$$

because the denominator will be same for every single class.

In naive bayes, the point of being naive, is that we're actualy we're assuming that all of these different things are all independent.

So in the soccer example, the probability that we are playing soccer or the probibility that it's windy and it's rainy, it's wednesday, all these things are independent. We're assuming that they're independent.

$$P(C_k \mid x_1, x_2, \dots, x_n) \propto P(x_1, x_2, \dots x_n \mid C_k) \times P(C_k)$$

This part of the equation can be written as

$$P(x_1 \mid C_k) \times P(x_2 \mid C_k) \times \dots \times P(x_n \mid C_k)$$

So this means

$$P(C_k \mid x_1, x_2, \dots, x_n) \propto P(x_1 \mid C_k) \times P(x_2 \mid C_k) \times \dots \times P(x_n \mid C_k) \times P(C_k)$$

or

$$\propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

this means it's a huge multiplication. means multiply everything to the right of it.

So the final deduction means is a probability that you know, We're in some category, given that we have all these features is proportional to the probability of that class in general times the probability of each of those features

given that we're in this one class that we are testing. So the probability of us playing soccer today given that it's rainy, not windy and it's wednesday is proportional to what is the probability that we play soccer times the probability that it's rainy given that we're playing soccer times the probability that it's not windy, given that we're playing soccer and what is the probability that it's wednesday, given that we're playing soccer.

So how do we use the above to make up the classification.

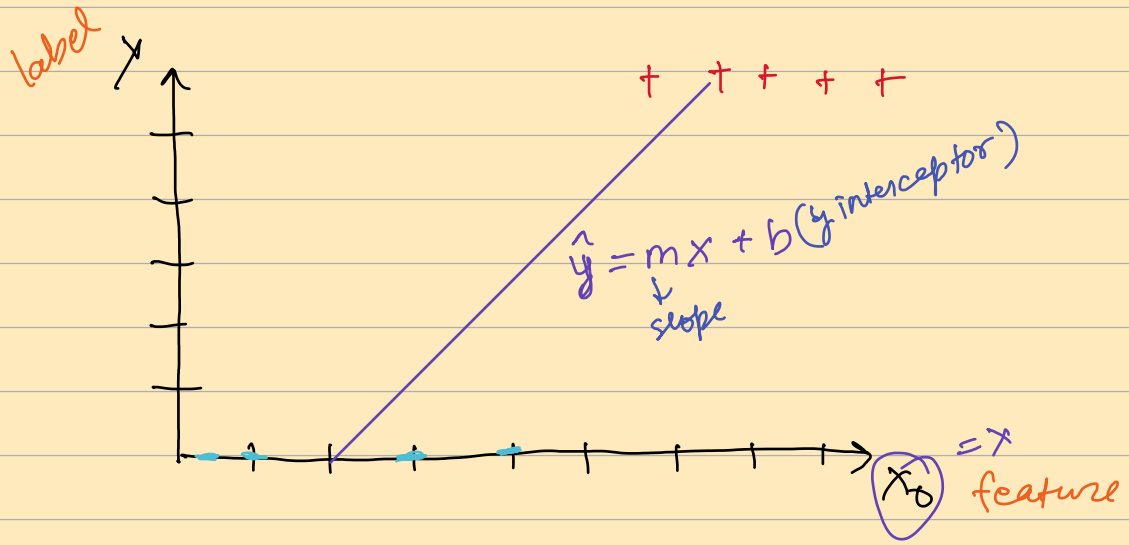
$$\hat{y} \text{ (predicted } y) = \underset{k \in \{1, K\}}{\operatorname{argmax}} \; p(C_k \mid x_1, x_2, \ldots, x_n)$$

We're going to take the k that maximizes this expression. That's what argmax means. We're going to go through each k and solve the expression and find the k that makes that the largest.

$$= \underset{k \in \{1, K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

The principal of going through each of these and finding whatever class whatever category maximizes this expression on the right is known as MAP or maximum A Posteriori. So pick the k that is the most probable so that we minimize the probability of misclassification.

Logistic Regression



How can we model probability?

$$p = mx + b$$

$mx + b$ can range from negative infinity to infinity for any value of x from negative infinity to infinity But one of the rules of probability is that it has to stay between zero and one.

So instead of setting the probability, we can set the odds equal to this

$$\frac{p}{1-p} = mx + b$$

So now this becomes a ratio. Now ratio is allowed to take on infinite values.
Still $mx + b$ can still be negative like if have a negative slope or negative b or negative x, in there To fix that we can have log of the odds

$$\ln\left(\frac{p}{1-p}\right) = mx + b$$

Now this is out of range of negative infinity to infinity

Now we need to solve for the p the probability

We can take this by taking e to the both sides

$$e^{\ln\left(\frac{p}{1-p}\right)} = e^{(mx+b)}$$

$$\frac{p}{1-p} = e^{(mx+b)}$$

$$p = e^{(mx+b)}(1-p)$$

$$p = e^{(mx+b)} - pe^{mx+b}$$

$$p\left(1+e^{mx+b}\right) = e^{(mx+b)}$$

$$p = \frac{e^{(mx+b)}}{1+e^{mx+b}}$$

$$p = \frac{e^{(mx+b)}}{1+e^{mx+b}} \times \frac{e^{-(mx+b)}}{e^{-(mx+b)}}$$

$$p = \frac{1}{1+e^{-(mx+b)}}$$

The above is a form of a special function

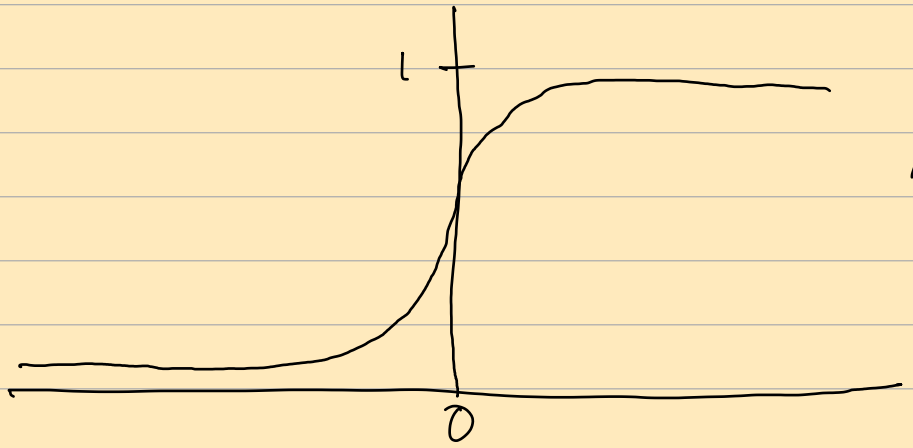
which is known as sigmoid function.

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

So the above can be written as

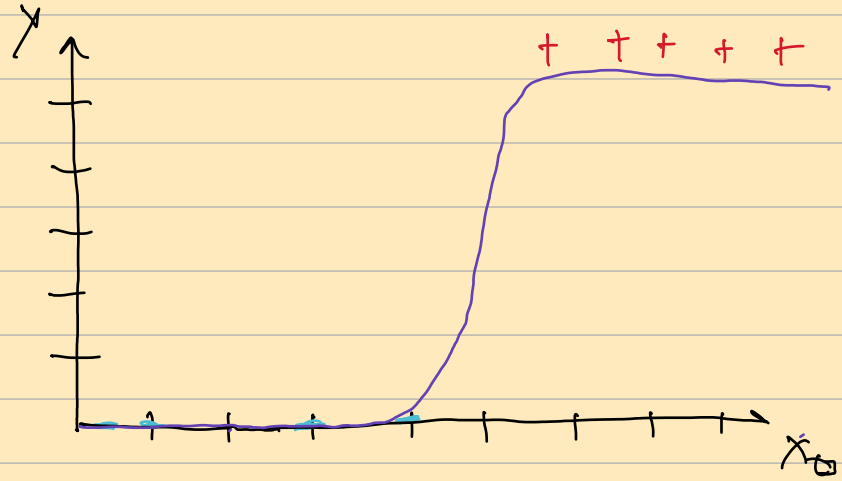
$$p = \frac{1}{1+e^{-(mx+b)}} = S(mx+b)$$

The sigmoid function usually looks like



goes in between 0 and 1

So if we look at the graph at the top for the logistic regression



So that is what logistic regression is, we are trying to set our data to sigmoid function.

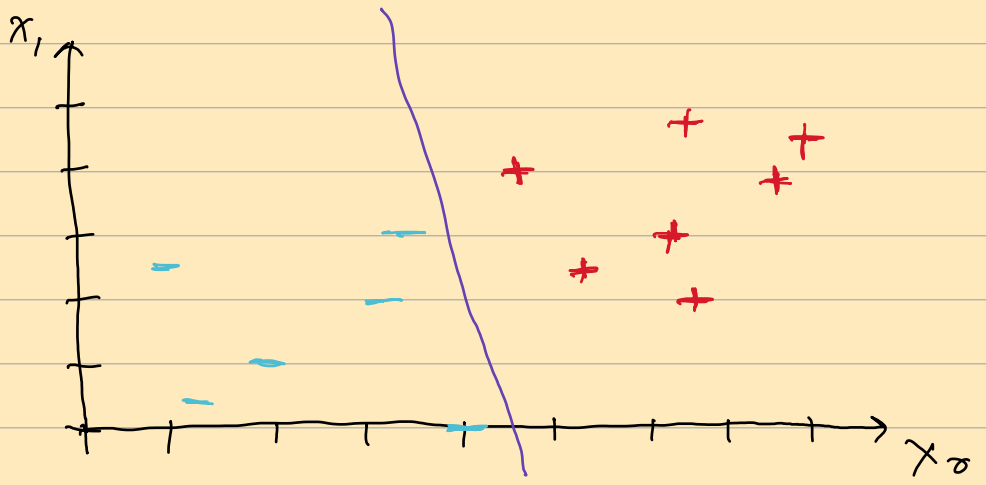
When we have only one data point, one feature x, that's what we call simple logistic regression.

x_0 = simple logistic regression

But if we have

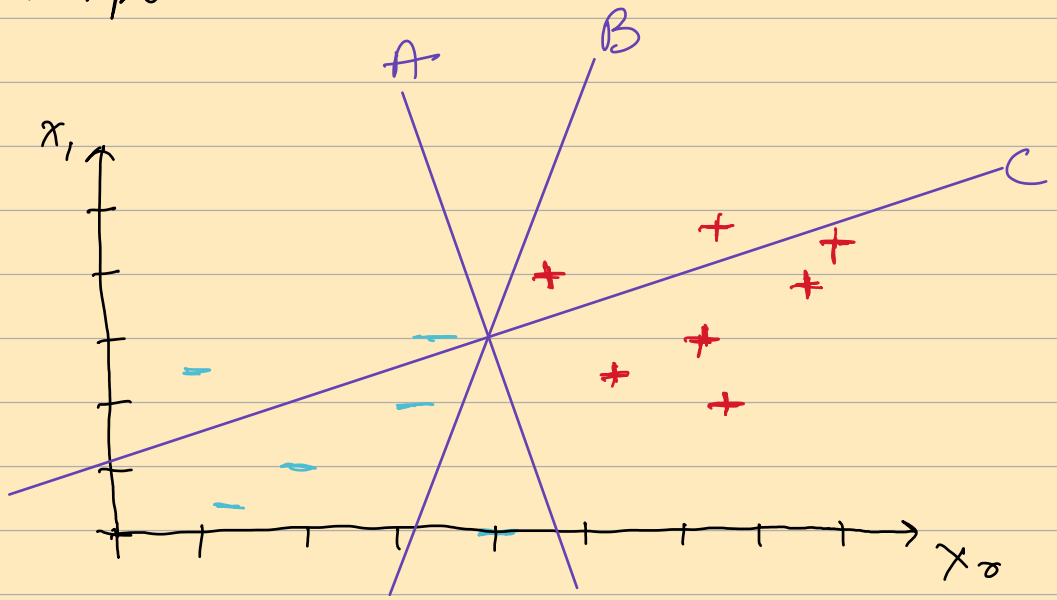
$x_0, x_1, \cdots, x_n$ = multiple logistic regression.

Support vector machines (SVM)

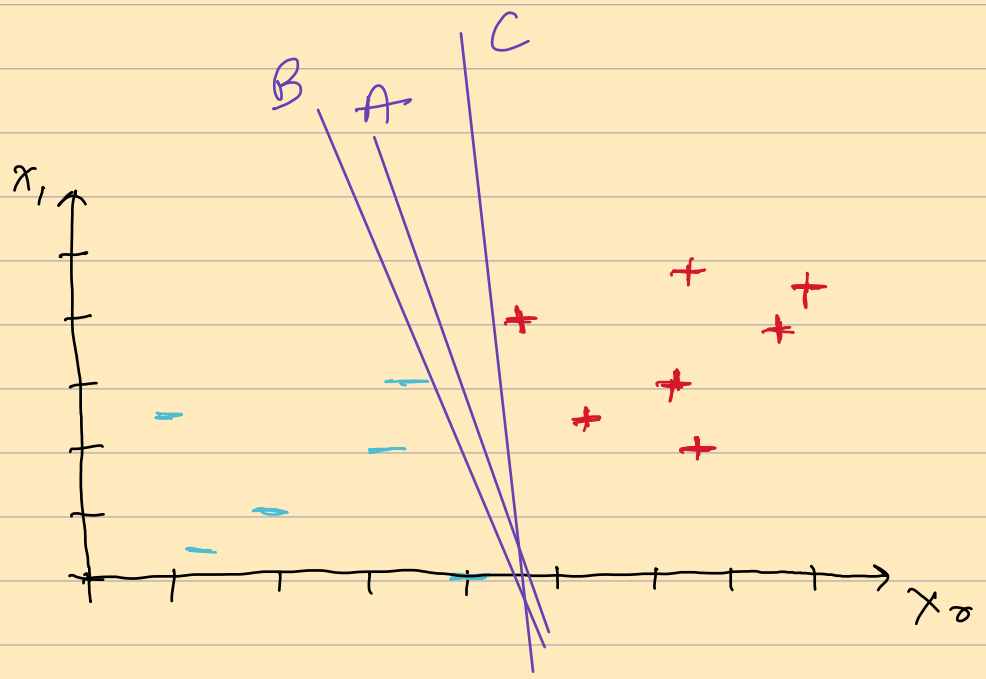


find a line between the two labels that divides the data in best way. The line will be the SVM model.

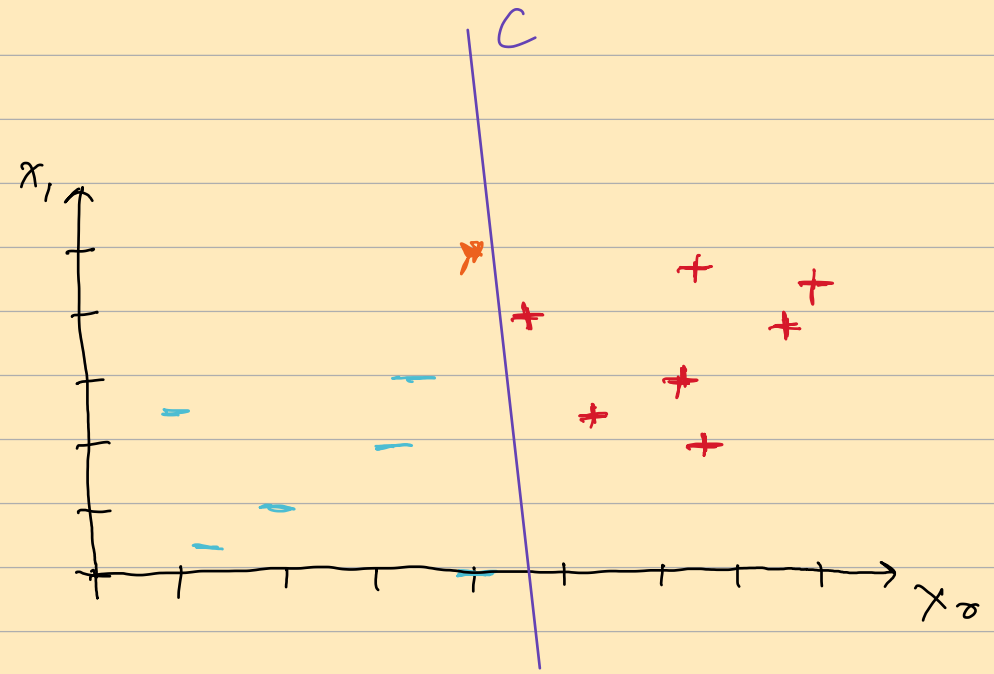
Few examples:-



Between A, B and C which one is the best divider? which defines the best boundries between the 2 data group - and + ? → The answer should be A. But what if

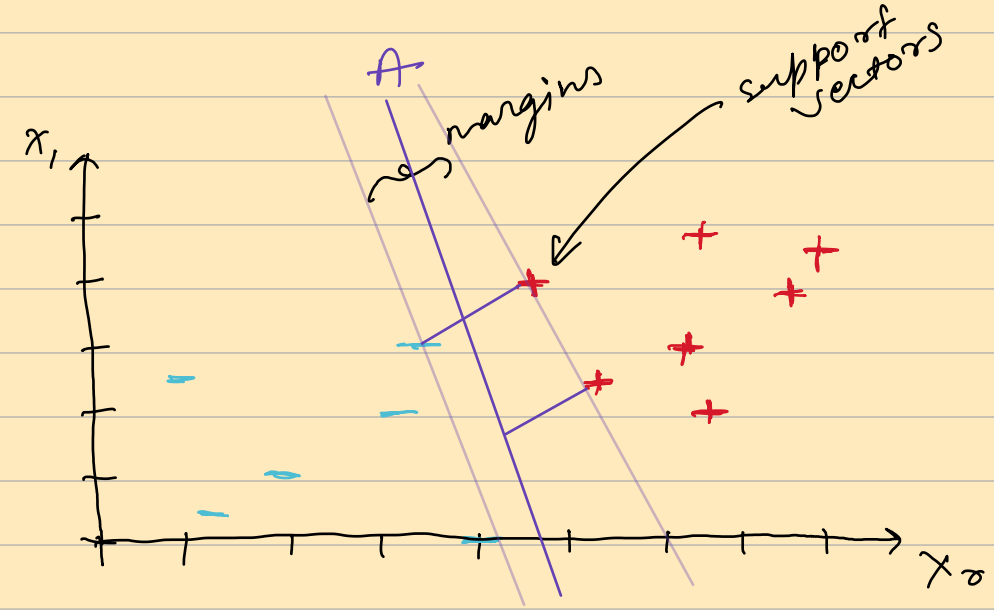


Now which one? Can still conside A because for B and C the labels are very near to them. Because let's say we have a new point and the line which we choose is C



Seems like logically the ✖ will be +.

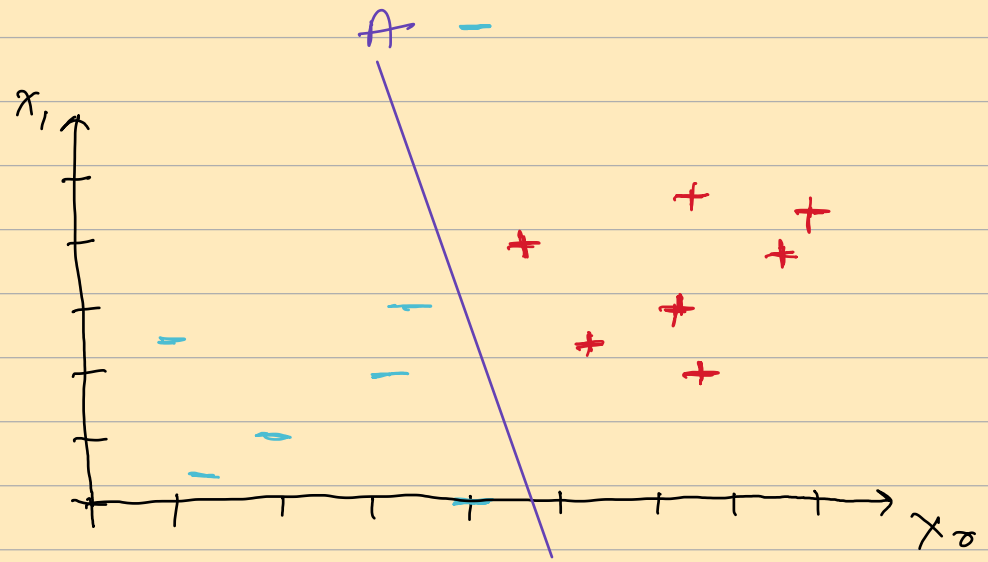
So one thing that we care about in SVM is margin. Not only we want to seperate the 2 classes really well, we also care about the boundary in between where the points in those classes in our dataset are and the line we are drawing.



So the light shade lines are known as the margins.

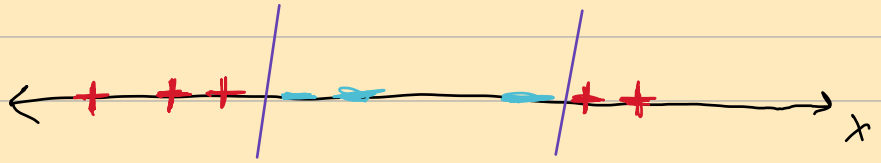
Our goal is to maximize our margins. And the data points which helps in finding the margins are called support vectors. Hence the name SVM.

Issues with SVM is that they are not so robust to outliners.

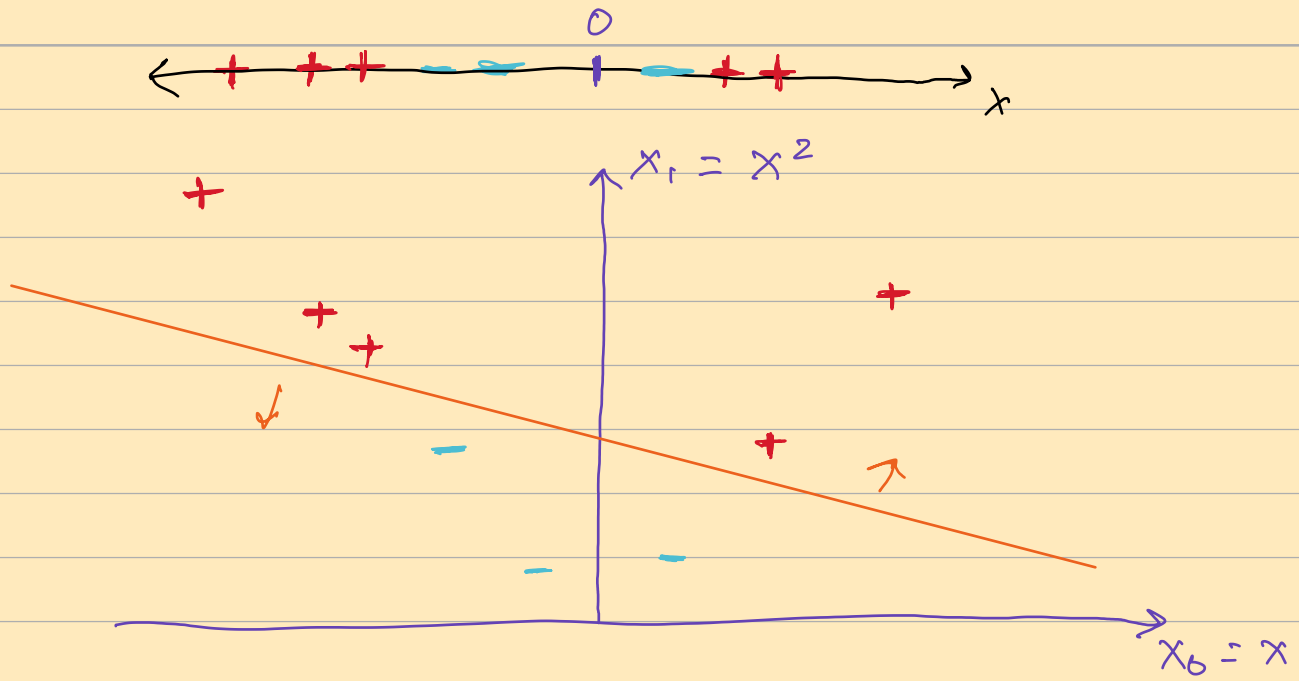


The above outlier would totally change where the support vector to be, even though that would be the only outlier.

Another example :-

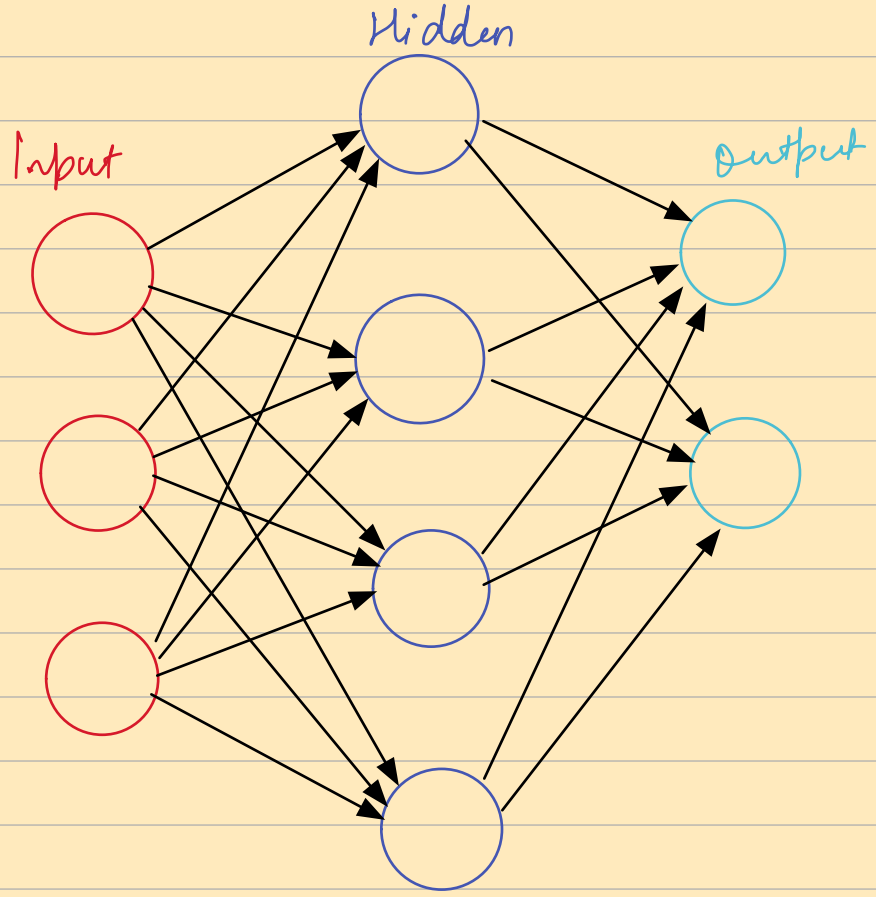


For this can have 2 SVM but that's not how SVM works. But one thing that we can do is we can create some sort of projection.



The x^2 and x transformation which we did above is knowns as kernel trick. $x \rightarrow (x, x_2)$

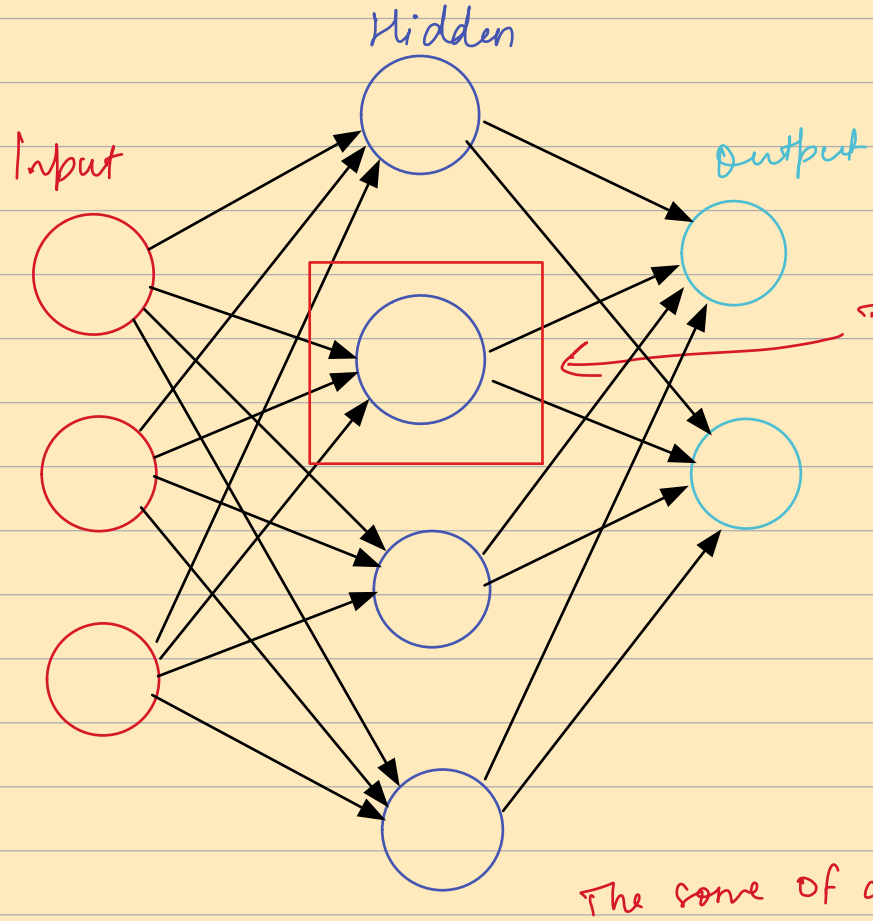
Neural net or Neural Network



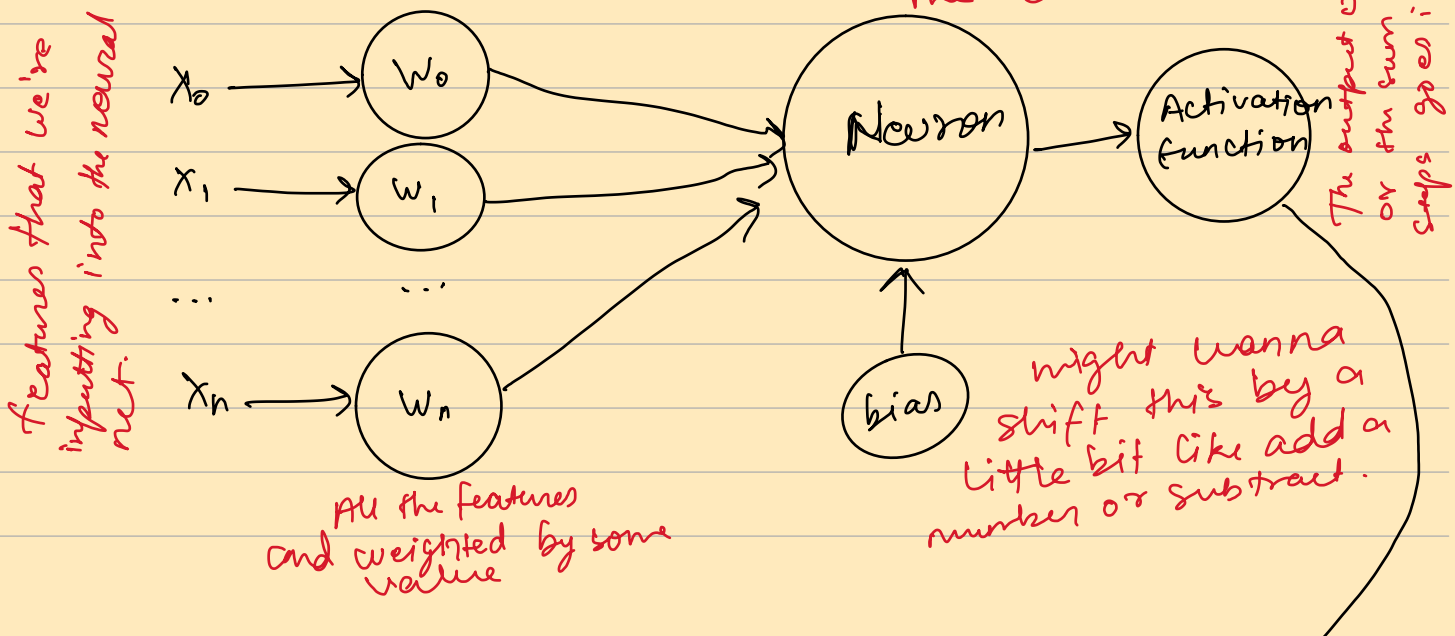
The neural nets look something like above. You

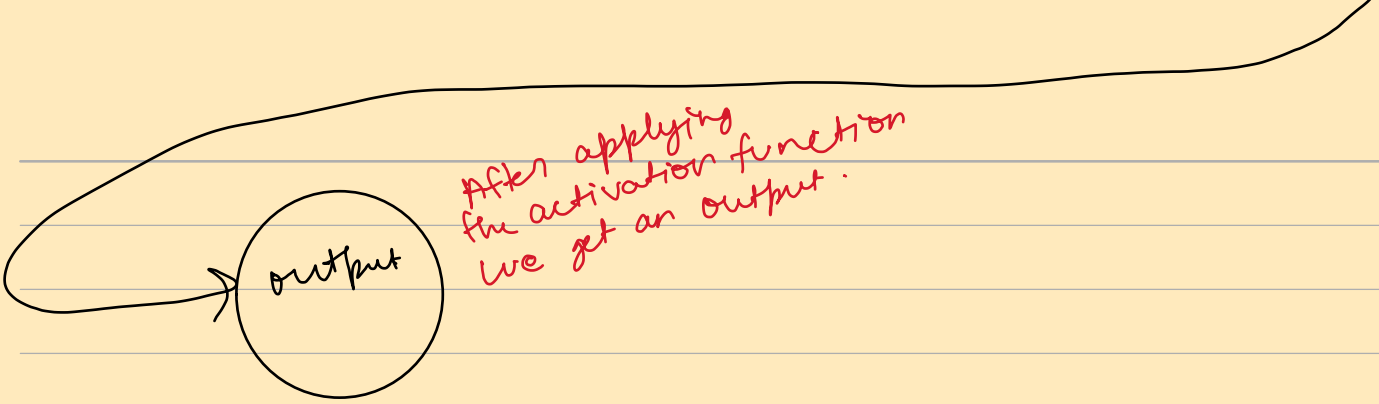
have an input layer, this is where all your features would go. And they have some arrows pointing to some hidden layers. And from that some arrows point to output layer.

Each of these layers is something known as neuron.



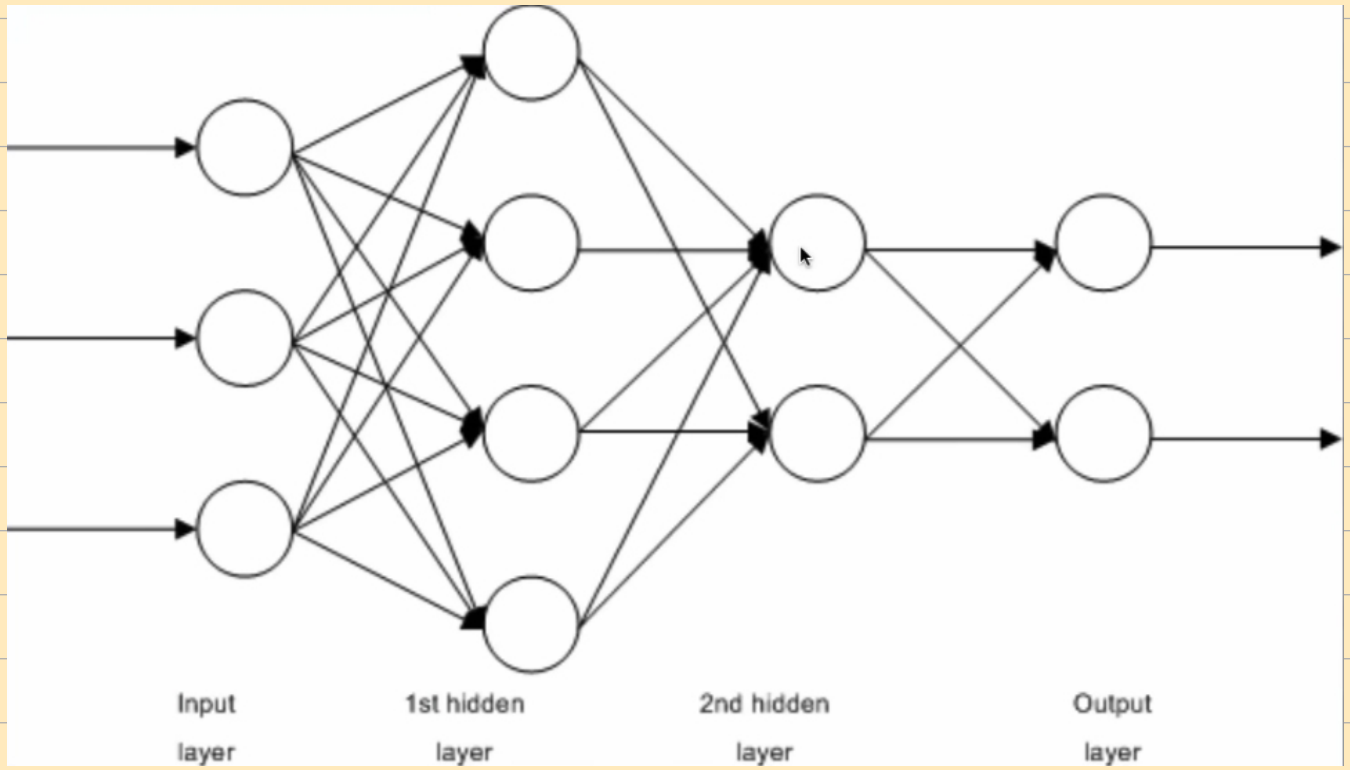
In a neural net,





after applying the activation function we get an output.

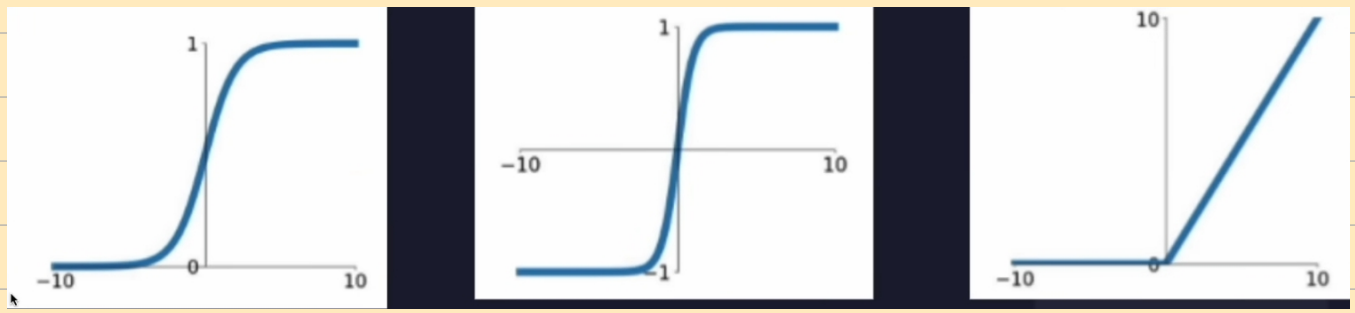
what is an activation function?



so as we discussed above, for each layer we keep on adding and move forward. So with that logic this would become a linear model which we can create by just a formula.

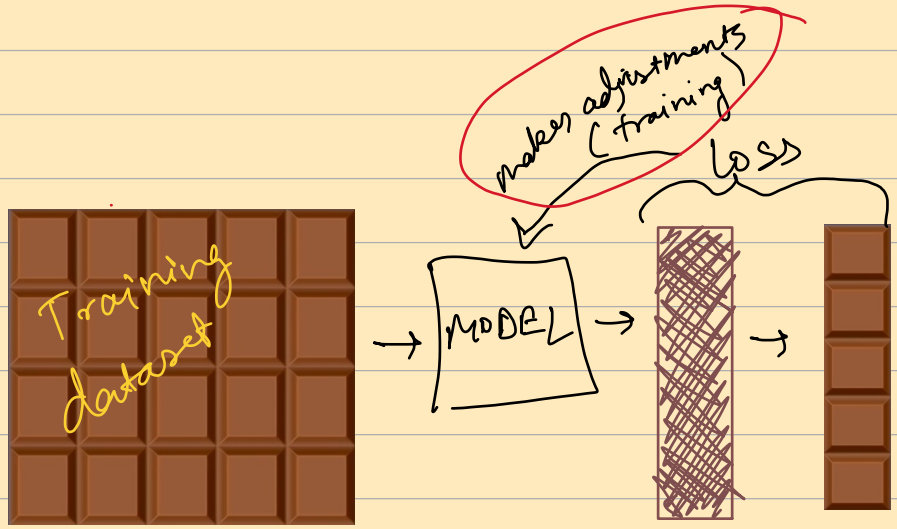
so that is where the activation function comes into picture.

The activation function might look something like below

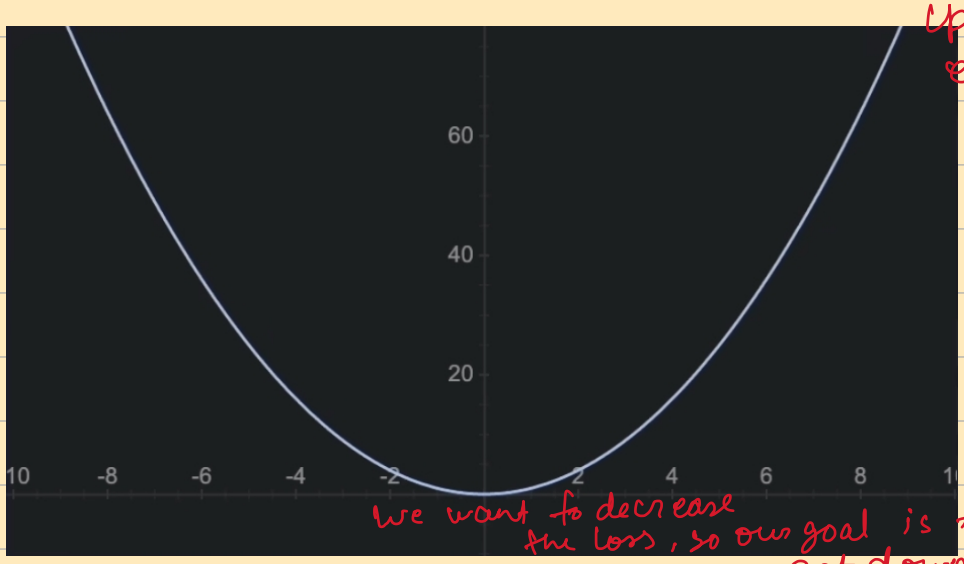


Sigmoid Tanh RELU

And as if can be seen it is not linar. Because of this the output of a neuron is no conger linear combination of the before layer. It's some sort of altered linear state which means that the input into the next neuron is not linear because of the introduction of non linearities.

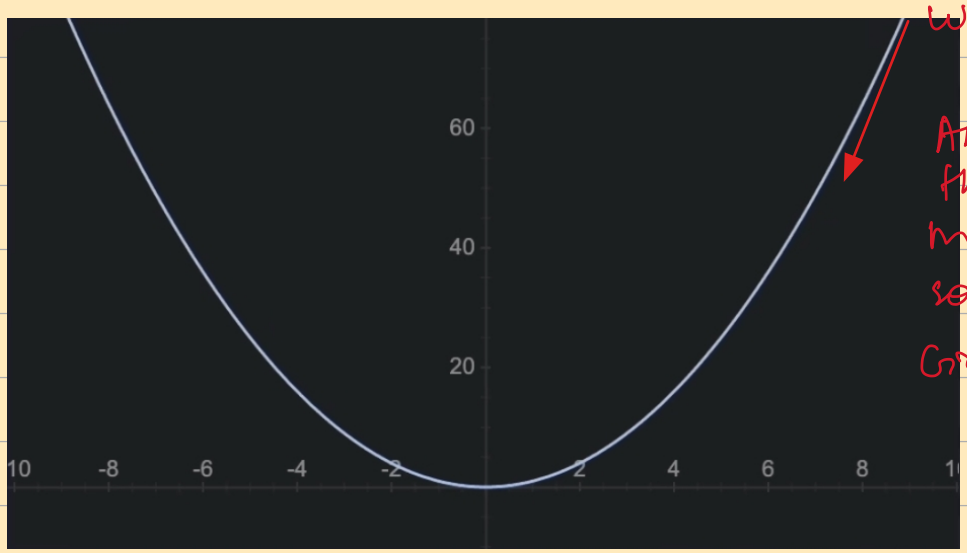


what goes on during the training step?



L2 loss function
(Quadratic formula)

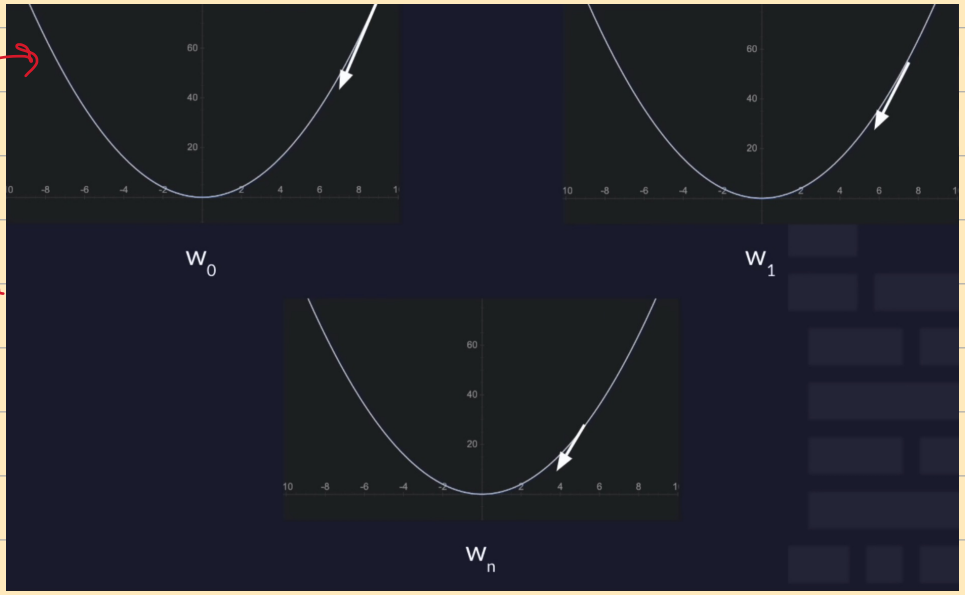
So from the above diagram we can deduce that



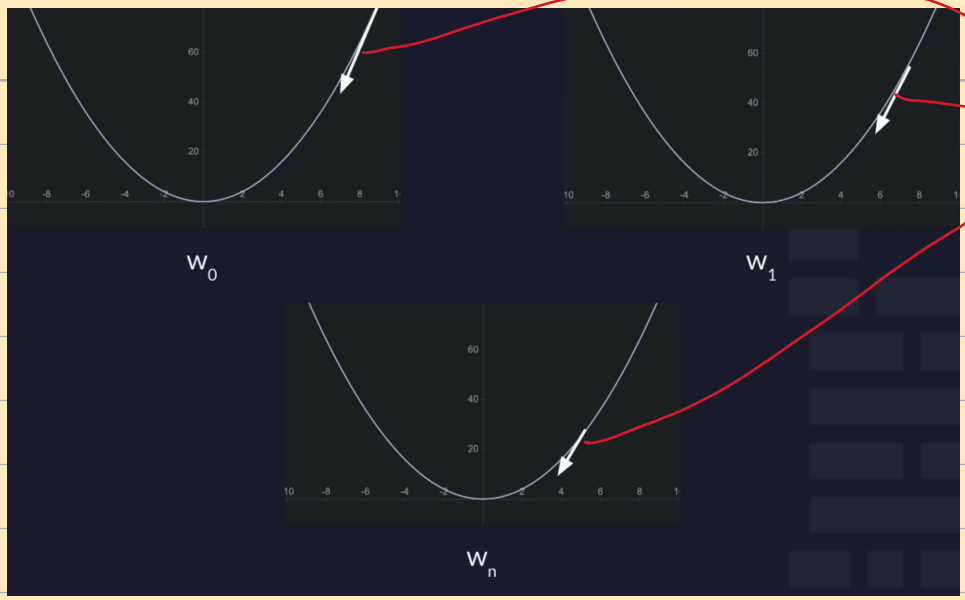
We wanna go this way

And to follow the slope down math has something called Gradient Descent

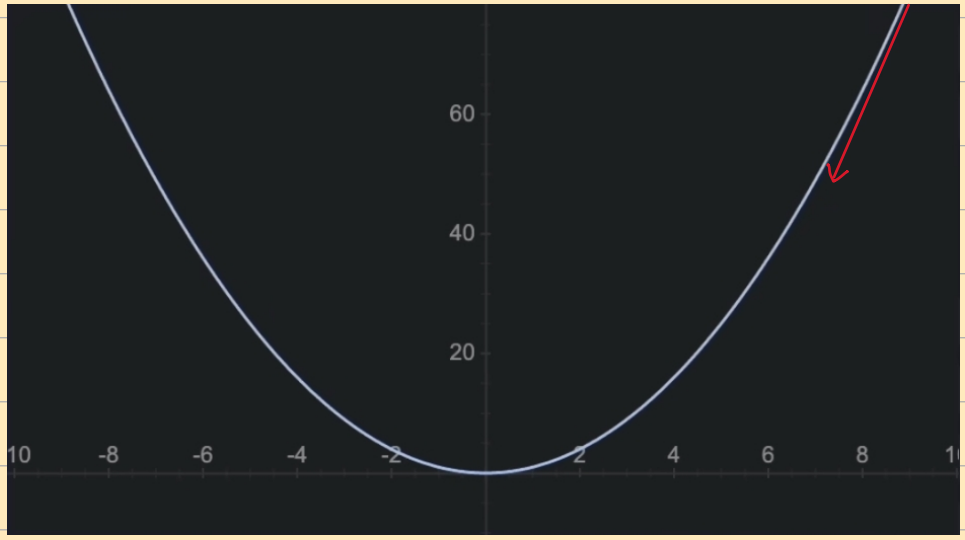
This quadratic has different slopes with respect to some value. The loss to some weight w_0 vs w_1 vs w_n they might all be different.



To what value w_0, w_1, or w_n is contributing the loss can be figured out by calculus to know how much do we have to backstep by.



You might notice the closer we get to the No loss the smaller this step becomes.



So my new value, that is what we call a weight update,

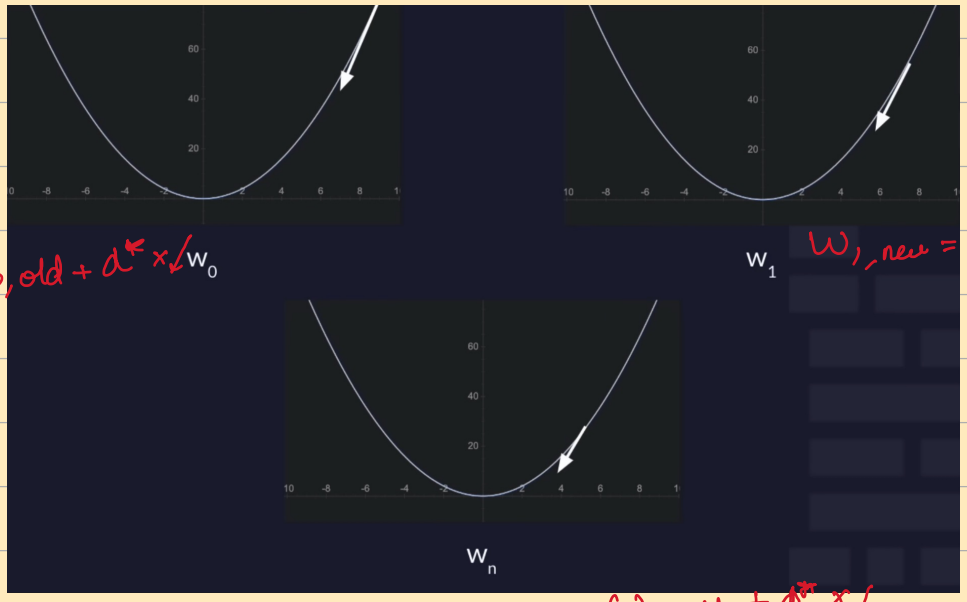
$$W_{0,new} = W_{0,old} + \alpha^* \times$$

Take the old W_0 (old weight) and decrease it in the direction of the arrow. α^* tells don't take a huge step, take small step and see if we are getting closer. The reason why alpha* (α^*) is added because the direction is a negative gradient.

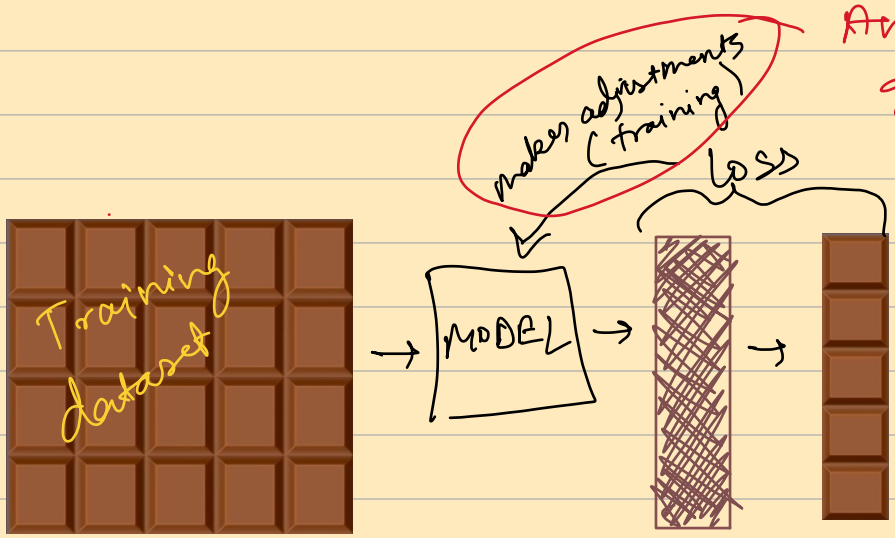
$$W_{0,new} = W_{0,old} + \alpha^* \times$$

This called the learning rate That adjust how we're taking steps. And it controls how

long it takes for our neural net to converge. Or if we set it too high it might even diverge.



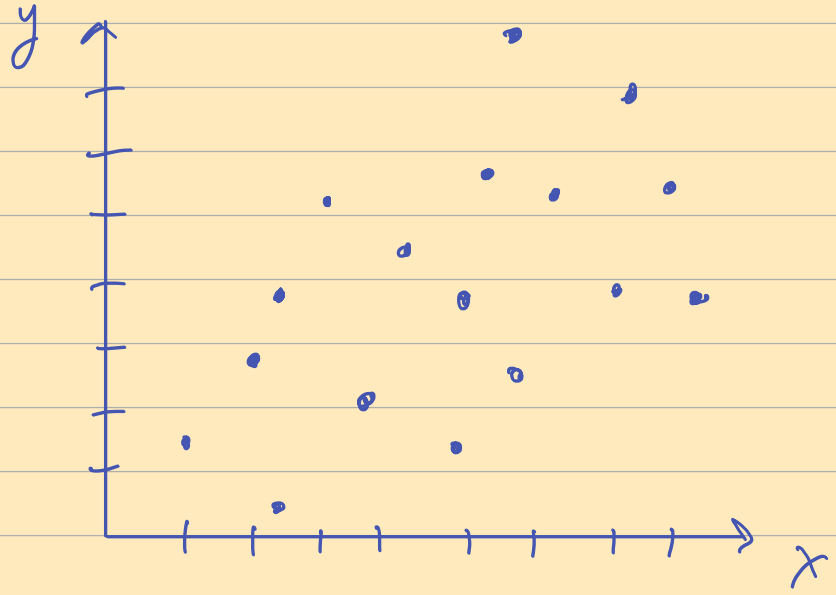
We make the same update to all (w_0, w_1, w_n) them after we calculate the loss, the gradient of the loss with respect to that weight. And that's how backpropagation works.



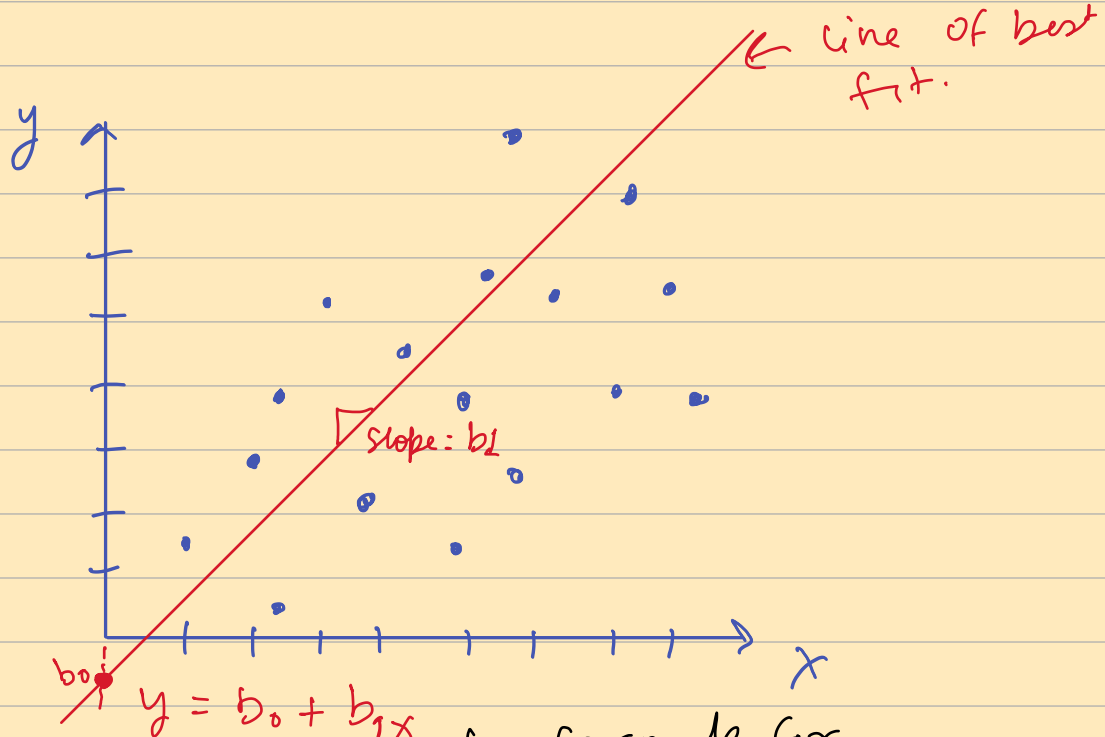
And that is everything going on here, after we calculate the loss, we're calculating gradients, making adjustments in the model. So we're setting all the weights to something adjusted slightly. And then we're going to calculate the gradient. And then take the training set and run it throught the model again, and go throught

the loop again.

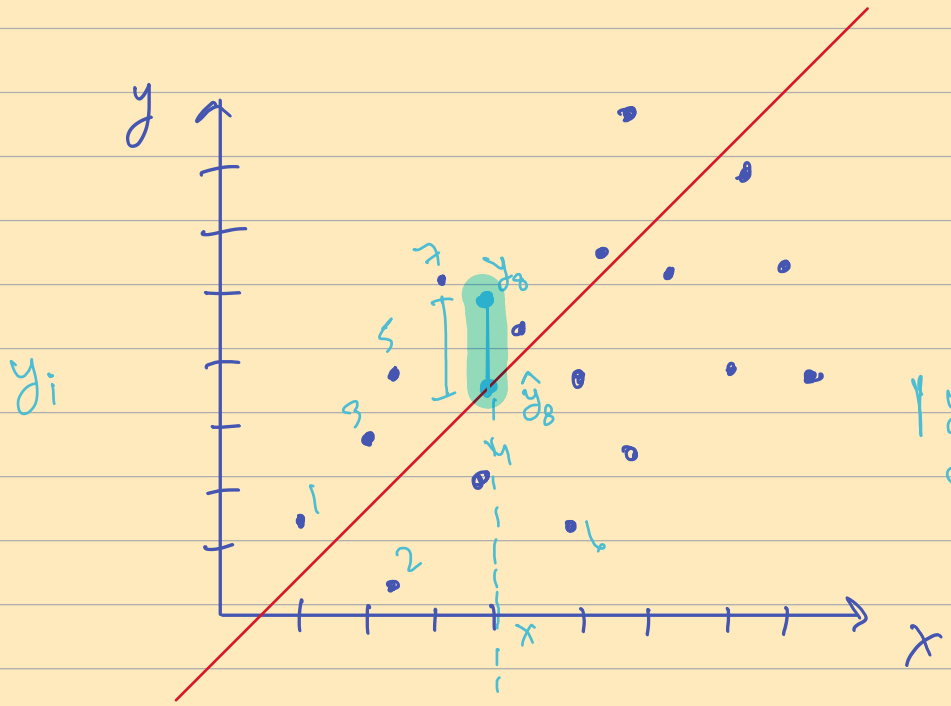
Linear Regression



Our goal in regression is to find the line of best fit that best model the above data.
In linear regression, we want to take our data and fit a linear model to the above data.



Residuals / Error : How far off is our prediction from a data point that we already have.



$\hat{y}_8$ → this is what is the guess, prediction for this specific value of x.

$|y_8 - \hat{y}_8|$ is the residual

Residual can be calculated $|y_i - \hat{y}_i|$ (absolute value because the point can be below the line as well and distance can't be negative)
so the line is best fit is generally trying to decrease these residuals as much as possible.
That means minimizing the sum of all the residuals

$$\sum_i |y_i - \hat{y}_i|$$

Find the b_0 and b_1 which gives the lowest value of the above.

In different circumstances we might attach a

squared to the formula. like

$$\sum_i |y_i - \hat{y}_i|^2$$

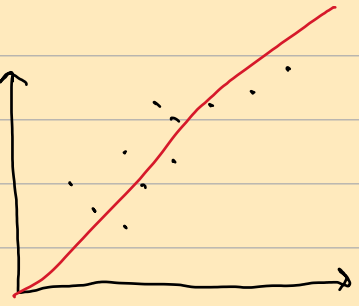
so we are trying to decrease the sum of the squared residuals. This adds a higher penalty for how far off we are from points that are further off.

$y = b_0 + b_1 x$ This is known as simple linear regression.

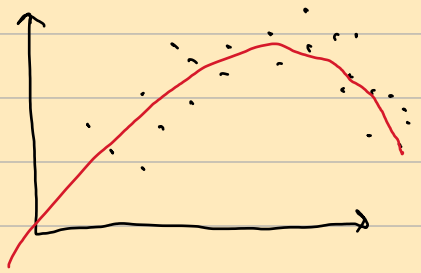
Multiple linear regression : $y = b_0 + b_1 X_1 + b_2 X_2 + \cdots b_n X_n$

Assumptions :

- linearity :



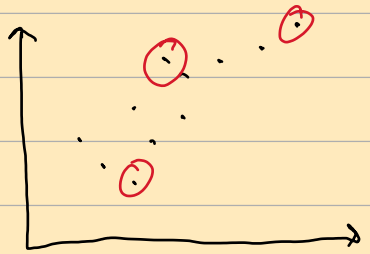
It means the data follow a linear pattern y increase as X increase or y decrease as x increases. So if y and x increases or decreases at constant rate then it's probably linear.



For non-linear data the line has some curve and we don't satisfy that linearity assumption.

So with linearity the data should follow a linear trajectory.

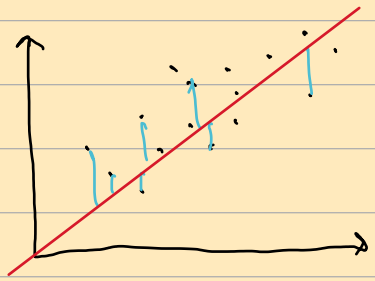
- Independence



That one point should not have influence on other points/data.
All the points/samples in the data set should be independent.
Should not rely on one another and should not affect one another.

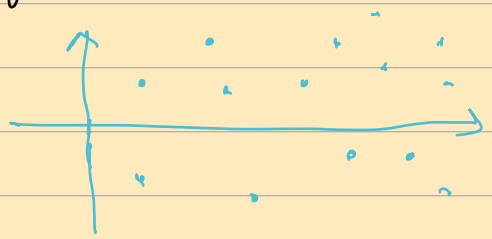
- normality
- homoskedasticity

These two uses the concept residual.

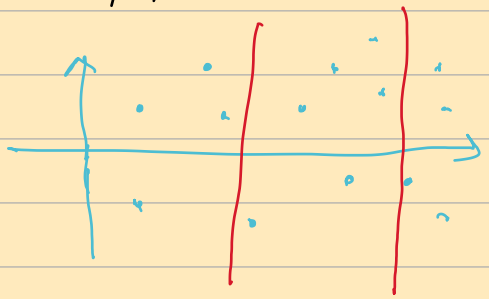


residual plot

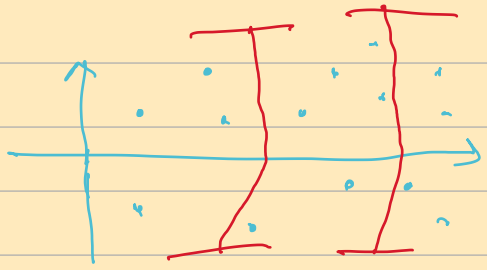
$|y - \hat{y}|$



So the assumption for normality is that the residuals should be normally distributed around the line of best fit.



And homoskedasticity says our variants of these points should remain constant throughout.



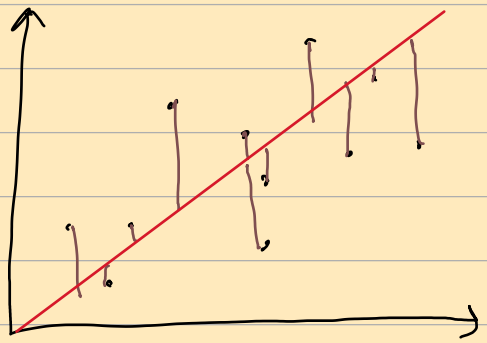
The spread for the two should be same.

If the spread is not equal it might not be appropriate to use linear regression.

Evaluate Linear Regression Model

① Mean Absolute Error (MAE)

Take all the errors, these residuals, sum up the distance for all of them, and then take the average. That can describe how far off are we.



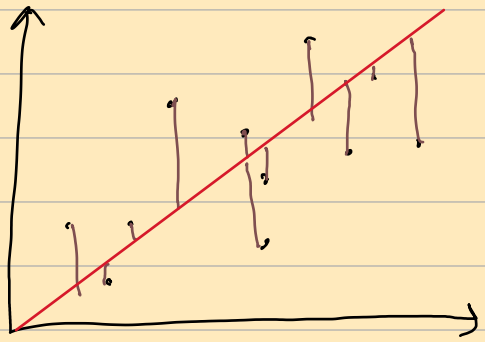
$$\frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} = \text{MAE}$$

(n = how many points there are)

(i = data point)

This is good because when we get the value, we can literally directly compare it to whatever units the y value is.

② Mean Squared Error (MSE)

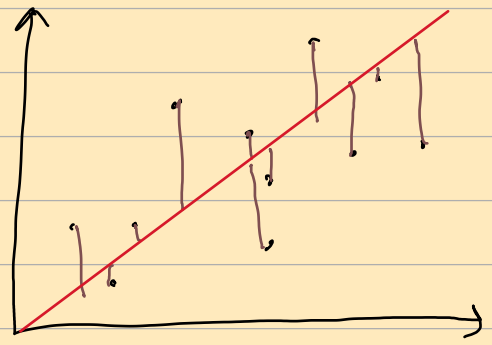


$$\text{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Reason for using MSE is that it helps us punish large errors in the prediction. And later on MSE might be important because of differentiability.

one downside of MSE is that once calculated and go back to compare the values. It gets trickier because now the mean squared error is in terms of y squared.

(3) Root Mean Squared Error (RMSE)



$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

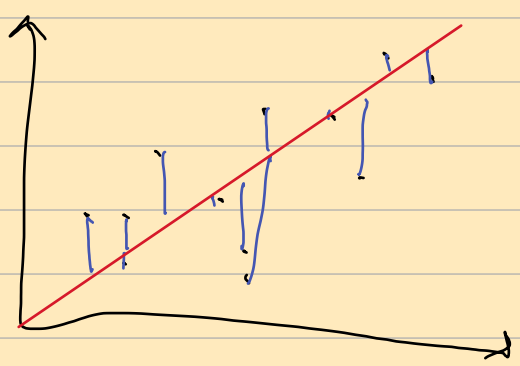
This helps with the downside which we have in MSE.

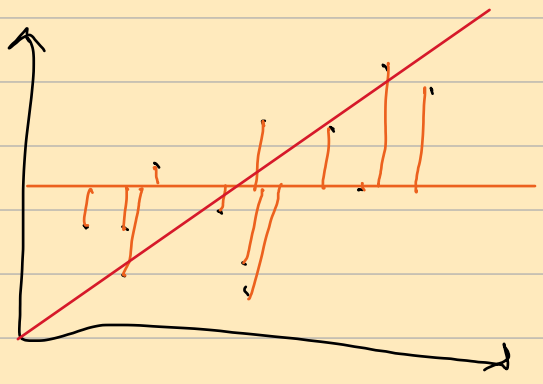
(4) R^2 – Coefficient of Determination

$$R^2 = 1 - \frac{RSS}{TSS}$$

(sum of squared residuals) / (total sum of squares)

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$





RSS tells us what is our error with respect to the line of best fit. And TSS says what is the error with respect to just the average y value.

If the line of best fit is a better fit then its numerator will be smaller than its denominator. And if errors in line of best fit are much smaller then that means that this ratio of the RSS over TSS is going to be very small which means R^2 is going towards 1. If R^2 is going towards 1 then that means we have a good predictor.

So the above models and details are for supervised learning. And in that we have data we have bunch of features for a bunch of different samples but each of those samples has some sort of labels on it. whether that a number a category a class etc. We used those labels and predict new labels of other points we haven't seen yet.

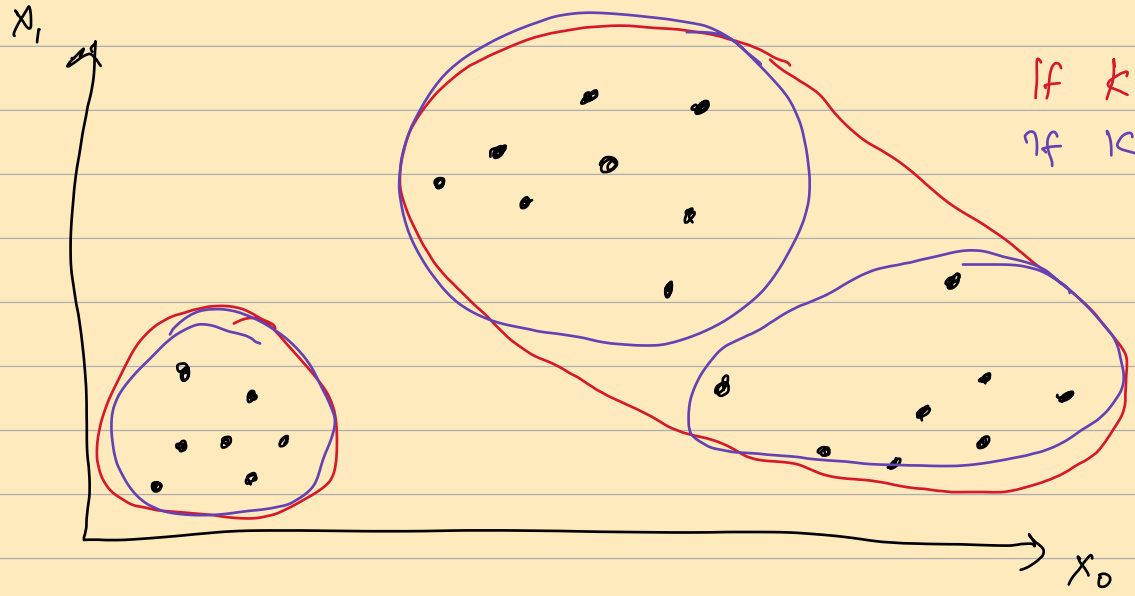
Unsupervised Learning

We have bunch of un-labelled data.

K-means clustering

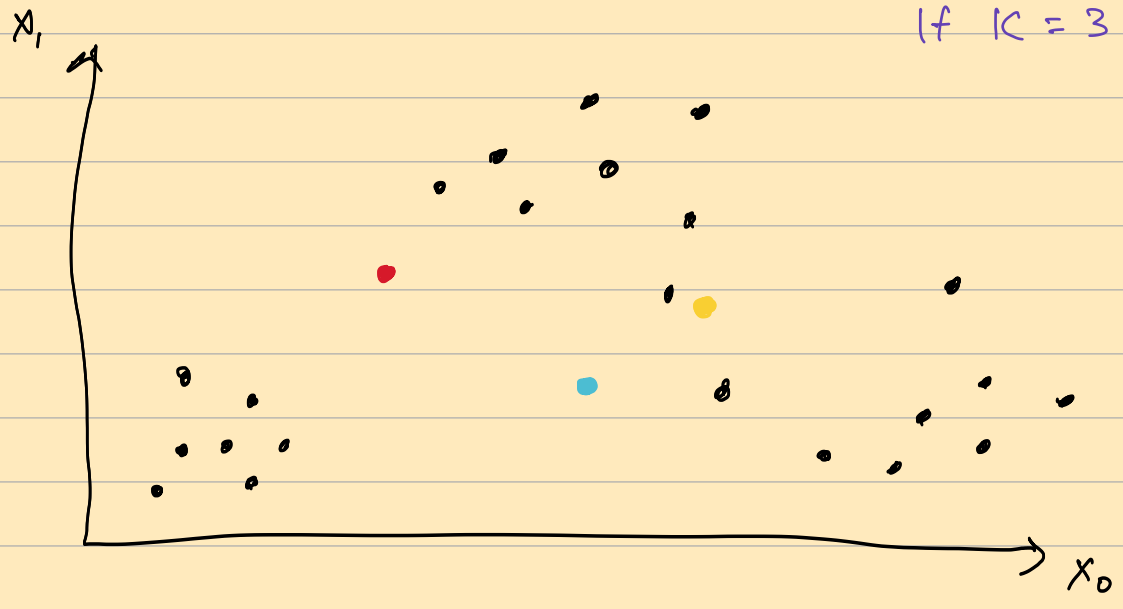
Compute K-clusters from the data.

The K is predefined by the person running the model. And computer goes through and computes the K-clusters.

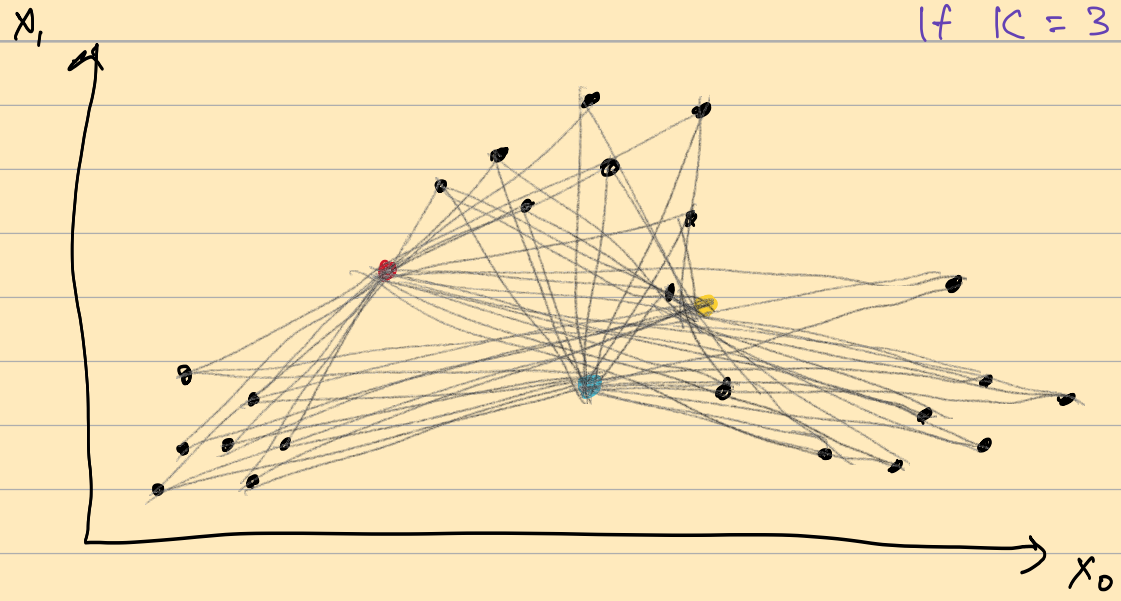


Step computer takes to find the K clusters

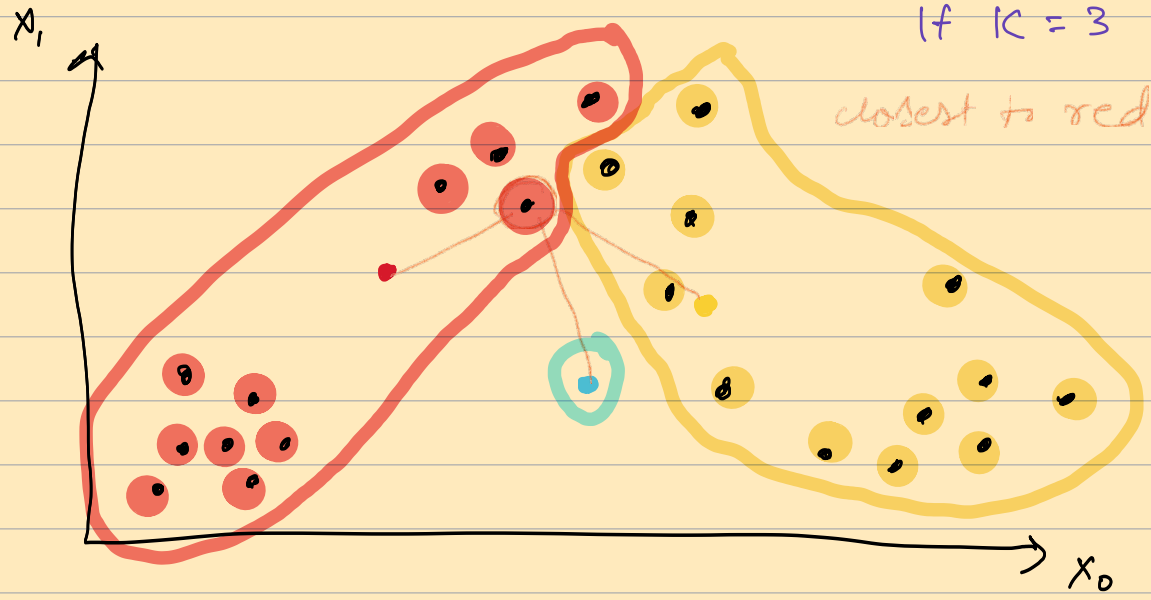
① Chooses 3 random points on the plot to be centroids.



② Calculate the distance between all points and centroids (between points and centroids not between centroids).

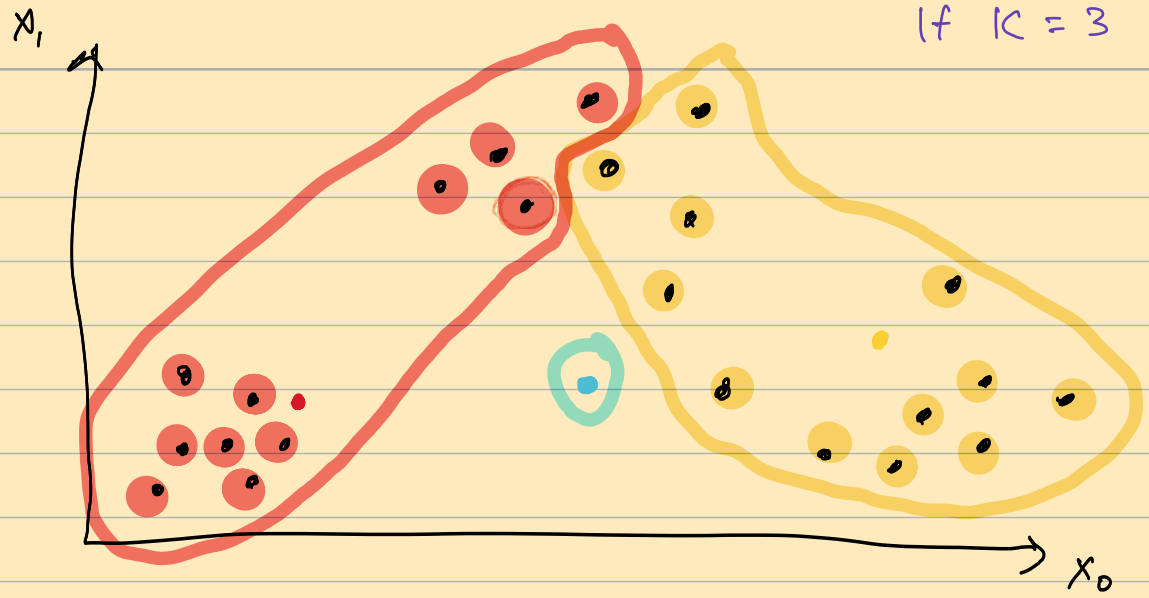


- And also assigning points to closest centroids

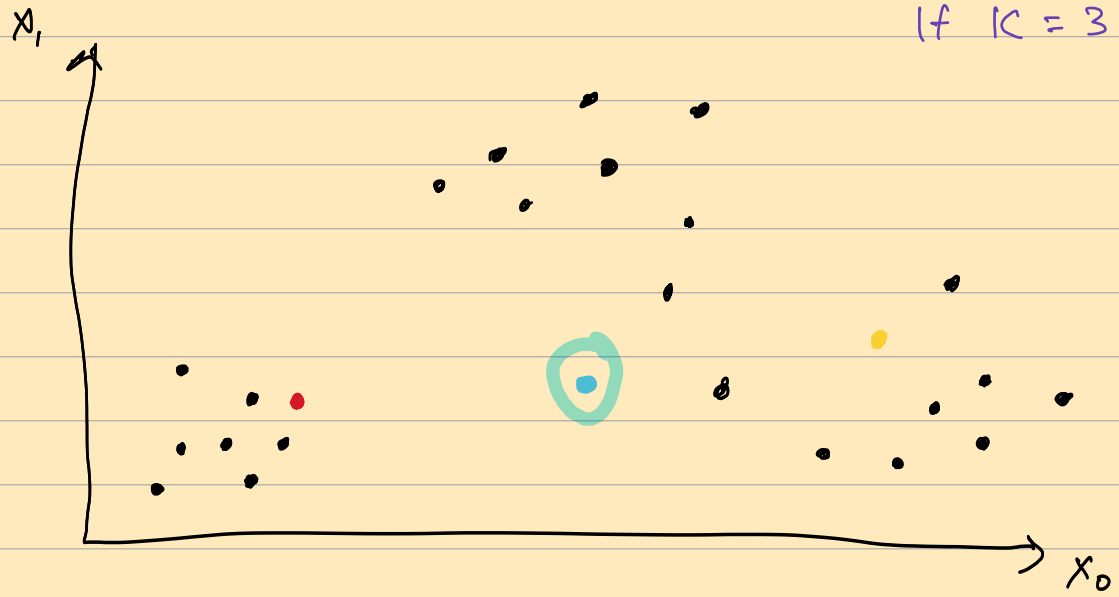


(3) Compute new centroids based on the points we have in all the centroids.

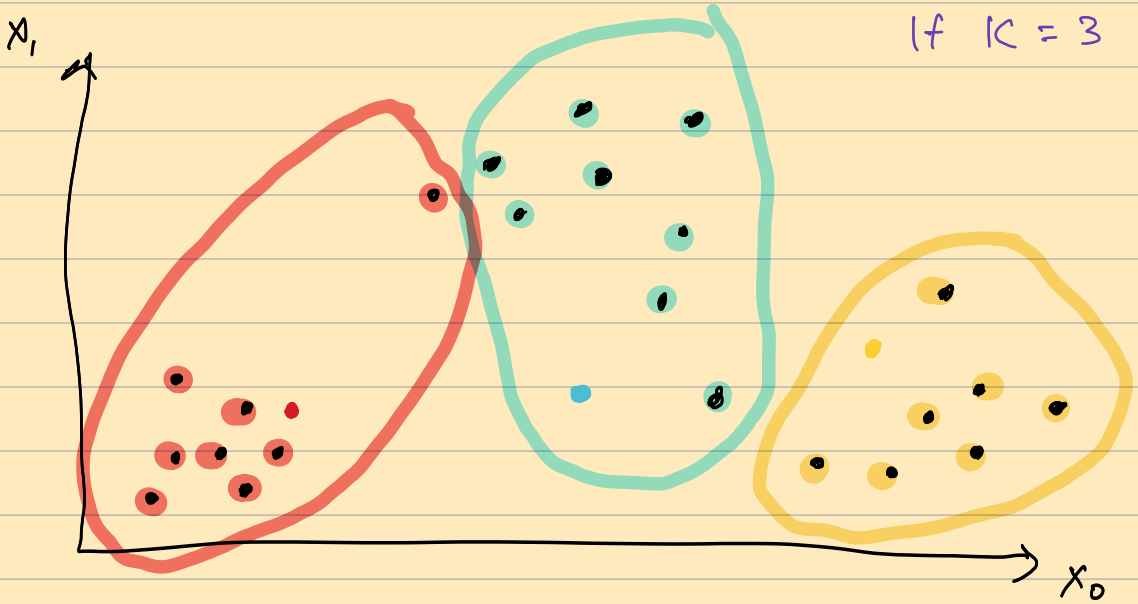
What it means is lets take the average of the points and where is that new centroids



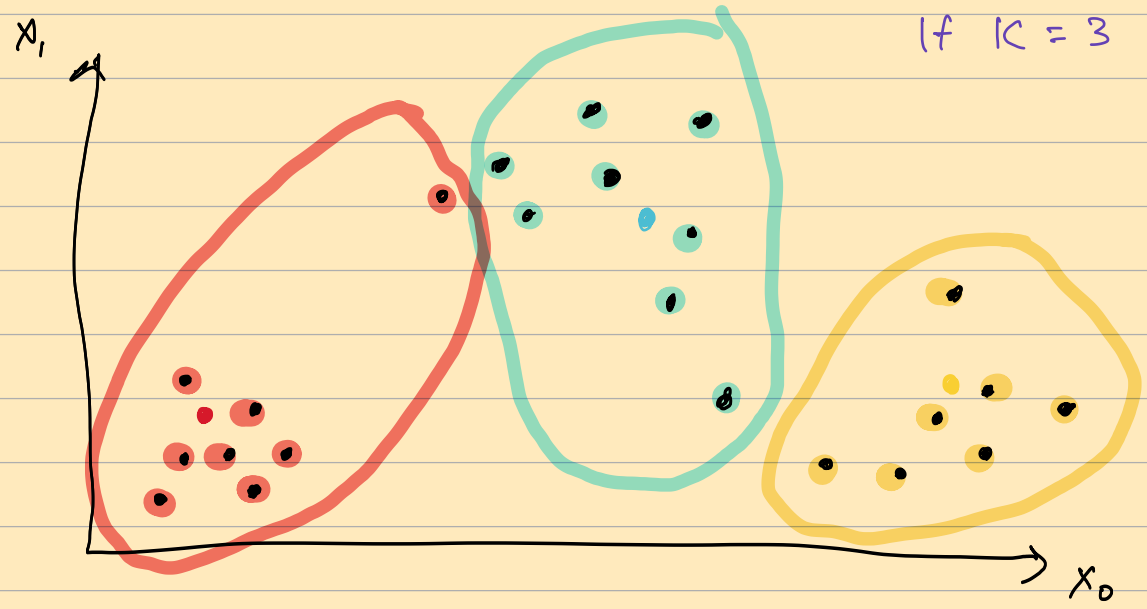
After this erase the previously created centroids



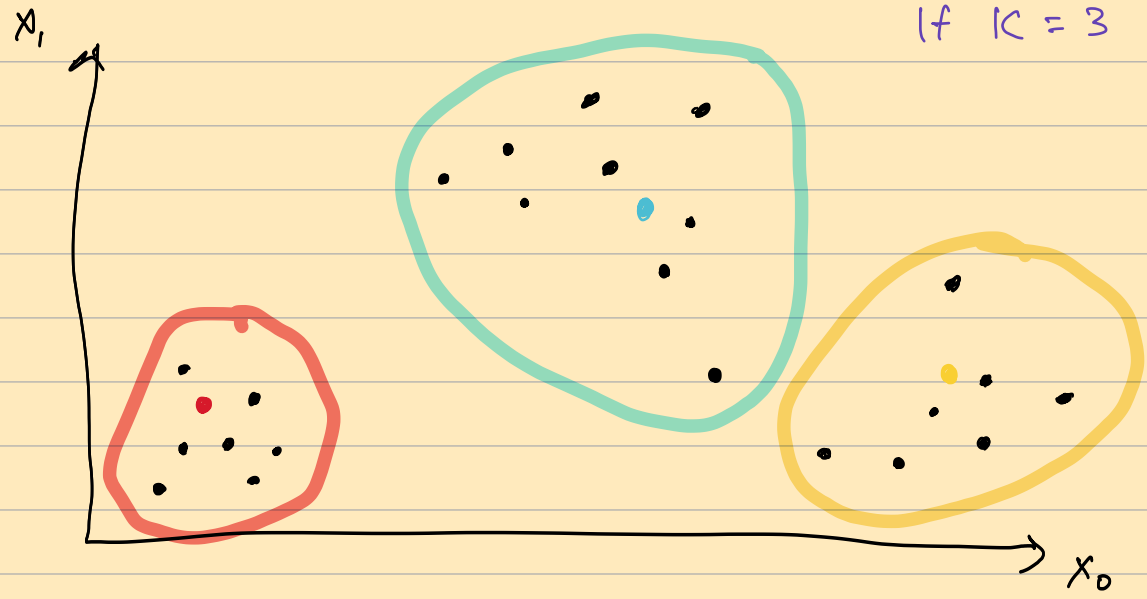
And redo the Step 2



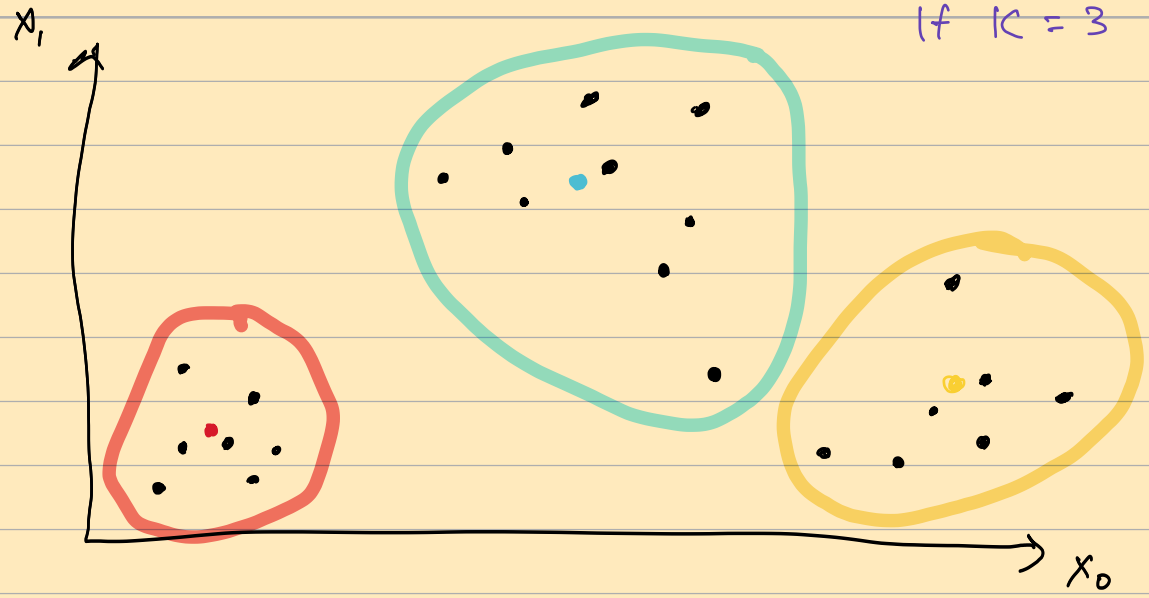
And then we re-compute the new centroids



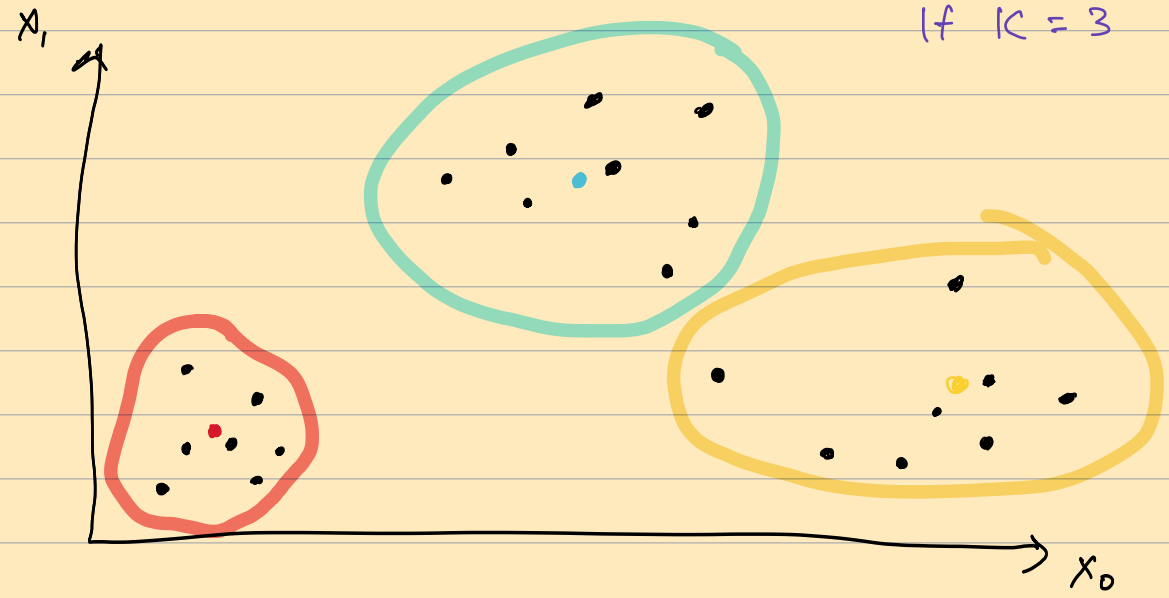
And then again we find the closest point and the new clusters



Then we find the centroids new position



And compute new clusters.

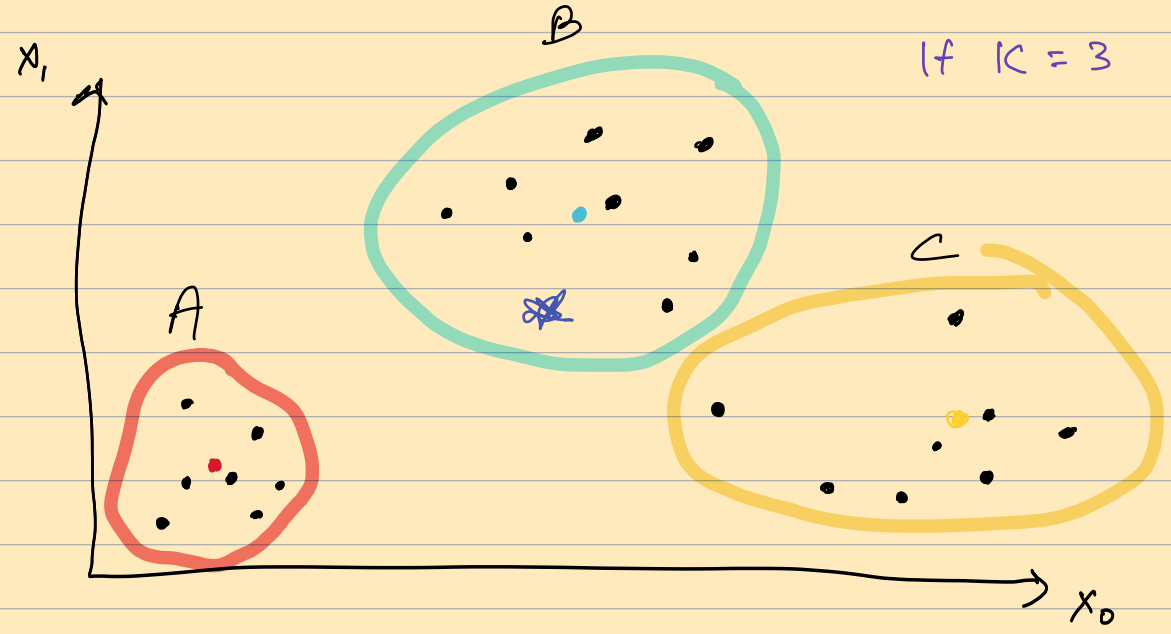


And now if we do the same steps again we get the same result. And then we know we can stop the step 2 and 3 and these are final 3 clusters.

This process is known as Expectation – maximization. (assigning points to closest centroid is expectation step and compute

new centroids is our maximization step)

This finds some pattern or structure in our data. So if we come up with some other data point we will know where it belong to in the cluster.



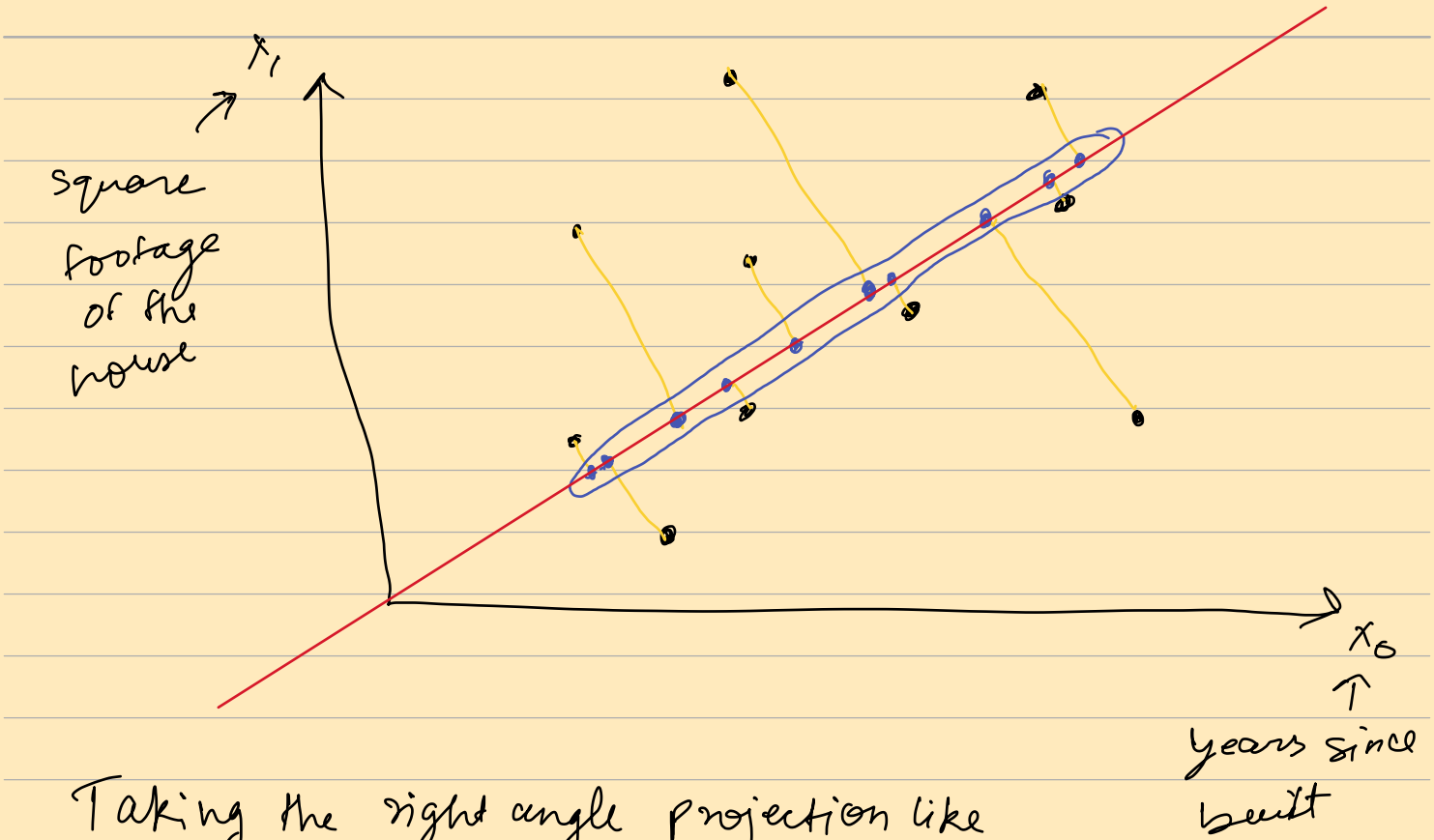
So ✱ is nearest to B so can be put in cluster B.

Principal Component Analysis (PCA)

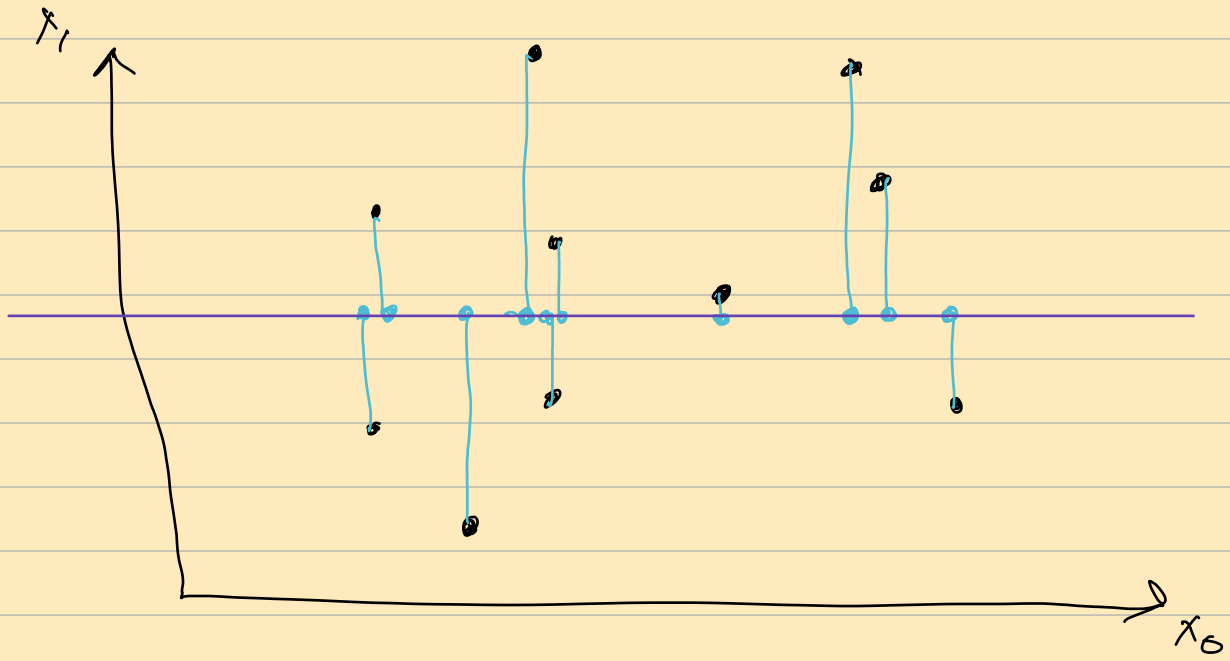
Used dimensionality reduction, which means if we have a bunch of features like x_1, x_2, x_3, x_4, etc. can we reduce that down to one dimension that gives me the most information about all the points are spread relative to one another.

Principal component means basically the component (some direction in the space) with largest variance.

Take the graph for housing prices

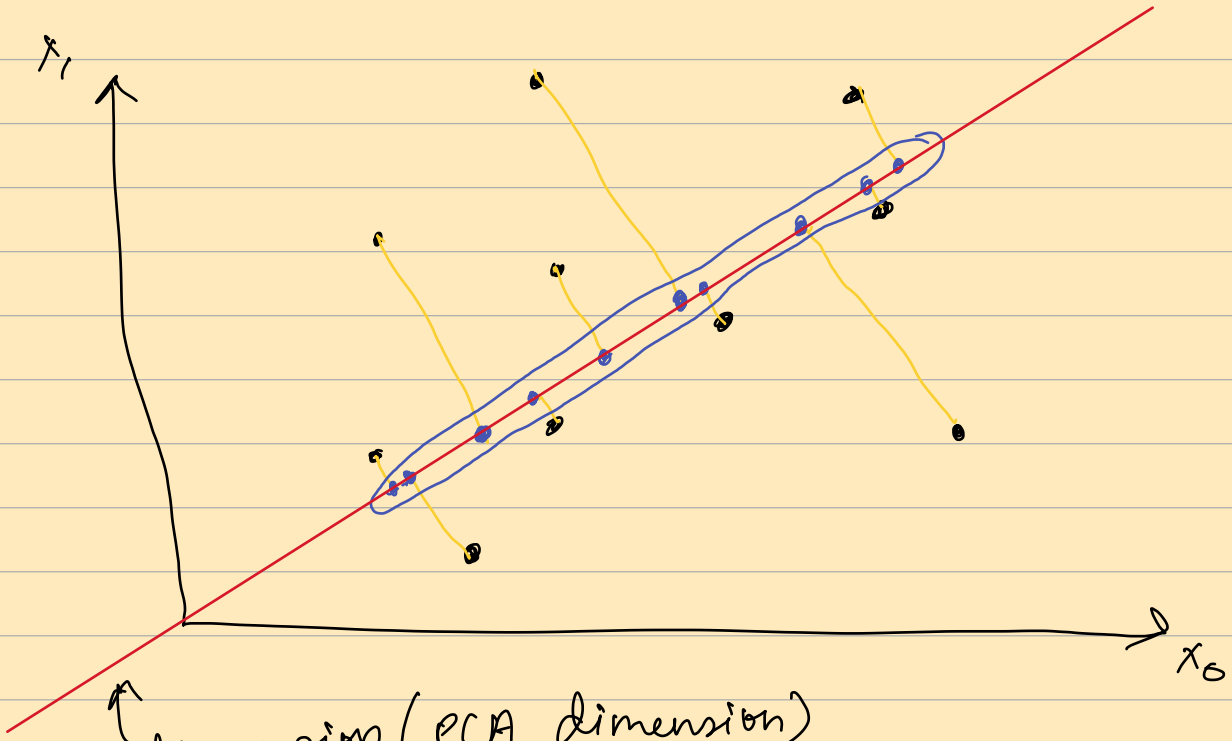


Taking the right angle projection like the above graph onto the line. And what PCA says map all these points on this one-dimensional space. The line with the map will be the 1-D data set. This is not a prediction or anything, this will be the new dataset.



The above graph the points on the line are squished to closer together. which means the variance is not with the largest variance.

Largest Variance will gives us the most discrimination between all of these points. The larger the variance the further spread out these points should be.



this dimension (PCA dimension)

① minimising the projection residual (the |)

② maximising the variance between the points)

THE END