Setting up

Website: <https://techwithtim.net/>

video guide: <https://www.youtube.com/watch?v=WFr2WgN9_xE&t=1890s>

Tensorflow support python 3.6 not 3.7 or higher atm.

Making virtual environment: ( “tensorflow” is just the name of the virtual environment )

conda create -n tensorflow1 python=3.6

activate environment:

*activate tensorflow1*

few pip install commands

Imports:

pip install tensorflow

pip install keras

pip install numpy

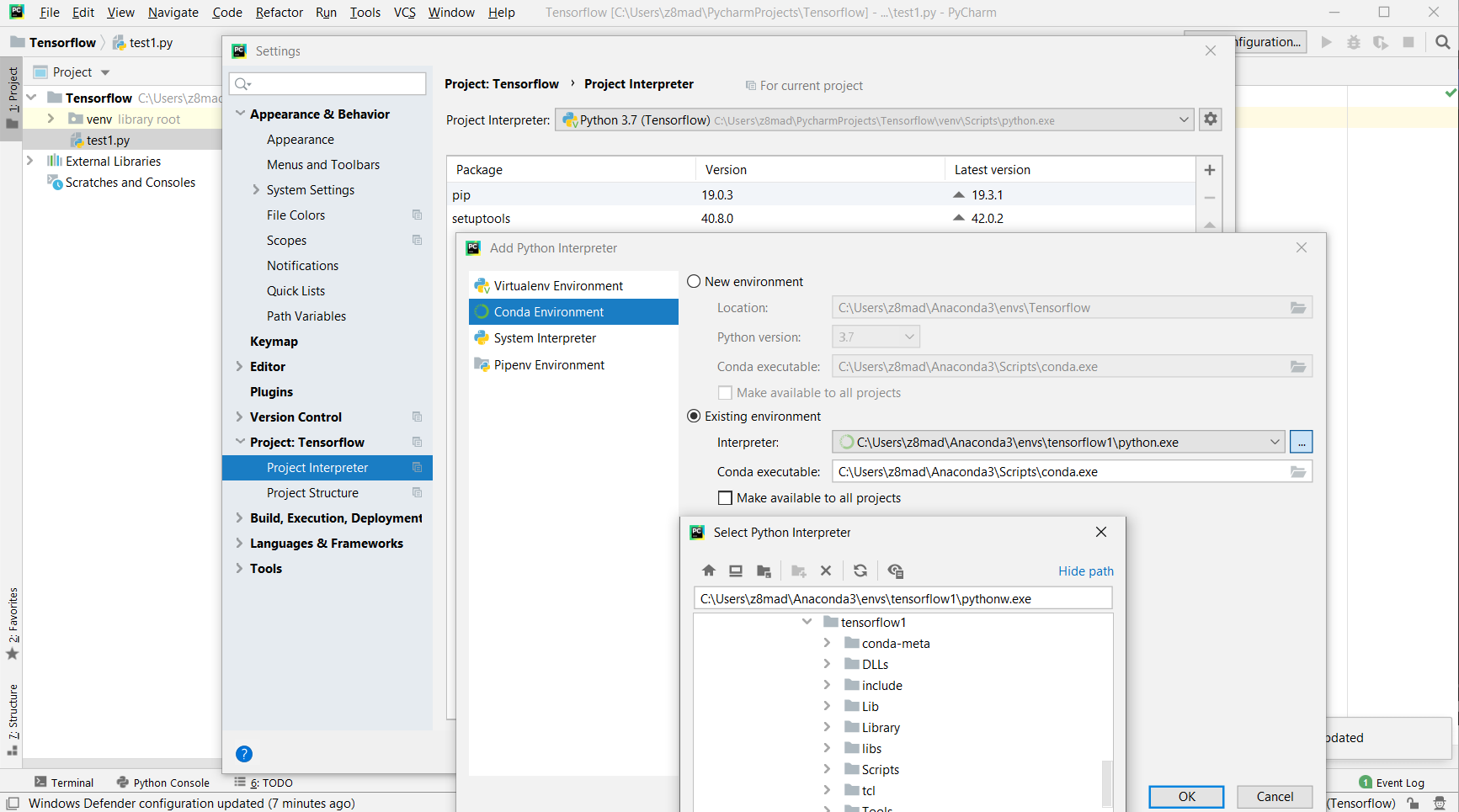
pip install pandas

pip install sklearn

pip install matplotlib (library for basic graph’s and grids)

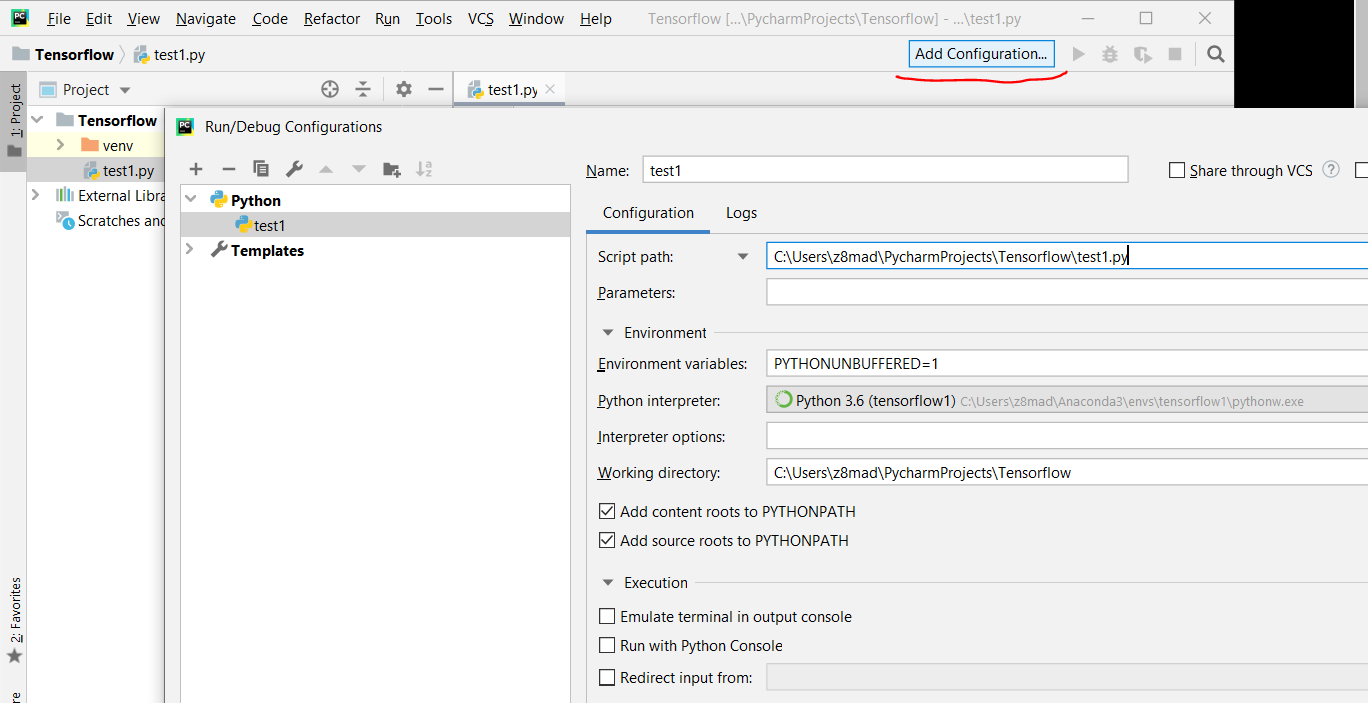
setting up pyCharm

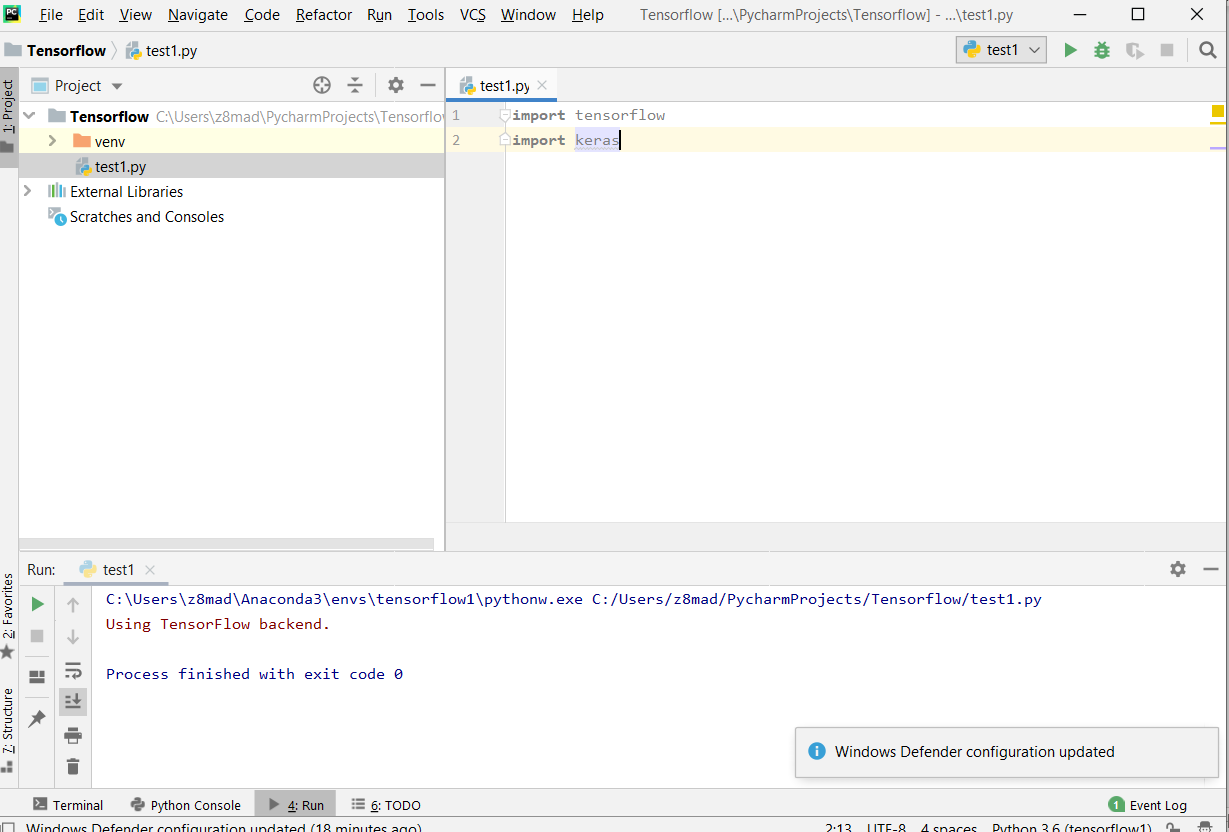
go to Files -> settings -> project ( name ) -> project interpreter -> select the setting tap  and click use existing environment and find your virtual environment path is : C: \Users\( PC Name ) \anaconda3\envs\ ( file name ) \ Pythonw.exe



Now add configuration’s

Click on the add configuration -> click the plus button -> give the file a name just gives it the name that your other file has -> after that select your file by clicked the file icon .



Type to see if its working :

Import tensorflow

Import keras

If its working as it should you will using tensorflow backend.

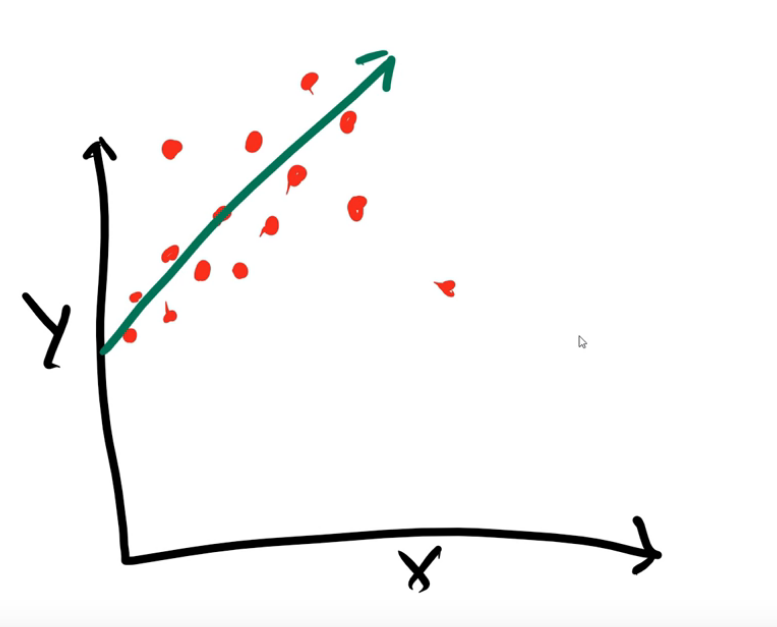
Machine learning with the student-mat.csv dataset

Text-Based Guide: <https://techwithtim.net/tutorials/machine-learning-python/introduction/>

UCI Student Data Set: <https://archive.ics.uci.edu/ml/datasets/Student+Performance>

Most information will be in the PyCharm file

**import** tensorflow  
**import** keras  
**import** sklearn  
**from** sklearn **import** linear\_model  
**from** sklearn.utils **import** shuffle  
**import** pandas **as** pd  
**import** numpy **as** np  
  
*#reads all the data in the csv file with pandas ( panadas make all the data into array's )  
#the ", sep=";" line just replaces the ; with a comma ( , )*data = pd.read\_csv(**"student-mat.csv"**, sep=**";"**)  
*#this will print the 5 first datasets ( this has no filter so it will just take everything from the student-mat file*print(data.head())  
  
*#here we can select the data we want from the student-mat.csv file ( all intingers ( numbers ))*data = data [[**"G1"**,**"G2"**,**"G3"**,**"studytime"**,**"failures"**,**"absences"**,**"health"**,**"freetime"**]]  
*#with the filter*print(data.head())

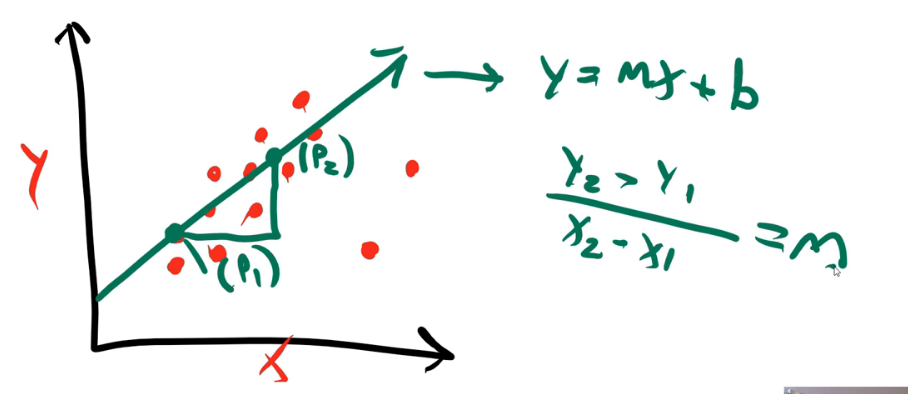
linear regression

linear regression is a basic algorithmic

it looks at a lot of different datapoints and tries to make a best fit line to the datapoints

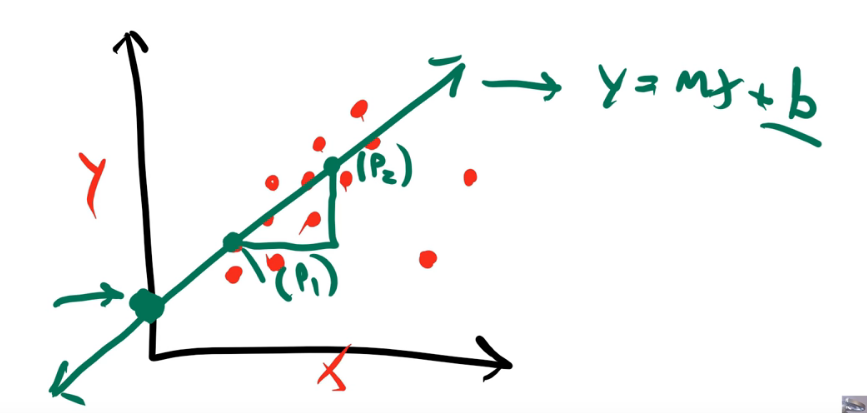
this is how it looks but if the data is more random linear regression should not be used.

Linear regression is when we have data to correlate to each other (data that have meaning to each other)



M = the slope of the line (slow is how fast the line increases ( you can have a positive and a negative slope in this case its positive))

You can calculate M with

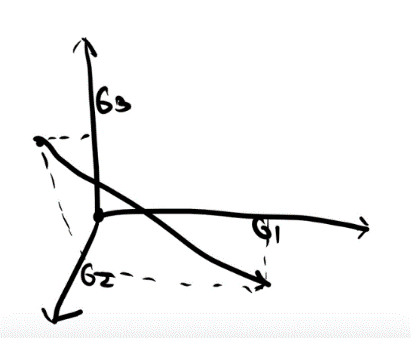


B is where we start the line in this case its

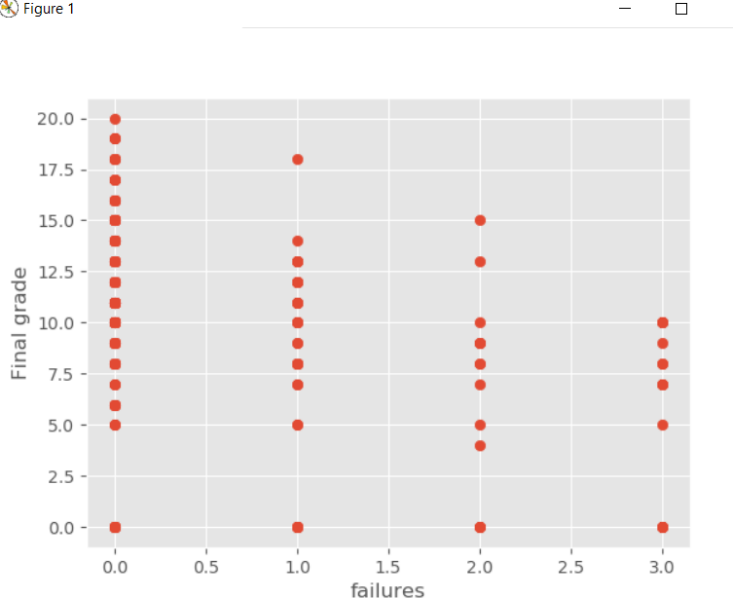
This at the arrow on the y axis

Fx grades:

Y axis is the max grade (20) and x axis is the a student grade (17) you would calculate it by

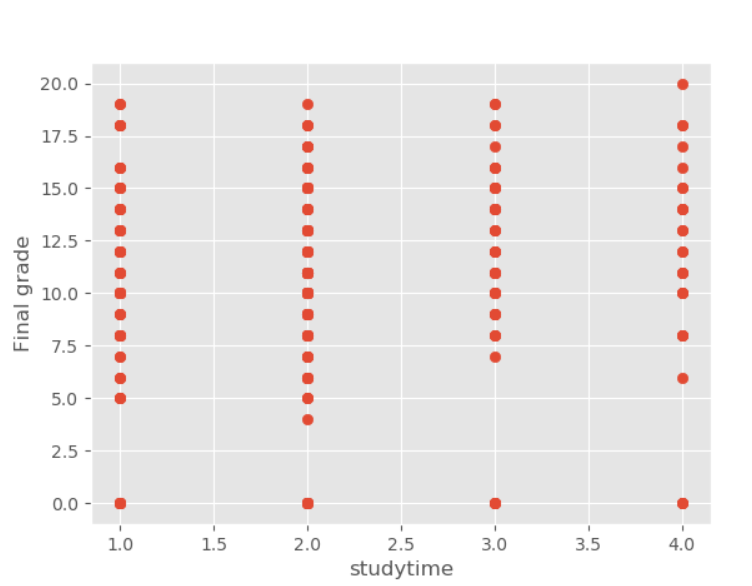


It makes a multi-dimensional space where it tries to make a graph of how the linear regression would look like.

Explaning the graph

failurs

This is their failurs and you can see people with 3 failturs have a lower grade compaired to people with student’s with a lower failur rateing



Study time

Here you can see how to study time effect’s their grades and here you can see that having more study time is not necessarily better grades.

Final code for all this:

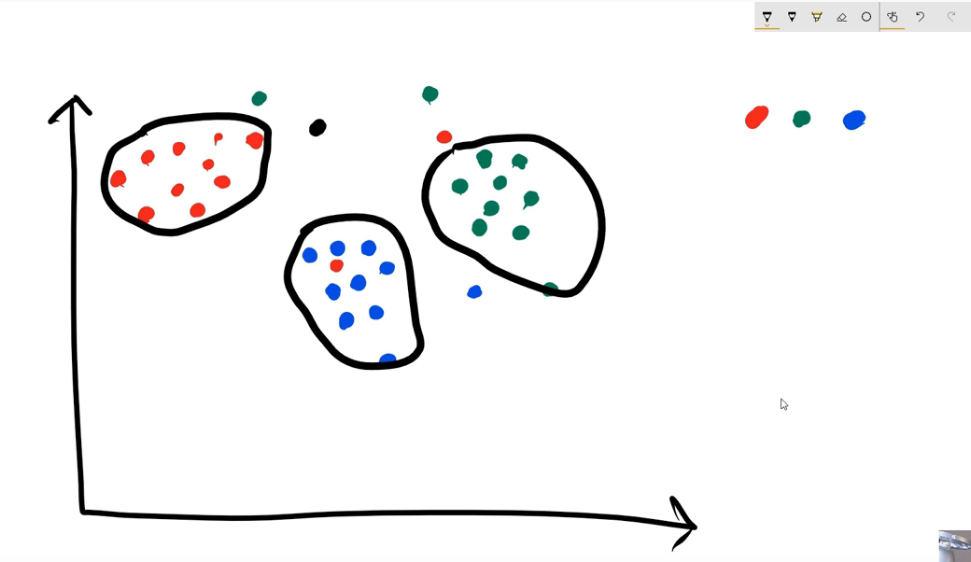
**import** tensorflow  
**import** keras  
**import** sklearn  
**from** sklearn **import** linear\_model  
**from** sklearn.utils **import** shuffle  
**import** pandas **as** pd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** pyplot  
**import** pickle  
**from** matplotlib **import** style  
  
*#reads all the data in the csv file with pandas ( panadas make all the data into array's )  
#the ", sep=";" line just replaces the ; with a comma ( , )*data = pd.read\_csv(**"student-mat.csv"**, sep=**";"**)  
  
*#here we can select the data we want from the student-mat.csv file ( all intingers ( numbers ))  
#everything is a labes fx G1 G2 G3 ...  
#and all the information the labes have are attributes fx first row  
#Table: G1  
#attributes: 5*data = data [[**"G1"**,**"G2"**,**"G3"**,**"studytime"**,**"failures"**,**"absences"**,**"health"**,**"freetime"**]]  
*#the (predict = "G3") is also knowed as labels and lables are what you are trying to get / looking for*predict = **"G3"***#two arrays  
  
#this will make a dataFrame. it will also use the data from on top  
#x is also all the training data*x = np.array(data.drop([predict], 1))  
  
*#y are all the labels*y = np.array(data[predict])  
  
*#training  
#we are taking the x and y from on top and splitting them into 4 diffrent arrays  
#x\_train is = x from on top and y\_train is y from on top aswell.  
#we are useing this to test ot output of the model*x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size=0.1)  
  
  
*#commen this out since i dont need to run it any more i have found the best  
#if you want to run it more times you can remove the commens ''' ''' and then just run it as many time as wanted***'''  
#this just loops it 10000 times to keep looking for the best model  
best = 0  
for \_ in range(10000):  
  
 #training data is for training the bot and the testing data is when useing the bot for testing ( testing is all the data and training is only a little amount)  
 x\_train,x\_test,y\_train ,y\_test = sklearn.model\_selection.train\_test\_split(x,y,test\_size=0.1)  
  
  
 #skipping the training and the saving  
 linear = linear\_model.LinearRegression()  
  
 linear.fit(x\_train,y\_train)  
 acc = linear.score(x\_test,y\_test)  
 print(acc)  
 #this is just saying if acc ( the model ) is bettere then the old one save it  
 if acc > best:  
 best = acc  
 #useing Pickel  
 #this will save a pickel file in the dir  
 #write binary = wb  
 with open("studentmodel.pickle","wb") as f:  
 pickle.dump(linear, f)  
   
'''***#read pickle file  
#read binary = rb*pickle\_in = open(**"studentmodel.pickle"**,**"rb"**)  
*#will load the the model into the variable linear*linear = pickle.load(pickle\_in)  
  
  
*#this is equel to M in the calculation ( coefficient )  
#the more M's there are the mode dimensional is caluclation is fx there are 5 outputs in this one so its 5 dimensional space.*print (**"coefficient: \n"**, + linear.coef\_)  
*#this is equl to B in the calculation ( Intercept )*print(**"Intercept: \n"**, + linear.intercept\_)  
  
*#this will take array's and will do a ton of predictions and guss on the test data ( that is not trained )*predictions = linear.predict(x\_test)  
  
**for** x **in** range(len(predictions)):  
 print(predictions[x], x\_test[x],y\_test[x])  
  
*#saving model  
#why save the model ?  
#to use the best model and not to retrain your model if you are useing a lot of data it will take a while but saving it will save time.  
# and if you can find a model that has a very high accuracy to use it.  
  
  
  
  
#useing matplotlib to make a graph / grid  
#this will show G1  
#keep in mind there are 600 studients but you are not seeing all of the datapoints to the 600 student's since a lot of them are overlapping  
#chose between: "G1","G2","G3","studytime","failures","absences","health","freetime"*p = **'studytime'**style.use(**"ggplot"**)  
*#useing a scatter plot*pyplot.scatter(data[p],data[**"G3"**])  
pyplot.xlabel(p)  
pyplot.ylabel(**"Final grade"**)  
pyplot.show()

Machine learning with the Car dataset ( incorrect data / missing data )

The dataset

Video time stamp 58:00

UCI Car Evaluation Data Set: [http://techwithtim.net/wp-content/upl...](https://www.youtube.com/redirect?redir_token=1Y39WD_ppdMVU3kL_gMFQGZbTr98MTU3NTk2MjA5NUAxNTc1ODc1Njk1&q=http%3A%2F%2Ftechwithtim.net%2Fwp-content%2Fuploads%2F2019%2F01%2FCar-Data-Set.zip&v=WFr2WgN9_xE&event=video_description)

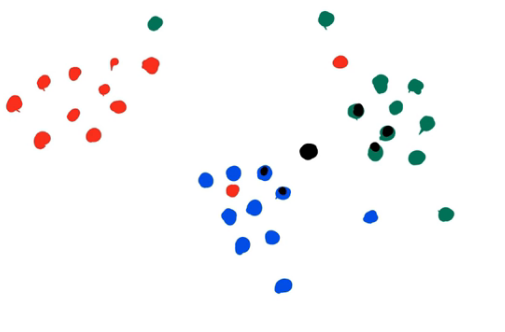
The k-nearest neighbors

It’s basically where you classify the dots and see where they belong. You can see that most of the red, green and blue are clustered together in a pile for them self but the black one is a unknown dot there for you need to say what’s closes to the black dot in this case it would be red and you can classify it as a red dot.

How the AI works:

It looks for datapoints and trys to group them and group the black dots to the group that is closes to it.

If then it will be looking for the 3 closes dots after that it would take a vote to how many of the same dots are closes when it picks the right clock it will classify it as that color. Fx look at the green dot and k =3 it will probly be like green, red, red and since there are more red’s then green’s it will classify as a red datapoint



Looking at the image to the right side you can see that k = 5 and there are 5 dots it’s the same principle that you do a vote and so how many greens & blues are closes the two dot but in this case its green. The K value is normally uneven numbers so It can calculate which datapoint is closes if K is a even number like 2 4 6 8 … it won’t work since if you have 2 green 2 blues you won’t be able to figure out what is what.

**Math**

3-dimensional space:

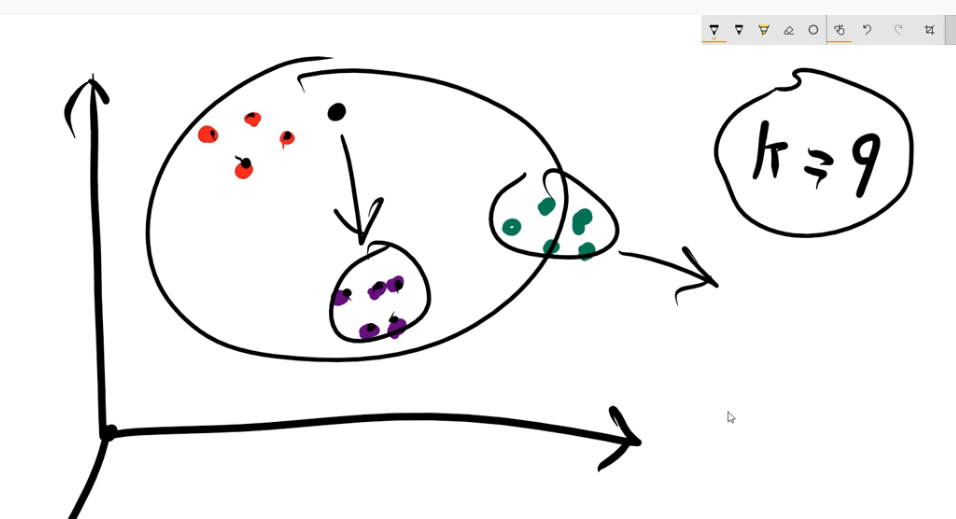
D is the distance between the datapoints the c

Normal dimensional space:

How to use the calculation: (this is 2 points the unknown and fx a blue dot to distance between them.

x1, y1 = 0,0: x2, y2= 0,4 =

when we are using the AI, we are using 3-dimensional space so we need to use the D calculation seen below



Need to be careful not picking a too high number for K if you do this will happen (look on the photo)

(imagen the green dots are feather away)

But since there are more purple dots then there are red ones even the red ones are closes it will choose purple as it’s datapoint

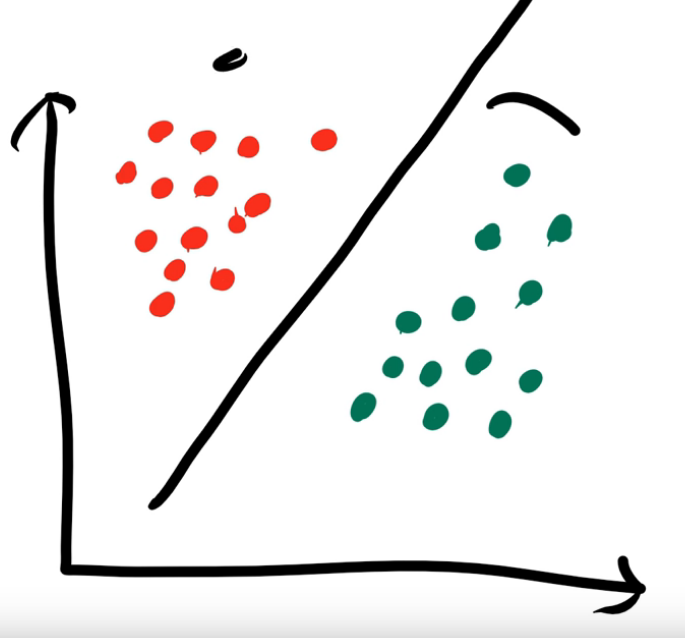
Limitations

If there are a million datapoints and the AI will calculate the datapoints of the distance between all of them, it will take unnecessary time.

Final code:

**import** tensorflow  
**import** pandas **as** pd  
**import** numpy **as** np  
**import** sklearn  
**from** sklearn.utils **import** shuffle  
**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** linear\_model,preprocessing  
  
*#k-Nearest Neighbors - p.1 - irregular data ( video time stamp 55:00 )  
#data is is not 100% ( like missing data )***'''  
Pandas will take the frist row as what the inputs are so buying = vhigh and maint is also vhigh  
1: buying,maint,door,persons,lug\_boot,safety,class  
2: vhigh,vhigh,2,2,small,low,unacc  
3: vhigh,vhigh,2,2,small,med,unacc  
4: vhigh,vhigh,2,2,small,high,unacc  
  
'''**data = pd.read\_csv(**"car.data"**)  
print(data.head())  
  
  
*#converting words to intigers fx vhigh to 3 and med to 2...  
#lableencoder will take the lables and convert them into intigers ( this is just a object )*le = preprocessing.LabelEncoder()  
*#this will get everything from the buying colum and take the word fx vhigh and turn it into a list & a intiger.  
#this will also make a numpy array*buying = le.fit\_transform(list(data[**"buying"**]))  
maint = le.fit\_transform(list(data[**"maint"**]))  
door = le.fit\_transform(list(data[**"door"**]))  
persons = le.fit\_transform(list(data[**"persons"**]))  
lug\_boot = le.fit\_transform(list(data[**"lug\_boot"**]))  
safety = le.fit\_transform(list(data[**"safety"**]))  
cls = le.fit\_transform(list(data[**"class"**]))  
  
print(buying)  
predit = **"class"***#this is a list conversion ( convert the numpy array to a list ( the zip function puts the list / arrays togeather in one line ))*x = list(zip(buying,maint,door,persons,lug\_boot,safety))  
  
y = list(cls)  
  
x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size=0.1)  
*#the n\_neighbors is how many neighbors it will have ( change the number for a better accuracy )*model = KNeighborsClassifier(n\_neighbors=9)  
  
model.fit(x\_train,y\_train)  
acc = model.score(x\_test,y\_test)  
print(acc)  
  
predicted = model.predict(x\_test)  
  
name = [**"unacc"**, **"acc"**, **"good"**, **"vgood"**]  
**for** x **in** range(len(predicted)):  
 print(**"predicted: "**,name[predicted[x]], **"Data: "**, x\_test[x], **"Actual: "**, name[y\_test[x]])  
 n = model.kneighbors([x\_test[x]],9, **True**)  
 print(**"N: "**, n)

SVM = support vector machine

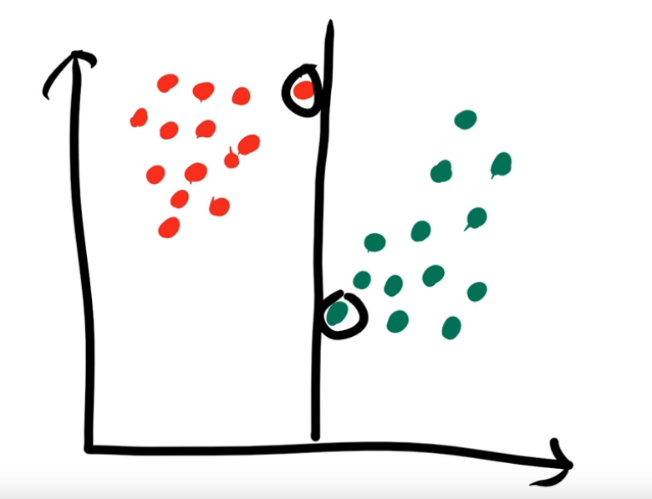


Support vector machine

In geometry, a **hyperplane** is a subspace whose dimension is one less than that of its ambient space. If a space is 3-dimensional then its **hyperplanes** are the 2-dimensional planes, while if the space is 2-dimensional, its **hyperplanes** are the 1-dimensional lines

The hyperplane will use something like a straight line to divide the data, basically a linear way of dividing the data ( look on the image to see how it would look ).

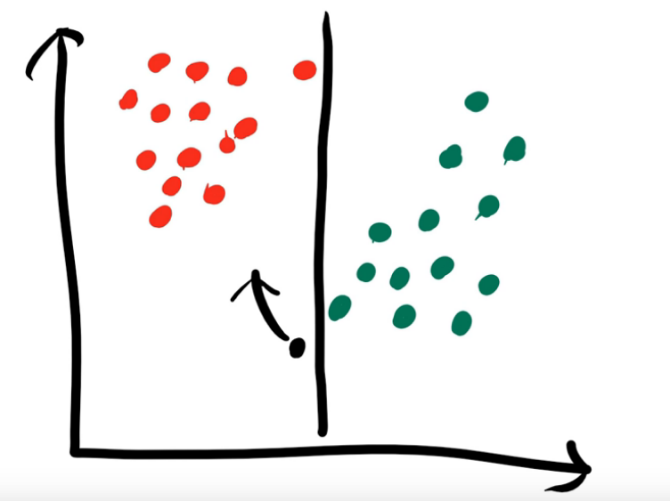
Hyperplane requrements:

Same distance from the 2 obbisit class ( datapoints )

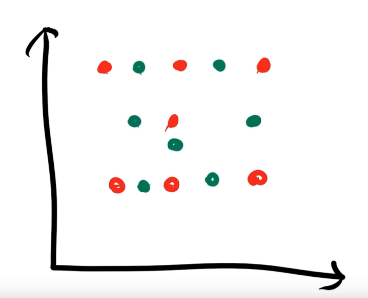
There are many ways of drawing a hyperplane if you look at the 2 images both are hyperplanes but the line if different. That’s because the 2 main factors in drawing a hyperplane is having 2 datapoints with the same distance to the line.

Remember with hyperplanes you want the distance to be as big as possible to make a good **margin**

What is a **margin**

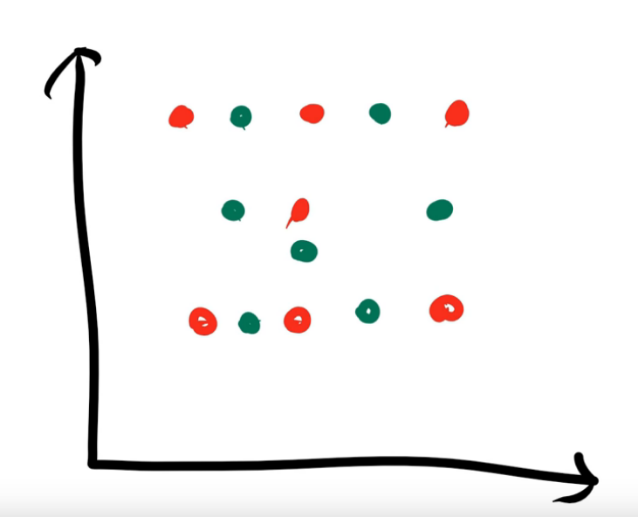
a margin is the space between the point and the line this margin is extremely important when predicting if a datapoint in this case is red or green. if you look at the 3rd image if you see the line is going down and the black dot is closes to the green one’s it will classify it as red since its on the side where all the red dots are, **this is also the main reason why you want as much margin as possible**.

The 2 closes points are known as support vectors



Hyperplane kernel are used for data that is not perfect like showed in the image to the side

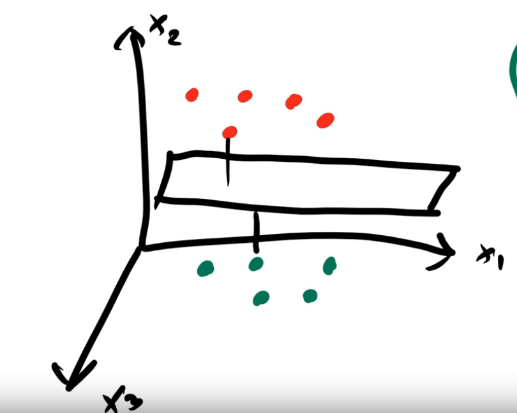
hyperplane kernel



we will need to bring the data into 3-diamension space when using more real data when we are doing this, we need to use a kernel.

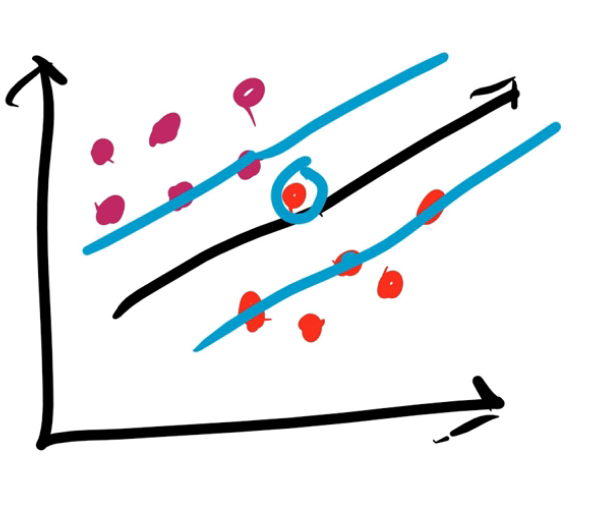
We when we are unable to use the hyperplane in 2-dimension space we need to convert it into 3-diamension space. If the 3-diamension space does not work, we would make it into 4-diamension space.

To find 3-diamension space you need to calculate x3 (3-dimension space ) :

s3-diamension space hyperplane with kernel

When it’s in 3-diamension space the line goes straight up to make a line and to find the margin.

**Soft margin**:

this is an image of a soft margin where you would allow a few points inside the margin to not effect the margin but would increases performance.

**Hard Margin**:

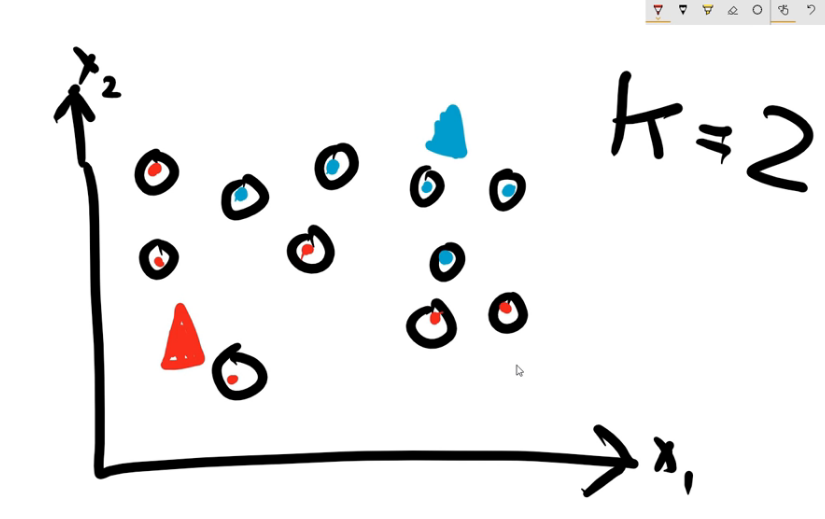
You can’t have any datapoint’s in the middle would influence performance.

K means clustering

Link to Clustering : <https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation>

Video time stamp: 02:15

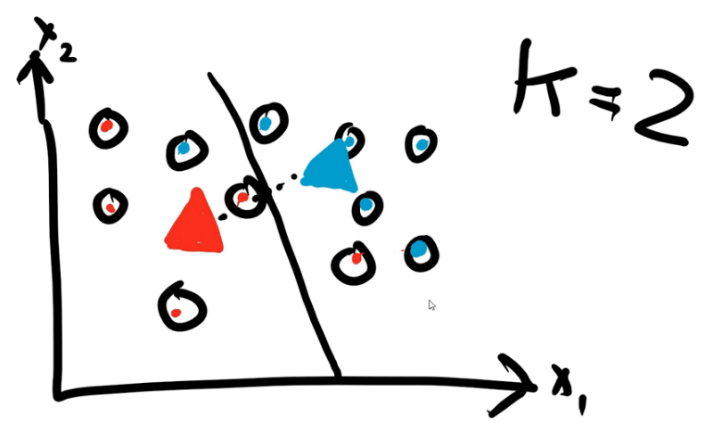
**unsupervised learning** is when you have data and the AI classifies unlabeled data. We are just giving the AI a bunch of features.

centroid’s = (trekanter)

How it works is K = amount of centroid’s that are placed randomly in the area of the datapoints

All the dots that are closes to the red centroid will become red dots and all that are closes the to blue will be blue.

After that you will need to find the center of all the red dots and put the red centroid in the middle same goes with the blue centroid.

You plus all the x value of all the dots together and divide them by the number of dots. this will give you the center location of X1. Do the same for x2 if you look on the image.

After that the image should look something like this after drawing a line between the centroids and throw that line is another line to see what goes on what (what dots are red & which are blue).

Then you just repeat everything over but with the new dots locations which would give a new location of the centroid if there is no chance you are done.

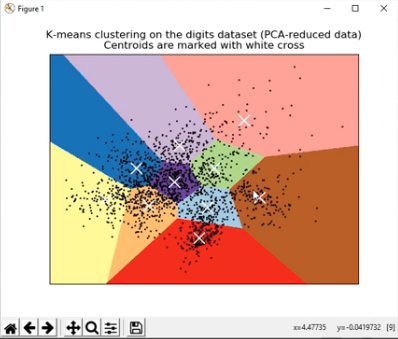
Good:

Bad:

Features = dots

If you have a lot of centroids it takes a while to calculate all the points since it’s like meaning the ai needs to calculate a lot if you have 700 features/ dots it also need to recalculate every time it does a initialization ( every time the centroid moves location ). But it is faster than other cluster algorithm’s.

Image Link : <https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html>

In the beginning all the centroid’s where placed randomly but after the AI have done some calculation all the dots are now centered where they should be. Look om the image how it can look when having 10 centroids.

Neutral network

Video time stamp: 02:27

Information: <https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05>

Weights

matrices

Weight are view as Matrices in math & programming

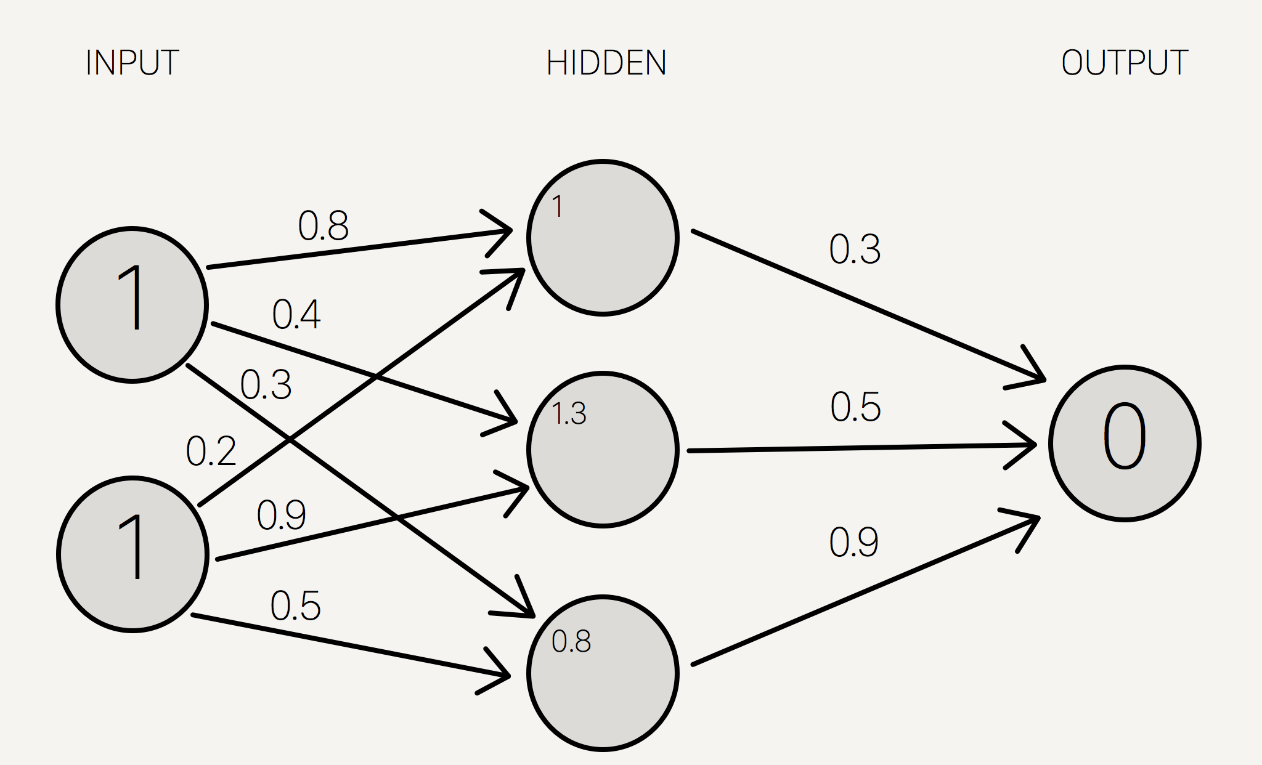
A matrix is an array way of calculating more one just one number, but you calculate arrays.

There are rows and columns in matrices fx a 3x3 is [1 2 3; 4 5 6; 7 8 10]. Each row is how many numbers there are in the columns.

Weight sum

To find weight sum you need to plus all the weights in the neural network

When having the weight and start training the neutral network will start adjusting the bias of the weights.



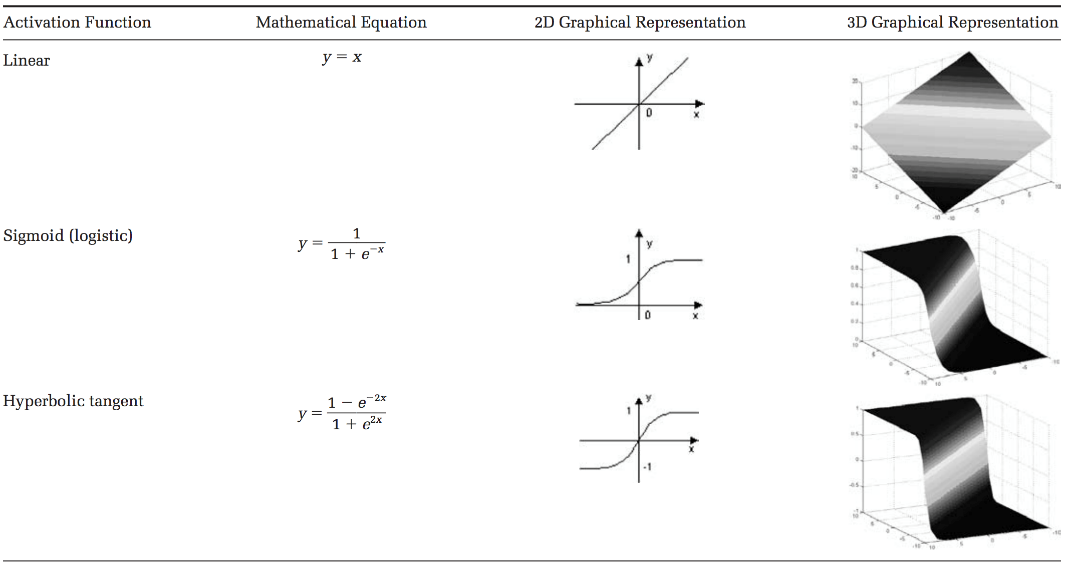
You calculate the input like this ( look on the image above)

The input goes In the hidden layer and in the hidden layer to use a function like the linear, sigmoid, hyperbolic tangent

1 \* 0.8 + 1 \* 0.2 = 1

1 \* 0.4 + 1 \* 0.9 = 1.3

1 \* 0.3 + 1 \* 0.5 = 0.8



In this case we are using the sigmoid logistic funstion.

**Bias**

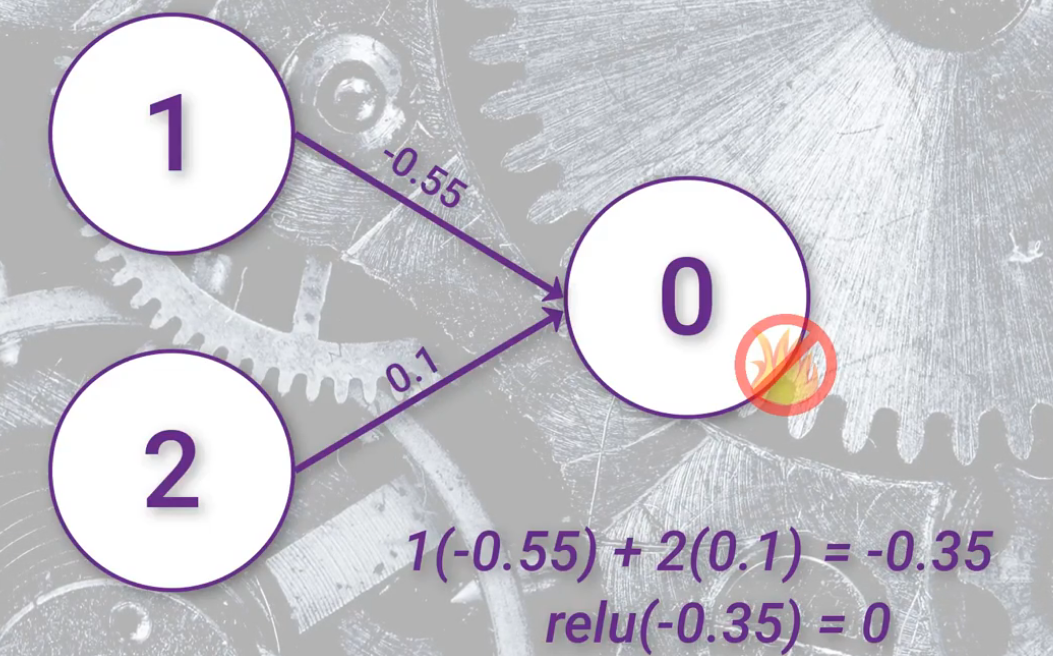
What is a bias

Earch neural have a different bias therm

Bias are learnable

Bias is what determent the ouput of the neuron ( bias will determen how much the neuron will fire )

Bias increases the flexibility of the model



Having 2 neuron’s in a 1 layr neural network this is with out the bias but only using the weight and you are able to see how its calculated. In this case 0 is the thresh hole for the weight sum to detement if a neuron is firing on not.

If we detement that the neuron should fire if its greater then or equal to 1 there we use bias.

The value we set our bias to is the oppisit of the thresh hole value.

( this is with out bias)

(this is with bias)

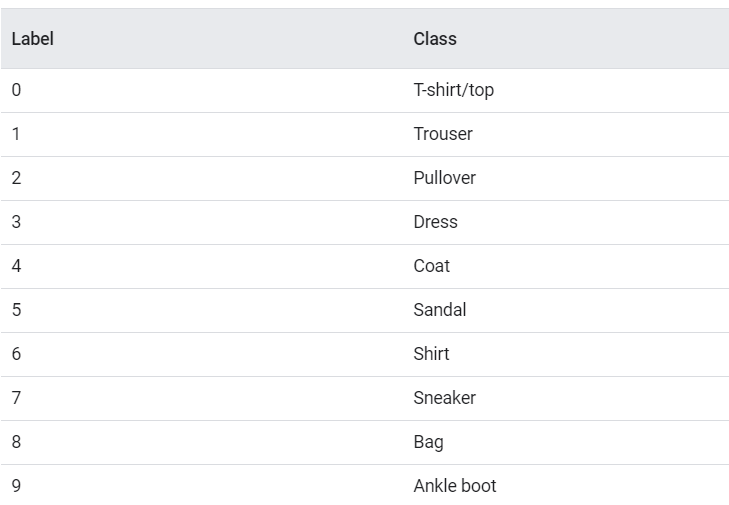
If we determent that the neuron should be fyring when its greather then or equel to 5 then our bias would be -5 just like with weight we don’t really controle the bias since it’s the neuron network that adjustest the bias & weight by it self.

Lost function (a way of calculating error’s)

The lost function will tell you how wrong your answer is (the output of the neuron). These lost functions are not a linear function. You are comparing the output to what it should be. Using hidden layer will increases the Accorsi of the neural network but will also make it more complex.

Little information about the code the data set is this image

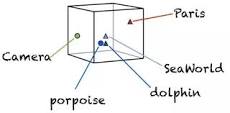
Link: <https://www.tensorflow.org/tutorials/keras/classification>

This image shows all the label types.

**Flatting data**

Normally the data is in array form as in [[1], [2], [3]] when flatting - > [1,2,3] this will increases the pressing of the data.

**Embedding**

embedding is a special term that simply means projecting an input into another more convenient representation space

fx: having 2 sentences which mean the same but differ in words. (each word will be giving a integer like how the bot will see it)

Sentences 1.

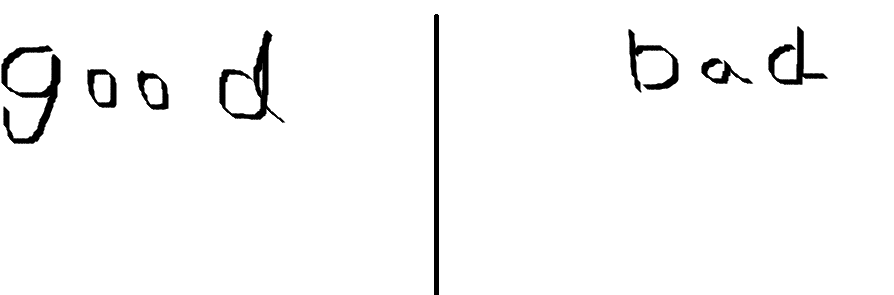
[0, 1, 2, 3 ]

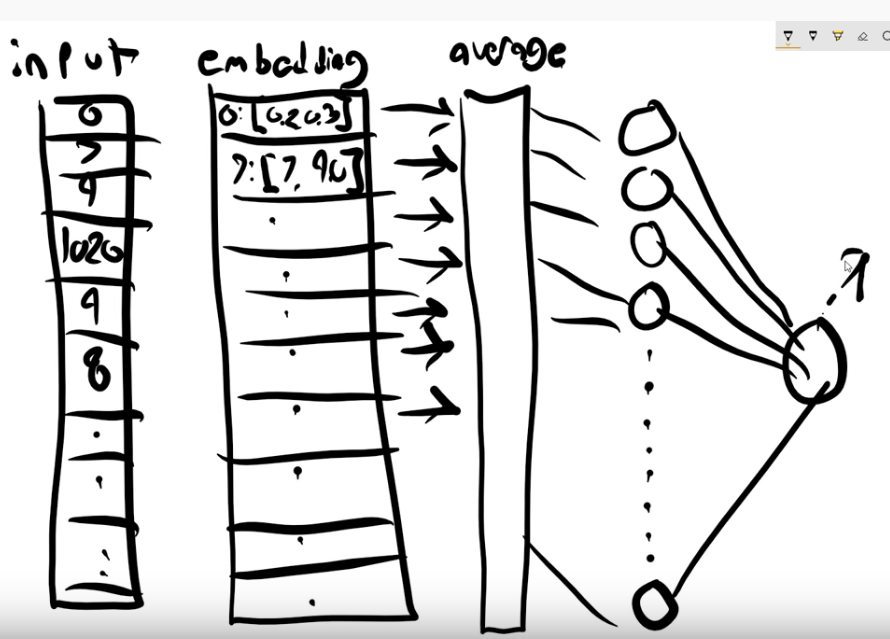
Have a great day

Sentence 2

[1, 2, 4, 3]

Have a good day

The bot can only read the words so it will only see [0, 1, 2, 3] in the first sentence and will see something different in the second sentence [0, 1, 2, 4, 3] and will give it another meaning. But embedding it you will put it in like a cluster of good and bad words and split it words it will put the words which have a good meaning more to the left side ( good side ) and the negative words more towards the bad side.

**how the network works**

takes the input and put it into the embedded layer and makes them into the vectors and there for they go into the average layer and then into our 16 neurons which will give an output 0 - 1.

**Epochs**

Number of times the algorithm runs

**Dense layer**

It is looking for patterns of words and try to classify them by a positive and a negative view.

**Chat bot**

**Intents**

Intents are bases on Padden’s and Reponses where the Padden is the input of the user and the response is the response of the AI. So, when the bot take in the input it will try to figure out what we are asking for.