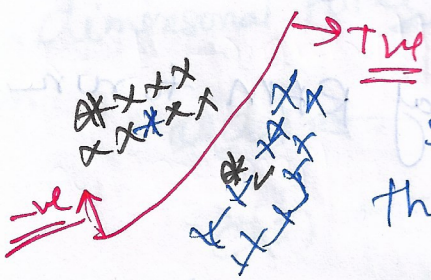
One of the downside of KNN it slow in calculating the test data. We will see some techniques little later

Decision Surface for KNN as k changes

k in KNN is a hyperparameter.

The curves that seprates +ve and -ve points from each other also Decision Surfaces.

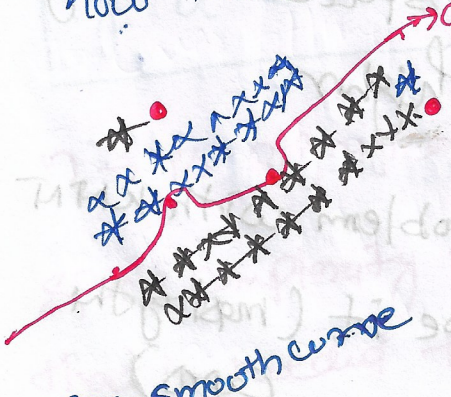


This is called as decission surfaces, ∵ these curve decide the +ve/-ve based on direction

3D → surfaces ND → hypersurfaces

Lets understand what happens as k ↑

how the decision surface behave:



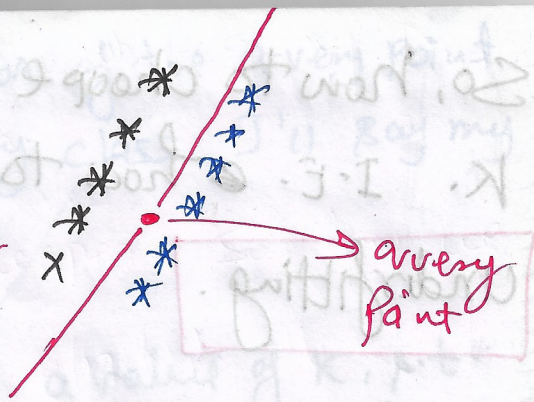
→ let K = 1

I've taken many query point and apply KNN

-seprate regions using a curve

Let $k=5$

Hence as the k ↑ my decision surface is smoother and smoother.



Let's take the extreem case $k=N$ i.e. number of points in dataset.

Suppose out of N points 60% are true and 40% are -ve.

If you think, for any avvery point this result will always remain same.

$n_1 = 60\%$, $n_2 = 40\%$. $n_1 + n_2 = N = k$

i.e. all the avvery points alway be true.

i.e. every new avvery is true.

Not good So, if $k=N$, whatever be the mazority class the result will remain same.

Hence $k=N$ not a good value of k.

* When $k=N$, every avvery point become the mazority class.

So, how to choose the correct value of K. I.E. how to avoid **overfitting** and **underfitting**.

Overfitting and Underfitting are very-very fundamental concept ~~in machine~~ machine learning

Let's understand these concept in more detail

$K=1$ → overfitting $K=N$ underfitting in our example. And $K=5$, well fit / Best fit

Say $K=1$, the point ~~that~~ those makes our surface non-smoother may be outliers. But the decision surface did overfitting to accomodate this.

Overfitting ⇒ The classifier tries to fix a function or surface, tries to make no mistake, which makes the decision surface more non-smoother such case overfitting.

Underfitting : Don't care of the avery point, whatever is the majority class I'll say my data is same.

So, we have to choose a value of K, s.t. the ML model should not be either a overfitting or underfitting solution.

It should be a well balanced system. As in our examp K=5.

~~How to determine K~~

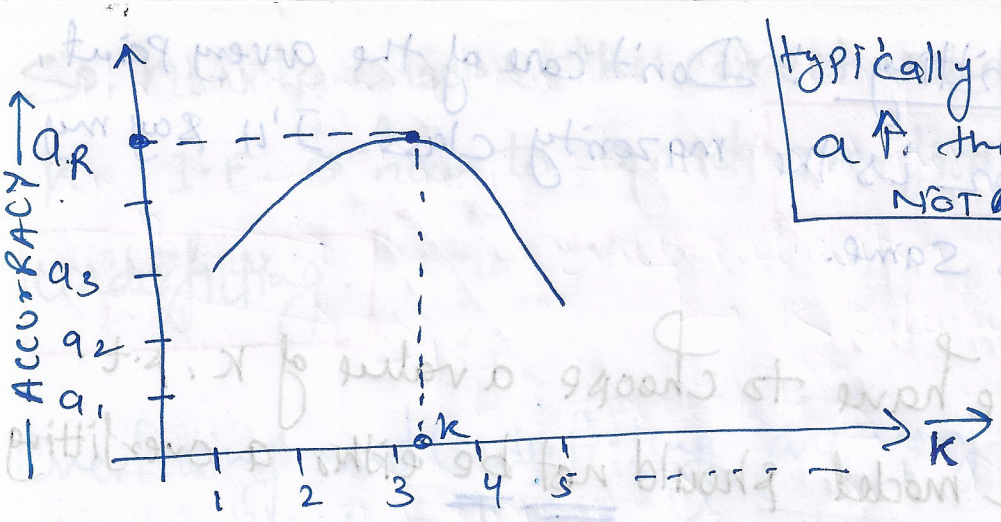
How to determine K (Need of cross Validation)

D_n → D_{train} (70%) } we choose this randomly.
→ D_{test} (30%) }

Now let's follow the below method to choose the value of K.

	D_{train}	Calculate Accuracy D_{test}
K = 1	D_{train}	q_1
K = 2	"	q_2
K = 3	"	q_3
⋮	⋮	q_n
		⋮

Accuracy = $\frac{\#\text{ of correctly classified points}}{\text{total \# points}}$



typically as k ↑ a ↑. the start NOT ALWAYS

but at k= 4 I'm getting heighest accu

∴ we will choose $\boxed{k=4}$

Lets talk about in terms of Amazon Food Review Ds. Lets take an hypothetical number for accuracy = 96%

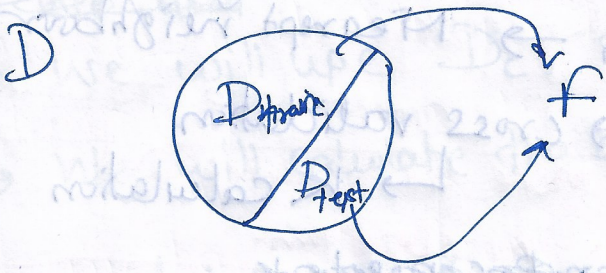
Using D_{train} and 4NN, on amazon review dataset, we get an acc of 96%

Hmmm- But there is a small problem with this statement

but see what

What As we dipcussed in very begning the objective of ML $\boxed{\text{Data pet}} \longrightarrow f_n$

But here, we broke the data into 2 parts



96-1:

~~What about~~ If you think, we also use D_{Test} to calculate the value of K RT?

So, we use all the Data D_N to get the value of K.

How this ML model behave for a future unseen point? what would be my accuracy.

~~So, for~~ That we don't know.

And for any ML model it has to work very well for future UNSEEN point.

Otherwise it is of no importance/useless.

An algorithm which does very well on future unseen point is k/as Generatization.

So, our ML model should be Generalized

Let's take an alternative strategy:

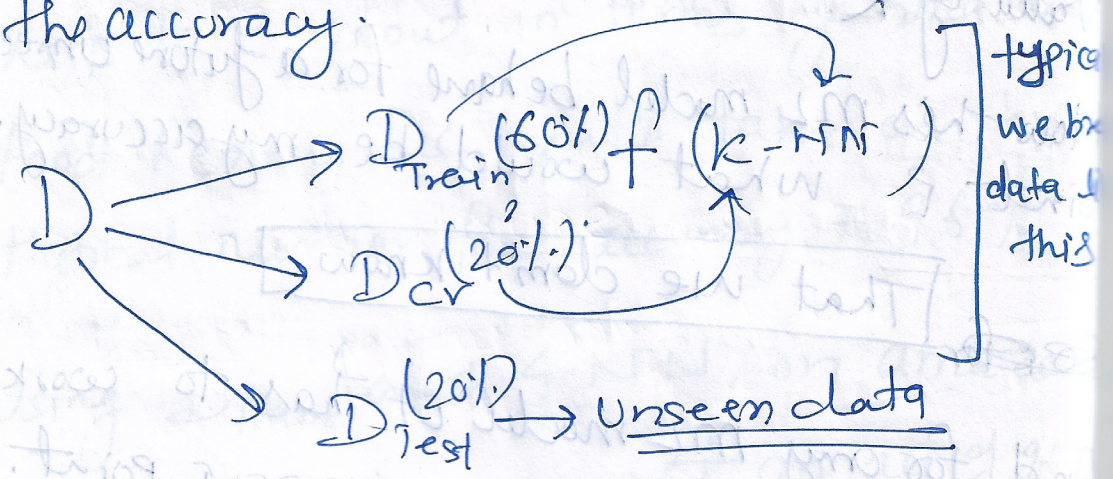
$D_n \rightarrow D_{train}$ → Nearest neighbour

$D_n \rightarrow D_{cv}$ → cross validation → K calculation

$D_n \rightarrow D_{Test}$

~~Decision trees~~

So, we are using D_{train} and D_{cv} to compute my function. We will apply this function on D_{Test} to calculate the accuracy.



Say if my acc = 93%. Can I say, my ~~acc~~ algorithm has an accuracy of 93% on unseen data?

We can. ~~But there is a small~~

This is something known as Cross Validati

→ We will train our model uping D_{train}.

~~→ date add~~

→ We will upe D_{cv} to calculate K

→ We will calculate the accuracy on Test Data.

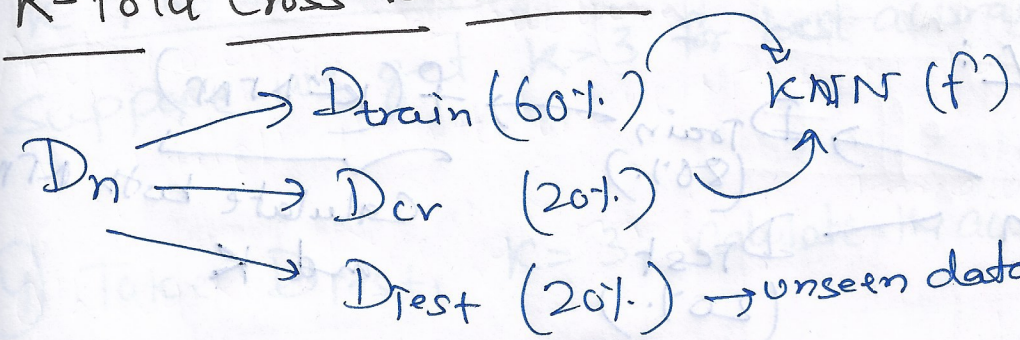
Statement: Using D_{train} as training data, I find 6-NN to have an acc of 93% on future/unseen data.

93% → Generalize Accuracy.

~~There This~~ or 100-93 = 7% Generalized Error

This is small introduction of **CROSS VALIDATION**

K-fold Cross-validation



→ There is a problem with this model.

Problem: We are uping only 60% of data to get our Nearest Neighbour (NN)

20%. → CV and 20%. → Test.
(K)

We can't do anything with 20% test

∴ we need ~~some~~ 20%. unseen data to calc accuracy.

> ML Rule: More the training Data better is the algorithm.

So can we do one thing combine this

60%. NN + 20 CV → to compute NN

→ More the training data, better is your algorithm, this is always true for any ML Algorithm.

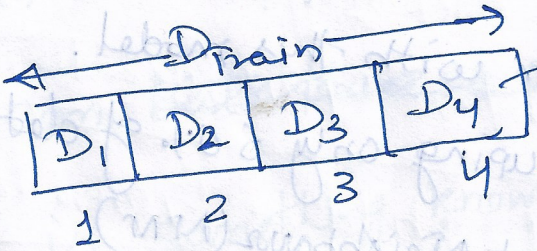
→ So in order to use 60%. + 20%. data ~~is known~~ we are going to use K-fold

CV:

① D_n → D_{train} (80%.) → f(K-NN)
D_n → D_{test} (20%.)

Calculate both NN and K

② D_{train} → randomly → Break it into 4 equal size parts



(3)

	train	cv
k=1	$D_1 D_2 D_3$	$D_4 = a_4$
k=1	$D_1 D_2 D_4$	$D_3 = a_3$
k=1	$D_1 D_3 D_4$	$D_2 = a_2$
k=1	$D_2 D_3 D_4$	$D_1 = a_1$
k=2		
⋮		

Remember all D_1, D_2, D_3 and D_4 20% of the Data

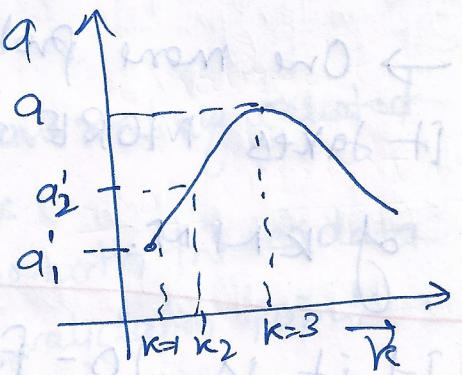
Same 20% CV

→ Perform the o/p for k=1, k=2, k=3 - - - -

$$k=1 \text{ accuray} = \frac{a_1 + a_2 + a_3 + a_4}{4} = a_1'$$

→ So, I've use every part of my data for training and cross validation.

This is called as 4 fold cross validation.



k-flod validation

Suppose we get k=3 for best accuracy.

(4) Take Dtest, k=3, Calulate the accuray e.g. 90%.

90% is the accuray of the above modul having k=3

Now the question again remain is what is the right value of K for K-fold.

Here we have K=4 or K=4, K=10, K=100 is the right #

→ typically for many m/c learning model we use 10-fold CV.

→ One more problem with k-fold cross valid It takes MORE time to calculate the K of KNN.

If it is 10-fold, the time ↑ by 10 time

But the good thing is that It is one time effort.