Jan 17th 2016

How do we validate ML algorithms?

Typical setting
1. Data given (i.e. tables)
2. choose & train a method
3. Test it!
4. if good enough, release it

How do we make sure of this?

do this by Cross Validation

Cross Validation

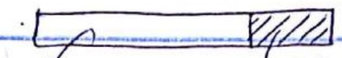
(hold out method)

(rotation estimation)

Given entire data $\boxed{}$

$\underbrace{\underbrace{\text{input/features}}_{\text{(unsupervised)}} + \text{targets}}_{\text{(supervised)}}$

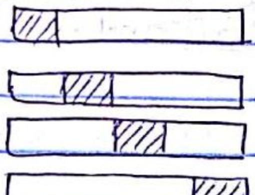
① hold out : keep one part for testing



We hold out a particular part of data for testing & use the rest of for training.
This is not reliable : the part you take may be biased (lucky / unfortunate)

Usually ② & ③ used

② Cross Validation.

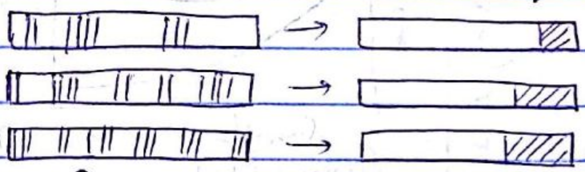


4 fold cross validation
"k-fold" cross validation.

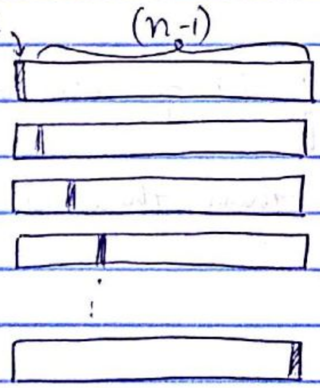
(smaller part) use the ▨ part for testing and ▭ part for training

Usually between 3-5 fold.

③ Random Sampling.
We do not slice as in cross validation. (we do random sampling)
(same result as in cross validation)



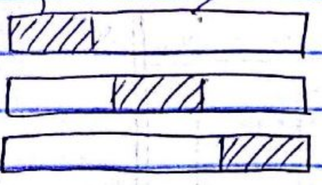
④ Leave-one-out Validation



- we use n-1 for training and only one for testing... repeat this "n" numbers of times.
- very exhaustive. (very)
- done when there is (very) small data set

K-fold cross validation

(training size ▭ is always > testing ▨)



error (testing) = e_1
error (testing) = e_2
error (testing) = e_3

$$E = \frac{1}{B} \sum e_i$$

average error and standard deviation (very imp)

error of 10% ± ~~3%~~ 2% is ~~worse~~ better than
error of 4% ± 35%

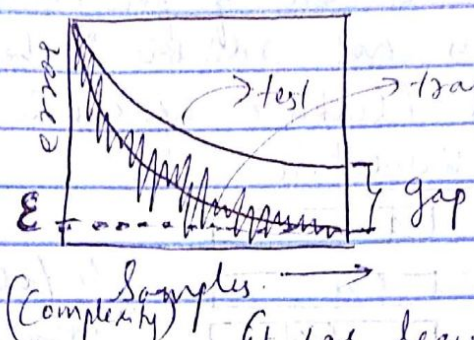
is it good enough to release? how do we test?

e.g. $e = [10\% \;\; 22\%]$ (confidence interval) with confidence 0.05 ⇒ 95%

95% of the times, the error generated is between 10% to 22%

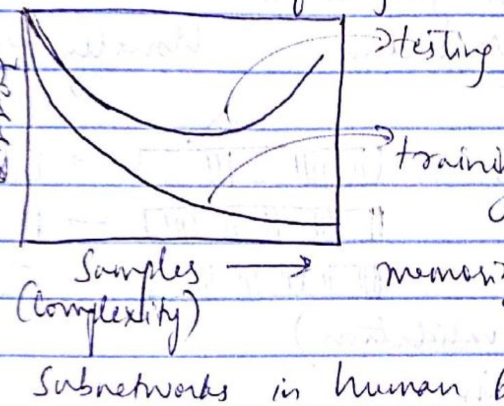
Usually:

$\mathcal{E}$ = acceptable error.



if gap is small, the training is done properly, (it has learnt properly)

this shows overfitting:

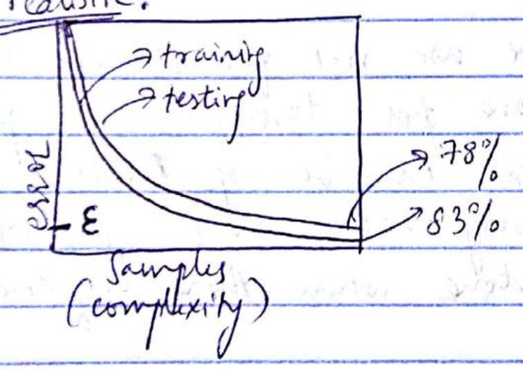


(complexity → neural n/w with 30 layers or 3 layers?)

memorized data, not learnt.

Subnetworks in human brain change their configuration

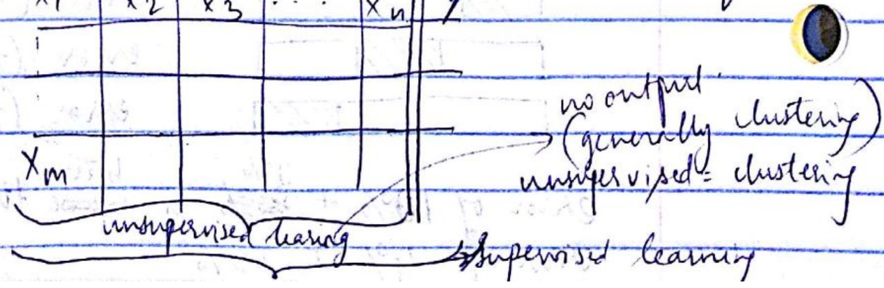
realistic:



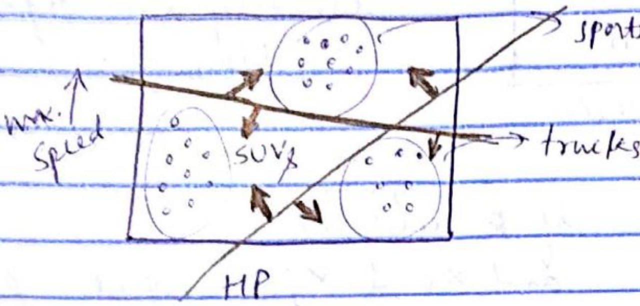
if training accuracy is < testing accuracy then there is a bug

Clustering

Grouping given "unlabelled" data into "clusters" of similar objects.



Eg horse power and maximum speed.



- Nicely separated (space between clusters)
- linearly separated. (we can draw lines, by just 2 lines, we have separated them. we need not know the actual contour

lines in feature space acting as decision boundaries

lines in hyper dimensional space → hyperplanes.

separation → distinction → understanding

finding hyperplanes to separate clusters means we understand the meaning of features.

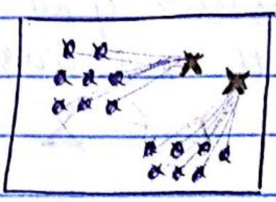
Clustering
- we need to know the # of clusters
- don't ~~need~~ to ~~know~~ know
  - (hierarchical clustering)
  - (density based → finds dense regions in feature space

K-Means algorithm.

finds (k) centroids of data.
→ has to be told how many.

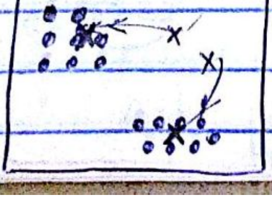
① Randomly place k centroids.

② Assign each data point to a centroid.



[Similarity measure = L2 (Euclidean)]

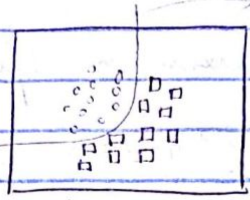
③ Update the ~~centroid~~ centroids.



update the centroid!

Centroid = prototype of the cluster.
= average of all data points that belong to that centroid.

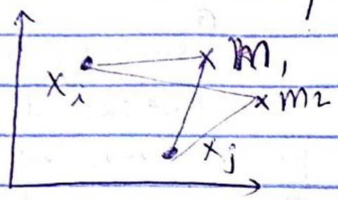
if we have.



curve, not a line!

linearly not separable.
∴ difficult.

this is complicated.



if k = centroid (m_1 & m_2)
and x_i & x_j are data points:

$$d(i, j) = \|x_i - x_j\|$$
$$= \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots}$$

Error:

$$E = \sum_{k=1}^{K} \sum_{x = X} \|x - m_k\|_{L2}$$

(minimize this) eucl dist between all data points & centres.

trail of centroids: