Cross Validation

- How to find appropriate k value for k-NNC.
- Some improvements to k-NNC

Cross Validation

- *r-fold cross validation:*
  1. Partition the training set into r blocks. Let these are $D_1, D_2, \ldots, D_r$.
  2. For i = 1 to r do
     1. Consider $D - D_i$ as the training set and D_i as the validation set.
     2. For a range of k values (say from 1 to m) find the error rates on the validation set.
     3. Let these error rates are $e_{i1}, e_{i2}, \ldots, e_{im}$
  3. Take e_i = mean of $\{e_{1i}, e_{2i}, \ldots, e_{ri}\}$, for i = 1 to m.
  4. k value = $\underset{j}{\operatorname{argmin}} \{e_1, e_2, \ldots, e_j, \ldots, e_m\}$

Cross validation

- One should not use *the test set* to decide the value of k.
- Test set should be used only after fixing k, to get the final *error-rate* for the classifier.
- Cross validation is only to fix the value of parameters like k . So the error rates on validation sets should be called *validation error rates*.

k-NNC

- Since k-NNC is a simple classifier, it attracted many researchers.
- First k-NNC is invented in 1952. Since then people tried to improve it in various ways.
- k-NNC is often not seen as a related classifier to Bayes classifier. This is because, distributions are not explicitly calculated.

An improvement of k-NNC

- k-NNC gives equal importance to the first NN and to the last NN.
- *S.A. Dudani (1976)* has given a method where we give *weights* to the NNs.
- Voting is done according to these weights.
- Let the distances (with given pattern) of k NNs be an ordered set = { d_1, d_2, …, d_k}
- For i th NN the weight is, $w_i = (d_k - d_i)/(d_k - d_1)$
- Use these weights as vote values and classify accordingly.
- This is called *modified k-NNC* or *weighted k-NNC,* and is found to improve the performance in almost all cases.

Bootstrapping

- Another promising improvement is to regenerate the training set, so that the training patterns belonging to different classes are separated well.
- *Hamamoto(1997)* proposed the following:
- For each training pattern y do:
  1. Find r NNs of y in the training set that belongs to the same class as y.
  2. Find the mean of these r NNs. Let this is y_r
  3. Replace y by y_r

Other improvements

- Let *n* be the number of training patterns.
- Let *k* be a small constant when compared with *n*
- The time and space complexity of k-NNC are both equal to *O(n).*
- To reduce the computational burden of k-NNC is another important direction of research.
  - Prototype selection.
  - Not all training patterns are important for k-NNC, so remove those which are unimportant.

Condensed k-NNC

- *Cover and Hart (1967)* gave the following procedure to reduce the training set size.
- The new reduced training set is called the *condensed training set.*
  - Start with empty condensed set.
  - Let *x* be a training pattern.
  - Classify *x* using condensed set as the training set.
  - If *x* is misclassified then add *x* to the condensed set.
  - Repeatedly do this for all training patterns until no more changes to the condensed set.
- Training error using only the condensed set is zero.

Some other ways …

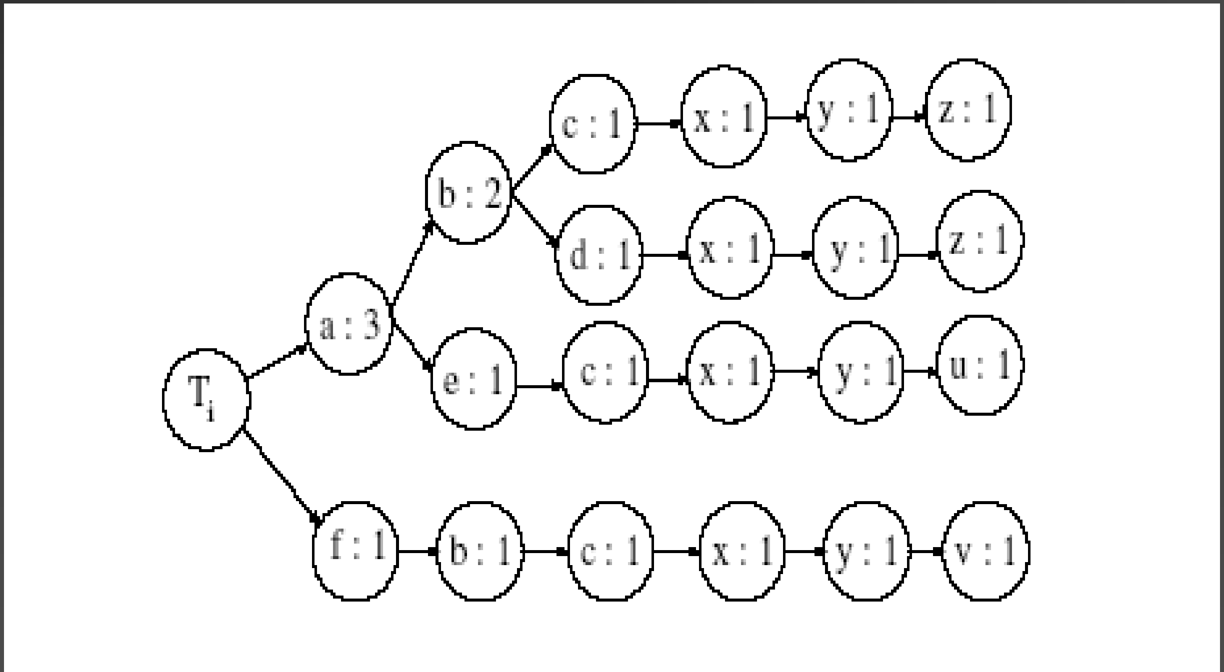
- Building an *index* over the training set can reduce the search time.
- Similar to *B+ trees* there are indexing methods for high dimensional data like *R-tree index*, *KD tree index*, etc.

Some more …

- To reduce the space, a compact representation for the training set can be found.
- *Anantanarayana (2001)* gave a compact representation called PC-tree. And also he gave nearest neighbor search method directly over PC-tree representations. So actually the search time also is reduced.

PC-tree

Let the training patterns for a class are: $(a,b,c,x,y,z)^t$, $(a,b,c,x,y,z)^t$, $(a,e,c,x,y,u)^t$, $(f,b,c,x,y,v)^t$



You too can …

- It is easy to find some other ways to improve on k-NNC.
- What one can do:
  1. Do a closer study of existing improvements.
  2. Propose your improvement.
  3. Experiment on some datasets and hence show it really improves.
- One can closely study k-NNC along with SVM or MLP and find ways to speedup SVM or MLP learning.

Next class

- We will go back to Bayes classifiers.