COMP90049 Knowledge Technologies

Introduction Classification (Lecture Set4) 2017

Rao Kotagiri
School of Computing and Information Systems
The Melbourne School of Engineering

Some of slides are derived from Prof Vipin Kumar and modified, http://www-users.cs.umn.edu/~kumar/

What is classification?

Given a collection of training data

- Each item of the data contains a set of *attributes (features)*, at least one of the attributes is the *class*.

The goal is to build a *model* for class attributes (dependent variables) as a function of the values of other attributes (independent or decision variables).

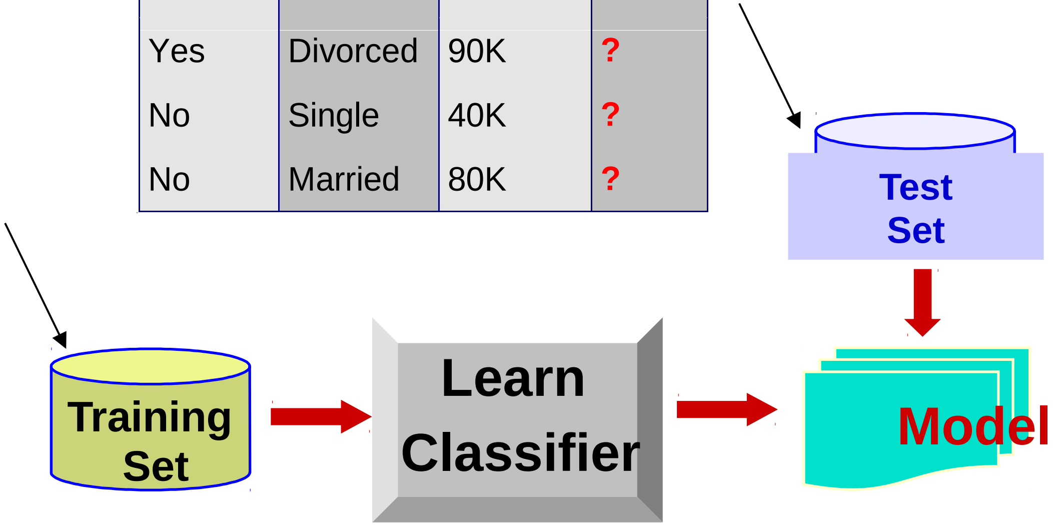
The discovered model (function) is used to predict the label of a <u>previously unseen</u> item.

- For determining how good the discovered model is a *test set* is used. Usually, the given data set is partitioned into two disjoint training and test sets. The training set is used to build the model and the test set is used to validate it by for example the % of the time the class label is correctly discovered.

Classification Example

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Refund	**Marital Status**	**Taxable Income**	**Cheat**
No	Single	75K	**?**
Yes	Married	50K	**?**
No	Married	150K	**?**
Yes	Divorced	90K	**?**
No	Single	40K	**?**
No	Married	80K	**?**



- Given
  - A set of training tuples and their associated class labels
  - Each tuple X is represented by an n dimensional attribute vector X = (x_1, x_2, …, x_n)
  - There are *K* classes C_1, C_2, …, C_K.

Given a tuple X, predict which class X belongs to?

- Naïve Bayesian classifier
  - For each class C_i, estimate the probability $p(C_i|X)$:

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})} \qquad posteriori = \frac{likelihood \times prior}{evidence}$$

  - Predicts X belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *K* classes
  - Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$ needs to be maximized

  - Naïve assumption: attributes are conditionally independent (i.e., no dependence relation between attributes given the condition):

$$\boxed{p(c_{i_|}|x) = p(c_i) * \prod_{k=1}^{n} p(x_k|c_i)}$$

Given a training data set, what are the probabilities we need to estimate?

Naïve Bayesian Classifier : An example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Ann comes to the clinic with severe headache, no soreness, normal temperature and with cough. What does she has? Choose the case with highest probability.

P(Flu| Headache = severe, Sore = no, Temperature = normal, Cough = yes)

≅ P(Flu)*P(Headache = severe|Flu)*P(Sore = no|Flu)*P(Temperature = normal |Flu)*P(Cough = yes|Flu)

P(Cold| Headache = severe, Sore = no, Temperature = normal, Cough = yes)

≅ P(Cold)*P(Headache = severe|Cold)*P(Sore = no|Cold)*P(Temperature = normal |Cold)*P(Cough = yes |Cold)

Naïve Bayesian Classifier: An example

We need labelled data to build a classifier

Headache	Cough	Temperature	Sore	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

We need to estimate probabilities from the data we have.

Naïve Bayesian Classifier

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

P(FLU) = 3/5	P(Cold) = 2/5
P(Headache = severe | Flu) = 2/3	P(Headache = severe | Cold) = 0/2
P(Headache = mild | Flu) = 1/3	P(Headache = mild | Cold) = 1/2
P(Headache = no | Flu) = 0/3	P(Headache = no | Cold) =1/2
P(Sore = severe | Flu) = 1/3	P(Sore = severe | Cold) = 1/2
P(Sore = mild | Flu) = 2/3	P(Sore = mild | Cold) = 0/2
P(Sore = no | Flu) = 0/3	P(Sore = no | Cold) = 1/2

Naïve Bayesian Classifier

Headache	Cough	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

P(FLU) = 3/5	P(Cold) = 2/5
P(Temperature = High | Flu) = 1/3	P(Temperature = High | Cold) = 0/2
P(Temperature = Normal | Flu) = 2/3	P(Temperature = Normal | Cold) = 2/2
P(Cough = yes | Flu) = 3/3	P(Cough = yes | Cold) = 1/2
P(Cough = no | Flu) = 0/3	P(Cough = no | Cold) = 1/2

P(FLU) = 3/5	P(Cold) = 2/5
P(Headache = severe | Flu) = 2/3	P(Headache = severe | Cold) = 0/2 ~= e
P(Headache = mild | Flu) = 1/3	P(Headache = mild | Cold) = 1/2
P(Headache = no | Flu) = 0/3 ~= e	P(Headache = no | Cold) =1/2
P(Sore = severe | Flu) = 1/3	P(Sore = severe | Cold) = 1/2
P(Sore = mild | Flu) = 2/3	P(Sore = mild | Cold) = 0/2 ~ e
P(Sore = no | Flu) = 0/3 ~ e	P(Sore = no | Cold) = 1/2
P(FLU) = 3/5	P(Cold) = 2/5
P(Temperature = High | Flu) = 1/3	P(Temperature = High | Cold) = 0/2 ~e
P(Temperature = Normal | Flu) = 2/3	P(Temperature = Normal | Cold) = 2/2
P(Cough = yes | Flu) = 3/3	P(Cough = yes | Cold) = 1/2
P(Cough = no | Flu) = 0/3 ~=e	P(Cough = no | Cold) = 1/2

e= small value = 10^{-7} (one can use e to be less than 1/n where n is the number of training instances)

P(Flu| Headache = severe, Sore = no, Temperature = normal, Cough = yes)

= P(Flu)*P(Headache = severe|Flu)*P(Sore = no|Flu)*P(Temperature = normal |Flu)*P(Cough = yes|Flu)

= 3/5 x 2/3 x e x 2/3 x 3/3 = 0.26e

P(Cold| Headache = severe, Sore = no, Temperature = normal, Cough = yes)

~ P(Cold)*P(Headache = severe|Cold)*P(Sore = no|Cold)*P(Temperature = normal |Cold)*P(Cough = yes |Cold)

= 2/5 x e x ½ x 1 x ½ = 0.1e

=> Diagnosis is Flu

Naïve Bayesian Classifier

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	cold
severe	severe	normal	yes	Flu

P(Flu| Headache = severe, Sore = no, Temperature = normal, Cough = yes) = ?

P(Cold| Headache = severe, Sore = no, Temperature = normal, Cough = yes) = ?

Abbreviations:

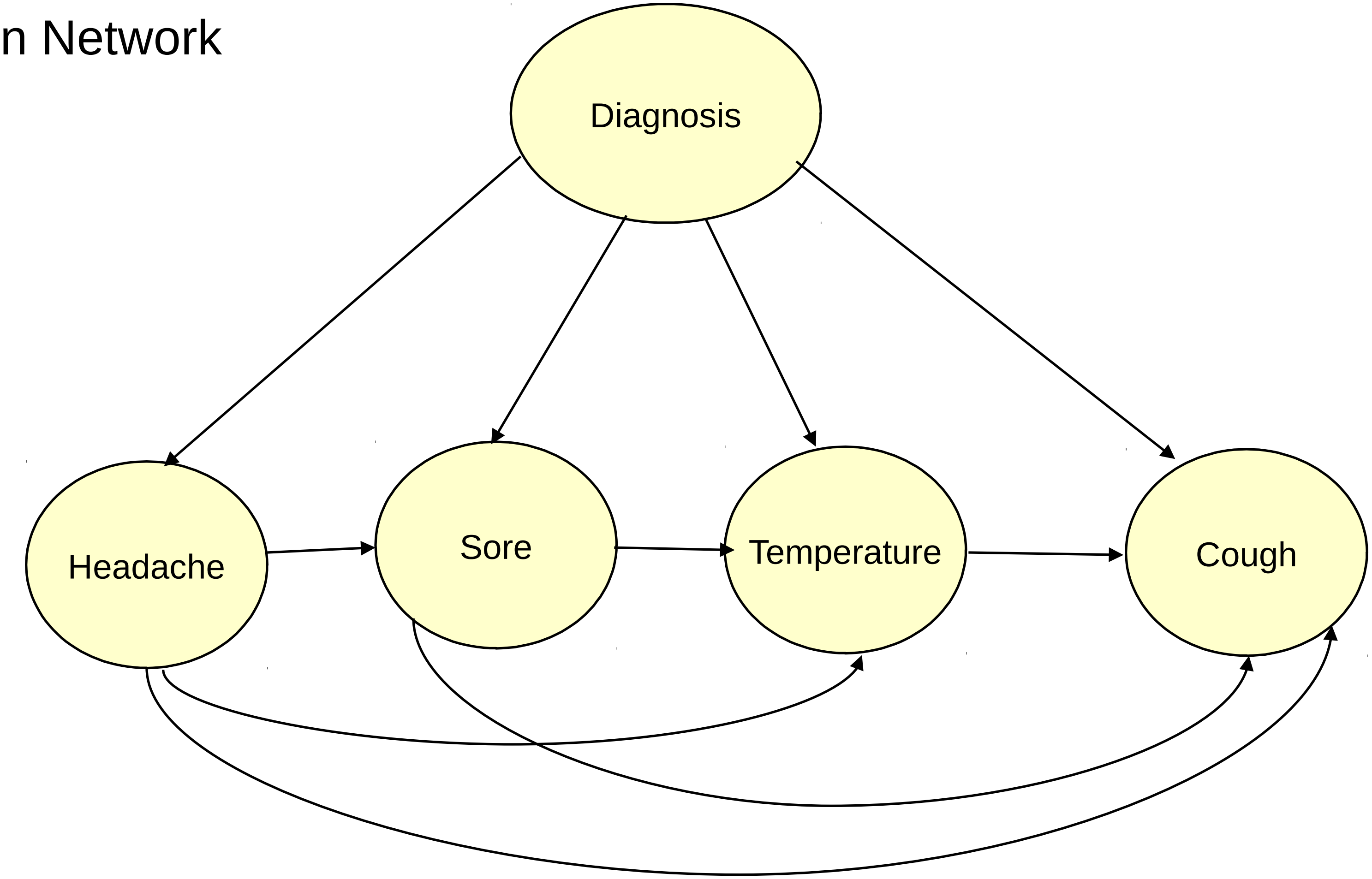
F = Flu, C = Cold, H = Headache, S = Sore, T = Temperature, Cou = Cough

se = severe, mi = mild, nor = normal, hi = high, ye = yes, no= no

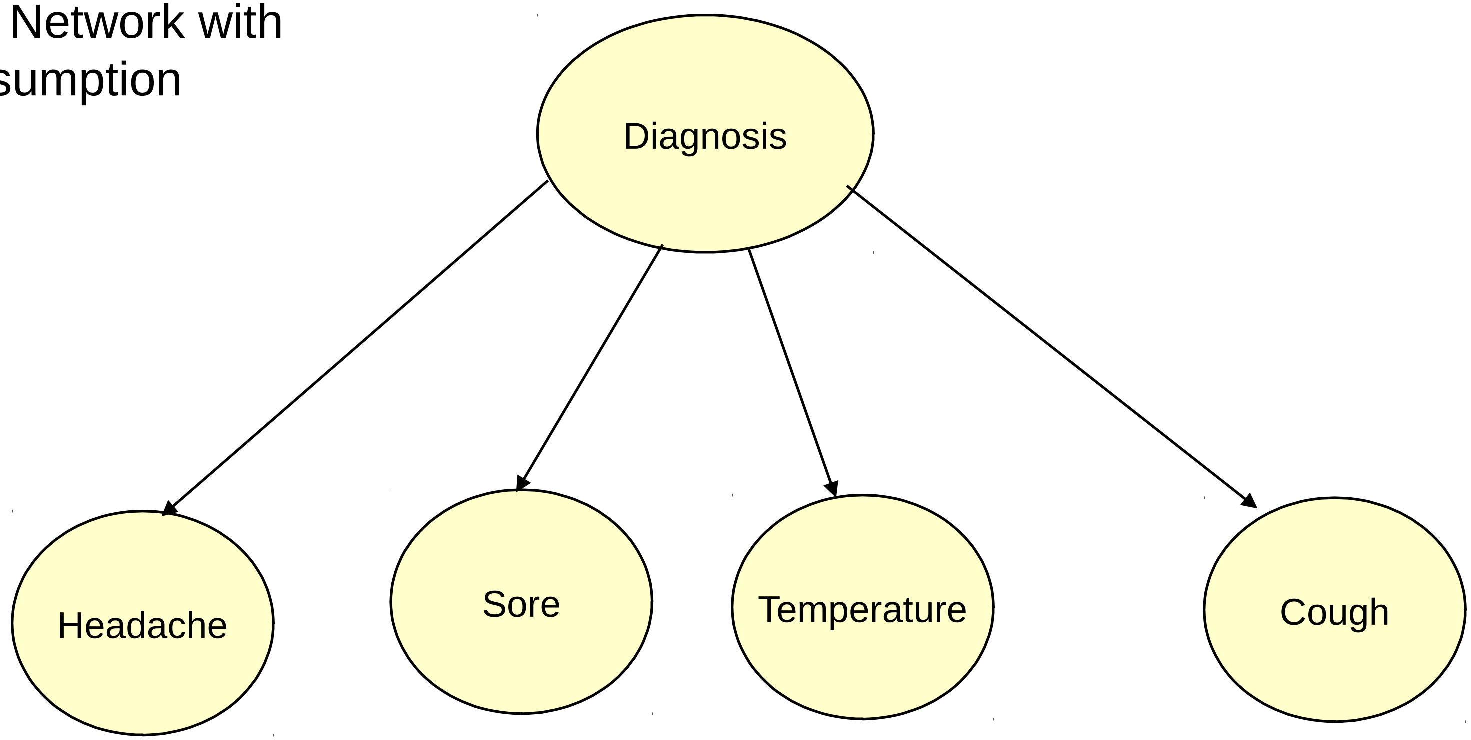
P(F| H = se, S= no, T = nor, Cou= ye)

= P(F, H = se, S= no, T = nor, Cou= ye)/P(H = se, S= no, T = nor, Cou= ye)
= P(F)P(H = se |F) P(S= no |F) P(T = nor | F) P(C= ye | F)/ P(H = se, S= no, T = nor, Cou= ye)
We assumes H, T, S and Cou are conditionally independent (naïve assumption) when the individual is suffering from Flue or Cold

Bayesian Network



Bayesian Network with Naïve assumption



Naïve Bayesian Classifier

$$p(c_{i_|}|x) = p(c_i) * \prod_{k=1}^{n} p(x_k|c_i)$$

Conclusions

- Naïve Bayesian (NB) Classifier is very simple to build, extremely fast to make decisions, and easy to update the probabilities when the new data becomes available.
- Works well in many application areas.
- Scales easily for large number of dimensions (100s) and data sizes.
- Easy to explain the reason for the decision made.
- One should apply NB first before launching into more sophisticated classification techniques.

$$\text{Storage required is } O(C - 1 + C\sum_{i=1}^{D}(Di - 1))$$

$$= O(C - 1 - CD + C\sum_{i=1}^{D} Di)$$

$$= O(C(\sum_{i=1}^{D} Di - D))$$

$where$

$D = number \quad of \quad \mathrm{dim}ensions$

$Di = ith \quad domain \quad size$

$C = number \quad of \quad classes$

How to evaluate Classifiers?

For two class problem:

There are Positive (P) cases and Negative (N) cases.

A classifier may classify a Positive instance as Positive (this case is called True Positives, TP) or as Negative (False Negatives, FN).

Similarly a Negative instance can be classified as Negative instance (this case is False Positive, FP) or as Negative (this case is True Negative, TN)

		Actual	
		P	N
Predicted	P	TP	FP
	N	FN	TN

How to evaluate Classifiers?

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

Accuracy with respect to positive cases also called true positive rate

$$Specificity = \frac{TN}{TN + FP}$$

Accuracy with respect to negative cases

$$\mathrm{Re}\,call = \frac{TP}{TP + FN}$$

$$\mathrm{Pr}\,ecision = \frac{TP}{TP + FP}$$

$$F1\_Score = \frac{2\,\mathrm{Re}\,call * \mathrm{Pr}\,escision}{\mathrm{Re}\,call + precision}$$

		Actual	
		P	N
Predicted	P	TP	FP
	N	FN	TN

False negative rate = F/(TP+FN)

How to evaluate Classifiers?

		Actual	
		P	N
Predicted	P	TP	FP
	N	FN	TN

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

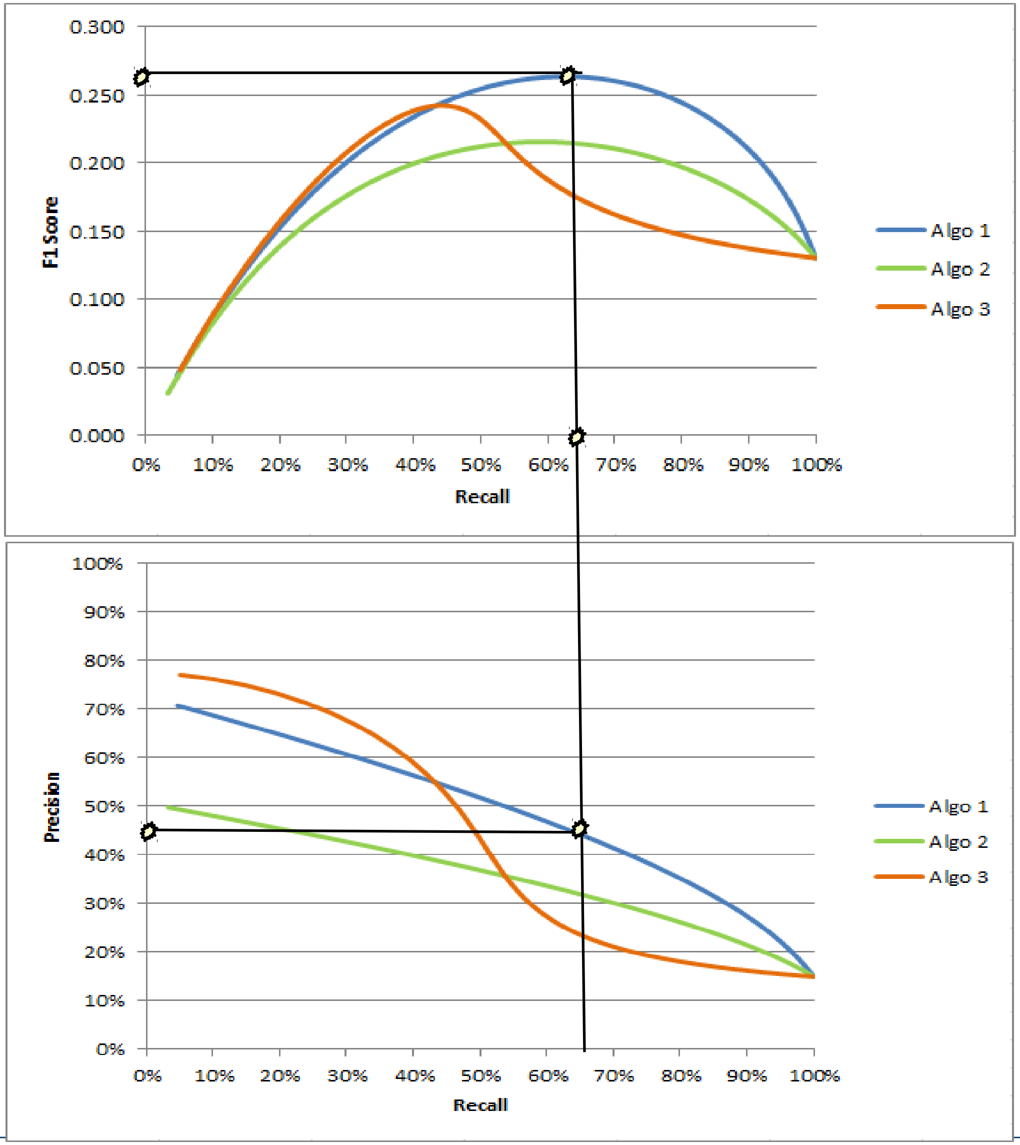
$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$\mathrm{Re}\,call = \frac{TP}{TP + FN}$$

$$\mathrm{Pr}\,ecision = \frac{TP}{TP + FP}$$

$$F1\_Score = \frac{2\,\mathrm{Re}\,call * \mathrm{Pr}\,escision}{\mathrm{Re}\,call + precision}$$



Leave-One-Out

Let us assume we have N data points for which we know the labels. We chose each data point as test case and the rest as training data.

This means we have to train the system N times and the average performance is computed.

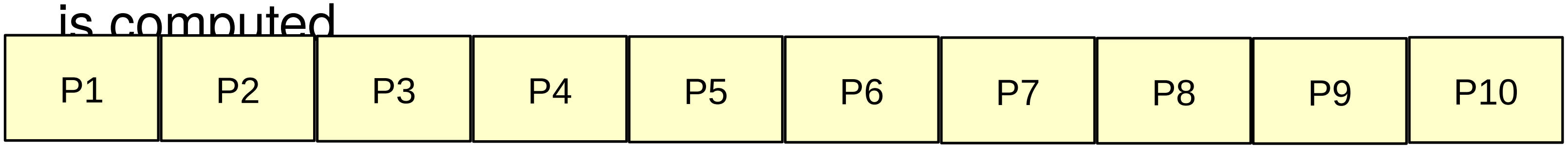
Good points: There is no sampling bias in evaluating the system and the results will be unique and repeatable for given method. The method also generally gives higher accuracy values as all N -1 points are used in training. (We are assuming more data points means a more accurate classifier can be built – this may not be always be true with certain data.)

Bad point: It is infeasible if we have large data set and the training is itself very expensive.

10 Fold cross validation.

Let us assume we have N data points for which we know the labels. We partition the data into 10 (approximately) equal size partitions. We choose each partition for testing and the remaining 9 partitions for training.

This means we have to train the system 10 times and the average performance is computed



P1	P2	P3	P4	P5	P6	P7	P8	P9	P10

Train with {P2,P3,…,P10} and test with {P1}

Train with {P1,P3,…,P10} and test with {P2}

Train with {P1,P2,P4,…,P10} and test with {P3}

Train with {P1,P2…,P3,P5,…,P10} and test with {P4}

Train with {P1,P2…,P4,P6,…,P10} and test with {P5}

…

Train with {P1,P2…,P8,P10} and test with {P9}

Train with {P1,P2…,P9,} and test with {P10}

Evaluation schemes

10 Fold cross validation.

Good points: We need to train the system only 10 times unlike Leave-One-Out which requires training N times.

Bad Points: There can be a bias in evaluating the system due to sampling (the way we do the partitioning), that is how data is distributed among the 10 partitions. The results will not be unique unless we always partition the data identically. One solution is repeat the 10 Fold Cross Validation by randomly shuffling the data say by 5 or more times. The results will give slightly lower accuracy values as only 90% data is used for training. For small data sets it is not always possible to partition the data properly such that each partition represents the data IID (Identically Independently Distributed).

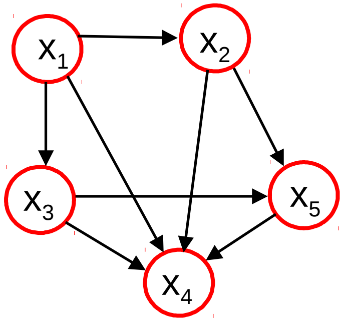
Exercise 1

Income	Student?	Credit rating	Buy computer
High	No	Fair	No
High	No	Excellent	No
High	No	Fair	Yes
Medium	No	Fair	Yes
low	yes	Excellent	No
Low	Yes	Excellent	Yes
Medium	No	Fair	No

- What are the probabilities we need to estimate in a Naïve Bayesian classifier?
- Will a student with high income, excellent credit rating buy a computer?

Exercise 2

Given graph:



$p(x_1,x_2,x_3,x_4,x_5) = ?$

Exercise 3

$p(x_1,x_2,x_3,x_4,x_5) = ?$

Build a graph for $p(x_1,x_2,x_3,x_4,x_5)$.