Lecture 12: Sampling and re-sampling

Sampling provides an effective way of solving problems for which a closed form solution is not feasible, and numeric solutions are too computationally demanding. A simple example is the problem "What is the probability of wining a game of Solitaire?". There may be an analytical solution, since the game can be defined unambiguously, but it is so complex that it may be too difficult to characterise. We could consider writing a program to enumerate all possible games and checking which ones we would win, but there are so many of them that, even with the worlds fastest computers and the most optimistic forecasts on global warming, the world won't last long enough for the computation to finish. However, you can just play 100 games and see how many you win and use this to estimate the probability. You probably won't be far wrong. Sampling solutions to problems like estimating the probability of winning at cards are called Monte Carlo methods.

Monte Carlo Methods

Given a set of discrete variables, and the ability to make random samples from them can we infer the probability distribution over the data. Basically we just take samples and for each new sample we recalculate the probability distribution using objective frequencies. We could also do a number of other things such as:

fit a distribution to the data
train a classifier (neural net, Bayesian net etc) with the data
retain the samples as the data model
etc.

Sampling provides one way of computing probabilities over the states of the unknown variables in a Bayesian Network. Given a Bayesian Network with variables, $X = (X_1, X_2..X_N)$, we can draw a sample from the joint probability distribution as follows.

```
1. Instantiate all the known variables to their values
2. Instantiate all unknown variables to a random state
3. while (sampling)
    3.1 Choose one of the unknown variables, say Xi, at random
    3.2 Compute the posterior probability over the states of Xi
    3.3 Select a state of Xi at random, based on the distribution
    3.4 Record the state of all the unknown variables (one sample)
```

We can always do this even if the network is multiply connected, because of the fact that, whenever we sample, every variable except one is treated as instantiated. This also means that the computation required for sampling is kept small.

If all nodes in a network except node X_i are instantiated, then only a small set of variables are needed to compute the posterior distribution overX_i. This set is made up of:

The Children of X_i
The Parents of X_i
The parents of the children of X_i

These variables are termed the Markov Blanket of X_i.

Monte Carlo Markov Chain (MCMC) methods.

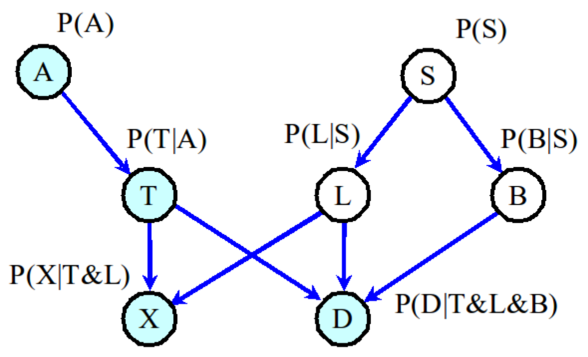
Given a Bayesian Network with some nodes instantiated in cases where propagation is not feasible we can estimate the posterior probabilities of the un-instantiated variables by drawing n samples as described above and calculating the posterior distributions from the frequencies. However, the state space may be big, so we need a very large number of samples.

It will be seen that there is a difference between the sampling strategy to compute the chance of winning at solitaire and the strategy proposed for estimating the unknown variables in a Bayesian network. In the first case we sampled at random, and indeed this is the only strategy that we can adopt since any one game is equally likely. However, in a Bayesian network some of the samples are more likely than others, reflecting the fact that some data points turn up more frequently than others in practical applications. So, for the Bayesian network, we set up a chain of samples each depending to some degree on the previous sample. We could have used random sampling from a Bayesian Network, for example at each step setting a subset of our unknown variables to a random state and then computing the probabilities of the rest. However this may not be the best strategy. The advantage of the Markov chain is that it enables us to pick the most representative samples and let the process converge quickly. At every iteration, we weight our selection towards the most probable sample. Hence our samples should follow the most common states accurately.

Gibbs Sampling in Bayesian Networks

One important MCMC method is called Gibbs Sampling and is a widely used technique. The main requirement is that the sampling process is ergodic and balanced. An ergodic chain is aperiodic and can reach every state in the space of the model. A balanced process will converge to a stable distribution given enough time.

The algorithm for calculating the probabilities of a Bayesian network, given above, is an example of Gibbs sampling. We will give an illustration here using the famous Asia problem.



A	Asia	A=a1
B	Bronchitis	Unknown
D	Dyspnea	D=d1
L	Lung Cancer	Unknown
S	Smoking	Unknown
T	Tuberculosis	T=t0
X	XRay	X=x0

Let us suppose that in a particular use of the network we have instantiations of the nodes A,T,X and D. This is a case where Pearl's propagation algorithm will fail to terminate on the loop L$\rightarrow$S$\rightarrow$B$\rightarrow$D.

The first step is to make a random initialisation of the unknown nodes. For example we could set B=b0, L=l1 and S=s1. We are now in a position to begin drawing a chain of samples. For each sample to be drawn we carry out the following steps:

1. Select one of the unknown nodes at random. For example we might select B.
2. Find the Markov blanket of the selected node (see above). For B the Markov blanket consists of S, D, L and T.
3. Using the assumed (or instantiated) states of the rest of the network recalculate the selected node. In this case B receives a π message from S. L and T have π messages for D which sends a λ message to B. The posterior probability of B can be calculated as normally.
4. Instantiate the selected node to a state depending on the calculated distribution. For example if we calculate P′(B) = [0.7,0.3] we select b0 with probability 0.7 and b1 with probability 0.3. Let us suppose we choose B=b0.
5. Record the sample which is [b0,l1,s1]. Note that we are not really interested in the known nodes as they will be fixed throughout the sampling process.
6. Go to step 1 to draw another sample if required.

Suppose that after drawing eight samples we have:

[b0,l1,s1],[b0,l1,s0],[b0,l1,s0],[b1,l1,s0],[b1,l0,s0],[b1,l0,s0],[b1,l0,s0],[b1,l1,s0]

we estimate the probabilities directly from the frequencies found in the samples:

$$P'(B) = (3/8, 5/8)\ P'(L) = (3/8, 5/8)\ P'(S) = (7/8, 1/8)$$

Metropolis-Hastings Algorithm

There are other strategies for drawing samples that have been extensively investigated. One is the Metropolis Hastings algorithm. As before, given a chain of samples $X_0, X_1, X_2..X_t$, compute sample X_{t+1} from X_t. The sample is then added to the chain according to the following criterion. A "proposal density" $Q(X_{t+1}|X_t)$ is computed. This is the probability of sampling X_{t+1} from X_t. Suppose that the sample has a probability $P(X_{t+1})$ (this is the joint probability in a Bayesian network). A probability ratio is calculated as follows:

$$\alpha = [P(X^{t+1})Q(X^t|X^{t+1})]/[P(X^t)Q(X^{t+1}|X^t)]$$

and the probability of acceptance is set to $p_t = min(\alpha, 1)$. Finally we add X^{t+1} to the chain with probability p_t. To give an intuition on what is going on we note that in many cases:

$$Q(X^{t+1}|X^t) \sim Q(X^t|X^{t+1})$$

meaning that all samples are equally probable, thus $p_t = min(P(X^{t+1})/P(X^t), 1)$ ie: always accept the sample if it is more probable than the previous. Otherwise weight acceptance in favour of more probable samples. This means that lower probability samples are less likely to be added to the chain.

Re-sampling

Re-sampling gives us a way of estimating statistical properties from a finite data set. Instead of using the data once, we use samples from it several times over to estimate properties like model accuracy and parameter variance

Hold out methods are the most widely used techniques involving re-sampling. Typically they involve holding back a proportion of the data to use in testing a model. This can be used to measure model accuracy.

The simplest hold out method is the **leave one out** resampling technique. Suppose we have a data set and we are using it to determine the structure of a Bayesian network. We proceeded as follows:

```
for each data point Dj:
    Compute the model parameters with all the other data points
    Calculate the prediction accuracy for Dj
Estimate the prediction accuracy by averaging the individual values
```

The average prediction accuracy is used as an estimate of the accuracy of the network when trained using all the data.

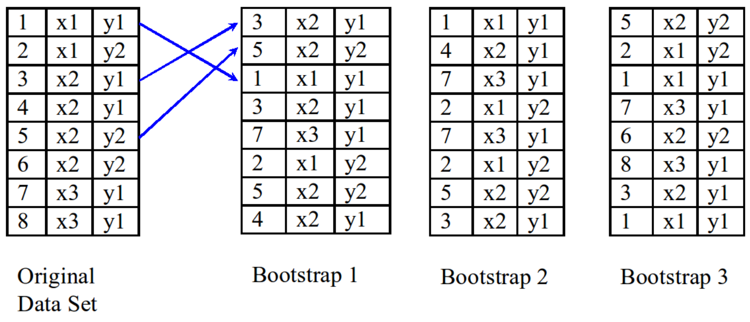
Leave one out is computationally expensive. An alternative is **cross validation**. Essentially we proceed as above but leave out a larger portion of the data than one point.

```
Divide the data into k similarly sized subsets
for each subset:
    Compute the model parameters using all the other subsets
    Find the average prediction accuracy for the subset
Estimate the prediction accuracy by averaging the individual values
```

Hold out methods are widely used to choose between competing models.

Bootstrapping

Bootstrapping is a method that can be used to estimate statistical properties from a finite data set. The technique involves sampling a large number of data sets from a single data set. Given a data set with m data points, a bootstrap data set is a data set of m points chosen at random with replacement from the original data set.



Original Data Set

1	x1	y1
2	x1	y2
3	x2	y1
4	x2	y1
5	x2	y2
6	x2	y2
7	x3	y1
8	x3	y1

Bootstrap 1

3	x2	y1
5	x2	y2
1	x1	y1
3	x2	y1
7	x3	y1
2	x1	y2
5	x2	y2
4	x2	y1

Bootstrap 2

1	x1	y1
4	x2	y1
7	x3	y1
2	x1	y2
7	x3	y1
2	x1	y2
5	x2	y2
3	x2	y1

Bootstrap 3

5	x2	y2
2	x1	y2
1	x1	y1
7	x3	y1
6	x2	y2
8	x3	y1
3	x2	y1
1	x1	y1

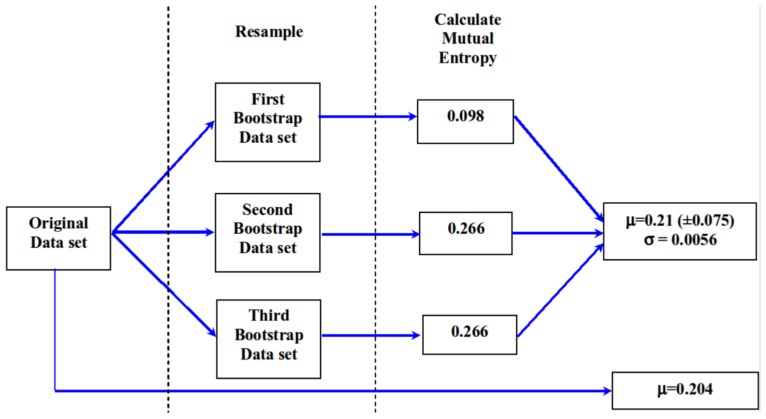
Bootstrapping can be used to estimate statistical properties of a data set. For example suppose that we have a data set with two variables (as shown above). One statistic we may be interested in is the mutual information:

$$Dep(X,Y) = P(X\&Y)log_2(P(X\&Y)/(P(X)P(Y)))$$

We can apply the method previously described of calculating the co-occurrences in the data, normalising them into a joint probability matrix, marginalising and then summing. However, this only gives us one value. Bootstrapping allows us to make several estimates of the mutual information from which we can determine a mean and variance.

```
Given a data set D in X-Y
Select n equi-probable bootstrap data sets from D
Calculate the statistic of interest (Dep(X,Y)) from each bootstrap set
Find the mean and variance of the estimate
```

The method is illustrated by the figure below.



Bagging, Boosting and Arcing

There are a number of variants of the bootstrapping technique which have been used in practice. Bagging (bootstrap-aggregating) is one of these. The objective is to improve the accuracy of prediction error estimates.

The method is to create a set of predictors using bootstrap samples of the data set. For example, instead of finding one Bayesian network from a data set we can generate a number of bootstraps and on each bootstrap we can find a separate Bayesian network. Given an inference problem we solve it on all of our networks and aggregate (average) the predictions to get a better estimate. Aggregating in this way will reduce the variance of the prediction.

Boosting, also known as Arcing (adaptive resampling and combining), is another related technique. Bagging creates bootstrap data sets by sampling the original with equal probability. Boosting changes the probability of selection, eg:

```
Sample a bootstrap data set $T_i$
Test the data set on $T_i$
Increase the probability of selection for misclassified points in further bootstrap samples
```

Bagging and Boosting are both found to reduce the variance prediction error in simulation studies. They are therefore proposed as good techniques for building classifiers, particularly with models that suffer from high variance (neural networks).