6.867 Machine Learning

Problem set 6 - solutions

Wednesday, November 27

What and how to turn in?

Turn in short written answers to the questions explicitly stated, and when requested to explain or prove. Do **not** turn in answers when requested to "think", "consider", "try" or "experiment" (except when specifically instructed). You may turn in answers to questions marked "optional"— they will be read and corrected, but a grade will not be recorded for them.

Turn in all MATLAB code explicitly requested, or that you used to calculate requested values. It should be clear exactly what command was used to get the answer to each question.

To help the graders (including yourself...), please be neat, answer the questions briefly, and in the order they are stated. Staple each "Problem" separately, and be sure to write your name on the top of every page.

Problem 1: Clustering

In this problem we will experiment with the spectral clustering algorithm, and explore its properties and the influence of its parameters. A Matlab implementation of spectral clustering is provided in `spectral.m`. You can call it as

```
labels = spectral(X,k,beta);
```

where `X` is the 2D data, `k` is the number of neighbors to use in the neighborhood graph, and `beta` is the weight falloff parameter. The default values for `k` is 3, for `beta` 1. Use `spectral` to experiment with the algorithm, and to test your hypotheses regarding the questions.

You may also find use for the little function

```
X = mkdata(m,n);
```

that generates random data points, from `m` Gaussian distributions, with `n` points from each. Such data typically has `m` sets of points that form natural clusters.

The code is written to perform binary clustering; let us first see how the algorithm deals with the data where there are indeed two clusters.

1. [**5pt**] What will typically happen if only a single nearest neighbor is used to create the graph (i.e. `k=1`)? Explain why that happens.

   Answer: The neighborhood graph will typically not be connected. Therefore, the basic operating assumption that motivates spectral clustering (higher probability of paths within a cluster than paths across the clusters in the random walk) breaks down. As a result the cluster assignments look "chaotic" and bear little connection to the underlying structure of the data (one would expect a reasonable clustering result to somewhat reflect the two Gaussian components).

2. [**5pt**] What happens if `beta` is set to a very large value (say, 100), when `k` is reasonable - say, 3? Explain.

   Answer: The clusters are assigned spuriously, with little relation to the underlying structure.

   The value of β influences the slope of the decaying weight on an edge as a function of the distance. Very low (close to zero) β means the weight is almost constant, and depends very little on the distance; very high β, on the other hand, means the weight falls off rapidly as the distance increases. With $\beta = 100$, the probability of transition from $\mathbf{x}$ to any of its neighbors is negligibly small (assuming we include self-transition as a possibility), and, more importantly, the relative difference between these probabilities for different points and for different neighbors of the same point are vast (many orders of magnitude). Such weights carry little information about the arrangement of the points to be clustered.

Now assume that there are in fact three clusters in the data. In the form given to you, the clustering algorithm of course can not discover the three clusters (since it only looks for two).

3. [**5pt**] How will the algorithm, as given to you, behave on the data with three clusters? Explain.

   Answer: There are two main possibilities. To simplify our discussion, let us denote the three components by A, B and C. One possibility is to have all the points assigned to a single cluster. This will happen when each component has

sufficiently many points nearby the other components thus creating significant transitions across the components in the random walk.

In the second typical case the data from two of the Gaussian components (say, A and B) are assigned to one cluster and the remaining component C is a cluster of its own. This happens when there are significant probabilities of transition between the points in A and B, but not between A and C or B and C.

Of course, one will occasionally encounter other scenarios. For instance, when the number of the data points drawn from each Gaussian is low, the neighborhood graph may be disconnected, often leading to a separate connected component per Gaussian. However, to use this the algorithm would have to include either a simple preprocessing stage, which the current implementation does not do.

4. [**10pt**] Modify the algorithm in order to find the three clusters. Note that deciding how many clusters are in fact present in the data is an important problem in clustering; you do not have to solve it, however - the modified algorithm should simply assume there are three clusters. Explain your modifications, and turn in your code.

Answer: One could think of a few ways to handle the multiple cluster situation. Below we mention two of those; any sensible proposal was given the credit. Note that we are not concerned here with the problem of finding the number of clusters: we assume that the data is known to contain 3 clusters, and we only need to find them.

One way is to implement a hierarchical clustering scheme, like the one mentioned in lecture. Namely, once the original clustering algorithm finds the two clusters, we can apply the clustering again within each cluster to search for further subdivisions. This does not require any change in the code. Note that unless some prior information is available on the size of the clusters, we do not know which cluster should be subdivided. We can apply the algorithm on both clusters, and the "right" one will produce a reasonable number of points assigned to different clusters.

An alternative is to use the 2 eigenvectors $\mathbf{v}, \mathbf{u}$ corresponding to the second and third largest eigenvalues. The pair (v_i, u_i) defines a 2D mapping of the data points. Typically the points are easily clustered in this 2D plane (using some simple clustering method, such as k-means).

Problem 2: HMM

One of the most notable contributions of HMM has been is speech recognition. In this problem we will take a look at an application of HMM to a relatively simple problem in speech – recognition of isolated words. We will focus on a "toy" version of this problem, namely, we are only interested in understanding whether the input utterance (acoustic speech signal) represents the word "four" or "five".

We will be working with the Mel-frequency cepstral coefficients (MFCC) of the signal, which are essentially the Fourier coefficients transformed in a way consistent with how the biological systems process acoustic information. You can find more about this representation in the literature devoted to speech recognition; one reference is

L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: Prentice Hall, 1993.

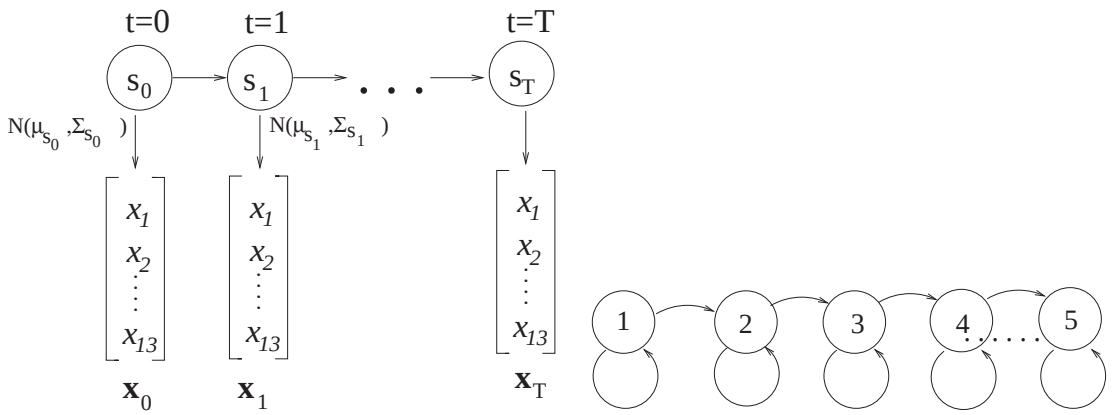
The 13-element MFCC vectors are the observations $\mathbf{x}_t$ in our model. The MFCC are computed within sequential time windows of fixed size, and since the length of the utterance varies from sequence to sequence , so does the number of observations. All the data used in this problem is found in the file `data45.mat`. The data was collected from 8 male speakers, each uttering the two words a number of times.The cell arrays `train4,train5` contain in each element (e.g. `train4{j}` a $13 \times T$ matrix of normalized MFCC, where T is the length of the corresponding observation sequence.

Our HMM for each spoken digit will contain 5 hidden states. The states roughly correspond to the phonemes (sounds) in the word; in our case, both "four" and "five" have three phonemes, and we expect the states to model these phonemes, plus the "silent" segments before and after the word. The transitions between states correspond to a sequential process, and are constrained in the following way: from a state s, the model can only transition to the state $s + 1$, no "skipping states" or going "backwards" is allowed. Figure 1(b) schematically shows the state transition diagram for this model.

Finally, we will model the probability density of observation (MFCC vectors) for each state by a single multivariate (13-dimensional) Gaussian with full covariance. The model is summarized in Figure 1. Before we proceed, it is important to note that this is an oversimplified model, and the HMM typically used for similar task (isolated word recognition) in speech would usually be more complex (larger number of states, observations modeled by a mixture of Gaussians, etc.)

We will use a (slightly modified) subset of the Matlab HMM toolkit written by Dr. Kevin Murphy; the original toolkit and many other useful software tools can be found at
http://www.ai.mit.edu/ murphyk/Software.
Extracting the archive `HMM.zip` or `HMM.tar.gz` will create a directory `HMM`; make sure you include it in Matlab working path (use the `addpath` command).

1. [**2pt**] What aspect of the HMM captures the estimated durations of the word parts (i.e., if one part usually takes longer time than another, how will this affect the HMM



(a) Our HMM for spoken digits: an overview

(b) State diagram

Figure 1: The HMM used in Problem 2

parameters)?

Answer: A part (phoneme) is modeled by a recurring state corresponding to that part. The event of staying in the same state k at time t is a Bernulli event, with probability $P(s_{t+1} = k|s_t = k)$ of success. The duration of the part correspond to the length of a sequence of consecutive "successes" of that process; longer expected duration therefore means a higher value of the corresponding diagonal element of the transition probability matrix.

2. [**3pt**] Prove that if an entry i, j in the initial "guess" of the transition probability matrix provided to the EM is set to zero, it will remain zero in the result obtained with the EM.

Answer: It is sufficient to show that the property is preserved through a single EM iteration. That is, we assume that in the beginning of iteration m, the estimate $\hat{P}^m(s_{t+1} = j|s_t = i) = 0$ (which is known to be true for $m = 0$), and prove by induction. Let us start with the E-step. The value of $\xi_t^l(i, j)$ is, by definition,

$$\xi_t^l(i,j) = \hat{P}^m(s_{t+1} = j|s_t = i)\frac{\alpha_t(i)\hat{P}^m(x_{t+1}|j)\beta_{t+1}(j)}{\sum_k \alpha_t(k)\beta_t(k)} = 0$$

for all training sequences $l = 1, \ldots, L$ and all $t = 0, \ldots, n$. Therefore, in the M-step

$$\hat{\eta}(i,j) = \sum_{l=1}^{L}\sum_{t=0}^{n} \xi_t^l(i,j) = 0,$$

and consequently

$$\hat{P}^{m+1}(i,j) = \hat{\eta}(i,j) / \sum_{j'} \hat{\eta}(i,j') = 0.$$

3. [**2pt**] Precisely what constraints do the 5×5 transition probabilities matrix $P(s_{t+1}|s_t)$ and the 5-element initial state probabilities vector P_0 have to satisfy in our model?

   Answer: We always start from the first state - therefore the initial state probabilities are

   $$P_0(s_t) = [1,0,0,0,0]^T.$$

   The constraints on the transitions dictate that the transition probabilities matrix be in the form

   $$P(s_{t+1}|s_t) = \begin{pmatrix} p_{00} & p_{01} & 0 & 0 & 0 \\ 0 & p_{11} & p_{12} & 0 & 0 \\ 0 & 0 & p_{22} & p_{23} & 0 \\ 0 & 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4. [**3pt**] Using the implementation of the EM algorithm provided to you, estimate the parameters of the two HMMs, based on the data in `train4` and `train5` respectively. The function you should use is

   ```
   [p0, pt, mu, sigma] = learn_ghmm(X, initP0, initPt);
   ```

   The inputs to `learn_ghmm` are the cell array of observation sequences `X` and the values, respectively, of $P_0(i)$ and $P(s_{t+1}|s_t)$ that the EM should use as the initial guess. In the output, `p0` is the estimated P_0, `pt` the estimated transition probabilities matrix $P(s_{t+1}|s_t)$, and `mu` and `sigma` are the estimated parameters of the Gaussian distributions of observations for each state (the Gaussian for state k is determined by `mu(:,k)` and `sigma(:,:,k)`). Make sure you appropriately constrain the transition matrix and the starting state probabilities, according to your answers to the previous two questions.

   Turn in the resulting transition probability matrix for digit "four".

   Answer: As usual with the EM, the results are non-deterministic due to random initialization. The results we got were

   $$\begin{pmatrix} 0.9341 & 0.0659 & 0 & 0 & 0 \\ 0 & 0.9406 & 0.0594 & 0 & 0 \\ 0 & 0 & 0.9622 & 0.0378 & 0 \\ 0 & 0 & 0 & 0.9480 & 0.0520 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Having trained an HMM (estimated its parameters) we can apply it to various tasks related to the spoken digit recognition. One of the main goals is, of course, classification - given an unlabeled utterance, we would like to tell whether it is a "four" or a "five". The cell array `test45` contains 252 MFCC representations of spoken digits; the true labels are given in the array `labels`. We will now test how well our HMMs can classify these utterances. To do so, we will need to use the function

```
loglikes = log_lik_ghmm(data, prior, transmat, mu, Sigma);
```

which takes the cell array `data`, the probabilities of the initial state `prior`, the transition probability matrix `transmat`, and the parameters of observation densities `mu,sigma`, and returns in `loglikes(k)` the log-likelihood of the observation sequence `data{k}` given the HMM.

5. [**5pt**] Classify the utterances in `test45` based on their log-likelihoods under the two models. Explain in one or two sentences how you perform the classification and why it makes sense (no need to turn in code). What is the misclassification rate on `test45`?

   Answer: If we denote the two HMMs (corresponding to the two classes) as M_1 ahd M_2, and assume that both digits are equally likely to occur, the classification of the utterance D in the Bayesian formulation can be expressed as finding

   $$M^* = \operatorname*{argmax}_M p(D|M),$$

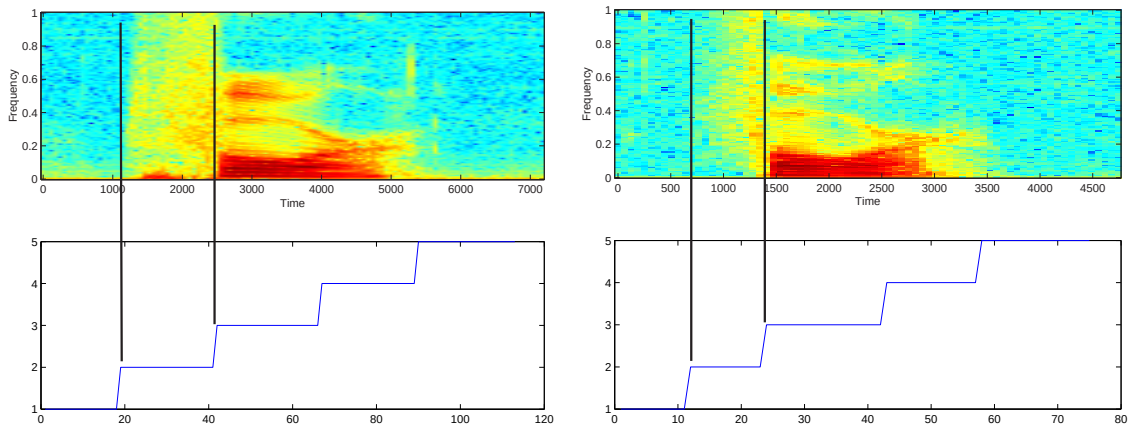
   which corresponds to finding the model with higher log-likelihood for the data. The trained HMMs perform remarkably well on this test set; in our experiments, all the test utterances were classified correctly (misclassification rate zero).

Another possible use for HMMs can be to align two observation sequences. In the case of speech analysis, this could mean marking the segments of the signal corresponding to the same hidden state. For this part, you will need to use the function

```
path= viterbi_path(data, prior, transmat, mu, sigma);
```

which for the observation sequence `data` returns the most probable sequence of states, found by Viterbi algorithm with the HMM described by `prior,transmat,mu` and `sigma`. We will try this on just one example - a pair of utterances of the word "four". The original sampled audio signal is given in the vectors `signal1, signal2`, and the MFCC representation in `mfcc1, mfcc2`. Note that these utterances (by different speakers) have quite different lengths.

6. [**3pt**] Explain how the most probable sequence of states helps us to align the sequences.



(a) Utterance 1 (b) Utterance 2

Figure 2: Spectrogram alignment using HMM; top - spectrogram of a signal, bottom - the most likely sequence of states obtained by Viterbi. Parts corresponding to state 1 are marked in the spectrograms.

> **Answer:** From the most likely sequence, we can find the most likely location of transition between states k and $k+1$. Having found those for both sequences, we can align the data by matching these locations.

7. [**7pt**] Using Matlab function `specgram`, plot the spectrogram of each signal, and under it plot the graph showing the states found by Viterbi algorithm (use the `subplot` command). Assuming that the states roughly correspond to sounds, mark the areas on the two spectrograms that probably correspond to the sound "f". (Recall that the observed sequences typically have a "silence" period in the beginning, which is modeled by a separate hidden state).

   > **Answer:** Figure2 shows the resulting alignment. Under our model, 'f' in both "four" and "five" is supposed to be modeled by the second state (the first state of the HMM models the initial silence).

We now turn to a more interesting task: separating words in a sequence. We will do it in a simplified setting, for a sequence of words known to be either "four" or "five". In order to separate the words, we must model not only the separate words, but also the transition between the words. Normally, these would be estimated based on a training set of word sequences; here, we shall simply assume that the probability of leaving the final state of the word and going to the initial state of the next word is 0.2, and that the probabilities of "four" and "five" are equal.

8. [**5pt**] Explain how a single HMM would combine the two individual HMMs; turn in the initial state probabilities and the transition matrix of this combined HMM.

Answer: One can view this as an hierarchical model, in which we have two meta-states, corresponding to the two digits. Under our assumptions, the transition probabilities between all the meta-states are 1/2 (the next word is equally likely to be "four" or "five").

Let the states of the HMM trained for "four" be numbered 1 through 5, and the states of the HMM for "five" 6 through 10. Then transitions between the meta-states corresponds to transitions 5→1, 5→6, 10→1 and 10→6. We are equally likely to start from either meta-state, which means starting from 1 or 6. To summarize, the (hand-crafted) parameters of the combined HMM, for the particular results of training the 5-state HMMs,

$$P_0 = \begin{pmatrix} .5 \\ 0 \\ 0 \\ 0 \\ 0 \\ .5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and

$$P(s_{t+1}|s_t) = \begin{pmatrix} 0.934 & 0.066 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.941 & 0.059 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.962 & 0.038 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.948 & 0.052 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0.1} & 0 & 0 & 0 & \mathbf{0.8} & \mathbf{0.1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.926 & 0.074 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.941 & 0.059 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.958 & 0.042 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.929 & 0.071 \\ \mathbf{0.1} & 0 & 0 & 0 & 0 & \mathbf{0.1} & 0 & 0 & 0 & \mathbf{0.8} \end{pmatrix}$$

Finally, we need to define the probability densities of the observation for each state; these, of course, correspond to the internal state, and thus are the same as before. To summarize: the following Matlab code will set up the combined HMM (assuming `pt4,mu4,sigma4` are the estimated parameters of the HMM for "four" and similarly for "five"):

```
>> p0=[.5 0 0 0 0 .5 0 0 0 0]';
>> pt=[pt4 zeros(5,5);zeros(5,5) pt5];
>> pt(5,6)=.1;pt(5,1)=.1;pt(5,5)=.8;
```

```
>> pt(10,6)=.1;pt(10,1)=.1;pt(10,10)=.8;
>> mus = [mu4 mu5];
>> sigmas(:,:,1:5)=sigma4;
>> sigmas(:,:,6:10)=sigma5;
```

9. [**5pt**] We will work with 4 sequences of two spoken digits `seq1,...,seq4`. Using the combined HMM, and the function `viterbi_path`, find for each sequence what are the two digits, and where is the boundary, according to the HMM estimate. Along with your answers turn in the plots, or numerical results, on which you base your estimates.

   Answer: We can plot the most likely sequence of states by running

   ```
   plot(viterbi_path(seq1,p0,pt,mus,Sigmas));
   ```

   The results, for the four MFCC sequences, are shown in Figure 3. Recall that the states 1 through 5 correspond to the meata-state "four", while states 6 through 10 correspond to "five". The transitions between these two sets of states correspond, therefore, to transitions between the words. Moreover, since within one word the path can only proceed through the states in ascending order, any transition from states 5 and 10 correspond to a new word. Based on this reasoning, one can identify that the words in the sequences are, according to the HMM estimate:

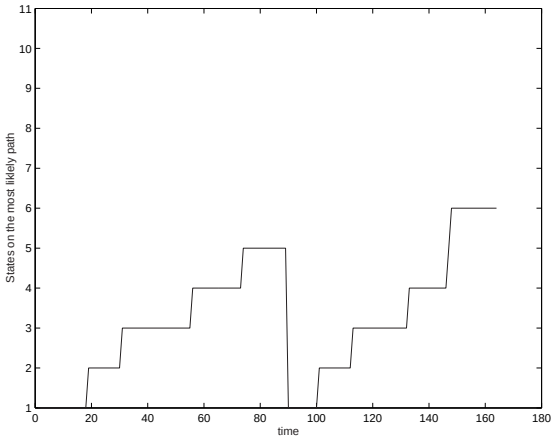
   `seq1` "four" at $t = 0 \ldots 90$, followed by "four",

   `seq2` "four" at $t = 0 \ldots 75$, followed by "five",

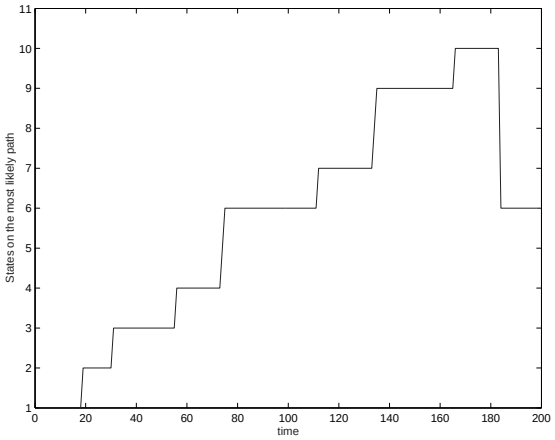
   `seq3` "five" at $t = 0 \ldots 94$, followed by "five",

   `seq4` "four" at $t = 0 \ldots 86$, followed by "five".

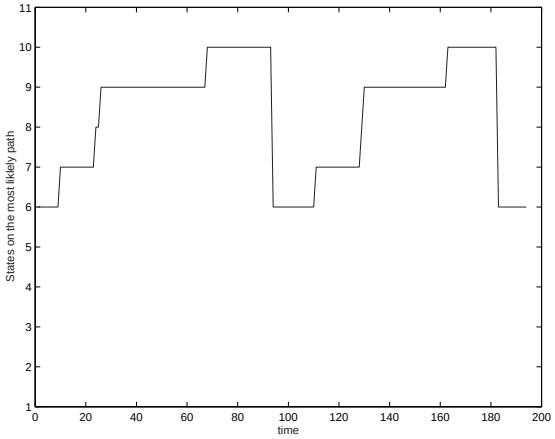
   The word identification is correct, and all the boundaries are within the silence period between the two words, so the results could be said to be "correct". One can note, however, some clear effects of the imperfections of our naive model. For instance, the final observations are attributed to the beginning of a non-existing third word (we do not constrain our model to end in states 5 or 10). Also, the model for "five" tends to attribute very short observation portions to the third state (the correspondence of states to phonemes is probably not very true in this case). Such problems could be solved, at least partially, by improving the model, and by introduction of supervision (e.g., human-marked phonemes). Nevertheless, our naive HMM does a reasonable job segmenting the sequence, and identifying the words.



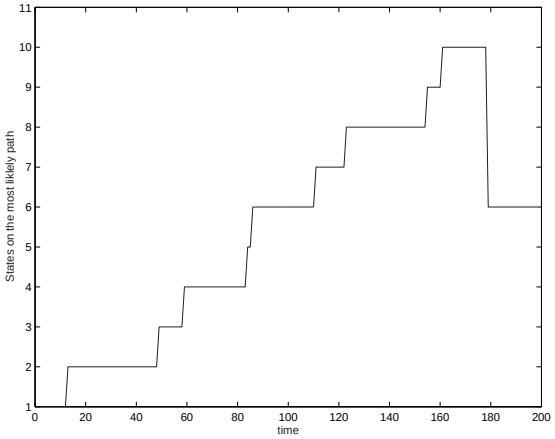
(a) `seq1`



(b) `seq2`



(c) `seq3`



(d) `seq4`

Figure 3: The four test cases for word identification

C	$P(C)$
g	.3
a	.5
b	.2

(a)

	$P(T_1\|C)$		$P(T_2\|C)$	
C	1	0	1	0
g	.7	.3	.8	.2
a	.5	.5	.4	.6
b	.2	.8	.1	.9

(b)

T_1	T_2	$P(D\|T_1,T_2)$	
		0	1
0	0	1	0
0	1	1	0
1	0	0	1
1	1	0	1

(c)

Table 1: Settings for Problem 3. (a) $P(C)$. (b) $P(T_1|C)$ and $P(T_2|C)$. (c) $P(D|T_1, T_2)$.

Problem 3: Graphical models

Let us start by warming up with a simple belief network. Consider the following scenario. A person (hereafter called the "buyer") wants to buy a used car; the condition of the car is a random variable C, with the value in $\{g, a, b\}$ corresponding, respectively, to good, average and bad quality. Our belief about the frequency of each kind in the market is expressed by the distribution of C, shown in Table 1(a). The buyer bases his decision whether to buy a car on the outcome of two tests which he runs on the car, T_1 and T_2. The outcomes, t_1 and t_2 respectively, are binary - i.e. the car either passes a test (1) or fails (0). Unfortunately, the results of the tests are non-deterministic, even given the condition of the car. We do know the distribution of this results, conditioned on the car quality - see Table 1(b). Based on the test outcomes, the buyer decides whether to buy a car; for the sake of convenience, let us for now treat this decision as a random variable, even though its distribution (conditioned on the test outcomes) shown in Table 1(c) shows that it is in fact a deterministic decision. We encode the decision to buy as 1 and not to buy as 0.

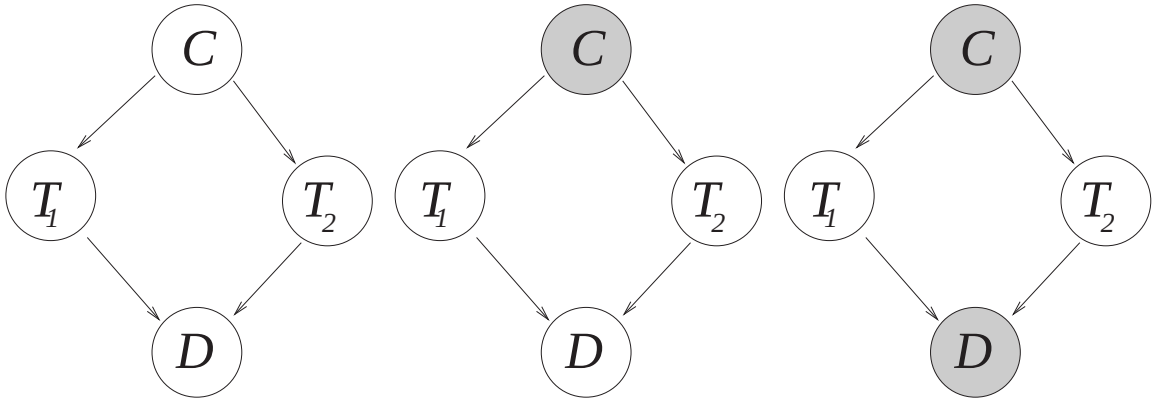
1. [**2pt**] Draw the graphical model corresponding to the described settings. Your model should contain 4 nodes (C, T_1, T_2, D) – mark them clearly.

   Answer: The model is drawn in Figure 4(a).

2. [**2pt**] Write down the joint probability of observing the quadruple $(C = c, T_1 = t_1, T_2 = t_2, D = d)$ as a product of known distributions.

   Answer:
   $$P(c, t_1, t_2, d) = P(c)p(t_1|c)p(t_2|c)p(d|t_1, t_2)$$

3. [**2pt**] What are the marginal distributions of test outcomes, $P(T_1)$ and $P(T_2)$? What is their joint distribution? Are these two random variables marginally independent?



(a) The graphical model (b) Observing C (c) Observing C and D

Answer: The marginal distribution of T_1 can be found from the joint:

$$P(t_1) = \sum_{c,t_2,d} P(c, t_1, t_2, d).$$

The summation over the three variables includes, in principle, 3×2×2=12 terms (for each possible value of t_1). However, this can be simplified due to the structure of the model:

$$\begin{aligned} P(t_1) &= \sum_{c,t_2,d} p(c)p(t_1|c)p(t_2|c)p(d|t_1,t_2) \\ &= \sum_{c} P(c)P(t_1|c) \sum_{t_2} P(t_2|c) \sum_{d} P(d|t_1,t_2) \end{aligned}$$

Since $\sum_x P(x|y) = 1$, we can take the sum over only 3 terms for each value of t_1, and get the (obvious)

$$P(t_1) = \sum_{c} P(c)P(t_1|c) = \begin{cases} 1/2 & \text{for } t_1 = 0, \\ 1/2 & \text{for } t_1 = 1. \end{cases}$$

Similarly, one gets

$$P(t_2) = \begin{cases} .54 & \text{for } t_2 = 0, \\ .46 & \text{for } t_2 = 1. \end{cases}$$

The joint of the test outcomes can be computed in a similar manner:

$$P(t_1, t_2) = \sum_{c} P(t_1|c)P(t_2|c)P(c) = \begin{cases} .312 & (0,0) \\ .188 & (0,1) \\ .228 & (1,0) \\ .272 & (1,1) \end{cases}$$

which is different from $P(t_1)P(t_2)$ (e.g. $P(t_1 = 0)P(t_2 = 0) = .5 \times .54 = .27$). Therefore, T_1 and T_2 are not marginally independent. This is also what we would expect based on the graph in Figure 4(a) since both variables depend on C. However, the parameters could be chosen in such a way that T_1 and T_2 would become marginally independent despite the graph (the probability distribution associated with the graph may satisfy additional independencies beyond those expressed by the graph).

4. [**2pt**] You are told that the car in question is actually in average condition. What is the joint distribution of T_1, T_2 given that knowledge? Are T_1 and T_2 independent given C?

   Answer: This situation is graphically depicted in Figure 4(b). The joint distribution of T_1, T_2 is now given (after eliminating the sum over d) by

   $$P(t_1, t_2|c) = P(t_1|c = a)P(t_2|c = a),$$

   and the the variables are independent given C (another way of checking that is by building the moralized ancestral graph for T_1, T_2, C).

5. [**2pt**] Suppose that in addition to knowing that the car is in average condition, you also know that the buyer purchased the car. What is the joint distribution of T_1, T_2 given that knowledge? Are T_1 and T_2 independent given C and D?

   Answer: The situation is depicted in Figure 4(c). Now we can not simply eliminate d:

   $$P(t_1, t_2|c, d) = \frac{P(t_1, t_2, c, d)}{P(c, d)}$$

   We can calculate the denominator as

   $$\begin{aligned} P(c = a, d = 1) &= \sum_{t_1,t_2} P(t_1, t_2, c = a, d = 1) \\ &= P(c = a) \sum_{t_1,t_2} p(t_1|a)p(t_2|a)p(d = 1|t_1, t_2) = .25 \end{aligned}$$

   and therefore

   $$P(t_1, t_2|c, d) = \begin{cases} .6 & (1, 0) \\ .4 & (1, 1) \\ 0 & (0, 0), (0, 1) \end{cases}$$

   Finding the marginals in much the same manner as before, we get

   $$P(t_1|c, d) = \begin{cases} 0 & t_1 = 0, \\ 1 & t_1 = 1 \end{cases} \text{ and } P(t_1|c, d) = \begin{cases} .6 & t_2 = 0, \\ .4 & t_2 = 1 \end{cases}$$

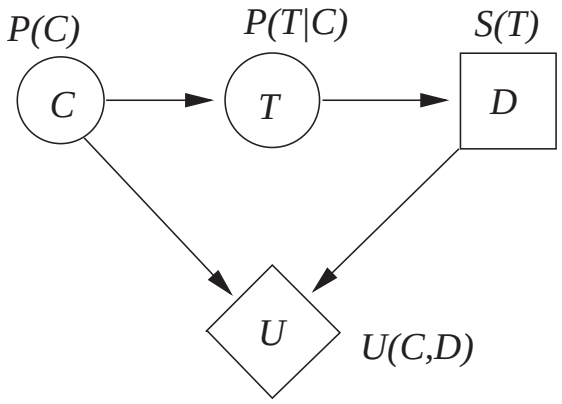


Figure 4: Simple influence diagram. Semantics of the nodes: C – condition of the car, T – the outcome of the test, D - the decision whether to buy or not, U - the utility to be maximized by the decision.

> Therefore, perhaps surprisingly, T_1 and T_2 are actually independent given C and D. This example shows why one cannot conclude *dependence* from the presence of an arrow in a graphical model - only independence from the *absence* of arrow. In our case, the conditional distribution $P(D|T_1, T_2)$ is in fact $P(D|T_1)$ - i.e. the arrow from T_2 to D carries no real effect.

Treating D as a random variable is somewhat of an abuse - it is, in fact, a special entity – a *decision node*. Such nodes are usually shown in a graphical model as squares, to distinguish them from the random variable nodes (circles). A decision node represent choices available to a decision-maker. The arcs entering a decision node have different meaning from the arcs entering a random variable node; rather than denoting probabilistic dependence, they show what informational input is available to the decision maker at that node.

In addition, we can define yet another special kind of nodes in a graphical model – *value nodes*. A value node, which is shown as a diamond, represents a utility (function) to be maximized by the decision maker. Arcs entering a value node show functional dependence (as in Bayesian network,the exact form of the dependence must be stored in the node). The belief network, augmented by decision and value nodes, is called an *influence diagram*.

Before we proceed, let us look at an example. Suppose we have a simplified version of our car purchase scenario, in which there is a single test, T, which the car passes with probability 0.8, 0.4 or 0.1 if its condition is respectively g, a or b. The distribution of C is the same as in Table 1. Based on the test outcome, the buyer decides whether to buy the car or not. The influence diagram for this scenario is shown in Figure 4.

In order to determine the utility function $U(c, d)$, we will also need the information on losses and gains associated with each decision. Let the price of the car be \$1000, and assume that the market value for a good car of this model is $V(g)$=\$1,500, for a bad car it is $V(b)$=\$300, and for an average car $V(a)$=\$1100. The buyer must also pay the fee for

the test - say, \$50. The utility function then is

$$U(c,d) = \begin{cases} u^0 = -\$50, & \text{if } d = 0 \\ u^1(c) = V(c) - \$1050, & \text{if } d = 1. \end{cases} \quad (1)$$

The task typically associated with an influence diagram is to select the strategy at the decision nodes that would maximize the value of the utility. A strategy $S(X)$ is a function associated with a decision node, which given the input X to that node, computes the decision value. There are four possible strategies in our case:

$$S_1(t) : \quad t = 0 \rightarrow d = 0, \quad t = 1 \rightarrow d = 0, \quad (2)$$
$$S_2(t) : \quad t = 0 \rightarrow d = 0, \quad t = 1 \rightarrow d = 1, \quad (3)$$
$$S_3(t) : \quad t = 0 \rightarrow d = 1, \quad t = 1 \rightarrow d = 0, \quad (4)$$
$$S_4(t) : \quad t = 0 \rightarrow d = 1, \quad t = 1 \rightarrow d = 1. \quad (5)$$

We can compute the expected utility for a strategy S, given the distribution of the random variables in the diagram:

$$E_S = \mathbf{E}_{C,T}\left[U(c, S(t))\right] = \mathbf{E}_C\left[\mathbf{E}_{T|C}\left[u(c, S(t))\right]\right] = \sum_{c=g,a,b} P(c) \sum_{t=0,1} P(t|c)u(c, S(t)), \quad (6)$$

which becomes, for our four strategies,

$$\begin{aligned} E_{S_1} &= -50, \\ E_{S_2} &= .3(.8 \times 450 - .2 \times 50) + .5(.4 \times 50 - .6 \times 50) + .2(-.1 \times 750 - .9 \times 50) = 76, \\ E_{S_3} &= .3(.2 \times 450 - .8 \times 50) + .5(.6 \times 50 - .4 \times 50) + .2(-.9 \times 750 - .1 \times 50) = -116, \\ E_{S_4} &= .3 \times 450 + .5 \times 50 - .2 \times 750 = 10. \end{aligned}$$

Thus, perhaps not surprisingly, the strategy that maximizes the expected utility value is S_2 - buy the car only if it passes the test. Of course, a different cost, or a different probability distribution of a random variable could change that.

Let us now proceed and develop a somewhat more interesting framework for the car buying problem. The meaning of C and its distribution are same as before. Test 1, with probability distribution of the outcome T_1 as described in Table 1, consists of simply looking at the car, and costs nothing. Its outcome can be interpreted as whether the buyer liked the car (1) or did not (0). We can make a decision whether to buy the car (perhaps based on the outcome of T_1), or ask a mechanic to check the car first – test 2, with outcome T_2, for which there is a fee of \$50 – and then decide. The car price and the market value are the same as described above. Note that different decision strategies involve decisions based on both tests, one of the tests, or no tests at all. The utility of the decision is again measured by the financial gain of the buyer.

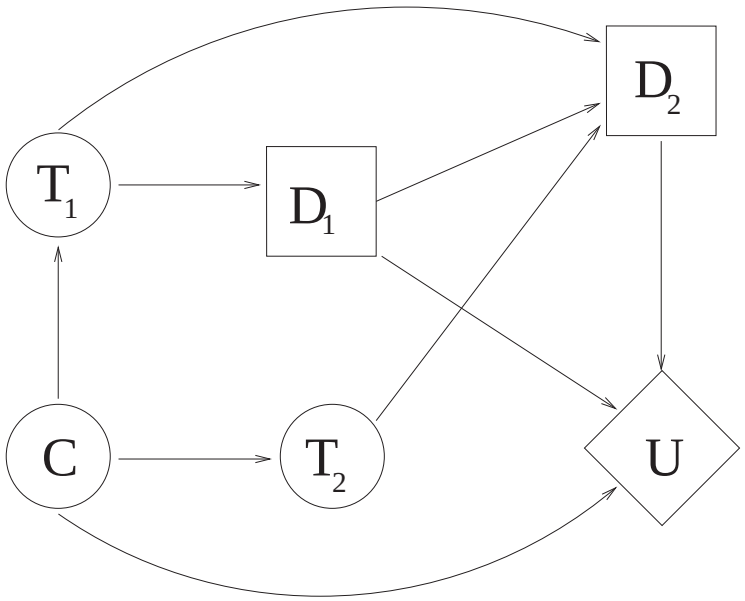


Figure 5: Influence diagram for 2.6

6. [**10pt**] Draw an influence diagram corresponding to this setting. Include the value node U. For each node clearly show its kind (by drawing it as a circle/square/diamond), name it, and describe its semantics (meaning) in no more than one sentence.

   Answer: The diagram is shown in Figure 5. The condition of car influences the test outcomes, but does not directly affect the decisions (i.e. we do not about C anything other than conveyed by T_1 and T_2). The result of the first decision affects the second decision: technically, D_2 has 2 inputs if $D_1 = 1$ and only one (T_1) if $D_1 = 0$. Note that the test outcome T_2 *does not depend* on D_1. Conceptually, the semantics of our model are as follows: we assume that the test outcome T_2 "exists" whether we look at it or not (and is drawn from the corresponding conditional distribution). The value of D_1 tells us whether to look at this outcome (and pay for it), or not.

   Finally, the utility value U is a function of the decisions made by the buyer, and of the true conditions of the car. That includes D_1, which determines whether the mechanic's fee is to be subtracted from the gain. Utility does not depend on the test outcomes (other than through the decisions).

7. [**3pt**] How many possible strategies are implementable in this influence diagram? Explain what they are (do not enumerate all the strategies!). Note that the strategy is described by the functions associated with *all* the decision nodes in the diagram.

   Answer: Let us denote the "local" strategy in D_1 by $S_1(t_1)$, and the local strategy in D_2 by $S_2(t_1, t_2, d_1)$. The "global" strategy can be written as

$S_2(t_1, t_2, S_1(t_1))$. Let us first focus on S_1. Similar to the example above, there are 4 strategies for that decision node, which correspond to the 4 Boolean functions of the single binary input t_1.

The decision in D_2 implements a Boolean function of two variables (test outcomes), but it is also constrained by D_1. Not all combinations of feasible local strategies in the two decision nodes create a feasible global strategy, however. Let us look at the possibilities.

D_1: never do T_2 The decision in D_2 is a function of t_1 only; there are 4 such functions, and thus 4 global strategies that include this local strategy in D_1.

D_1: always do T_2 Any Boolean function of t_1, t_2 is feasible - therefore, there are 16 global strategies corresponding to this case.

D_1: do T_2 iff T_1 fails First, we have to provide a decision for the case no T_2 is done; this only can happen if $t_1 = 1$ because otherwise T_2 would be required. That means a constant Boolean function f_1 (no arguments) - i.e. 2 possibilities. Otherwise, we do T_2 and need to decide based on the outcome of T_2 (again, we know that in this case necessarily $t_1 = 0$). There are 4 possible Boolean functions $f_2(t_2)$ that provide the decision. Any combination of f_1, f_2 provides a feasible strategy, therefore, we have 8 global strategies for this D_1.

D_1: do T_2 iff T_1 passes Very similar to the previous case, but for the opposite values of t_1. Again, 8 strategies.

To summarize we have a total of 36 strategies. Note that we are not concerned here with the question whether some of the may be equivalent, in terms of the resulting utility, under the given distributions; we are counting strategies that *could* produce different expected utilities for some distributions associated with the model.

8. [**2pt**] Write down the expected value of the utility as a function of the strategy.

Answer: Let us denote the market value of the car in condition c by $V(c)$. We also will for simplicity write the decision function in D_2 as a function of two variables $d_2(t_1, t_2)$ including the cases where the functional dependence on one or both arguments is trivial. The utility value is determined as

$$U(c, d_1, d_2, t_1, t_2) = U(c, d_1(t_1), d_2(t_1, t_2, d_1(t_1))) = d_2(V(c) - 1000) - 50d_1$$

Taking the expectation w.r.t. the random variables, we have

$$\mathbf{E}_{P(c,t_1,t_2)}[U] = \mathbf{E}_{P(c,t_1,t_2)}[d_2(t_1, t_2)(V(c) - 1000)] - 50\mathbf{E}_{P(t_1)}[d_1(t_1)].$$

9. [**5pt**] Find the strategy with the maximal expected utility, and the maximum value attained by that strategy. *Advice: You may find it appropriate to run a short script in Matlab or another programming language that will compute the expected utility for different strategies; no need to turn it in.*

Answer: A simple script, which iterates over the 36 strategies and computes (by simple summation over c, t_1, t_2) the expected utility for each, finds that the maximal expected utility, \$111.8, is reached by the following strategy: if $T_1 = 1$, buy the car. Otherwise, do T_2 and buy the car if it passes the test.