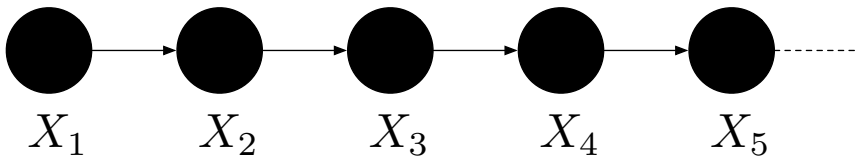
Markov Models

Markov models are a fundamental construct in time series analysis, dynamical systems, and reinforcement learning. Abstractly, a sequence of random variables $X_1, X_2, \ldots, X_N$ is said to be a **Markov chain** if we can factor their joint pdf as

$$f_{\{X_n\}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f_{X_1}(\boldsymbol{x}_1) f_{X_2}(\boldsymbol{x}_2|\boldsymbol{x}_1) f_{X_3}(\boldsymbol{x}_3|\boldsymbol{x}_2) \cdots f_{X_N}(\boldsymbol{x}_N|\boldsymbol{x}_{N-1}).$$

This means that $X_1, \ldots, X_{n-1}$ is conditionally independent of $X_{n+1}, \ldots, X_N$ given X_n.

We can depict the (conditional) dependency structure of a Markov chain as



Of course, the most well-known Markov process is the Gauss-Markov process where the X_n are Gaussian. For example, a first-order autoregressive model (AR1) takes

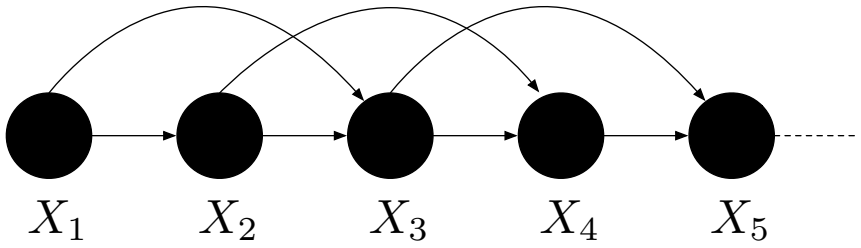
$$X_{n+1} = \alpha X_n + Z_{n+1}, \quad Z_{n+1} \sim \text{Normal}(0, \sigma^2).$$

This is an example of a **random walk**, or a discrete-time version of Brownian motion.

The ideas above can be extended to second-order Markov models, where X_n is conditionally independent of $X_1, \ldots, X_{n-3}$ given X_{n-1} and X_{n-2}. In this case, the joint pdf is factored as

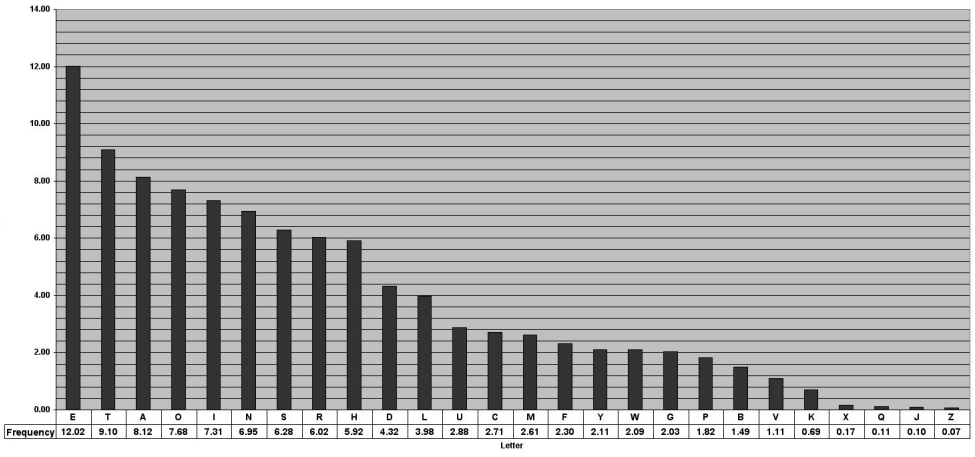
$$f_{\{X_n\}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f_{X_1}(\boldsymbol{x}_1) f_{X_2}(\boldsymbol{x}_2|\boldsymbol{x}_1) f_{X_3}(\boldsymbol{x}_3|\boldsymbol{x}_2, \boldsymbol{x}_1) f_{X_4}(\boldsymbol{x}_4|\boldsymbol{x}_3, \boldsymbol{x}_2) \cdots$$

and the associated graph looks like



It does not take too much imagination to know how a third- or fourth- or k-th order Markov process is structured.

Here is another process that can be (at least roughly) approximated as Markov: sequences of letters that appear in written English text. It is no surprise that the 26 letters (27 if you include the space) in the alphabet occur at different frequencies; here is one estimate[1]:



Here is an example of random text generated using iid samples from this pmf[2]:

[1] Taken from http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

[2] This example, and the subsequent ones of its kind, comes from [CT91, Chapter 6].

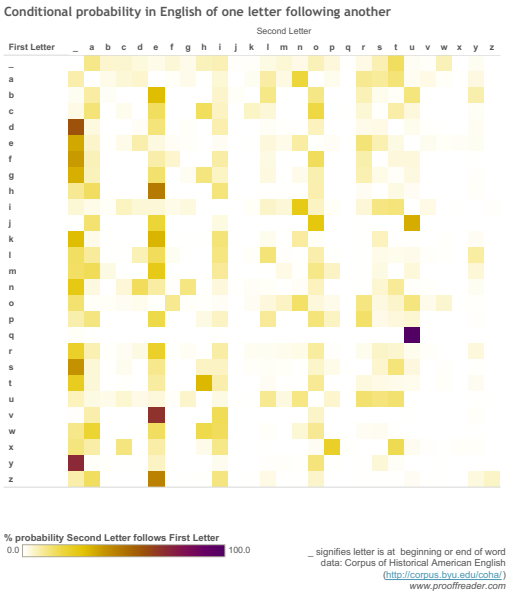
Zero-order approximation. **(The symbols are independent and equiprobable.)**

`XFOML RXKHRJFFJUJ ZLPWCFWKCYJ`

`FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD`

It should also not be a surprise that given a *context* of one letter, the conditional distribution of the next letter changes. The easiest example is the letter 'u' — at any point in the text, without context, the probability of a letter being 'u' is 2.8%. However, if the previous letter was 'q', then the conditional probability becomes 98.7%.

You can imagine a 27×27 matrix whose columns are indexed by first letter and rows indexed by second letter that captures the **transition probabilities** from each possible character to the next. Here is a graphical depiction of such a matrix[3] (with the rows and columns reversed):



If we assume that the English language is *stationary*, meaning that

[3] Taken from http://www.prooffreader.com/2014/09/how-often-does-given-letter-follow.html.

the probability of a 'b' following a 'u' is the same when the 'u' appears on page 1 as when is appears on page 37, then this transition matrix along with the marginal probabilities are enough to define a joint pdf for a sequence of N letters. Here is an example of random text generated using this first-order Markov model:

> *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)
>
> OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI
>
> ALHENHTTPA OOBTTVA NAH BRL

Higher order models can also be trained from data (and there is a lot of data!). Here are examples in a similar spirit to the above

> *Second-order approximation.* (The frequency of pairs of letters matches English text.)
>
> ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY
>
> ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO
>
> TIZIN ANDY TOBE SEACE CTISBE

> *Third-order approximation.* (The frequency of triplets of letters matches English text.)
>
> IN NO IST LAT WHEY CRATICT FROURE BERS GROCID
>
> PONDENOME OF DEMONSTURES OF THE REPTAGIN IS
>
> REGOACTIONA OF CRE

Fourth-order approximation. (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

```
THE GENERATED JOB PROVIDUAL BETTER TRAND THE
DISPLAYED CODE, ABOVERY UPONDULTS WELL THE
CODERST IN THESTICAL IT DO HOCK BOTHE MERG.
(INSTATES CONS ERATION. NEVER ANY OF PUBLE AND TO
THEORY. EVENTIAL CALLEGAND TO ELAST BENERATED IN
WITH PIES AS IS WITH THE)
```

You can also, of course, assign states to entire words. The size of the state space would now be much larger (maybe $Q = 20,000$), but you have a much more accurate model:

First-order word model. (The words are chosen independently but with frequencies as in English.)

```
REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME
CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO
OF TO EXPERT GRAY COME TO FURNISHES THE LINE
MESSAGE HAD BE THESE.
```

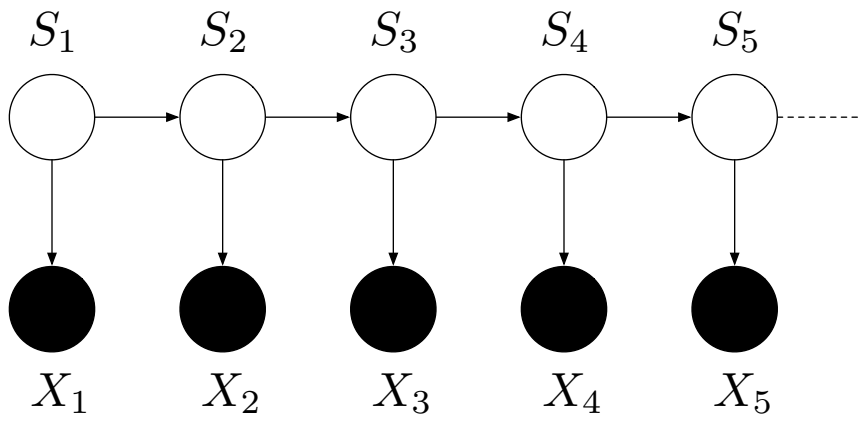
Second-order word model. (The word transition probabilities match English text.)

```
THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH
WRITER THAT THE CHARACTER OF THIS POINT IS
THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE
TIME OF WHO EVER TOLD THE PROBLEM FOR AN
UNEXPECTED
```

Hidden Markov Models

An excellent resource for the material in this section is the paper [Rab89].

Hidden Markov Models (HMMs) are a nice combination of mixture models and Markov models. The idea is that we observe a sequence of random variables/vectors $X_1, \ldots, X_N$ that are governed by an unobserved sequence of states $S_1, \ldots, S_N$ that are Markov. Given S_n, X_n is conditionally independent of everything else. Here is a graph that depicts the structure



Here is a concrete example of when such a model might be useful. In optical character recognition (OCR), we are trying to decode a sequence of images into a letter sequence.

The three revolutions in parametric statistical inference are due to Laplace [148], Gauss and Laplace (1809–1811), and Fisher [67].

We use $p(\cdot)$ generically to denote a frequency function, continuous or discontinuous, and $p(x|\theta)$ to denote a statistical model defined on a given sample space and parameter space. Let $\underline{x} = (x_1, \ldots, x_n)$ denote a sample of n independent observations. From the model we can find the sampling distribution

The letters themselves can be modeled as a Markov chain, with the images as observations of the letters. The images are the vectors X_n, with the states S_n taking one of 27 different values.

A HMM has the following elements:

State space The states S_n can take any of Q different values, which we index by $q = 1, \ldots, Q$. In the OCR example, we would have $Q = 27$ (or slightly larger if we include punctuation).

Marginal distribution on states This is the collection of probabilities

$$p_S[q] = \mathrm{P}\left(S_n = q\right), \quad q = 1, \ldots, Q.$$

In our OCR example, those would be specified by the histogram in the previous section.

Transition matrix This is a $Q \times Q$ matrix $\boldsymbol{T}$ with entries

$$T[q, r] = \mathrm{P}\left(S_n = q | S_{n-1} = r\right).$$

In the OCR example, this would be the (transpose of) the colorful matrix in the previous section. It is possible that this matrix changes with n, but for our discussions here, we will assume that the process is stationary, and so $\boldsymbol{T}$ is constant.

Observation model This is a conditional likelihood function for the observations $\boldsymbol{x}_n$ given the state S_n,

$$f_{X_n}(\boldsymbol{x}_n | S_n = q), \quad q = 1, \ldots, Q.$$

In the OCR example, we might use a Gaussian mixture model for the letter images, so in the case the conditional distribution would be a multivariate Gaussian with known mean vector and covariance matrix. Again, this could in theory change with n, but we will just take it to be constant.

MAP state sequence estimation

We are given a series of observations $X_1 = \boldsymbol{x}_1, \ldots, X_N = \boldsymbol{x}_N$ from a hidden Markov process. Given these observations, we want to

estimate the most likely sequence of states:

$$\mathcal{S}^* = \arg\max_{s_1,\ldots,s_N} \mathrm{P}\left(S_1 = s_1, \ldots, S_N = s_N | X_1 = \boldsymbol{x}_1, \ldots, X_N = \boldsymbol{x}_N\right).$$

We can do this using Bayes rule:

$$\begin{aligned}\mathcal{S}^* &= \arg\max_{s_1,\ldots,s_N} f_{X_1,\ldots,X_N}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N | S_1 = s_1, \ldots, S_N = s_N) \cdot \mathrm{P}\left(S_1 = s_1, \ldots, S_N = s_N\right) \\ &= \arg\max_{s_1,\ldots,s_N} f_{X_1}(\boldsymbol{x}_1 | S_1 = s_1) \cdots f_{X_N}(\boldsymbol{x}_N | S_N = s_N) \\ &\qquad \cdot \mathrm{P}\left(S_1 = s_1\right) \mathrm{P}\left(S_2 = s_2 | S_1 = s_1\right) \cdots \mathrm{P}\left(S_N = s_N | S_{N-1} = s_{N-1}\right)\end{aligned}$$

where the second equality comes from the fact that X_n is conditionally independent from all other X_i given the state S_n, and the Markov property for the state sequence $S_1, \ldots, S_N$.

This optimization program can be solved using the *Viterbi algorithm*[4], which basically consists of a forward and backward pass over the data (this is also known as the *max-product* algorithm). Here is how it works:

Initialize For $q = 1, \ldots, Q$, set

$$\begin{aligned}\delta_1[q] &= \mathrm{P}\left(S_1 = q\right) f_{X_1}(\boldsymbol{x}_1 | S_1 = q) \\ \psi_1[q] &= 0.\end{aligned}$$

Recurse For $n = 2, \ldots, N$, set

$$\begin{aligned}\delta_n[q] &= \left[\max_{1 \le r \le Q} \delta_{n-1}[r] T[q, r]\right] \cdot f_{X_n}(\boldsymbol{x}_n | S_n = q) \\ \psi_n[q] &= \arg\max_{1 \le r \le Q} \left(\delta_{n-1}[r] T[q, r]\right)\end{aligned}$$

[4] Andrew Viterbi's clean solution to this problem was a massive breakthrough in error correcting codes in the 1960s ... it is still used in a wide variety of digital communication devices today. He did not patent this algorithm, but a decade or so later he did co-found Qualcomm and made billions of dollars.

for all $q = 1, \ldots, Q$.

Terminate Set

$$p^* = \max_{1 \le q \le Q} \delta_N[q]$$
$$\hat{s}_N = \arg\max_{1 \le q \le Q} \delta_N[q]$$

We now have the most likely state at the last time $n = N$.

Backtrack We cover the rest of the optimal state sequence for $n = N-1, \ldots, 1$ using

$$\hat{s}_n = \psi_{n+1}[\hat{s}_{n+1}]$$

The Markov structure on the states is what allows this problem to be solved so efficiently. This structure induces a *principle of optimality*, where if a state S_n in the middle of the chain is fixed, then the two state sequences $S_1, \ldots, S_{n-1}$ and $S_{n+1}, \ldots, S_N$ can be solved for independently. During the forward sweep, we are able at index n to say "if the state at $n+1$ is q_0, then I know exactly what the optimal state is at n". The $\psi[q]$ vector above is keeping track of this answer.

References

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.