Hidden Markov Models

Rapid Learning Session 2014

This is intended to be a short and informal introduction to HMMs.

1 Introduction

1.1 What is a hidden Markov model?

In general, a **Markov process** is a random process that generates values with probabilities that are dependent on the state of the system at that time. The system randomly switches between different states according to some *transition probabilities* that are dependent only on the current state of the system - this is known as the **Markov property**. When we look at data generated from a Markov process, we can see an ordered sequence of states (e.g. A, A, A, B, B) paired with values that were generated at that point in time (e.g. 1, 1, 2, 4, 5). We can use this kind of data to calculate the transition probabilities between states and also the probabilities of generating particular values given the state of the system at time t.

In constrast, a **hidden Markov model** is a Markov process with unobserved or "hidden" states - that is, we only see the resulting values generated from the process and are not sure of what the true state of the underlying Markov process is at each given time point. Inferring information about the states of these processes and the probabilities that govern how such processes work has a wide range of applications, particularly for studying sequential data in which we are fairly certain the Markov property is satisfied.

1.2 What kind of data is well modeled by a HMM?

Sequential data is really well modeled by HMMs, espcially when we are interested in assigning a label to a specific segment of the sequence. Examples of this kind of data are time series data or DNA sequence data - the latter of which concerns us most.

1.3 Well-known examples of HMMs in biology

- Gene finding (GENSCAN: `http://www.ncbi.nlm.nih.gov/pubmed/9149143`, `http://genes.mit.edu/GENSCAN.html`)

- Modeling protein sequences and homologs (HMMER: `http://hmmer.janelia.org/`)
- Chromatin state annotation (ChromHMM: `http://www.ncbi.nlm.nih.gov/pubmed/22373907`, `http://compbio.mit.edu/ChromHMM/`)

2 Algorithms

2.1 Formal Definition of a HMM

2.2 Interpretation

2.3 The Viterbi Algorithm

2.4 The Baum-Welch Algorithm

This is the algorithm used to learn the parameters of an HMM given unlabeled training data. It is a special case of the expectation-maximization (EM) algorithm. These types of algorithms work by iterating over two steps: the E-step and the M-step.

In the case of HMMs, if we knew the state that each emission came from, then we could easily infer the emission and transition probabilities (we will do this in Exercise 1). However, in the case of unlabeled data, we do not have this information. The idea is therefore to initialize the parameters with some values then find the most likely state assignment (E-step). With that assignment in hand, we now have labeled data and can use maximum likelihood estimation to update our estimates for the emission and transition parameter values (M-step). We keep iterating between the E- and M-steps until convergence. It is important to note that we may not arrive at an optimal solution. Different initializations of the parameter values can lead to different results.

The inference of the most likely sequence of states (E-Step) is done with the forward-backward algorithm. The forward-backwards algorithm returns a probability distribution over the states for each emission. If you want to use hard EM, you would select the single most likely state (i.e. the state with the highest assigned probability). A better idea is to use soft EM where you keep the soft assignments to states (i.e. the probability distribution).

2.5 Caveats

3 Exercise 1

3.1 Problem Statement

Consider a simple two-state HMM that models the GC content of a DNA sequence. We have a state to represent GC-rich regions and another for regions that are not GC rich.

3.2 Estimating emission probabilities for individual states

3.3 Estimating transition probabilities between states

4 Exercise 2

4.1 Problem Statement

4.2 Estimating HMM parameters with the Baum-Welch algorithm

4.3 Predicting the most likely state sequence using the Viterbi algorithm