Hidden Markov Models in Spoken Language Processing

Björn Johnsson
dat171
Sveaborgsgatan 2b
21361 Malmö
dat02bjj@ludat.lth.se

Abstract

This is a report about Hidden Markov Models, a data structure used to model the probabilities of sequences, and the three algorithms associated with it. The algorithms are the forward algorithm and the Viterbi algorithm, both used to calculate the probability of a sequence, and the forward-backward algorithm, used to train a Hidden Markov Model on a set of sequences, raising the propabilites of these and similar sequences.

1 Introduction

The theories and methods of spoken language processing have evolved much over the last years and the field is one of the most interesting of computer science. This report will explain Hidden Markov Models, a graph-based data-structure used to model and calculate probabilities of sequences. The Hidden Markov Model is used in almost every speech-recognition environment.

2 The Model

2.1 The Markov Chain

The Hidden Markov Model data-structure is based on a data-structure called the Markov Chain. The Markov Chain is a directed, weighted graph, where each node contains a symbol from the output alphabet to wich it is applied, and an initial probability. The graph is complete i.e all nodes have vertices to all nodes, including itself. The weight of each vertice is the probability of a transition. Given a Markov Chain and a sequence it is then easy to calculate the probability of the given sequence by taking the product of the initial propablity of the node associated with the first symbol of the sequence and all the transition-propabilites to the nodes associated with the following symbols in the sequence. That is, for sequence S with of length n, the propabilite of sequence S is

$$prop(S) = init(S_1) * \prod_{i=2}^{n} (S_{i-1}|S_i)$$

where $(i|j)$ is the transition-probability from state i to state j.

2.2 The Hidden Markov Model

In the Hidden Markov Model there is no one-to-one relation between the alphabet and the nodes as in the Markov Chain, instead each node contains every symbol in the alphabet and relates each symbol with a probability. As a consequence there is no longer a ”true” path through the graph corresponding to a sequence, as there often exists several possible paths. Therein lay one of the problems of the Hidden Markov Model, as the evaluation no longer is as simple as in the Markov Chain. In most cases is there even O^t possible ways through the graph corresponding to the same sequence, making the calculations extensive.

3 Algorithms

In the Hidden Markov Model there are basically two problems, the evaluation and the training. The evaluation problem is solved by two different algorithms giving different probabilities of a sequence. The first, the forward algorithm, gives the sum of the probabilities of all possible paths through the graph. The Viterbi algoritm gives the probability of the path with the highest probability. The training is performed by the forward-backward algorithm, wich trains the Hidden Markov Model to give a set of similar sequences a higher probability.

3.1 The Forward Algorithm

The forward algorithm calculates the probability of a sequence by adding up the sum of probabilities of all possible paths giving the right outputsequence. To do this in an easy way I first define the forward probability.

3.1.1 The forward Probability

Given a Hidden Markov Model, Φ, and a sequence X, it is possible to calculate the forward probability α. $\alpha_t(j)$ is defined as the probability of being at node j at time t giving the output in the sequence. Using a recursion this is fairly easy to calculate using the formula:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}\right] b_j(X_t)$$

$$\alpha_1(i) = \pi_i b_i(X_1)$$

3.1.2 The Final Step

Having calculated the forward probability the sum of all probabilities is easy to calculate.

$$P(X|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$$

This gives that $P(X|\Phi)$ is the sum over all nodes, of being at that node after sending out the output given in the sequence, ie the sum of the probabilities over all possible path in the graph.

3.2 The Viterbi Algorithm

The Viterbi algorithm calculates the probability and the nodesequence of the most likely traversion through the graph giving the expected outputsequence. This is implemented in my Hidden Markov Model but I have not investigated it more closely as I rarely use it.

3.3 The Forward-Backward Algorithm

The forward backward algorithm trains the Hidden Markov Model by raising the probability of the given sequence and thereby all sequences similar to this. To calculate the new values for a and b, labeled $\hat{a}$ and $\hat{b}$, I first define two new probabilities, the backward probability, β, and the transition probability, γ.

3.3.1 The backward probability

Just as I defined the forward probability, I can also defin the backward probability $\beta_t(i)$ that is, the probability that after being at node i at the time t, the model outputs the sequence $X_{t+1}...X_T$. Similar to the forward probability, this can be calculated using a recursion as follows.

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right]$$

$$\beta_T(i) = \frac{1}{N}$$

3.3.2 The transition probability

The transtion probability $\gamma_t(ij)$ is the probability of taking the transition from node i to node j at time t given a Hidden Markov Model and an outputsequence. Using the prior defined forward and backward-probabilities it can be calculated as follows.

$$\gamma_t(ij) = \frac{\alpha_{t-1}(i)a_{ij}b_j(X_t)\beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}$$

This is interpreted as the forwardprobability of being at node i after giving the output $X_1...X_{t-1}$ multiplied by the probability of taking the transition to node j and there give the output X_t multiplied by the probability of going from node j and give the output $X_{t+1}...X_T$ and dividing the hole product by the sum of all possible paths giving that outputsequence.

3.3.3 Calculating $\hat{a}$

To calculate $\hat{a}_{ij}$, the new values meant to replace the prior value at a_{ij},Itake the sum of all transitions between node i and node j, at all possible times t and divide it with the sum of all possible transitions from node i at all possible times t.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(ij)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(ik)}$$

3.3.4 Calculating $\hat{b}$

It is possible to calculate $\hat{b}_j(k)$ in a similar way as the sum of all transitions to node j if $X_t = O_k$ and dividing it by the sum of all transitions to node j.

$$\hat{b} = \frac{\sum_{t \in (x_t = O_k)} \sum_{i=1}^{N} \gamma_t(ij)}{\sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(ij)}$$

3.3.5 Trainging on a set of sequences

These calculations can easy be extended to train on a set of sequences instead of a single sequence. To do soIadd another dimension to the γ so that $\gamma_t^m(ij)$ is the probability of going from node i to node j at time t given the m'th sequence of the trainingset. he calculations are still the same. Howether, the calculations for $\hat{a}$ and $\hat{b}$ has to be changed, so they are based on the hole set. $\hat{a}$ and $\hat{b}$ should instead be calculated as follows.

$$\hat{a}_{ij} = \frac{\sum_{m=1}^{M} \sum_{t=1}^{T} \gamma_t^m(ij)}{\sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t^m(ik)}$$

$$\hat{b} = \frac{\sum_{m=1}^{M} \sum_{t \in (x_t = O_k)} \sum_{i=1}^{N} \gamma_t^m(ij)}{\sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t^m(ij)}$$

3.3.6 Post calculations

After calculating $\hat{a}$ and $\hat{b}$ you just exchange a and b by them. In theory. On computers this will be a problem as almost all evaluation then will become zero as the values go beyond the reach of a double floating point variable. To cope with that, the easiest way is to define that the values in a and b are not allowed to go beneath a value, in my case 1e-20. As computers calculating values of this size almost always does errors I normalize the values in a and b so that the probabilities sum up to one.

3.3.7 Result of the forwardbackward algorithm

Using this algorithm on fairly large traingingset, it can train a Hidden Markov Model in about five iterations and after that recognize sequences similar to those in the trainingset.

4 Appendix

4.1 Notation

The following notations are used in the calculations:
N is the number of nodes in the Hidden Markov Model
O is the number of symbols in the outputalphabet
T is the number of symbols in a particular outputsequnce
a_{ij} is the transitionprobability between node i to node j
$b_i(x)$ is the probability of the output x at node i
X_t is the output at time t in sequence X
π_i is the initial probability of node i, ie the probability that the sequence starts at node i
$\alpha_t(i)$ is the forward probability defined in chapter 3.1.1
$\beta_t(i)$ is the backward probability defined in chapter 3.3.1
$\gamma_t(ij)$ is the transition probability defined in chapter 3.3.2
$\hat{a}_{ij}$ is the new transitionprobability used temporary in the training, defined in chapter 3.3.3
$\hat{b}_{ij}$ is the new outputprobability used temporary in the training, defined in chapter 3.3.4
M is the number of sequences in a trainingset

4.2 The use of Hidden Markov Models in Speech Recognition

Speech and sound is in computers often represented as PCM data or Pulse CodeModulation. It is a sequence of values representing the porition of the membran on either the speaker or the microphone. Unfortunatly this sequence contains to much data to be evaluated by a Hidden Markov Model so it has to be transformed before it is used this data structure. Usually the PCM data is divided in frames of approxomatly 20 milliseconds and each frame the converted to single number or a small vector of numbers. There are multipla ways to do this step.

- Fast Fourier Transfom
  A Fast Fourier Transform, or a FFT, is a fast but inaccurate way of calculating the energy of the sound at different frequences. Picking the numbers of the right frequences, this is a good way of turning the PCM into a useful value.

- Linear Prediction Coding
  The Linear Prediction Coding, or the LPC,

is a way of creating a polynomial that, given the prior values of the data, try to predict the coming value.

- Energy
  Adding up the absolute value of the PCM data gives the total energy of a frame. This is a useful as such, but if one take the difference between the frames insted hte values give a better representation of the sound. Especially if all the frames energies are subtracted with the energy of the frame with the highest energy, thus removing the possible error of different recording volumes.

After this is done, the values are made discrete using a codebook to spread the values over all the discrete symbols in the output alphabet. This gives us a sequence of symbols, in my case, integer between 0 and 255. It is then possible to do this for several recordings of the same command, getting a set of sequences that can be used to train a Hidden Markov Model. The Hidden Markov Model can the be used to calculate the probability of sequence extracted from a sound recording i the same way as the trauning set, and using a treshold, determine whether the sound was the command or not. Using several Hidden Markov Models, one per command, this can be used to controll a computer using speech commands.

5 References

Huang, Acero and Hon. 2001. Spoken Language Processing, A Guide to Theory, Algorithm, and System Development. Prentice Hall PTR. ISBN 0-13-022616-5