Proc. of the 12th IAPR Int. Conf. on Pattern Recognition, IEEE Computer Society Press, 1994, p. 140-144

Hidden Markov Models for Labeled Sequences

Anders Krogh
Electronics Institute, Building 349
Technical University of Denmark, 2800 Lyngby, Denmark
Email: krogh@nordig.ei.dth.dk

Abstract

A hidden Markov model for labeled observations, called a CHMM, is introduced and a maximum likelihood method is developed for estimating the parameters of the model. Instead of training it to model the statistics of the training sequences it is trained to optimize recognition. It resembles MMI training, but is more general, and has MMI as a special case. The standard forward-backward procedure for estimating the model cannot be generalized directly, but an "incremental EM" method is proposed.

1 Introduction

Hidden Markov Models (HMMs) are often used to model the statistical structure of a set of observations like speech signals [12]. A model is estimated so as to maximize the likelihood of the observations or, in a Bayesian setting, the *a posteriori* probability of the model. Often a set of different models are estimated independently, for instance one model for each word in a small vocabulary speech application. After estimation they are used for *discrimination*, although they were not trained for discrimination. Two algorithms have been developed for *discriminative training*. In the maximum mutual information (MMI) method the mutual information between the models is maximized [2, 11, 5, 7], whereas the similar method of corrective training is based on a heuristic derivation [3]. It has been shown that both methods yield better results than standard maximum likelihood (ML) training on realistic speech recognition problems.

In this paper a novel type of HMM for *labeled* data is proposed. It is an HMM in which each state also contains a probability distribution over class labels, and it is called a class HMM (CHMM). In a speech signal the labels could be the words and the CHMM could be a multiple word model that is trained on whole spoken sentences. One could (in principle at least) train an entire language HMM with phone and word submodels on continuous speech. Another possible application is for modeling DNA or proteins [8, 9]. In protein modeling the primary data is the sequence of amino acids and the labels could be for instance the type of three-dimensional structure (alpha helix, beta sheet or coil).

In the simplest setup the CHMM consists of states with assigned classes. In speech modeling it means that a word is represented by a group of states that are assigned to this word. The aim is that a decoding of an observation sequence yields the correct labels. That is, the objective of the training is to optimize the probability of the labeling, Prob(labels|model,observations), rather than the probability of the observations Prob(observations|model) which is optimized in standard HMM training. In terms often used in the neural network community, one can view the usual ML training of HMMs as *unsupervised training* and the new training method as *supervised training* [6].

The extension of an HMM to a CHMM is very simple. The CHMM has two modes. In the decoding or recognition mode an unlabeled sequence of observations has to be labeled, and it works exactly like a standard HMM. To decode a spoken sentence the most probable path through the model is found by the Viterbi algorithm or the most probable words are found by the standard forward algorithm [12]. In the other mode, which is used during training, the labels of the observations are taken into account, and the CHMM is treated as an HMM with states that produce both the observation symbols *and* the class labels. In order to calculate the probability of a sequence of observations with associated labels, both the probability of the observations and the labels are taken into account.

The two modes are much like the "clamped" and the "free" modes of a Boltzmann machine [1, 6], and the reestimation of the model is also similar to Boltzmann learning: A path through the model that gives the correct labels contribute positively to the reestimated parameters, whereas a path giving wrong labels has a negative contribution.

MMI estimation comes out as a special case of CHMMs. As for MMI, the standard forward-backward estimation procedure does not generalize directly to CHMM. Some possible estimation methods based on gradient descent will be discussed in this paper.

2 Modeling labeled data with a class HMM

A CHMM models observations with associated class labeling. The basis is a standard HMM with observation symbol emission probabilities for each state and probabilities for transitions between states. The probability of emitting symbol a in state i is called $b_i(a)$ and the probability for making a transition from state i to j is called a_{ij}. A null state is used as begin state, so the usual initial state probabilities (called π in [12]) are replaced by transition probabilities from the begin state to the other states. That is, if the begin state is state number 0, the probability of starting in state i is a_{0i}. All the parameters specifying the model are placed in the vector θ for convenience, and the total number of these parameters is called N. Discrete models in which the observations are symbols from a finite alphabet are considered here, but it can easily be generalized to continuous observations.

A state can also emit a class label from a probability distribution associated with the state. These class membership probability parameters are called ϕ, and the probability of a label x in state i is $\phi_i(x)$. Often a particular state i models only *one* class, say class y, which means that $\phi_i(y) = 1$ and $\phi_i(x) = 0$ for $x \neq y$. Sometimes, however, it is desirable to assign a probability to several classes. In speech recognition, for instance, a state in between two regions of the model assigned to two different phones could represent both phones with some probability, or some states could be shared by two or more similar phones. In many applications class membership probabilities will be fixed *a priori*, but they can also be estimated just like the θs.

Given a model (θ, ϕ) one can calculate the usual quantities for the underlying HMM characterized by θ. For instance the probability of a sequence of observation symbols s, $P(s|\theta)$, is calculated by the forward algorithm, and the most probable path through the model can be found by the Viterbi algorithm [12].

Since one is interested in finding the most probable labeling of an unlabeled sequence, an interesting quantity is the probability of a labeling for a given sequence

$$P(c|s, \theta, \phi) = P(s, c|\theta, \phi)/P(s|\theta), \tag{1}$$

where c denotes the sequence of labels associated with s ($P(s|\theta, \phi) = P(s|\theta)$ by construction). $P(s|\theta)$ can be found by the standard forward algorithm, as mentioned above. The class probabilities ϕ are used to calculate the other quantity $P(s, c|\theta, \phi)$, *i.e.* the probability of a sequence s with associated labels c, by a straightforward generalization of the forward algorithm. In fact, it is precisely the forward algorithm for an HMM in which state i emits *pairs* of symbols with the probability $\tilde{b}_i(a, x) = b_i(a)\phi_i(x)$ for state i.

The most probable labeling of a sequence s,

$$c^* = \operatorname*{argmax}_c P(c|s, \theta, \phi), \tag{2}$$

can not be found by the Viterbi algorithm. The reason is that several paths through the model will give the same labeling and thus contribute to this probability. It is possible to calculate c^* by an algorithm with much worse time complexity than the Viterbi algorithm, but here it will be assumed that the labeling found from the Viterbi path is a good approximation to c^*.

3 Estimating the model

For training the model it is assumed that a number of observation sequences s^μ and associated labels c^μ are available. For simplicity the superscript μ is dropped, which corresponds to only considering one training sequence. Generalizing to several sequences is straightforward. Thus we have

observations	s:	$s_1, s_2, \ldots, s_n$
labels	c:	$c_1, c_2, \ldots, c_n$

The labels are only known for the training set. Since the object is to find a model that assigns correct labels to a sequence, the ML parameters $\hat{\theta}$ are found as $\hat{\theta} = \operatorname{argmax}_\theta P(c|s, \theta, \phi)$. Using equation (1):

$$\begin{aligned} \hat{\theta} &= \operatorname*{argmax}_\theta P(c|s, \theta, \phi) && (3) \\ &= \operatorname*{argmax}_\theta \frac{P(c, s|\theta, \phi)}{P(s|\theta)} && (4) \end{aligned}$$

(A Bayesian MAP approach is a possible alternative to this ML method.)

To do the optimization in (4), we first convert to logarithms. Define

$$\begin{aligned} \mathcal{L}_f &= -\log P(s|\theta) && (5) \\ \mathcal{L}_c &= -\log P(c, s|\theta, \phi). && (6) \end{aligned}$$

The subscripts c and f means "clamped" and "free" in analogy with the Boltzmann machine. Then, to find $\hat{\theta}$ in (4) we have to minimize

$$\mathcal{L} = -\log \frac{P(c, s|\theta, \phi)}{P(s|\theta)} = \mathcal{L}_c - \mathcal{L}_f. \tag{7}$$

This minimization can be done in various ways, but most are based on the gradient of $\mathcal{L}$ in one way or the other.

4 Calculation of the gradient

First the gradient of the negative log likelihood of the observations given the model ($\mathcal{L}_f$) will be found. This is a standard calculation, but is included for completeness and to establish notation.

The probability $P(s|\theta)$ is written as a sum over all possible paths through the model,

$$P(s|\theta) = \sum_{\pi} P(s, \pi|\theta), \tag{8}$$

where π denotes a path and the sum is over all possible paths. The probability for a given path is

$$P(\pi, s|\theta) = \prod_{i=1}^{N} \theta_i^{n_i(\pi,s)} \tag{9}$$

where $n_i(\pi, s)$ is the number of times the parameter θ_i is used in the path π for observation sequence s. Remember that θ is a mixture of transition probabilities and letter emission probabilities. If for instance the letter "a" is emitted twice in a particular state the corresponding emission probability, say θ_k, contributes to the probability twice, so $n_k(\pi, s) = 2$. The derivative of the above with respect to a parameter θ_k is

$$\begin{aligned} \frac{\partial P(\pi, s|\theta)}{\partial \theta_k} &= \frac{n_k(\pi, s)}{\theta_k} \prod_{i=1}^{N} \theta_i^{n_i(\pi,s)} \\ &= \frac{n_k(\pi, s)}{\theta_k} P(\pi, s|\theta). \end{aligned} \tag{10}$$

Now it is a fairly simple matter to find the gradient of the log likelihood

$$\begin{aligned} \frac{\partial \mathcal{L}_f}{\partial \theta_k} &= -\sum_{\pi} \frac{n_k(\pi, s)}{\theta_k} \frac{P(\pi, s|\theta)}{P(s|\theta)} \\ &= -\sum_{\pi} \frac{n_k(\pi, s)}{\theta_k} P(\pi|s, \theta) \\ &= -\frac{n_k(s)}{\theta_k}. \end{aligned} \tag{11}$$

Here $n_k(s)$ is the expected number of times the kth parameter is used by the observation s, *i.e.*, the average of $n_k(\pi, s)$ over all possible paths. Bayes theorem was used for $\frac{P(\pi,s|\theta)}{P(s|\theta)} = P(\pi|s, \theta)$.

The other term, $\mathcal{L}_c$, can be calculated in a similar manner. First,

$$P(\pi, c, s|\theta, \phi) = \prod_{i=1}^{N} \theta_i^{n_i(\pi,s)} \prod_{j=1}^{M} \phi_j^{q_j(\pi,c,s)} \tag{12}$$

where $q_j(\pi, c, s)$ is the number of times the parameter ϕ_i is used in the path π for observation sequence s with associated labels c. Proceeding as before,

$$\frac{\partial}{\partial \theta_k} P(\pi, c, s|\theta, \phi) = \frac{n_k(\pi, s)}{\theta_k} P(\pi, c, s|\theta, \phi). \tag{13}$$

Then the gradient of the log likelihood is calculated as in (11)

$$\begin{aligned} \frac{\partial \mathcal{L}_c}{\partial \theta_k} &= -\sum_{\pi} \frac{n_k(\pi, s)}{\theta_k} \frac{P(\pi, c, s|\theta, \phi)}{P(c, s|\theta, \phi)} \\ &= -\sum_{\pi} \frac{n_k(\pi, s)}{\theta_k} P(\pi|c, s, \theta, \phi) \\ &= -\frac{m_k(c, s)}{\theta_k}. \end{aligned} \tag{14}$$

Now $m_k(c, s)$ is the expected number of times the kth parameter is used by the sequence of observations s and giving the correct labeling c.

Finally the derivative of the total log likelihood $\mathcal{L}$ is

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\frac{m_k(c, s) - n_k(s)}{\theta_k}. \tag{15}$$

To get a more intuitive understanding of n and m, consider the simple case where the label probabilities are 0 or 1, *i.e.*, any given state belongs to only one class — it can only emit one label. Then the probability of any path in which the labels of the observations do not fit the labels of the states is zero. Any path where the labels fit the states is called a permissible path. So the ms are the expected number of times the parameters are used in permissible paths, and the ns the expected number of times the parameters are used in *any* paths. In this case it is also easy to see how to calculate the ms. They are just calculated by the usual forward and backward algorithms, where all paths are restricted to permissible ones, *i.e.*, all the forward variables and all backward variables (called α and β in [12]) corresponding to non-permissible paths are set to zero. The ns are found by the standard forward-backward algorithm.

In the more general case, the ms are calculated by a forward-backward algorithm in which each state is viewed as emitting a two character observation, as was already mentioned earlier.

5 Algorithms for parameter estimation

The forward-backward algorithm is a so-called EM method (EM means Expectation-Maximization). It is fairly easily shown that it does not work for the CHMM. A straightforward generalization of the standard reestimation formulas would give

$$\theta_k = \frac{m_k(c,s) - n_k(s)}{\sum_{i \in S(k)} (m_i(c,s) - n_i(s))}, \tag{16}$$

where $S(k)$ denotes the set of parameters in the model that has to sum to one and includes θ_k. This formula may (and almost certainly will) lead to negative probabilities, and therefore an alternative method is proposed.

A version of EM that incrementally updates the model on a subset of the training data is discussed in [10]. Here a heuristic incremental algorithm is suggested. It is based on the idea of a running average implemented by a kind of "momentum" method often used in incremental neural network learning. Assume p observation sequences s^μ, $\mu = 1, \ldots, p$, are available. Then the reestimation formula above would change to

$$\theta_k = \frac{\sum_\mu (m_k^\mu - n_k^\mu)}{\sum_\mu \sum_{i \in S(k)} (m_i^\mu - n_i^\mu)}, \tag{17}$$

where the c and s dependency is implicit.

The contribution to this sum from each sequence is

$$\frac{m_k^\mu - n_k^\mu}{\mathcal{N}(k)} \tag{18}$$

where $\mathcal{N}(k)$ is the normalization in the above formula. Thus, an obvious incremental algorithm is

$$\theta_i^{new} = \mathrm{Normalize}(\theta_i^{old} + \alpha(m_k^\mu - n_k^\mu)), \tag{19}$$

where "Normalize" is a normalization operator. The added term will usually be smaller than θ in size, and it is easy to built in security checks to avoid negative probabilities. The observation sequences are used to iteratively update the parameters by this formula until they stabilize. Obviously the size of the parameter α may be important.

This update rule is an incremental version of the "extended Baum-Welch" algorithm [11], which could also be used in this context. Apart from these EM-like methods, there are several possible gradient descent type algorithms, see *e.g.* [4, 7]. Based on the experiments in [7] one might expect those to be less efficient, however.

6 Estimation of class probabilities

In many applications the class of each submodel would be fixed in advance. In speech for instance one might design a submodel for each word in the vocabulary, and these assignments would not change. In some instances, however, it might be desirable to also estimate the class probabilities for the whole model or for parts of the model. One might have states between the word submodels that does not belong to one word, but rather has a probability of belonging to any of the words — these probabilities could then be estimated from the data.

The estimation of the class probabilities is done as standard ML estimation of an HMM:

$$\hat{\phi}_k = \underset{\phi}{\mathrm{argmax}}\, P(c|s,\theta,\phi) = \underset{\phi}{\mathrm{argmin}}\, \mathcal{L}_c, \tag{20}$$

where the last equality comes from the fact that $P(s|\theta)$ (or $\mathcal{L}_c$) does not depend on ϕ. Starting from equation (12) one finds

$$\frac{\partial}{\partial \phi_k} P(\pi, c, s|\theta, \phi) = \frac{q_k(\pi, c, s)}{\phi_k} P(\pi, c, s|\theta, \phi). \tag{21}$$

Proceeding as above, this will eventually lead to

$$\frac{\partial \mathcal{L}}{\partial \phi_k} = -q_k(c,s)/\phi_k \tag{22}$$

where $q_k(c,s)$ is the expected number of times the parameter ϕ_k is used for observation sequence s with associated labels c. These qs can again be calculated by a forward-backward procedure the same way as the ms are calculated, and it can be done at the same time as the ms are calculated.

7 Recovering MMI estimation

In MMI estimation it is assumed that a number of submodels that each represent a class (*e.g.* a word) is combined to a larger model with *fixed transition probabilities between submodels* (or rather these probability parameters are determined independently of the submodels).

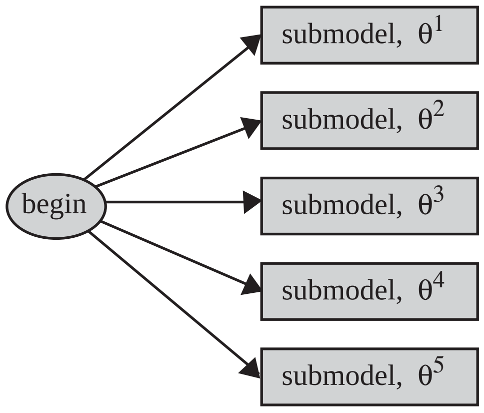

Figure 1: A CHMM that gives an MMI recognition system. Each submodel models one class that could for instance be a word.

Assume there are K submodels with parameters $\theta^1 \dots \theta^K$ that represent K different classes $1, \dots, K$. These submodels make up a larger CHMM as shown in Figure 1. Assume also that a training sequence belongs to one class l, *i.e.*, all observation symbols in a sequence have the same label l. The probability of the observations with associated labels is then zero for all submodels except submodel l, $P(c, s|\theta, \phi) = P(s|\theta^l)$. Then the reestimation according to equation (4) is

$$\begin{aligned}\hat{\theta} &= \underset{\theta}{\operatorname{argmax}} \frac{P(c,s|\theta,\phi)}{P(s|\theta)} = \underset{\theta}{\operatorname{argmax}} \frac{P(s|\theta^l)}{\sum_{k=1}^K P(s|\theta^k)} \\ &= \underset{\theta}{\operatorname{argmax}} \left[\log P(s|\theta^l) - \log(\sum_{k=1}^K P(s|\theta^k)) \right]. \quad (23)\end{aligned}$$

This is exactly the MMI reestimation formula [12].

8 Conclusion

A hidden Markov model for labeled observations, called a CHMM, was introduced and a maximum likelihood method for supervised training developed. Instead of training it to model the statistics of the training sequences it is trained to optimize recognition. It resembles MMI training, but it is more general, and has MMI as a special case. The standard forward-backward procedure for estimating the model cannot be generalized directly, but an "incremental EM" method was proposed.

In the future the method will be tested on biological sequences (DNA and proteins) and on problems in speech recognition.

Acknowledgements

Numerous discussions with David Haussler are greatfully acknowledged. I would also like to thank Ian Koch, Yoram Singer, and Manfred Warmuth for comments and suggestions.

References

[1] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] L.R. Bahl, P Brown, P.V. de Souza, and R.L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE ICASSP*, pages 49–52, 1986.

[3] L.R. Bahl, P Brown, P.V. de Souza, and R.L. Mercer. Estimating hidden Markov model parameters so as to maximize speech recognition accuracy. *IEEE Trans. on Speech and Audio Processing*, 1:77–83, 1993.

[4] P. Baldi and Y. Chauvin. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6(2), 1993.

[5] R Cardin, Y Normandin, and R De Mori. High performance connected digit recognition using maximum mutual information estimation. In *Proc. ICASSP*, pages 533–536, 1991.

[6] D. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.

[7] S. Kapadia, V. Valtchev, and S.J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proc. ICASSP*, 1993.

[8] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.

[9] A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in *e. coli* DNA. Tech. Rep. UCSC-CRL-93-33, Computer and Information Sciences, Univ. of California at Santa Cruz, 1994.

[10] Radford M. Neal and Geoffrey E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Preprint, Dept of Computer Science, Univ. of Toronto, 1993.

[11] Y Normandin and S D Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proc. ICASSP*, pages 537–540, 1991.

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