e-PGPathshala

Subject : Computer Science

Paper: Machine Learning

Module: HMM – EM Algorithm

Module No: CS/ML/36

Quadrant I – e-text

Welcome to the e-PG Pathshala Lecture Series on Machine Learning.In this module we will discuss the solution to the learning problem associated with Hidden Markov Model.

Learning Objectives:

The learning objectives of this module are as follows:

- To understand the concept of Model Learning in HMM
- To discuss the Expectation Maximization method for parameter estimation
- To understand parameter estimation using supervised and unsupervised techniques

36.1 Recap: Hidden Markov Model

As we have already discussed a Hidden Markov Model is an extension of a Markov model in which the input symbols are not the same as the states.This means we don't know which state we are in (hence called Hidden State). For example in HMM POS-tagging, the input symbols are the words, states are the part of speech tags. In addition we make two impittant and the sattes **Markov assumption** states that the state transition depends only on the origin and destination and the **Output-independent assumption** which s all observation frames are dependent on the state that generated them, not on neighbouring observation frames.

36.1.1 The Three Problems of HMM

The following are the three problems associated with HMM:

Problem 1: Evaluation

Here given the observation sequence $O=o_1,\ldots,o_T$and an HMM model how do we compute the probability of O given the model?

Q1: **How do we compute the probability of a given sequence of observations?**

A1: Forward – Backward dynamic programming algorithm – **the Baum Welch algorithm**

Problem 2: Decoding

Here given the observation sequence $O=o_1,\ldots,o_T$and an HMM model, how do we find the state sequence that best explains the observations?

Q2:How to compute the most probable sequence of states, given a sequence of observations?

A2: Viterbi's dynamic programming Algorithm

Given an observation sequence, compute the most likely hidden state sequence

We have already the solutions to the above two problems in the previous module.

Problem 3 – Learning – this is the problem we will be now discussing

36.2 Problem 3: Learning

Given an observation sequence and set of possible models, which model most closely fits the data?How do we adjust the model parameters

$$\lambda = (A, B, \pi)$$

to maximize

$$P(O \mid \lambda)$$

Q3: Given an observation sequence and set of possible models, which model most closely fits the data?

A3: The Expectation Maximization (EM) heuristic.

It is this learning problem that we will be discussing in this module. This is the hardest problem to solve.

Given an observation sequence O = (O_1 O_2 … O_L), and a class of models, each of the form M = {A,B,p}, which specific model "best" explains the observations?A solution to the above question enablesthe efficient computation of P(O|M) (the probability that a specific model M produces the observation O). This can be viewed as a learning problem: We want to use the sequence of observations in order to “train” an HMM and learn the optimal underlying model parameters essentially we are learning the transition and observation probabilities.

In other words we need to estimate the emission and transition probabilities given observations and assuming that hidden states are observable during the learning process. We need to estimate emission and transition probabilities given observations only.

Up to now we’ve assumed that we know the underlying model. However often these parameters are estimated on annotated training data, which gives rise to two drawbacks, basically annotation is difficult and/or expensive and training data is different from the current data. We want to maximize the parameters with respect to the current data, i.e., we’re looking for a model λ', such that

$$\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$$

36.3 Parameter Estimation

We first make an initial guess for the transition probabilities $\{a_{ij}\}$ and observation or emission probabilities $\{b_{km}\}$. We need to then compute the probability that one hidden state follows another given the guessed values of $\{a_{ij}\}$ and $\{b_{km}\}$ and sequence of observations which are computed using forward-backward algorithm. We then compute the probability of observed state given a hidden state and the guessed values of $\{a_{ij}\}$ and $\{b_{km}\}$ and the sequence of observations again computed using the forward-backward algorithm. Now using these computed probabilities we make an improved guess for the transition and observation probabilities $\{a_{ij}\}$ and $\{b_{km}\}$. We then repeat this process until convergence. It can be shown that this algorithm does in fact converge to correct values for {aij} and {bkm} assuming that the initial guess was close enough.

That is given an observation sequence, find the model that is most likely to produce that sequence. Here we use no analytic method. We just assume a model and given a model and observation sequence, we update the model parameters to better fit the observations.

36.3 Review of Probabilities

Let us now review the different probabilities that we have discussed in connection with HMM.

- Forward probability: $\alpha_t(i)$

  The probability of being in state s_i, given the partial observation $o_1,\dots,o_t$

- Backward probability: $\beta_t(i)$

  The probability of being in state s_i, given the partial observation $o_{t+1},\ldots,o_T$

- Transition probability: $\xi_t(i,j)$

  The probability of going from state s_i, to state s_j, given the complete observation $o_1,\ldots,o_T$

- State probability: $\gamma_t(i)$

  The probability of being in state s_i, given the complete observation $o_1,\ldots,o_T$

Now with these basic probabilities we will discuss how the parameters of the model are estimated.

36.4 Parameter Estimation

The probability of moving from one state to another:

$$p_t(i,j) = \frac{\alpha_i(t)\, a_{ij}\, b_{jo_{t+1}}\, \beta_j(t+1)}{\sum_{m=1\ldots N} \alpha_m(t)\, \beta_m(t)}$$

and the probability of being in state i

$$\gamma_i(t) = \sum_{j=1\ldots N} p_t(i,j)$$

With the above probabilities we can recalculate the initial probability, and the transition and emission probabilities.

$$\hat{\pi}_i = \gamma_{\mathrm{i}}(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} p_t(i, j)}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\hat{b}_{ik} = \frac{\sum_{\{t:o_t=k\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_i(t)}$$

36.5 Expectation Maximization

The forward-backward algorithm is an instance of the more general EM algorithm where

- The E Step: where we compute the forward and backward probabilities for a given model
- The M Step where we re-estimate the model parameters

Use the forward-backward (or Baum-Welch) algorithm, which is a hill-climbing algorithm

Using an initial parameter instantiation, the forward-backward algorithm iteratively re-estimates the parameters and improves the probability that given observations are generated by the new parameters.Three parameters need to be re-estimated:

- Initial state distribution:
- Transition probabilities: $a_{i,j}$
- Emission probabilities: $b_i(o_t)$

Re-estimating Transition Probabilities

Here we need to determine the probability of being in state s_i at time t and going to state s_j, given the current model and parameters as follows:

$$\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

Now we can calculate the probability as given in Figure 36.2.

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

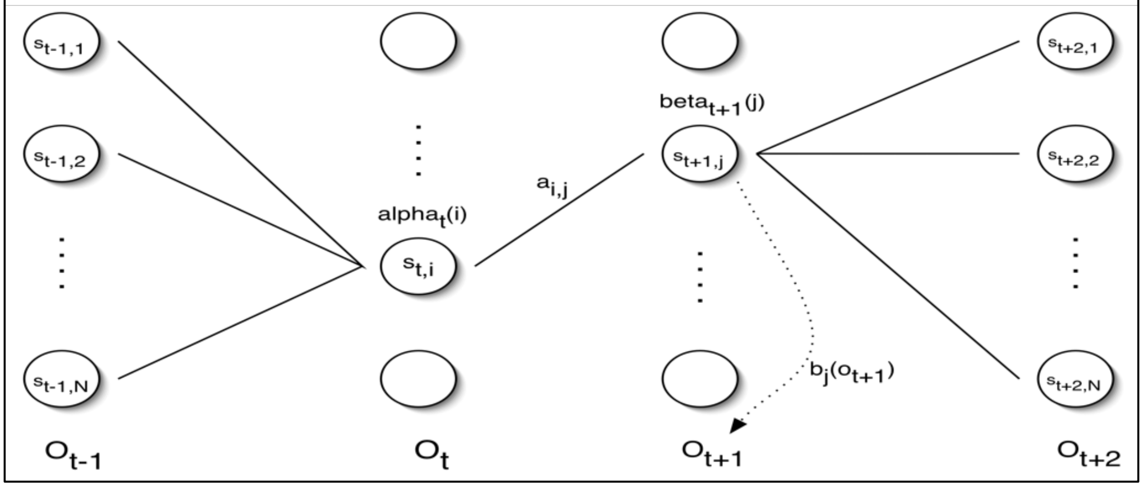

Figure 36.2 Trellis Diagram

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}$$

Here the probability is based on previous values of initial, transition and emission probabilities.

In other words **the probability ξ_t (i,j) between the states i, j at time slice t** is the ratio of the product of the forward probability of state i at time slice t,
the transition probability between states i and j,
the emission probability of observation symbol o_{t+1}at time slice t+1, at state j,
and the backward probability of state j at time slice t+1 and
the sum of the products of the terms explained above considered for all N values of i and j.

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i}$$

Formally

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{j'=1}^{N} \xi_t(i,j')}$$

Defining

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

as the probability of being in state s_i, given the complete observation O.

We can say:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Re-estimating Initial State Probabilities

Initial state distribution: $\pi_i$is the probability that s_i is a start state

Re-estimation is easy:

$$\hat{\pi}_i = \text{expected number of times in state s}_i \text{ at time 1}$$

Formally:

$$\hat{\pi}_i = \gamma_1(i)$$

36.6 Explaining EM algorithm with an Example

First let us consider a document corpus as an example. The problem we are trying to solve is tagging of the words of a sentence. An example of such a tagging problem is the Part of Speech (POS) tagging where each word in the sentence is tagged with a POS.

Example The man eats the sweet mango.

Det Noun Vrb Det ADJ Noun

In the POS tagging problem we want to find the tag sequence $t_1 \ldots t_n$with the highest probabilityof the word sequence$w_1 \ldots w_n$. Here we need to estimate probabilities. Here we need to determine transition probability, emission probability, carry out some type of smoothing since we need to deal with unknown words. The transition probability $P(t_i|t_{i-1})$is the probability of tag t_i following tagt_{i-1}. The emission probability is the probability of a tag emitting a particular word.

The general idea is as follows:

Start by devising a noisy channel, any model predicts the corpus observations via some hidden structure (tags, parses, …). Initially a guess is made about the parameters of the model. It is best to make an educated guess but a random guess can also work.

As in any EM algorithm we repeat the following two steps until convergence:

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure
- **Maximization step:** Use that hidden structure (and observations) to re-estimate parameters

Now let us go back to our HMM problem. We want to find out of from all the possible sequences of n tags $t_1 \ldots t_n$, the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n)$$

Hat ^ means "our estimate of the best one" and Argmax_x f(x) means "the x such that f(x) is maximized".

This equation is guaranteed to give us the best tag sequence. But how to make it operational? How to compute this value? We use the intuition of Bayesian classification where we use Bayes rule to transform into a set of other probabilities that are easier to compute as follows:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(w_1^n | t_1^n) P(t_1^n)$$

Now we would like to define likelihood and prior as follows:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \; \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Therefore we find the best tag sequence (POS sequence) for a given sequence of words as the sequence that gives the maximum probability of the product of likelihood (emission probability) and prior (transition probability). Here we assumed we had an already tagged corpus from which we could estimate the required probabilities.

36.7 Estimation from Untagged Corpus: EM – Expectation-Maximization

When we use a plain-text corpus and a lexicon to obtain transition and emission probabilities by applying the EM algorithm. The steps are:

1. Start with some initial model
2. Compute the probability of (virtually) each state sequence given the current model
3. Use this probabilistic tagging to produce probabilistic counts for all parameters, and use these probabilistic counts to estimate a revised model, which increases the likelihood of the observed output W in each iteration
4. Repeat until convergence

***No** labeled training required. Initialize by lexicon constraints regarding possible POS for each word (cf. "noisy counting" for PP's). In other words we cannot calculate transition probabilities but do have values of emission probabilities. Therefore we start with some random values for the transition probabilities and continue until no improvement occurs.*

The EM algorithm is as follows:

- Choose initial model = <**a**,**b**,**g**(1)>

- Repeat until results don’t improve (much):
    - Compute p_k based on current model, using Forward & Backwards algorithms to compute A and B (Expectation for counts)
    - Compute new model <**a’**,**b’**,**g**’(1)> (Maximization of parameters)

Note: Output likelihood is guaranteed to increase in each iteration, but might converge to a *local* maximum

Summary

- Explained the concept of Model Learning in HMM
- Discussed Expectation Maximization method for HMM parameter estimation
- Explained parameter estimation using unsupervised techniques