Off-line handwritten word recognition using hidden Markov models

Davide Modolo and **Thijs Kooi**
Davide.Modolo@gmail.com
Thijs.Kooi@student.uva.nl

***Abstract*—**

I. Introduction

II. Pre-processing

III. Feature extraction

In order to exploit the sequential nature of the data, we need a model that does not treat the data as i.i.d. This is provided by the *hidden Markov model*, a model widely used in the areas of speech recognition and language modelling.

IV. Hidden Markov models

A Hidden Markov model is described by:

- A set of N states $S = (s_1, s_2, \ldots, s_N)$, where the state of the system at time t is denoted q_t
- A set of priors $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$, providing the probability $P(q_1 = s_i)$.
- A transition function $\mathbf{A}$, where $a_{ij} = P(q_j = s_j | q_i)$.
- An observation function $\mathbf{B}$, mapping each observation at every state to a probability $b_i(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = s_i, \lambda)$, where λ denotes the model parameters.

The model is trained to estimate the posterior probability $P(\mathbf{O}|\lambda)$ of an observation sequence $\mathbf{O}$, with D-dimensional observation vectors $\mathbf{o}_t = (o_1, o_2, \ldots, o_D)$. We assume time to be discrete and the transition and observation probabilities to be static, i.e., they do not change over time. The hyperparameters we have to concern ourselves with are the network topology, i.e., the number of states and the transition probability and choise of the observation or emmision probability function. For speech and handwritten a left-to-right model is generally applied, as show in figure 1, though variations exist.

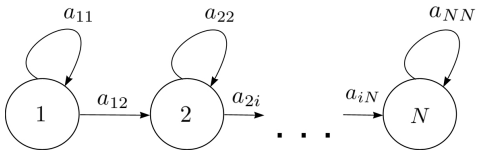


Fig. 1. Illustration of a simple left-to-right model

The three main problems we want to solve for this model architecture are:

1) The probability of an observation sequence, given the model, $P(\mathbf{O}|\lambda)$.
2) The most likely state sequence, underlying a given observation sequence and the model, $Q^* = \max P(Q|\mathbf{O}, \lambda)$.
3) The most likely parameters of the model $\lambda^* = \max P(X|\lambda)$, given a training set of M observation sequences $X = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_M)$.

Probability of a state sequence Q, under the model λ:

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{1}$$

Probability of an observation sequence O, given a state sequence Q and the model λ:

$$P(\mathbf{O}|Q, \lambda) = P(\mathbf{o}_1|q_1)P(\mathbf{o}_2|q_2)\ldots P(\mathbf{o}_T|q_T) =$$

$$b_1(o_1)b_2(o_2)\ldots b_T(o_T) = \prod_{t=1}^{T} b_t(o_t) \tag{2}$$

By the identity $P(A, B) = P(A|B)P(B)$, we have that:

$$P(\mathbf{O}, Q|\lambda) = P(O|Q, \lambda)P(Q|\lambda) =$$

$$[\prod_{t=1}^{T} b_t(\mathbf{o}_t)]\pi_{q_1} \prod_{t=1}^{T} a_{q_{t-1} q_t} =$$

$$\pi_{q_1} b_i(\mathbf{o}_1) \prod_{t=2}^{T} a_{q_{t-1} q_t} b_t(\mathbf{o}_t)$$

With the probability $P(\mathbf{O}|\lambda)$ calculated by marginalising over all possible state sequences Q:

$$P(\mathbf{O}|\lambda) = \sum_{Q} P(\mathbf{O}|Q, \lambda)P(Q|\lambda) =$$

$$\sum_{Q} \left[\pi_{q_1} b_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(\mathbf{o}_t)\right] \tag{3}$$

A naive way to estimate the posterior of an observation sequence, would be to simply try all possible state sequences and evaluate their probability. Unfortunately, the number of state sequences increases exponentially with the number of observations, rendering the method $\mathcal{O}(N^T)$ and thereby infeasable for all but the smallest models. Luckily, an alternative exists in the form of the *forward-backward* algorithm, a special instance of the *sum-product* algorithm.

A. Forward-backward algorithm

Forward probability, the probability of a partial observation sequence $o_1, o_2, \ldots, o_t$ ending in state i at time t.

$$\alpha_t(i) \equiv P(o_1, o_2, \ldots, o_t | q_t = s_i, \lambda) =$$

$$\Big[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij}\Big] b_j(o_t) \tag{4}$$

$$\beta_t(j) \equiv P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = s_j, \lambda) =$$

$$\sum_{i=1}^{N} a_{ij} b_i(o_{t+1}) \beta_i(o_{t+1}) \tag{5}$$

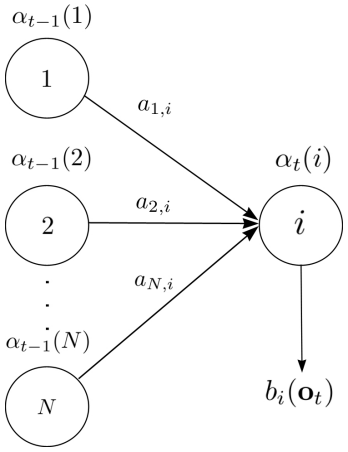


Fig. 2. Forward probability

The posterior probability of the observation sequence is subsequently calculated by summing over all possible end-states:

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{6}$$

This holds a complexity of only $\mathcal{O}(N^2 T)$, which is considerably less than the exponential time.

B. Emmision functions

$$b_j(\mathbf{o}_t) = \mathcal{N}(\mathbf{o}_t; \mu_j, \Sigma_j) \tag{7}$$

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{o}_t; \mu_{jk}, \Sigma_{jk}) \tag{8}$$

Where w_k denotes the prior probability of mixture component k, or weight of the mixture.

Training the model is done by *Baum-Welch reestimation*, a special instance of the EM-algorithm.

C. Baum-Welch reestimation

To train the model, we need to maximise the likelihood of the parameters $\mathcal{L}(\lambda|\mathbf{O})$, by taking partial derivatives of every variable and setting it to zero.

To estimate π_i, the probability that a sequence is starting in state i, we divide the number of sequences starting in the state, by the total number M of sequences used to train the model:

$$\hat{\pi_i} = \frac{\sum_{m=1}^{M} \delta(q_1^m = s_i)}{M} \tag{9}$$

Where $\delta(\cdot)$ denotes the Kronecker delta function. For the left-to-right model, the probability of entering state 1, is initialised to 1 and therefore π_i will remain fixed during the whole reestimation procedure. To make the rest of the update steps more comprehensible, a set of intermediate variables is defined. First, the probability of begin in a state i at timestep t, $P(q_t = s_i|\mathbf{O}, \lambda)$, which by Bayes' theorem can be written as:

$$(P(\mathbf{O}|\lambda))^{-1} P(\mathbf{O}|\lambda) P(q_t = s_i|\lambda) =$$

$$\gamma_t(i) \equiv (P(\mathbf{O}|\lambda))^{-1} \alpha_t(i) \beta_t(i) \tag{10}$$

Where normalisation constant $P(\mathbf{O}|\lambda) = \sum_{j=1}^{N} \alpha_t(i)\beta_t(i) = \sum_{j=1}^{N} \alpha_T(j)$.

$$P(q_t = s_i, q_{t+1} = s_j)|\mathbf{O}, \lambda) =$$

$$\xi_t(i,j) \equiv (P(\mathbf{O}|\lambda))^{-1} \alpha_t(i) a_{ij} b_j(\mathbf{o}_t) \beta_t(j)$$

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

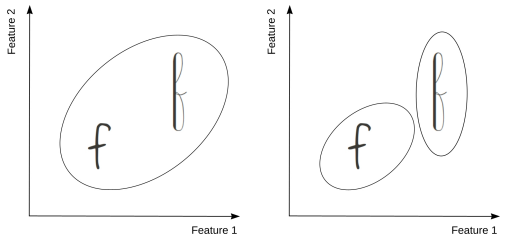
With these variables defined, we can assign the derivative of the likelihood functions with respect to the model parameters an intuitive meaning. With $\xi_t(i,j)$ the probability of being in state i and subsequently transferring to state j at time t and $\gamma_t(i)$, the total probability of being in state i at time t, the transition probability can be updated as:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{11}$$

A disadvantage about using a single Gaussian, is that for different writing styles of the same class, the Gaussian model will be trained to give the highest probability for a letter in between two instances, as illustrated by figure 3, where two different writing styles of an f are plotted in a two dimensional feature space. In the case of a GMM, and aditional latent variable, the weight of each mixture component, is introduced to the system.

$$P(q_t = s_i, C = k|\mathbf{O}, \lambda) =$$

$$P(q_t = s_i|\mathbf{O}, \lambda) \frac{P(\mathbf{o}_t, q_t = s_i, C = k|\lambda)}{P(\mathbf{o}_t, q_t = s_i|\lambda)} =$$



(a) Fit for single Gaussian (b) Fit for mixture of Gaussians

Fig. 3. Intuition behind the Gaussian mixture model

$$\gamma_t(i,k) \equiv \gamma_t(i)\frac{b_{i,k}(\mathbf{o}_t)}{b_i(\mathbf{o}_t)} \tag{12}$$

$$\hat{w}_{i,k} = \frac{\sum_{t=1}^{T}\gamma_t(i,k)}{\sum_{t=1}^{T}\gamma_t(i)} \tag{13}$$

$$\hat{\mu}_{i,k} = \frac{\sum_{t=1}^{T}\gamma_t(i,k)\mathbf{o}_t}{\sum_{t=1}^{T}\gamma_t(i,k)} \tag{14}$$

$$\hat{\Sigma}_{i,k} = \frac{\sum_{t=1}^{T}\gamma_t(i,k)(\mathbf{o}_t-\mu_{i,k})(\mathbf{o}_t-\mu_{i,k})^t}{\sum_{t=1}^{T}\gamma_t(i,k)} \tag{15}$$

When using a continuous observation distribution in the form of a single Gaussian or Gaussian mixture model, we have to decide upon the shape of the covariance matrix. This can either be full, diagonal or spherical/isotropic.

V. Experiments

VI. Discussion