

Machine Learning

Winter 2011/12

Roland Memisevic

Lecture 7, Dec. 5, 2011

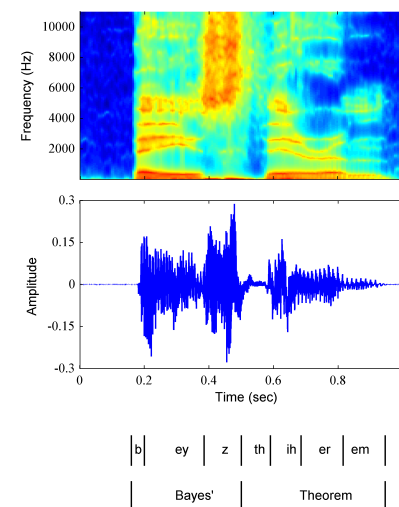
Modeling sequences

- ▶ So far we made the *iid* assumption in practically all models that we looked at to get a convenient learning rule.
- ▶ But in many real-world data-sets, observations are not iid.
- ▶ Dropping the assumption of independencies altogether would lead to intractable models, because the joint distribution would get too complex.
- ▶ A good compromise that fits many real-world data-sets is to assume a **sequence** structure for the observations: Examples include sound/speech signals, language, financial time-series, DNA strands, daily rainfall at some location, etc. etc. etc.

Outline

- ▶ Modeling sequence data
- ▶ Markov Models
- ▶ Hidden Markov Models

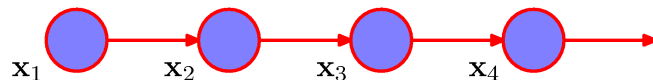
Ways to represent a speech signal as a sequence



What can we do with sequence models

- ▶ Models of sequences can be useful in various applications.
 1. An obvious application is prediction: Given a sequence up to time n , predict the value at time $n + 1$.
 2. A probabilistic model of sequences can be used also as a plug-in distribution in a generative classifier. This may work better than a naive Bayes model.
 3. And a sequence model allows us to pre-process a sequence in order to bring it into a better, more useful representation for further processing.

Markov Models



- ▶ We can represent this dependency structure graphically.
- ▶ In this graph, every arrow represents a *conditional distribution* of a node, given its predecessor.
- ▶ This hints at the fact that a Markov Model is an instance of the class of models known as *graphical models*, which we shall return to in another lecture.
- ▶ The exact form that the conditional distributions take on depends on the type of data we are dealing with:

Markov Models

- ▶ A simple probabilistic model of a sequence is the **Markov Model**

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^N p(\mathbf{x}_n | \mathbf{x}_{n-1})$$

- ▶ The Markov Model assumes that every observation depends only on its predecessor.
- ▶ Since the observations form a chain, a common alternative term is “Markov Chain”.
- ▶ “Markov Model” is sometimes used in a more general sense than this. We will use the term to denote only chains.

Parameterizing Markov Models

- ▶ One way to define a *continuous* Markov Model is to use conditional Gaussians:

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}(\mathbf{x}_{n-1}), \boldsymbol{\Sigma}(\mathbf{x}_{n-1}))$$

- ▶ To define a *discrete* Markov Model, we can use conditional discrete distributions

$$p(\mathbf{x}_n = j | \mathbf{x}_{n-1} = i)$$

- ▶ When we use the same conditional distribution $p(\mathbf{x}_n | \mathbf{x}_{n-1})$ for all n , then the model is called *homogeneous*. This is the most common way to define a Markov Model.

Discrete conditional distributions and CPTs

- ▶ We can think of a conditional discrete distribution as a matrix, or *table*, with entries:

$$A_{ij} = p(\mathbf{x}_n = j | \mathbf{x}_{n-1} = i)$$

- ▶ It is so common to represent conditional discrete distributions as tables, that these are often referred to as **CPT** for “conditional probability table”.
- ▶ CPTs are used not only in Markov models but also in other discrete models containing conditional distributions as components.

Estimating CPTs

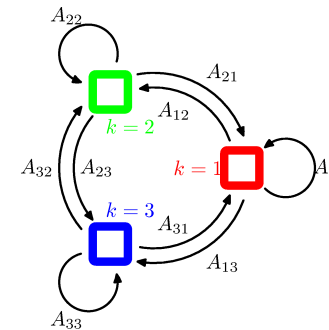
- ▶ Estimating the entries of a CPT is easy: It amounts to estimating multinomial parameters.
- ▶ If we assume \mathbf{x}_n to be in one-of- K encoding, we have

$$A_{ij} = \frac{\sum_n x_{nj} x_{n-1,i}}{\sum_n x_{ni}}$$

- ▶ The A_{ij} are also known as **transition probabilities**.
- ▶ One can easily verify that these are the *maximum likelihood estimates* for the Markov Model.
- ▶ We also need **start-probabilities** $\pi_i := p(\mathbf{x}_1 = i)$, which can be estimated as

$$\pi_i = \frac{\sum_n x_{ni}}{N}$$

Discrete Markov Models and CPTs



- ▶ An alternative way to visualize a homogeneous discrete Markov Model as a graph:
- ▶ A node represents a value that \mathbf{x}_n can take on, and the edges are labeled with the conditional probabilities.
- ▶ This shows that we can think of Markov Model also as a probabilistic *state machine*.
- ▶ The values \mathbf{x}_n can take on are therefore also called *states*.

Higher-order Markov Models

- ▶ Representing dependencies that range over no more than a single time-step can be a severe restriction in practice.
- ▶ One solution is to extend the dependencies to look back more than just one time-step, leading to *higher-order Markov Models*.
- ▶ Example: A second-order Markov model is defined by

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) p(\mathbf{x}_2 | \mathbf{x}_1) \prod_{n=3}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2})$$

- ▶ However, unfortunately the number of parameters grows exponentially with the order of the model!
- ▶ A much more common approach to modeling longer-range dependencies in sequences is to utilize *latent variables*...

Hidden Markov Models

Hidden Markov Model (HMM)

A Hidden Markov Model defines the data distribution

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{\mathbf{z}_1, \dots, \mathbf{z}_N} p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N)$$

with

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}) \prod_{m=1}^N p(\mathbf{x}_m | \mathbf{z}_m)$$

- ▶ $\mathbf{x}_1, \dots, \mathbf{x}_N$ are the observed data-points
- ▶ $\mathbf{z}_1, \dots, \mathbf{z}_N$ are latent variables

Hidden Markov Models

- ▶ We can think of a Hidden Markov Model as consisting of an unobserved, latent Markov Model

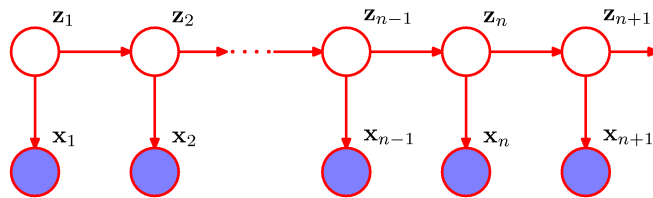
$$p(\mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1})$$

where each state emits an observation stochastically, according to

$$p(\mathbf{x}_n | \mathbf{z}_n)$$

- ▶ The observation probabilities are called “**emission probabilities**”.

Hidden Markov Models



- ▶ An HMM is like a Markov Chain that we observe indirectly through the emissions.
- ▶ We can also think of an HMM as mixture model whose states are coupled through time.

Parameterizing HMMs

- ▶ The latent distributions form a discrete Markov chain, so they are parameterized by the transition and start probabilities $A_{ij} = p(\mathbf{z}_n = j | \mathbf{z}_{n-1} = i)$ and $\pi_i = p(\mathbf{z}_1 = i)$
- ▶ The emission probabilities depend on the type of data we are dealing with. Common choices are discrete observations:

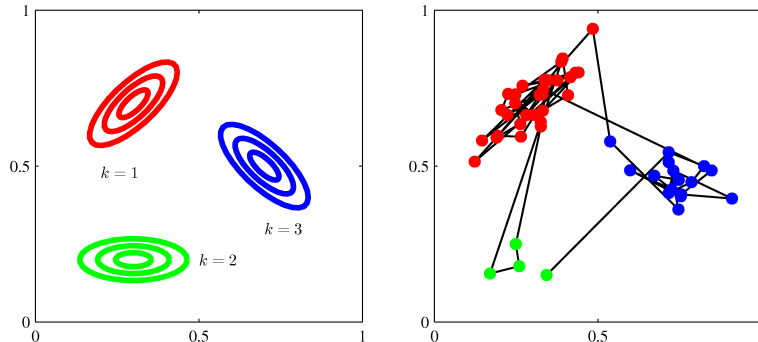
$$p(x_{nk} | \mathbf{z}_n = i) = \mu_{ik}$$

or Gaussian observations:

$$p(\mathbf{x}_n | \mathbf{z}_n = i) = \mathcal{N}(\mathbf{x}_n | \mu_i, \Sigma_i)$$

- ▶ It will be convenient to stack all the parameters for state i in some vector ϕ_i .

HMM as a Gaussian mixture model with dependent states



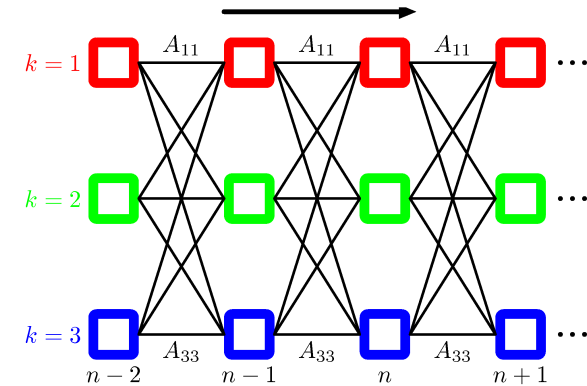
Independence properties

- ▶ By applying the rules of probability, one can derive various conditional independence properties for the HMM. A particularly useful one is the following, fairly intuitive property:

$$p(\mathbf{X}|\mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N|\mathbf{z}_n)$$

- ▶ This property is useful in deriving the EM algorithm for the HMM.
- ▶ There are many more (see Bishop, page 619, for a long list)

Hidden Markov Models



- ▶ We can represent the state transitions as a lattice.
- ▶ Each path through this lattice represents one possible sequence $\mathbf{z}_1, \dots, \mathbf{z}_N$ of latent states.
- ▶ There are exponentially many!

Evaluating the log-probability of data

- ▶ In the following, we stack data and hidden variables row-wise in matrices \mathbf{X} and \mathbf{Z} .
- ▶ We can write the probability that the model assigns to data \mathbf{X} as:

$$p(\mathbf{X}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}) \quad \left(= \sum_{\mathbf{z}_1, \dots, \mathbf{z}_N} p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) \right)$$

- ▶ In a model with K states and N observations, there will be K^N terms in this sum!
- ▶ Before we figure out a way how to compute this sum, we notice that this is not the only seemingly intractable sum that one needs to compute somehow:

Training using EM

- ▶ Since the HMM is a latent variable model, we will estimate parameters using EM.
- ▶ Recall that running EM amounts to iterating
 1. (*Inference*) Compute $p(\mathbf{Z}|\mathbf{X})$
 2. (*Parameter updates*) Optimize the expected complete log-likelihood

$$\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}) \log p(\mathbf{X}, \mathbf{Z})$$

- ▶ Again, we see that we need to sum over the K^N hidden states.

Simplifying summations

- ▶ We can not only compute sums over the joint $p(\mathbf{X}, \mathbf{Z})$ efficiently, but also also over other, related distributions.
- ▶ In particular, marginalizing $p(\mathbf{Z}|\mathbf{X})$ will allow us to compute marginals $p(\mathbf{z}_n|\mathbf{X})$ of the posterior over states, which will be useful for EM training.
- ▶ Instead of “pushing in” sums from the left and starting with \mathbf{z}_1 , one may “push in” sums from the right and start with \mathbf{z}_N .
- ▶ These two ways to define the sum over the joint distribution form the basis for the so-called α and β recursions, that we shall define later.

Simplifying summations

- ▶ The definition of the joint

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n|\mathbf{z}_{n-1}) \prod_{m=1}^N p(\mathbf{x}_m|\mathbf{z}_m)$$

together with the distributive law allows us to re-write the sums over \mathbf{Z} so they are computed piece-by-piece:

$$\begin{aligned} & \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}) \\ &= \sum_{\mathbf{z}_1} \dots \sum_{\mathbf{z}_N} p(\mathbf{X}, \mathbf{Z}) \\ &= \sum_{\mathbf{z}_1} \dots \sum_{\mathbf{z}_N} p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n|\mathbf{z}_{n-1}) \prod_{m=1}^N p(\mathbf{x}_m|\mathbf{z}_m) \\ &= \sum_{\mathbf{z}_1} p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) \sum_{\mathbf{z}_2} p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{x}_2|\mathbf{z}_2) \sum_{\mathbf{z}_3} p(\mathbf{z}_3|\mathbf{z}_2) \dots \end{aligned}$$

Rewriting the expected complete log-likelihood

- ▶ The marginals of the posterior are useful, because we can write the complete log-likelihood in terms of these:

$$\begin{aligned} & \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}) \log p(\mathbf{X}, \mathbf{Z}) \\ &= \sum_{k=1}^K \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \log A_{jk} \\ & \quad + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \log p(\mathbf{x}_n|\phi_k) \end{aligned}$$

where

$$\gamma(\mathbf{z}_n) := p(\mathbf{z}_n|\mathbf{X}), \quad \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) := p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X})$$

- ▶ Note that also write $\gamma(z_{nk})$ and $\xi(z_{n-1,j}, z_{nk})$ with some abuse of notation, assuming one-hot encodings for \mathbf{z} .

Transition probability updates

- ▶ By setting the derivatives of the expected complete log-likelihood to zero (and using Lagrange multipliers where required), one arrives at the following update equations for the transition parameters:

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})}$$
$$A_{jk} = \frac{\sum_{n=2}^N \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^K \sum_{n=2}^N \xi(z_{n-1,j}, z_{nl})}$$

- ▶ In the same way, we get the emission probability updates:

EM training summary

- ▶ To summarize: In order to deploy the EM algorithm, we need to compute the posteriors over hidden variables given data.
- ▶ For the HMM, we need these posteriors only in the form of the *marginals* $\gamma(\mathbf{z}_n)$ over single time-steps, and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ over adjacent time-steps.
- ▶ We now turn to an efficient way to compute these marginals:

Emission probability updates

- ▶ Gaussian HMM:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^N \gamma(z_{nk})}$$
$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{\sum_{n=1}^N \gamma(z_{nk})}$$

- ▶ Discrete HMM:

$$\mu_{ik} = \frac{\sum_{n=1}^N \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^N \gamma(z_{nk})}$$

The forward-backward algorithm

- ▶ We have

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}) = \frac{p(\mathbf{X} | \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})}$$

- ▶ Because of the conditional independence property (slide 19), we can write

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n) \beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

with

$$\alpha(\mathbf{z}_n) := p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n)$$

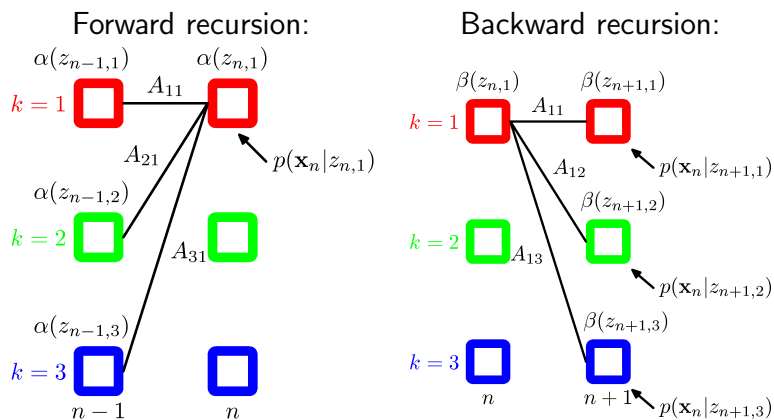
and

$$\beta(\mathbf{z}_n) := p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)$$

The forward-backward algorithm

- ▶ $\alpha(\mathbf{z}_n)$ represents the probability of seeing the sequence up to time-step n and being in state \mathbf{z}_n at time-step n .
- ▶ $\beta(\mathbf{z}_n)$ represents the probability of seeing the sequence from time-step $n + 1$ on, given that we were in state \mathbf{z}_n at time-step n .
- ▶ We can compute all α and β (and thereby γ and ξ) recursively. This is known as **forward-backward algorithm**.
- ▶ It is also an instance of a more general procedure known as *dynamic programming*.

Illustration of the forward and backward recursions



Recursions for α and β

The α recursions

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = \prod_{k=1}^K (\pi_k p(\mathbf{x}_1|\phi_k))^{z_{1k}}$$

The β recursions

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

$$\beta(\mathbf{z}_N) = 1$$

Computing ξ and $p(\mathbf{X})$

- ▶ We can use α and β also to compute ξ as follows:

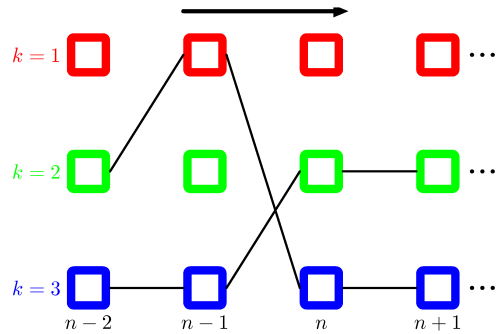
$$\begin{aligned} & \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) \\ &= p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}) \\ &= \frac{p(\mathbf{X} | \mathbf{z}_{n-1}, \mathbf{z}_n) p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}) \beta(\mathbf{z}_n)}{p(\mathbf{X})} \end{aligned}$$

- ▶ And to evaluate the probability that the model assigns to data: Sum over the definition of $\gamma(\mathbf{z}_n)$ to get the expression

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n) \beta(\mathbf{z}_n) \quad \forall n$$

Viterbi decoding

- ▶ Computing $\gamma(\mathbf{z}_n)$ involves marginalizing (summing) over exponentially many joint instantiations of \mathbf{z}_i .
- ▶ By replacing the sum by a \max , one can similarly recursively compute the *most probable path* through the hidden states, given \mathbf{X} .
- ▶ This is known as the **Viterbi algorithm**.



Kalman filter

- ▶ Hidden Markov Models are discrete hidden variable models, where the hidden variables are coupled through time.
- ▶ Not surprisingly, there are also continuous time-dependent hidden variable models (in analogy to using Gaussian mixtures vs. PCA).
- ▶ With Gaussian distributions as the continuous distributions $p(\mathbf{z}_n|\mathbf{z}_{n-1})$, one obtains a model known as the *Kalman filter* or *Linear Dynamical System* (LDS).