GMM (continued), Hidden Markov Models

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, Andrew Moore and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

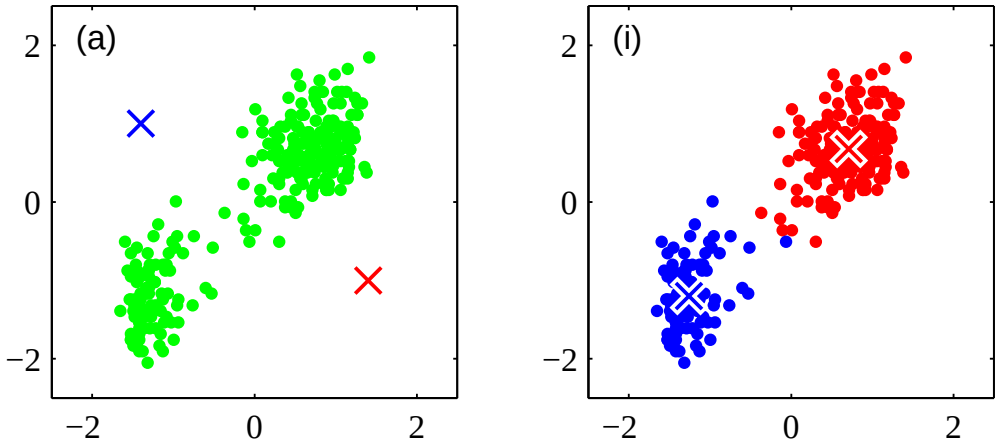
Outline

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: centroids of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

Toy Example Cluster data into two clusters.



Applications

- Identify communities within social networks
- Find topics in news stories
- Group similiar sequences into gene families

K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$,

Distortion measure (clustering objective function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Algorithm

Minimize distortion measure alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Assume the current value of $\{r_{nk}\}$ fixed, minimize J over $\{\boldsymbol{\mu}_k\}$, which leads to the following rule to update the centroids of the clusters

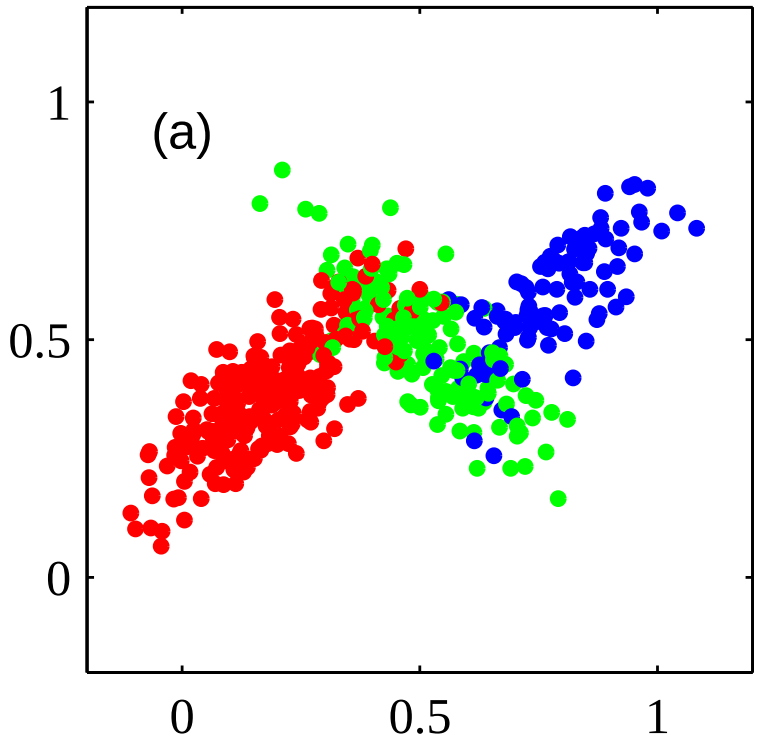
$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- **Step 3** Determine whether to stop or return to Step 1

Remarks

- Centroid $\boldsymbol{\mu}_k$ is the mean of data points assigned to the cluster k, hence 'K-means' (you considered an alternative in problem set 4)
- The procedure reduces J in both Step 1 and Step 2 and thus makes improvements on each iteration
- No guarantee we find the global solution; quality of local optimum depends on initial values at Step 0

Gaussian mixture models: intuition



- Probabalistic interpretation of K-means
- We can model *each* region with a distinct distribution, e.g., Gaussian mixture models (GMMs)
- Can be viewed as generative model

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – priors on each component that satisfy:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

- Given *unlabeled* data, $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, we must learn:
  - *parameters* of mixture components
  - *mixture weights*

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

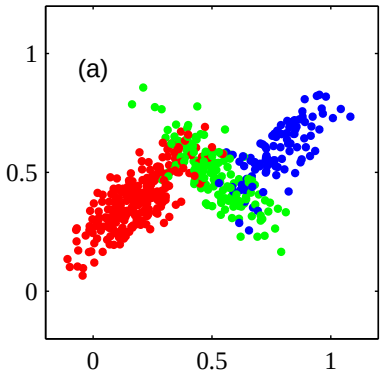
$$p(\boldsymbol{x}|z = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

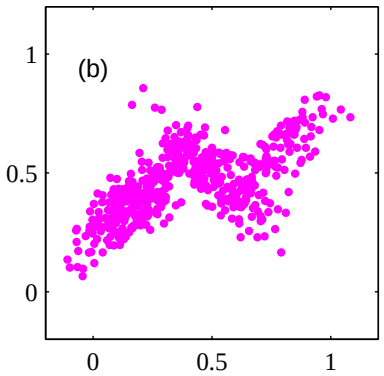
Then, the marginal distribution of $\boldsymbol{x}$ is

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Namely, the Gaussian mixture model

GMMs: example





The conditional distribution between $\boldsymbol{x}$ and z (representing color or cluster membership) are

$$p(\boldsymbol{x}|z = red) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

The marginal distribution is thus

$$p(\boldsymbol{x}) = p(red)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(blue)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p(green)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Given parameters $\boldsymbol{\theta}$, how would we choose a cluster assignment for $\boldsymbol{x}$?

Parameter estimation for GMMs: complete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$$

Complete Data: We (unrealistically) assume z is observed for every $\boldsymbol{x}$,

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$$

MLE: Maximize the complete likelihood

$$\boldsymbol{\theta} = \arg\max \log P(\mathcal{D}') = \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Parameter estimation for GMMs: complete data

Group likelihood by values of z_n

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_{k=1}^{K} \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

Introduce dummy variables

$\gamma_{nk} \in \{0, 1\}$ indicate whether $z_n = k$:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log p(z = k)p(\boldsymbol{x}_n|z = k)$$

In the complete setting the γ_{nk} just add to the notation, but later we will 'relax' these variables and allow them to take on fractional values

Parameter estimation for GMMs: complete data

We can simplify the complete likelihood as follows:

$$\begin{aligned}\sum_n \log p(\boldsymbol{x}_n, z_n) &= \sum_k \sum_n \gamma_{nk} \log p(z=k) p(\boldsymbol{x}_n | z=k) \\ &= \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \\ &= \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}\end{aligned}$$

ω_k appears only in left term, and the k-th component's parameters only appear inside braces of right term. We can easily compute MLE (exercise):

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Outline

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved random variables, and hence is incomplete

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved random variables, and hence is incomplete

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Goal Obtain the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}} = \arg\max \ell(\boldsymbol{\theta}) = \arg\max \log P(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved random variables, and hence is incomplete

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Goal Obtain the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}} = \arg\max \ell(\boldsymbol{\theta}) = \arg\max \log P(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

$$= \arg\max \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

The objective function $\ell(\boldsymbol{\theta})$ is called the *incomplete* log-likelihood.

Issue with Incomplete log-likelihood

No simple way to optimize the incomplete log-likelihood

Expectation-Maximization (EM) algorithm provides a strategy for iteratively optimizing this function

Two steps as they apply to GMM:

- E-step: 'guess' values of the z_n using existing values of $\boldsymbol{\theta}$
- M-step: solve for new values of $\boldsymbol{\theta}$ given imputed values for z_n

Issue with Incomplete log-likelihood

No simple way to optimize the incomplete log-likelihood

Expectation-Maximization (EM) algorithm provides a strategy for iteratively optimizing this function

Two steps as they apply to GMM:

- E-step: 'guess' values of the z_n using existing values of $\boldsymbol{\theta}$
- M-step: solve for new values of $\boldsymbol{\theta}$ given imputed values for z_n (maximize complete likelihood!)

E-step: Soft cluster assignments

We define γ_{nk} as $p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta})$

- This is the posterior distribution of z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$
- Recall that in complete data setting γ_{nk} was binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component, with $\boldsymbol{x}_n$ assigned to each component with some probability

E-step: Soft cluster assignments

We define γ_{nk} as $p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta})$

- This is the posterior distribution of z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$
- Recall that in complete data setting γ_{nk} was binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component, with $\boldsymbol{x}_n$ assigned to each component with some probability

Given $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, we can compute γ_{nk} using Bayes theorem:

$$\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$$

$$= $$

E-step: Soft cluster assignments

We define γ_{nk} as $p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta})$

- This is the posterior distribution of z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$
- Recall that in complete data setting γ_{nk} was binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component, with $\boldsymbol{x}_n$ assigned to each component with some probability

Given $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, we can compute γ_{nk} using Bayes theorem:

$$\begin{aligned}\gamma_{nk} &= p(z_n = k|\boldsymbol{x}_n) \\ &= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)}\end{aligned}$$

E-step: Soft cluster assignments

We define γ_{nk} as $p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta})$

- This is the posterior distribution of z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$
- Recall that in complete data setting γ_{nk} was binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component, with $\boldsymbol{x}_n$ assigned to each component with some probability

Given $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, we can compute γ_{nk} using Bayes theorem:

$$\begin{aligned}
\gamma_{nk} &= p(z_n = k|\boldsymbol{x}_n) \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^K p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}
\end{aligned}$$

M-step: Maximimize complete likelihood

Recall definition of complete likelihood from earlier:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Previously γ_{nk} was binary, but now we define $\gamma_{nk} = p(z_n = k | \boldsymbol{x}_n)$ (E-step)

M-step: Maximimize complete likelihood

Recall definition of complete likelihood from earlier:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Previously γ_{nk} was binary, but now we define $\gamma_{nk} = p(z_n = k | \boldsymbol{x}_n)$ (E-step)

We get the same simple expression for the MLE as before!

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

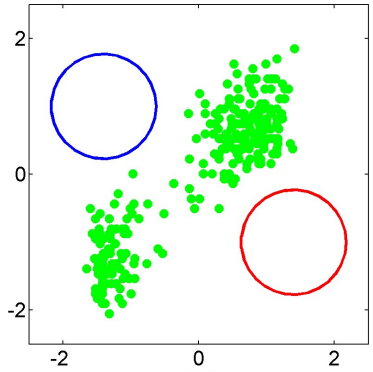
Intuition: Each point now contributes some fractional component to each of the parameters, with weights determined by γ_{nk}

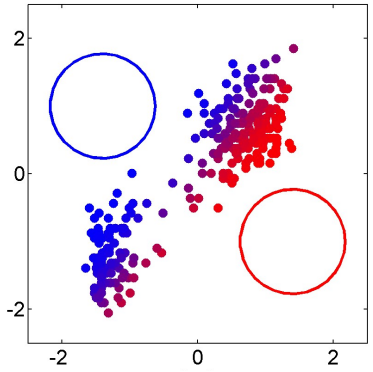
EM procedure for GMM

Alternate between estimating γ_{nk} and estimating $\boldsymbol{\theta}$

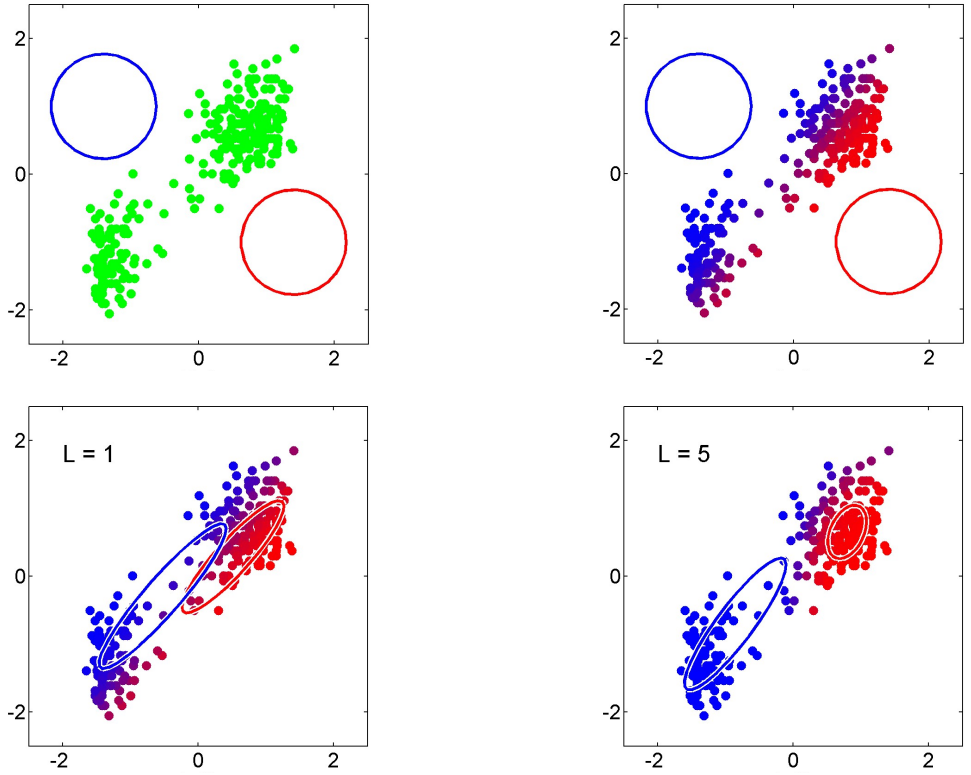
- Initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Repeat
  - E-Step: Compute γ_{nk} using the current $\boldsymbol{\theta}$
  - M-Step: Update $\boldsymbol{\theta}$ using the γ_{nk} we just computed
- Until Convergence

Example of GMM

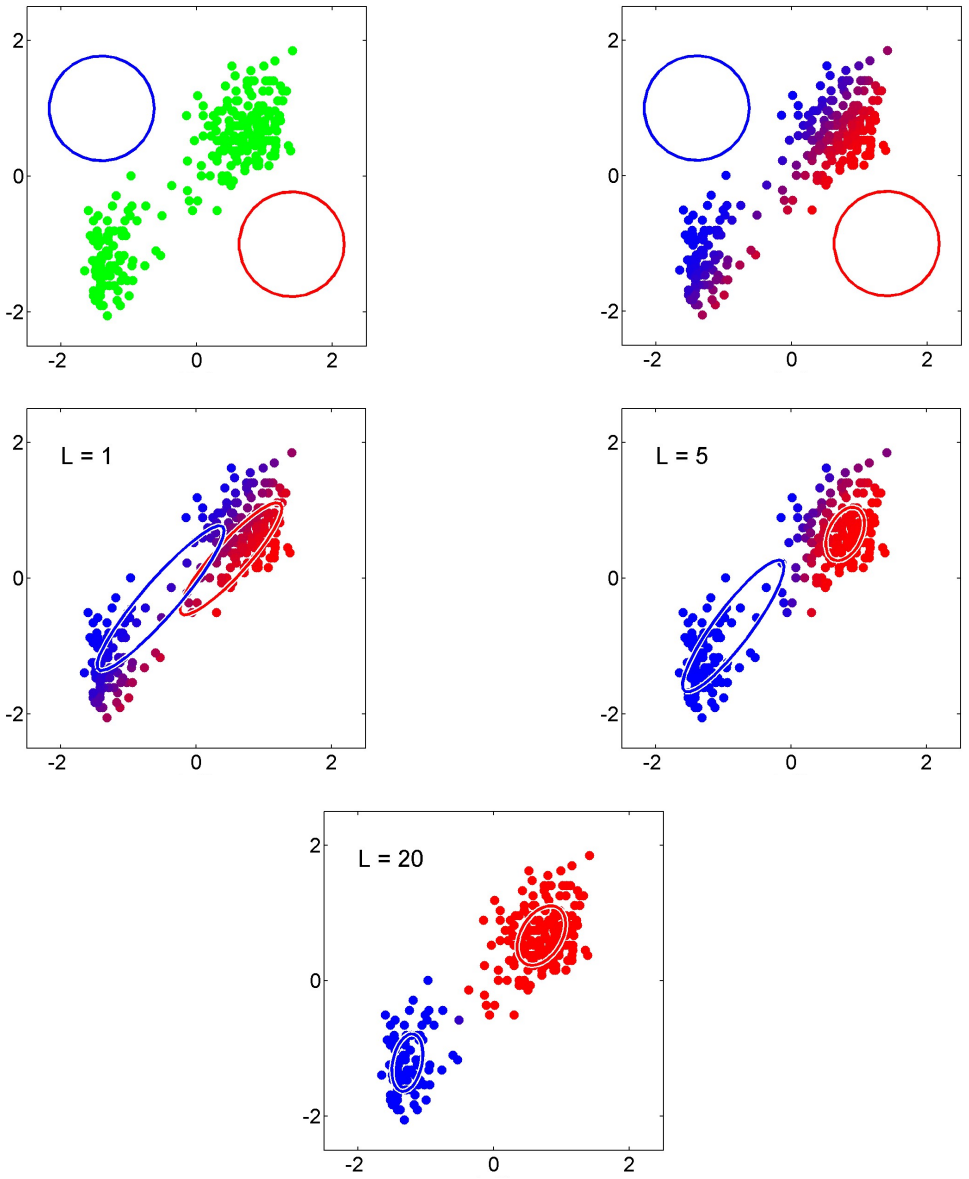




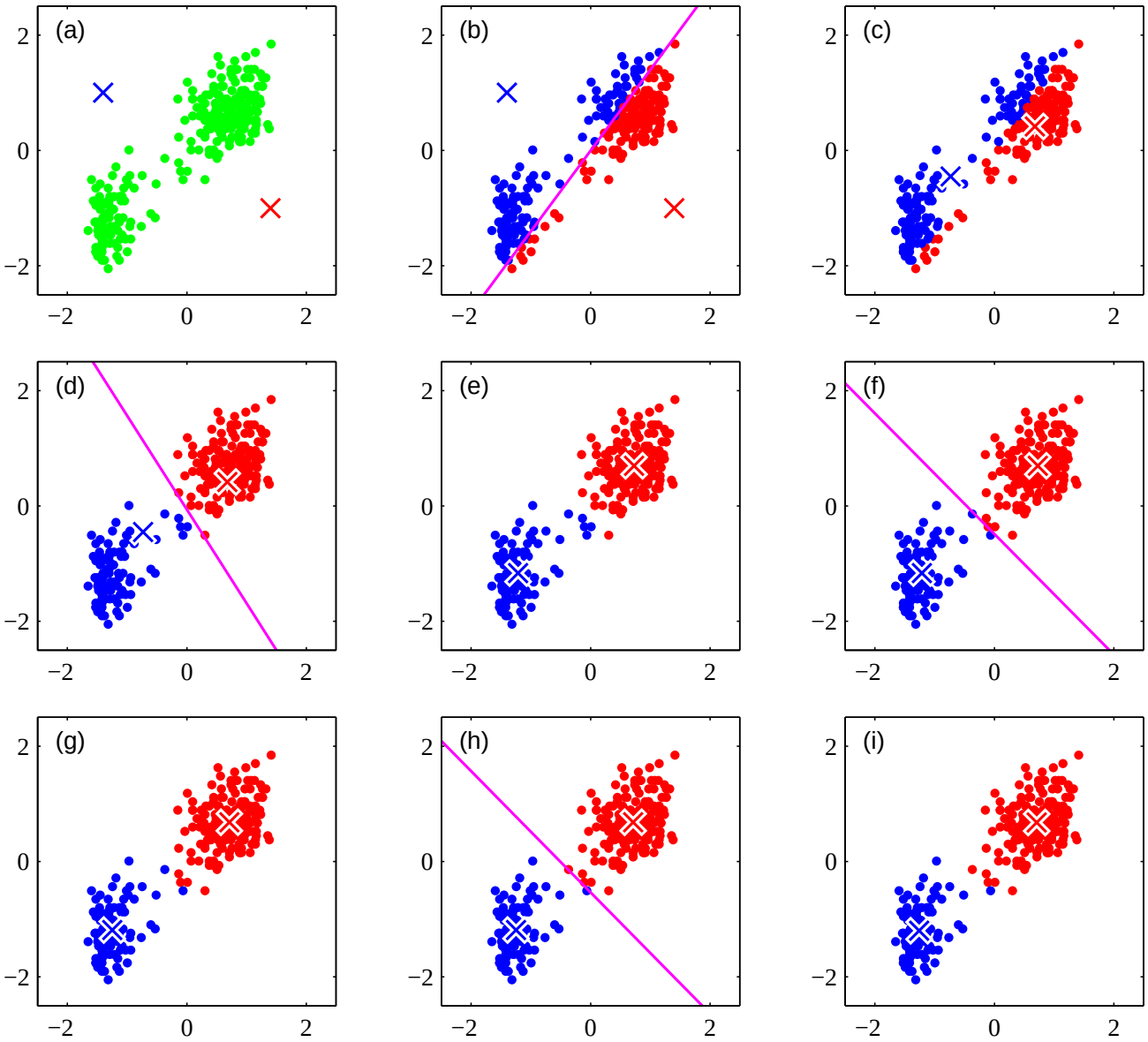
Example of GMM



Example of GMM



Compare to K-means example



EM procedure for GMM

Questions to be answered next

- How does GMM relate to K-means?
- Is this procedure reasonable, i.e., are we optimizing a sensible criterion?
- Will this procedure converge?

GMMs and K-means

GMMs provide probabilistic interpretation for K-means

GMMs and K-means

GMMs provide probabilistic interpretation for K-means

GMMs reduce to K-means under the following assumptions (in which case EM for GMM parameter estimation simplifies to K-means):

- Assume all mixture weights ω_k are equal
- Assume all Gaussians have $\sigma^2 \boldsymbol{I}$ covariance matrices
- Further assume $\sigma \to 0$, so we only need to estimate $\boldsymbol{\mu}_k$, i.e., means
- GMMs are more general model.

K-means is often called "hard" GMM or GMMs is called "soft" K-means

The posterior γ_{nk} provides a probabilistic assignment for $\boldsymbol{x}_n$ to cluster k

EM algorithm

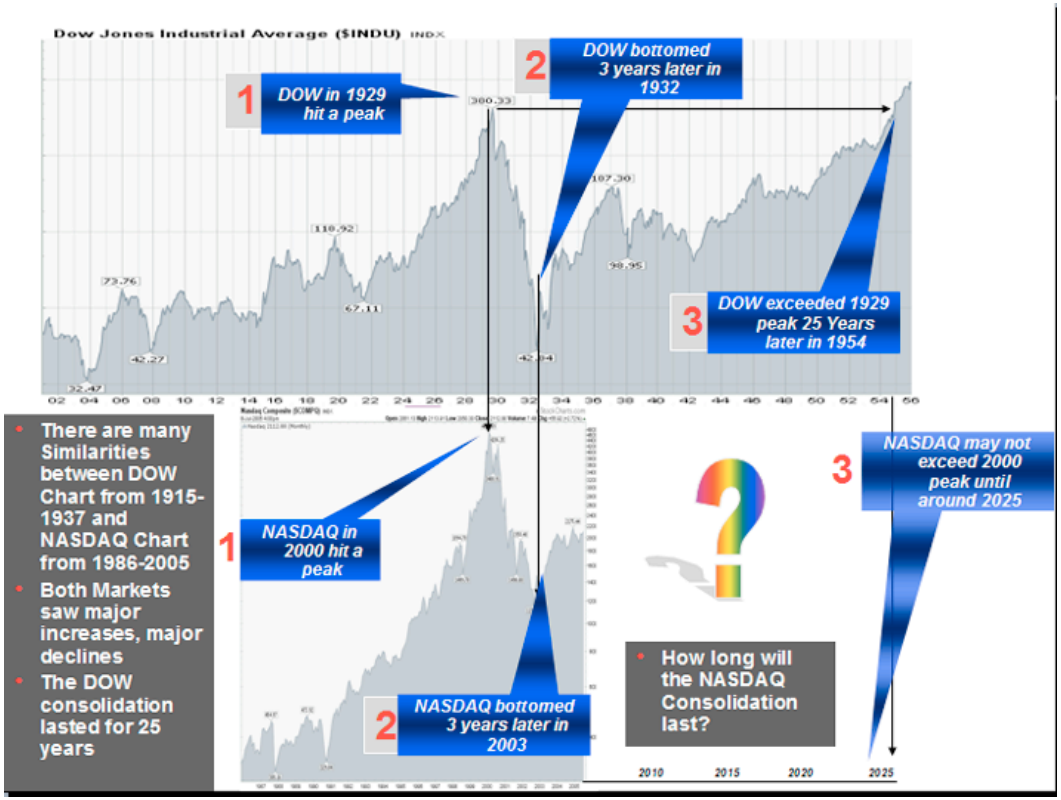
- The estimates of the parameter θ in each iteration increase the likelihood.
- EM algorithm converges but only to a local optimum.

Outline

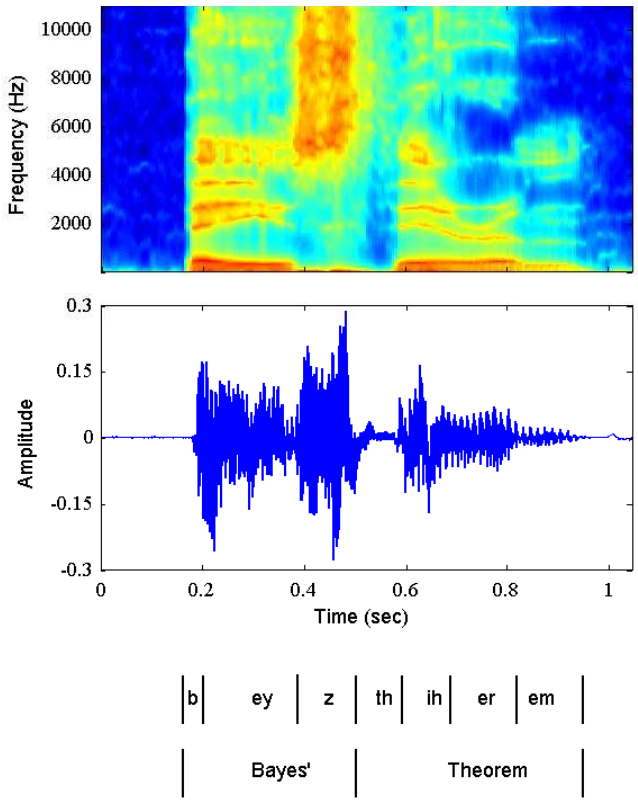
Assumptions in our models

- Linear
  - Can be removed using non-linear basis functions and kernels.
- Independent distribution of training data
  - Can we remove this assumption ?

Financial forecasting

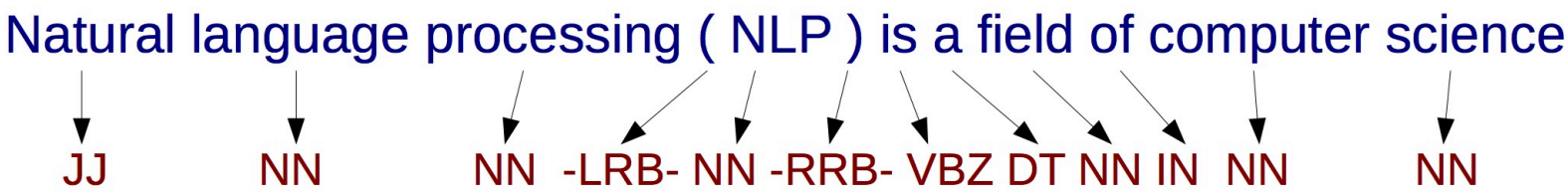


Speech recognition



Natural Language Processing

Part of Speech Tagging



How can we predict the part of speech?

Sequential process

- A system that can occupy one of K states or categories.
  - States: in speech recognition, one of 26 characters
  - States: in POS tagging, one of the parts of speech (noun, verb, etc.)
- X_t is the state of the sytem at time t.
- X_t is a random variable.
- Takes values in $\{1, \ldots, K\}$.
- We will often refer to t as time but t can be any index (*e.g.*, word in a sentence, position in a sequence).

Sequential process

- If we observe the state of the system from time 1 to t, what is its state at time $t+1$?
  Given by the conditional distribution:

$$P(x_{t+1}|x_1, \ldots, x_t)$$

- What is the probability of observing that the system is in states $(x_1, \ldots, x_t)$ at times 1 to t

$$P(x_1, \ldots, x_t) = P(x_1)P(x_2|x_1) \ldots P(x_t|x_1, \ldots, x_{t-1})$$

If I tell you each of the conditional probabilities, then we can compute the joint probability of any sequence.

Markov Process

Also known as Markov Chain or Markov model

- For a Markov process, the next state depends on the current state :

$$P(x_{t+1}|x_1, \ldots, x_t) = P(x_{t+1}|x_t)$$

Conditioned on the present, the future is independent of the past

X_{t+1} is independent of $X_i, i < t$ given X_t

- This property implies that

$$P(x_1, \ldots, x_t) = P(x_1)P(x_2|x_1) \ldots P(x_t|x_{t-1})$$

Specifying a Markov chain

Initial probability

$$\pi_i = P(X_1 = i)$$

Transition probability

$$q_{ij} = P(X_{t+1} = i | X_t = j)$$

Specifying a Markov chain

Assume we have three states.

Initial probability vector

$$\boldsymbol{\pi} = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix}$$

Constraints: $\pi_i \geq 0, \sum_i \pi_i = 1$

Transition probability matrix

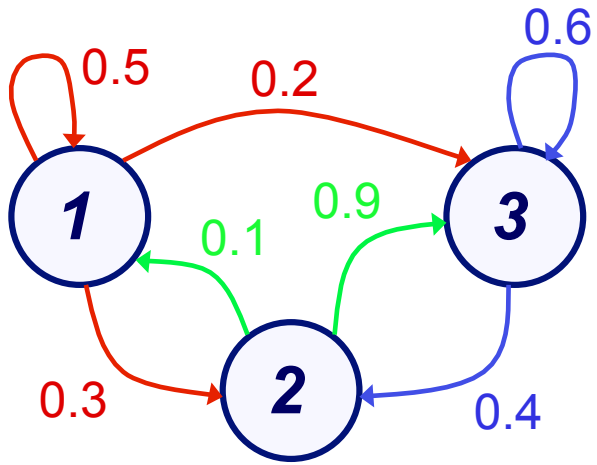
$$\boldsymbol{Q} = \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix}$$

Constraints: $q_{ij} \geq 0, \sum_i q_{ij} = 1$

State transition diagram

States $= \{1, 2, 3\}$

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$



Computing on Markov chains

Compute $P(x_1, x_2, \ldots, x_T)$ for any sequence of states (path) $(x_1, \ldots, x_T)$

Assume uniform probability of starting in each of the states

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

What is the probability of observing a sequence of states $(X_1, X_2, X_3) = (1, 1, 3)$?

Computing on Markov chains

Compute $P(x_1, x_2, \ldots, x_T)$ for any sequence of states (path) $(x_1, \ldots, x_T)$

Assume uniform probability of starting in each of the states

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

What is the probability of observing a sequence of states $(X_1, X_2, X_3) = (1, 1, 3)$?
More generally

$$P(x_1, \ldots, x_T) = \pi_{x_1} q_{x_2, x_1} \cdots q_{x_T, x_{T-1}}$$

Computing on Markov chains

Compute $P(x_T = i)$

Assume uniform probability of starting in each of the states

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

What is the probability of observing a state $X_3 = 3$?

Computing on Markov chains

Compute $P(x_T = i)$

Assume uniform probability of starting in each of the states

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

What is the probability of observing a state $X_3 = 3$?

More generally, use the previous computation to sum over all paths of length t that end in the state i

$$P(X_t = i) = \sum_{x_1, x_2, \ldots, x_{t-1}} P(x_1, x_2, \ldots, x_{t-1}, x_t = i)$$

Computational cost?

O(K^{T-1}): Exponential in T!

Computing on Markov chains

Compute $P(x_T = i)$

Define $p_t(i) = P(X_t = i)$.

$$p_1(i) = \pi_i$$

Assume we already know $p_{t-1}(i)$ for all i.
Use inductive definition to compute $p_t(i)$:

$$\begin{aligned} p_t(i) &= P(X_t = i) \\ &= \sum_j P(X_{t-1} = j, X_t = i) \\ &= \sum_j P(X_{t-1} = j) P(X_t = i | X_{t-1} = j) \\ &= \sum_j p_{t-1}(j) q_{ij} \end{aligned}$$

Computing on Markov chains

Compute $P(x_T = i)$

Define $p_t(i) = P(X_t = i)$.

$$p_1(i) = \pi_i$$

$$p_t(i) = \sum_j p_{t-1}(j) q_{ij}$$

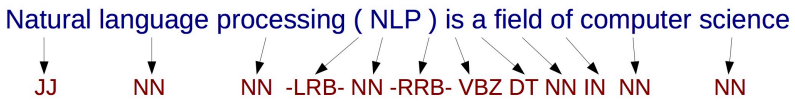
Computational cost?

- Computing each $p_t(i)$ for a given t, i takes $O(K)$.
- Computing each $p_t(i)$ for a given t and all i takes $O(K^2)$.
- Computing each $p_t(i)$ for all $t \in \{1, \ldots, T\}$ and all i takes $O(K^2(T-1))$.

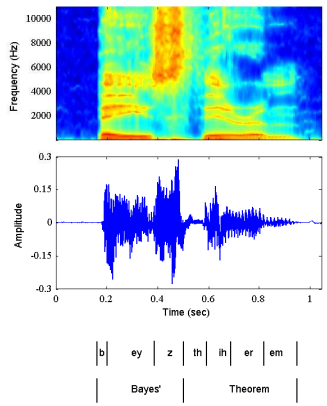
Example of Dyanmic Programming

Hidden Markov models

In many applications, we observe only a noisy or indirect measurement of the state X_t



In POS tagging, the state is the part-of-speech which we do not see. Instead, we observe words.



In speech recognition, the state is the word spoken but we only get to see the waveform.

Hidden Markov models

- Previously, in Markov chains, we directly observed the state X_t.
- Now, we observe a new random variable Y_t for each time t that is affected by the state X_t at that time t.
- Can think of Y_t as a noisy version of the true state.

Having observed $(Y_1, \ldots, Y_T)$, we want to ask questions about $X_1, \ldots, X_T$

Hidden Markov models

- We now define the set of observed states (also called emission symbols) $B = \{1, \ldots, L\}$: the set of values that Y_t can take.
- Since X_t is not observed, X_t are called hidden states.

To link up the hidden and the observed states, we have emission probabilities

$$P(Y_t = b | X_t = k) = e_k(b)$$

Constraints: $\sum_b e_k(b) = 1$.

Hidden Markov model

Hidden states: $\{1, \ldots, K\}$
Observed states: $\{1, \ldots, L\}$
Initial probability

$$\pi_i = P(X_1 = i)$$

Transition probability

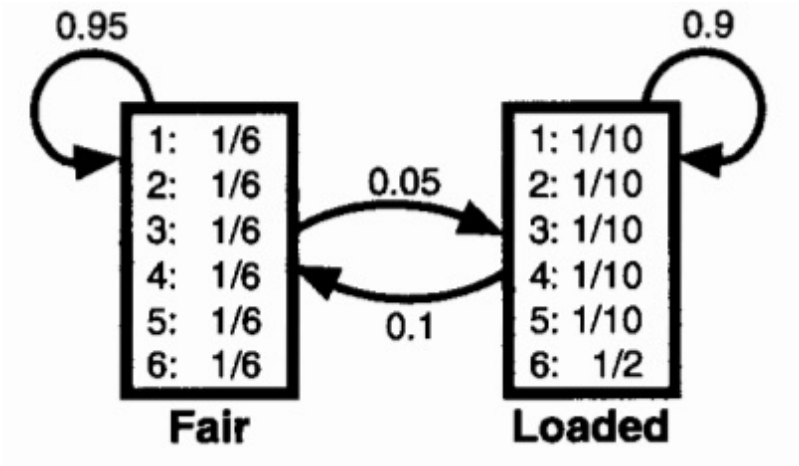
$$q_{ij} = P(X_{t+1} = i | X_t = j)$$

Emission probability

$$e_i(b) = P(Y_t = b | X_t = i)$$

HMM: Example

The occasionally dishonest casino



1

- Hidden State: Is the casino using the fair or unfair die?
- Observed State: Roll of die ($\{1, \ldots, 6\}$)

Durbin et al. Biological Sequence Analysis

HMM: Example

HMMs can be used to generate or emit observations

1. Pick state x_1 according to the distribution π.
2. Set $t \leftarrow 1$.
3. Pick (or emit) an observation y_t according to e_{x_t}.
4. Choose the state x_{t+1} according to the distribution $q_{.,x_t}$.
5. $t \leftarrow t + 1$
6. Go to step 3.

Querying an HMM

We observe $(Y_1, \ldots, Y_5) = (2, 6, 6, 1, 3)$. Can we infer for which of the throws the casino used the unfair die ?

Querying an HMM

POS tagging Given the sequence of words, what is the part-of-speech tag for each word?

Speech recognition Given the audio waveform, what is the sequence of words?

HMM

The joint probability of the sequence of hidden states and the observed states

$$
\begin{aligned}
P(\boldsymbol{y}, \boldsymbol{x}) &= P(y_{1:T}, x_{1:T}) \\
&= P(y_{1:T}|x_{1:T})P(x_{1:T}) \\
&= \prod_{t=1}^{T} P(y_t|x_t) \quad \underbrace{P(x_{1:T})}_{\text{Markov chain}} \\
&= \prod_{t=1}^{T} e_{x_t}(y_t)\pi_{x_1} \prod_{t=1}^{T-1} q_{x_{t+1}x_t}
\end{aligned}
$$

Easy to compute but not very useful because we don't know the hidden states.

HMM: The most probable path problem

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the most probable sequence of hidden states $(x_1, \ldots, x_T)$?

$$\arg\max_{x_{1:T}} P(y_{1:T}, x_{1:T})$$

- Often called the decoding problem.
- One way to solve this problem is to search over all possible values of $(x_{1:T})$.

HMM: The most probable path problem

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the most probable sequence of hidden states $(x_1, \ldots, x_T)$?

$$\arg\max_{x_{1:T}} P(y_{1:T}, x_{1:T})$$

- Often called the decoding problem.
- One way to solve this problem is to search over all possible values of $(x_{1:T})$.
- There are exponentially many of them (O(K^T))

HMM: The most probable path problem

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the most probable sequence of hidden states $(x_1, \ldots, x_T)$?

$$\arg\max_{x_{1:T}} P(y_{1:T}, x_{1:T})$$

- Often called the decoding problem.
- One way to solve this problem is to search over all possible values of $(x_{1:T})$.
- There are exponentially many of them (O(K^T))
- Fortunately, it turns out there is an efficient dynamic programming algorithm to solve this problem.

The Viterbi algorithm

A recursive algorithm

- Suppose we have computed the the probability $v_t(k)$ for all values of $t \in \{1, \ldots, T\}$ and $k \in \{1, \ldots, K\}$:
  The probability of the most probable path (MPP) for observations $(y_1, \ldots, y_t)$ (so that path has length t) that ends up in state k.

$$v_t(k) = \max_{x_{1:t-1}} P(y_{1:t}, x_{1:t-1}, x_t = k)$$

  Why is this a useful quantity?

- If we look at $v_T(k)$, it tells us the probability of the MPP of length T that ends in state k.
- The answer to the MPP problem then is $\max_k v_T(k)$!

The Viterbi algorithm

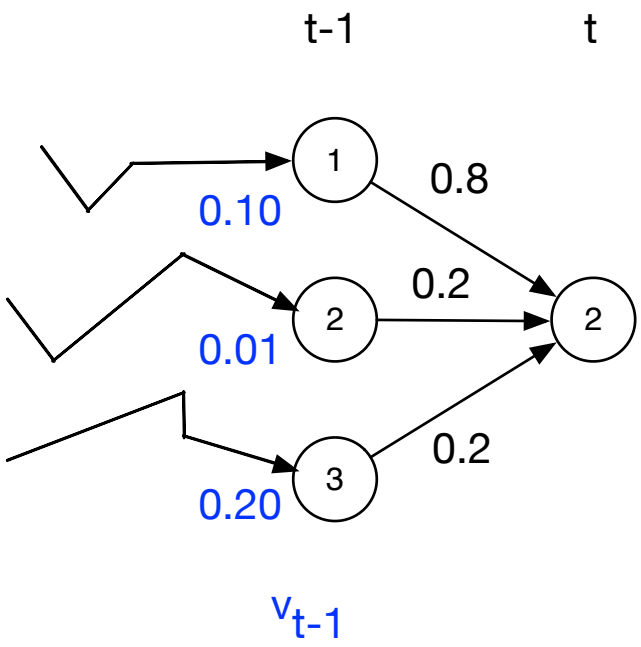
Can we compute $v_t(k)$ efficiently?

Assume we have computed $v_{t-1}(l)$.

We have computed the probability of the MPP of length $t-1$ that ends in state l for all values of l

How do we use this to compute the probability of MPP of length t that ends in state k?

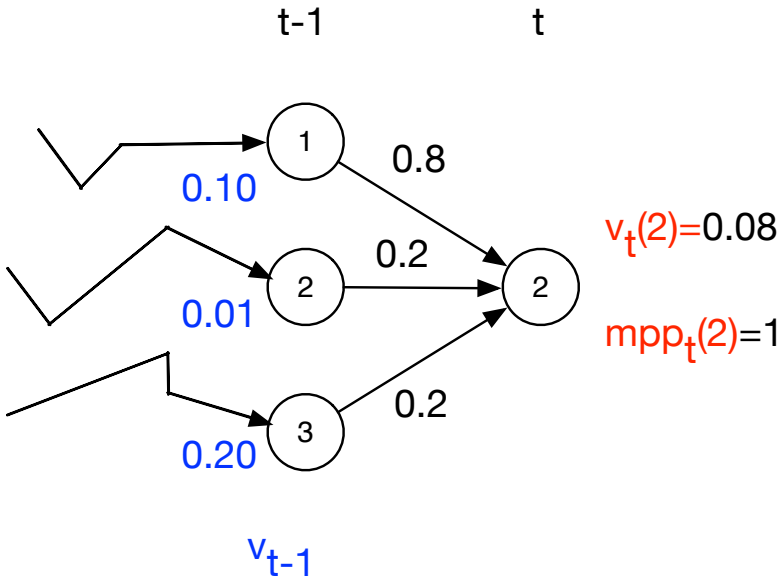
Viterbi example



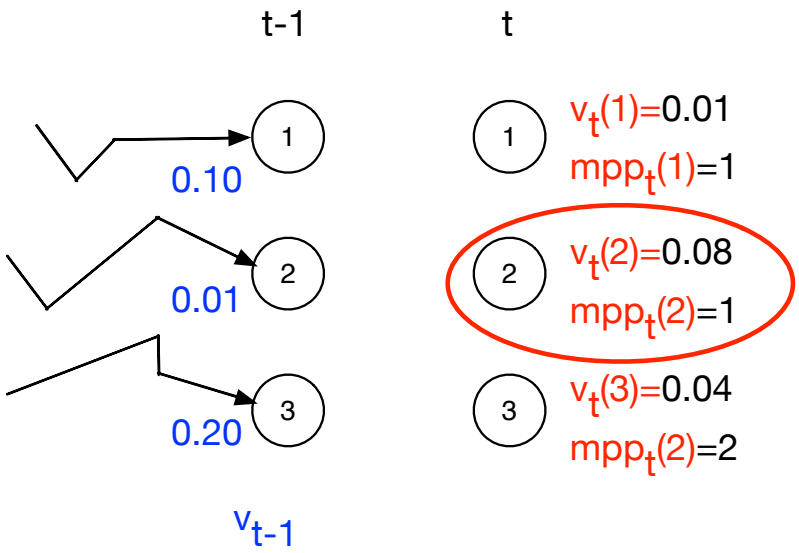
$v_t(2)$=max(0.8*0.10,
0.2*0.01,
0.20*0.2)

$mpp_t(2)$=argmax(0.8*0.10,
0.2*0.01,
0.20*0.2

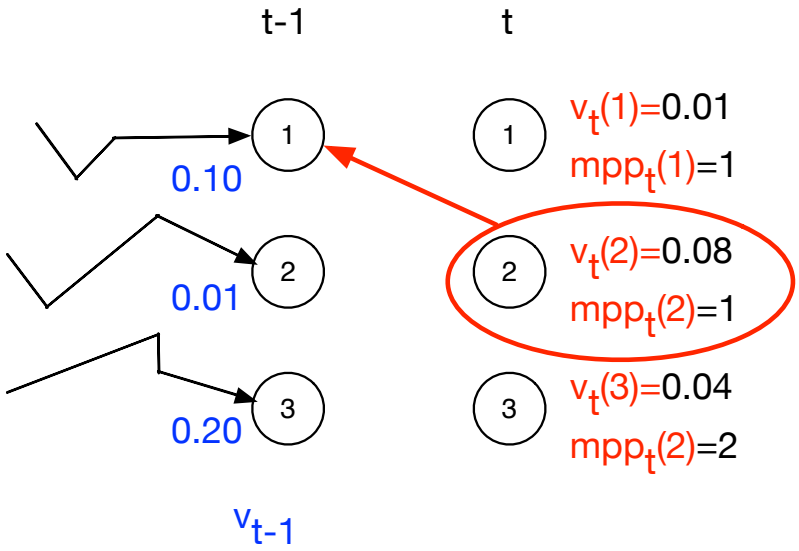
Viterbi example



Viterbi example



Viterbi example



MPP for observations upto time t (backwards):
$(2, 1, mpp_{t-1}(1), mpp_{t-2}(mpp_{t-1}(1)), \ldots)$

In other words, state 2 at time t, state 1 at time $t-1$,...

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

Begin with $t = 1$

$$\begin{aligned} v_1(k) &= P(y_1, x_1 = k) \\ &= P(y_1 | x_1 = k) P(x_1 = k) \\ &= e_k(y_1) \pi_k \end{aligned}$$

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

Assume we have computed $v_{t-1}(l)$.

We have computed the probability of the MPP of length $t-1$ that ends in state l for all values of l

How do we use this to compute the probability of MPP of length t that ends in state k?

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

The most probable path with last two states (l, k) is the most probable path with state l at time $t-1$ followed by a transition from state l to state k and emitting the observation at time t.
What is the probability of this path?

$$v_{t-1}(l)P(X_t = k|X_{t-1} = l)P(y_t|X_t = k)$$
$$= v_{t-1}(l)q_{lk}e_t(y_t)$$

So the most probable path that ends in state k at time t is obtained by maximizing over all possible states l in the previous time $t-1$.

$$v_t(k) = \max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

Also keep a pointer to the state that lead to the current state

$$mpp_t(k) = l^*$$
$$l^* = \arg\max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

Continue till we compute $v_T(k), k \in \{1, \ldots, K\}$. Let:

$$k^* = \arg\max_k v_T(k)$$

To obtain the MPP, follow the pointers defined by $mpp_t(k)$.

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

- The cost of computing this is $\mathsf{O}(K)$ for a given k.
- The total cost of computing this is $\mathsf{O}(K^2)$ for all k.
- Total cost of computing $v_T(k)$ is $\mathsf{O}(TK^2)$.

Other HMM computations

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the probability that state at time t is k ?

$$P(X_t = k | y_{1:T})$$

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the probability of the observations ?

$$P(y_{1:T})$$

Can also be computed efficiently using dynamic programming.

Learning HMMs

We assume the parameters are known. Can we learn parameters from data?

Parameters of the HMM

$$\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{Q}, \boldsymbol{E})$$

Here $\boldsymbol{E}$ is the matrix of emission probabilities. $E_{kb} = e_k(b)$.

Given training data of observed states $(y_{1:T})$, find parameters $\boldsymbol{\theta}$ that maximizes the log likelihood.

Learning HMMs

We assume the parameters are known. Can we learn parameters from data?

Our data contains observed and unobserved data and hence is incomplete.

- Observed: $\mathcal{D} = y_{1:T}$
- Unobserved (hidden): $x_{1:T}$

$$\begin{aligned}\widehat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \log P(y_{1:T}|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{x_{1:T}} \log P(y_{1:T}, x_{1:T}|\boldsymbol{\theta})\end{aligned}$$

The objective function $\ell(\boldsymbol{\theta})$ is called the incomplete log likelihood.

We can optimize this function using the EM algorithm (we won't get into the details in this course).

Summary

HMM

- Allows us to model dependencies.
- Can perform computations efficiently using dynamic programming.
- Can learn the parameters using the EM algorithm.
- Is widely used.