ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Ψηφιακή Επεξεργασία Φωνής

Ενότητα 9η: Μοντέλα Μείξης Γκαουσιανών

Στυλιανού Ιωάννης
Τμήμα Επιστήμης Υπολογιστών

CS578- Speech Signal Processing
Lecture 10: Gaussian Mixture Model, GMM

Yannis Stylianou

University of Crete, Computer Science Dept., Multimedia Informatics Lab
yannis@csd.uoc.gr

Univ. of Crete

Outline

1. Gaussian Statistics
2. Statistical Pattern Recognition
   - Bayesian classification
3. Unsupervised training
4. Acknowledgments

Formulas and definitions

- A d-dimensional random variable follows a Gaussian, or Normal, probability law: $x \rightarrow \mathcal{N}(\mu, \Sigma)$

$$g_{(\mu,\Sigma)}(x) = \frac{1}{\sqrt{2\pi}^{d}\sqrt{\det(\Sigma)}}\, e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

  where μ is the mean vector and Σ is the variance-covariance matrix.
- If $x \rightarrow \mathcal{N}(0, I)$ and if $y = \sqrt{\Sigma}\,x + \mu$, then $y \rightarrow \mathcal{N}(\mu, \Sigma)$.
- $\sqrt{\Sigma}$ defines the *standard deviation* of the random variable x. Note this square root is meant in the *matrix sense*.

Formulas and definitions

- A d-dimensional random variable follows a Gaussian, or Normal, probability law: $x \rightarrow \mathcal{N}(\mu, \Sigma)$

$$g_{(\mu,\Sigma)}(x) = \frac{1}{\sqrt{2\pi}^d \sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

  where μ is the mean vector and Σ is the variance-covariance matrix.

- If $x \rightarrow \mathcal{N}(0, I)$ and if $y = \sqrt{\Sigma}\, x + \mu$, then $y \rightarrow \mathcal{N}(\mu, \Sigma)$.
- $\sqrt{\Sigma}$ defines the *standard deviation* of the random variable x. Note this square root is meant in the *matrix sense*.

- A d-dimensional random variable follows a Gaussian, or Normal, probability law: $x \rightarrow \mathcal{N}(\mu, \Sigma)$

$$g_{(\mu,\Sigma)}(x) = \frac{1}{\sqrt{2\pi}^d \sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

  where μ is the mean vector and Σ is the variance-covariance matrix.
- If $x \rightarrow \mathcal{N}(0, I)$ and if $y = \sqrt{\Sigma}\, x + \mu$, then $y \rightarrow \mathcal{N}(\mu, \Sigma)$.
- $\sqrt{\Sigma}$ defines the *standard deviation* of the random variable x. Note this square root is meant in the *matrix sense*.

- **Mean estimator :**

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- **Unbiased covariance estimator :**

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

Formulas and definitions

- **Mean estimator :**

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- **Unbiased covariance estimator :**

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

Formulas and definitions

- **Likelihood :**

$$p(x_i|\theta) = p(x_i|\mu, \Sigma) = g_{(\mu,\Sigma)}(x_i)$$

- **Joint likelihood :**

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) = \prod_{i=1}^{N} p(x_i|\mu, \Sigma) = \prod_{i=1}^{N} g_{(\mu,\Sigma)}(x_i)$$

for $X = \{x_1, x_2, \cdots, x_N\}$ being a set of independent identically distributed (i.i.d.) points

- **Likelihood :**

$$p(x_i|\theta) = p(x_i|\mu, \Sigma) = g_{(\mu,\Sigma)}(x_i)$$

- **Joint likelihood :**

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) = \prod_{i=1}^{N} p(x_i|\mu, \Sigma) = \prod_{i=1}^{N} g_{(\mu,\Sigma)}(x_i)$$

for $X = \{x_1, x_2, \cdots, x_N\}$ being a set of independent identically distributed (i.i.d.) points

Formulas and definitions

- **Log likelihood :**

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \quad \Leftrightarrow \quad \log p(X|\theta) = \sum_{i=1}^{N} \log p(x_i|\theta)$$

- *In the Gaussian case:*

$$\begin{aligned} p(x|\theta) &= \frac{1}{\sqrt{2\pi}^{d}\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \\ \log p(x|\theta) &= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x - \end{aligned}$$

- *Property:*

$$p(x|\theta_1) > p(x|\theta_2) \quad \Leftrightarrow \quad \log p(x|\theta_1) > \log p(x|\theta_2)$$

Formulas and definitions

- **Log likelihood :**

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \quad \Leftrightarrow \quad \log p(X|\theta) = \sum_{i=1}^{N} \log p(x_i|\theta)$$

- *In the Gaussian case:*

$$\begin{aligned} p(x|\theta) &= \frac{1}{\sqrt{2\pi}^{d}\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \\ \log p(x|\theta) &= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x - \end{aligned}$$

- *Property:*

$$p(x|\theta_1) > p(x|\theta_2) \quad \Leftrightarrow \quad \log p(x|\theta_1) > \log p(x|\theta_2)$$

Formulas and definitions

- **Log likelihood :**

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \quad \Leftrightarrow \quad \log p(X|\theta) = \sum_{i=1}^{N} \log p(x_i|\theta)$$

- *In the Gaussian case:*

$$\begin{aligned} p(x|\theta) &= \frac{1}{\sqrt{2\pi}^d \sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \\ \log p(x|\theta) &= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x - \end{aligned}$$

- *Property:*

$$p(x|\theta_1) > p(x|\theta_2) \quad \Leftrightarrow \quad \log p(x|\theta_1) > \log p(x|\theta_2)$$

Outline

1. Gaussian Statistics
2. Statistical Pattern Recognition
   - Bayesian classification
3. Unsupervised training
4. Acknowledgments

DISCUSSION

- A-priori class probability
- Gaussian modeling of classes

Discussion

- A-priori class probability
- Gaussian modeling of classes

- **Bayes' decision rule :**

$$X \in q_k \;\text{ if }\; P(q_k|X,\Theta) \geq P(q_j|X,\Theta),\ \forall j \neq k$$

with $P(q_k|X,\Theta)$ being *a posteriori probability* (while $P(q_k|\Theta)$ is the *a priori probability*) for classes q_k. Note Θ represents the set of all θ.

- **A posteriori probability :**

$$P(q_k|X,\Theta) = \frac{p(X|q_k,\Theta)P(q_k|\Theta)}{p(X|\Theta)}$$

(*Bayes' law*)

- *For speech :*

$$\forall k,\quad P(q_k|X,\Theta) \propto p(X|q_k,\Theta)P(q_k|\Theta)$$

- or in log domain :

$$\log P(q_k|X,\Theta) \simeq \log p(X|q_k,\Theta) + \log P(q_k|\Theta)$$

Formulas and definitions

- **Bayes' decision rule :**

$$X \in q_k \text{ if } P(q_k|X,\Theta) \geq P(q_j|X,\Theta),\ \forall j \neq k$$

with $P(q_k|X,\Theta)$ being *a posteriori probability* (while $P(q_k|\Theta)$ is the *a priori probability*) for classes q_k. Note Θ represents the set of all θ.

- **A posteriori probability :**

$$P(q_k|X,\Theta) = \frac{p(X|q_k,\Theta)P(q_k|\Theta)}{p(X|\Theta)}$$

(*Bayes' law*)

- *For speech :*

$$\forall k,\quad P(q_k|X,\Theta) \propto p(X|q_k,\Theta)P(q_k|\Theta)$$

- or in log domain :

$$\log P(q_k|X,\Theta) \simeq \log p(X|q_k,\Theta) + \log P(q_k|\Theta)$$

- **Bayes' decision rule :**

$$X \in q_k \text{ if } P(q_k|X,\Theta) \geq P(q_j|X,\Theta),\ \forall j \neq k$$

with $P(q_k|X,\Theta)$ being *a posteriori probability* (while $P(q_k|\Theta)$ is the *a priori probability*) for classes q_k. Note Θ represents the set of all θ.

- **A posteriori probability :**

$$P(q_k|X,\Theta) = \frac{p(X|q_k,\Theta)P(q_k|\Theta)}{p(X|\Theta)}$$

(*Bayes' law*)

- *For speech :*

$$\forall k,\quad P(q_k|X,\Theta) \propto p(X|q_k,\Theta)P(q_k|\Theta)$$

- or in log domain :

$$\log P(q_k|X,\Theta) \simeq \log p(X|q_k,\Theta) + \log P(q_k|\Theta)$$

Formulas and definitions

- **Bayes' decision rule :**

$$X \in q_k \ \text{ if } \ P(q_k|X,\Theta) \geq P(q_j|X,\Theta), \ \forall j \neq k$$

with $P(q_k|X,\Theta)$ being *a posteriori probability* (while $P(q_k|\Theta)$ is the *a priori probability*) for classes q_k. Note Θ represents the set of all θ.

- **A posteriori probability :**

$$P(q_k|X,\Theta) = \frac{p(X|q_k,\Theta)P(q_k|\Theta)}{p(X|\Theta)}$$

(*Bayes' law*)

- *For speech :*

$$\forall k, \quad P(q_k|X,\Theta) \propto p(X|q_k,\Theta)P(q_k|\Theta)$$

- or in log domain :

$$\log P(q_k|X,\Theta) \simeq \log p(X|q_k,\Theta) + \log P(q_k|\Theta)$$

Outline

PRELIMINARIES

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

Preliminaries

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

Preliminaries

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

Preliminaries

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

Preliminaries

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

Preliminaries

All unsupervised training algorithm assume:

- a set of models q_k (not necessarily Gaussian), defined by some parameters Θ (means, variances, priors,...);
- a measure of membership, telling to which extent a data point "belongs" to a model;
- the above implicitly defines global criterion of "goodness of fit" of the models to the data, e.g. :
  - in the case of a distance, the models that are globally closer from the data characterize it better;
  - in the case of a probability measure, the models bringing a better likelihood for the data explain it better.
- a "recipe" to update the model parameters in function of the membership information.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:
  $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$
  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :
  $$d_k(x_n) \leq d_l(x_n), \quad \forall l \neq k$$
  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

     $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

     $$d_k(x_n) \leq d_l(x_n), \;\; \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

$$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

$$d_k(x_n) \leq d_l(x_n), \ \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

     $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

     $$d_k(x_n) \leq d_l(x_n), \quad \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

  $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

  $$d_k(x_n) \leq d_l(x_n), \quad \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-MEANS ALGORITHM

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

     $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

     $$d_k(x_n) \leq d_l(x_n), \ \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

- Start with K initial prototypes μ_k, $k = 1, \cdots, K$.
- **Do** :
  1. For each data-point x_n, $n = 1, \cdots, N$, compute:

     $$d_k(x_n) = (x_n - \mu_k)^T (x_n - \mu_k)$$

  2. Assign each data-point x_n to its **closest** prototype μ_k, i.e. assign x_n to the class q_k if :

     $$d_k(x_n) \leq d_l(x_n), \quad \forall l \neq k$$

  3. Replace each prototype with the mean of the data-points assigned to the corresponding class;
  4. Go to 1.
- **Until** : no further change occurs.

K-means algorithm

Global criterion :

$$J = \sum_{k=1}^{K} \sum_{x_n \in q_k} d_k(x_n)$$

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :

       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$

     - update the variances :

       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$

     - update the priors :

       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$

  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :

$$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$

     - update the variances :

$$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$

  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

- Assume K *initial* Gaussian models $\mathcal{N}(\mu_k, \Sigma_k),\ k = 1 \cdots K$, and initial prior probabilities $P(q_k) = 1/K$.
- **Do** :
  1. Classify each data-point to its most probable cluster $q_k^{(old)}$ using Bayes' rule.
  2. Update the parameters :
     - update the means :
       $$\mu_k^{(new)} = \text{mean of the points belonging to } q_k^{(old)}$$
     - update the variances :
       $$\Sigma_k^{(new)} = \text{variance of the points belonging to } q_k^{(old)}$$
     - update the priors :
       $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{\text{number of training points belonging to } q_k^{(old)}}{\text{total number of training points}}$$
  3. Go to 1.
- **Until** : no further change occurs.

Viterbi-EM algorithm for Gaussians

Global criterion :

$$\mathcal{L}(\Theta) = \sum_{k=1}^{K} \sum_{x_n \in q_k} \log p(x_n|\Theta_k)$$

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step** :
     - update the means :

$$\mu_k^{(new)} = \frac{\sum_{n=1}^{N} x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances :

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step** :
     - update the means :

$$\mu_k^{(new)} = \frac{\sum_{n=1}^{N} x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances :

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step** :
     - update the means :

$$\mu_k^{(new)} = \frac{\sum_{n=1}^{N} x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances :

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N}\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step** :
     - update the means :

$$\mu_k^{(new)} = \frac{\sum_{n=1}^N x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances :

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do**:
  1. **Estimation step**:

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step**:
     - update the means:

$$\mu_k^{(new)} = \frac{\sum_{n=1}^N x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances:

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors:

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do**:
  1. **Estimation step**:

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step**:
     - update the means:

$$\mu_k^{(new)} = \frac{\sum_{n=1}^{N} x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances:

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors:

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N}\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :

$$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$

  2. **Maximization step** :
     - update the means :

$$\mu_k^{(new)} = \frac{\sum_{n=1}^{N} x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the variances :

$$\Sigma_k^{(new)} = \frac{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})}$$

     - update the priors :

$$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$

  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :
  $$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$
  2. **Maximization step** :
     - update the means :
     $$\mu_k^{(new)} = \frac{\sum_{n=1}^N x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$
     - update the variances :
     $$\Sigma_k^{(new)} = \frac{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$
     - update the priors :
     $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N} \sum_{n=1}^{N} P(q_k^{(old)}|x_n, \Theta^{(old)})$$
  3. Go to 1

EM algorithm for Gaussian clustering

- Assume K *initial* models $\mathcal{N}(\mu_k, \Sigma_k)$, with $P(q_k) = 1/K$.
- **Do** :
  1. **Estimation step** :
  $$P(q_k^{(old)}|x_n, \Theta^{(old)}) = \frac{P(q_k^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_k^{(old)}, \Sigma_k^{(old)})}{\sum_j P(q_j^{(old)}|\Theta^{(old)}) \cdot p(x_n|\mu_j^{(old)}, \Sigma_j^{(old)})}$$
  2. **Maximization step** :
     - update the means :
     $$\mu_k^{(new)} = \frac{\sum_{n=1}^N x_n P(q_k^{(old)}|x_n, \Theta^{(old)})}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$
     - update the variances :
     $$\Sigma_k^{(new)} = \frac{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})(x_n - \mu_k^{(new)})(x_n - \mu_k^{(new)})^T}{\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})}$$
     - update the priors :
     $$P(q_k^{(new)}|\Theta^{(new)}) = \frac{1}{N}\sum_{n=1}^N P(q_k^{(old)}|x_n, \Theta^{(old)})$$
  3. Go to 1

EM algorithm for Gaussian clustering

Global criterion :

$$\mathcal{L}(\Theta) = \log \sum_{k=1}^{K} P(q_k|X,\Theta)p(X|\Theta)$$

Outline

1. Gaussian Statistics
2. Statistical Pattern Recognition
   - Bayesian classification
3. Unsupervised training
4. Acknowledgments

Acknowledgments

Most of this material is from Hervé Bourlard's course on Speech processing and speech recognition

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως **Μη Εμπορική** ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Στυλιανού Ιωάννης. «Ψηφιακή Επεξεργασία Φωνής. Μοντέλα Μείξης Γκαουσιανών». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://www.csd.uoc.gr/~hy578