Machine Learning, One in Depth: Gaussian Mixture Models

Mark Hasegawa-Johnson

University of Illinois

2015/06/23
Outline

1. Intro
2. GMM
3. Gaussians
4. Training
5. Testing
6. MCE
7. Conclusion
8. Extras
Goal of most software:

\[ \vec{x} = \text{some input, } \vec{x} \in \mathcal{X} \]

ALGORITHM

\[ z = h(\vec{x}) \]

\[ z = \text{some output, } z \in \mathcal{Z} \]

CLASSIFIER: \( \mathcal{Z} = \{ \text{set of all class labels} \} \)
\( \text{(person names; object tags; sequence of words; sequence of phonemes)} \)

REGRESSOR: \( \mathcal{Z} = \{ \text{real numbers or vectors} \} \)
\( \text{(person’s age; tomorrow’s stock price; second-language fluency rating; positions of tongue, velum, and glottis)} \)
Goal of most software:

\[ \vec{x} = \text{some input, } \vec{x} \in \mathcal{X} \quad \Rightarrow \quad \text{ALGORITHM} \quad z = h(\vec{x}) \quad \Rightarrow \quad z = \text{some output, } z \in \mathcal{Z} \]

Old School:

**DESIGN:** Try to figure out \( z = h(\vec{x}) \) in your brain, then write a program to implement \( z = h(\vec{x}) \).

**TEST:** Try it on some examples. How’d you do?
Goal of most software:

\[ \vec{x} \text{ = some input, } \vec{x} \in \mathcal{X} \quad \rightarrow \quad \text{ALGORITHM} \quad z = h(\vec{x}) \quad \rightarrow \quad z \text{ = some output, } z \in \mathcal{Z} \]

Machine Learning:

**DESIGN:** Figure out some parameterized function class 
\[ z = h(\vec{x}, \vec{\theta}) \text{ that’s sufficiently general s.t., for some parameter vector } \vec{\theta}, h(\cdot) \text{ will fit the data.} \]

**TRAIN:** Learn \( \vec{\theta} \) from a training dataset 
\[ \mathcal{D} = \{(x_1, z_1), \ldots, (x_n, z_n)\} \]

**TEST:** Try it on some examples. How’d you do?
Today’s Function Class: Gaussian Mixture Models (GMM)

\[ h(\vec{x}) = E [z | \vec{x}] \text{, or arg max } p(\vec{x} | z), \text{ or something like that} \]

\[ p(\vec{x} | z) = \sum_{k=1}^{K} c_k \mathcal{N} (\vec{x} | \vec{\mu}_k, \Sigma_k) \]

\[ \mathcal{N} (\vec{x} | \vec{\mu}_k, \Sigma_k) = \frac{1}{\sqrt{\det(2\pi \Sigma_k)}} e^{-\frac{1}{2}(\vec{x} - \vec{\mu}_k)^T \Sigma_k^{-1}(\vec{x} - \vec{\mu}_k)} \]

- \( c_k \) = \( k \)th mixture weight
- \( \vec{\mu}_k \) = \( k \)th mean vector
- \( \Sigma_k \) = \( k \)th covariance matrix
Example: 1-D

**Training Data**

**Gaussian Approximation**

**2-Gaussian Mixture Model (2-GMM)**
Example: 2-D
1-D Gaussian you learned in probability class

**Formula**

\[
p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma}\right)^2}
\]

**Picture**

Keypoints of a 1-D Gaussian PDF
2-D Gaussian w/Diagonal Covariance

Definitions

\[ \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \]

A few facts about linear algebra

\[ \det(\Sigma) = \sigma_2^2 \sigma_2^2, \quad \Sigma^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \]

Formula

\[ \frac{1}{\sqrt{\det(2\pi\Sigma)}} e^{-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})} = \prod_{j=1}^{2} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2} \frac{(x_j - \mu_j)^2}{\sigma_j^2}} \]
2D Gaussian, Diagonal Covariance (contour plot)
First useful fact about linear algebra:
Every non-trivial covariance matrix is “positive definite,”
meaning that it has non-negative eigenvalues...

\[
\Sigma = \begin{bmatrix}
\sigma_1^2 & \rho_{12} \\
\rho_{21} & \sigma_2^2
\end{bmatrix} = [\vec{v}_1, \vec{v}_2] \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} [\vec{v}_1, \vec{v}_2]^T
\]

...which specify the determinant and the inverse

\[
\det(\Sigma) = \lambda_1 \lambda_2, \quad \Sigma^{-1} = [\vec{v}_1, \vec{v}_2] \begin{bmatrix}
\frac{1}{\lambda_1} & 0 \\
0 & \frac{1}{\lambda_2}
\end{bmatrix} [\vec{v}_1, \vec{v}_2]^T
\]

Formula

\[
\frac{1}{\sqrt{\det(2\pi \Sigma)}} e^{-\frac{1}{2} (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})} = \prod_{j=1}^{2} \frac{1}{\sqrt{2\pi \lambda_j}} e^{-\frac{1}{2} \frac{(\vec{v}_j^T \vec{x} - \vec{v}_j^T \vec{\mu})^2}{\lambda_j}}
\]
2D Gaussian, Full Covariance (contour plot)
Outline

1. Intro
2. GMM
3. Gaussians
4. Training
5. Testing
6. MCE
7. Conclusion
8. Extras
Maximum Likelihood Estimation of a Gaussian Model

Maximize the Likelihood

- Given some data $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_n\}$
- Find $\vec{\theta} = \begin{bmatrix} \vec{\mu} \\ \Sigma \end{bmatrix}$ to maximize $p(\mathcal{D} | \vec{\theta}) = \prod_{i=1}^{n} p(\vec{x}_i | \vec{\theta})$

Maximize the Log Likelihood

$$\vec{\theta} = \arg \max \log p(\mathcal{D} | \vec{\theta}) = \arg \max \sum_{i=1}^{n} \log p(\vec{x}_i | \vec{\theta})$$

$$= \arg \max \sum_{i=1}^{n} -\frac{1}{2} \left( (\vec{x}_i - \vec{\mu})^T \Sigma^{-1} (\vec{x}_i - \vec{\mu}) + \log \det(2\pi \Sigma) \right)$$
ML Estimate of Mean = “Sample Mean”

$$\hat{\mu}_{ML} = \frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i$$

ML Estimate of Covariance = “Sample Covariance”

$$\Sigma_{ML} = \frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i - \bar{\mu})(\tilde{x}_i - \bar{\mu})^T$$

...which is just the matrix way of saying that...

$$\sigma_{j,ML}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \mu_j)^2$$

$$\rho_{jk,ML} = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \mu_j)(x_{ik} - \mu_k)$$
2D Gaussian, Full Covariance (contour plot)
...and now, let's learn a Gaussian Mixture Model
GMM Definition

\[ p(\tilde{x}) = \sum_{k=1}^{K} c_k \mathcal{N}(\tilde{x} | \tilde{\mu}_k, \Sigma_k) \]

Parameter Vector

\[ \tilde{\theta} = [c_1, c_2, \tilde{\mu}_1, \tilde{\mu}_2, \Sigma_1, \Sigma_2]^T \]

Maximum Likelihood Estimation

\[ \tilde{\theta}_{ML} = \arg \max \log p(D | \tilde{\theta}) \]

\[ = \arg \max \sum_{i=1}^{n} \log \left( \sum_{k=1}^{K} c_k \mathcal{N}(\tilde{x} | \tilde{\mu}_k, \Sigma_k) \right) \]
GMMs are Different: There’s No Single Best Solution

Example

Swap $c_1 \leftrightarrow c_2$, and $\vec{\mu}_1 \leftrightarrow \vec{\mu}_2$, and $\Sigma_1 \leftrightarrow \Sigma_2$, then $p(D|\vec{\theta})$ is unchanged!!

Example Pictures: Symmetry of a 2-GMM Under Swap
No Global Optimum $\Rightarrow$ Find a Local Optimum

**Local Optimum: Algorithm**

**Pseudocode**

1. Start with initial guess $\vec{\theta}$
2. Find some $\hat{\theta}$ such that $p(D|\hat{\theta}) > p(D|\vec{\theta})$
3. Repeat until $\vec{\theta}$ stops changing
Find a Local Optimum: Gradient Ascent

Gradient Ascent: Algorithm Definition

\[ \hat{\theta} = \tilde{\theta} + \eta \nabla_{\tilde{\theta}} \log p(D|\tilde{\theta}) \]

Practical Issues

- Guaranteed to work if \( \eta \) is small enough
- Problem: what’s a good value of \( \eta \)?
  - \( \eta \) too large \( \Rightarrow \) \( \hat{\theta} \) jumps right over the local optimum
  - \( \eta \) too small \( \Rightarrow \) gradient ascent takes a very long time to converge
- Usual approach: choose \( \eta = 0.002 \) and cross your fingers
  - If \( \tilde{\theta} \) eventually stops changing: Hooray! You found a local optimum!
  - If \( \tilde{\theta} \) never stops changing: Reduce \( \eta \) and try again.
EM: Algorithm Definition

\[ \hat{\theta} = \arg \max Q(\vec{\theta}, \hat{\theta}), \]

where \( Q(\vec{\theta}, \hat{\theta}) \) is some function chosen so that

\[ Q(\vec{\theta}, \hat{\theta}) \text{ has a global optimum} \]
\[ Q(\vec{\theta}, \hat{\theta}) \leq \log p(D|\hat{\theta}) \text{ for all values of } \hat{\theta} \]

and at the starting point \( \hat{\theta} = \vec{\theta} \),

\[ Q(\vec{\theta}, \vec{\theta}) = \log p(D|\vec{\theta}) \text{ and } \nabla_{\hat{\theta}} Q(\vec{\theta}, \hat{\theta}) = \nabla_{\hat{\theta}} \log p(D|\hat{\theta}) \]

Practical Issues

- **PROBLEM**: Can we find such a \( Q \) function?
- **ANSWER**: For GMM and HMM, yes. For most other models, no.
Find a Local Optimum: Expectation Maximization (EM)
EM Algorithm for the GMM

E-Step: Expected frequency of each Gaussian

\[ \gamma_i(k) = \operatorname{Pr}\{k\text{th Gaussian}|\vec{x}_i\} = \frac{c_k \mathcal{N}(\vec{x}_i|\vec{\mu}_k, \Sigma_k)}{\sum_{\ell=1}^{K} c_\ell \mathcal{N}(\vec{x}_i|\vec{\mu}_\ell, \Sigma_\ell)} \]

\[ n_k = E\left[\text{# occurrences } k\text{th Gaussian}\right] = \sum_{i=1}^{n} \gamma_i(k) \]

M-Step: Parameters that Maximize the Q Function

\[ \hat{c}_k = \frac{n_k}{n}, \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{n} \gamma_i(k) \vec{x}_i \]

\[ \hat{\Sigma}_k = \frac{1}{n_k} \sum_{i=1}^{n} \gamma_i(k)(\vec{x}_i - \hat{\mu}_k)(\vec{x}_i - \hat{\mu}_k)^T \]
Outline

1. Intro
2. GMM
3. Gaussians
4. Training
5. Testing
6. MCE
7. Conclusion
8. Extras
Classifying with a GMM

TRAINING a GMM Classifier

GIVEN: \( \mathcal{D}_1 = \{ \vec{x}_1, \ldots \vec{x}_n \} \) examples of class \( z = +1 \),
GIVEN: \( \mathcal{D}_{-1} = \{ \vec{x}_{n+1}, \ldots \vec{x}_{2n} \} \) examples of class \( z = -1 \),
LEARN: \( p(\vec{x}|z) = \sum_{k=1}^{K} c_{zk} \mathcal{N}(\vec{x}|\mu_{zk}, \Sigma_{zk}) \)

TESTING a GMM Classifier

\( LLR = \log \frac{p(\vec{x}|z = +1)p(z = +1)}{p(\vec{x}|z = -1)p(z = -1)} \)

\( h(\vec{x}) = \begin{cases} +1 & \text{if } LLR > 0 \\ -1 & \text{if } LLR < 0 \end{cases} \)
Training Data

GMMs, as function of test datum x

LLR, as function of test datum x

Classifier Output Label, for test datum x
Outline

1. Intro
2. GMM
3. Gaussians
4. Training
5. Testing
6. MCE
7. Conclusion
8. Extras
The EM Mismatch Problem

TRAINING Criterion: Maximum Likelihood

LOG LIKELIHOOD: \( \mathcal{L}(\vec{x}_i, \vec{\theta}) = \log p(\vec{x}_i | \vec{\theta}) \)

TESTING Criterion: Minimum Classification Error

CLASSIFICATION ERROR: \( \ell(\vec{x}_i, \vec{\theta}) = \begin{cases} 
1 & h(\vec{x}_i) \neq z_i \\
0 & h(\vec{x}_i) = z_i 
\end{cases} \)

MISMATCH

Maximum likelihood training might not minimize classification error. The lab has a striking example of this.
**Minimum Classification Error (MCE) Training**

(1) Write classification error as a function of $\vec{\theta}$

$z_i \in \{-1, +1\}$ is the correct label of $\vec{x}_i$.

$$E_n = \frac{1}{n} \left( \# \text{ for which } LLR(\vec{x}_i) \text{ and } z_i \text{ have opposite signs} \right)$$

$$= \frac{1}{n} \sum_{i=1}^{n} u(z_i LLR(\vec{x}_i))$$

where $u(b) = \begin{cases} 
1 & b > 0 \\
0 & b < 0 
\end{cases}$ is the unit step function.
(2) Approximate with a differentiable function

\[ \tilde{E}_n = \frac{1}{n} \sum_{i=1}^{n} \sigma(z_i LLR(\hat{x}_i)) \]

where \( \sigma(b) = 1/(1 + e^{-b}) \) is called the logistic function, and is a differentiable approximation of the unit step function:

- \( \lim_{b \to \infty} \sigma(b) = 1 \)
- \( \lim_{b \to -\infty} \sigma(b) = 0 \)
- \( \sigma(0) = \frac{1}{2} \)
Minimum Classification Error (MCE) Training

(3) Gradient Descent

$$\hat{\theta} = \bar{\theta} - \eta \nabla_{\theta} \tilde{E}_n$$

Practical Issues

- From this point on, MCE training looks exactly like training a neural net!
- So let’s take a break, then talk about how to train neural nets.
Key Points: GMM

**MACHINE LEARNING:** Instead of writing a set of if-then statements to figure out the right label, we create a function $h(\vec{x}, \vec{\theta})$, then learn the parameter vector from data.

**GMM:** a GMM is a universal approximator, meaning that, as $K \to \infty$, a GMM can represent any real-world probability distribution.

**Gaussians:** maximum likelihood training of a Gaussian is just the sample mean and sample covariance.

**Training** a GMM is harder, because there’s no global optimum. EM algorithm efficiently computes a local optimum.

**Testing:** if $LLR > 0$, call it class 1, otherwise call it class -1. MCE training explicitly minimizes classification error.
Regression with a GMM

**TRAINING a GMM Regression**

**GIVEN:** \( \mathcal{D} = \{ (\vec{x}_1, \vec{z}_1), \ldots, (\vec{x}_n, \vec{z}_n) \} \)

**LEARN:** \( p \left( \begin{bmatrix} \vec{x} \\ \vec{z} \end{bmatrix} \right) = \sum_{k=1}^{K} c_k \mathcal{N} \left( \begin{bmatrix} \vec{x} \\ \vec{z} \end{bmatrix} | \begin{bmatrix} \vec{\mu}_{x,k} \\ \vec{\mu}_{z,k} \end{bmatrix}, \begin{bmatrix} \Sigma_{xx,k} & \Sigma_{xz,k} \\ \Sigma_{zx,k} & \Sigma_{zz,k} \end{bmatrix} \right) \)

**TESTING a GMM Regression**

\( \tilde{z}(\vec{x}) = E [\vec{z} | \vec{x}] = \sum_{k=1}^{K} \gamma_{\vec{x}}(k) \left( \vec{\mu}_{z,k} + \Sigma_{zx,k} \Sigma_{xx,k}^{-1} \left( \vec{x} - \vec{\mu}_{x,k} \right) \right) \)