Clustering with Gaussian Mixtures

Andrew W. Moore
Associate Professor
School of Computer Science
Carnegie Mellon University

Unsupervised Learning

• You walk into a bar.
  A stranger approaches and tells you:
  "I've got data from k classes. Each class produces
  observations with a normal distribution and variance
  \( \sigma^2 \). Standard simple multivariate gaussian
  assumptions. I can tell you all the \( P(w) \)'s ."

• So far, looks straightforward.
  "I need a maximum likelihood estimate of the \( \mu \)'s ."

• No problem:
  "There's just one thing. None of the data are labeled. I
  have datapoints, but I don't know what class they're
  from (any of them!)"

• Uh oh!!

The GMM assumption

• There are k components. The
  i'th component is called \( \omega_i \)
• Component \( \omega_i \) has an
  associated mean vector \( \mu_i \)
• Each component generates data
  from a Gaussian with mean \( \mu_i \)
  and covariance matrix \( \sigma^2 \)

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random.
2. Choose component i with probability \( P(\omega_i) \).
3. Generate a datapoint from a Gaussian with mean \( \mu_i \) and covariance matrix \( \sigma^2 \)
The GMM assumption

- There are $k$ components. The $i$'th component is called $\omega_i$.
- Component $\omega_i$ has an associated mean vector $\mu_i$.
- Each component generates data from a Gaussian with mean $\mu_i$ and covariance matrix $\sigma_i^2 I$.

Assume that each datapoint is generated according to the following recipe:
1. Pick a component at random. Choose component $i$ with probability $P(\omega_i)$.
2. Datapoint $\sim N(\mu_i, \sigma_i^2 I)$.

The General GMM assumption

- There are $k$ components. The $i$'th component is called $\omega_i$.
- Component $\omega_i$ has an associated mean vector $\mu_i$.
- Each component generates data from a Gaussian with mean $\mu_i$ and covariance matrix $\Sigma_i$.

Unsupervised Learning: not as hard as it looks

- Sometimes easy
- Sometimes impossible
- and sometimes in between

Computing likelihoods in unsupervised case

We have $x_1, x_2, \ldots, x_N$.
We know $P(w_1) \cdot P(w_2) \ldots P(w_k)$. We know $\sigma$.

$$P(x|w_i, \mu_1, \mu_2, \ldots, \mu_k) = \text{Prob that an observation from class } w_i \text{ would have value } x \text{ given class means } \mu_1, \ldots, \mu_k$$

Can we write an expression for that?

[YES, IF WE ASSUME THE X'S WERE DRAWN INDEPENDENTLY]

Unsupervised Learning: Mediumly Good News

We now have a procedure s.t. if you give me a guess at $\mu_1, \mu_2, \ldots, \mu_k$.
I can tell you the prob of the unlabeled data given those $\mu$'s.

Suppose $x$'s are 1-dimensional.
(From Duda and Hart)

There are two classes; $w_1$ and $w_2$.
$P(w_1) = 1/3 \quad P(w_2) = 2/3 \quad \sigma = 1$.

There are 25 unlabeled datapoints

Data Scattergram

$[x_i, y_i] = [...]$
Graph of
\( \log P(x_1, x_2, \ldots, x_{25} \mid \mu_1, \mu_2) \)
against \( \mu_1 \) (→) and \( \mu_2 \) (↑).

Max likelihood = (\( \mu_1 = -2.13 \), \( \mu_2 = 1.668 \))
Local minimum, but very close to global at (\( \mu_1 = 2.085 \), \( \mu_2 = -1.257 \))∗
∗ corresponds to switching \( w_1 + w_2 \).

Finding the max likelihood \( \mu_1, \mu_2, \ldots, \mu_k \)
We can compute \( P(\text{data} \mid \mu_1, \mu_2, \ldots, \mu_k) \)
How do we find the \( \mu_i \)'s which give max. likelihood?

• The normal max likelihood trick:
  Set \( \frac{\partial}{\partial \mu_i} \log \text{Prob (….)} = 0 \)
  and solve for \( \mu_i \)'s.
  # Here you get non-linear non-analytically-solvable equations
• Use gradient descent
  Slow but doable
• Use a much faster, cuter, and recently very popular method…

The E.M. Algorithm
• We’ll get back to unsupervised learning soon.
• But now we’ll look at an even simpler case with hidden information.
• The EM algorithm
  ❑ Can do trivial things, such as the contents of the next few slides.
  ❑ An excellent way of doing our unsupervised learning problem, as we’ll see.
  ❑ Many, many other uses, including inference of Hidden Markov Models (future lecture).

Silly Example
Let events be “grades in a class”
\( w_1 = \text{gets an A} \quad P(A) = \frac{1}{2} \)
\( w_2 = \text{gets a B} \quad P(B) = \mu \)
\( w_3 = \text{gets a C} \quad P(C) = 2\mu \)
\( w_4 = \text{gets a D} \quad P(D) = \frac{1}{2} - 3\mu \)
(Note \( 0 \leq \mu \leq 1/6 \))

Assume we want to estimate \( \mu \) from data. In a given class there were

\begin{align*}
\text{a A's} & \\
\text{b B's} & \\
\text{c C's} & \\
\text{d D's} & \\
\end{align*}

What's the maximum likelihood estimate of \( \mu \) given \( a, b, c, d \)?
Computing

\begin{align*}
P(A) &= \frac{1}{2} \quad P(B) = \mu \\ P(C) &= 2\mu \\ P(D) &= \frac{1}{2} - 3\mu \\
\end{align*}

\begin{align*}
P(a,b,c,d | \mu) &= K(\frac{1}{2})^a(\mu)^b(2\mu)^c(\frac{1}{2} - 3\mu)^d \\
\log P(a,b,c,d | \mu) &= \log K + a \log \frac{1}{2} + b \log \mu + c \log 2\mu + d \log \left(\frac{1}{2} - 3\mu\right) \\
\end{align*}

\[
\frac{\partial \log P}{\partial \mu} = \frac{2c}{\mu} - \frac{3d}{2\mu} - 3\mu = 0 \\
\]

Gives max like \( \mu = \frac{b + c}{6(b + c + d)} \)

So if class got

\[
\begin{array}{cccc}
A & B & C & D \\
14 & 6 & 5 & 10 \\
\end{array}
\]

Max like \( \mu = \frac{1}{10} \)

---

**Same Problem with Hidden Information**

Someone tells us that

- Number of High grades (A's + B's) = \( h \)
- Number of C's = \( c \)
- Number of D's = \( d \)

What is the max. like estimate of \( \mu \) now?

- Define \( \mu(t) \) the estimate of \( \mu \) on the \( t \)th iteration
- Define \( b(t) \) the estimate of \( b \) on \( t \)th iteration

\[
\mu(t) = \text{initial guess} \\
b(t) = \frac{\mu(t)}{\frac{1}{2} + \mu(t)} = E[b | \mu(t)] \\
\mu(t + 1) = \frac{b(t) + c}{6(b(t) + c + d)} = \text{max like est of } \mu \text{ given } b(t)
\]

Continuous iterating until converged.

**Good news:** Converging to local optimum is assured.

**Bad news:** I said "local" optimum.

---

**E.M. for our Trivial Problem**

We begin with a guess for \( \mu \)

We iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of \( \mu \) and \( a \) and \( b \).

\[
\begin{align*}
\text{E-step} & \quad \mu(t) = \frac{b(t)}{b(t) + c + d} \\
\text{M-step} & \quad b(t + 1) = \frac{b(t) + c}{6(b(t) + c + d)} \\
\end{align*}
\]

---

**E.M. Convergence**

- Convergence proof based on fact that \( \text{Prob(data | } \mu) \) must increase or remain same between each iteration \( \text{[not obvious]} \)
- But it can never exceed 1 \( \text{[obvious]} \)
- So it must therefore converge \( \text{[obvious]} \)

In our example, suppose we had

\[
\begin{array}{ccc}
\text{t} & \mu(t) & b(t) \\
0 & 0 & 0 \\
1 & 0.0833 & 2.857 \\
2 & 0.0937 & 3.158 \\
3 & 0.0947 & 3.185 \\
4 & 0.0948 & 3.187 \\
5 & 0.0948 & 3.187 \\
6 & 0.0948 & 3.187 \\
\end{array}
\]

---

**Back to Unsupervised Learning of GMMs**

Remember:

- We have unlabeled data \( x_1, x_2, \ldots, x_K \)
- We know there are \( K \) classes
- We know \( P(w_1), P(w_2), \ldots, P(w_K) \)
- We don't know \( \mu_1, \mu_2, \ldots, \mu_K \)

We can write \( P(x) | \mu, \ldots, \mu_K \)

\[
\begin{align*}
&= P(x_i | \mu_i, \ldots, \mu_K) \\
&= \prod_i P(x_i | \mu_i, \ldots, \mu_K) \\
&= \prod_i \sum_{k=1}^{K} K \exp \left( -\frac{1}{2\sigma_k^2} (x_i - \mu_k)^2 \right) p(w_k) \\
\end{align*}
\]

Copyright © 2001, Andrew W. Moore
E.M. for GMMs

For Max likelihood we know \( \frac{\partial}{\partial \lambda} \log \text{Prob}(\text{data}|\lambda, \mu_j) = 0 \)

Some wild'n’crazy algebra turns this into: "For Max likelihood, for each j,\]

\[
\sum_{i=1}^{R} \frac{p(w_i|k_j, \mu_j, \sigma_k^2)}{\sum_{j=1}^{K} p(w_i|k_j, \mu_j, \sigma_k^2)}
\]

This is \( n \) nonlinear equations in \( \mu_j \)'s."

If, for each \( x_i \) we knew that for each \( w_j \) the prob that \( \mu_j \) was in class \( w_j \) is \( P(w_j|x_i, \mu_1...\mu_K) \) Then... we would easily compute \( \mu_j \).

If we knew each \( \mu_j \) then we could easily compute \( P(w_j|x_i, \mu_1...\mu_j) \) for each \( w_j \) and \( x_i \).

"I feel an EM experience coming on!!"

---

E.M. for GMMs

Iterate. On the \( t \)'th iteration let our estimates be

\[
\lambda_t = \{ \mu_1(t), \mu_2(t) ... \mu_c(t) \}
\]

E-step

Compute "expected" classes of all datapoints for each class

\[
P(w_i|k_j, \lambda_t) = \frac{p(x_i|w_i, \mu_j, \sigma_k^2) P(w_i|\lambda_t)}{\sum_{j=1}^{K} p(x_i|w_i, \mu_j, \sigma_k^2) P(w_i|\lambda_t)}
\]

M-step.

Compute Max. like \( \mu \) given our data's class membership distributions

\[
\mu_j(t+1) = \frac{\sum_{i=1}^{R} P(w_i|k_j, \lambda_t) x_i}{\sum_{j=1}^{K} P(w_i|k_j, \lambda_t)}
\]

---

E.M. for General GMMs

Iterate. On the \( t \)'th iteration let our estimates be

\[
\lambda_t = \{ \mu_1(t), \mu_2(t) ... \mu_c(t), \Sigma_1(t), ... \Sigma_c(t), p_1(t), p_2(t) ... p_c(t) \}
\]

E-step

Compute "expected" classes of all datapoints for each class

\[
P(w_i|k_j, \lambda_t) = \frac{p(x_i|w_i, \mu_j, \Sigma_j)}{p(x_i|\lambda_t)}
\]

M-step.

Compute Max. like \( \mu \) given our data's class membership distributions

\[
\mu_j(t+1) = \frac{\sum_{i=1}^{R} P(w_i|k_j, \lambda_t) x_i}{\sum_{j=1}^{K} P(w_i|k_j, \lambda_t)}
\]

\[
p_j(t+1) = \frac{\sum_{i=1}^{R} P(w_i|k_j, \lambda_t)}{R}
\]

---

Gaussian Mixture Example: Start

Advance apologies: in Black and White this example will be incomprehensible

---

After first iteration
Some Bio Assay data

GMM clustering of the assay data

Resulting Density Estimator

Three classes of assay
(each learned with it's own mixture model)
(Sorry, this will again be semi-useless in black and white)

Resulting Bayes Classifier

Resulting Bayes Classifier, using posterior probabilities to alert about ambiguity and anomalousness

Yellow means anomalous
Cyan means ambiguous
Unsupervised learning with symbolic attributes

It's just a "learning Bayes net with known structure but hidden values" problem. Can use Gradient Descent.

EASY, fun exercise to do an EM formulation for this case too.

Final Comments
- Remember, E.M. can get stuck in local minima, and empirically it DOES.
- Our unsupervised learning example assumed P(w_i)'s known, and variances fixed and known. Easy to relax this.
- It's possible to do Bayesian unsupervised learning instead of max. likelihood.
- There are other algorithms for unsupervised learning. We'll visit K-means soon. Hierarchical clustering is also interesting.
- Neural-net algorithms called "competitive learning" turn out to have interesting parallels with the EM method we saw.

What you should know
- How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data.
- Be happy with this kind of probabilistic analysis.
- Understand the two examples of E.M. given in these notes.

For more info, see Duda + Hart. It's a great book. There's much more in the book than in your handout.

Other unsupervised learning methods
- K-means (see next lecture)
- Hierarchical clustering (e.g. Minimum spanning trees) (see next lecture)
- Principal Component Analysis simple, useful tool
- Non-linear PCA
  - Neural Auto-Associators
  - Locally weighted PCA
  - Others...