Unsupervised Learning with Mixtures of Gaussians

EM Algorithm - continued

Machine Learning – CSE446
Carlos Guestrin
University of Washington
May 20, 2013

Supervised Learning of Mixtures of Gaussians

- Mixtures of Gaussians:
  - Prior class probabilities: \( P(y) \)
  - Likelihood function per class: \( P(x|y=i) \)

- Suppose, for each data point, we know location \( x \) and class \( y \)
  - Learning is easy... 😊

- For prior \( P(y) \)
  \[ p(y=i) = \frac{\text{Count}(y=i) \text{ in data}}{N} \]

- For likelihood function:
  \[ p(x|y=i) = \frac{\sum \text{points in class } i \text{ for } y \text{ in data}}{\sum \text{points in class } i} \]

\[ m_i = \frac{1}{N} \sum_{x \text{ in class } i} x \]

\[ \Sigma_i = \frac{1}{N} \sum_{x \text{ in class } i} (x - m_i)(x - m_i) \]
Unsupervised Learning: not as hard as it looks

Sometimes easy
Sometimes impossible
and sometimes in between

EM: “Reducing” Unsupervised Learning to Supervised Learning

- If we knew assignment of points to classes \( \rightarrow \) Supervised Learning!

- Expectation-Maximization (EM)
  - Guess assignment of points to classes or clusters
  - Recompute model parameters
  - Iterate
Back to Unsupervised Learning of Mixtures of Gaussians – a simple version

A simple case:
We have unlabeled data \( x_1, x_2, \ldots, x_m \)
We know there are \( k \) classes
We know \( P(y_1), P(y_2), \ldots, P(y_k) \)
We don’t know \( \mu_1, \mu_2, \ldots, \mu_k \)

We can write \( P(\text{data} | \mu_1, \ldots, \mu_k) \)

\[
P(x_1, x_2, \ldots, x_m | \mu_1, \ldots, \mu_k) = \prod_{i=1}^{m} p(x_i | \mu_i) P(y=i) 
\]

\[
\propto \left( \prod_{i=1}^{m} \sum_{j=1}^{k} \exp \left( -\frac{1}{2\sigma^2} \left\| x_i - \mu_j \right\|^2 \right) P(y=i) \right)^{\text{prior}} 
\]

\[
\text{optimal objective wrt. } \mu \]

EM for simple version of Mixtures of Gaussians: The E-step

If we know \( \mu_1, \ldots, \mu_k \) → easily compute prob. point \( x_i \) belongs to class \( y=i \)

\[
x_i = (69.8 = 3.49, 44.66 = 3.49) 
\]

\[
p(y=i|x_i, \mu_1, \mu_k) \propto \exp \left( -\frac{1}{2\sigma^2} \left\| x_i - \mu_j \right\|^2 \right) P(y=i) 
\]

\[
p(y=0|x_i, \mu) = 3.7 \approx 0.6 
\]

\[
p(y=1|x_i, \mu) = 3.2 \approx 0.4 
\]

\[
\text{2 data points: } (y=0, x_i) \text{ weight } 0.6 
\quad (y=1, x_i) \text{ weight } 0.4 
\]

©Carlos Guestrin 2005-2013
EM for simple version of Mixtures of Gaussians: The M-step

- If we know prob. point \( x^j \) belongs to class \( y = i \)
  \[ \rightarrow \text{MLE for } \mu_i \text{ is weighted average} \]
  - imagine \( k \) copies of each \( x^j \), each with weight \( P(y = i|x^j) \):
  \[
  \mu_i = \frac{\sum_{j=1}^{n} P(y = i|x^j) x^j}{\sum_{j=1}^{n} P(y = i|x^j)}
  \]

E.M. for Simple version of Mixtures of Gaussians

**E-step**

Compute “expected” classes of all datapoints for each class

\[
p(y = i|x^j, \mu_1, ... \mu_k) \propto \exp\left(-\frac{1}{2\sigma^2} \|x^j - \mu_i\|^2\right) P(y = i)
\]

**M-step**

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

\[
\mu_i = \frac{\sum_{j=1}^{m} P(y = i|x^j) x^j}{\sum_{j=1}^{m} P(y = i|x^j)}
\]
E.M. Convergence

- EM is coordinate ascent on an interesting potential function
- Coord. ascent for bounded pot. func. ! convergence to a local optimum guaranteed

This algorithm is REALLY USED. And in high dimensional state spaces, too. E.G. Vector Quantization for Speech Data

E.M. for axis-aligned GMM

Iterate. On the \( t \)th iteration let our estimates be

\[
\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_k^{(t)} \}
\]

E-step

Compute "expected" classes of all datapoints for each class

\[
P(\mathbf{y} = i | \mathbf{x}^i, \lambda_t) \propto p_i^{(t)} p(\mathbf{x}^i | \mu_i^{(t)}, \Sigma_i^{(t)})
\]

Just evaluate a Gaussian at \( x^i \)

M-step

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

\[
\mu_i^{(t+1)} = \frac{\sum_j p_j \mathbf{y} = j | \mathbf{x}^i, \lambda_t \mathbf{x}^j}{\sum_j p_j \mathbf{y} = j | \mathbf{x}^i, \lambda_t}
\]

\( i = 1 \ldots k \)

\[
p_i^{(t+1)} = \frac{\sum_j p_j \mathbf{y} = j | \mathbf{x}^i, \lambda_t \mathbf{x}^j}{N} \quad N = \#\text{records}
\]
E.M. for General GMMs

Iterate. On the \(t\)th iteration let our estimates be
\[
\lambda_t = \{ \mu_1(t), \mu_2(t) \ldots \mu_k(t), \Sigma_1(t), \Sigma_2(t) \ldots \Sigma_k(t), p_1(t), p_2(t) \ldots p_k(t) \}
\]

E-step
Compute "expected" classes of all datapoints for each class

\[
P(y=i|x', \lambda_t) \propto p_i(t)p(x'|\mu_i(t), \Sigma_i(t))
\]

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

\[
\mu_i^{(t+1)} = \frac{\sum_j p(y=i|x', \lambda_t) \cdot x_j}{\sum_j p(y=i|x', \lambda_t)}
\]

\[
\Sigma_i^{(t+1)} = \frac{\sum_j p(y=i|x', \lambda_t) \cdot (x_j - \mu_i^{(t+1)}) (x_j - \mu_i^{(t+1)})^T}{\sum_j p(y=i|x', \lambda_t)}
\]

\[
p_i^{(t+1)} = \frac{\sum_j p(y=i|x', \lambda_t)}{N}
\]

Gaussian Mixture Example: Start

\[
\text{Shaped by given } \mu, \Sigma, P(y)
\]
After first iteration

After 2nd iteration
After 5th iteration

After 6th iteration
GMM clustering of the assay data

Resulting Density Estimator
Three classes of assay (each learned with its own mixture model)

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

Yellow means anomalous
Cyan means ambiguous

E.M.: The General Case

- E.M. widely used beyond mixtures of Gaussians
  - The recipe is the same...
- Expectation Step: Fill in missing data, given current values of parameters, \( \theta^{(t)} \)
  - If variable \( y \) is missing (could be many variables)
  - Compute, for each data point \( x \), for each value \( i \) of \( y \):
    - \( P(y=i|x, \theta^{(t)}) \)
- Maximization step: Find maximum likelihood parameters for (weighted) "completed data":
  - For each data point \( x \), create \( k \) weighted data points
    - \( \{y:f_i, x_j\} \) with \( f_i \) as the maximum likelihood parameter estimate for this weighted data
- Repeat
What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent

- EM for mixture of Gaussians:
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data

- Be happy with this kind of probabilistic analysis

- Remember, E.M. can get stuck in local minima, and empirically it DOES

- EM is coordinate ascent

Acknowledgements

- K-means & Gaussian mixture models presentation contains material from excellent tutorial by Andrew Moore:
  - http://www.autonlab.org/tutorials/

- K-means Applet:
  - http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html

- Gaussian mixture models Applet:
  - http://www.neurosci.aist.go.jp/%7Eakaho/MixtureEM.html
Dimensionality Reduction
PCA

Machine Learning – CSE446
Carlos Guestrin
University of Washington
May 20, 2013

Dimensionality reduction
- Input data may have thousands or millions of dimensions!
  - e.g., text data has tens of thousands or millions of dimensions.
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover “intrinsic dimensionality” of data
    - high dimensional data that is truly lower dimensional
Lower dimensional projections

- Rather than picking a subset of the features, we can new features that are combinations of existing features

\[
Z_1 = 2.5x_1 + 2.9x_2 - 3.2x_3 \quad \ldots
\]

- Let's see this in the unsupervised setting
  - just \( X \), but no \( Y \)

Linear projection and reconstruction

- Project into 1-dimension
- Reconstruction: only know \( z_1 \), what was \( (x_1, x_2) \)