Expectation Maximization & Regression
Lecture 21

David Sontag
New York University

Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer, Dan Weld, Vibhav Gogate, and Andrew Moore
The General GMM assumption

- \( P(Y) \): There are \( k \) components
- \( P(X|Y) \): Each component generates data from a multivariate Gaussian with mean \( \mu_i \) and covariance matrix \( \Sigma_i \)

Each data point is sampled from a generative process:

1. Choose component \( i \) with probability \( P(y=i) \)
2. Generate datapoint \( \sim N(m_i, \Sigma_i) \)

Gaussian mixture model (GMM)
Mixtures of Gaussians
**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

$$P(Y_j = k | x_j, \lambda_t) \propto p_k^{(t)} p(x_j | \mu_k^{(t)}, \Sigma_k^{(t)})$$

**M-step**

Compute weighted MLE for $\mu$ given expected classes above

$$\mu_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t) x_j}{\sum_j P(Y_j = k | x_j, \lambda_t)}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t) [x_j - \mu_k^{(t+1)}] [x_j - \mu_k^{(t+1)}]^T}{\sum_j P(Y_j = k | x_j, \lambda_t)}$$

$$p_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t)}{m}$$

$p_k^{(t)}$ is shorthand for estimate of $P(y=k)$ on $t'$th iteration

$m = \#training\ examples$
Gaussian Mixture Example: Start
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
After 20th iteration
What if we do hard assignments?

**Iterate:** On the $t'$th iteration let our estimates be

$$
\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)}, \ldots, \mu_K^{(t)} \}
$$

**E-step**

Compute “expected” classes of all datapoints

$$
P(Y_j = k | x_j, \mu_1, \ldots, \mu_K) \propto \exp \left( -\frac{1}{2\sigma^2} \| x_j - \mu_k \|^2 \right) P(Y_j = k)
$$

**M-step**

Compute most likely new $\mu$s given class expectations

$$
\mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k | x_j) x_j}{\sum_{j=1}^{m} P(Y_j = k | x_j)}
$$

$$
\mu_k = \frac{\sum_{j=1}^{m} \delta(Y_j = k, x_j) x_j}{\sum_{j=1}^{m} \delta(Y_j = k, x_j)}
$$

$\delta$ represents hard assignment to “most likely” or nearest cluster

Equivalent to k-means clustering algorithm!!!
The general learning problem with missing data

- **Marginal likelihood:** $X$ is observed, $Z$ (e.g. the class labels $Y$) is missing:

$$
\ell(\theta : D) = \log \prod_{j=1}^{m} P(x_j | \theta) \\
= \sum_{j=1}^{m} \log P(x_j | \theta) \\
= \sum_{j=1}^{m} \log \sum_{z} P(x_j, z | \theta)
$$

- **Objective:** Find $\arg\max_{\theta} l(\theta:\text{Data})$
Properties of EM

• We will prove that
  – EM converges to a local maxima
  – Each iteration improves the log-likelihood

• How? (Same as k-means)
  – E-step can never decrease likelihood
  – M-step can never decrease likelihood
EM pictorially

\[ L(\theta) = l(\theta|\theta_n) \]

\[ L(\theta) \]

\[ L(\theta_{n+1}) \]

\[ l(\theta_{n+1}|\theta_n) \]

\[ L(\theta_n) = l(\theta_n|\theta_n) \]

\[ \theta_n \]

\[ \theta_{n+1} \]

(Figure from tutorial by Sean Borman)
What you should know

• Mixture of Gaussians

• EM for mixture of Gaussians:
  – Coordinate ascent, just like k-means
  – How to “learn” maximum likelihood parameters (locally max. like.) in the case of unlabeled data
  – Relation to K-means
    • Hard / soft clustering
    • Probabilistic model

• Remember, E.M. can get stuck in local minima,
  – And empirically it DOES
Logistic Regression

- Learn $P(Y|X)$ directly!
  - Assume a particular functional form
  - Sigmoid applied to a linear function of the data:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

Logistic function (Sigmoid):

Features can be discrete or continuous!
Naïve Bayes vs. Logistic Regression

**Learning:** $h: X \rightarrow Y$
- $X$ – features
- $Y$ – target classes

**Generative**
- Assume functional form for
  - $P(X|Y)$ assume cond indep
  - $P(Y)$
  - Est. params from train data
- Gaussian NB for cont. features
- Bayes rule to calc. $P(Y|X=x)$:
  - $P(Y | X) \propto P(X | Y) P(Y)$
- **Indirect** computation
  - Can generate a sample of the data
  - Can easily handle missing data

**Discriminative**
- Assume functional form for
  - $P(Y|X)$ no assumptions
  - Est params from training data
- Handles discrete & cont features
- **Directly calculate** $P(Y|X=x)$
  - Can’t generate data sample
Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

• Generative vs. Discriminative classifiers

• Asymptotic comparison
  (# training examples → infinity)
  
  – when model correct
    • NB, Linear Discriminant Analysis (with class independent variances), and Logistic Regression produce identical classifiers

  – when model incorrect
    • LR is less biased – does not assume conditional independence
      – therefore LR expected to outperform NB
Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Non-asymptotic analysis
  - convergence rate of parameter estimates, 
    \( (n = \# \text{ of attributes in } X) \)
    - Size of training data to get close to infinite data solution
    - Naïve Bayes needs \( O(\log n) \) samples
    - Logistic Regression needs \( O(n) \) samples

  - Naïve Bayes converges more quickly to its \( (\text{perhaps less helpful}) \) asymptotic estimates
Some experiments from UCI data sets

Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. $m$ (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naïve Bayes.
Logistic regression for discrete classification

Logistic regression in more general case, where set of possible $Y$ is $\{y_1,\ldots,y_R\}$

- Define a weight vector $w_i$ for each $y_i$, $i=1,\ldots,R-1$

\[
P(Y = 1|X) \propto \exp(w_{10} + \sum_i w_{1i}X_i)
\]

\[
P(Y = 2|X) \propto \exp(w_{20} + \sum_i w_{2i}X_i)
\]

\[\cdots\]

\[
P(Y = r|X) = 1 - \sum_{j=1}^{r-1} P(Y = j|X)
\]
Logistic regression for discrete classification

- Logistic regression in more general case, where $Y$ is in the set \{y_1, \ldots, y_R\}

for $k<R$

$$P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}$$

for $k=R$ (normalization, so no weights for this class)

$$P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}$$

Features can be discrete or continuous!
Prediction of continuous variables

• Billionaire says: Wait, that’s not what I meant!
• You say: Chill out, dude.
• He says: I want to predict a continuous variable for continuous inputs: I want to predict salaries from GPA.
• You say: I can regress that...
Linear Regression

Prediction
\[ \hat{y} = w_0 + w_1 x_1 \]

Prediction
\[ \hat{y} = w_0 + w_1 x_1 + w_2 x_2 \]
Ordinary Least Squares (OLS)

$$\text{total error} = \sum_{i} (y_i - \hat{y}_i)^2 = \sum_{i} \left( y_i - \sum_{k} w_k x_k^{(i)} \right)^2$$

- Observation $y$
- Prediction $\hat{y}$
- Error or “residual”
- Sum over data points
- Features
The regression problem

- Precisely, minimize the residual squared error:

\[
\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_i \left( y_i - \sum_k w_k x_i^k \right)^2
\]
Regression: matrix notation

\[ w^* = \arg \min_{w} \sum_i \left( y_i - \sum_k w_k x_i^k \right)^2 = \sum_i \left( x_i^T w - y_i \right)^2 \]

\[ w^* = \arg \min_{w} (Hw - t)^T (Hw - t) \]

residual error

One data point per row

\[ H = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} \]

\[ w = \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix} \]

\[ t = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \]
Regression solution: simple matrix math

\[ w^* = \arg \min_w (Hw - t)^T (Hw - t) \]

residual error

solution: \[ w^* = (H^T H)^{-1} H^T t = A^{-1} b \]

where \[ A = H^T H \]

K×K matrix of feature correlations

b = H^T t = K×1 vector
But, why?

- Billionaire (again) says: Why sum squared error???
- You say: Gaussians, Dr. Gateson, Gaussians...
- Model: prediction is deterministic linear function plus Gaussian noise:
  \[ y_{\text{observed}} = \sum_{k} w_k x_k + \epsilon \]
  \[ \epsilon \sim \mathcal{N}(0, \sigma^2) \]

- Learn \( w \) using MLE:
  \[
  \Pr(y_{\text{observed}} \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(y_{\text{observed}} - \sum_k w_k x_k)^2}{2\sigma^2}}
  \]
Maximizing log-likelihood

Maximize wrt \(w\):

\[
\ln P(D \mid w, \sigma) = \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N \prod_{j=1}^{N} e^{-\frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}}
\]

\[
\arg \max_w \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N + \sum_{j=1}^{N} -\frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}
\]

\[
= \arg \max_w \sum_{j=1}^{N} -\frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}
\]

\[
= \arg \min_w \sum_{j=1}^{N} [t_j - \sum_i w_i h_i(x_j)]^2
\]

Least-squares Linear Regression is MLE for Gaussian noise!!!