Machine Learning

Gaussian Mixture Models
Discriminative vs Generative Models

- Discriminative: Just learn a decision boundary between your sets.
  
  Support Vector Machines

- Generative: Learn enough about your sets to be able to make new examples that would be set members
  
  Gaussian Mixture Models
The Generative Model POV

- Assume the data was generated from a process we can model as a probability distribution

- Learn that probability distribution

- Once learned, use the probability distribution to
  - “Make” new examples
  - Classify data we haven’t seen before.
Non-parametric distribution not feasible

• Let’s probabilistically model ML student heights.
• Ruler has 200 marks (100 to 300 cm)
• How many probabilities to learn?
• How many students in the class?
• What if the ruler is continuous?
Learning a Parametric Distribution

- Pick a parametric model (e.g. Gaussian)
- Learn just a few parameter values

\[ p(x \mid \Theta) \equiv \text{prob. of } x, \text{ given parameters } \Theta \]

of a model, \( M \)
Using Generative Models for Classification

Gaussians whose means and variances were learned from data

Machine learning students

NBA players

New person. Which class does he belong to?

Answer: the class that calls him most probable.
Learning a Gaussian Distribution

\[ p(x | \Theta) \equiv \text{prob. of } x, \text{ given parameters } \Theta \]

of a model, \( M \)

\[ \Theta \equiv \{ \mu, \sigma \} \]

\[ M \equiv \frac{1}{(2\pi)^{1/2} \sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

The “normal” Gaussian distribution, often denoted \( N \), for “normal”
Goal: Find the best Gaussian

• Hypothesis space is Gaussian distributions.
• Find parameters $\Theta^*$ that maximize the prob. of observing data $X \equiv \{x_1, ... x_n\}$

$$\Theta^* = \arg\max_{\Theta} p(X | \Theta)$$

where each $\Theta \equiv \{\mu, \sigma\}$
Some math

$$\Theta^* = \argmax_{\Theta} p(X | \Theta), \text{where each } \Theta \equiv \{\mu, \sigma\}$$

$$p(X | \Theta) = \prod_{i=1}^{n} p(x_i | \Theta)$$

...if can we assume all $x_i$ are i.i.d.
We get underflow if $n$ is, say, 500

$$p(X | \Theta) \propto \sum_{i=1}^{n} \log(p(x_i | \Theta))$$ solves underflow.
Remember what we’re maximizing

\[ \Theta^* \equiv p(X \mid \Theta) = \sum_{i=1}^{n} \log(p(x_i \mid \Theta)) \]

fitting the Gaussian into this...

\[ \log(p(x \mid \Theta)) = \log \left( \frac{e^{-(x-\mu)^2/2\sigma^2}}{(2\pi)^{1/2}\sigma} \right) \]
Some math gets you…

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

\[
= \frac{-(x-\mu)^2}{2\sigma^2} - \log \sigma - \log(2\pi)^{1/2}
\]

Plug back into equation from slide 11
..which gives us

\[ \Theta^* \equiv \argmax_{\Theta} p(X \mid \Theta) \]

\[ = \sum_{i=1}^{n} \log(p(x_i \mid \Theta)) \]

\[ = \sum_{i=1}^{n} \left( \frac{-(x_i - \mu)^2}{2\sigma^2} - \log\sigma \right) \]
Maximizing Log-likelihood

• To find best parameters, take the partial derivative with respect to parameters \( \{\sigma, \mu\} \) and set to 0.

\[
\Theta^* = \sum_{i=1}^{n} \left( \frac{-(x_i - \mu)^2}{2\sigma^2} - \log \sigma \right)
\]

\( \argmax \Theta \)

• The result is a closed-form solution

\[
\mu = -\frac{1}{n} \sum_{i=1}^{n} x_i \\
\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2
\]
What if...

• ...the data distribution can’t be well represented by a single Gaussian?

• Can we model more complex distributions using multiple Gaussians?
Gaussian Mixture Model (GMM)

Model the distribution as a mix of Gaussians

\[ P(x) = \sum_{j=1}^{K} P(z_j)P(x | z_j) \]

- \( x \) is the observed value
- \( z_j \) is the \( j \)th Gaussian

Machine learning students & NBA players

Two Gaussian components

Height (cm)
What are we optimizing?

\[ P(x) = \sum_{j=1}^{K} P(z_j)P(x \mid z_j) \]

Notating \( P(z_j) \) as weight \( w_j \) and using the Normal (a.k.a. Gaussian) distribution \( N(\mu_j, \sigma_j^2) \) gives us...

\[ = \sum_{j=1}^{K} w_j N(x \mid \mu_j, \sigma_j^2) \quad \text{such that} \quad 1 = \sum_{j=1}^{K} w_j \]

This gives 3 variables per Gaussian to optimize:

\[ w_j, \mu_j, \sigma_j \]
Bad news: No closed form solution.

\[
\Theta^* \equiv p(X \mid \Theta) = \sum_{i=1}^{n} \log(p(x_i \mid \Theta))
\]

argmax \Theta

\[
= \sum_{i=1}^{n} \log \left( \sum_{j=1}^{K} w_j p(x_i \mid N(\mu_j, \sigma^2_j)) \right)
\]

argmax \Theta
Expectation Maximization (EM)

• Solution: The EM algorithm

• EM updates model parameters iteratively.

• After each iteration, the likelihood the model would generate the observed data increases (or at least it doesn’t decrease).

• EM algorithm always converges to a local optimum.
EM Algorithm Summary

• Initialize the parameters

• E step: calculate the likelihood a model with these parameters generated the data

• M step: Update parameters to increase the likelihood from E step

• Repeat E & M steps until convergence to a local optimum.
EM for GMM - Initialization

• Choose the number of Gaussian components $K$
  $K$ should be much less than the number of data points to avoid overfitting.

• (Randomly) select parameters for each Gaussian $j$: $w_j, \mu_j, \sigma_j$

...such that $1 = \sum_{j=1}^{K} w_j$
EM for GMM – Expectation step

The responsibility $\gamma_{j,n}$ of Gaussian $j$ for observation $x_n$ is defined as...

$$
\gamma_{j,n} \equiv p(z_j \mid x_n) = \frac{p(x_n \mid z_j) p(z_j)}{p(x_n)}
$$

$$
= \frac{p(x_n \mid z_j) p(z_j)}{\sum_{k=1}^{K} p(z_k) p(x_n \mid z_k)} = \frac{\sum_{k=1}^{K} w_k N(x_n \mid \mu_k, \sigma_k^2)}{\sum_{k=1}^{K} w_k N(x_n \mid \mu_k, \sigma_k^2)}
$$
EM for GMM – Expectation step

Define the responsibility $\Gamma_j$ of Gaussian $j$ for all the observed data as...

$$\Gamma_j \equiv \sum_{n=1}^{N} \gamma_{j,n}$$

You can think of this as the proportion of the data explained by Gaussian $j$. 
EM for GMM – Maximization step

Update our parameters as follows...

new \( w_j = \frac{\Gamma_j}{N} \)

new \( \mu_j = \frac{\sum_{i=1}^{N} \gamma_{j,i} x_i}{\Gamma_j} \)

new \( \sigma_j^2 = \frac{\sum_{i=1}^{N} \gamma_{j,i} (x_i - \mu_j)^2}{\Gamma_j} \)
Why does this work?

• We need to prove that, as our model parameters are adjusted, likelihood of the data never goes down (monotonically non-decreasing)

• This is the part where I point you to the textbook
What if…

• …our data isn’t just scalars, but each data point has multiple dimensions?

• Can we generalize to multiple dimensions?

• We need to define a covariance matrix.
Covariance Matrix

Given d-dimensional random variable vector \( \mathbf{X} = [X_1, ..., X_d] \)
the covariance matrix denoted \( \Sigma \) (confusing, eh?) is defined as...

\[
\Sigma \equiv \begin{bmatrix}
E[(X_1 - \mu_1)(X_1 - \mu_1)] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & E[(X_1 - \mu_1)(X_d - \mu_d)] \\
E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & E[(X_2 - \mu_2)(X_d - \mu_d)] \\
\vdots & \vdots & \ddots & \vdots \\
E[(X_d - \mu_d)(X_1 - \mu_1)] & E[(X_d - \mu_d)(X_2 - \mu_2)] & \cdots & E[(X_d - \mu_d)(X_d - \mu_d)]
\end{bmatrix}
\]

This is a generalization of one-dimensional variance for a scalar random variable \( X \)
\[
\sigma^2 = \text{var}(X) = E\left[ (X - \mu)^2 \right]
\]
Multivariate Gaussian Mixture

The $d$ by $d$ covariance matrix $\Sigma$ describes the shape and orientation of an ellipse.

$$P(\tilde{X}) = \sum_{j=1}^{K} w_j p(\tilde{X} \mid N(\tilde{\mu}, \Sigma_j))$$

Given $d$ dimensions and $K$ Gaussians, how many parameters?
Example: Initialization

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #1

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #2

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #3

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #4

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #5

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #6

(Illustration from Andrew Moore's tutorial slides on GMM)
After Iteration #20

(Illustration from Andrew Moore's tutorial slides on GMM)
GMM Remarks

• GMM is powerful: any density function can be arbitrarily-well approximated by a GMM with enough components.

• If the number of components $K$ is too large, data will be overfitted.
  – Likelihood increases with $K$.
  – Extreme case: $N$ Gaussians for $N$ data points, with variances $\rightarrow 0$, then likelihood $\rightarrow \infty$.

• How to choose $K$?
  – Use domain knowledge.
  – Validate through visualization.
GMM is a “soft” version of K-means

- **Similarity**
  - $K$ needs to be specified.
  - Converges to some local optima.
  - Initialization matters final results.
  - One would want to try different initializations.

- **Differences**
  - GMM Assigns “soft” labels to instances.
  - GMM Considers variances in addition to means.
GMM for Classification

- Given training data with multiple classes...
  1) Model the training data for each class with a GMM
  2) Classify a new point by estimating the probability each class generated the point
  3) Pick the class with the highest probability as the label.
GMM for Regression

Given dataset $D=\{<x_1, y_1>, ..., <x_n, y_n>\}$, where $y_i \in \mathbb{R}$ and $x_i$ is a vector of $d$ dimensions...

Learn a $d+1$ dimensional GMM.

Then, compute $f(x) = \mathbb{E}[y \mid x]$