Expectation Maximization

November 11, 2020

Table of contents

Acknowledgements

This slide deck borrows heavily from an excellent course on statistical ML by Peter Orbanz.

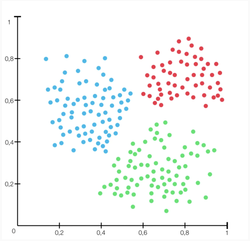
Another important resource was Christopher Bishop's Machine Learning textbook.

The k-means algorithm

Clustering

Problem

- Given: Data $x_1, \dots, x_n$.
- Assumption: Each data point belongs to exactly one group or class. These groups are called **clusters**.
- Our task is to find the clusters, given only the data.



Representation

Fror K clusters, we encode assignments to clusters as a vector $\boldsymbol{z} \in \{1, \dots, K\}^n$ where:

$$\boldsymbol{z}_i = k \iff x_i \text{ assigned to cluster } k$$

A very simple clustering algorithm: K-means

K-means algorithm

- Randomly choose K "cluster centers" (the "means") $\mu_1^{(0)}, \dots, \mu_K^{(0)} \in \mathbb{R}^d$
- Iterate until convergence (j = iteration number):
    1. Assign each x_i to the closest (in Euclidean distance) mean:

$$z_i^{(j+1)} := \arg \min_{k \in \{1,\dots,K\}} ||x_i - \mu_k^{(j)}||$$

    2. Recompute each $\mu_k^{(j)}$ as the mean of all points assigned to it.

$$\mu_k^{(j+1)} := \frac{1}{|i : z_i^{(j+1)} = k|} \sum_{i : z_i^{(j+1)} = k} x_i$$

Convergence Criterion

For example: Terminate when the total change of the means satisfies:

$$\sum_{k=1}^{K} ||\mu_k^{(j+1)} - \mu_k^{(j)}|| < \tau$$

The threshold value τ is set by the user.

Illustration

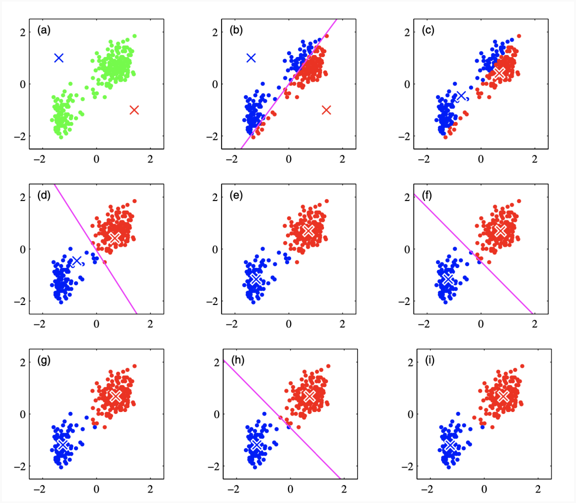


Illustration of the K-means algorithm using the re-scaled Old Faithful dataset (in green). The initial choices for centers μ_1 and μ_2 are shown by the red and blue crosses, respectively.

Image Credit: Christopher Bishop

Q: Look at this for a minute, who wants to try to walk us through what's happening?

Application: Image Segmentation and Clustering

Image Segmentation

Image segmentation is the problem of partitioning an image into "coherent" regions. The problem is not well-posed: Its solution depends on the meaning of "coherent".

K-means on images

- Each pixel is treated as a separate point representing $\{R, G, B\}$ intensities.
- Note: not sophisticated; spatial proximity ignored.

$K = 2$ $K = 3$ $K = 10$ original

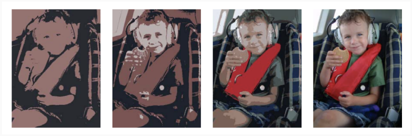


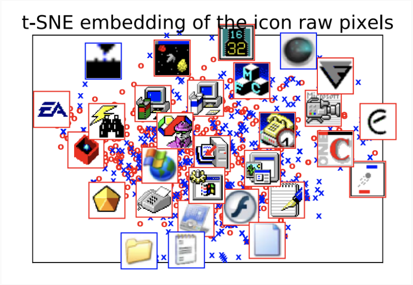
Image Credit: Christopher Bishop

Utility for image compression

(Lossy) image compression is a side effect; one can store only the cluster identity k and the *code-book vectors* $\boldsymbol{\mu}_k$.

Application: Anomaly Detection in Cybersecurity

- Malware not uncommonly contains icons which look like legitimate applications (to try to trick the user into clicking on it.
- Often these icons have slight blurriness, or color shifting, to avoid detection.
- Clustering algorithms (such as k-means, t-SNE or HDBScan) can be used to identify if there is an embedded icon, what it is, and if it looks anomalous.



Python Demo

K-means demo in python

K-Means: Gaussian Interpretation

K Gaussians

Consider the following algorithm:

- Suppose each μ_k is the expected value of a Gaussian density $p(x|\mu_k, \mathbb{I})$ with unit covariance.
- Start with K randomly chosen means and iterate.
  1. Assign each x_i to the Gaussian under which it has the highest density.
  2. Given the assignments, fit $p(x|\mu_k, \mathbb{I})$ by maximum likelihood estimation of μ_k from all points assigned to cluster k.

Comparison to K-means

- Since the Gaussians are spherical with identity covariance, the density $p(x|\mu_k, \mathbb{I})$ is largest for the mean μ_j which is closest to x_i in Euclidean distance. (Why?)
- The maximum likelihood estimator of μ_k is

$$\mu_k^{(j+1)} := \frac{1}{|i : z_i^{(j+1)} = k|} \sum_{i:z_i^{(j+1)}=k} x_i$$

This is precisely the K-means algorithm!

What next

- We will discuss a more sophisticated version of K-means called the *Expectation-Maximization (EM) algorithm.*
- EM gives
  1. A better statistical explanation of what is going on.
  2. A direct generalization to other distributions. We can consider Gaussians with general covariance structure, or other distributions as well.
  3. Better support for the anomaly detection use case.

Mixture Models

Finite mixture models

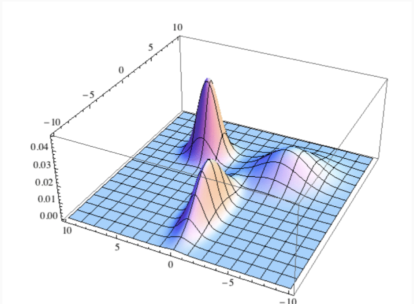
Finite Mixture Model

A finite mixture model is a distribution with density of the form

$$\pi(x) = \sum_{k=1}^{K} c_k \; p(x \mid \theta_k)$$

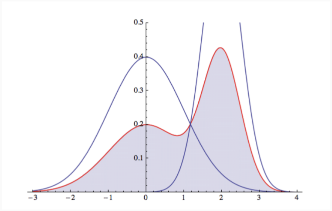
where $\sum_k c_k = 1$ and $c_k \geq 0$.

Example: Finite mixture of Gaussians



Illustration

Mixture of two Gaussians

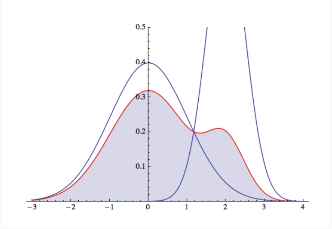


The curve outlined in red is the mixture

$$\pi(x) = 0.5\, p(x \mid 0, 1) + 0.5\, p(x \mid 2, 0.5)$$

where p is the Gaussian density. The blue curves are the component densities.

Influence of the weights



The curve outlined in red is the mixture

$$\pi(x) = 0.8\, p(x \mid 0, 1) + 0.2\, p(x \mid 2, 0.5)$$

Here, the weights $c_1 = c_2 = 0.5$ above have been changed to $c_1 = 0.8$ and $c_2 = 0.2$.

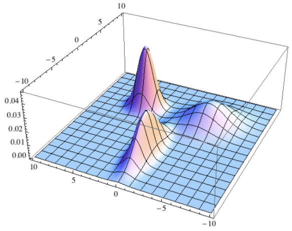
Sampling from a finite mixture

For a finite mixture with fixed parameters c_k and θ_k, the two-step sampling procedure is:

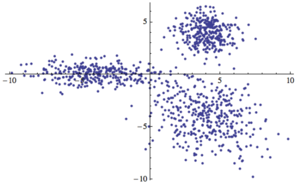
1. Choose a mixture component at random. Each component k is selected with probability c_k.
2. Sample x_i from $p(x|\theta_k)$.

Note: We always repeat both steps, i.e. for x_{i+1}, we choose again choose a (possibly different) component at random.

Note that k-means does not support sampling new points.



Plot of the mixture density.



A sample of size 1000.

Maximum likelihood for finite mixtures

Writing down the maximum likelihood problem is straightforward:

$$(\widehat{\boldsymbol{c}}, \widehat{\boldsymbol{\theta}}) = (\widehat{c}_1, ..., \widehat{c}_K, \widehat{\theta}_1, ..., \widehat{\theta}_K) = \arg\max_{\boldsymbol{c},\boldsymbol{\theta}} \prod_{i=1}^{n} \left(\sum_{k=1}^{K} c_k \; p(x_1|\theta_k) \right)$$

The maximality equation for the logarithmic likelihood is

$$\frac{\delta}{\delta(\boldsymbol{c},\boldsymbol{\theta})} \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} c_k \; p(x_1|\theta_k) \right) = 0$$

The component equation for each θ_k is:

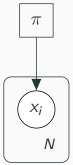
$$\sum_{i=1}^{n} \frac{c_k \frac{\delta}{\delta\theta_k} p(x_i|\theta_k)}{\sum_{k=1}^{K} c_k p(x_i|\theta_k)} = 0$$

Solving this problem is analytically infeasible (note that we cannot multiply out the denominator, because of the sum over i). Even numerical solution is often difficult.

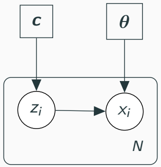
Latent Variables

Cluster assignments

- The mixture assumption implies that each x_i was generated from one component.
- For each x_i, we again use an **assignment variable** $z_i \in \{1, ..., K\}$ which encodes which cluster x_i was sampled from.



$\Rightarrow$



Latent Variables

Since we do not know which component each x_i was generated by, the values of the assignment variables are *unobserved*. Such variables whose values are not observed are called **latent variables** or **hidden variables**.

Estimation with latent variables

Latent variables as auxiliary information

If we knew the correct assignments z_i, we could:

- Estimate each component distribution $p(x|\theta_k)$ separately, using only the data assigned to cluster k.
- Estimate the cluster proportions c_k as $\widehat{c}_k = \frac{\#\text{ points in cluster } k}{n}$

EM algorithm: Idea

The EM algorithm estimates values of the latent variables to simplify the estimation problem. EM alternates between two steps:

1. Estimate assignments z_i given current estimates of the parameters c_k and θ_k ("E-step").
2. Estimate parameters c_i and θ_k given current estimates of the assignments ("M-step").

These two steps are iterated repeatedly.

Representation of Assignments

We re-write the assignments as vectors of length K:

$$\boldsymbol{x}_i \text{ in cluster } k \quad \text{as} \quad Z_i := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{th entry}$$

so $Z_{ik} = 1$ if x_i in cluster k, and $Z_{ik} = 0$ otherwise.

We collect the vectors into a matrix

$$\boldsymbol{Z} := \begin{bmatrix} Z_{11} & \dots & Z_{1K} \\ \vdots & & \vdots \\ Z_{n1} & \dots & Z_{nK} \end{bmatrix}$$

Note: Rows = observations, columns = clusters
Row sums = 1, column sums = cluster sizes.

E-Step

Hard vs. soft assignments

- The vectors Z_i are “hard assignments” with values in $\{0, 1\}$ (as in k-means)
- EM computes “soft assignments” r_{ik} with values in $[0, 1]$.[1]
- The vectors Z_i are the the latent variables in the EM algorithm. The r_{ik} are their current estimates

Assignment probabilities

The soft assignments are computed as

$$r_{ik} = \frac{c_k\, p(x_i \mid \theta_k)}{\sum_{l=1}^{K} c_l\, p(x_i \mid \theta_l)}$$

They can be interpreted as

$$r_{ik} := \mathbb{E}[Z_{ik} \mid x_i, \boldsymbol{c}, \boldsymbol{\theta}] = \Pr\{x_i \text{ generated by component } k \mid \boldsymbol{c}, \boldsymbol{\theta}\}$$

[1] Once the algorithm terminates, each point could still be assigned to a cluster by setting

$$z_i = \arg\max_k r_{ik}$$

M-Step (1)

Objective

The M-step re-estimates $\boldsymbol{c}$ and $\boldsymbol{\theta}$. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights r_{ik} instead of Z_{ik}

Cluster sizes

If we knew which points belong to which cluster, we could estimate the cluster proportions c_k by counting points:

$$\widehat{c}_k = \frac{\#\text{ points in cluster } k}{n} = \frac{\sum_{i=1}^{n} Z_{ik}}{n}$$

Since we do not know Z_{ik}, we substitute our current best guess, which is the expectations r_{ik}

$$\widehat{c}_k = \frac{\sum_{i=1}^{n} r_{ik}}{n}$$

M-Step (2)

Gaussian special case

The estimation of the component parameters θ depends on which distribution we choose for p. For now, we assume a Gaussian.

Component parameters

We use maximum likelihood to estimate $\theta = (\mu, \Sigma)$. We can write the MLE of μ_k as

$$\widehat{\mu}_k = \frac{1}{\#\text{ points in cluster } k} \sum_{i:x_i \text{ in } k} x_i = \frac{\sum_{i=1}^{n} Z_{ik} x_i}{\sum_{i=1}^{n} Z_{ik}}$$

By substituting current best guesses $(= r_{ik})$ again, we get

$$\widehat{\mu}_k = \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}}$$

For the covariance matrices:

$$\widehat{\Sigma}_k = \frac{\sum_{i=1}^{n} r_{ik}(x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T}{\sum_{i=1}^{n} r_{ik}}$$

Notation Summary

Assignment probabilities

$$\boldsymbol{r} := \begin{bmatrix} r_{11} & \dots & r_{1K} \\ \vdots & & \vdots \\ r_{n1} & \dots & r_{nK} \end{bmatrix} = \mathbb{E}\left[\begin{bmatrix} Z_{11} & \dots & Z_{1K} \\ \vdots & & \vdots \\ Z_{n1} & \dots & Z_{nK} \end{bmatrix}\right] = \begin{bmatrix} \mathbb{E}[Z_{11}] & \dots & \mathbb{E}[Z_{1K}] \\ \vdots & & \vdots \\ \mathbb{E}[Z_{n1}] & \dots & \mathbb{E}[Z_{nK}] \end{bmatrix}$$

Mixture parameters

$\boldsymbol{\tau} = (\boldsymbol{c}, \boldsymbol{\theta})$, $\boldsymbol{c} =$ cluster proportions $\boldsymbol{\theta} =$ component parameters

Iterations

$\boldsymbol{\theta}^{(j)}, \boldsymbol{r}^{(j)}, \dots =$ values in jth iteration

Summary: EM for Gaussian Mixture

Gaussian special case

$$\theta = (\mu, \Sigma)\,(\textit{mean}\,\&\,\textit{covariance}) \quad p(x \mid \theta) = p(x \mid \mu, \Sigma)\,(\text{Gaussian density})$$

Algorithm

The EM algorithm for a finite mixture of Gaussians looks like this

1. **Initialize:** Choose (e.g., random) values $c_k^{(0)}$ and $\theta_k^{(0)}$.
2. **E-Step:** Recompute the assignment weight matrix as

$$r_{ik}^{(j+1)} = \frac{c_k^{(j)} p(x_i \mid \theta_k^{(j)})}{\sum_{l=1}^{K} c_l^{(j)} p(x_i \mid \theta_l^{(j)})}$$

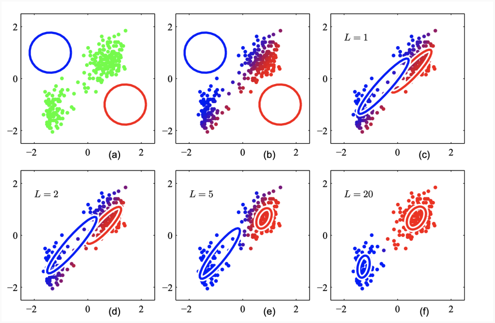
3. **M-Step:** Recompute the proportions c_k and parameters $\theta = (\mu, \Sigma)$ as

$$\mu_k^{(j+1)} = \frac{\sum_{i=1}^{n} r_{ik}^{(j+1)} x_i}{\sum_{i=1}^{n} r_{ik}^{(j+1)}} \quad \text{and} \quad \Sigma_k^{(j+1)} = \frac{\sum_{i=1}^{n} r_{ik}^{(j+1)} (x_i - \mu_k^{(j+1)})(x_i - \mu_k^{(j+1)})^T}{\sum_{i=1}^{n} r_{ik}^{(j+1)}}$$

The E-Step and M-Step are repeated alternatingly until convergence criterion (e.g. threshold) is satisfied.

EM: Illustration

EM for a mixture of two Gaussians



The algorithm fits both the mean and the covariance parameter.

Two group activities

1. Implementational
2. API Usage / Conceptual

GMMs as “Universal Approximators”

A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components.

Ian Goodfellow et al. 2016

More formally, if $\mathcal{P}(\mathbb{R}^d)$ is the set of probability Borel measures on $\mathbb{R}^d$ (with its Euclidean topology), then “Gaussian mixtures” (a.k.a. convex combinations of Gaussian measures) are dense in $\mathcal{P}(\mathbb{R}^d)$ for the weak* topology.

Gaussian Mixture Models for Non-Gaussian Distributions

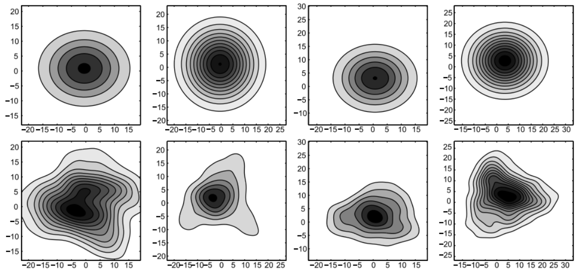


Fig. 3. *Contour plots of the best fit Gaussian (top) and kernel density estimate (bottom) for the top two principal components of the audio features associated with each of the four speakers present in the AMI_20041210-1052 meeting. Without capturing the non-Gaussianity of the speaker-specific emissions, the speakers are challenging to identify.*

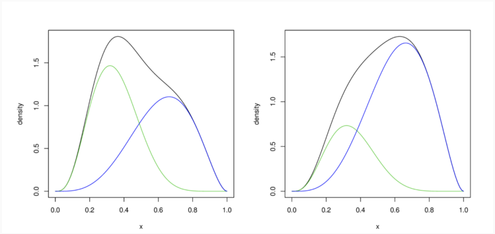
Fox, E. B., Sudderth, E. B., Jordan, M. I., & Willsky, A. S. (2011). A sticky HDP-HMM with application to speaker diarization. The Annals of Applied Statistics, 1020-1056.

Q: So why not use GMM's for everything?

Non-Gaussian Mixture Models

The mixture components do not need to be Gaussian in order for the model to be well-posed, and for EM to estimate the parameters.

Example: mixture of two betas

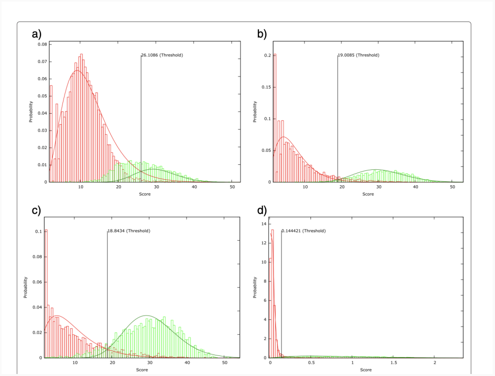


Shown are two beta mixture models, each with two components. The models have fixed components but differ in their mixture weights: $\boldsymbol{c} = (.25, .75)$ vs. $\boldsymbol{c} = (.75, .25)$.

Q: Under what conditions might one use a mixture of betas rather than a mixture of Gaussians?

Non-Gaussian Mixture Models

Example: mixture of two Gammas



Shown are four gamma mixture models.

Landesfeind, M., & Meinicke, P. (2014). Predicting the functional repertoire of an organism from unassembled RNA–seq data. BMC genomics, 15(1), 1003.

EM for Exponential Family Mixture Models

Algorithm

- **E-step:** Recompute the assignment matrix $r_{ik}^{(j)}$ as

$$\underbrace{r_{ik}^{(j+1)}}_{posterior} = \frac{c_k^{(j)} p(x_i \mid \theta_k^{(j)})}{\sum_{l=1}^{K} c_l^{(j)} p(x_i \mid \theta_l^{(j)})} \quad \propto \quad \underbrace{c_k^{(j)}}_{prior} \quad \underbrace{p(x_i \mid \theta_k^{(j)})}_{likelihood}$$

EM for Exponential Family Mixture Models

Algorithm

- **E-step:** Recompute the assignment matrix $r_{ik}^{(j)}$ as

$$\underbrace{r_{ik}^{(j+1)}}_{\textit{posterior}} = \frac{c_k^{(j)} p(x_i \mid \theta_k^{(j)})}{\sum_{l=1}^{K} c_l^{(j)} p(x_i \mid \theta_l^{(j)})} \quad \propto \quad \underbrace{c_k^{(j)}}_{\textit{prior}} \quad \underbrace{p(x_i \mid \theta_k^{(j)})}_{\textit{likelihood}}$$

- **M-step:** Recompute (c, θ) as

$$c_k^{(j+1)} = \frac{\sum_{i=1}^{n} r_{ik}^{(j+1)}}{n} \quad \text{and} \quad \theta_k^{(j+1)} = \frac{\sum_{i=1}^{n} r_{ik}^{(j+1)} s(x_i)}{\sum_{i=1}^{n} r_{ik}^{(j+1)}}$$

Weighted Maximum Likelihood

Recall that for any exponential family distribution, the MLE of $p(x \mid \theta)$ is of the form

$$\widehat{\theta}_{\mathrm{ML}} = \frac{1}{n} \sum_{i=1}^{n} s(x_i) = \frac{\sum_{i=1}^{n} 1\, s(x_i)}{\sum_{i=1}^{n} 1}$$

where s is the sufficient statistics function. Thus, the M-step computes the "weighted maximum likelihood estimate".

EM for Exponential Family Mixture Models

Algorithm

- **E-step:** Recompute the assignment matrix $r_{ik}^{(j)}$ as

$$\underset{posterior}{r_{ik}^{(j+1)}} = \frac{c_k^{(j)} p(x_i \mid \theta_k^{(j)})}{\sum_{l=1}^K c_l^{(j)} p(x_i \mid \theta_l^{(j)})} \quad \propto \quad \underset{prior}{c_k^{(j)}} \quad \underset{likelihood}{p(x_i \mid \theta_k^{(j)})}$$

- **M-step:** Recompute (c, θ) as

$$c_k^{(j+1)} = \frac{\sum_{i=1}^n r_{ik}^{(j+1)}}{n} \quad \text{and} \quad \theta_k^{(j+1)} = \frac{\sum_{i=1}^n r_{ik}^{(j+1)} s(x_i)}{\sum_{i=1}^n r_{ik}^{(j+1)}}$$

Weighted Maximum Likelihood

Recall that for any exponential family distribution, the MLE of $p(x \mid \theta)$ is of the form

$$\widehat{\theta}_{\mathrm{ML}} = \frac{1}{n} \sum_{i=1}^n s(x_i) = \frac{\sum_{i=1}^n 1\, s(x_i)}{\sum_{i=1}^n 1}$$

where s is the sufficient statistics function. Thus, the M-step computes the "weighted maximum likelihood estimate". (Note that the ML problem, and therefore the M-step, need not have closed form solution; e.g. click here for the Gamma distribution.)

Application: Identity verification with behavioral biometrics

- Recall our earlier discussion on intruder detection and behavioral biometrics.
- Gaussian (and other exponential family) mixture models have been used to construct the *personalized models* of typing behavior.
    - Often these models provide better models than non-mixtures, due to increased representational capacity.
- In this context, anomaly detection is often improved when using a *universal background model* (UBM) (itself a mixture model) that models the typing behavior of a larger population of individuals
- The level of anomalousness of typing behavior can be obtained via the log likelihood ratio of the universal background model to the personalized model.

Expectation Maximization (more generally)

Convergence Properties

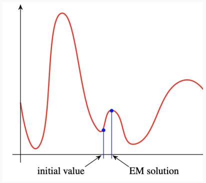
Marginal likelihood

- It can be shown that the marginal likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ always increases from each step to the next, unless $\boldsymbol{\theta}$ is already a stationary point.
- The theory guarantees only that the algorithm terminates at a stationary point. That point can be a saddle point rather than a maximum (very rare)

The real problem: Local maxima

- EM is effectively a gradient method.
- The maxima it finds are the **local maxima of the log-likelihood**
- There are no guarantees on the global quality of the solution: The global maximum may differ arbitrarily from the one we find.

$\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$



$\boldsymbol{\theta}$

Q: So what can we do about this?

EM in practice

Comparing solutions

- If $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are two different EM solutions, we can always compute the log-likelihoods
$$\sum_i \log p(x_i \mid \boldsymbol{\theta}) \quad \text{and} \quad \sum_i \log p(x_i \mid \boldsymbol{\theta}')$$
- The solution with the higher likelihood is better.
- This is a very convenient feature of EM: Different solutions are comparable.

Random restarts

In practice, the best way to use EM is often

- Restart EM repeatedly with randomly (or intelligently) chosen initial values.
- Compute the log-likelihoods of all solutions and compare them.
- Choose the solution achieving maximal log-likelihood

Q: What would be an intelligent way to initialize?

Expectation Maximization (more generally)

Latent variable models

Latent variable models

- A *parametric statistical model* may posit observed random variables ($\boldsymbol{x}$), parameters ($\boldsymbol{\theta}$), and latent random variables ($\boldsymbol{z}$).
- The distinguishing feature between latent variables $\boldsymbol{z}$ and parameters $\boldsymbol{\theta}$ is that the dimensionality of $\boldsymbol{z}$ increases with the size of the data set, whereas the dimensionality of $\boldsymbol{\theta}$ does not.[2]
- Note that some presentations refer to $\boldsymbol{z}$ as local hidden variables and $\boldsymbol{\theta}$ as global hidden variables.

[2] A more technical definition can be provided via conditional independence. For example, when there is one latent variable per observation, a latent variable satisfies $p(\boldsymbol{x}_n, \boldsymbol{z}_n \mid \boldsymbol{x}_{-n}, \boldsymbol{z}_{-n}, \theta) = p(\boldsymbol{x}_n, \boldsymbol{z}_n \mid \theta)$, where the $-n$ subscript refers to the set of variables besides the *n*th. In other words, the *n*th observation and *n*th latent variable is independent of all other observations and latent variables, given the model parameters.

Frequentist estimation for latent variable models

- Latent variable models provide a **complete data likelihood** $p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$, where $\boldsymbol{z}$ is unobserved. The model often factorizes as

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta})$$

- For frequentist latent variable models, the inferential goal is to compute $\boldsymbol{\theta}_{\mathrm{ML}}$, the maximum likelihood value of the parameter.
- Since $\boldsymbol{z}$ is not observed, so one seeks to find

$$\boldsymbol{\theta}_{\mathrm{ML}} := \mathrm{argmax}_{\boldsymbol{\theta}}\ p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \mathrm{argmax}_{\boldsymbol{\theta}} \int p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\ d\,\boldsymbol{z} \tag{3.1}$$

- In particular, one requires access to the *marginal* likelihood

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \int p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\ d\,\boldsymbol{z} \tag{3.2}$$

Expectation Maximization for Latent Variable Models

The **expectation maximization algorithm** estimates $\boldsymbol{\theta}$ by attempting to maximize the marginal likelihood.

The expectation maximization algorithm is

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t)})}\left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\right] \tag{3.3}$$

Discuss: What does and doesn't make sense about this algorithm?

Expectation Maximization for Latent Variable Models

The **expectation maximization algorithm** estimates $\boldsymbol{\theta}$ by attempting to maximize the marginal likelihood.

The expectation maximization algorithm is

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t)})}\left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) \right] \tag{3.3}$$

Discuss: What does and doesn't make sense about this algorithm?
Rk: There are many derivations. One will fall out naturally when we do the section on variational inference.

Expectation Maximization for Latent Variable Models

The **expectation maximization algorithm** estimates $\boldsymbol{\theta}$ by attempting to maximize the marginal likelihood.

The expectation maximization algorithm is

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t)})} \left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) \right] \tag{3.3}$$

Discuss: What does and doesn't make sense about this algorithm?
Rk: There are many derivations. One will fall out naturally when we do the section on variational inference.
Discuss: When will this algorithm will not be useable?

EM for Exponential Family Latent Variable Models

Let us assume that $(\boldsymbol{x}, \boldsymbol{z}) = ((x_1, z_1), \ldots, (x_n, z_n))$ are n independent samples from the same exponential family, where $\boldsymbol{x}$ is observed data and $\boldsymbol{z}$ is unobserved data. Moreover, let us assume that the complete data likelihood is in the exponential family.[3]

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \theta) = \prod_{i=1}^{n} h(x_i, z_i) \exp\left\{\theta^T \sum_{i=1}^{n} s(x_i, z_i) - n\, a(\eta(\theta))\right\} \tag{3.4}$$

[3] And that we are using the *mean parameterization*; i.e. $\theta = \mathbb{E}[s(x_1, z_1)]$.

EM for Exponential Family Latent Variable Models

Let us assume that $(\boldsymbol{x}, \boldsymbol{z}) = ((x_1, z_1), \ldots, (x_n, z_n))$ are n independent samples from the same exponential family, where $\boldsymbol{x}$ is observed data and $\boldsymbol{z}$ is unobserved data. Moreover, let us assume that the complete data likelihood is in the exponential family.[3]

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \theta) = \prod_{i=1}^{n} h(x_i, z_i) \exp\left\{\theta^T \sum_{i=1}^{n} s(x_i, z_i) - n\, a(\eta(\theta))\right\} \tag{3.4}$$

The EM algorithm (at iterate j) attempts to find $\boldsymbol{\theta}$ to maximize

$$f(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(j)})}\left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\right]$$

[3] And that we are using the *mean parameterization*; i.e. $\theta = \mathbb{E}[s(x_1, z_1)]$.

EM for Exponential Family Latent Variable Models

Let us assume that $(\boldsymbol{x}, \boldsymbol{z}) = ((x_1, z_1), \ldots, (x_n, z_n))$ are n independent samples from the same exponential family, where $\boldsymbol{x}$ is observed data and $\boldsymbol{z}$ is unobserved data. Moreover, let us assume that the complete data likelihood is in the exponential family.[3]

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \theta) = \prod_{i=1}^{n} h(x_i, z_i) \exp\left\{ \theta^T \sum_{i=1}^{n} s(x_i, z_i) - n\, a(\eta(\theta)) \right\} \tag{3.4}$$

The EM algorithm (at iterate j) attempts to find $\boldsymbol{\theta}$ to maximize

$$f(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(j)})} \left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) \right]$$

The solution is to select $\theta^{(j+1)}$ such that

$$\theta^{(j+1)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(j)})} s(x_i, z_i)$$

[3] And that we are using the *mean parameterization*; i.e. $\theta = \mathbb{E}[s(x_1, z_1)]$.

EM for Exponential Family Latent Variable Models

Let us assume that $(\boldsymbol{x}, \boldsymbol{z}) = ((x_1, z_1), \ldots, (x_n, z_n))$ are n independent samples from the same exponential family, where $\boldsymbol{x}$ is observed data and $\boldsymbol{z}$ is unobserved data. Moreover, let us assume that the complete data likelihood is in the exponential family.[3]

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \theta) = \prod_{i=1}^{n} h(x_i, z_i) \exp\left\{\theta^T \sum_{i=1}^{n} s(x_i, z_i) - n\, a(\eta(\theta))\right\} \tag{3.4}$$

The EM algorithm (at iterate j) attempts to find $\boldsymbol{\theta}$ to maximize

$$f(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(j)})}\left[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\right]$$

The solution is to select $\theta^{(j+1)}$ such that

$$\theta^{(j+1)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(j)})} s(x_i, z_i)$$

This is why an EM iteration is often described and implemented as iteratively computing **maximum likelihood with the expected sufficient statistics.**

[3] And that we are using the *mean parameterization*; i.e. $\theta = \mathbb{E}[s(x_1, z_1)]$.

Expectation Maximization (more generally)

Hidden Markov Models

Demo

- You have seen HMM's from Karin's presentation.
- HMM's are also in the exponential family (so long as the emissions distributions are)!
- Thus, we can easily work with HMM's with non-gaussian emissions.
- The M-step simply updates the emissions parameters via the expected sufficient statistics.
- The E-step (and the other M-step updates) remain the same as before.)

Hidden Markov Model

A hidden Markov model (HMM) is a tool for representing probability distributions over sequences of observations.

The HMM assumes that

- The observation at time t, y_t, was generated by some process whose state x_t is hidden from the observer.
- The sequence of states satisfies the *Markov property*: conditional on the current state x_t, past and future hidden states are independent.
- There is an additional Markov property on outputs: conditional on the current state x_t, the output y_t is independent of all other hidden states and outputs.

Notation

- $y_{1:T} = (y_1, ..., y_T)$ observed sequence
- $x_{1:T} = (x_1, ..., x_T)$: hidden state sequence ($x_t \in \{1, ..., K\}$)
- $\pi = \{\pi_k\}, \pi_k = P(x_1 = k)$: initial state distribution
- $A = \{A_{kk'}\}, A_{kk'} = P(x_t = k' \mid x_{t-1} = k)$: state transition probability matrix
- $\phi = (\phi_k)_{k=1}^{K}$ a set of parameters, each governing an output distribution (also called emissions distribution) associated to each hidden state; that is, $P(y_t \mid x_t = k) = P(y_t \mid \phi_k)$.
- $\theta = (\pi, A, \phi)$: model parameters

HMM: Complete Data Likelihood (CDL)

The complete data likelihood for the HMM is given by

$$
\begin{aligned}
p(x_{1:T}, y_{1:T} \mid \theta) &= p(x_1 \mid \theta) p(y_1 \mid x_1, \theta) \prod_{t=2}^{T} p(x_t \mid x_{t-1}, \theta) p(y_t \mid x_t, \theta) \\
&= p(x_1 \mid \pi) p(y_1 \mid x_1, \phi) \prod_{t=2}^{T} p(x_t \mid x_{t-1}, A) p(y_t \mid x_t, \phi) \\
&= \pi_{x_1} \prod_{t=2}^{T} A_{x_{t-1}, x_t} \prod_{t=1}^{T} p(y_1 \mid \phi_{x_t}) \qquad (3.5)
\end{aligned}
$$

HMM: Complete Data Likelihood is an Exponential Family

We continue

$$
\begin{aligned}
p(x_{1:T}, y_{1:T} \mid \theta) &= \exp\left\{ \log p(x_1 \mid \pi) + \sum_{t=2}^{T} \log p(x_t \mid x_{t-1}, A) + \sum_{t=1}^{T} \log p(y_t \mid x_t, \phi) \right\} \\
&= \exp\left\{ \log \pi_{x_1} + \sum_{t=2}^{T} \log A_{x_{t-1}, x_t} + \sum_{t=1}^{T} \log p(y_1 \mid \phi_{x_t}) \right\} \qquad (3.6) \\
&= \exp\left\{ \sum_{k=1}^{K} x_1^k \log \pi_k + \sum_{t=2}^{T} \sum_{k,k'=1}^{K} x_{t-1}^k x_t^{k'} \log A_{kk'} \right. \\
&\qquad \left. + \sum_{t=1}^{T} \sum_{k=1}^{K} x_t^k \log p(y_t \mid x_t, \phi_k) \right\} \qquad (3.7)
\end{aligned}
$$

where we have defined

$$
x_t^k = \begin{cases} 1, & \text{if the latent state at time } t \text{ is } k \\ 0, & \text{otherwise} \end{cases}
$$

Thus, the complete data likelihood (although not the marginal likelihood $p(x_{1:T} \mid \theta)$) in the exponential family, so long as the emissions distributions are. The sufficient statistics for $\log \pi_k$ are x_1^k, and the sufficient statistics for $\log A_{kk'}$ are $\sum_{t=2}^{T} x_{t-1}^k x_t^{k'}$.

Expectation Maximization (more generally)

Application of General EM

Introduction

- Commercial anti-virus software traditionally memorizes specific byte sequences (known as **signatures**) in the file contents of previously encountered malware.
- They could use these signatures to attempt to detect malware in the future.
- Malware authors can evade signature-based detection in many ways; for instance, by
  - tampering with existing malware signatures (sometimes flipping a single bit)
  - using obfuscation techniques (e.g. encryption or compression) to hide snippets of malicious code
  - writing metamorphic malware
- As a result, classical AV detections have **low false positive** rates *but also* **low true positive** rates.

Better Malware Ground Truth

Kantchelian et al. (2015) examine the problem of aggregating the results of multiple anti-virus (AV) vendors' detectors into a single authoritative ground-truth label for every binary

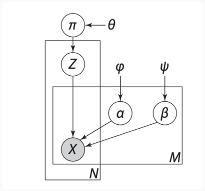
Kantchelian, A., Tschantz, M. C., Afroz, S., Miller, B., Shankar, V., Bachwani, R., ... & Tygar, J. D. (2015, October). Better malware ground truth: Techniques for weighting anti-virus vendor labels. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (pp. 45-56).

Model

Random variables

$X_{ij} \in \{0,1\}$	AV label (i = instance, j =vendor)
$Z_i \in \{0,1\}$	ground truth label
α, β	vendor tp, fp rates
π	malware prevalence

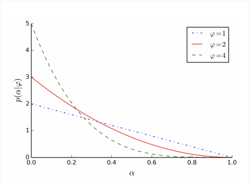
In our terminology, $\boldsymbol{Z}$ are latent variables and α, β are parameters.



Model

$$X_{ij} \mid Z_i, \alpha_j, \beta_j = \begin{cases} \alpha_j & \text{if } Z_i = 1 \text{ and } X_{ij} = 1 \text{ (TP)} \\ 1 - \alpha_j & \text{if } Z_i = 1 \text{ and } X_{ij} = 0 \text{ (FN)} \\ \beta_j & \text{if } Z_i = 0 \text{ and } X_{ij} = 1 \text{ (FP)} \\ 1 - \beta_j & \text{if } Z_i = 0 \text{ and } X_{ij} = 0 \text{ (TN)} \end{cases}$$

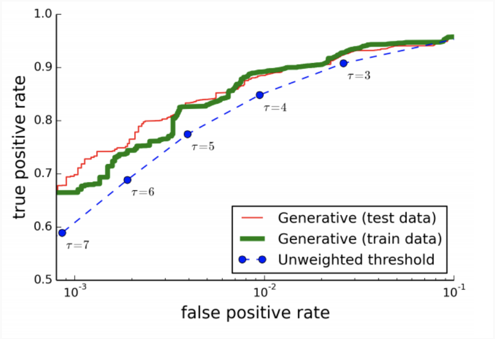
$$Z_i \mid \pi \sim \text{Bernouli}(\pi)$$
$$\pi \sim \text{Beta (symmetric)}$$
$$\alpha \mid \psi \sim \text{Beta (assymmetric)}$$
$$\beta \mid \phi \sim \text{Beta (assymmetric)}$$



Asymmetric (right skewed) priors were used for α, β since both are expected to be low based on prior domain knowledge.

Estimation and Results

- The model was fit using EM.
- The model far outperforms a common baseline in estimating ground truth.



- Moreover, it accomplished this *despite being fully unsupervised*! (No ground truth was available during training.)