THE EXPECTATION-MAXIMIZATION METHOD

David Kirkby, UC Irvine

LSSTC Data Science Fellows Program
Caltech, January 2017

INTRODUCTION

- The expectation maximization (EM) algorithm is one of the "miracles" of numerical analysis:
  - remarkably fast and stable, with good convergence.
  - try to adapt your problem to take advantage of it.
  - (another "miracle" is the fast Fourier transform).
- The main applications of EM are:
  - unsupervised learning: discover clusters in your data.
  - non-linear regression: maximum-likelihood fit of data to a "mixture model" (usually Gaussian).

OUTLINE

- Gaussian mixture models
- Latent variables
- Expectation maximization
- Practical advice
- Advanced applications

GAUSSIAN MIXTURE MODELS

- Model N-dimensional data as a sum of K independent Gaussians:

$$P(\boldsymbol{x}|\overbrace{\alpha_1, \alpha_2, \ldots, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, C_1, C_2, \ldots}^{\textit{model parameters}}) = \sum_{k=1}^{K} \alpha_k \, G(\boldsymbol{x}|\boldsymbol{\mu}_k, C_k)$$

N-dim. data point

amount of k-th Gaussian

N-dim. means

NxN-dim. covariances

Gaussian density

coefficients are normalized: $\sum_{k=1}^{K} \alpha_k = 1$

GAUSSIAN MIXTURE MODELS

- The multivariate Gaussian (a.k.a. "normal") distribution is:

$$G(\boldsymbol{x}|\boldsymbol{\mu}, C) = (2\pi)^{-N/2}\ |C|^{-1/2}\ \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t C^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

- This compact formula glosses over a lot of details!
- The covariance matrix is built from the RMS values (a.k.a. "sigmas") and correlation coefficients:

$$C_{ii} = \sigma_i^2 \quad , \quad C_{ij} = \rho_{ij}\sigma_i\sigma_j \qquad\qquad -1 < \rho_{ij} = \rho_{ji} < +1$$

EXERCISE 1: EVALUATE GAUSSIAN PROBABILITY DENSITY

- Write a numpy expression to calculate the 2D Gaussian probability density:

```
def gauss2d(x1, x2, mu1, mu2, sigma1, sigma2, rho12):
  # your code here
  …
  # hint: use np.dot and lookup np.linalg
```

- Test your expression using:

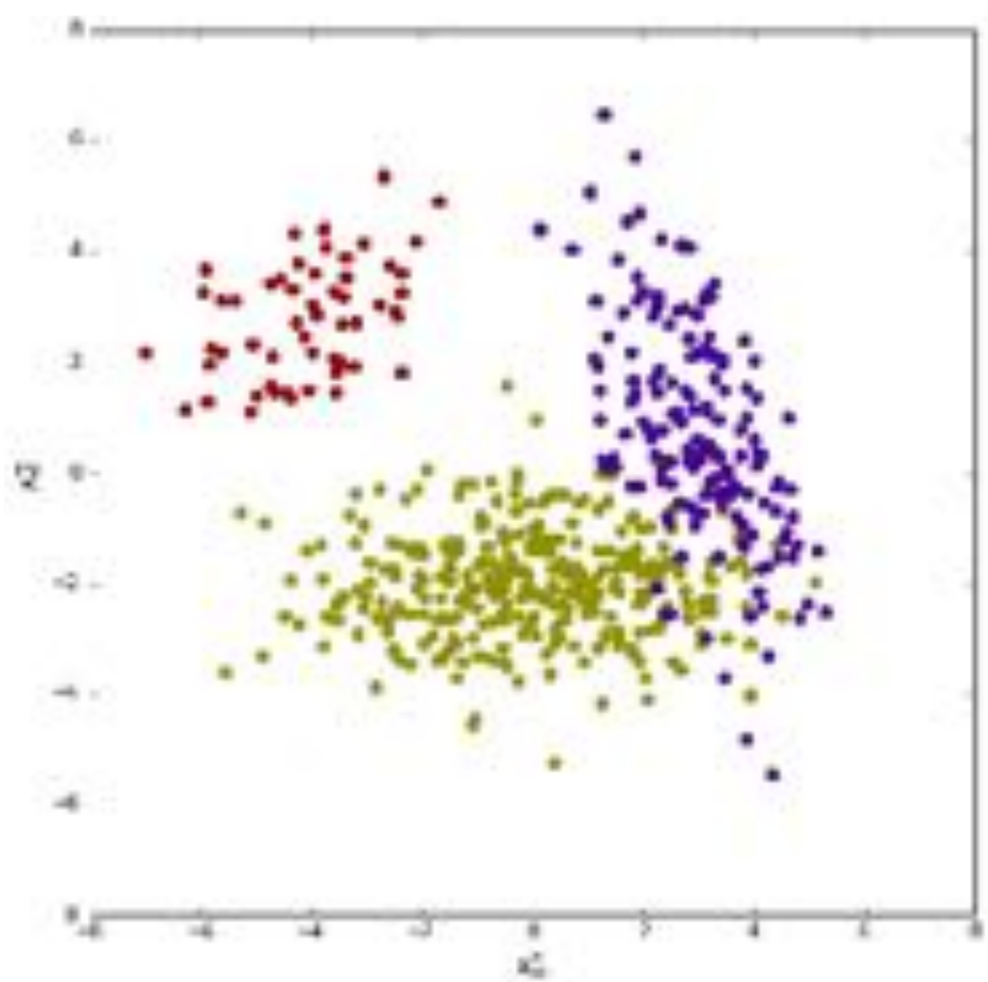
```
print gauss2d(1, 2, -1, 1, 2, 0.5, -0.5)
0.00172815191818
```

EXERCISE 2: ESTIMATE PARAMETERS OF A GAUSSIAN MIXTURE

➤ Read the tagged data and estimate the model parameters:

$$P(\boldsymbol{x}|\overbrace{\alpha_1, \alpha_2, \ldots, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, C_1, C_2, \ldots}^{\textit{model parameters}}) = \sum_{k=1}^{K} \alpha_k \, G(\boldsymbol{x}|\boldsymbol{\mu}_k, C_k)$$

```
!head tagged-gmm.dat
1  -0.19261   1.44914
2   2.90957   3.90351
2   1.63016   3.90409
2   2.19753   2.19192
0  -5.19830   4.19952
1  -0.66910   2.61219
2   3.62593   0.79678
2   1.95651   1.17622
1  -0.34609   1.10940
2   4.13730   3.84287
```



EXERCISE 2: ESTIMATE PARAMETERS OF A GAUSSIAN MIXTURE

- Hint: how many parameters are there?
- Hint: use the following skeleton to read the data file:

```
def estimate_tagged():
  data = np.loadtxt('em-tagged.dat')
  tags = data[:, 0].astype(int)
  ...
```

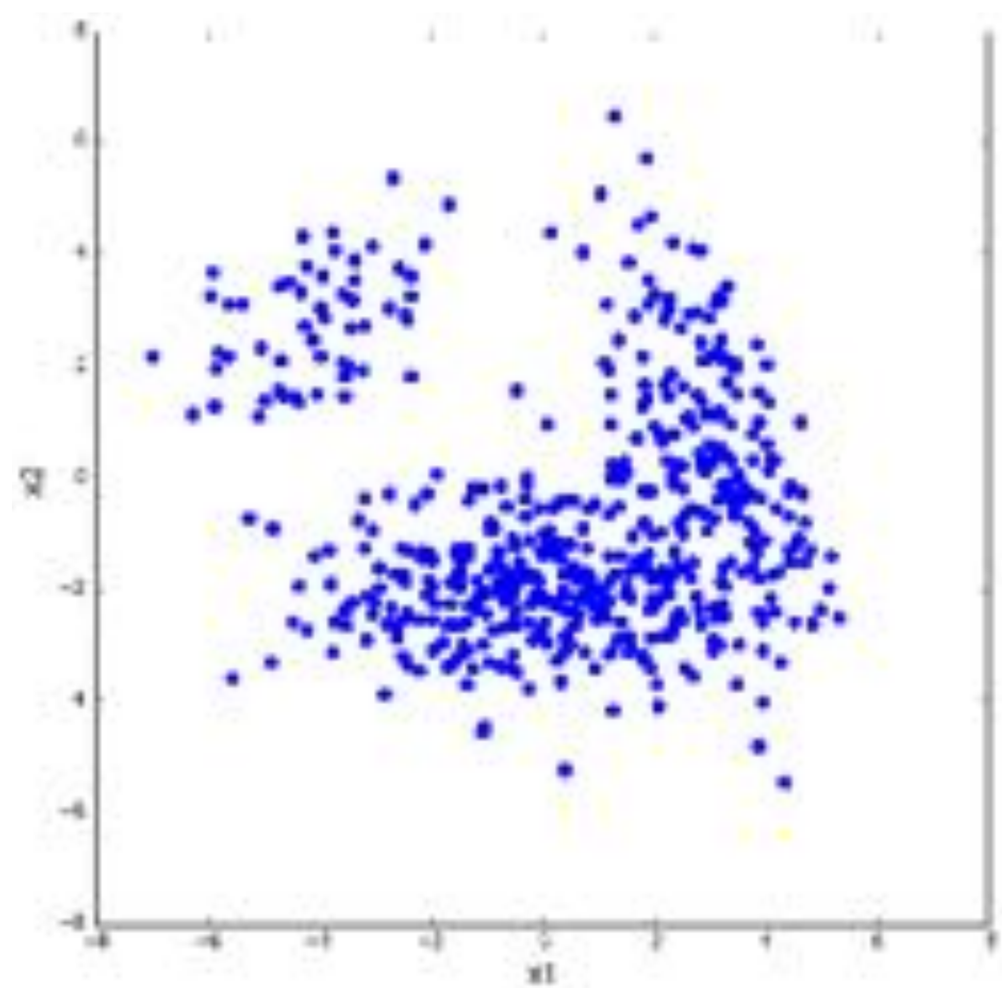
- Hint: lookup `np.mean` and `np.cov`.
- Hint: check your answers:

```
TAG  ALPHA    MU1     MU2 SIGMA1 SIGMA2  RHO12
  0  0.102 -4.079  2.788  1.205  1.035  0.429
  1  0.598 -0.031 -1.986  1.970  0.950 -0.006
  2  0.300  2.967  0.673  1.013  2.003 -0.560
```

UNTAGGED MIXTURE MODELS

- You just solved the “tagged mixture model” problem.
- It was relatively easy because each observation was tagged with the Gaussian it belongs to.
- What if we remove the tags from the data?

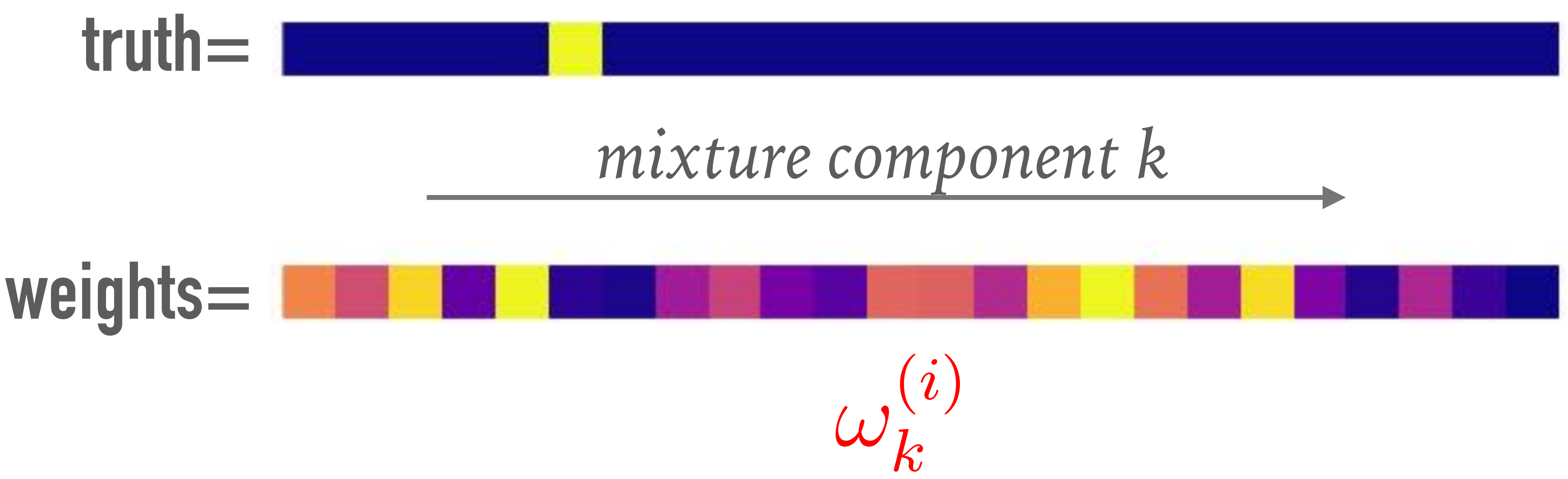
~~1~~	-0.19261	1.44914
~~2~~	2.90957	3.90351
~~2~~	1.63016	3.90409
~~1~~	2.19753	2.19192
~~0~~	-5.19830	4.19952
~~1~~	-0.66910	2.61219
~~2~~	3.62593	0.79678
~~2~~	1.95651	1.17622
~~1~~	-0.34609	1.10940
~~2~~	4.13730	3.84287



LATENT VARIABLES

- The missing tags make the problem easy, so we re-introduce them as "latent" (unobserved) variables.
- Replace the certainty of a tag with a set of weights when estimating the parameters:

for data sample i:



LATENT VARIABLES

- The missing tags make the problem easy, so we re-introduce them as "latent" (unobserved) variables.
- Replace the certainty of a tag with a set of weights when estimating the parameters for each component k:

$$\alpha_k = \frac{1}{K}\sum_{i=1}^{M} \omega_k^{(i)}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)}\,\boldsymbol{x}_i}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

$$C_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)}\,(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

EXERCISE 3: LATENT VARIABLES

- What do the constants K and M represent?
- The weights are normalized: write down an equation for this.
- Sketch $(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^t$ using row and column vectors.

$$\alpha_k = \frac{1}{K}\sum_{i=1}^{M} \omega_k^{(i)}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)} \boldsymbol{x}_i}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

$$C_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

ESTIMATING THE LATENT VARIABLES

- If we knew the weights, we would be done.
- What weights should we use?
- Suppose we already knew the true means and covariances, then we could calculate weights using Bayes' rule:

$$
\begin{aligned}
\omega_k^{(i)} &= P(t^{(i)} = k | \boldsymbol{x}_i; \Theta) \\
&= \frac{P(\boldsymbol{x}_i | t^{(i)} = k; \Theta)\, P(t^{(i)} = k, \Theta)}{P(\boldsymbol{x}_i)} \\
&= \frac{G(\boldsymbol{x}_i | \boldsymbol{\mu}_i, C_i)\, \alpha_i}{P(\boldsymbol{x}_i)} \\
&= \frac{\alpha_i\, G(\boldsymbol{x}_i | \boldsymbol{\mu}_i, C_i)}{\sum_{k=1}^{K} \alpha_k\, G(\boldsymbol{x}_i | \boldsymbol{\mu}_k, C_k)}
\end{aligned}
$$

$t^{(i)} = k$

"data sample i belongs to mixture k"

$$
\Theta = \{\alpha_1, \alpha_2, \ldots, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, C_1, C_2, \ldots\}
$$

PUTTING THE PIECES TOGETHER

- Start from an initial guess at the Gaussian parameters.
- Repeat until converged:
  - E step: estimate weights using assumed Gaussian params.
  - M step: estimate Gaussian params using estimated weights.

E step

$$\boxed{\omega_k^{(i)} = \frac{\alpha_i\, G(\boldsymbol{x}_i|\boldsymbol{\mu}_i, C_i)}{\sum_{k=1}^{K} \alpha_k\, G(\boldsymbol{x}_i|\boldsymbol{\mu}_k, C_k)}}$$

M step

$$\alpha_k = \frac{1}{K}\sum_{i=1}^{M} \omega_k^{(i)}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)}\, \boldsymbol{x}_i}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

$$C_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)}\, (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

PUTTING THE PIECES TOGETHER

- Start from an initial guess at the Gaussian parameters.
- Repeat until converged:
  - E step: estimate weights using assumed Gaussian params.
  - M step: estimate Gaussian params using estimated weights.

- Initial guess does not need to be close to the final answer.
  - But must start with distinguishable Gaussians.
  - K-means algorithm is often used to obtain starting points.

PUTTING THE PIECES TOGETHER

- Start from an initial guess at the Gaussian parameters.
- Repeat until converged:
  - E step: estimate weights using assumed Gaussian params.
  - M step: estimate Gaussian params using estimated weights.

- Log-likelihood of the mixture model is guaranteed to increase with each step:

$$\log \mathcal{L}(\Theta) = \sum_{i=1}^{M} \log \sum_{k=1}^{K} \alpha_k G(\boldsymbol{x}_i | \mu_k, C_k)$$

- Continue until fractional change is below some tolerance.

EXERCISE 4: GMM IN PRACTICE

- The scikit-learn mixture package implements a robust and convenient GMM solver:

```
import sklearn.mixture

model = sklearn.mixture.GaussianMixture(n_gauss)
model.fit(data)

print model.weights_, model.means_, model.covariances_
```

- Use scikit-learn to fit the previous dataset without the tags:

```
def estimate_untagged():
  data = np.loadtxt('em-tagged.dat')[:, 1:]
  ...
```

EXERCISE 4: GMM IN PRACTICE

- Do the tagged and untagged results agree?
- Which has smaller errors?

Truth:

TAG	ALPHA	MU1	MU2	SIGMA1	SIGMA2	RHO12
0	0.1	-4.	3.	1.	1.	0.5
1	0.6	0.	-2.	2.	1.	0.0
2	0.3	3.	1.	1.	2.	-0.5

Tagged:

TAG	ALPHA	MU1	MU2	SIGMA1	SIGMA2	RHO12
0	0.102	-4.079	2.788	1.205	1.035	0.429
1	0.598	-0.031	-1.986	1.970	0.950	-0.006
2	0.300	2.967	0.673	1.013	2.003	-0.560

Untagged:

TAG	ALPHA	MU1	MU2	SIGMA1	SIGMA2	RHO12
0	0.102	-4.074	2.783	1.205	1.035	0.427
1	0.327	2.929	0.433	1.016	2.083	-0.481
2	0.571	-0.151	-1.977	1.918	0.950	0.036

HOW MANY GAUSSIANS?

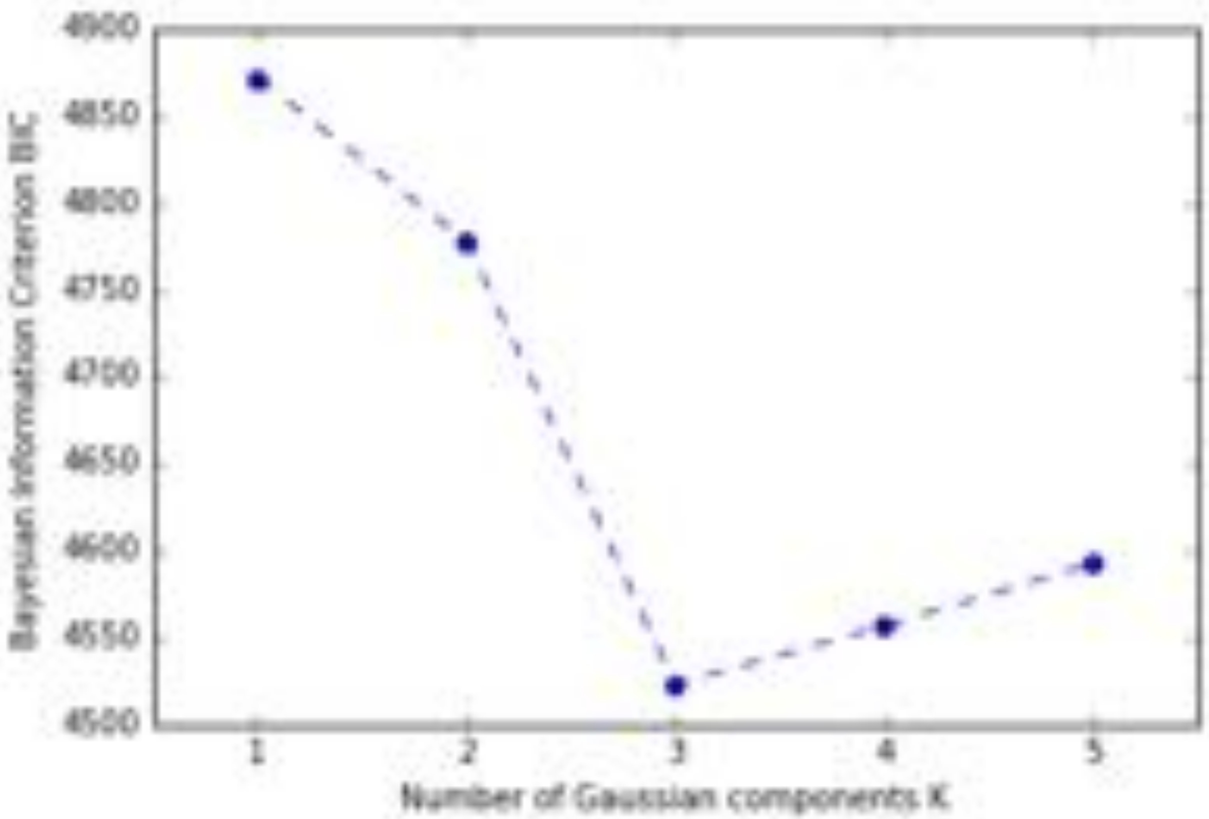
- The number of Gaussians K is a "hyper-parameter" of the EM algorithm:
  - Must either be set a-priori, or estimated from the data.
  - A pragmatic solution is to use the "Bayesian information criterion" (BIC) to pick the "best" value of K.
  - A more rigorous Bayesian approach is to consider a range of values K_1, K_2, … and average the posterior over the resulting models M_1, M_2, …
  - sklearn also provides BayesianGaussianMixture, but it has more hyper-parameters and isn't an obvious improvement over using BIC.

EXERCISE 5: RE-FIT WITH DIFFERENT NUMBERS OF GAUSSIANS

- How many parameters does a GMM have in terms of K, M?
- Repeat the previous (M=2) fit with K = 1, 2, 3, 4, 5.
- Plot the BIC of each fit vs. K.

```
model = sklearn.mixture.GaussianMixture(n_gauss)
model.fit(data)
bic = model.bic(data)
```

BAYESIAN INFORMATION CRITERION



goodness of fit *naturalness of fit*

$$\mathrm{BIC} = -2\log\mathcal{L}(\Theta) + N_p \log M$$

[details]

$$N_p = \underbrace{(K-1)}_{\alpha} + \underbrace{KM}_{\mu} + \underbrace{K\frac{M(M+1)}{2}}_{C}$$

EXERCISE 6: FIT DATA WITH UNKNOWN NUMBER OF GAUSSIANS

- Find a good GMM fit to the mystery data:

```
data = np.loadtxt('em-mystery.dat')
```

- Is the model with the minimum BIC really the most natural?

GMM USE CASES

- Discover clustering in data.
- Create empirical sampler to simulate data.

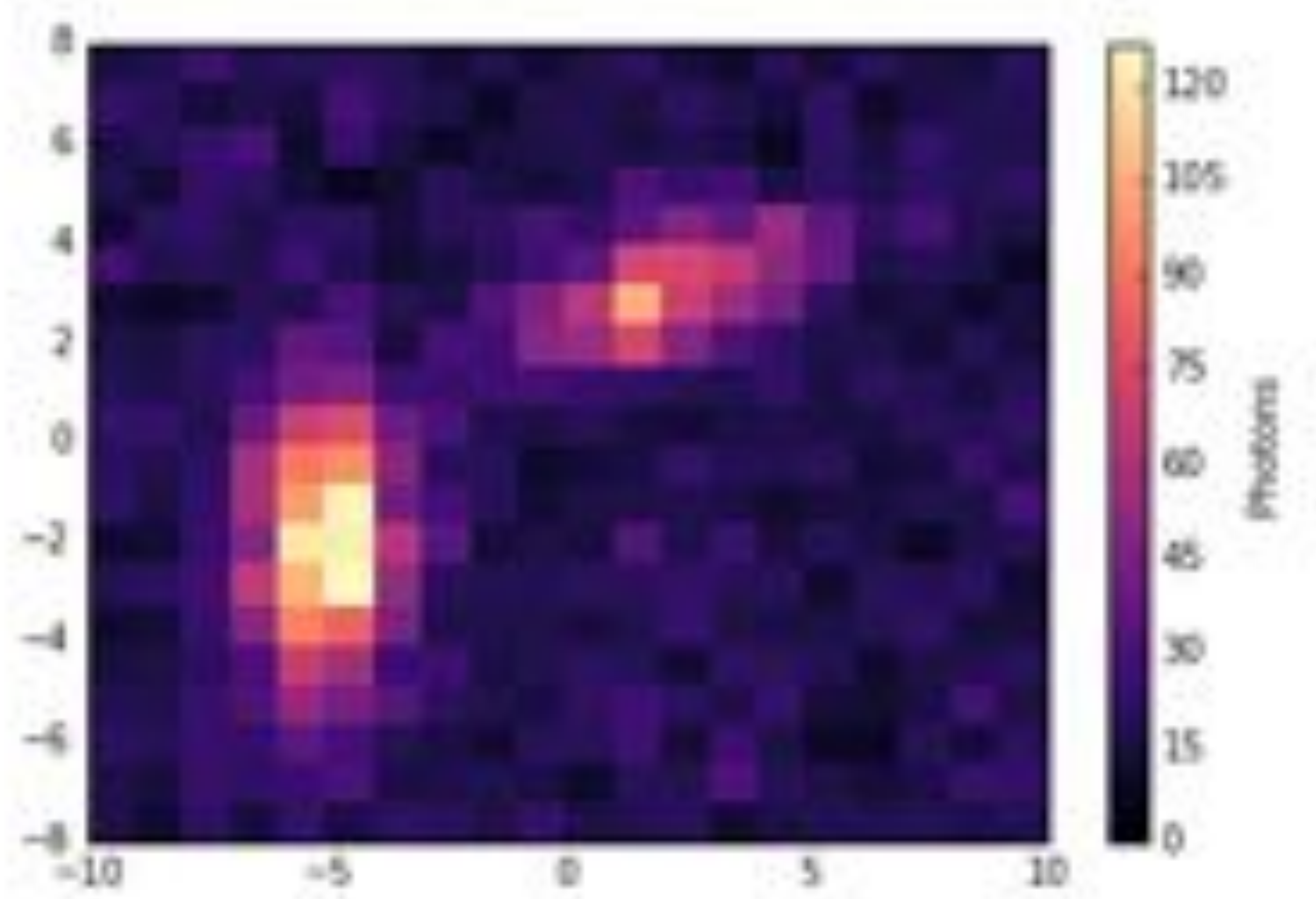
```
model = sklearn.mixture.GaussianMixture(n_gauss)
model.fit(data)
simulated = model.sample(n_sim)
```

- Maximum likelihood fit to mixture of Gaussians.
- Pick good starting point for Markov chain Monte Carlo sampling of Bayesian posterior.

DISCUSS: FIT GALAXIES WITH MIXTURE MODELS?

- How can mixture models help to detect and measure faint galaxies in images?

```
photons_per_pixel = load_image(…)
data = …
model = sklearn.mixture.GaussianMixture(n_gauss)
model.fit(data)
```

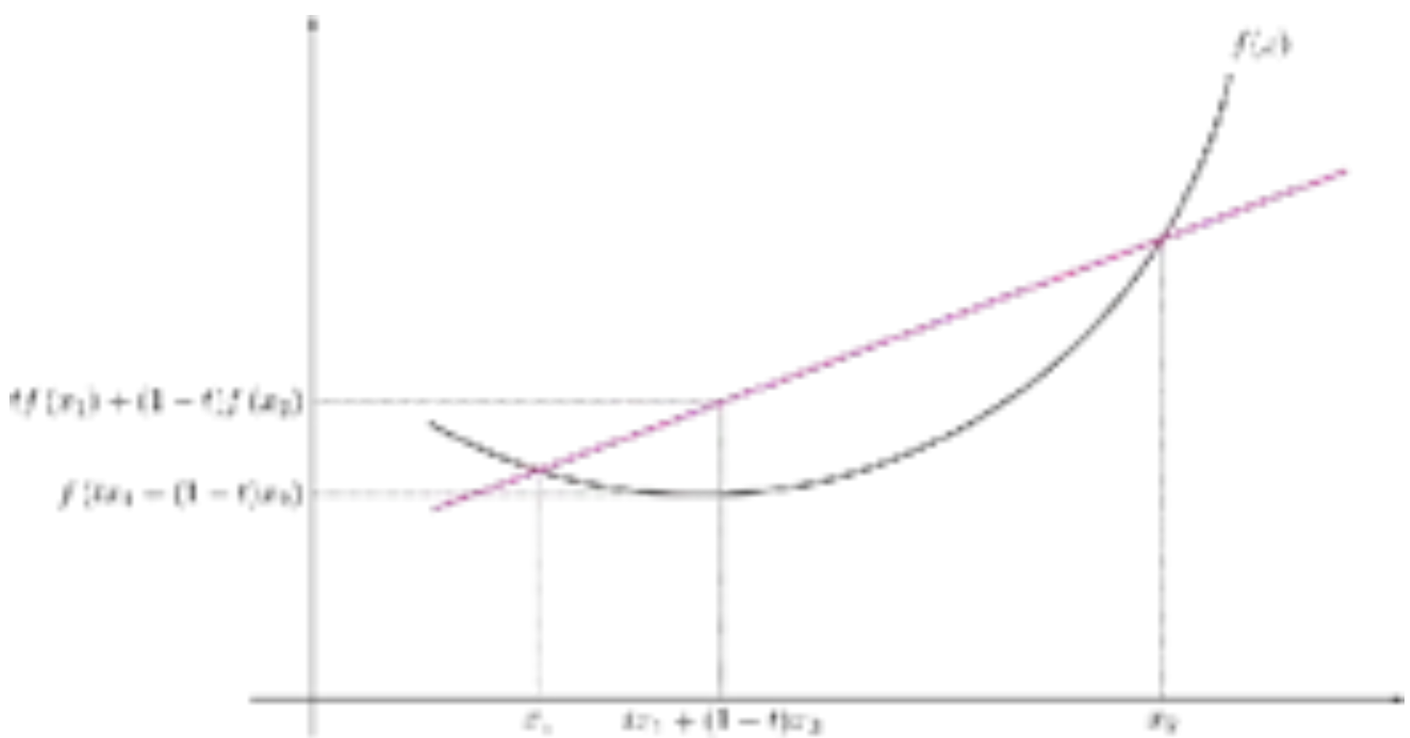


PRACTICAL ADVICE

- EM always converges to a local maximum, but with no guarantee that it is the global maximum.
  - Repeat with different random initial parameters.
- GMM does not perform well under certain conditions:
  - ~flat non-zero background
  - any component truncated at a boundary
- Consider your choice of K carefully.
  - Sometimes you just want a fit that is “good enough”.
  - Sometimes you need a full-blown Bayesian approach.

EXPECTATION MAXIMIZATION: THE BIGGER PICTURE

- The EM algorithm can be generalized to iteratively find the maximum likelihood fit to any mixture model.
- Each step is guaranteed to improve the likelihood.
- Proof follows from “Jensen’s inequality” for convex functions *f(x)*:



$$\boxed{f\left(\langle X\rangle\right) \leq \langle f(X)\rangle}$$

EXPECTATION MAXIMIZATION: THE BIGGER PICTURE

- The EM algorithm can solve a bigger class of problems:

$$P(x|\Theta) = \sum_{k=1}^{K} P(x, k|\Theta)$$

- The latent variable *k* is arbitrary (and could be continuous).
- The model and parameters are arbitrary.
- The E step is essentially unchanged.
- The M step is more expensive, in general, and requires iterative non-linear maximization.

EXPECTATION MAXIMIZATION: THE BIGGER PICTURE

➤ The Gaussian mixture is a special case with a closed-form solution for the M step.

E step

$$\omega_k^{(i)} = P(t^{(i)} = k | \boldsymbol{x}_i, \theta)$$

M step

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^{(i)} \log \frac{P(\boldsymbol{x}_i, t^{(i)} = k | \Theta)}{\omega_k^{(i)}}$$

Bayes' rule

lengthy calculation!

$$\omega_k^{(i)} = \frac{\alpha_i \, G(\boldsymbol{x}_i | \boldsymbol{\mu}_i, C_i)}{\sum_{k=1}^{K} \alpha_k \, G(\boldsymbol{x}_i | \boldsymbol{\mu}_k, C_k)}$$

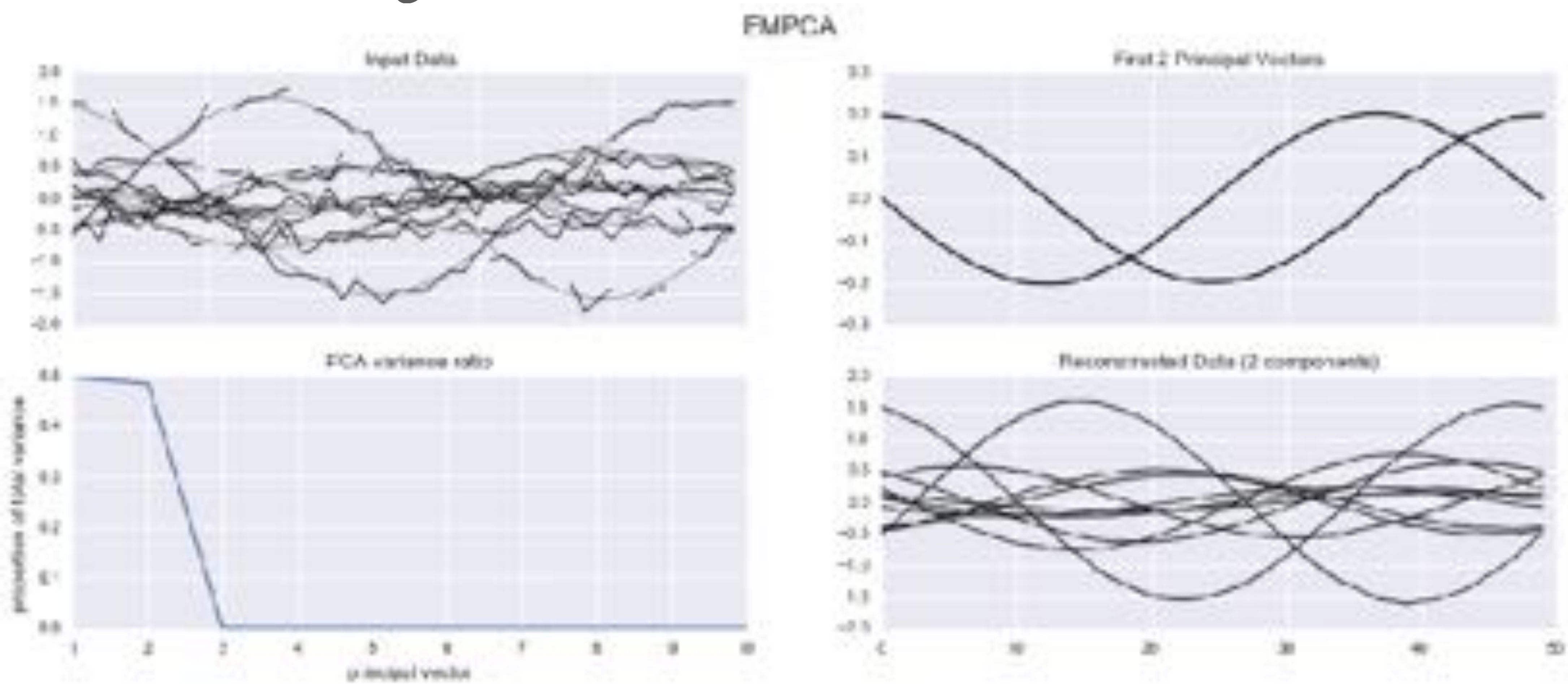
$$\alpha_k = \frac{1}{K} \sum_{i=1}^{M} \omega_k^{(i)}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)} \boldsymbol{x}_i}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

$$C_k = \frac{\sum_{i=1}^{M} \omega_k^{(i)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{M} \omega_k^{(i)}}$$

ADVANCED TECHNIQUE: EM PCA

- Use iterative E-M steps to simultaneously solve for the principal vectors and principal components of <u>weighted</u> data (or even missing data!)



S. Bailey 2012 https://arxiv.org/abs/1208.4122

https://github.com/jakevdp/wpca

ADVANCED TECHNIQUE: BINNED GMM

- Generalize (unbinned) standard GMM to add:
    - Flat background component (sky).
    - Binned photons (pixels).
- The binning requires adding an extra latent variable for each photon (!), but the resulting algorithm is fast & accurate.

https://github.com/dkirkby/bbmix