Opinionated Lessons in Statistics

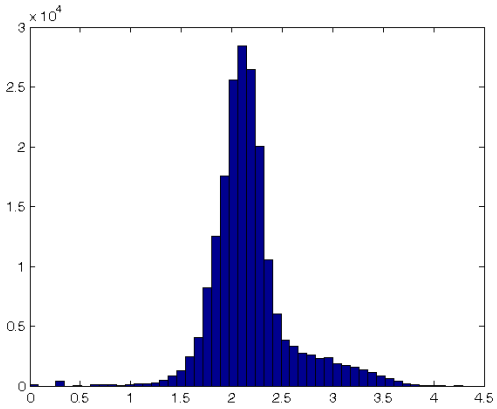
by Bill Press

#28 Gaussian Mixture Models in 1-D

Gaussian Mixture Models (GMMs)

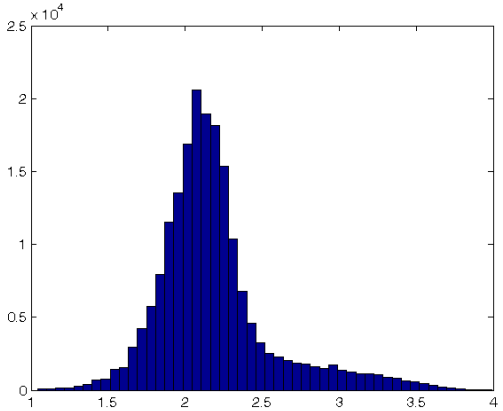
- Yet another method for fitting multiple Gaussians to (possibly) a set of multi-dimensional data points
  - properties of Gaussians are used in detail: doesn't easily generalize to other fitting functions
- But uses the raw data points
  - no binning!
  - hence useful in multidimensions
- Exemplifies Expectation Maximization (EM) methods
  - an important class of methods (Dempster, Laird, & Rubin)
  - we'll show some theory later
- Let's first try it in 1-D on the data set of exon lengths

```
g = readgenestats('genestats.dat');
exons = cell2mat(g.exonlen);
hist(log10(exons),50)
```



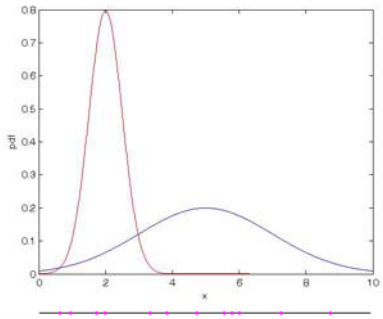

```
data = log10(exons(log10(exons)>1
& log10(exons)<4));
hist(data,50)
```

(Let's trim the outliers.)



Key to the notational thicket:

M dimensions
$k = 1 \ldots K$ Gaussians "components"
$n = 1 \ldots N$ data points
$P(k)$ population fraction in k
$P(\mathbf{x}_n)$ model probability at $\mathbf{x}_n$



$\boldsymbol{\mu}_k$ (the K means, each a vector of length M)

$\boldsymbol{\Sigma}_k$ (the K covariance matrices, each of size $M \times M$)

$P(k|n) \equiv p_{nk}$ (the K probabilities for each of N data points)

"probabilistic assignment" of a data point to a component!

$$\mathcal{L} = \prod_n P(\mathbf{x}_n)$$ overall likelihood of the model

$$P(\mathbf{x}_n) = \sum_k N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P(k)$$ specify the model as a mixture of Gaussians

$$N(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left[-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}) \cdot \boldsymbol{\Sigma}^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu})\right]$$

Goal is to find *all* of the above, starting with only the $\mathbf{x}_n$

(So far this could be frequentist or Bayesian, although it was invented by frequentists.)

Expectation, or E-step: suppose we know the model, but not the assignment of individual points.

(so called because it's probabilistic assignment by expectation value)

$$p_{nk} \equiv P(k|n) = \frac{N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P(k)}{P(\mathbf{x}_n)}$$

Maximization, or M-step: suppose we know the assignment of individual points, but not the model.

$$\hat{\boldsymbol{\mu}}_k = \sum_n p_{nk} \mathbf{x}_n \Big/ \sum_n p_{nk}$$

$$\hat{\boldsymbol{\Sigma}}_k = \sum_n p_{nk} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k) \otimes (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k) \Big/ \sum_n p_{nk}$$

$$\hat{P}(k) = \frac{1}{N} \sum_n p_{nk}$$

(so called because [theorem!] the overall likelihood increases at each step)

- Can be proved that alternating E and M steps converges to (at least a local) maximum of overall likelihood
- Convergence is sometimes slow, with long "plateaus"
- Often start with k randomly chosen data points as starting means, and equal (usually spherical) covariance matrices
  - but then had better try multiple re-starts

Because Gaussians underflow so easily, a couple of tricks are important:

1) Use logarithms!

$$\log N(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\cdot\boldsymbol{\Sigma}^{-1}\cdot(\mathbf{x}-\boldsymbol{\mu}) - \frac{M}{2}\log(2\pi) - \frac{1}{2}\log\det(\boldsymbol{\Sigma})$$

2) Do the sum $$P(\mathbf{x}_n) = \sum_k N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P(k)$$

by the "log-sum-exp" formula:

$$\log\left(\sum_i \exp(z_i)\right) = z_{\max} + \log\left(\sum_i \exp(z_i - z_{\max})\right)$$

We'll skip these tricks for our 1-D example, but use them (via NR3) in multidimensional examples.

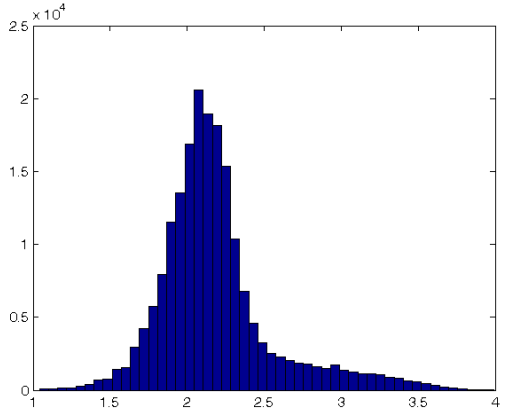
The E-step in 1-D looks like this:

```
mu = [2. 3.];
sig = [0.2 0.4];
pr = @(x) exp(-0.5*((x-mu)./sig).^2)./sig;
pr(2.5)
ans =
    0.2197    1.1446

prn = @(x) pr(x)./sum(pr(x));
prn(2.5)
ans =
    0.1610    0.8390

prns = zeros([numel(data),2]);
for j=1:numel(data); prns(j,:)=prn(data(j)); end;
prns(100:110,:)
ans =
    0.9632    0.0368
    0.0803    0.9197
    0.7806    0.2194
    0.6635    0.3365
    0.5819    0.4181
    0.9450    0.0550
    0.9801    0.0199
    0.8824    0.1176
    0.9703    0.0297
    0.9661    0.0339
    0.7806    0.2194
```

Probabilities of each component. Don't need to get the normalizing π's right, since will (Bayes) normalize across components...

…like this. Normalized probability.

Compute for all the points (show only 10).



The M-step in 1-D looks like this:

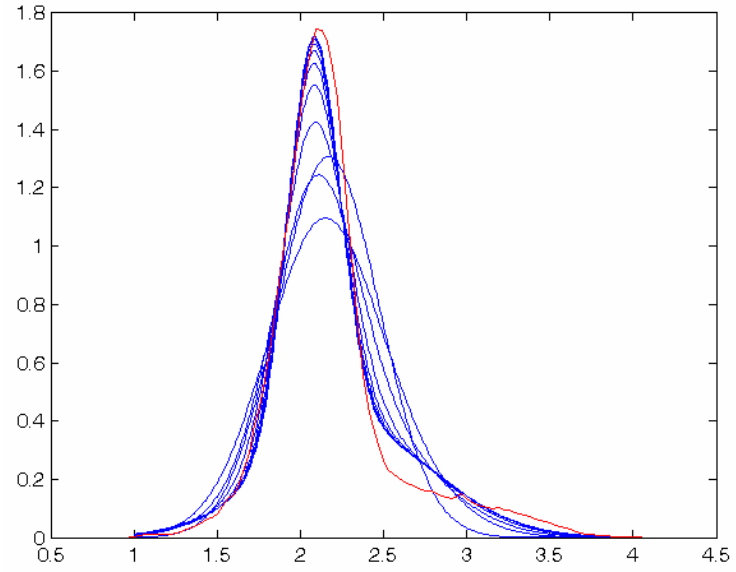
```
mu = sum(prns.*repmat(data,[1,2]), 1) ./ sum(prns,1)
xmmu = repmat(data,[1,2]) - repmat(mu,[numel(data),1]);
sig = sqrt(sum(prns .* xmmu.^2, 1) ./ sum(prns,1))
pop = sum(prns,1)/numel(data)
```

(Elegant in Matlab's data-parallel language. But, unfortunately, doesn't generalize well to multidimensions. We'll use NR3 instead, which also includes the tricks already mentioned.)

Let's show 10 iterations

```
mu = [randsample(data,1) randsample(data,1)]
sig = [.3 .3]
for jj=1:10,
    pr = @(x) exp(-0.5*((x-mu)./sig).^2)./(2.506*sig);
    prn = @(x) pr(x)./sum(pr(x));
    for j=1:numel(data); prns(j,:)=prn(data(j)); end;
    mu = sum(prns.*repmat(data,[1,2]), 1) ./ sum(prns,1);
    xmmu = repmat(data,[1,2]) - repmat(mu,[numel(data),1]);
    sig = sqrt(sum(prns .* xmmu.^2, 1) ./ sum(prns,1));
    pop = sum(prns,1)/numel(data);
    thefunc = @(x) sum(pop.*pr(x),2);
    x = 1:.01:4;
    f = arrayfun(thefunc,x);
    plot(x,f,'b');
    hold on;
end;
[f x] = ksdensity(data);
plot(x,f,'r')
hold off;
```

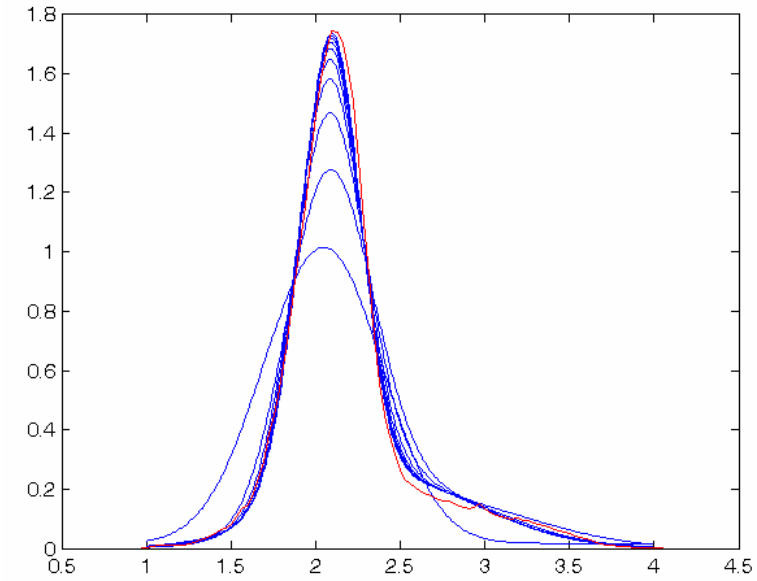
Matlab has "kernel smoothing density estimate" that is convenient for plotting the actual data as a smooth curve. (But I wouldn't trust it further than that!)



2 components

```
mu =
    2.0806    2.3100
sig =
    0.1545    0.5025
pop =
    0.5397    0.4603
```

Notice that this makes a different set of "compromises" from other fitting methods. It *hates* having points in regions of "zero" probability and would rather tolerate only fair fits in the "shoulders". It is not the same as weighted LS to binned data!



3 components

```
mu =
    2.1278    2.0260    2.4186
sig =
    0.1515    0.1892    0.5451
pop =
    0.3403    0.3399    0.3198
```

More components will converge to an excellent approximation. This does *not* mean that the components mean anything physically!

In this example, almost all starting points give the same, presumably global, max likelihood.