The EM algorithm

Analysis of a T cell frequency assay

Karl Broman

Biostatistics & Medical Informatics, UW–Madison

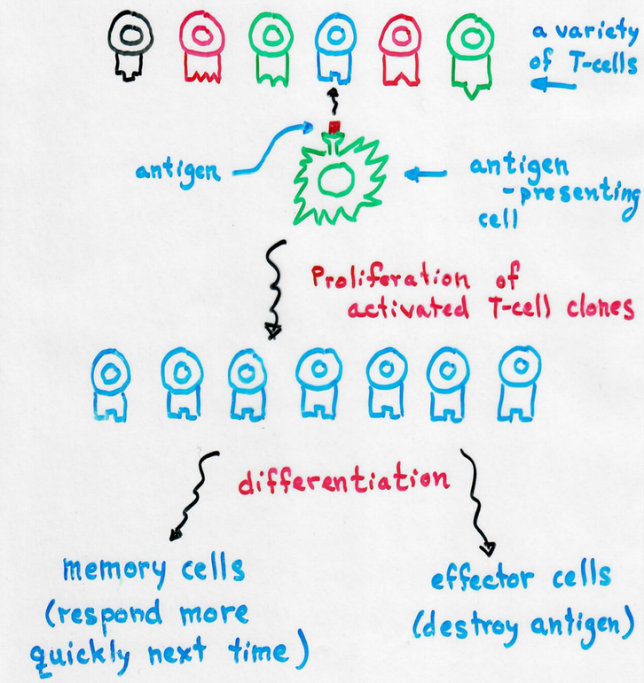
kbroman.org
github.com/kbroman
@kwbroman
Course web: kbroman.org/AdvData

In this lecture, we'll look at the EM algorithm, which is a broadly useful algorithm for getting maximum likelihood estimates. We'll learn about it through a case study on a T cell frequency assay, for measuring the effectiveness of a vaccine.

Goal: Estimate the frequency of T-cells in a blood sample that respond to two test antigens.

Real goal: Determine whether a vaccine causes an increase in the frequency of responding T-cells.

Broman K, Speed T, Tigges M (1996) J Immunol Meth 198:119-132 doi.org/b54v33

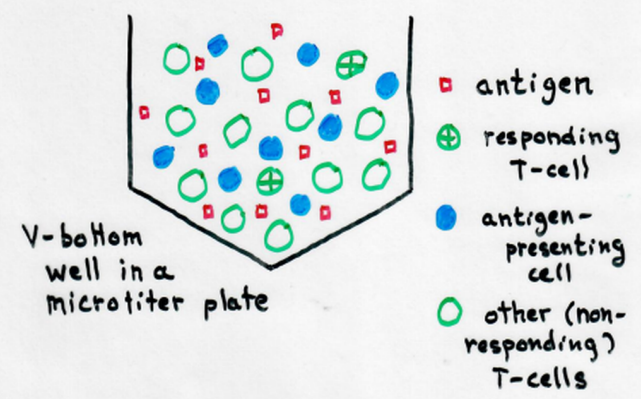


We're considering an assay that seeks to estimate the frequency of T cells in a blood sample that respond to a particular antigen. Antigens are bits of foreign protein, such as chewed-up virus. Antigen-presenting cells take up these proteins and present them on their surface. This ultimately leads to the proliferation of specific T cells with receptors complementary to the antigen, which differentiate to create memory cells (which lead to a faster response later) or effector cells (which go around destroying antigen).

The real goal was to determine whether a particular vaccine was looking to be effective by leading to an increase in the frequency of responding T-cells.

The assay

- Combine:
  - diluted blood cells + growth medium
  - antigen
  - ^{3}H-thymidine
- Replicating cells take up ^{3}H-thymidine.
- Extract the DNA and measure its radioactivity



In the assay we're considering, you combine diluted blood cells and growth medium with antigen and ^{3}H-thymidine (the T in DNA, but radioactive). Replicating cells take up ^{3}H-thymidine, and you then extract the DNA and measure the radioactivity.

Usual approaches

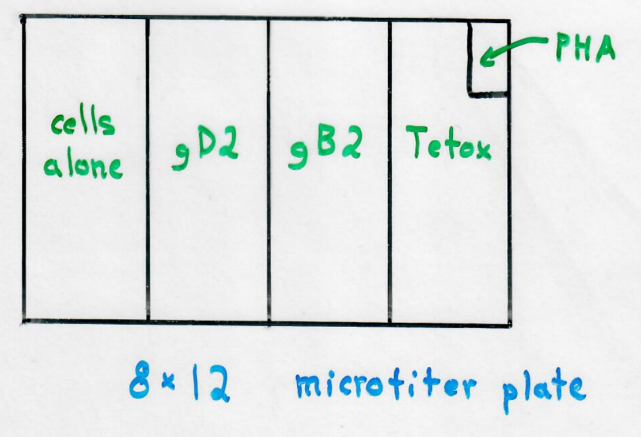
- Use 3 wells with antigen and 3 wells without antigen, and take the ratio of the averages
- Limiting dilution assay
  - Several dilutions of cells
  - Many wells at each dilution

The usual approaches use three wells with antigen and three cells without, and just look at like the ratio of the average response. This was viewed as too crude for the current context.

Alternatively, you could do a limiting dilution assay: use several dilutions of cells, with many wells at each dilution. But this was viewed as requiring much more blood than the subjects would generally be willing to offer.

Our assay

Study a single plate or pair of plates at a single dilution.



And so our collaborator was interested in trying to get by with a single 8×12 plate of wells, or maybe a pair of plates, at a single dilution. There would be some control wells (with just cells), and then a couple of sections of wells with one of the two test antigens, and then some wells with Tetanus toxin (as a positive control; all subjects will have been exposed to Tetanus toxin, via a tetanus vaccine). The PHA wells are another positive control.

Data

LDA 713, plates 1 and 2
11,400 cells per well

cells alone			gD2			gB2			Tetox		PHA
179	249	460	2133	2528	2700	2171	1663	6200	761	9864	12842
346	1540	306	8299	1886	3245	1699	2042	3374	183	7748	10331
117	249	1568	1174	4293	979	1222	1536	2406	6497	2492	6188
184	414	308	2801	2438	1776	2193	3211	1936	2492	5134	927
797	233	461	1076	1527	2866	2205	2278	2215	3725	3706	4050
305	348	480	3475	902	3654	2046	1285	1187	9899	5891	3646
1090	159	89	1472	90	3639	657	2393	1814	3330	4174	2389
280	571	329	4448	3643	881	3462	2118	1013	8793	4313	672
178	111	630	4699	5546	5182	3982	3104	2496	4275	2831	9727
244	593	259	5622	560	1073	1479	2978	4362	5017	5074	10706
261	964	167	2991	3390	3986	2321	2157	3278	8216	3579	3538
221	544	299	1838	4368	322	1022	1554	2980	2732	6177	5212
533	228	615	1938	4046	333	3253	5091	2843	200	1110	5063
818	98	160	1032	3269	4918	1778	3810	2372	6355	1869	2695
234	472	243	4143	3351	1118	530	1174	1881	3447	4491	2945
169	481	478	3237	1565	2211	2460	2715	4793	3029	6225	4679

Here's an example data set. Higher numbers mean more radioactivity, indicating cells have taken up ^{3}H-thymidine into their DNA.

Traditional analysis

- Split wells into +/– using a cutoff (e.g., mean + 3 SD of "cells alone" wells)

  positive = one or more responding cells
  negative = no responding cells

- Imagine that the number of responding cells in a well is Poisson(λ_i) for group *i*

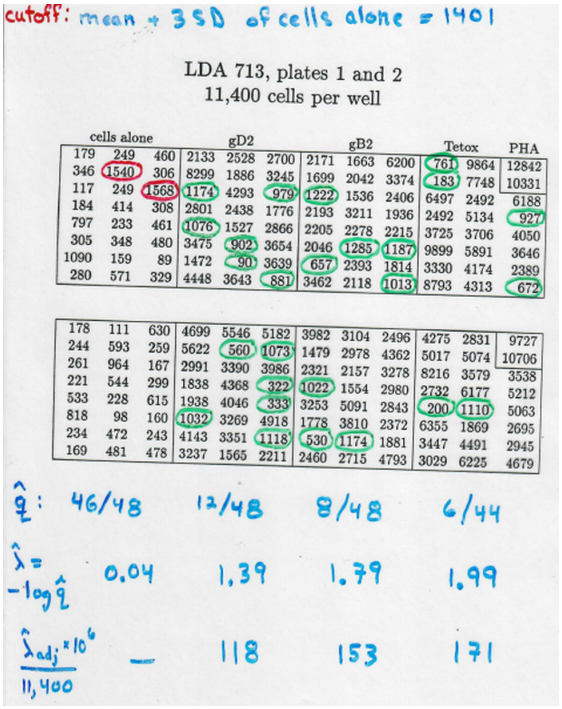
  Pr(no responding cells) = $e^{-\lambda_i}$

  $$\hat{\lambda}_i = -\log\left(\frac{\text{\# negative wells}}{\text{\# wells}}\right)$$

The tranditional analysis of such data was to split wells into positive and negative wells according to whether they seem to have one or more responding cells, or to have no responding cells. Assuming that the number of responding cells in a well follows a Poisson distribution, you can use the proportion of negative cells to get an estimate of the average number of responding cells per well.

The Poisson distribution is like a binomial(n,p) distribution where n is really large and p is really small, with the mean of the Poisson distribution $\lambda = np$.

Analysis



cutoff: mean + 3 SD of cells alone = 1401

LDA 713, plates 1 and 2
11,400 cells per well

cells alone			gD2			gB2			Tetox		PHA
179	249	460	2133	2528	2700	2171	1663	6200	761	9864	12842
346	1540	306	8299	1886	3245	1699	2042	3374	183	7748	10331
117	249	1568	1174	4293	979	1222	1536	2406	6497	2492	6188
184	414	308	2801	2438	1776	2193	3211	1936	2492	5134	927
797	233	461	1076	1527	2866	2205	2278	2215	3725	3706	4050
305	348	480	3475	902	3654	2046	1285	1187	9899	5891	3646
1090	159	89	1472	90	3639	657	2393	1814	3330	4174	2389
280	571	329	4448	3643	881	3462	2118	1013	8793	4313	672

178	111	630	4699	5546	5182	3982	3104	2496	4275	2831	9727
244	593	259	5622	560	1073	1479	2978	4362	5017	5074	10706
261	964	167	2991	3390	3986	2321	2157	3278	8216	3579	3538
221	544	299	1838	4368	322	1022	1554	2980	2732	6177	5212
533	228	615	1938	4046	333	3253	5091	2843	200	1110	5063
818	98	160	1032	3269	4918	1778	3810	2372	6355	1869	2695
234	472	243	4143	3351	1118	530	1174	1881	3447	4491	2945
169	481	478	3237	1565	2211	2460	2715	4793	3029	6225	4679

$\hat{q}$: 46/48 12/48 8/48 6/44

$\hat{\lambda} = -\log \hat{q}$: 0.04 1.39 1.79 1.99

$\frac{\hat{\lambda}_{adj} \times 10^6}{11,400}$: — 118 153 171

Applying the simple method to the example data, we might derive a cutoff between negative and positive wells as mean + 3 SD of the cells-alone wells. Then count the negative wells and convert to get estimates of the number of responding cells in a well.

Ultimately, we subtract off the baseline estimate for the cells-alone wells and re-scale to number of responders per million cells.

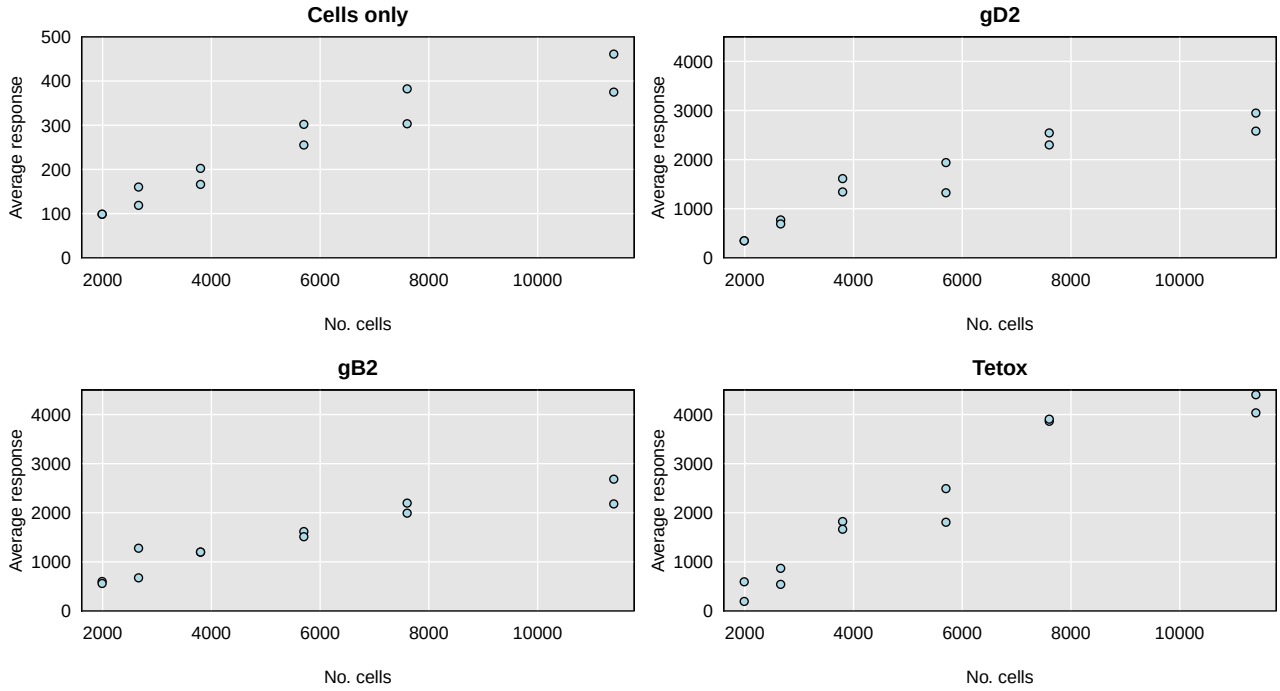
Problems

- Hard to choose cutoff
- Potential loss of information

But it can be hard to choose a cutoff, and there's a potential loss of information by converting the quantitative values into binary data.

And since we're trying to get by with just one or two plates of data at a single dilution, we really want to try to extract as much information as possible from the observed data.

Response vs no. cells



For a couple of subjects we had a dilution series of data. If you look at the average response in each section of cells by cell dilution in a subject, there is a clear dose-response relationship which suggests that there's more information about the cellular response than just positive/negative status of the wells.

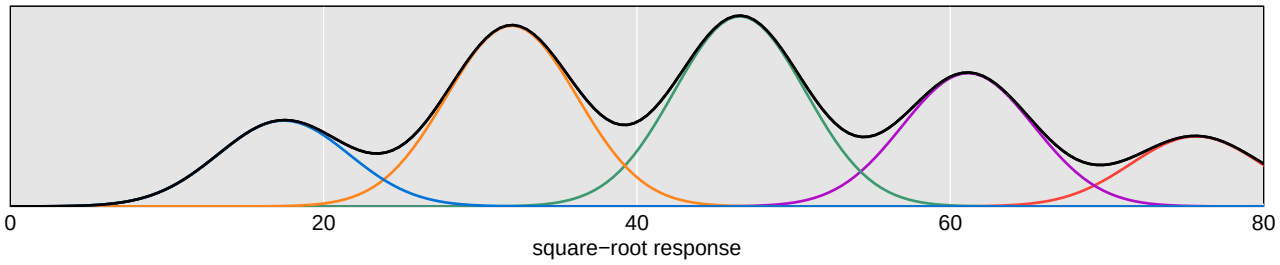
Model

k_{ij} = Number of responding cells (unobserved)
y_{ij} = square-root of response

Assume $k_{ij} \sim \text{Poisson}(\lambda_i)$

$y_{ij} \mid k_{ij} \sim \text{Normal}(a + bk_{ij}, \sigma)$

(k_{ij}, y_{ij}) mutually independent



To extract more information from the quantitative response, let's create a model for these observed data. It's natural to assume that the number of responding cells in a well follows a Poisson distribution. Let's further assume that some transformation of the quantitative response is linear in the number of responding cells, with normally distributed residual variation. And partly due to the shape of the curves on the previous slide, we went with square-root of the response.

So we have normally distributed response for wells with 0 responders, a shift upwards for wells with 1 responder, etc. The overall response distribution is a "mixture" of these normally distributed components, with the relative proportions of the components being according to the Poisson probabilities.

log Likelihood

$$
\begin{aligned}
l(\boldsymbol{\lambda}, a, b, \sigma) &= \sum_{i,j} \log \Pr(y_{ij}|\lambda_i, a, b, \sigma) \\
&= \sum_{i,j} \log \left[\sum_k \Pr(k|\lambda_i) \Pr(y_{ij}|k, a, b, \sigma) \right] \\
&= \sum_{i,j} \log \left[\sum_k \left(\frac{e^{-\lambda_i} \lambda_i^k}{k!} \right) \phi \left(\frac{y_{ij} - a - bk}{\sigma} \right) \right]
\end{aligned}
$$

Our goal is to get estimates of the mean numbers of responding cells in each group of wells, as well as the plate-specific parameters a, b, and σ.

We can write down the likelihood for the data; optimizing this likelihood would give us the MLEs for the parameters.

EM algorithm

- Iterative algorithm useful when there is missing data that if observed would make things easy
- Dempster et al. (1977) JRSS-B 39:1-22 doi.org/gfxzrv
- Start with some initial estimates
- E-step: expected value of missing data given current estimates
- M-step: MLEs replacing missing data with their expected values
- Advantages
  - often easy to code
  - super stable
  - log likelihood is non-decreasing

The EM algorithm is a particular optimization method for getting the MLEs that is useful in this sort of situation where we have missing data (the k's). If we knew the number of responding cells in each well, we could estimate the averages by just taking the averages of the k's in each group, and we could estimate a, b, and σ by linear regression of the responses on the k's.

Not knowing the k's, the EM algorithm works by iterating between an E step (where you get expected values for the k's, given the observed data and given current estimates of the parameters) and an M step (where you derive new and improved estimates, using the expected values of the k's in place of their true (but unknown) values.

The EM algorithm has a number of advantages: it is often easy to implement in software, it can be super stable (meaning no matter what starting values you use, the algorithm will converge to some finite values that are at least reasonable), and it can be proven that across iterations, the log likelihood is non-decreasing.

Normal/Poisson model

E-step:

$$\begin{aligned}
\Pr(k = s|y, \lambda, a, b, \sigma) &= \frac{\Pr(k = s|\lambda)\Pr(y|k = s, a, b, \sigma)}{\sum_s \Pr(k = s|\lambda)\Pr(y|k = s, a, b, \sigma)} \\
&= \frac{\left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}{\sum_s \left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}
\end{aligned}$$

$$\mathrm{E}(k|y, \lambda, a, b, \sigma) = \frac{\sum_s s\left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}{\sum_s \left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}$$

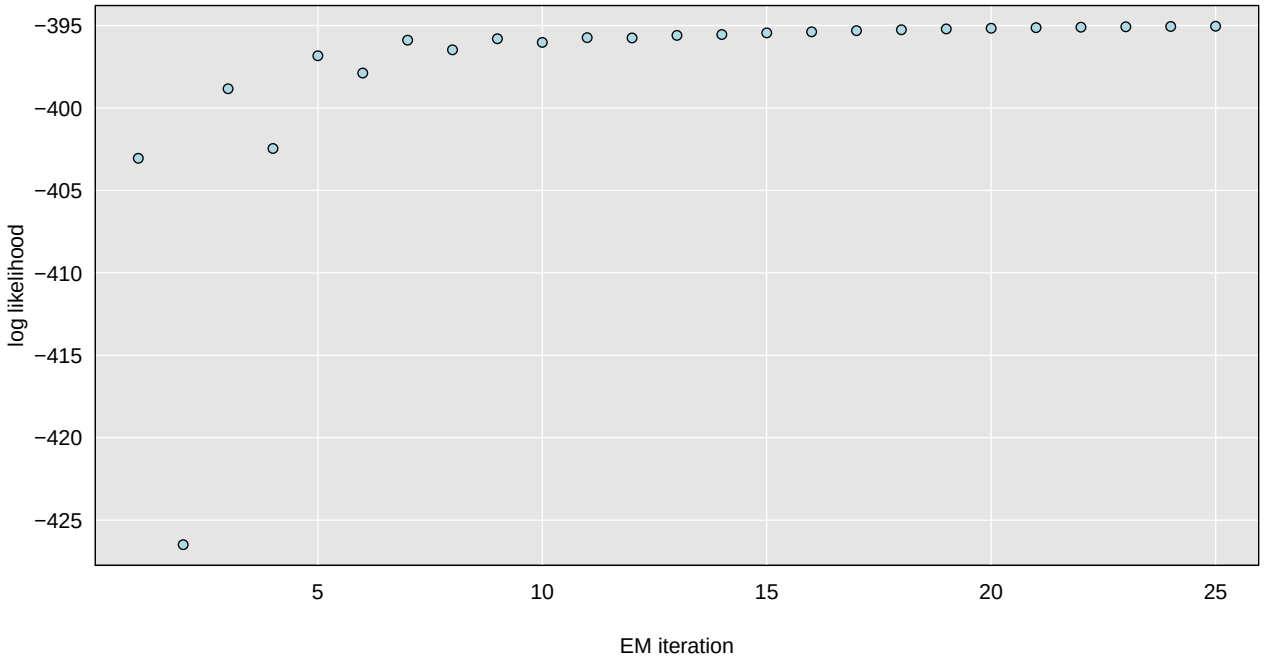
M-step: Regress y on $\mathrm{E}(k|y)$

For this particular model, the idea is to calculate expected values for the k's given the observed data and given current values for the parameters, using Bayes's rule. (In practice, the sums go up to some maximum k, like 20, where the posterior probability for k has gotten really small.)

At the M step, you then take averages of these values, and regression y on these values.

Go back and forth between the two steps until the estimates converge.

Oops, that didn't work



Here's the log likelihood plotted against iteration. As you can see, the thing didn't really work as it should have. The log likelihood is going up and down rather than being non-decreasing, as I said it should be.

EM algorithm, more formally

- Calculate expected complete-data log likelihood, given observed data and observed parameters, and then maximize that.

$$l^{(s)}(\theta) = \mathrm{E}\{\log f(y, k|\theta)|y, \hat{\theta}^{(s)}\}$$

- In practice, it's usually a linear combination of the sufficient statistics, so you focus on those.
- Here, we need not just $\sum k$ and $\sum ky$, but also $\sum k^2$.

It turns out that, formally, you need to calculate the expected value of the complete-data log likelihood function. In practice, this is a linear combination of the sufficient statistics.

And in this situation the sufficient statistics include not just $\sum k$ and $\sum ky$, but also $\sum k^2$.

Note taking account of $\sum k^2$ was our problem, as $\mathrm{E}(k^2) \neq [\mathrm{E}(k)]^2$.

This took a while for me to figure out, as I initially thought there was a bug in my code, but really there was a bug in my understanding of the EM algorithm. Calculating the log likelihood and following it by iteration was a super-useful diagnostic.

EM algorithm, again

E step: we also need

$$\mathrm{E}(k^2|y,\lambda,a,b,\sigma) \;=\; \frac{\sum_s s^2 \left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}{\sum_s \left(\frac{e^{-\lambda}\lambda^s}{s!}\right)\phi\left(\frac{y-a-bs}{\sigma}\right)}$$

M step: we want $\hat{\beta} = (X'X)^{-1}(X'y)$

$$\text{where } (X'X) \quad \text{is like} \quad \begin{pmatrix} n & \sum k \\ \sum k & \sum k^2 \end{pmatrix}$$

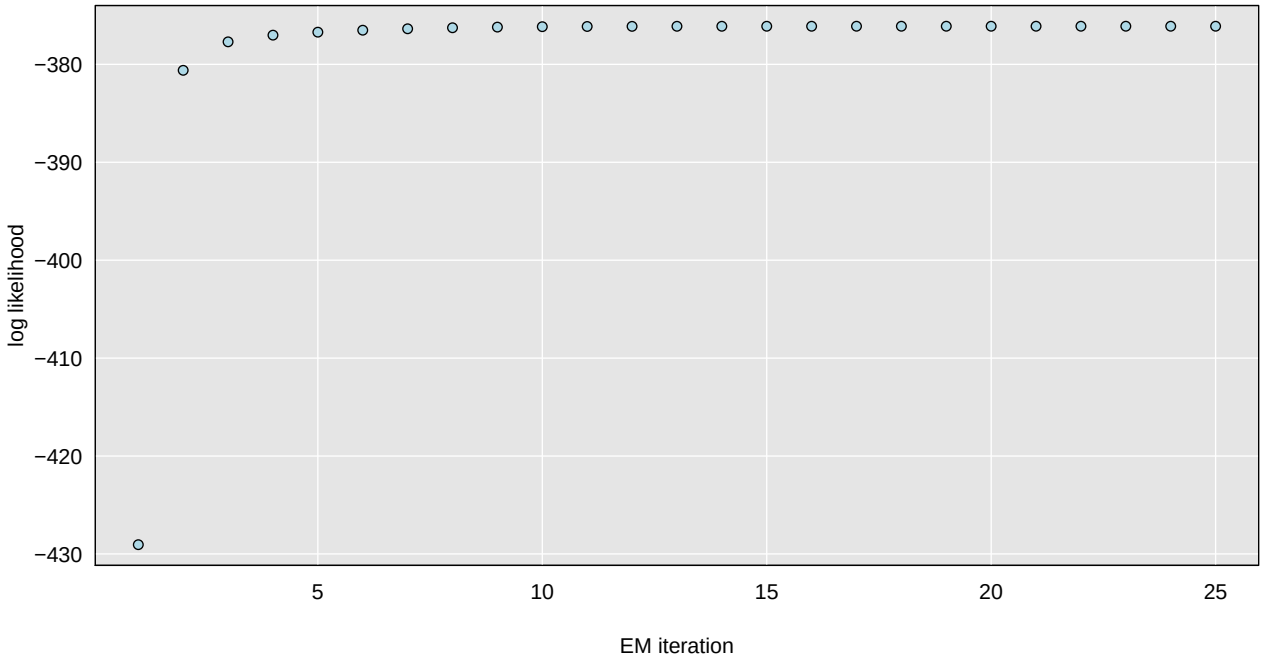
$$\text{and } (X'y) \quad \text{is like} \quad \begin{pmatrix} \sum y \\ \sum ky \end{pmatrix}$$

Going back to the EM algorithm, we need to include the calculation of the expected value of k^2 given the observed data and given the parameter estimates. This is easy enough, because it's just like calculation the expected value of k.

More difficult is that the linear regression part of the M step involves sort of going back-to-basics. You need to figure out where the k^2 values enter into things and plug in $\mathrm{E}(k)$ for k and $\mathrm{E}(k^2)$ for k^2.

I think the easiest way to do this is to look at the $X'X$ matrix, which will be 2×2, and stick the $\mathrm{E}(k^2)$ values in there.

Ah, that's better



Having corrected our algorithm, here is the trace of the log likelihood by iteration. Non-decreasing, as it should be.

Difficulties

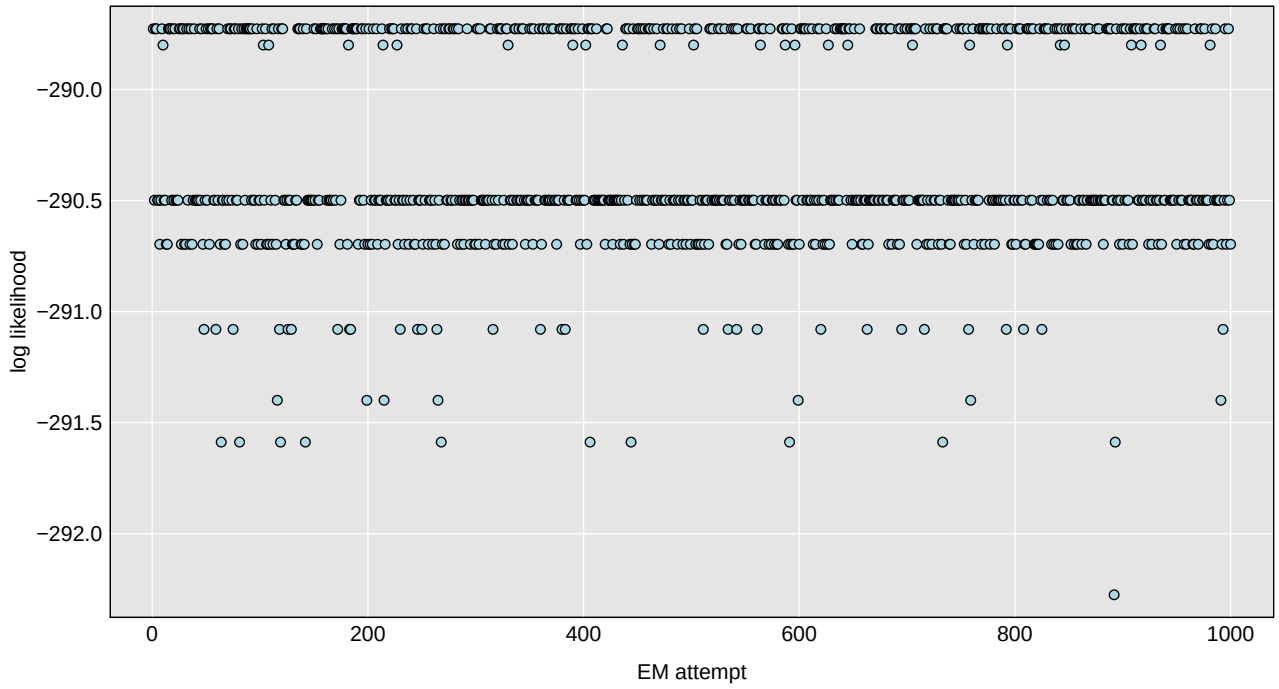
- Starting values
- Multiple modes

The EM algorithm may sound great, but I've swept some important difficulties under the rug.

First, we need starting values. How do we get those? The easiest way is to use our crude method of splitting wells into positive/negative; that will give us some estimates of the λ values. We can look at the relationship between the average responses and those λ's to get estimates of a, b, and σ.

More difficult is that while the EM algorithm will converge to something, it won't necessarily converge to the global MLEs. There can be multiple modes in the likelihood surface. A solution to this is to use lots of random starting points and pick the best of the converged estimates.

Multiple modes



Here's a particular example. I initiated the EM algorithm at 1000 different random starting points, and it converged to 8 different sets of estimates.

Multiple modes

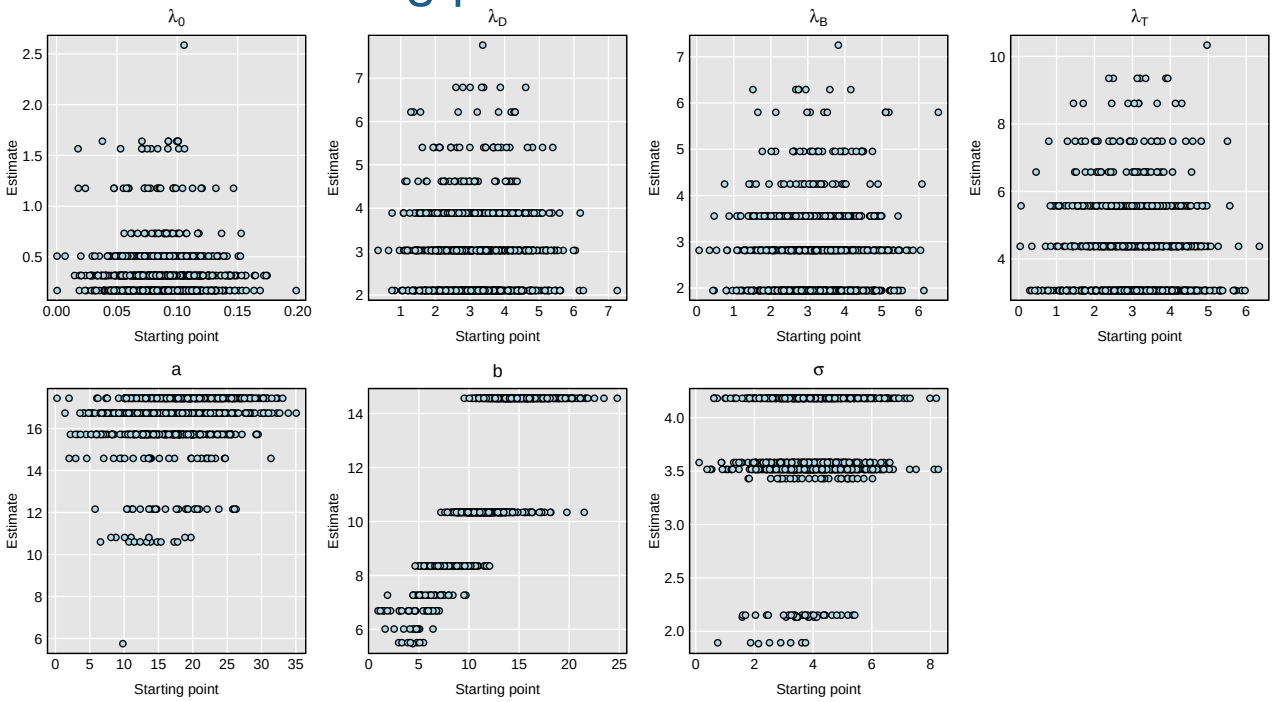
	λ_0	λ_D	λ_B	λ_T	a	b	σ	log lik	no. hits
1	0.32	3.03	2.82	4.37	16.73	10.34	3.52	–289.73	331
2	1.18	5.40	4.95	7.49	12.16	6.69	2.15	–289.80	26
3	0.17	2.10	1.95	3.07	17.44	14.56	4.18	–290.50	415
4	0.51	3.89	3.56	5.58	15.72	8.35	3.58	–290.70	180
5	0.73	4.62	4.25	6.58	14.58	7.27	3.43	–291.08	30
6	1.64	6.79	6.29	9.35	10.81	5.51	1.89	–291.40	7
7	1.57	6.22	5.80	8.61	10.60	6.02	2.13	–291.59	10
8	2.59	7.76	7.25	10.34	5.75	5.47	1.88	–292.27	1

Here are the 8 modes we found, along with their log likelihood and the numbers of times they were hit, out of 1000.

This is maybe a bit concerning. Actually, it is quite concerning. Our estimated mode has estimated λ's around 3, but there's another mode just slightly lower in likelihood that has the estimates around 5, and another mode just below that with estimates at 2.

The model is great, but the data aren't quite sufficient to fit it, is what you might conclude. The different modes here have a lot in common with that choice of cutoff in the traditional approach to analyzing these sort of data.

Estimate vs. starting point



Here's a plot of the estimates vs the starting value that I had used for that parameter. Each panel is one parameter and the 1000 points are the 1000 starting points. As you can see, for most of the parameters there seems no real relationship between where you start and where you end up.

But the b parameter (slope in response vs responding cells) shows a very strong relationship: the initial value for b has a big effect on where you end up.

Principles

- Start with an understanding of the problem and data
- Think about a model for the data-generating process

Some important principles that were guiding this work: start with an understanding of the problem and the data, and think about a model for the data-generating process. That led us to this normal/Poisson mixture model.

Lessons

- The EM algorithm is really useful
- Use the log likelihood as a diagnostic when implementing an EM algorithm

And what have we learned? The EM algorithm is really useful, and you should use the log likelihood as a diagnostic when implementing it.

Software development time

- Formulating the problem
- Writing the code
- Debugging the code
- Executing the code

Real selling points for the EM algorithm come up when you look at the time involved in writing a program to give parameter estimates. Programming time includes the time to formulate the problem, the time to write the code, debug the code, and execute the code.

Where the EM algorithm really shines is on the "writing the code" and "debugging the code" sections. The EM algorithm is often rather simple to code, and it comes with a built-in diagnostic.

Impact

- I'm pretty sure that the vaccine they were working on didn't work well.
- R package `npem`, but I never put it on CRAN, and no one has ever asked me about it.
- Our paper has like 9 citations: no one has ever really used the method.

What can we say about the impact of this work?

It didn't seem to have much impact, I think. The vaccine we were studying didn't seem to be very effective.

The R package I wrote, implementing the method, is on GitHub, but I never did put it on CRAN. I distributed it just through my personal web site. No one has ever asked me about it, so probably it has never been used.

The paper we wrote about this work has just like 9 citations. No one has every really used the method. Bummer.

Further things

- Standard errors should always be required.
  - But usually painful to obtain
  - We used the SEM algorithm of Meng and Rubin (1991) doi.org/dk27
- Could more formally investigate the appropriate transformation
  - See Box and Cox (1964) doi.org/gfrhvs
  - Box-Cox transformation is $g(y) = (y^c - 1)/c$ for $c \neq 0$ and $= \log y$ for $c = 0$
  - Key issue is change-of-variables in the density; as a result you add $\sum_{ij}(c-1)\log y_{ij}$ to the log likelihood

A couple of further points. Standard errors should always be required. If you give an estimate, you need to give a standard error. Getting standard errors here is a bit painful. We used the SEM algorithm, which uses the rate of convergence of EM. It's a bit clunky, but it doesn't require writing a whole lot of new code, and it seems to give reasonable results.

Further, we could more formally investigate the appropriate transformation. See the Box and Cox (1964) paper, The key issue is the change-of-variables in the density, so that you have to add a quantity to the log likelihood. It's slightly tricky to get right, but it is a nice way to use the data to determine the appropriate transformation.