Maximum-likelihood estimation, latent variables and the Expectation/Maximization algorithm

Micha Elsner

August 27, 2012

Building models

The next two lectures: overview of some statistical methods

- A review for people who know this stuff
- A basic survival guide for people who don't

This lecture

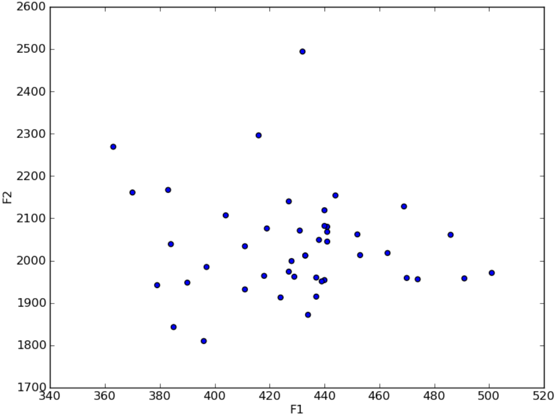
Standard *frequentist* techniques for building models with hidden variables
Classical techniques from the '70s (popular since the '90s)

Next lecture

Bayesian methods
Popular since the mid '00s

Fully observed data

A simple toy example: a baby observes the F_1 and F_2 of 45 tokens of the vowel /i/ (spoken by men, at "steady state")

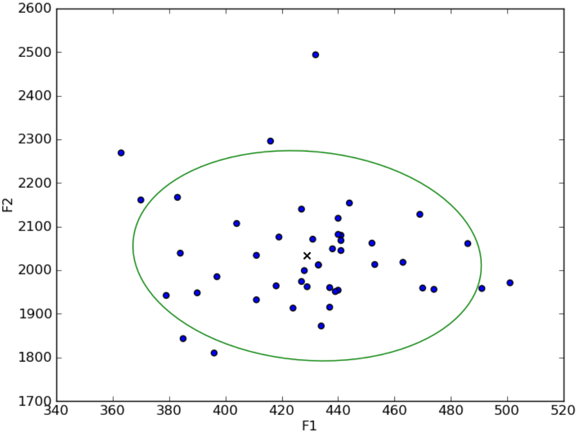


from Hillenbrand, Getty, Clark and Wheeler 99

High-level modeling assumption

/i/ sounds are distributed in an ellipse-shaped region surrounding a common mean

(Why? Mathematical convenience, mostly. Just go with it...)



Mathematically...

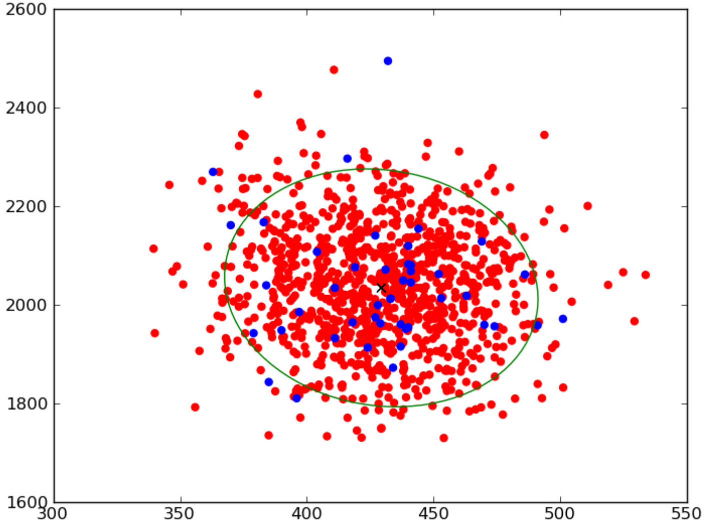
Treat the vowel tokens as samples from a normal (Gaussian) distribution with unknown mean μ and covariance Σ

Generative model

A probability distribution over the observed data:

- Different use of *generative* from Chomsky
- Contrast with models that fit part of the data (outcome) from other parts (predictors)— like regressions
- Usually has some unknown *parameters*
- Possible to *sample* a synthetic dataset from the model

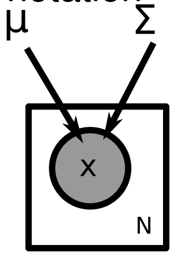
Sampled data



Notation

Let $X : x_0 \ldots x_N$ be the list of vowels, with $N = 45$.

Graphical model notation



$$x_i \sim N(\mu, \Sigma)$$

- $\sim$: sampled from, distributed according to
- N: normal distribution

- Circle = random variable
- Gray background: observed value
- Box = many variables
- No circle = parameter
- Arrow: conditioned on

Learning

Our hypothetical baby assumes the data *must* be generated from a model of this family... but what are μ and Σ?

Principle of maximum likelihood

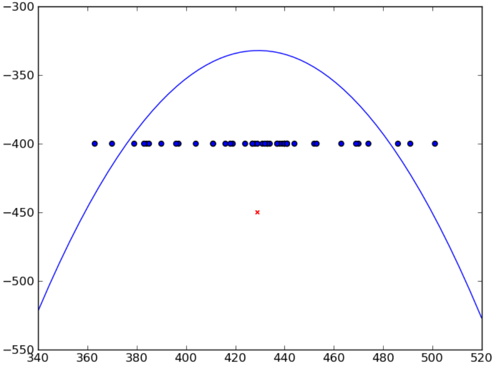
Choose values for the parameters that maximize the probability of the data

- *Likelihood*: data probability as function of the parameters
- Actually, often the *log*-likelihood
  - Mathematically convenient and doesn't underflow as much

The likelihood

Log-probability of our dataset as a function of μ_1 with other parameters at optimal values

(The graph for μ_1 and μ_2 is 3d; the whole graph is 6d)



(Blue points: observations in F_1 space; red x: sample mean)

The maximum-likelihood estimator

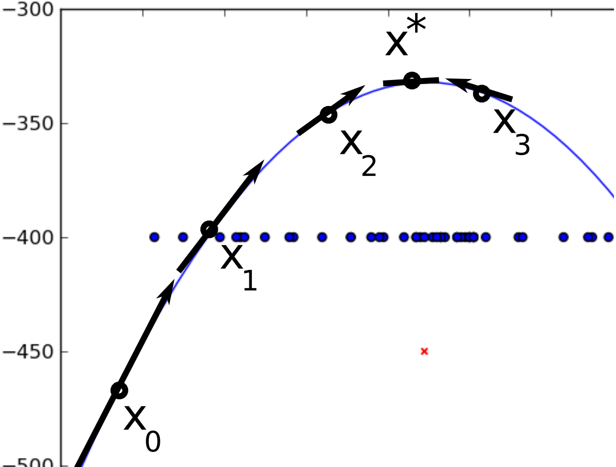
Choose $\hat{\mu}_1$ (the baby's *estimate* of the value of μ_1) according to principle of maximum likelihood

- In this case, can just choose the sample mean!
  - More general principle: *methods of moments*
- In a second, will see more complex models for which this doesn't work

Gradient ascent: generic MLE

Maximize function by moving uphill from point to point

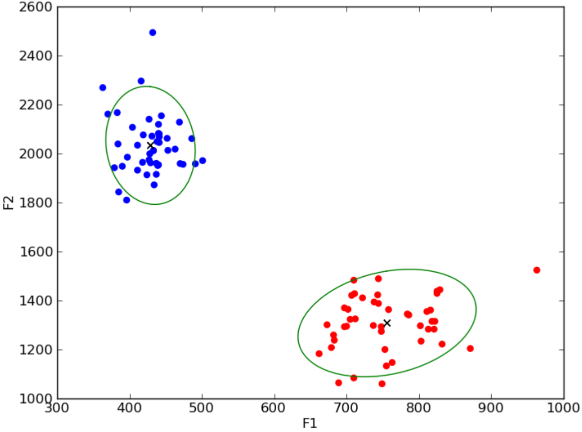
- Pick initial point
- Compute derivative
- Step uphill and repeat



A little more complicated

Now, the baby observes 90 vowel tokens...

- Given the language has two vowels, /i/ and /a/
- Each with unknown mean and covariance



Mathematics

Introduce some auxiliary indicator variables z: is this token /i/ or /a/?

- * /i/ and /a/ are labels for *our analysis*... actually cluster 1 or cluster 2
- z_i will be 0 if x_i is an /i/ and 1 if it's an /a/
- Prior probability of an /i/ determined by a new parameter π (ie, $\pi = .5$ means about half /i/ sounds)

Latent variables

Random variables in our model whose values we don't observe

Mixture model

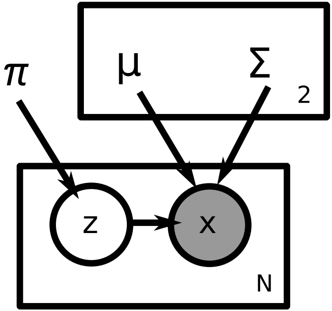
Models whose latent variables indicate which cluster an observation comes from are *mixtures*...
Since the individual vowels here are Gaussian, this is a *mixture of Gaussians*

Writing it down

Let $X : x_0 \dots x_N$ be the list of vowels, with $N = 90$.
Let $Z : z_0 \dots z_N \in \{0, 1\}$ be class indicators

$$z_i \sim Bernoulli(\pi)$$
$$x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$$

- Bernoulli: coin flip with pr of heads=π



- Now there are two μ and Σ
- One for /i/ and one for /a/
- z has white background: latent

Learning with latent variables

Conceptually, two approaches:

Marginalizing over the latent variables

$$\hat{\mu}, \hat{\Sigma}, \hat{\pi} = argmax_{\mu,\Sigma,\pi} \sum_z P(x|z;\mu,\Sigma)P(z|\pi)$$

(The actual likelihood function)

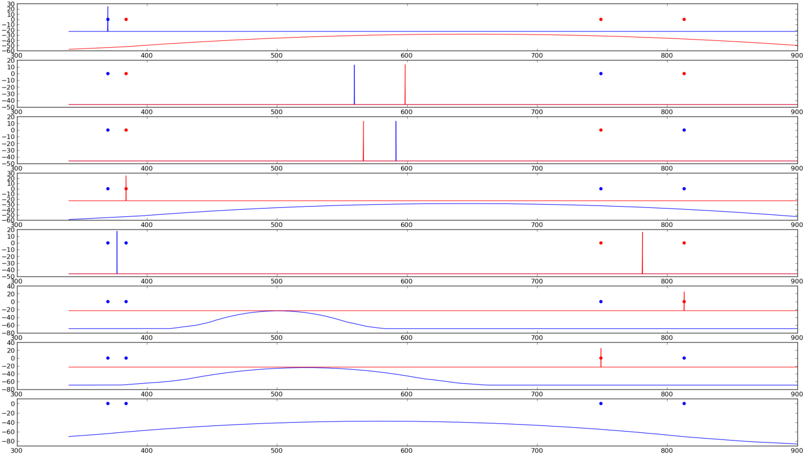
Maximizing the latent variables

$$\hat{\mu}, \hat{\Sigma}, \hat{\pi} = argmax_{\mu,\Sigma,\pi,z} P(x|z;\mu,\Sigma)P(z|\pi)$$

(Often fairly close to the likelihood)

The likelihood: intuition

Likelihoods as function of $\mu_1^{/i/}$ and $\mu_1^{/a/}$ under different assignments of z for four points



Difficulties

Since the likelihood doesn't have a fixed number of maxima, we can't solve for μ in closed form...

- Use iterative approaches (like gradient)

Expectation/Maximization (EM) algorithm

Most common iterative approach (Dempster+al '77)
(Approximately) a type of gradient method
Alternates between two phases:

- Improve z
- Improve μ, Σ, π

Insight 1: classification is easy

Given μ, Σ, π, it's easy to find the class probabilities for any sound *x*

$$P(x \in \text{/i/}) = \frac{\pi N(x|\mu_{/i/}, \Sigma_{/i/})}{\pi N(x|\mu_{/i/}, \Sigma_{/i/}) + (1-\pi) N(x|\mu_{/a/}, \Sigma_{/a/})}$$

- π: probability *z* for this *x* is 0
- $N(x|\mu_{/i/}, \Sigma_{/i/})$: probability of the sound fitting in the /i/ class
- Denominator: sound has to be either /i/ or /a/ (model assumption)
  - So $p(x \in /i/) + p(x \in /a/) = 1$

Insight 2: learning from labeled data is easy

As we saw at the beginning, computing μ, Σ is easy when there is only one vowel:

- So if we knew z, split data into /i/ and /a/, estimate each separately
- (Also trivial to estimate π, the probability of /i/ vs /a/: $\hat{\pi} = \frac{\#(/i/)}{n}$)

Basic (hard) EM

EM algorithm:

Set z_i at random
Alternate:

M-step (estimate)

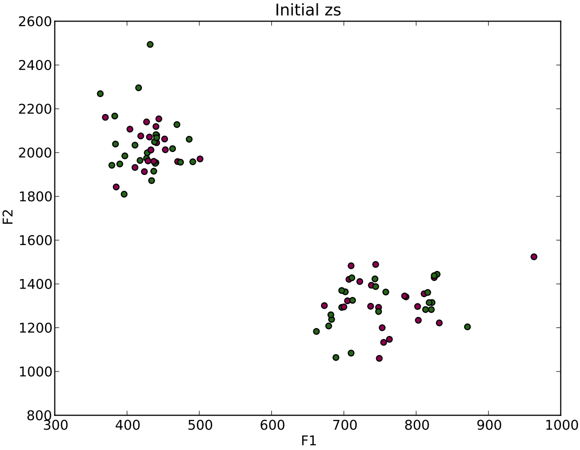
Split data into /i/ and /a/ according to current z
Estimate $\mu_{/i/}, \Sigma_{/i/}$ from /i/, $\mu_{/a/}, \Sigma_{/a/}$ from /a/, π from ratio of /i/ and /a/

E-step (classify)

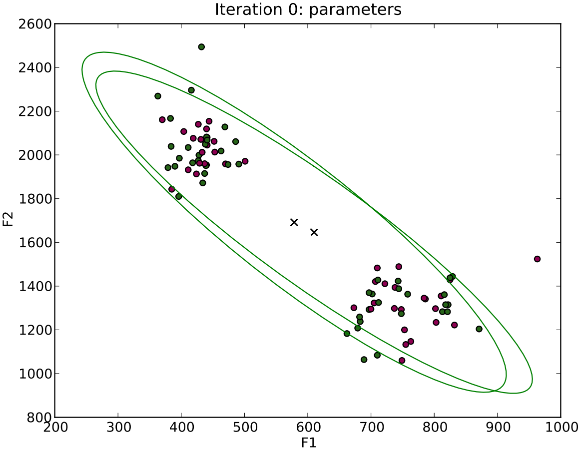
Using current parameters, compute $p(x \in /i/)$ for each x
For $x : p(x \in /i/) > .5$, set $z = 0$ (label as /i/)...
Otherwise label as /a/

Guarantee: each step improves likelihood until maximum reached

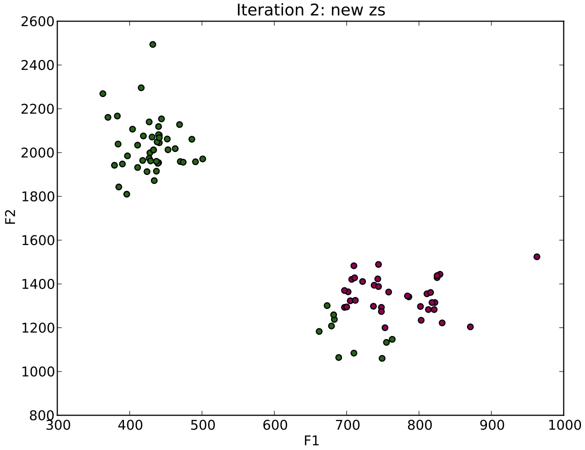
Random initialization



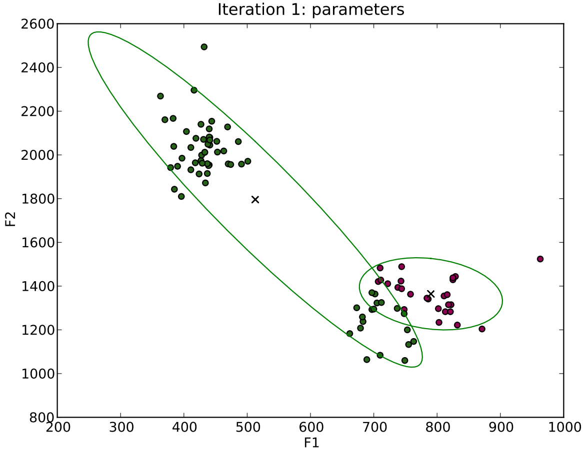
Parameter estimates



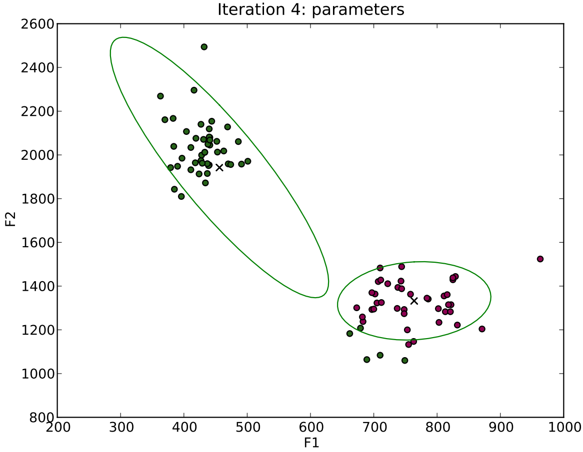
E-step 1 (new zs)



M-step 1 (new params)

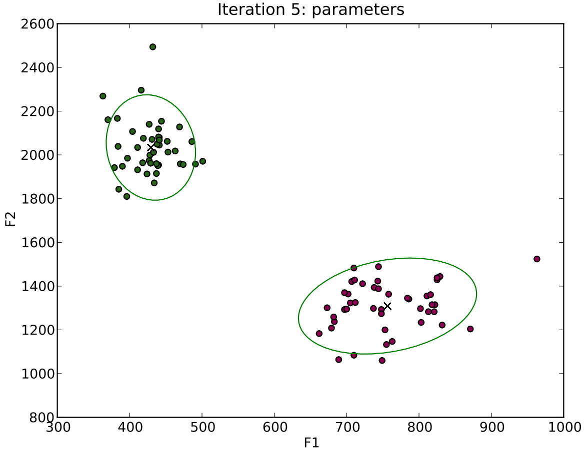


M-step 4



Iteration 4: parameters
2600
2400
2200
2000
1800
1600
1400
1200
1000
800
F2
200
300
400
500
600
700
800
900
1000
F1

M-step 5

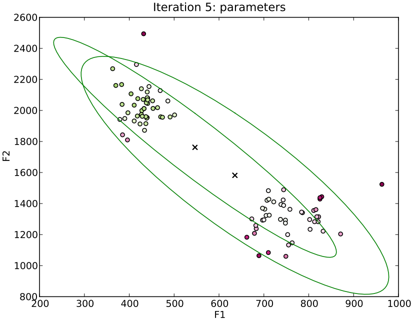


Iteration 5: parameters
2600
2400
2200
2000
1800
1600
1400
1200
1000
800
F2
300
400
500
600
700
800
900
1000
F1

Other things you might see in the wild

Soft (standard) EM

E-step: compute $p(x \in /i/)$... but don't assign to either class
M-step: compute **expected value** of parameters using distribution from E-step
(The standard EM algorithm)



Explicit gradient-based methods

- Require you to compute derivatives of the likelihood
- Simplest algorithm: add η times gradient to params
  - This can be very slow, though...
- Better algorithms exist (L-BFGS, OWL-QN, etc)
  - Often use approximations to 2nd derivative to decide step size
- A variety of off-the-shelf packages for doing this
- Incl. builtins in Matlab, R, etc

Batch vs. incremental

As presented here, each E-step (or computation of the gradient) iterates over *all* the data

- This is slow and cognitively implausible...
- *Incremental* variants exist which read a few datapoints at a time
- These few datapoints can be used to compute an *approximate* parameter update or gradient
- *Stochastic* gradient descent is one variant

Stochastic gradient

The value of the gradient itself at a point is a random variable

- Can be estimated from one or a small number of training exes
- Leads to fast online algorithms
  - Similar to perceptron
- Can be unstable though... must tune learning rate

EM and related methods

- Learn parameters for generative models with hidden variables
- Start with a (bad) initial model and gradually improve it
- Generally easy to implement
- Can be somewhat slow to run, but generally practical for real data

Problems: local maxima

Local maxima of the likelihood

As shown, the likelihood may have multiple maxima...

- EM/gradient always improve likelihood until convergence
  - Find a maximum (or saddle point)
- ...this doesn't mean they find the *global* maximum

Especially annoying when a model allows conflicting analyses...

- EM solution often internally consistent, but bad
- Eventual solution depends on initialization
- Hand-designed initialization scheme...
- Or random, but repeat many times

yuwanttu

yu • want • tu

y • u • w • a • n • t • t • u

Problems: model selection and overfitting

Model selection

Comparing two models which make different assumptions

- For instance, perhaps vowels are not perfect ellipses?
- Or perhaps there are really three vowels here?

Which model is better?

Maximum likelihood on its own is bad for model selection...

- Max likelihood: make training data as likely as possible
- Generalization: assigning probability outside the training set
- Models that generalize *less* have higher likelihood...
- More parameters: more specific model, less generalization

Model selection

Max likelihood chooses less general models (bad!)

- Means we can't use EM to learn models with varying levels of complexity
- (Like different numbers of vowels...)

For instance, learn lexicon from:

juwant, jukæn, ðejwant

Actual solution (lexicon is *ju, want, kæn, ðej*) generalizes to *ðejkæn*

Max-likelihood lexicon is: *juwant, jukæn, ðejwant*

Model selection techniques

Frequentist hypothesis testing

For each (more or less complex) model, run max-likelihood
Use hypothesis test to evaluate simpler vs more complex model
(ie, fit mixture of gaussians with 1, 2, 3... vowel classes)

Bayesian information criterion

Penalizes likelihood by number of parameters
Also generally used by running max-likelihood many times

Bayesian models

Don't require rerunning max-likelihood
Make explicit assumptions about what kind of complexity is likely
Next lecture!

Examples

Examples (python) online at:
http://www.ling.ohio-
state.edu/ melsner/course/stat-acq/em.tgz