Model diagnostics and information criteria

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

October 6, 2021

Outline

Today:

- Information criteria
- Approximate leave-one-out cross-validation (PSIS-LOO)
- Intro to multilevel models

Information criteria for scoring models

Kullback-Leibler divergence

Kullback-Leibler (KL) divergence:

$$D_{KL}(p, q) = \sum p_i (\log_2 p_i - \log_2 q_i)$$

Interpretation:

- p is the true outcome distribution
- q is the model predictive distribution
- KL divergence measures incorrectness, in some way

KL divergence for model comparison

We think of $D_{KL}(p, q)$ as measuring the distance from our model, q, to the truth, p.

Problem: we don't know p and never will!

But this isn't an obstacle for comparing models, because if we have two models q and r, then

$$D_{KL}(p, q) - D_{KL}(p, r) = \sum p_i(\log_2(r_i) - \log_2(q_i))$$

i.e. the $H(p)$ term drops out. We still don't know p_i, but we can estimate this from a sample of observations (because the observations are drawn from p_i); e.g. from a MCMC trace

Log score and deviance

Scoring models using log probabilities:

$$\log \text{score} \qquad S(q) = \sum_i \log(q_i)$$

where q_i is the probability (density) our model assigns to observation i

Deviance:

$$D = -2S(q) = -2 \sum \log(q_i)$$

In the Bayesian world, the posterior isn't one model, it's a distribution of models – so we should average:

$$\text{lppd}(y, \theta) = \sum_i \log \left(\frac{1}{S} \sum_s p(y_i|\theta_s) \right)$$

(log pointwise predictive density)

Out-of-sample prediction error

Problem: lppd only looks inside the sample

- Adding parameters nearly always improves fit within the sample
- Eventually, adding parameters reduces accuracy out of the sample (overfitting)
- How can we predict out-of-sample prediction accuracy?
    - Cross-validation
    - Information criteria

Overfitting in action

To demonstrate overfitting, we'll consider a few models fit to fake data.

True data-generating process:

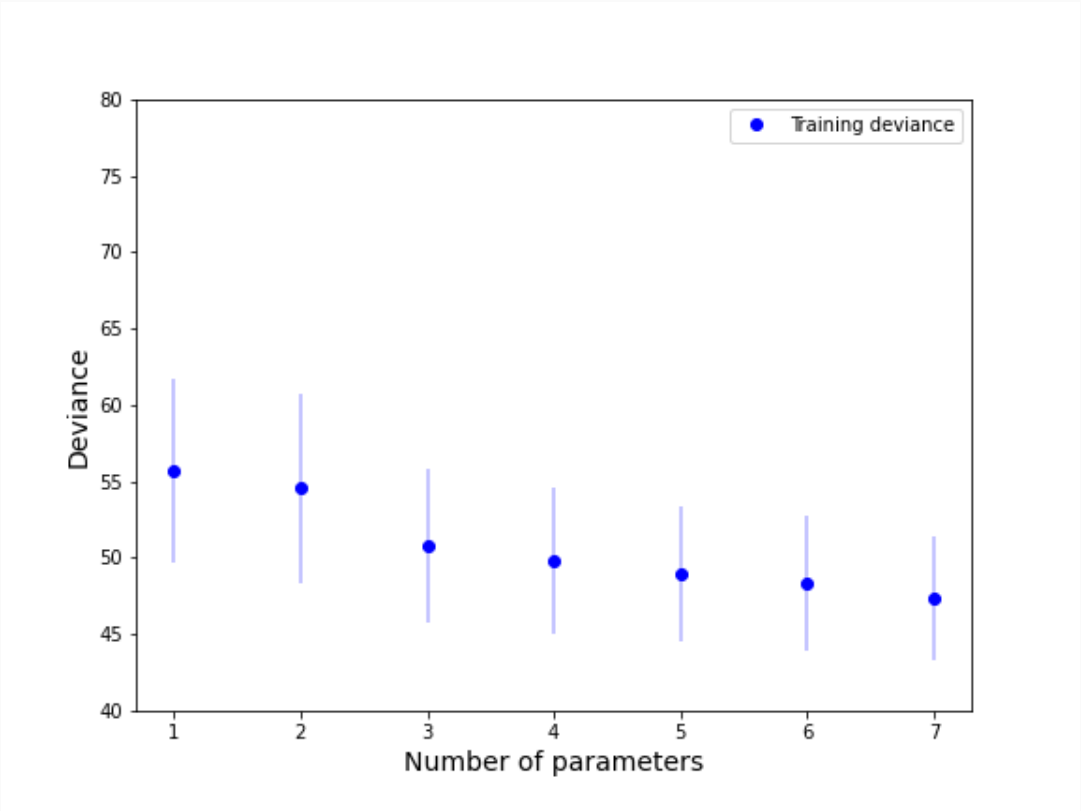
$$y_i \sim \text{Normal}(\mu_i, 1)$$
$$\mu_i = 0.15x_1 - 0.40x_2$$

We'll fit models with the same likelihood and

$$\mu_i = \alpha$$
$$\mu_i = \alpha + \beta_1 x_1$$
$$\mu_i = \alpha + \beta_1 x_1 + \beta_2 x_2$$
$$\mu_i = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$
$$\dots$$

Overfitting in action

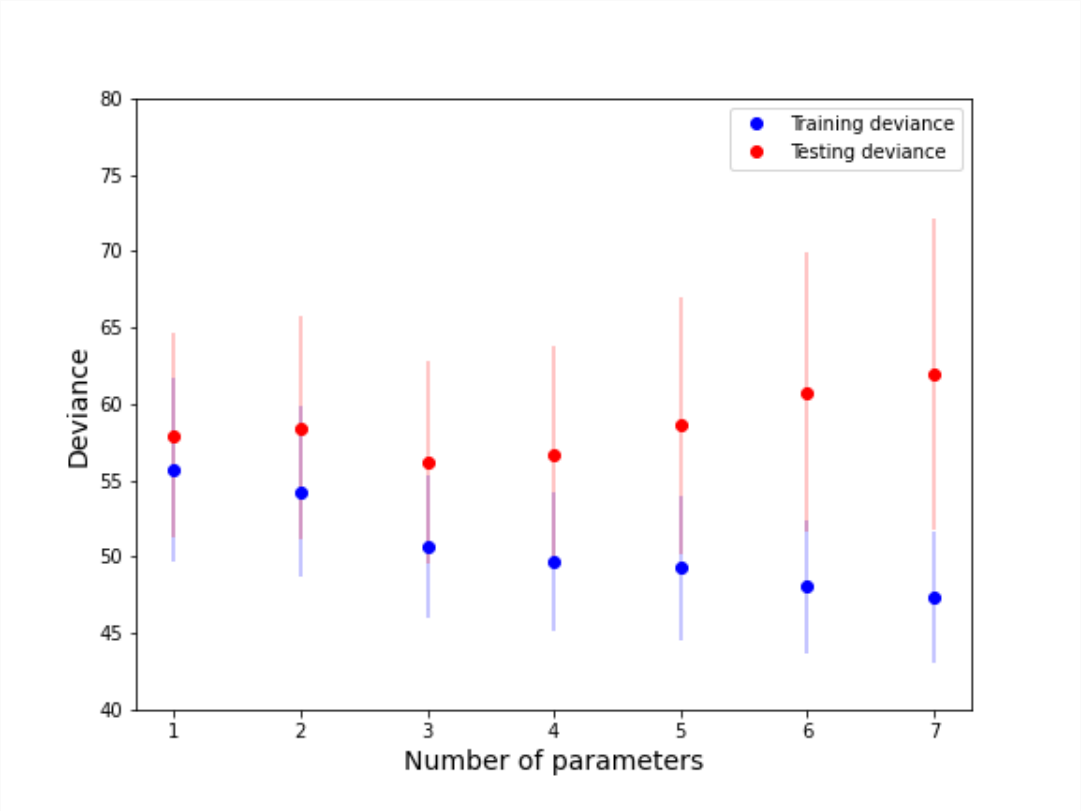
On the training set:



Remember: past 3 parameters, the predictors have no relationship to y in the true data generating process

Overfitting in action

Add in the testing set:



As expected, the additional parameters just make matters worse.

Goals

Since we use lppd to estimate the fit of our model, our goal with all of these tools is to estimate what our lppd will be on new data.

In other words, ultimately we are trying to estimate some form of:

$$\text{elpd} = \mathbb{E}(\log p(\tilde{y}|y))$$

the expected log predictive density of a new data point.

First approach: Akaike information criterion (AIC).

Akaike information criterion (AIC)

AIC: named for Akaike (but he called it "an information criterion"). Attempts to estimate the out-of-sample deviance.

Assuming a point estimate $\hat{\theta}$ for model parameters, calculate the log score/deviance and apply a penalty to correct for overfitting:

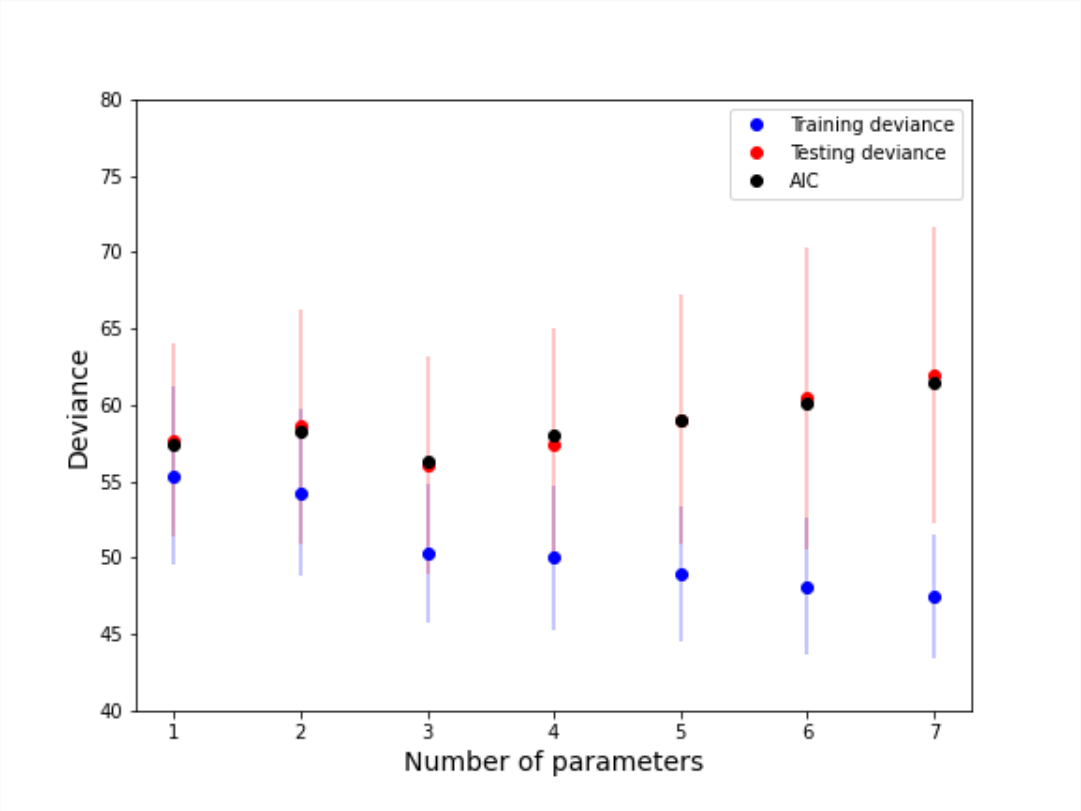
$$AIC = D_{\text{train}}|\hat{\theta} + 2k$$

Here $D_{\text{train}}|\hat{\theta}$ is the log predictive density, evaluated at the maximum likelihood estimate, multiplied by -2.

k is the number of parameters. Assumes Gaussian posterior. Where it comes from: Taylor expansion of the log posterior around the posterior mode.

Overfitting in action

Adding the AIC:



We see that for this model, the AIC is a pretty good estimate of the out-of-sample deviance. So, it is reasonable to choose the model with the smallest AIC.

Widely applicable information criterion

Introduced by Watanabe (2010); a more Bayesian generalization of the AIC; no requirements about the shape of the posterior.

$$WAIC = -2(\text{lppd}(y, \theta) - \sum_i \text{var} \log(p(y_i|\theta)))$$

lppd replaces the training deviance. The overfitting penalty is calculated by taking the log pointwise density and computing the variance over values of the parameters θ.

Reduces to AIC in the special case where AIC is exact: models with flat priors and Gaussian posterior, e.g. ordinary multiple regression

Leave-one-out cross-validation

LOO cross-validation

Idea behind cross-validation:

- Hold out some of your data for evaluation
- Fit the model on the remaining data
- Evaluate the model by estimating lppd on the held-out data
- Repeat, with different partitionings

Types of cross-validation

k-fold CV:

- Partition the data set into k equal subsets
- Each subset gets a turn as the hold-out set
- Problem: dependent on the (arbitrary) partition

Leave-one-out CV:

- Each observation gets a turn as the hold-out set
- Exhaustive: no arbitrary choices involved in choosing hold-out sets
- Problem: many re-fits required

Importance sampling

Importance sampling:

- Method related to rejection sampling and Metropolis
- Goal: calculate an average of a quantity h over some probability distribution, when we can only sample from an approximation to that distribution
    - Our case: want to calculate expected log score on ith observation over $p(\theta|y_{-i})$, the posterior with that observation dropped
    - But we don't want to calculate every one of those posteriors, so we use the full $p(\theta|y)$ as an approximation

Importance sampling in general

Idea: we want to calculate the average of $h(\theta)$, where θ follows a probability distribution $p(\theta)$. If we had a sample $\{\theta_s\}$ from $p(\theta)$, we could just evaluate h and average:

$$E[h(\theta)] \approx \frac{1}{S} \sum_s h(\theta_s)$$

But suppose our sample $\{\theta_s\}$ comes instead from an approximate distribution $q(\theta)$. Then the above doesn't work; but, if we re-weight each term in the sum we can recover a good approximation.

Importance sampling in general

Define the *importance ratio* or *importance weight*:

$$w(\theta_s) = \frac{p(\theta_s)}{q(\theta_s)}$$

then,

$$E[h(\theta)] \approx \frac{\sum_s h(\theta_s) w(\theta_s)}{\sum_s w(\theta_s)}$$

i.e. just a weighted average, weighted by the importance ratios.

Idea: samples with a high $q(\theta_s)$ are more "important" to the distribution we are trying to target

Importance sampling for LOO-CV

In LOO-CV we are trying to estimate $\log p(y_i|y_{-i})$, where y_{-i} denotes the set of observed y values without y_i.

We calculate importance weights for each sample θ_s:

$$w(\theta_s) = \frac{1}{p(y_i|\theta_s)}$$

and get an estimate for the lppd for that observation:

$$\text{lppd} \approx \frac{\sum_s p(y_i|\theta_s)w(\theta_s)}{\sum_s w(\theta_s)}$$

Then our estimated out-of-sample log score is the sum of the above over observations i.

Smoothing

Importance weights can be unreliable:

- If one or a few importance weights are much larger than the others, they can dominate the estimate and make it inaccurate
- So, we want to "smooth" the estimate

Under some standard conditions, largest importance weights should follow a *generalized Pareto distribution*:

$$p(w|u,\sigma,k) = \sigma^{-1}(1 + k(r-u)\sigma^{-1})^{-\frac{1}{k}-1}$$

- u, σ – location and scale
- k – shape; controls the weight of the tail

Smoothing

In Pareto-smoothed importance sampling:

- the largest 20% of importance weights are used to fit the parameters for a generalized Pareto distribution
- those weights are then replaced by quantiles from the same distribution

Still doesn't work very well if the Pareto distribution's shape is bad:

- $k > 0.5$: distribution has infinite variance
- In practice, still usually ok if $k < 0.7$; for larger k, can't necessarily trust approximation

Detecting high-influence points

When are the importance ratios really big?

$$w_(\theta_s) = \frac{1}{p(y_i|\theta_s)}$$

When the posterior distribution assigns low probability to an observation.

If k is large for the Pareto distribution fitted for y_i, that indicates that these weights are really big – suggesting that the model cannot accomodate that point well.

- use `az.loo(trace, pointwise = True)`

Let's see it...

Applying WAIC and LOO-CV

We now have two numerical tools for estimating out-of-sample deviance: WAIC and LOO-CV.

- In ordinary linear models, LOO-CV and WAIC perform pretty similarly. LOO-CV has higher variance, WAIC higher bias as estimates of the KL divergence.
- In practice differences are usually small; best practice is to compute both. If there are large differences, this may indicate that one or both are unreliable
- Computational problems with both can sometimes be resolved by using a more robust model

Intro to multilevel models

How much traffic is bicycle traffic?

In the book there is a data set with observations of bicycle traffic on a number of streets.

- Data collected by standing at the roadside for some amount of time
- Count number of bicycles and number of non-bicycle vehicles
- Parameter of interest: proportion of traffic that is bicycles

Fully pooled model

A fully pooled model:

$$y_j \sim \text{Binomial}(\theta, n_j)$$
$$\theta \sim \text{Beta}(\alpha_0, \beta_0)$$

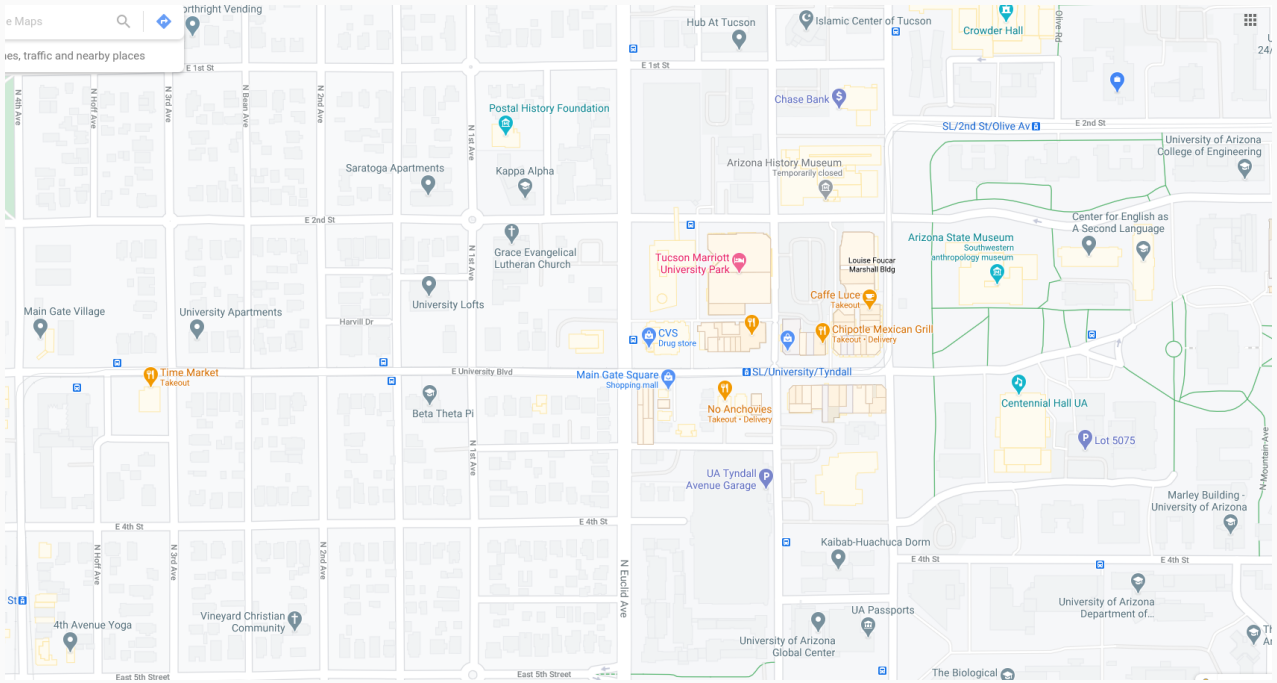
for fixed α_0, β_0.

- Choosing $\alpha_0 = 1, \beta_0 = 1$ gives a completely noninformative (flat) prior
- Weakly informative prior also reasonable, e.g. $\alpha_0 = 1, \beta_0 = 3$ for prior mean of 25% bicycle traffic

Why not pool?

This model we wrote in the previous section treats all streets as the same; each street's observation is an observation of the same underlying proportion.

But this isn't particularly reasonable:



Why multilevel modeling?

Imagine you're collecting data for this.

- You go out to University Blvd/1st Ave and count cars and bicycles for 30 minutes.
- Is this count going to be representative of the level of bicycle traffic in Tucson?

Why multilevel modeling?

Imagine you're collecting data for this.

- You go out to University Blvd/1st Ave and count cars and bicycles for 30 minutes.
- Is this count going to be representative of the level of bicycle traffic in Tucson?
    - No; bike traffic is much higher on University than other streets

Why a hierarchical model?

As an alternative, we could treat each street as an independent entity:

Why a hierarchical model?

As an alternative, we could treat each street as an independent entity:

$$y_j \sim \text{Binomial}(\theta_j, n_j)$$
$$\theta_j \sim \text{Beta}(\alpha_0, \beta_0)$$

for fixed α_0, β_0.

- Exactly like the previous model, except we now have 10 independent θ_js for the 10 streets
- Same considerations for choice of prior

Call this the separate-effects model.

Comparing the two models

Let's jump over to a notebook and compare the results of these two models.

Why a hierarchical model?

Choosing between the two models: classically, do an analysis of variance

- Compare variance within groups (streets) to variance between streets
- Test against the null hypothesis that all streets are the same
- If we reject the null, take the separate-effects model
- If we don't take the pooled model

Problem: false dichotomy!

Why a hierarchical model?

In reality, it is most plausible that both of the following are true:

- The streets are not identical; some of the streets are more popular with cyclists
- Observations of one street can inform our knowledge of the others
- The high bicycle traffic we observe on University is:
  - partly a reflection of that street's individual geographical properties
  - partly a reflection of the city's relatively high bicycle friendliness

So: neither side of this dichotomy is preferable.

Variability at two levels

There is variability at two levels in this problem:

- Different cities have, overall, more or less bicycle traffic
- Within a given city, different streets have more or less bicycle traffic

So, observations of one street should inform our estimates of other streets, *without* making the same estimate for every street

The hierarchical model

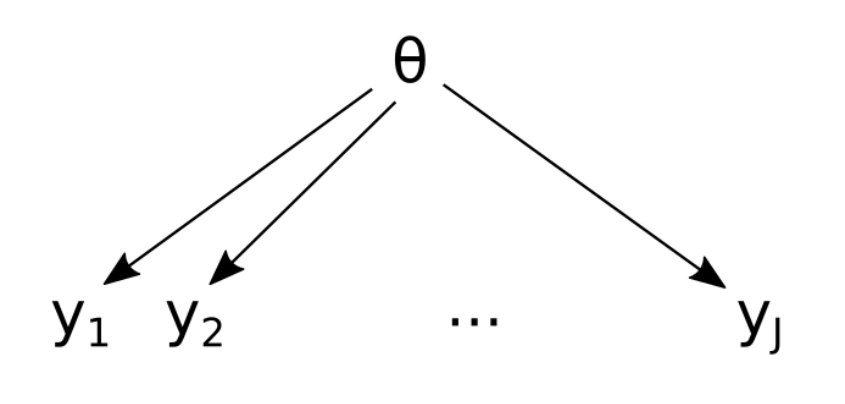
With a Bayesian approach, we can find a compromise.

- We have a θ for each street
- However, instead of being fully independent, each θ is drawn from a common probability distribution
- This probability distribution, a *hyperprior*, depends on *hyperparameters* which we estimate from the data

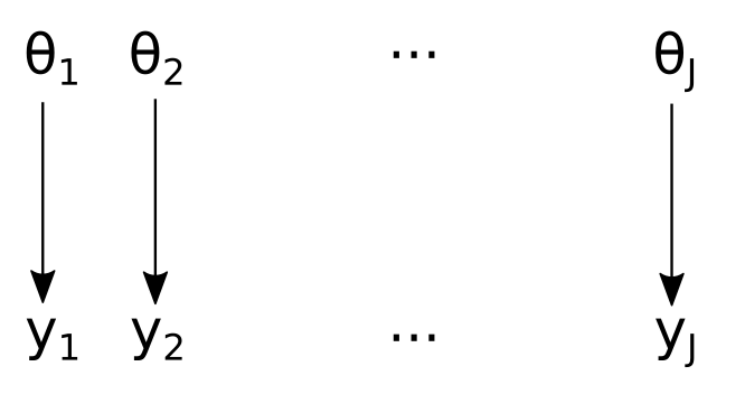
(note: slightly different sense of the term *hyperparameter* from its common use in ML)

Examining this graphically

Pooled model:

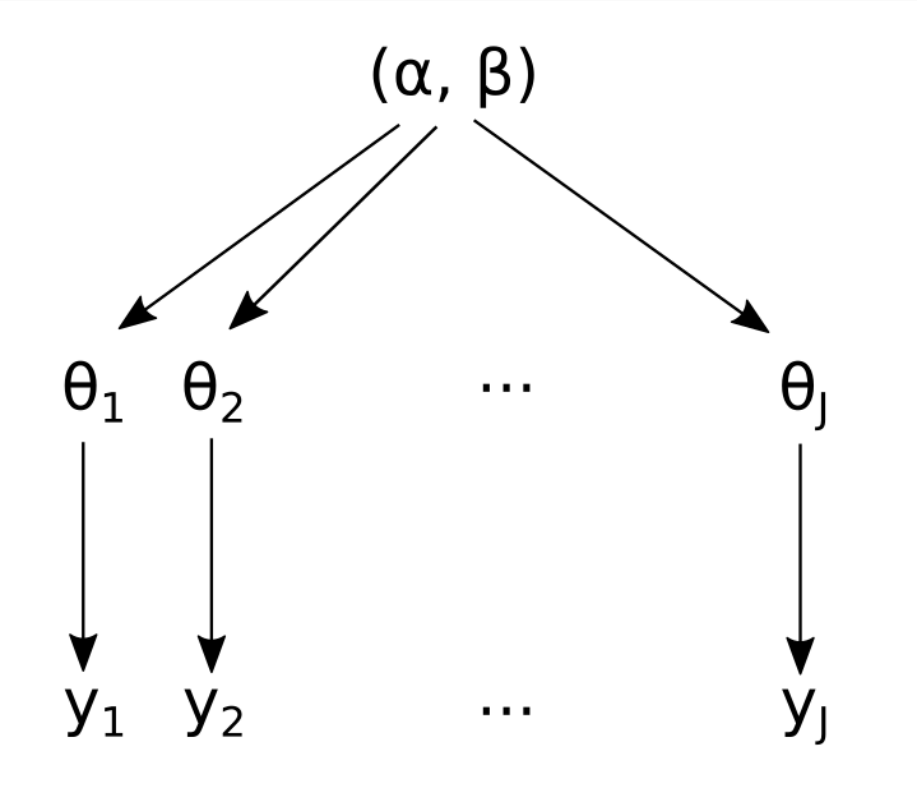


Separate model:



Examining this graphically

Hierarchical model combines the features of these two:



Setting up the model

This is conceptually only a slight difference from our previous model:

$$
\begin{aligned}
y_j &\sim \text{Binomial}(\theta_j, n_j) \\
\theta_j &\sim \text{Beta}(\alpha, \beta) \\
p(\alpha, \beta) &\propto \text{???}
\end{aligned}
$$

Choosing a hyperprior

We need a prior distribution for α, β; this can be a tricky part of this sort of modeling, because the interpretation of these parameters is not so simple compared to θ_j.

Starting with the idea that α and β can represent "pseudocounts", parameterize in terms of

- $\mu = \frac{\alpha}{\alpha+\beta}$ – prior expectation
- $\eta = \alpha + \beta$ – prior "sample size" (think of this like a precision)

Choosing a hyperprior

Prior for μ:

- Need $0 < \mu < 1$, so we'll choose a Beta
- Informative version: cars outnumber bikes, so try $\mathrm{Beta}(1, 3)$

Prior for η:

- $\eta > 0$, so choose something with range $(0, \infty)$
- Fairly spread out, but put more prior mass near 0; try a half-Cauchy

Setting up the model

This is conceptually only a slight difference from our previous model:

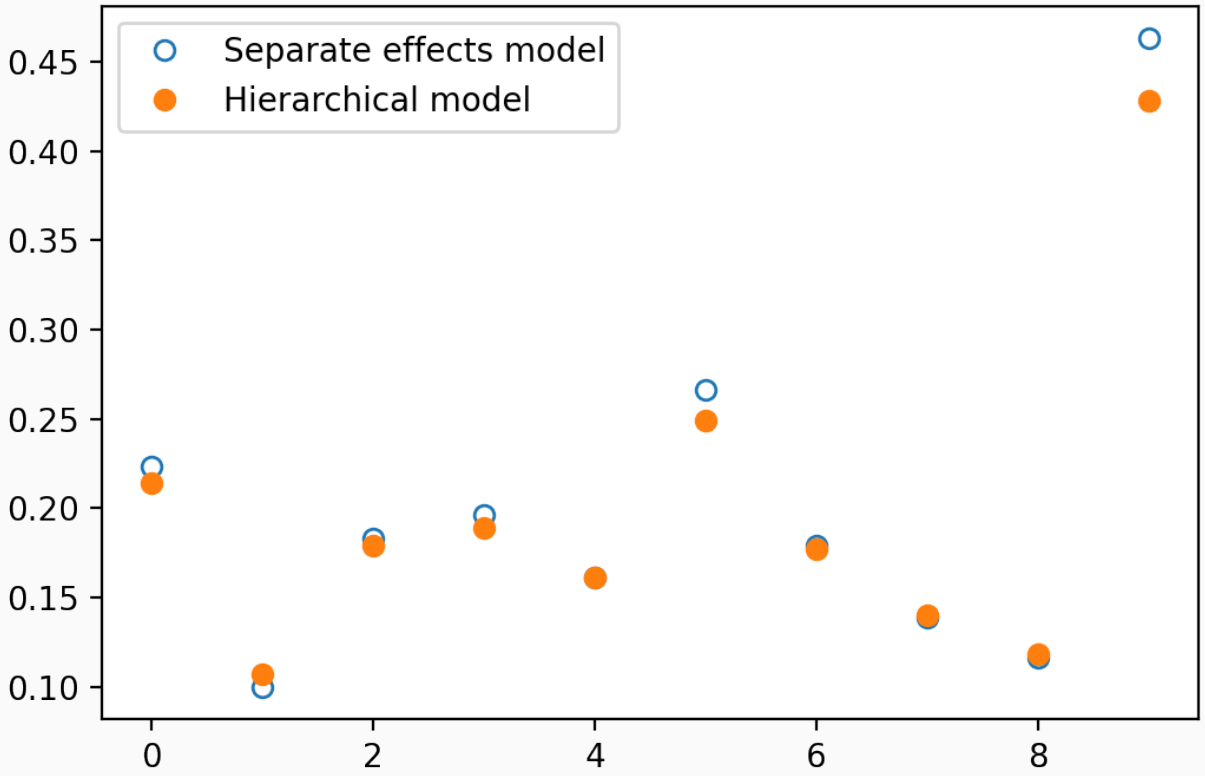
$$
\begin{aligned}
y_j &\sim \text{Binomial}(\theta_j, n_j) \\
\theta_j &\sim \text{Beta}(\alpha, \beta) \\
\mu &:= \frac{\alpha}{\alpha + \beta} \\
\eta &:= \alpha + \beta \\
p(\mu) &\sim \text{Beta}(1, 3) \\
p(\eta) &\sim \text{HalfCauchy}(1)
\end{aligned}
$$

Let's take a look at some code

Comparison

What is the difference in the results?

Let's compare point estimates:



Shrinkage and regularization

The shrinkage effect we see is a form of regularization:

- Most extreme observations "shrunk" toward a central value
- Amount of shrinkage tuned to relative sample size

Difference: we learned the strength of regularization from the data

Underfitting and overfitting

Another way to think about this, in terms of underfitting and overfitting:

- The pooled model: strong underfitting
- The separate-effects model: strong overfitting
- Hierarchical model: adaptive regularization

With enough observations the seperate effects model will estimate each street similarly to the hierarchical model.

Independence and exchangeability

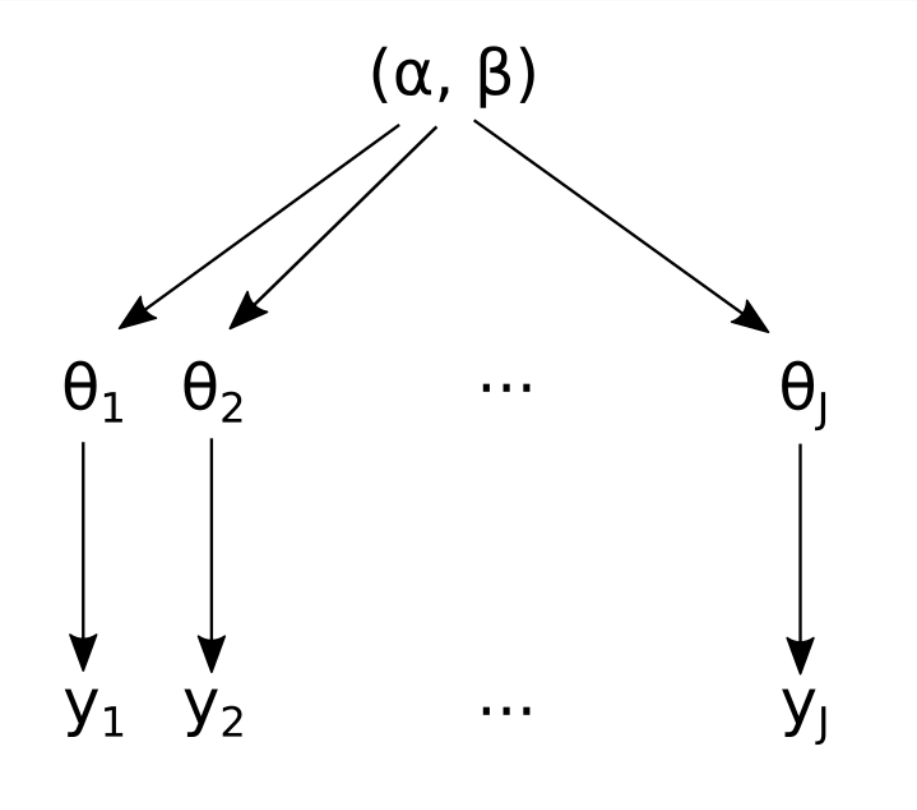
Independence of the θ_js

It's worth taking a moment to consider the independence properties of the parameters θ_j:

- In the hierarchical model, θ_js are not independent (they're independent in the pooled model)
- However, they satisfy two weaker properties:
  - conditional independence
  - exchangeability

Conditional independence

The θ_js are not independent, but *given fixed* α, β, they are:



Exchangeability

A closely related concept is *exchangeability*, which justifies the use of the hierarchical model:

- Observations are *exchangeable* if the joint probability distribution is invariant to permutations of the index
- Roughly: we would have the same model if we relabeled the $y_1, y_2, \ldots$
- Exchangeability is also evident in the directed graph model

Levels of exchangeability

The full data set contains observations from a total of 58 streets:

- small residential streets, medium streets, and busy arterial streets
- streets with or without bike lanes

Evidently, if we label the streets $y_1, \ldots, y_{58}$, they are not exchangeable.

But within the traffic/lane groups, the streets can be treated as exchangeable:

- Hierarchical model with several "levels"

Summary

Today:

- information theory / entropy
- Information criteria: AIC and WAIC
- Pareto-smoothed approximate LOO-CV
- Intro to multilevel models

Next time:

- More on multilevel models