Computational Statistics & Probability

Problem Set 5 - Multilevel Models
Due: 23:59:59 13.dec.2022

Fall 2022

Instructions

Assignments must be submitted through Canvas. See the course Canvas page for policies covering collaboration, acceptable file formats (`.Rmd` & `.pdf`), and late submissions. Completed assignments must include executable code (`.Rmd`) and a corresponding knitted markdown file (`pdf`). An R Markdown cheat sheet is available.

Note: *You must have* `rstan` *installed to complete this assignment.*

1. Varying Slopes and Effective Parameters

When is it possible for a varying slopes model to have fewer effective parameters (as estimated by WAIC or PSIS) than the corresponding model with fixed slopes? Explain your answer.

```
# A varying effects model can have fewer effective parameters than a corresponding
# fixed effect model when there is very little variation among clusters.  In such
# circumstances, a varying effects model will induce strong shrinkage across clusters,
# thereby constraining the variation of the individual parameters. So although by
# design the varying effects model must have more actual parameters in the posterior
# distribution than a comparable fixed effect model, the varying effects model can
# be less flexible in fitting the data because it adaptively regularizes. Both
# varying slopes and varying intercepts work the same way: there is nothing special
# about a varying slopes model in this respect.
```

```
# -------
# Here is a simulation to demonstrate this phenomenon. You do not need to include
# a computational example. This is just provided for you to understand the conditions
# of a toy example to produce this effect.

# First, let's simulate data that are not very different from one another.  Clusters
# in this example are individuals and the observations are simulated test scores.
# Each individual has some ability to influence his test score. The aim is to recover
# this "ability to influence one's test score" through estimation.

N_individuals <- 100
N_scores_per_individual <- 10
# simulate abilities
ability <- rnorm(N_individuals,0,0.1)

# simulate observed test scores
# sigma here large relative to sigma of ability
N <- N_scores_per_individual * N_individuals
id <- rep(1:N_individuals,each=N_scores_per_individual)
score <- round( rnorm(N,ability[id],1) , 2 )
```

```r
# put observable variables in a data frame
df <- data.frame(
  id = id,
  score = score )


# Next, we fit a fixed effect model `m_fixed` which has an intercept for each of
# the 100 individuals
m_fixed <- ulam(
  alist(
    score ~ dnorm(mu,sigma),
    mu <- a_id[id],
    a_id[id] ~ dnorm(0,10),
    sigma ~ dcauchy(0,1)
  ),
  data=df , chains=2, log_lik = TRUE )


# Now we fit a varying effects "partial pooling" model with an adaptive prior
m_vary <- ulam(
  alist(
    score ~ dnorm(mu,sigma),
    mu <- a + z_id[id]*sigma_id,
    z_id[id] ~ dnorm(0,1),
    a ~ dnorm(0,10),
    sigma ~ dcauchy(0,1),
    sigma_id ~ dcauchy(0,1)
  ),
  constraints=list(sigma_id="lower=0"),
  data=df, chains=4 , cores=4, log_lik = TRUE )
```

```r
compare( m_fixed , m_vary )
```

```
##             WAIC       SE    dWAIC      dSE     pWAIC       weight
## m_vary  2816.786 43.80603   0.0000       NA  8.934401 1.000000e+00
## m_fixed 2917.029 44.44441 100.2426 18.65869 95.015907 1.708467e-22
```

```r
# The number of effective parameters for the varying effects model is approximately
# 10, versus 96 effective parameters for the fixed effect model. That is a very
# large difference. (Results may vary slightly from simulation to simulation.)

# To understand why `m_vary` has comparatively so few effective parameters,
# compare the prior mean for `sigma` (1) to the posterior `sigma_id:

precis( m_vary)
```

```
## 100 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                  mean         sd        5.5%      94.5%     n_eff     Rhat4
## a        -0.007699877 0.03030798 -0.05603304 0.04020033 3319.0044 0.9984066
## sigma     0.985016088 0.02295857  0.94986078 1.02187220 2620.1815 0.9998477
## sigma_id  0.072509621 0.04814704  0.00840133 0.15611313  610.1541 1.0068318
```

```r
# The posterior mean is 0.14, which induces a lot of shrinkage across clusters.
# In short, those 100 parameters are only as flexible as 10 parameters.  The
```

```
# adaptive regularization of `sigma_id`-- which is learned from the data -- reduces
# the flexibility of model `m_vary`.

# TAKE AWAY: The number of dimensions of a model is NOT a relevant measure of complexity
# in terms of judging overfitting risk.  Just because one model makes more assumptions
# than another (i.e., has more parameters or more distributions) does not always mean
# that the risk of overfitting increases.
```

2. Gaussian Process Regression

a) Go to section §14.5 in the textbook and compare the Gaussian process model of Oceanic tools, `m14.8`, to all the models fit to the same data in §11.2 by WAIC. This first step asks you to just produce the table.

```
library(rethinking)
data(Kline)
d <- Kline
d$P <- scale( log(d$population) )
d$contact_id <- ifelse( d$contact=="high" , 2 , 1 )

set.seed(49)

dat <- list(
  T = d$total_tools ,
  P = d$P ,
  cid = d$contact_id )

# intercept only
m11.9 <- ulam(
  alist(
    T ~ dpois( lambda ) ,
    log(lambda) <- a ,
    a ~ dnorm( 3, 0.5)
  ), data=dat, chains=4, log_lik=TRUE )

# interaction model
m11.10 <- ulam(
  alist(
    T ~ dpois( lambda ) ,
    log(lambda) <- a[cid] + b[cid]*P ,
    a[cid] ~ dnorm( 3, 0.5) ,
    b[cid] ~ dnorm( 0 , 0.2 )
  ), data=dat, chains=4, log_lik=TRUE )

# dynamic equations model

# take population off log-scale
dat2 <- list(
  T = d$total_tools ,
  P = d$population ,
  cid = d$contact_id )

m11.11 <- ulam(
  alist(
    T ~ dpois( lambda ),
```

```
    lambda <- exp(a[cid])*P^b[cid]/g ,
    a[cid] ~ dnorm(1,1) ,
    b[cid] ~ dexp(1) ,
    g ~ dexp( 1 )
    ), data=dat2 , chains=4, log_lik=TRUE
)
```

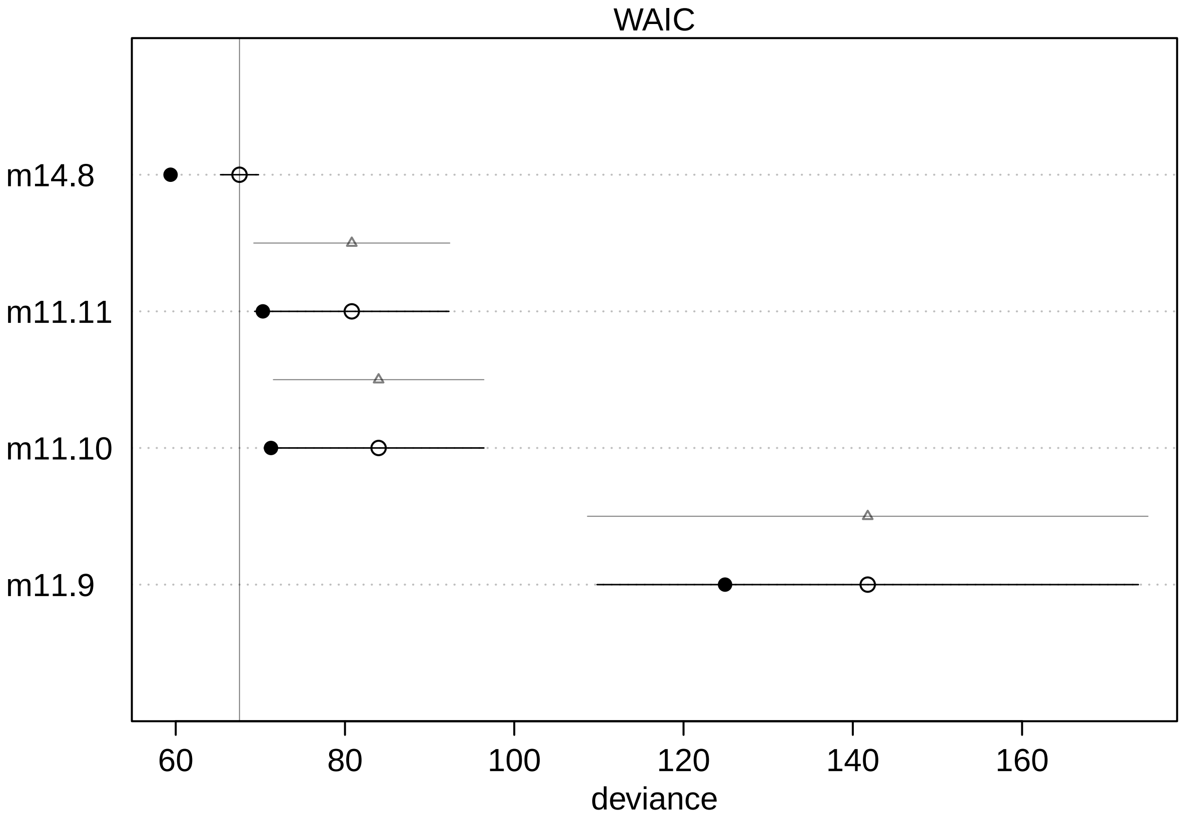
```
# Gaussian process model
data(islandsDistMatrix)
data(Kline2)
d2 <- Kline2
d2$society <- 1:10 # index observations
set.seed(49)

dat2_list <- list(
  T = d2$total_tools ,
  P = d2$population ,
  society = d2$society,
  Dmat= islandsDistMatrix)

m14.8 <- ulam(
  alist(
    T ~ dpois(lambda) ,
    lambda <- (a*P^b/g) * exp(k[society]),
    vector[10]:k ~ multi_normal( 0 , SIGMA ),
    matrix[10,10]:SIGMA <- cov_GPL2( Dmat , etasq , rhosq , 0.01) ,
    c(a,b,g) ~ dexp( 1 ),
    etasq ~ dexp( 2 ),
    rhosq ~ dexp( 0.5 )
  ), data=dat2_list, chains=4, cores=4, iter=2000, log_lik = TRUE)
```

```
compare( m11.9, m11.10 , m11.11, m14.8 )
```

```
##              WAIC        SE    dWAIC      dSE    pWAIC       weight
## m14.8    67.53386  2.243875  0.00000       NA 4.071870 9.984181e-01
## m11.11   80.80199 11.477385 13.26813 11.58226 5.252341 1.312727e-03
## m11.10   83.97129 12.441891 16.43743 12.43988 6.358355 2.691345e-04
## m11.9   141.76261 31.980671 74.22875 33.11580 8.429251 7.598786e-17
```

b) What can you learn about your models through their WAIC scores? In your analysis, pay special attention to the effective number of parameters estimated by WAIC.

```
# First, let's plot the comparison of the four models by WAIC.
plot(compare( m11.9, m11.10 , m11.11, m14.8 ))
```




```
# The Akaike weight for the GP regression, m14.8, is almost 1.  So, at first
# glance, the difference between the GP model and the three original models
# is very large.  However, as noted in the textbook, the Akaike weight is not
# sufficient for model comparison, since weight does not reflect standard errors.

# When we turn to the standard error of the difference between the original dynamic
# equation model m11.11 and the GP model m14.8, we see that the `dWAIC` (13.1) is
# almost identical to the difference in standard error `dSE` (11.57), which is
# indistinguishable (by WAIC) from the interaction model m11.10.  The intercept
# only model, m11.09, is clearly worse than others in the comparison class.

# Despite the near equivalence between the GP model m14.8 and the two GLMs, m11.11
# and m11.10, we can learn something from the effective number of parameters,
# `pWAIC`. Gaussian processes have complicated dependency structures with no
# obvious formula to summarize model complexity. Effective number of parameters,
# as measured by `pWAIC`, is a helpful tool in this context.
```

model	parameters	pWAIC
m14.8	5	4.1
m11.11	3	5.3
m11.10	2	6.4
m11.9	1	8.4

```
# The effective number of parameters is a correction for overfitting, where
# 0 is fully constrainted or all information comes from your prior, and
# positive values are less constrained.  The pointwise pWAIC score for
# m14.8 can be viewed by the convience function `WAIC` in the rethinking
```

```r
# package,

WAIC(m14.8, pointwise=TRUE)
```

```
##        WAIC      lppd   penalty  std_err
## 1  5.904238 -2.572621 0.3794981 2.243875
## 2  6.061882 -2.769374 0.2615668 2.243875
## 3  5.883620 -2.754165 0.1876446 2.243875
## 4  7.831201 -3.225407 0.6901935 2.243875
## 5  6.457843 -2.935930 0.2929912 2.243875
## 6  6.591864 -2.775505 0.5204274 2.243875
## 7  7.011320 -3.089439 0.4162211 2.243875
## 8  6.465891 -2.891075 0.3418711 2.243875
## 9  7.671792 -3.297166 0.5387300 2.243875
## 10 7.654207 -3.384378 0.4427258 2.243875
```

```r
sum(WAIC(m14.8, pointwise=TRUE)$penalty)
```

```
## [1] 4.07187
```

```r
# which indicates that the individual parameters are slightly more constrained than
# the corresponding Poisson GLM sharing the same dynamical systems model for lamba

WAIC(m11.11, pointwise=TRUE)
```

```
##         WAIC      lppd    penalty  std_err
## 1   5.795699 -2.641464 0.25638558 11.47739
## 2   6.520226 -2.940843 0.31926935 11.47739
## 3   5.444461 -2.662890 0.05934047 11.47739
## 4  10.800621 -4.234363 1.16594778 11.47739
## 5   5.924905 -2.882636 0.07981692 11.47739
## 6  17.761065 -7.033087 1.84744597 11.47739
## 7   5.849128 -2.889361 0.03520333 11.47739
## 8   5.860777 -2.836631 0.09375728 11.47739
## 9   9.206048 -3.645562 0.95746236 11.47739
## 10  7.639060 -3.381818 0.43771196 11.47739
```

```r
sum(WAIC(m11.11, pointwise=TRUE)$penalty)
```

```
## [1] 5.252341
```

```r
# which both are much more constrained than the intercept-only Poisson GLM, m11.9:
WAIC(m11.9, pointwise=TRUE)
```

```
##         WAIC       lppd    penalty  std_err
## 1  23.794554 -10.523093 1.37418364 31.98067
## 2  10.848509  -4.951993 0.47226216 31.98067
## 3   9.149164  -4.238458 0.33612415 31.98067
## 4   7.808975  -3.695706 0.20878123 31.98067
## 5   5.532520  -2.753457 0.01280354 31.98067
## 6  14.123133  -6.341264 0.72030272 31.98067
## 7   6.520974  -3.171866 0.08862124 31.98067
## 8   6.782165  -3.257101 0.13398150 31.98067
## 9  17.438720  -7.503937 1.21542335 31.98067
## 10 39.763894 -16.015180 3.86676760 31.98067
```

```r
sum(WAIC(m11.9, pointwise=TRUE)$penalty)
```

```
## [1] 8.429251
```

```r
# What is going on?  The Gaussian process model is applying addaptive regularization
# whereas the Poisson GLMs are not.  So, the GP regression model is *less* flexible
# than the Poisson GLMs and, through adaptive regularization, `m14.8` is the only
# model whose effective number of parameters is (roughly) the same as the actual
# number of parameters.  Put differently, the adaptive regularization is strong
# enough in the GP regresson model (m14.8) to not require any more than the offset
# of k dimensions of the parameter space to correct for overfitting.
```