Noam Ross GAMs notes

GAMs – generalized additive models. These are a category of nonlinear models that are useful for a lot of problems. We’re going to talk about what types of problem they’re good for.

mgcv package

Other stat ecologists who are good at GAMs – Gavin Simpson, Eric Pedersen, David Miller. Textbook Generalized Additive Models: An Introduction with R by Simon N. Wood.

Interpretability-complexity tradeoff:

linear models -> GAMs -> ML

Can use GAMs to:

Predict from complex, nonlin, maybe interact relationships

Understand and make inferences a/b those relationships

Control for those relationships

GAMs are good at prediction; comparable to random forests and SVMs. (Validation area under ROC curve – see github.com/klarsen )

**Theory behind GAMs**

Generalized – can handle many distribs of normal, binom, count, etc data

Additive: terms themselves aren’t linear, BUT just add terms together

In a GLM, the individual terms are linear

Polynomials are limited in what curves they can represent. We want a way to fit curves smoothly without overfitting. We’re gonna do this with splines.

GAMs are built up out of lots of little functions, called *basis functions*. Basis functions add together to get a spline.

Basis fcts can have 1, 2, or more dims; so you can create a fitted surface, not just a curve

Good for fitness landscapes?

Optimizing Wiggliness. GAM parameters:

Log(L) – lambda\*W

Lambda = smoothing parameter.

Likelihood or fit minus (smoothing parameter times wiggliness)

A good lambda fits your data well. A small lambda overfits. A too large lambda fails to capture the complexity.

Lambda selection is automated in mgcv!

**Fitting a GAM in R: not v diff from lin or GLM model**

lm(y – x1 + x2, data=data)

glm(y – x1 + x2, data=data, family=binomial)

gam(y – x1 + s(x2), #model fomula

data = data,

family = gaussian #or smthng stranger

method = “REML”) #method for picking lambda. Here, Relative Maximum Likelihood

REML isn’t the default but it *should* be. It’s usually what you want.

We can have more things in the GAM formula:

Y – x1 + #linear terms

S( #smooth terms

X2, #variable

Bs = “tp”, #the kind of basis function. Default is thin-plate spline.

#k=10 #how many basis fcts to use. You have a lot of leeway; just large enough to replicate complexity in #the model, smoothing fct will cut it down if it’s too big

…) #other stuff depending on your basis fct

GAM formula in 2D:

Y ~ s(x1) + s(x2) #Two additive smooths. Y is sum of spline of x1 and spline of x2

Y ~ s(x1, x2) #2D smooth/ interaction. Presumes both have same wiggliness

Y ~ te(x1, x2) #tensor spline. Uses two different lambdas. 2D smooth, two wigglinesses

Y ~ te(x1) + te(x2) + ti(x1, x2) #ti is tensor interaction.

**GAMs are great for spatial/ GIS data.**

Gives example of dolphin population modeling.

Gam(d ~ s(x, y) + s(depth), data=dolphin\_observations)

Get a predicted density, modeled as a 2D surface. Function of x y coordinates and depth.

The x and y can explain a lot of otherwise-unexplained variation.

Smoothing fcts for spatial data: use bs=””, the basis parameter, for these.

1. “Soap Films.” E.g. for a lake with a complex shape, or a U-shaped object where there are parts of it that are close by but *don’t* interact, so raw x y would mislead

Gam(d ~ s(x, y, bs=”so”, xt= #etc

1. Global-scale things. You want a spline mapped onto a sphere rather than simple x y
   1. Gam(y ~ s(latitude, longitude, bs=”sos”
2. Timeseries. Use Gaussian Process Smooths. Gam(y ~ s(time, bs=”gp”), data=bat\_antibodies)
   1. Gaussian processes are great for timeseries (and other situations with a lot of autocorrelation. )
3. Cyclic smooths. Good when there’s a seasonal, weekly, or monthly component. Bs=”cc”
   1. Cc = cubic cycles spline

Smooths that aren’t smooth. Basis-penalty form can express a lot of relationships. (= model it as one thing, then penalize errors to reshape it into the thing it actually is)

Example: Discrete Random Effects. Bs=”re”. Good for age classes.

Can have interactions b/w a discrete-level random effect and a continuous smooth! E.g. medical categorizations. Can specify one of two ways:

1. Gam(y ~ s(xc, xf, bs=”fs”), data=dat)
   1. Model as fct of a continuous var and a factor var
2. Gam(y ~ te(xc, xf, bs=c(“tp”,

Markov Random Fields. Good for spatial area data = countries, states, parcels of land use, etc. Adjacent areas of land that are discrete units, not continuous random variation.

Idea is that each unit has its own value, bt tends to correlate w/ value of nearby units

Gam(y ~ s(x, bs= “mrf”, xt=list(nb=nb)), data=dat)

Great for phylogenetics!

Adaptive smooths

Most models have a single penalizing component, which gives your curve an overall wiggliness

But sometimes that relationship should be different in different parts of the curve. Like, when oscillations get introduced into a previously stable system – vibration for example

Is more data-hungry, bc need to fit piecewise

Bs=”ad”

**Pr distribs**

So that’s the smooths. Now, you’ve got all kinds of diff data you might want to model. R packages can express data in lots of diff ways. Mgcv has a great many pr distribs built into it, so can handle a lot of diff kinds of data.

If you have lots of outliers/ fat-tailed data:

Gam(y ~ s(x), dta=fat\_tailed\_data, family=scat)

Family=scat is student’s T, adjusting data for fat-tailed distrib

Without this the GAM gets misleadingly bouncy

Count data – family=poisson, family=negbin (neg binomial, use for overly dispersed data), family=tw (Tweedy sp? distribution – good for count data when there is a lumpy distribution. E.g. for distribution modeling of social animals).

Ordered categorical data = ratings on your website, IUCN extinct through least concern categories, anything where categories are ordered but not continuous, and maybe not evenly spaced.

Family=ocat

e.g. for ratings where people are clearly hesitant to rate a 1 or a 5, rating data not linear

Has thing w/ IUCN categories and relationship with body size.

Non-ordered categories, categories that aren’t mutually exclusive… there are ways to do this. Multinomial and mvn

These have you pass in more details, e.g. family=multinom(K=2)

**Censored count data uses family = ziplss (zero-inflated Poisson)**

Variable selection: Let’s say you wanna just use the most predictive vars, so it’s parsimonious, runs faster, etc. Just add , select=TRUE parameter to the end of your model! Adds a penalty to wiggliness and overall slope

The slope penalty is similar to L1 or L2 regression – penalizes relationship when there isn’t enough data to support it.

GAMs can be computationally intensive. So we have an fct called bam() for high performance. It’s memory-efficient and parallelizable for handling big(ger) data

Gam() is fine for thousands of data points, tens of thousands or more (esp w/ lots of interactions) you want bam()

Bam() uses various approximations, and does a lot of things piecewise

Complex hierarchical data. (I dunno when I’d use this?) Generalized additive mixed models. You can put in a nested hierarchical structure for your slopes and intercepts.

Blah blah Bayes: you can get your priors in using an fct called jagam(), which generates JAGS code, which is a Bayes-y program

There’s also a link to Tensorflow: gretaGAM::jagam2greta()

…why?

Greta is Bayesian modeling built on Tensorflow. This isn’t, like, some shit like using GAMs as activation fcts to nnets.

**Getting started tips**

Help(package=”mgcv”)

?smooth.terms

?missing.data

?gam.selection

All of the above are basically essays. Great for getting you started.

GAMs An Introduction with R by Simon Wood takes you from linear to GAMs wonderfully

Fromthebottomoftheheap.net is Gavin Simpson’s blog of GAM experimentation with lots of handy code samples.

“Trolling-Driven Data Science” – tell Gavin a GAM can’t do something and he will do it.

noamross.github.io/mgcv-ea-workshop has tons of how-tos and code from a course.

Noam is skeptical of using too many interaction terms. Gets hard to interpret. If there’s a ton of interaction, might want a random forest.

3 interacting components are reasonable, 40 or not.

Using lasso in addition to select=TRUE would be redundant

Can you use select=TRUE for variable selection without running the whole body? No, can’t. gam() iterates trying to find the best smoothing penalty; tests likelihood. Select=TRUE adds more to that machinery, adding a slope-steepiness penalty in addition to wiggliness penalty, so has to happen while the model is being fit.

If you wanna eliminate vars early, do some pair plots! Kick out ones where you don’t see correlations

Look out for collinearity and concurvity!