Lecture 2: Generalized Additive Models
Overview

- The count model, from scratch
- What is a GAM?
- What is smoothing?
- Fitting GAMs using dsm
Building a model, from scratch

- Know count $n_j$ in segment $j$
- Want:

$$n_j = f([\text{environmental covariates}]_j)$$

- Additive model of smooths $s$:

$$n_j = \exp \left[ \beta_0 + s(y_j) + s(\text{Depth}_j) \right]$$

- model terms
- $\exp$ is the link function
Building a model, from scratch

• What about area and detectability?

\[ n_j = A_j \hat{p}_j \exp \left[ \beta_0 + s(y_j) + s(\text{Depth}_j) \right] \]

• \( A_j \) area of segment - "offset"

• \( \hat{p}_j \) probability of detection in segment
Building a model, from scratch

- It's a statistical model so:

\[ n_j = A_j \hat{p}_j \exp \left( \beta_0 + s(y_j) + s(\text{Depth}_j) \right) + \epsilon_j \]

- \( n_j \) has a distribution (count)

- \( \epsilon_j \) are residuals (differences between model and observations)
That's a Generalized Additive Model!
Now let's look at each bit...
Response

\[ n_j = A_j \hat{p}_j \exp \left[ \beta_0 + s(y_j) + s(\text{Depth}_j) \right] + \epsilon_j \]

where \( n_j \sim \text{count distribution} \)
Count distributions

- Response is a count
- Often, it's mostly zero
- mean $\neq$ variance
  - (Poisson isn't good at this)
Tweedie distribution

\[ \text{Var (count)} = \phi \mathbb{E}(\text{count})^q \]

- Poisson is \( q = 1 \)
- We estimate \( q \) and \( \phi \)

(NB there is a point mass at zero not plotted)
Negative binomial distribution

- $\text{Var (count)} = \mathbb{E}(\text{count}) + \kappa \mathbb{E}(\text{count})^2$
- Estimate $\kappa$
- (Poisson: $\text{Var (count)} = \mathbb{E}(\text{count})$)
Smoothes

\[ n_j = A_j \hat{p}_j \exp[\beta_0 + s(y_j) + s(\text{Depth}_j)] + \epsilon_j \]
What about these "s" things?

- Think $s=\text{smooth}$
- Want a line that is "close" to all the data
- Balance between interpolation and "fit"
What is smoothing?
Smoothing

- We think underlying phenomenon is *smooth*
  - "Abundance is a smooth function of depth"

- 1, 2 or more dimensions
Estimating smooths

- We set:
  - "type": bases (made up of basis functions)
  - "maximum wigglyness": basis size (sometimes: dimension/complexity)

- Automatically estimate:
  - "how wiggly it needs to be": smoothing parameter(s)
Splines

- Functions made of other, simpler functions
- **Basis functions** $b_k$, estimate $\beta_k$
- $s(x) = \sum_{k=1}^{K} \beta_k b_k(x)$
Measuring wigglyness

- Visually:
  - Lots of wiggles ⇒ *not smooth*
  - Straight line ⇒ *very smooth*
How wiggly are things?

- Set basis complexity or "size" $k$
- Fitted smooths have effective degrees of freedom (EDF)
- Set $k$ "large enough"
I can't teach you all of GAMs in 1 week

Good intro book

(also a good textbook on GLMs and GLMMs)

Quite technical in places

More resources on course website
Fitting GAMs using dsm
Translating maths into R

\[ n_j = A_j \hat{p}_j \exp[\beta_0 + s(y_j)] + \epsilon_j \]

where \( \epsilon_j \) are some errors, \( n_j \sim \text{count distribution} \)

- inside the link: formula=count ~ s(y)
- response distribution: family=nb() or family=tw()
- detectability: ddf.obj=df_hr
- offset, data: segment.data=segs,
  observation.data=obs
Your first DSM

```r
library(dsm)
dsm_x_tw <- dsm(count~s(x), ddf.obj=df,
               segment.data=segs, observation.data=obs,
               family=tw())
```

dsm is based on mgcv by Simon Wood
summary(dsm_x_tw)

##
## Family: Tweedie(p=1.326)
## Link function: log
##
## Formula:
## count ~ s(x) + offset(off.set)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -19.8115    0.2277  -87.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F  p-value
## s(x) 4.962  6.047 6.403 1.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0283   Deviance explained = 17.9%
## -REML = 409.94  Scale est. = 6.0413    n = 949
Plotting

- `plot(dsm_x_tw)`
- Dashed lines indicate +/- 2 standard errors
- Rug plot
- On the link scale
- EDF on $y$ axis
Adding a term

- Just use +

```r
dsm_xy_tw <- dsm(count ~ s(x) + s(y),
                  ddf.obj=df,
                  segment.data=segs,
                  observation.data=obs,
                  family=tw())
```
summary(dsm_xy_tw)

## Family: Tweedie(p=1.306)
## Link function: log
##
## Formula:
## count ~ s(x) + s(y) + offset(off.set)
##
## Parametric coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.0908     0.2381  -84.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F  p-value
## s(x) 4.943  6.057 3.224 0.004239 **
## s(y) 5.293  6.419 4.034 0.000322 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0678   Deviance explained = 27.4%
## -REML = 399.84  Scale est. = 5.3157    n = 949
Plotting

plot(dsm_xy_tw, pages=1)
Bivariate terms

- Assumed an additive structure
- No interaction
- We can specify $s(x, y)$ (and $s(x, y, z, \ldots)$)
Bivariate spatial term

dsm_xyb_tw <- dsm(count ~ s(x, y),
                 ddf.obj=df,
                 segment.data=segs,
                 observation.data=obs,
                 family=tw())
summary(dsm_xyb_tw)

##
## Family: Tweedie(p=1.29)
## Link function: log
##
## Formula:
## count ~ s(x, y) + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.2745     0.2477  -81.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##     edf Ref.df     F  p-value
## s(x,y) 16.89  21.12 4.333 3.73e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.102   Deviance explained = 34.7%
## -REML = 394.86   Scale est. = 4.8248   n = 949
plot(dsm_xyb_tw, select=1, scheme=2, asp=1)

- On link scale
- `scheme=2` makes heatmap
- (set `too.far` to exclude points far from data)
Comparing bivariate and additive models
Let's have a go...