

# BOOSTING 1

-INTRODUCTION TO DATA SCIENCE-

Lecturer: Darren Homrighausen, PhD

# Preamble:

- Boosting looks superficially like bagging, but has an important difference
- To understand boosting, we need to discuss nonparametric regression
- Nonparametric regression provides a flexible fit, but can only be used in low dimensions
- Additive models are a relaxation of nonparametric models that can be fit in much higher dimensions
- Boosting is an algorithm for fitting a type of additive model

# BOOSTING OVERVIEW

**RECALL:** Bagging is a procedure for taking a low bias, high variance procedure and (potentially) reducing its risk via averaging

**Boosting** has a similar philosophy: take a poor classifier and improve it

However, **boosting** is useful for the **opposite** situation: a classifier that has high bias but low variance!

**EXAMPLE: LINEAR REGRESSION.** It would make sense to...

- ...use bagging with linear regression on all of the features
- ...use boosting with simple linear regression on only one feature at a time

# BOOSTING OVERVIEW

A direct contrast:

- **BAGGING:** aggregates over many **independent** bootstrap draws

(To be clear, the bootstrap sampling mechanism is independent, the bootstrap samples themselves are of course dependent)

- **BOOSTING:** finds the observations that are poorly classified, up weights these observations, and then trains a new classifier

(Similar statement about regression)

# Boosting for Regression

# BOOSTING REGRESSION TREES

Set  $\hat{f} \equiv 0$  and  $R = Y \in \mathbb{R}^n$

Fix the tree complexity  $M$  and learning rate  $\lambda$

(ESL states that  $M \in \{4, \dots, 8\}$  works well. This controls how many interactions can be in the model. More on this later)

For  $b = 1, \dots, B$ , do:

1. Fit  $\hat{f}_b$  with  $M$  regions to  $\tilde{\mathcal{D}} = \{(X_1, R_1), \dots, (X_n, R_n)\}$
2. Update:  $\hat{f} \leftarrow \hat{f} + \lambda \hat{f}_b$
3. Update:  $R \leftarrow R - \lambda \hat{f}_b$

OUTPUT:

$$\hat{f} = \sum_{b=1}^B \lambda \hat{f}_b$$

This is a type of **additive** model, so called for obvious reasons

# BOOSTING TREES

In general

- A smaller  $\lambda$  means a larger required  $B$
- Too large of  $\lambda$  means we take too long of steps, leading to poor solutions  
(RECALL: gradient descent)

In practice,

- $B$  is set via cross-validation  
(Though boosting is somewhat insensitive to choosing  $B$  too large, it is still an important part of the overall performance)
- $\lambda$  is set at a small level, say  $\lambda = 0.01$

As for the additive model part...

# Curse of dimensionality and local averaging



# FROM LINEAR TO NONLINEAR MODELS

**GOAL:** Develop a prediction function  $\hat{f} : \mathbb{R}^p \rightarrow \mathbb{R}$  for predicting  $Y$  given an  $X$

Commonly,  $f(X) = X^\top \beta$

(e.g. Constrained linear regression)

This greatly simplifies algorithms, while not sacrificing too much flexibility

However, sometimes directly modeling the nonlinearity is more natural

# PREDICTION VIA LOCAL AVERAGING

The fundamental quantities of interest we have been modeling are the **Bayes' rules**

$$\mathbb{E}[Y|X] \quad \text{or} \quad \arg \max_g \mathbb{P}(Y = g|X)$$

We know how to estimate expectations:

→ If  $Y_1, Y_2, \dots, Y_n$  all have expectation  $\mu$ , then

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n Y_i$$

is an intuitive estimator of  $\mu$   
(and a reasonable prediction of a new  $Y$ )

# PREDICTION VIA LOCAL AVERAGING

Similarly, we can estimate  $\mathbb{E}[Y|X_*]$  with  $\mathcal{D}$ :

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(X_i = X_*)$$

where  $n_* = \sum_{i=1}^n \mathbf{1}(X_i = X_*)$  is the number of training observations where  $X_i = X_*$

**IN WORDS:** we are taking an average of all the observations  $Y_i$  such that  $X_i = X_*$

(Of course, this is all conditional expectation really is)

**There is a big problem:** There generally aren't any  $X_i$  at  $X_*$ !

# PREDICTION VIA LOCAL AVERAGING

Suppose we relax the constraint  $X_i = X_*$  a bit and include points that are **close enough** instead

Define a generic function “neighborhood” which takes in a value  $X_*$  and reports:

$$\text{neighborhood}(X_*) = \text{all } X_i \text{ in the “neighborhood” of } X_*$$

To estimate  $\mathbb{E}Y|X_*$ , average the **nearby** training observations

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(X_i \in \text{neighborhood}(X_*))$$

where  $n_* = |\text{neighborhood}(X_*)|$  is the number of training observations where  $X_i$  is in the “neighborhood” of  $X_*$

# PREDICTION VIA LOCAL AVERAGING

We need to define  $\text{neighborhood}(X_*)$

(If it hasn't become clear, yet, this is a key aspect characterizing different methods)

**EXAMPLE:** Let's define “neighborhood” in terms of having Euclidean norm less than or equal to some threshold  $t$

$$\mathbf{1}(X_i \in \text{neighborhood}(X_*)) = \mathbf{1}(\|X_i - X_*\|_2 \leq t)$$

Here...

- ... the Euclidean norm quantifies **distance**
- ...  $t$  quantifies **close**

(In fact, it is a tuning parameter)

# PREDICTION VIA LOCAL AVERAGING: $X \in \mathbb{R}$

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(\|X_i - X_*\|_2 \leq t), \quad \text{and} \quad X_* = 0.25$$

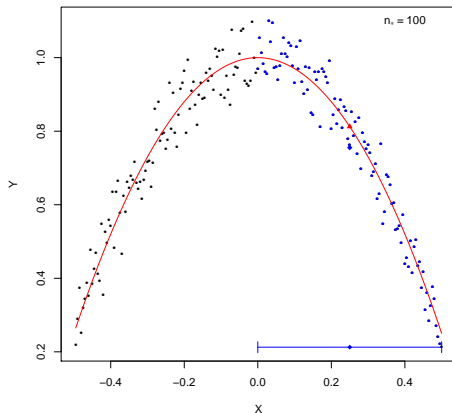


FIGURE:  $t = 0.25$

# PREDICTION VIA LOCAL AVERAGING: $X \in \mathbb{R}$

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(\|X_i - X_*\|_2 \leq t), \quad \text{and} \quad X_* = 0.25$$

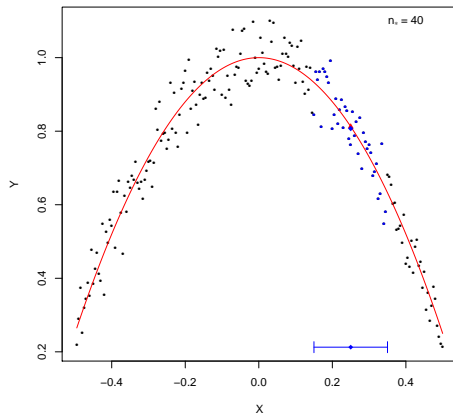


FIGURE:  $t = 0.1$

# PREDICTION VIA LOCAL AVERAGING: $X \in \mathbb{R}$

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(\|X_i - X_*\|_2 \leq t), \quad \text{and} \quad X_* = 0.25$$

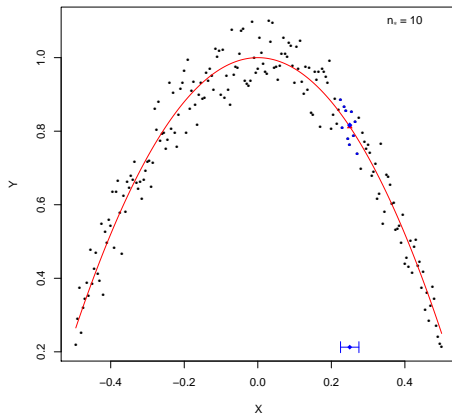


FIGURE:  $t = 0.025$



# PREDICTION VIA LOCAL AVERAGING: $X \in \mathbb{R}$

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(\|X_i - X_*\|_2 \leq t), \quad \text{and} \quad X_* = 0.25$$

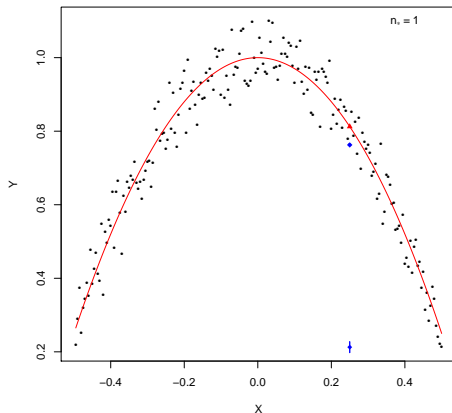


FIGURE:  $t = 0.0001$

# PREDICTION VIA LOCAL AVERAGING: $X \in \mathbb{R}$

$$\hat{f}(X_*) = \frac{1}{n_*} \sum_{i=1}^n Y_i \mathbf{1}(\|X_i - X_*\|_2 \leq t)$$

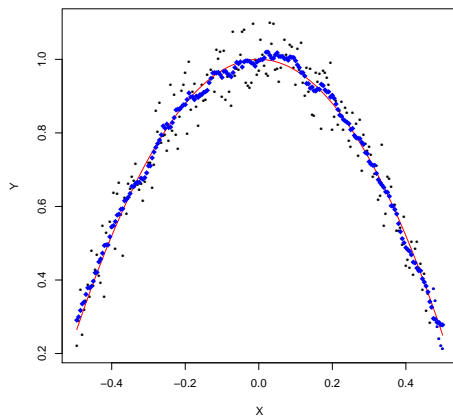


FIGURE: Over a grid of  $X_*$ 's, with  $t = 0.025$

# PREDICTION VIA LOCAL AVERAGING

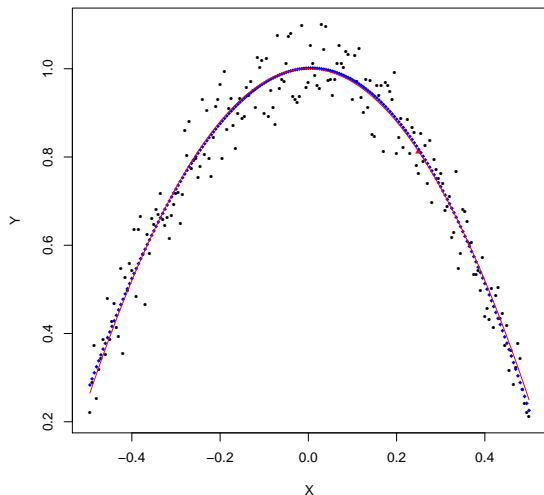


FIGURE: Using Regression Splines (ISL 7.4)

# Nonparametric regression: Splines

# NONPARAMETRIC REGRESSION

## EXAMPLE: Fitting polynomials

(See ISL 7.1 for details on fitting polynomials)

Suppose we want to consider the following 4-dimensional feature transformation of a original 2-dimensional feature vector  $X = (x_1, x_2)^\top \in \mathbb{R}^2$ :

$$\Phi(X) = (1, \phi_1(X), \phi_2(X), \phi_3(X)) = (1, x_1, x_2, x_2^2)$$

For regression, we can model

$$\begin{aligned} f_*(X) &= \beta^\top \Phi(X) \\ &= \beta_0 + \beta_1 \text{balance} + \beta_2 \text{income} + \beta_3 \text{income}^2 \end{aligned}$$

# NONPARAMETRIC REGRESSION

In nonparametric regression, the functions  $\phi_k$  are **basis functions**

(See ISL 7.3)

A general model would be

$$f_*(X) = \beta^\top \Phi(X)$$

Which we can fit via

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} ||Y - \Phi\beta||_2^2,$$

where  $\Phi = [\phi_k(X_i)] \in \mathbb{R}^{n \times K}$ ,

- $k$  indexes the columns of  $\Phi$ :  $k = 1, \dots, K$
- $i$  indexes the rows of  $\Phi$ :  $i = 1, \dots, n$

# NONPARAMETRIC REGRESSION: SPLINES

Choosing different  $\phi_k$ , leads to different fits

A classic choice is **splines**

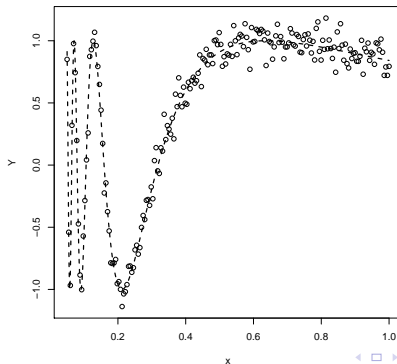
Splines are a basis that provides a very **smooth** representation of the regression function

But, again, we can view them as a way of defining a **distance** and **close**

Let's turn to the **DOPPLER FUNCTION** for an example..

# NONPARAMETRIC REGRESSION: EXAMPLE

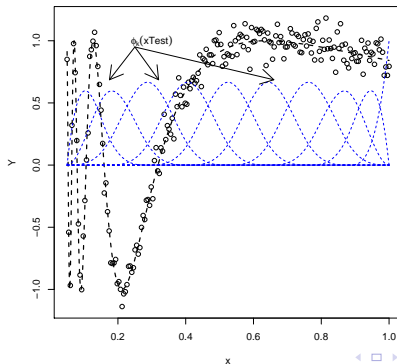
```
x = seq(.05,1,length=200)
Y = sin(1/x) + rnorm(100,0,.1)
plot(x,Y)
xTest = seq(.05,1,length=1000)
lines(xTest,sin(1/xTest),col='black',lwd=2,lty=2)
```





# NONPARAMETRIC REGRESSION: EXAMPLE

```
require(splines)
Phi = bs(x,df=10)
plot(x,Y)
lines(xTest,sin(1/xTest),col='black',lwd=2,lty=2)
matlines(x=x,Phi,lty=2,type='l',col='blue')
```



# NONPARAMETRIC REGRESSION: EXAMPLE

$\Phi = \text{bs}(x, \text{df}=K)$

$\hat{Y} = \text{predict}(\text{lm}(Y \sim ., \text{data}=\Phi))$

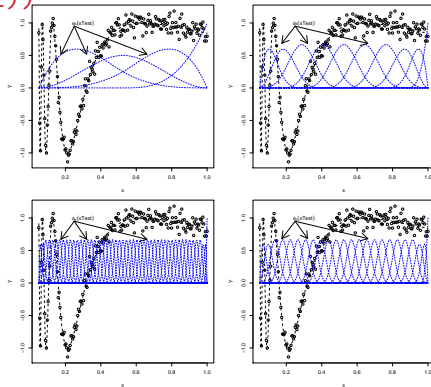
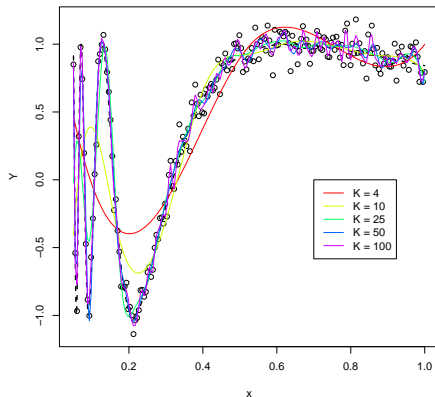


FIGURE:  $K = 4, 10, 24, 50$ ,  
clockwise from top left

# FROM LINEAR TO NONLINEAR MODELS

**QUESTION:** Why don't we always fit such a flexible model?

**ANSWER:** This works great if  $p$  is small (though it is easier to interpret linear regression output)

(and the specification of **distance** and **close** is good)

However, as  $p$  gets large

- **nothing** is nearby
- **all** points are on the boundary

(Hence, predictions are generally extrapolations)

These aspects make up (part) of the **curse of dimensionality**

# CURSE OF DIMENSIONALITY

Fix the dimension  $p$

(Assume  $p$  is even to ignore unimportant digressions)

Let  $S$  be a hypersphere with radius  $r$

Let  $C$  be a hypercube with side length  $2r$

Then, the volume of  $S$  and  $C$  are, respectively

$$V_S = \frac{r^p \pi^{p/2}}{(p/2)!} \text{ and } V_C = (2r)^p$$

(Interesting observation: this means for  $r < 1/2$  the volume of the hypercube goes to 0, but the diagonal length is always  $\propto \sqrt{p}$ . Hence, the hypercube gets quite 'spiky' and is actually horribly jagged. Regardless of radius, the hypersphere's volume goes to zero quickly.)

# CURSE OF DIMENSIONALITY

Hence, the ratio of the volumes of a circumscribed hypersphere by a hypercube is

$$\frac{V_C}{V_S} = \frac{(2r)^p \cdot (p/2)!}{r^p \pi^{p/2}} = \frac{2^p \cdot (p/2)!}{\pi^{p/2}} = \left(\frac{4}{\pi}\right)^d d!$$

where  $d = p/2$

**OBSERVATION:** This ratio of volumes is increasing **really** fast. This means that all of the volume of a hypercube is near the corners. Also, this is independent of the radius.

# Additive models

(ISL 7.7, 7.8.3)

# ADDITIVE MODELS

The linear model looks like:

$$f(X) = \beta^\top X = \sum_{j=1}^p \beta_j x_j$$

The general nonparametric model looks like:

$$f(X) = \beta^\top \Phi(X) = \sum_{j=1}^K \beta_j \phi_j(X)$$

COMBINE THEM:

$$f(X) = f_1(x_1) + \cdots + f_p(x_p) = \sum_{j=1}^p f_j(x_j)$$

Estimation of such a procedure is not much more complicated than a fully linear model (as all inputs enter separately)

# ADDITIVE MODELS (FOR REGRESSION)

Additive models are usually phrased using the **population level** expectation

(These get replaced with empirical versions)

The update is a Gauss-Seidel-type update

(The Gauss-Seidel method is an iterative scheme for solving linear, square systems)

This is for  $j = 1, \dots, p, 1, \dots, p, 1 \dots$ :

$$f_j(x_j) \leftarrow \mathbb{E} \left[ Y - \sum_{k \neq j} f_k(x_k) | x_j \right]$$

Under fairly general conditions, this converges to  $\mathbb{E}[Y|X]$



# ADDITIVE MODELS (FOR REGRESSION)

Backfitting for additive models is roughly as follows:

Choose a univariate nonparametric smoother  $\mathcal{S}$  and form all univariate fits  $\hat{f}_j$  on each  $x_j$  marginally

(Common choice: splines)

Iterate over  $j$  until convergence:

1. Define the residuals  $R_i = Y_i - \sum_{k \neq j} \hat{f}_k(X_{ik})$
2. Smooth the residuals  $\hat{f}_j = \mathcal{S}(R)$
3. Center  $\hat{f}_j \leftarrow \hat{f}_j - n^{-1} \sum_{i=1}^n \hat{f}_j(X_{ij})$

Report

$$\hat{f}(X) = \bar{Y} + \hat{f}_1(x_1) + \cdots + \hat{f}_p(x_p)$$

# FITTING ADDITIVE MODELS R

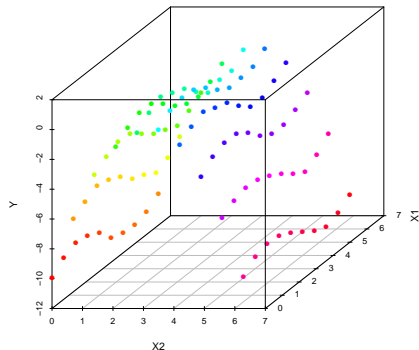
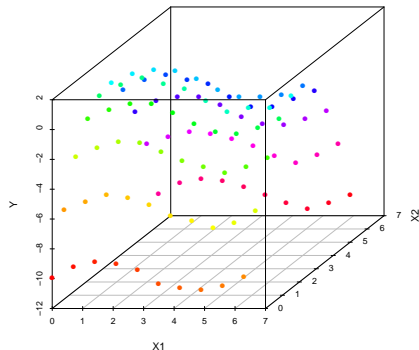
```
library(gam)
x  = seq(0,2*pi,length=10)
xx = expand.grid(x,x)
x1 = xx[,1]
x2 = xx[,2]

Y = sin(xx[,1]) - (xx[,2] - pi)^2 + rnorm(nrow(xx),0,.1)
sim = data.frame(x1=x1,x2=x2,Y=Y)

out = gam(Y~s(x1,3)+s(x2,3),data=sim)
```

# ADDITIVE MODELS: SIMULATION

Plotting the simulated data:  $Y$  as a function of  $x_1, x_2$ :



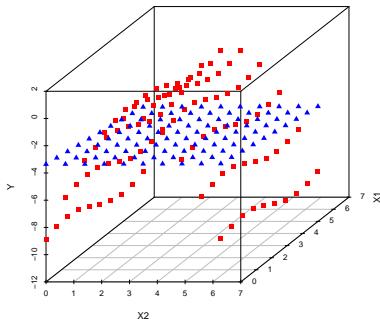
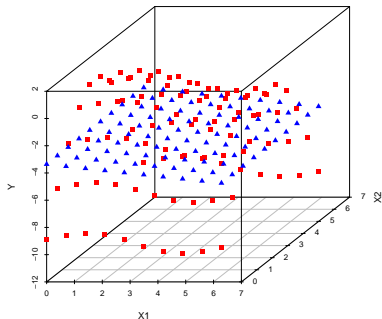
# ADDITIVE MODELS: SIMULATION RESULTS

If we fit the (multiple) linear regression:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon,$$

we miss a lot of the structure (blue triangles)

The smoother GAM is able to capture it, however:



(These are the fitted values only. Red squares: GAM, Blue triangles: linear regression)

## DETOUR: PLOTTING 3D IN R

```
out = scatterplot3d(X1,X2,Y,pch=16,type='n')
xyz = out$xyz.convert(X1,X2,out.pred)
points(xyz,col='red',pch=15)
xyz = out$xyz.convert(X1,X2,out.pred.lm)
points(xyz,col='blue',pch=17)
```

# ADDITIVE MODELS (FOR CLASSIFICATION)

As squared error loss isn't quite right for classification, **additive logistic regression** is a popular approach

Suppose  $Y \in \{-1, 1\}$

$$\log \left( \frac{\mathbb{P}(Y = 1|X)}{\mathbb{P}(Y = -1|X)} \right) = \sum_{j=1}^p f_j(x_j) = f(X)$$

This gets inverted in the usual way to acquire a probability estimate

$$\pi(X) = \mathbb{P}(Y = 1|X) = \frac{e^{f(X)}}{1 + e^{f(X)}}$$

$(f_j(x_j) = \beta_j x_j$  and  $f(X) = X^\top \beta$  gives us (linear) logistic regression)

These models are usually fit by numerically maximizing the binomial likelihood, and hence enjoy all the asymptotic optimality prospects of MLEs

# ADDITIVE MODELS (FOR CLASSIFICATION)

**EXAMPLE:** In **R**, this can be fit with the package **gam**

In the **gam** package there is a dataset **kyphosis**

This dataset examines a disorder of the spine

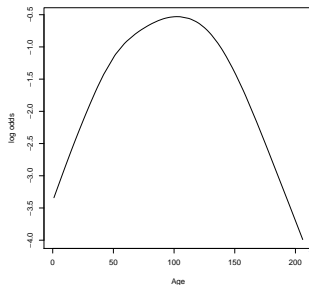
Let's look at two possible features **Age** and **Number**

(**Number** refers to the number of vertebrae that were involved in a surgery)

# ADDITIVE MODELS (FOR CLASSIFICATION)

```
library(gam)
data(kyphosis)

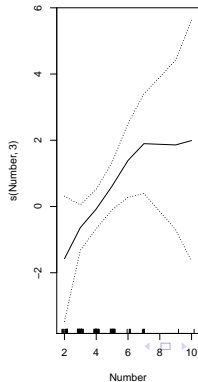
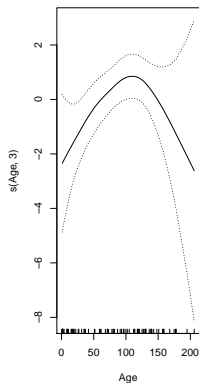
out = gam(Kyphosis~s(Age,3),family=binomial,data=kyphosis)
out.pred = predict(out)
plot(sort(kyphosis$Age),out.pred[order(kyphosis$Age)],
     type='l',xlab='Age',ylab='log odds')
```





# ADDITIVE MODELS (FOR CLASSIFICATION)

```
out = gam(Kyphosis ~ s(Age,3) + s(Number,3),  
          family = binomial, data=kyphosis)  
par(mfrow=c(1,2))  
plot(out,se=T)
```



# Postamble:

- Boosting looks superficially like bagging, but has an important difference  
(Apply to high bias/low variance procedures)
- To understand boosting, we need to discuss nonparametric regression
- Nonparametric regression provides a flexible fit, but can only be used in low dimensions  
(Curse of dimensionality)
- Additive models are a relaxation of nonparametric models that can be fit in much higher dimensions
- Boosting is an algorithm for fitting a type of additive model  
(More on this in the next set of notes)