# A Study on Spline Regression

Arman Terzyan

California State University, Northridge

December 9, 2019

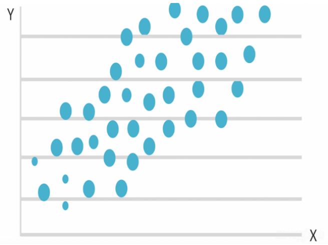
# Outline

# Regression

Estimating the relationship between a dependent variable and one or more independent variables.

Most common parametric methods are:

- Linear Regression
- Polynomial Regression



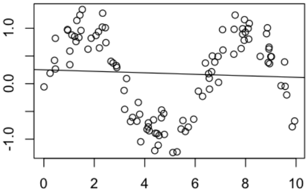

For example: hours studying $\longrightarrow$ final grade

# Linear Regression

A linear function is utilized to model the relationship with coefficients that minimize the sum of squared residuals.

$$\hat{Y} = \beta_0 + \beta_1 X + \epsilon$$

Choose $\beta_0$ and $\beta_1$ such that $\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ is minimized.

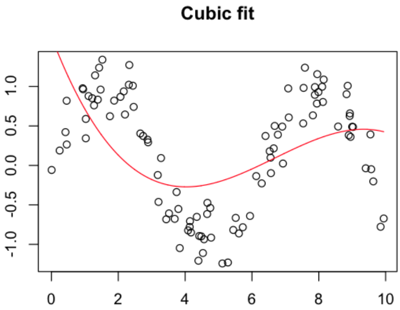



Could result in high bias due to underfitting.

# Polynomial Regression

A polynomial function of nth degree is used to model the relationship between the response and the predictors.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_n X^n + \epsilon$$




Potential for overfitting.

## Parametric

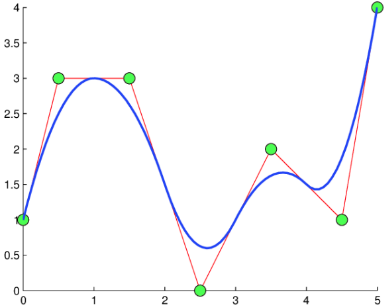
- Finite set of parameters
- Less data requirements
- Higher power (when assumptions are correct)
- Set global structure

## Nonparametric

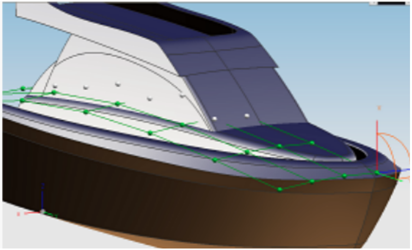
- Model learns from the data with no set amount of parameters
- More flexible but higher data requirements
- Interpretation of variable relationship more difficult

# What is a spline?

- A function that is constructed piecewise by polynomials.
- In other words, a set of two or more curves joined together at predetermined points.



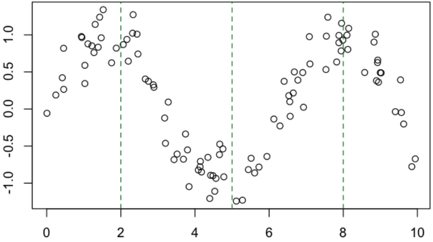
# History of Spline



- Originally developed for ship-building where builders would place weights at certain points and bend a rod through those weights.
- Heavily used in airplane design as well as plotting trajectories

# How to build a spline function

1. Divide data into a chosen number of segments
   - The point(s) at which the data is divided is called a knot usually denoted by $\xi$



2. Model each bin between the the segments with a polynomial(ideally of lower degree)
    - Select a degree for the model where degree=1 models each segment with a line while the most common being degree=3 for a cubic spline
    - For a smoother overall shape we want d-1 continuous derivatives at each knot where d is the degree chosen

3. Utilize the ReLu function to force the functions to join at the knots
   - Suppose we have chosen a single knot $\xi_1$ and want a spline of degree $d = 3$ then the equation for our spline will be:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \xi_1)_+^3$$

**Translated ReLu**

$$(X - a)_+^d = \begin{cases} 0 & X < a \\ (X - a)^d & X \geq a \end{cases}$$

# Basis

- **Truncated Power Basis**

$$h_1(x) = 1, h_2(x) = x, \cdots, h_{d+1}(x) = x^d, h_{d+1+k}(x) = (x - \xi_k)_+^d$$

$$Y = \sum_{i=1}^{d+k+1} B_i h_i(x)$$

$d$=degree, $k$=number of knots

4. After selecting number of knots and desired degree for spline, we must build the design matrix.
   - Suppose $d = 3$ with 2 knots $\xi_1$ and $\xi_2$ and a sample $x_1, x_2, ..., x_n$. Then the design matrix will be:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & (x_1 - \xi_1)_+^3 & (x_1 - \xi_2)_+^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & (x_n - \xi_1)_+^3 & (x_n - \xi_2)_+^3 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{pmatrix}$$

# Code implementation

```
set.seed(123)#set seed
x=runif(100,0,10)#build toy dataset and plot
y=sin(x)+ .25*rnorm(100)
plot(x,y)
knots=c(2,5,8)#choose knots
abline(v=knots,lty=2,col="darkgreen")#plot knots
df=data.frame(cbind(x,y))#dataframe of toy dataset values
colnames(df)=c('x','y')
```

# Code implementation

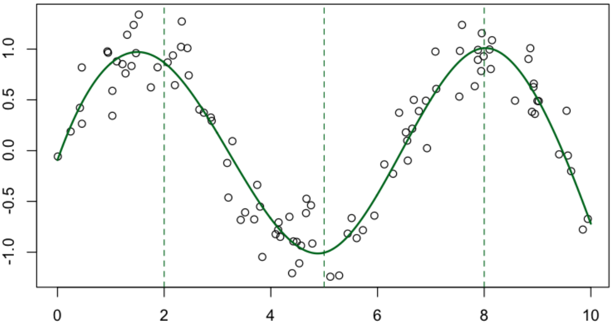
```
degree=3#set degree
X1=outer(df$x,1:degree,"^")#build design matrix of data to esimate Beta values
X2=outer(df$x,knots,">")* #check if value of x is in interval
  outer(df$x,knots,"-")^degree
ones=rep(1,nrow(df))
X=cbind(ones,X1,X2)
mydf=data.frame(cbind(X,y))#combine with response
model = lm(y ~.,data=mydf)#fit with linear model
```

# Code implementation

```r
#build function to create new data points that will be used for plotting the spline
newPoints=function(k,points,degree){
  X1=outer(points,1:degree,"^")
  X2=outer(points,k,">")*
    outer(points,k,"-")^degree
  ones=rep(1,length(points))
  X=cbind(ones,X1,X2)
  X=as.data.frame(X)
  return(X)
}

x.new=seq(0,10,by = .01)#new data points
points(x.new,predict(model,newdata = newPoints(knots,x.new,degree),type='response'),col="darkgreen",lwd=2,type="l")
```

# Final result




Issues can arise at the boundaries when performing spline regression.
**Fix**:*Natural Splines*-
Force the degree of the polynomial on the bounds to be equal to $\frac{(d-1)}{2}$.

# Why use spline regression?

1. Stability
2. Control over erratic regions
3. Appropriate knot selection will result in good bias-variance trade off

# Knot selection

- Trial and error
- Cross validation
- *Smoothing splines*-place knots at every data point and control for overfitting by adding penalty term that is large when 2nd derivative is "wiggly"

# References


1. "All of Nonparametric Statistics"- Larry Wasserman
2. "Elements of Statistical Learning"- Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie

Thank You!