Linear Statistical Modeling Methods with SAS

Moving Beyond Linearity

Xuemao Zhang
East Stroudsburg University

April 3, 2024

Outline

Linearity assumption is not always good enough.

- Polynomials
- Splines
  - Cubic Splines
  - Natural Cubic Splines
  - Smoothing Splines
- Local regression

Polynomial Regression

- Sometimes, simple linear regression model is not sufficient to describe the relationship between two numerical variables.
- The standard way to extend to **nonlinear** is to replace the standard linear model with a polynomial function we discussed before

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \varepsilon_i$$

- Logistic regression follows naturally.

$$Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}$$

Polynomial Regression

- Consider the `Wage` data:

```
PROC IMPORT
DATAFILE='/home/u5235839/my_shared_file_links/u5235839/Wage.csv'
DBMS=CSV
OUT=Wage;
GETNAMES=YES;
RUN;

proc contents data=Wage;
run;
```

Polynomial Regression

- Scatter plot of `wage` (response) verus age with SLR fit

```
proc sgplot data=Wage;
reg x=age y=wage;
run;
```

Polynomial Regression

- we first fit a simple linear regression model.

```
proc reg data=Wage;
   model wage = age;
   ods output parameterestimates=coef_summary;
run;
```

```
proc print data=coef_summary;
run;
```

Polynomial Regression

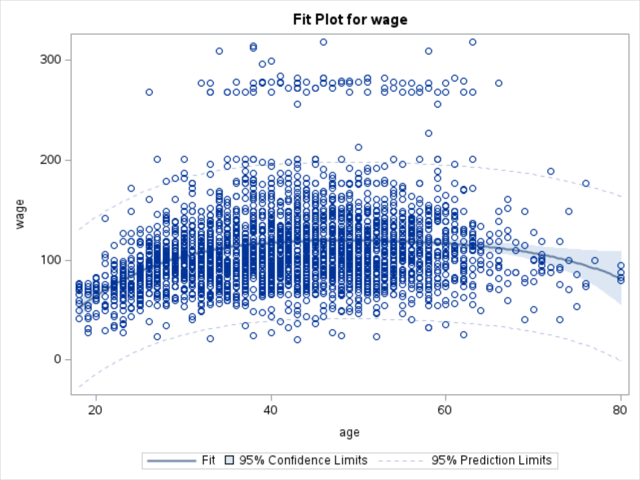
- For the Wage data, we fit a simple polynomial regression model with degree 4.
  - The bar notation means we fit an MLR with predictors age, age^2, age^3 and age^4.

```
proc glm data=Wage;
   model wage = age|age|age|age;
   ods output parameterestimates=coef_summary;
run;


proc print data=coef_summary;
run;
```

Polynomial Regression

- Check the scatter plot with the fit



Polynomial Regression

- In performing a polynomial regression we must decide on the degree of the polynomial to use.

```
proc glmselect data=Wage;
   model wage = age|age|age|age / selection=backward sls=0.01;
   output out=fit_results predicted=fit_values;
run;
```

- A cubic polynomial appears to provide a reasonable fit to the data.

Polynomial Regression

- Next we consider the task of predicting whether an individual earns more than \$250,000 per year. We fit a polynomial logistic regression model.
  - Introduce a binary variable `wage_level` which is 1 when `wage>250`, and 0 otherwise.

```
data Wage;
set Wage;
wage_level = (wage > 250);
run;
```

Polynomial Regression

- We used `proc genmod` fit logistic regression model before. Now we use proc logistic because it supports variable selection.
  - https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.4/statug/statug_logistic_toc.htm

```
proc logistic data=Wage outest=betas PLOTS = ALL;
   model wage_level(event='1') = age|age|age|age /
  selection=backward sls=0.01;
  output out=pred p=phat predprob=(individual);
run;
```

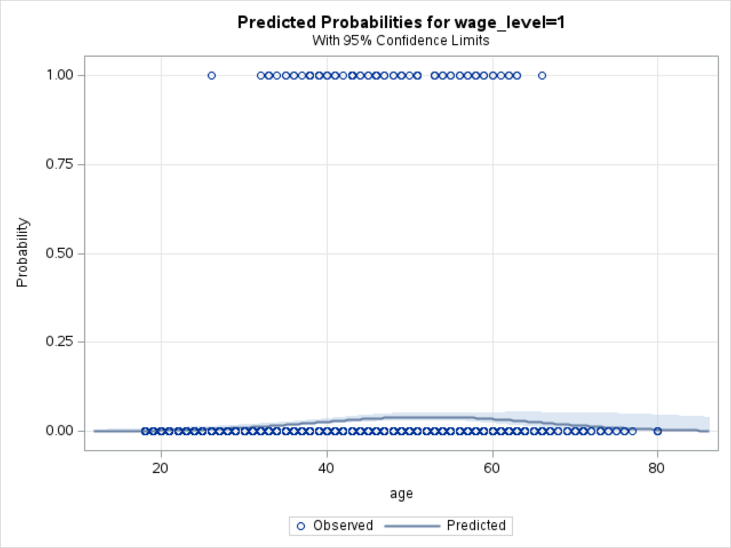
Polynomial Regression

```
proc print data=betas;
title2 'Parameter Estimates';
run;
```

```
proc print data=pred;
title2 'Predicted Probabilities and 95% Confidence Limits';
run;
```

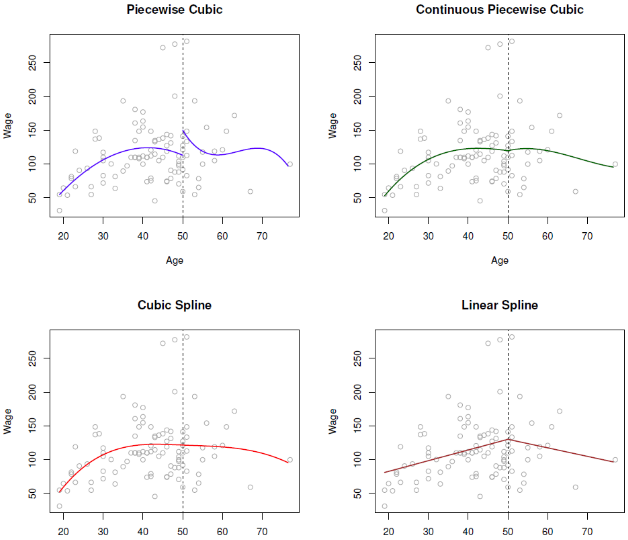
Polynomial Regression

- Fitted probabilities and the corresponding 95% confidence bands.



Piecewise Polynomials

- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by **knots**.



Linear Splines

- Better to add constraints to the polynomials, e.g. continuity.
- **Splines** have the "maximum'' amount of continuity.
- Suppose the knot is $\xi = 50$, then the model for the linear spline is

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \varepsilon_i,$$

where $b_1(x_i)$ and $b_2(x_i)$ are basis functions:

$$\begin{aligned} b_1(x_i) =& x_i \\ b_2(x_i) =& (x_i - 50)_+ = \begin{cases} x_i - 50, & \text{if } x_i > 50 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

- The construction guarantees that the linear spline is continuous at the knot 50.

Linear Splines

- A linear spline with knots at $\xi_k, k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot. We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \varepsilon_i,$$

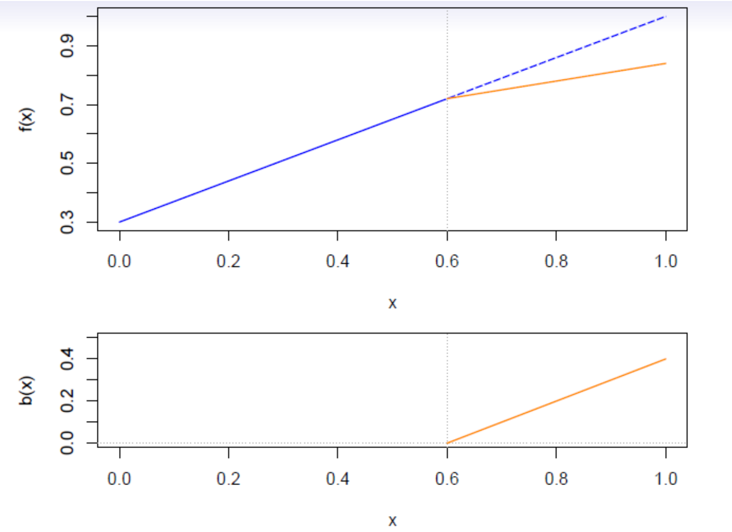
where $b_k(x_i)$ are basis functions:

$$\begin{aligned} b_1(x_i) =& x_i \\ b_{k+1}(x_i) =& (x_i - \xi_k)_+, k = 1, \ldots, K. \end{aligned}$$

Here $()_+$ means positive part:

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k, & \text{if } x_i > \xi_k \\ 0, & \text{otherwise} \end{cases}$$

Linear Splines



Cubic Splines

- A cubic spline with knots at $\xi_k, k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot. We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \varepsilon_i,$$

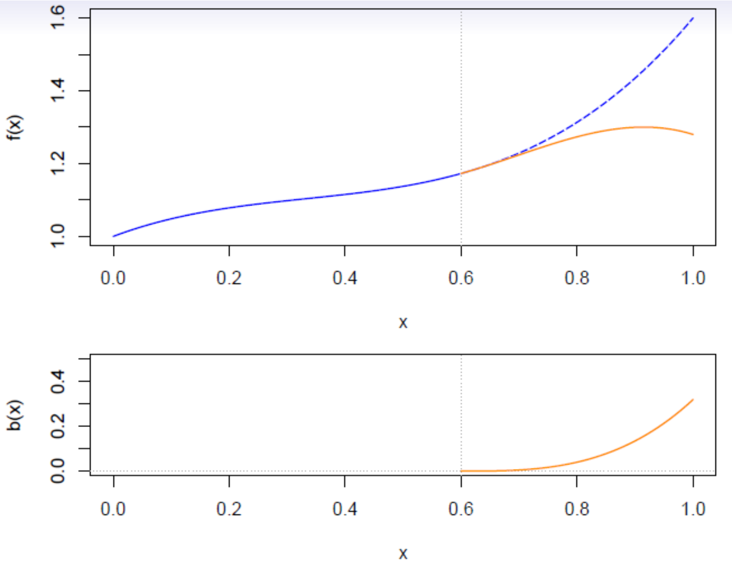
where $b_k(x_i)$ are basis functions:

$$\begin{aligned} b_1(x_i) =& x_i \\ b_2(x_i) =& x_i^2 \\ b_3(x_i) =& x_i^3 \\ b_{k+1}(x_i) =& (x_i - \xi_k)_+^3, k = 1, \ldots, K. \end{aligned}$$

where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3, & \text{if } x_i > \xi_k \\ 0, & \text{otherwise} \end{cases}$$

Cubic Splines



SAS: Cubic Splines

- In order to fit regression splines, we use The TRANSREG Procedure
  - https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.4/statug/statug_transreg_details.htm
  - IDENTITY: no transformation

```
proc transreg data=Wage;
title2 'A Cubic Spline Fit with Knots at X=25, 40, 60';
   model IDENTITY(wage) = spline(age / knots=25 40 60);
run;
```

Smoothing Splines

- In fitting a smooth curve to a set of data, what we really want to do is find some function, say $g(x)$, that fits the observed data well.
- Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\text{minimize}_{g\in\mathcal{S}} \sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int_{\text{range of data}} [g^{''}(t)]^2 \, dt$$

  - The equation takes the "Loss+Penalty'' formulation that we encounter in the context of ridge regression and the lasso regression.
  - The first term is SSE, and tries to make $g(x)$ match the data at each x_i.
  - The second term is a roughness penalty and controls how wiggly $g(x)$ is. It is modulated by the tuning parameter $\lambda \geq 0$.
    - ⋆ The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.
    - ⋆ As $\lambda \to \infty$, the function g(x) becomes linear.

Smoothing Splines

- The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.
- However, it is not the same natural cubic spline that one would get if one applied the basis function approach.
  - It is a shrunken version of such a natural cubic spline, where the value of the tuning parameter λ controls the level of shrinkage.
- In SAS, we use the same procedure `proc transreg`.
  - We can specify df rather than λ.

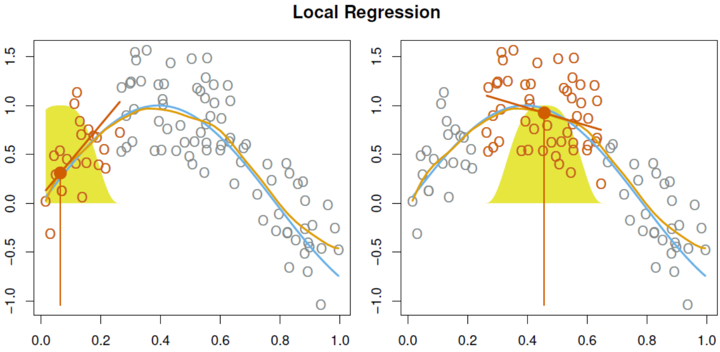
Smoothing Splines

- In order to fit a smoothing spline, we use the `smooth.spline()` function.
  - `SS2`: Displays regression results
  - `SM=`: You can specify the smoothing parameter with either the SM= or the PARAMETER= t-option. The default smoothing parameter is `SM=0`

```
proc transreg SS2 data=Wage;
   model identity(wage) = smooth(age / sm=50);
run;
```

Local Regression

- Local regression is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point x_0 using only the nearby training observations.



Local Regression

- With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares.
- Algorithm: Local Regression At $X = x_0$
  - 1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
  - 2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.
  - 3. Fit a weighted least squares regression of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_0$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$$

  - 4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Local Regression

- In order to perform local regression, we use the `loess` procedure
  - https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.4/statug/statug_loess_toc.htm
  - `DEGREE=`: Specifies the degree of local polynomials (1 or 2)
  - `SMOOTH=`: Specifies the list of smoothing values between 0 and 1

```
proc loess data=Wage plots=FitPlot;
   model wage = age / degree=2 smooth = 0.3;
   /* Specify the degree of the polynomial and the span for smoothing */
   run;
```

Local Regression

```
proc loess data=Wage plots=FitPlot;
   model wage = age / degree=2 smooth = 0.6;
   /* Specify the degree of the polynomial and the span for smoothin
   run;
```

Local Regression

- `SELECT=` specifies that automatic smoothing parameter selection be done in the `model statement`

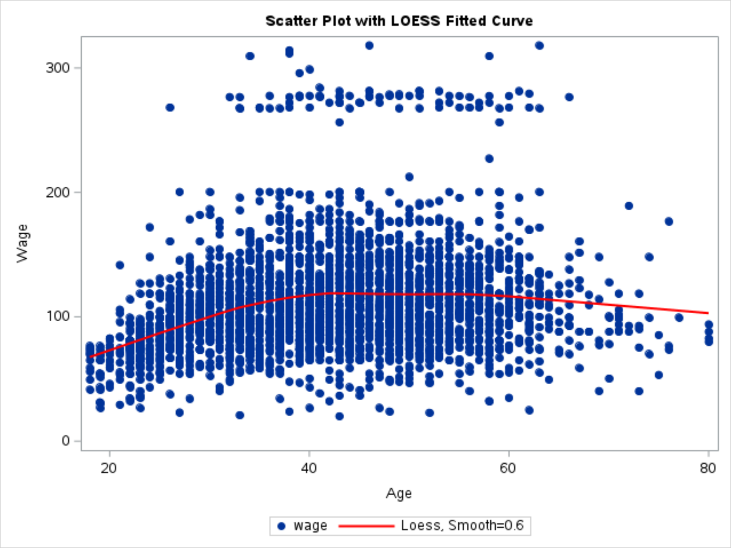
```
   proc loess data=Wage;
     model wage = age / degree=2 select=AICC
     /**select=AICC(steps) show steps of selection */
     smooth = 0.2 0.4 0.6 0.8 alpha=.01;
run;
```

Local Regression

- The `loess` fitted curve can be added to a scatter plot using `proc sgplot`

```
proc sgplot data=Wage;
   scatter x=age y=wage / markerattrs=(symbol=circlefilled);
   loess x=age y=wage / smooth=0.6 lineattrs=(COLOR=red);
   /* Specify the same smoothing parameter used in proc loess */
   xaxis label="Age";
   yaxis label="Wage";
   title2 "Scatter Plot with LOESS Fitted Curve";
run;
```

Local Regression



License