UNIVERSITY OF
SAN FRANCISCO

Master of Science
in Analytics

Beyond Linearity

Machine Learning 1

Outline of techniques

- Linear -/+
  - Data is almost never linear
  - A linear assumption is sometimes a good enough model
- When linear techniques are not good enough
  - Polynomial regression
  - Step functions
  - Splines
  - Local regression
  - Generalized additive models

Data - DJIA close

- See https://github.com/dbrizan/MSAN621-data > djia*csv
- Features: only using days since 1st Feb 1987, inferred from date
- Outcome: Close = csv_file[-1]

Polynomial Regression

- Generally form of model:

$$y_i = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$$

  - For degree “d” polynomial
- Other forms of regression have polynomial behaviour
  - Ridge regression, the lasso, etc.
  - These are not classic polynomial regression

Implementation in python

1) Import data
    a) If necessary, split data into train, test sets
2) Coerce data into: X (data) and Y (targets)

```
# 3) Wait… there's no implementation in sklearn?

import numpy as np

d = degree_of_polynomial    # 1 = linear; 2 = quadratic; etc.
poly_params = np.polyfit(X, Y, d)

# "poly_params" now contains betas
```

Lab 1: best polynomial for DJIA?

- Get data
  - https://github.com/dbrizan/MSAN621-data > djia*csv
  - X: days from 1st Feb, 1987
  - Y: Close
- Goal: find & show best-fitting polynomial
  - Best: d = 1 .. n
  - Using RSS / R^2, ..., (also cross validation)
- Use matplotlib

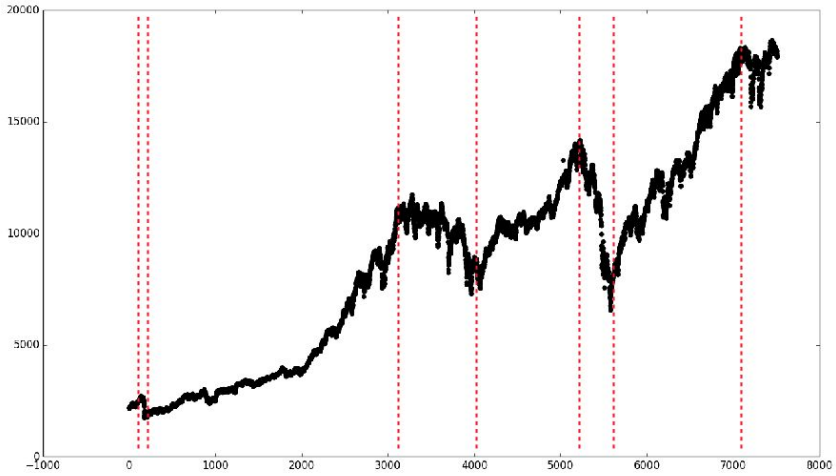
```
# Example for plotting data, targets, fitted polynomial

import matplotlib.pyplot as plt

plt.plot(data, targets, color='black')
plt.plot(data, np.polyval(poly_params, data, 'r-')
plt.show()
```

Polynomial regression - thoughts

- There are d + 1 outputs of polyfit (values for β_i)
- Not widely used for data science models
  - Note: as x —> 0 | x —> 7511, curve looks weird
  - A single global model is a weakness?
  - Note: no implementation in sklearn
- Options to polynomial regression
  - KNN regression
  - Ridge regression, the lasso, etc.
  - Other approaches in this unit
  - Decision trees, SVM, etc.

Step functions

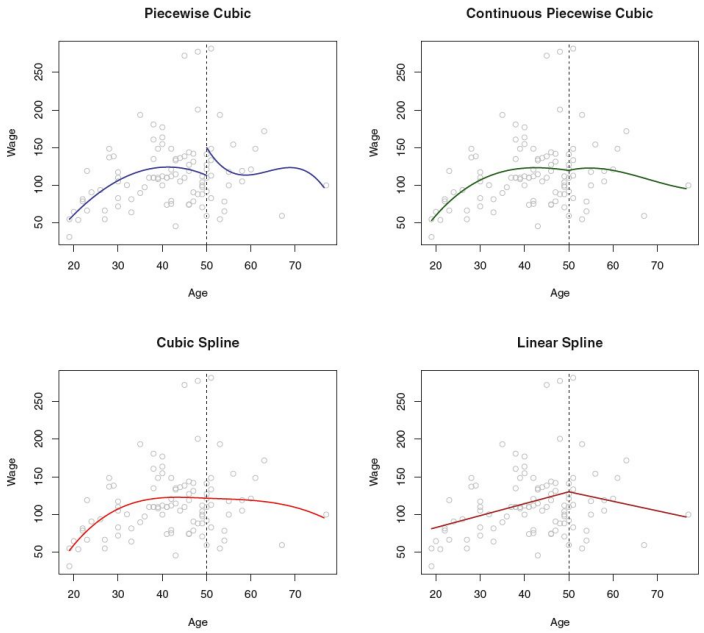
- Reconsider the DJIA data
  - Overall growth trend
  - Areas of growth/retraction (bull/bear) and flat markets



- Task (conceptual) : find regions; model each region
  - Each region: “cut” — “dummy variables”
  - Popular in advertising (“millennial consumers like…”)

Vocabulary & issues

- Knots
  - Knots / cutpoint: edge of a region (cuts)
  - May be implemented as dummy variables
  - How to choose them?
- Region-unique regression?
  - Global model may be a mix of linear / polynomial regressions
  - Better to add constraints to model so it is not discontinuous
  - Spline: function continuous at each knot
    - Linear spline: piecewise linear polynomial continuous at each knot
    - Cubic spline: piecewise polynomial with continuous derivatives up to order 2 at each knot
    - Natural cubic spline: additional smoothing terms



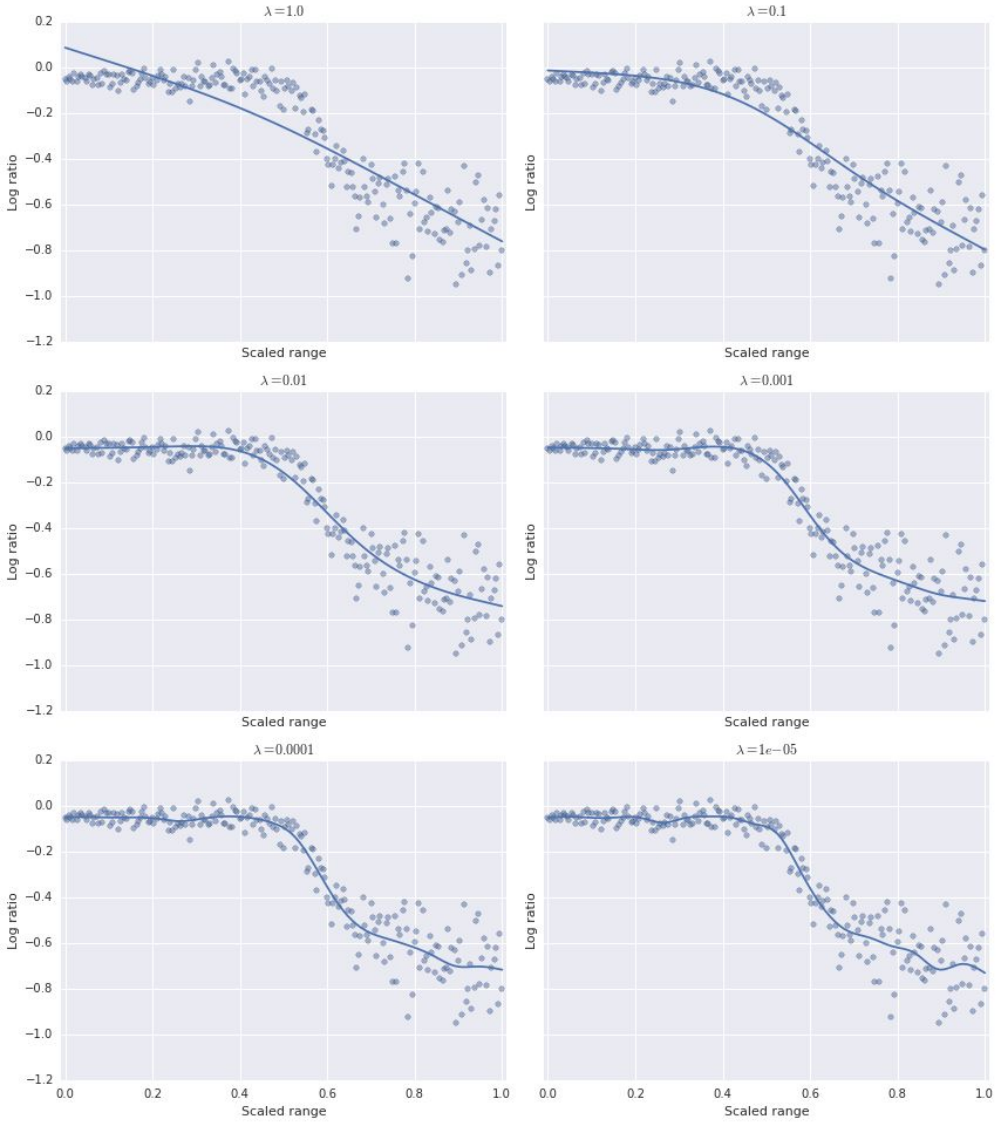
Where should the knots be?

- Decide on K (number of knots) & place at quantiles
  - Problem: this does not describe some data (eg. DJIA) well
  - Choice of quantiles is arbitrary
- Better to use a smoothing spline:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

  - Place knot at each data point
  - First term is RSS — i.e. find a $g(x_i)$ with lowest error relative to y_i
  - Second term is second derivative (acceleration of slope of line)
  - In English: a “roughness” score; the rate at which decreased smoothness is traded for a better fit
  - Score is a penalty (b/c we want to minimize summation)
  - Penalty is controlled by a single hyperparameter **λ** — lower **λ** accepts more wiggles

What should λ be?



- It depends on data
  - Some models for data are better when linear
  - Some values for **λ** overfit the data
- Values for **λ**
  - ∞ = linear regression
  - 0 = sacrifice for a good fit to the data
- Choose a good value using cross-validation

Implementation in python

1) Import data
   a) If necessary, split data into train, test sets
2) Coerce data into: X (data) and Y (targets)

```
# 3) Also no implementation in sklearn, but in scipy

import numpy as np
from scipy.interpolate import UnivariateSpline

smoothing = 5   # In range 0 .. 5
knots = 1   # Smoothing factor for knots, defaults to len(Y)

s = UnivariateSpline(X, Y, k=smoothing, s=knots)
```

4) Use spline, eg.

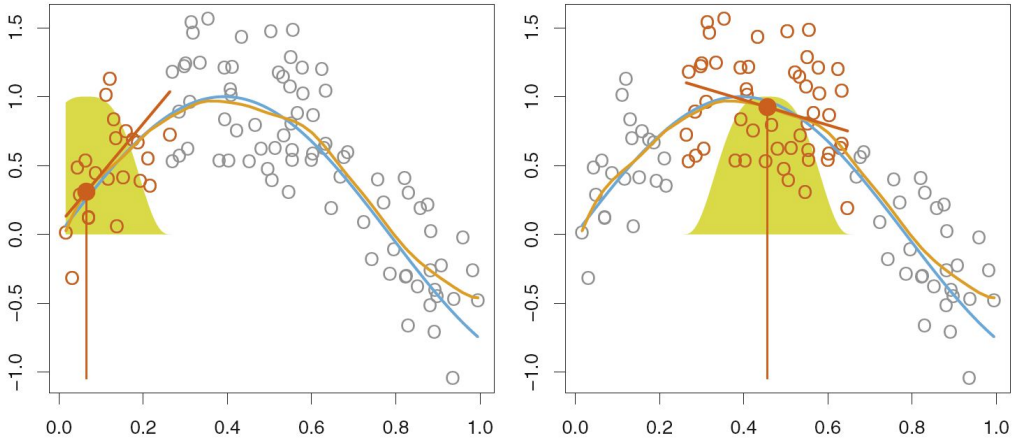
   a) get_coeffs(), get_knots(), get_residual()
   b) Plot: plt.plot(X, s(data))
   c) Continue with smoothing factor from last knot, etc.

Lab 2: fit smoothing spline to DJIA

- Use the same data
  - https://github.com/dbrizan/MSAN621-data > djia*csv
  - X: days from 1st Feb, 1987
  - Y: Close
- Goals:
  - Determine best **λ**
  - Graph best with matplotlib
  - Compare to best polynomial, KNN

Local regression

- Fit data using only local observations
  - Weigh contribution of points so that nearer data contributes more
  - A variant of kNN (i.e. works well with low-dimensional data)
  - Sometimes called "*memory-based procedure*"
- Decisions (hyperparameters)
  - Size of neighbourhood — K ("*span s*") — as fraction of total data
  - How to define weighting of observations
  - Type of fit: linear, quadratic, etc.



Generalized additive models

- Extend the models
  - So far, we've only handled p = 1
  - GAM handles multiple features (X_1, ..., X_p)
- Premise:
  - Extend multiple SLP while maintaining non-linear relationships
  - Replace each linear component with (non-linear?) function $f_j(x_{ij})$

    $y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + ... + + f_p(x_{ip})$

  - Replace each linear component with (non-linear?) function $f_j(x_{ij})$
- May also be used for classification
- Use with caution
  - Implementation in SciPy > statsmodels
  - GAMs exist in R