Lecture 11: Regression, the non-linear case

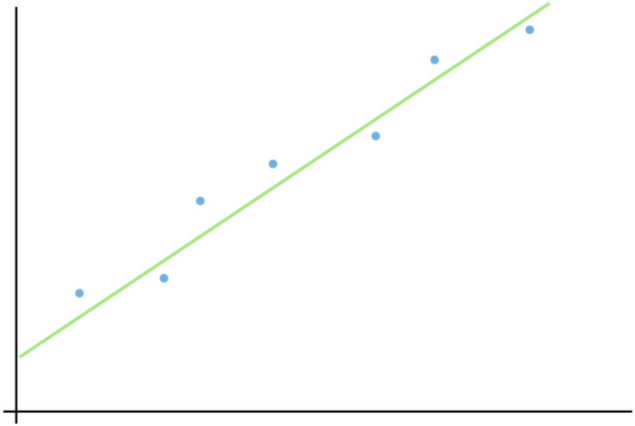
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
August–December 2020

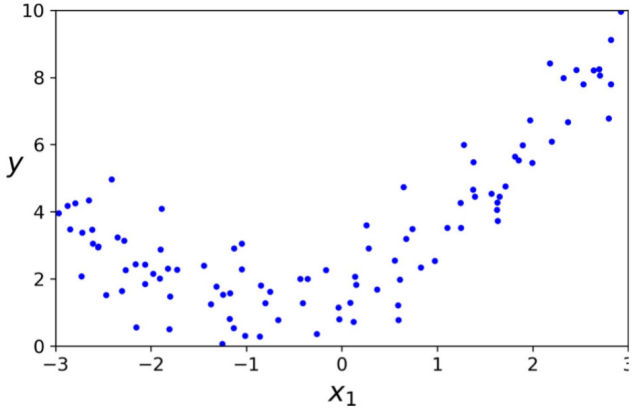
Linear regression

- Find the line that "fits" the data best
  - Normal equation
  - Gradient descent
- Linear: each parameter's contribution is independent
- Input $x : (x_1, x_2, \ldots, x_k)$
- $y = \theta_0 + \theta_1 x_1 + \cdots + \theta_k x_k$



The non-linear case

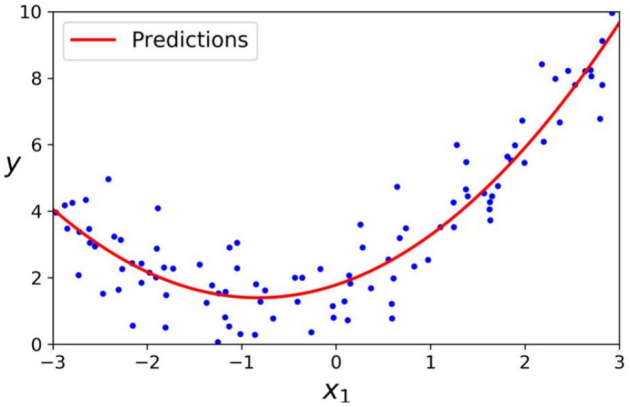
- What if the relationship is not linear?



The non-linear case

- What if the relationship is not linear?
- Here the best possible explanation seems to be a quadratic
- Non-linear : cross dependencies
- Input $x_i : (x_{i_1}, x_{i_2})$
- Quadratic dependencies:

$$y = \theta_0 + \theta_1 x_{i_1} + \theta_2 x_{i_2} + \theta_{11} x_{i_1}^2 + \theta_{22} x_{i_2}^2 + \theta_{12} x_{i_1} x_{i_2}$$



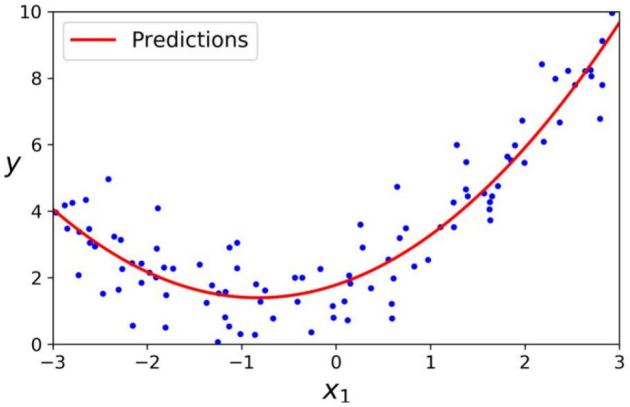
The non-linear case

- Recall how we fit a line

$$\begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

- For quadratic, add new coefficients and expand parameters

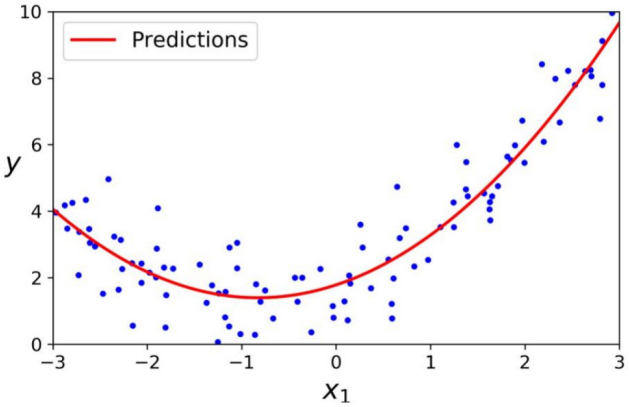
$$\begin{bmatrix} 1 & x_i & x_i^2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$



The non-linear case

- Input (x_{i_1}, x_{i_2})
- For the general quadratic case, we are adding new derived "features"

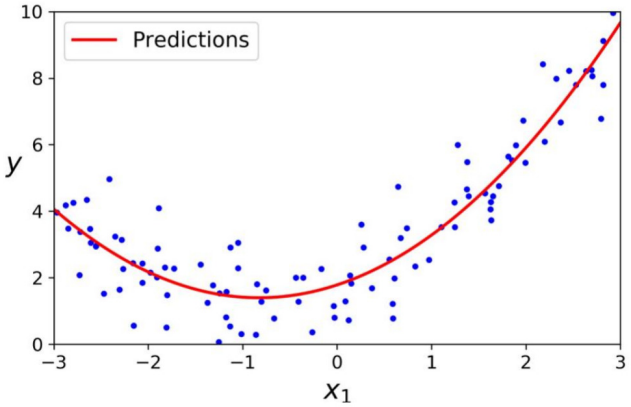
$$
\begin{aligned}
x_{i_3} &= x_{i_1}^2 \\
x_{i_4} &= x_{i_2}^2 \\
x_{i_5} &= x_{i_1} x_{i_2}
\end{aligned}
$$



The non-linear case

- Original input matrix

$$\begin{bmatrix} 1 & x_{1_1} & x_{1_2} \\ 1 & x_{2_1} & x_{2_2} \\ & \cdots & \\ 1 & x_{i_1} & x_{i_2} \\ & \cdots & \\ 1 & x_{n_1} & x_2 \end{bmatrix}$$

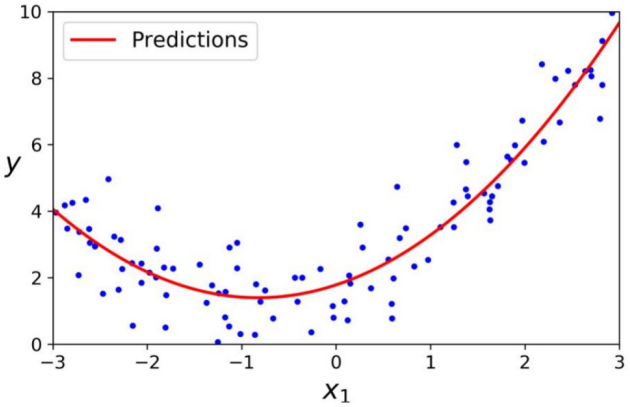


The non-linear case

- Expanded input matrix

$$\begin{bmatrix} 1 & x_{1_1} & x_{1_2} & x_{1_1}^2 & x_{1_2}^2 & x_{1_1}x_{1_2} \\ 1 & x_{2_1} & x_{2_2} & x_{2_1}^2 & x_{2_2}^2 & x_{2_1}x_{2_2} \\ & \cdots & & & & \\ 1 & x_{i_1} & x_{i_2} & x_{i_1}^2 & x_{i_2}^2 & x_{i_1}x_{i_2} \\ & \cdots & & & & \\ 1 & x_{n_1} & x_{n_2} & x_{n_1}^2 & x_{n_2}^2 & x_{n_1}x_{n_2} \end{bmatrix}$$

- New columns are computed and filled in from original inputs



Exponential parameter blow-up

- Cubic derived features

$x_{i_1}^3$, $x_{i_2}^3$, $x_{i_3}^3$,

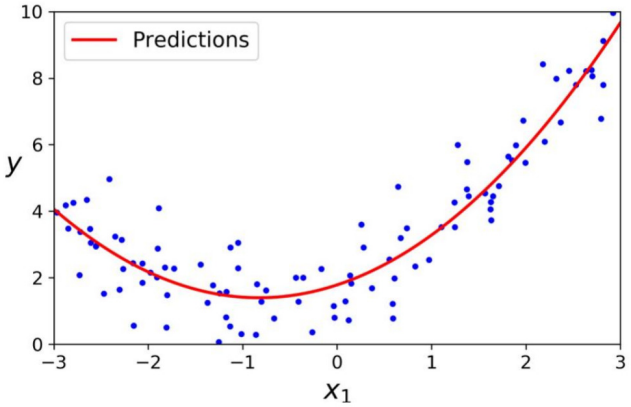
$x_{i_1}^2 x_{i_2}$, $x_{i_1}^2 x_{i_3}$,
$x_{i_2}^2 x_{i_1}$, $x_{i_2}^2 x_{i_3}$,
$x_{i_3}^2 x_{i_1}$, $x_{i_3}^2 x_{i_2}$,

$x_{i_1} x_{i_2} x_{i_3}$,

$x_{i_1}^2$, $x_{i_2}^2$, $x_{i_3}^2$,

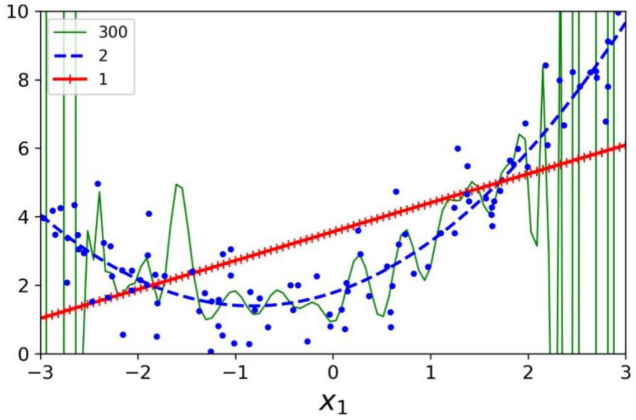
$x_{i_1} x_{i_2}$, $x_{i_1} x_{i_3}$, $x_{i_2} x_{i_3}$,

x_{i_1}, x_{i_2}, x_{i_3}.



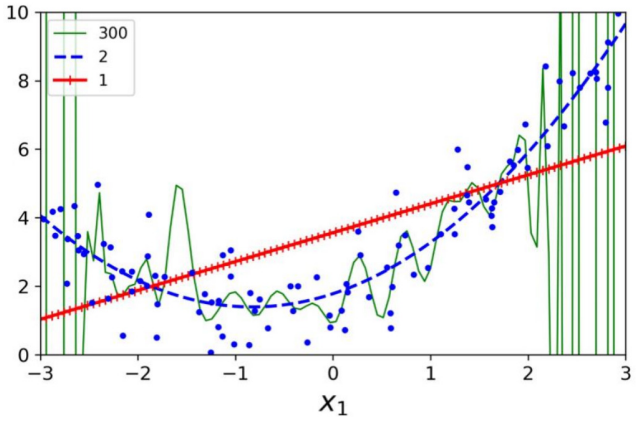
Higher degree polynomials

- How complex a polynomial should we try?
- Aim for degree that minimizes SSE
- As degree increases, features explode exponentially



Overfitting

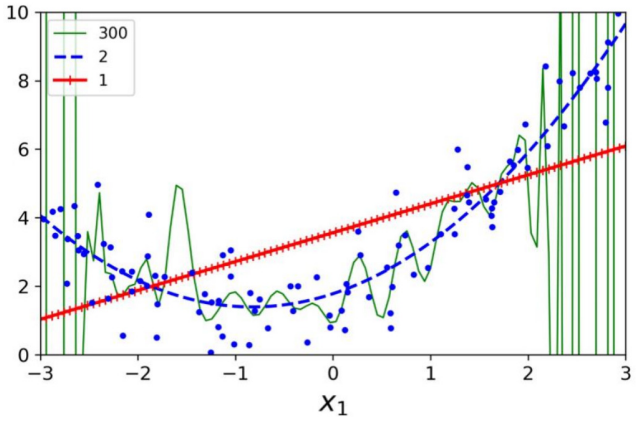
- Need to be careful about adding higher degree terms
- For n training points,can always fit polynomial of degree $(n-1)$ exactly
- However, such a curve would not generalize well to new data points
- Overfitting — model fits training data well, performs poorly on unseen data



Regularization

- Need to trade off SSE against curve complexity
- So far, the only cost has been SSE
- Add a cost related to parameters $(\theta_0, \theta_1, \ldots, \theta_k)$
- Minimize, for instance

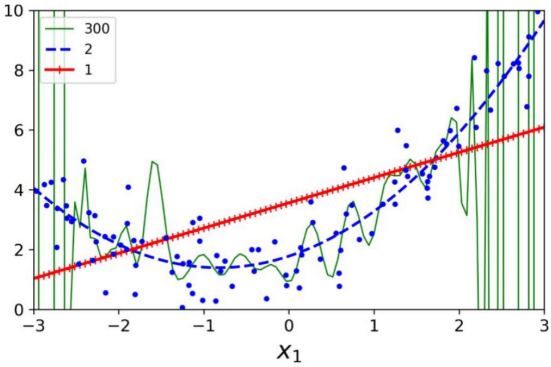
$$\frac{1}{2}\sum_{i=1}^{n}(z_i - y_i)^2 + \sum_{j=1}^{k}\theta_j^2$$



Regularization

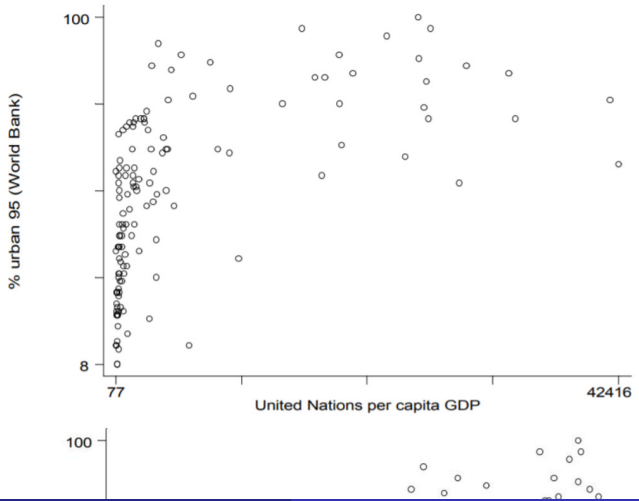
$$\frac{1}{2}\sum_{i=1}^{n}(z_i - y_i)^2 + \sum_{j=1}^{k}\theta_j^2$$

- Second term penalizes curve complexity
- Variations on regularatization
  - Ridge regression: $\sum_{j=1}^{k}\theta_j^2$
  - LASSO regression: $\sum_{j=1}^{k}|\theta_j|$
  - Elastic net regression: $\sum_{j=1}^{k}\lambda_1|\theta_j| + \lambda_2\theta_j^2$



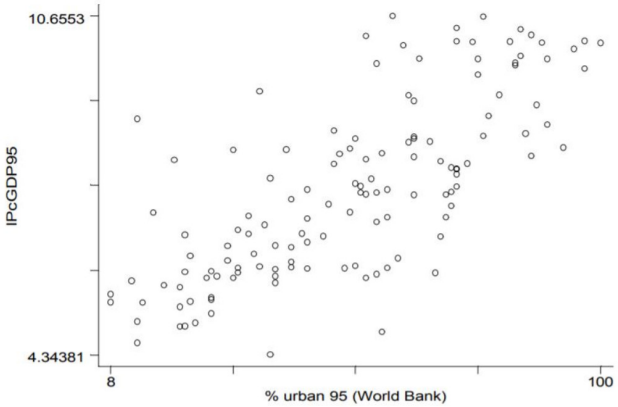
The non-polynomial case

- Percentage of urban population as a function of per capita GDP
- Not clear what polynomial would be reasonable
- Take log of GDP
- Regression we are computing is $y = \theta_0 + \theta_1 \log x_1$



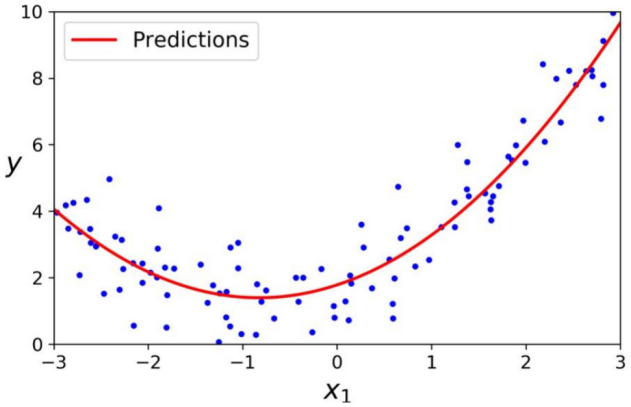
The non-polynomial case

- Reverse the relationship
- Plot per capita GDP in terms of percentage of urbanization
- Now we take log of the output variable
  $\log y = \theta_0 + \theta_1 x_1$
- Log-linear transformation
- Earlier was linear-log
- Can also use log-log



Summary

- Dealing with non-linear dependencies
- Add derived features — higher degree terms
  - Exponential blowup in number of features
- Danger of overfitting
  - Regularization — add penalty for curve complexity
- Logarithmic transformation of input/output



$x_{i_1}^3$, $x_{i_2}^3$, $x_{i_3}^3$,