

Lecture 7: 29 January, 2026

Madhavan Mukund

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

Data Mining and Machine Learning
January–April 2026

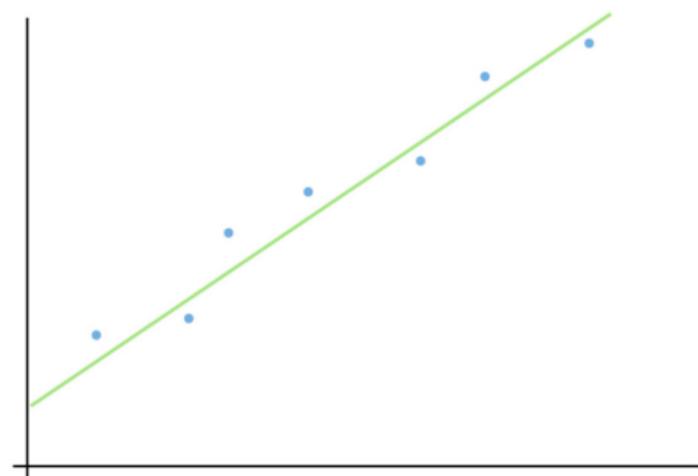
Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output
- How far away is our prediction $h_\theta(x_i)$ from the true answer y_i ?

- Define a cost (loss) function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^n (h_\theta(x_i) - y_i)^2$$

- Minimize $J(\theta)$
- Why do we use squared error for loss?



Regression and SSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Outputs are noisy samples from a linear function
 - $y_i = \theta^T x_i + \epsilon$
 - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
 - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \theta^T x_i$
- Model gives us an estimate for θ , so regression learns μ_i for each x_i
- How good is our estimate?
- **Likelihood** — probability of current observation given θ

$$\mathcal{L}(\theta) = \prod_{i=1}^n P(y_i | x_i; \theta)$$

- How good is our estimate?
- Want **Maximum Likelihood Estimator (MLE)**

- Find θ that maximizes $\mathcal{L}(\theta) = \prod_{i=1}^n P(y_i | x_i; \theta)$

- Equivalently, maximize **log likelihood**

$$\ell(\theta) = \log \left(\prod_{i=1}^n P(y_i | x_i; \theta) \right) = \sum_{i=1}^n \log(P(y_i | x_i; \theta))$$

- Easier to work with summation than product

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i | x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}}$

- Log likelihood (assuming natural logarithm)

$$\ell(\theta) = \sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}} \right) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^n \frac{(y_i - \theta^T x_i)^2}{2\sigma^2}$$

- To maximize $\ell(\theta)$ with respect to θ , ignore all terms that do not depend on θ

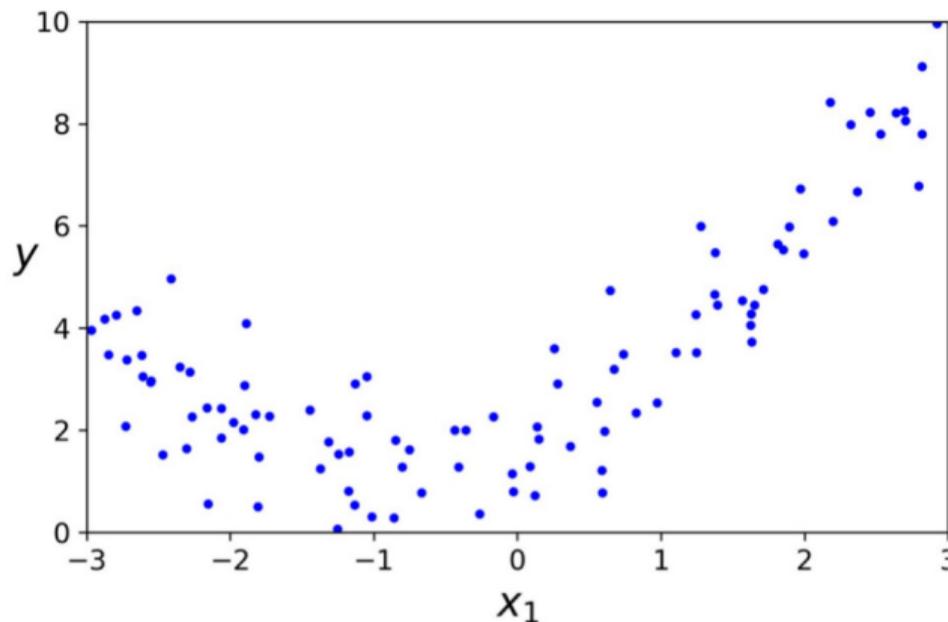
- Optimum value of θ is given by

$$\hat{\theta}_{\text{MSE}} = \arg \max_{\theta} \left[- \sum_{i=1}^n (y_i - \theta^T x_i)^2 \right] = \arg \min_{\theta} \left[\sum_{i=1}^n (y_i - \theta^T x_i)^2 \right]$$

- Assuming data points are generated by linear function and then perturbed by Gaussian noise, SSE is the “correct” loss function to maximize likelihood

The non-linear case

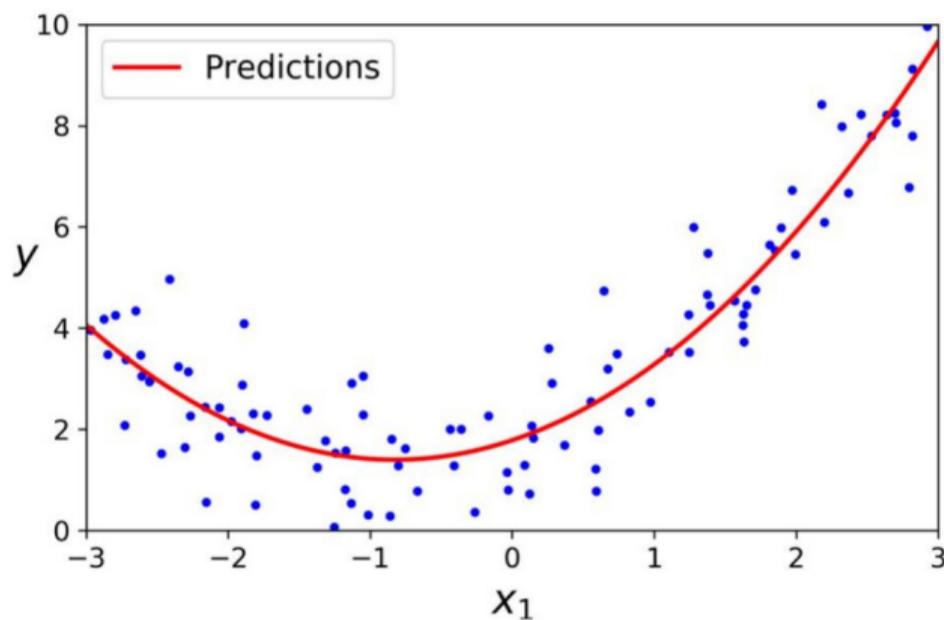
- What if the relationship is not linear?
- Here the best possible explanation seems to be a quadratic



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:

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



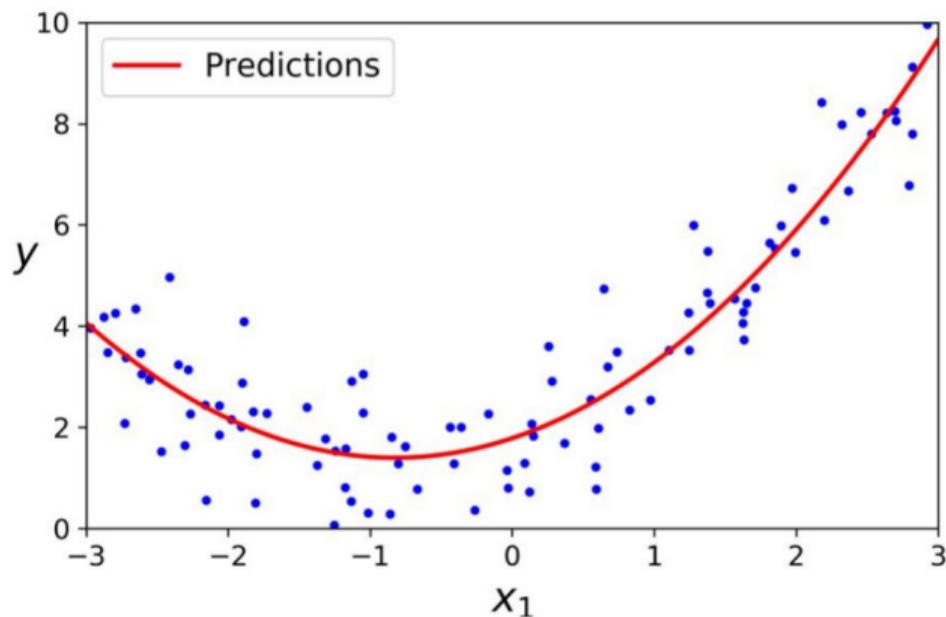
The non-linear case

- 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

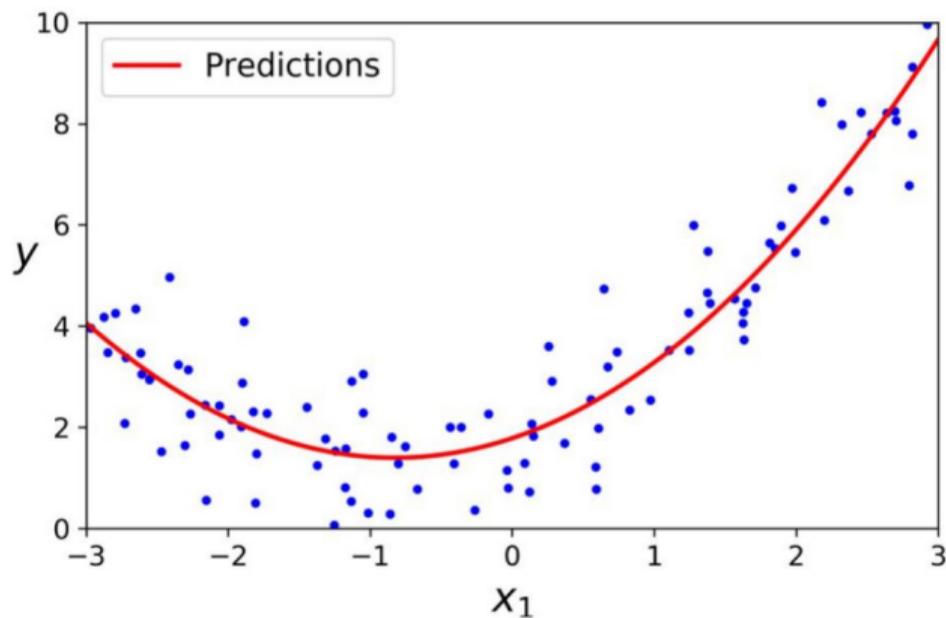
$$\begin{bmatrix} 1 & x_i & x_i^2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$



The non-linear case

- Input (x_{i_1}, x_{i_2})
- For the general quadratic case, we add new derived “features”

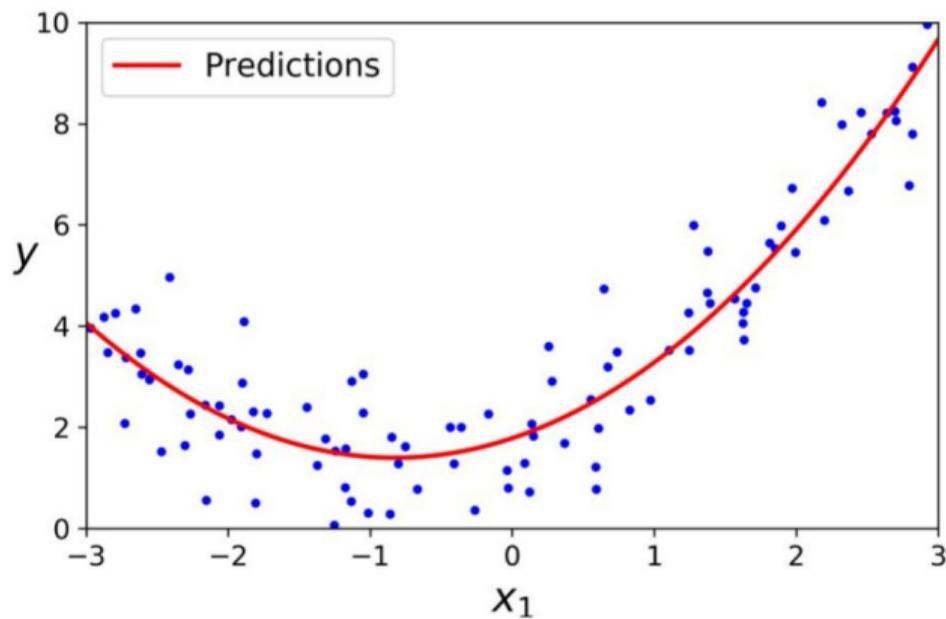
$$x_{i_3} = x_{i_1}^2$$
$$x_{i_4} = x_{i_2}^2$$
$$x_{i_5} = x_{i_1} x_{i_2}$$



The non-linear case

- Original input matrix

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

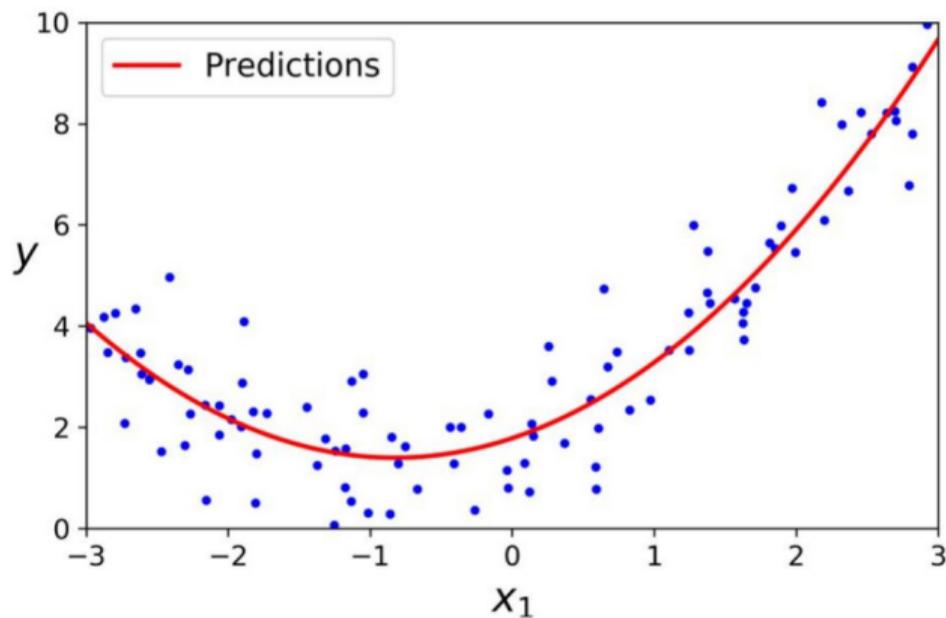


The non-linear case

- Expanded input matrix

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

- New columns are computed and filled in from original inputs



Exponential parameter blow-up

■ Cubic derived features

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

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

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

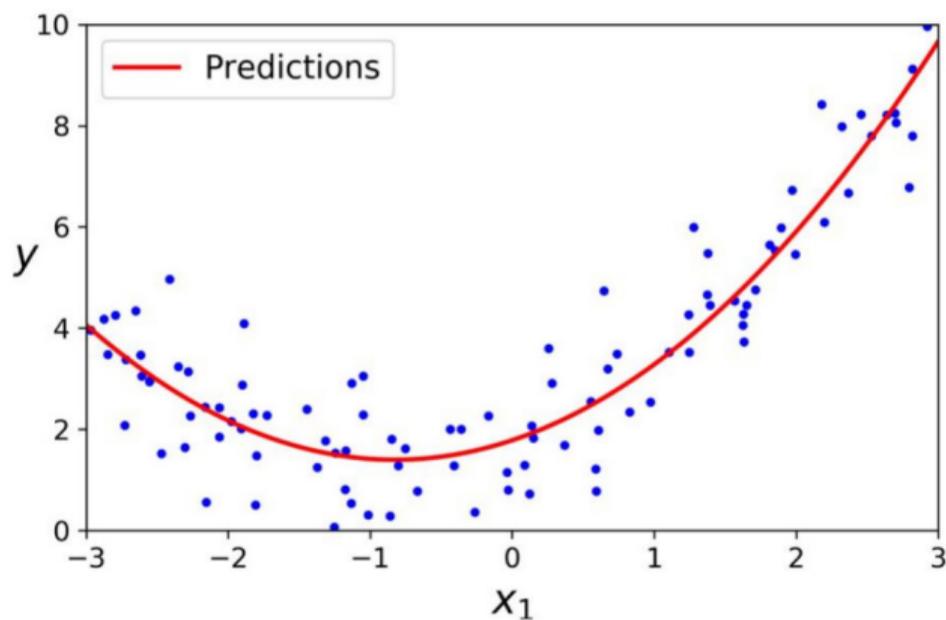
$$x_{i_3}^2 x_{i_1}, x_{i_3}^2 x_{i_2},$$

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

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

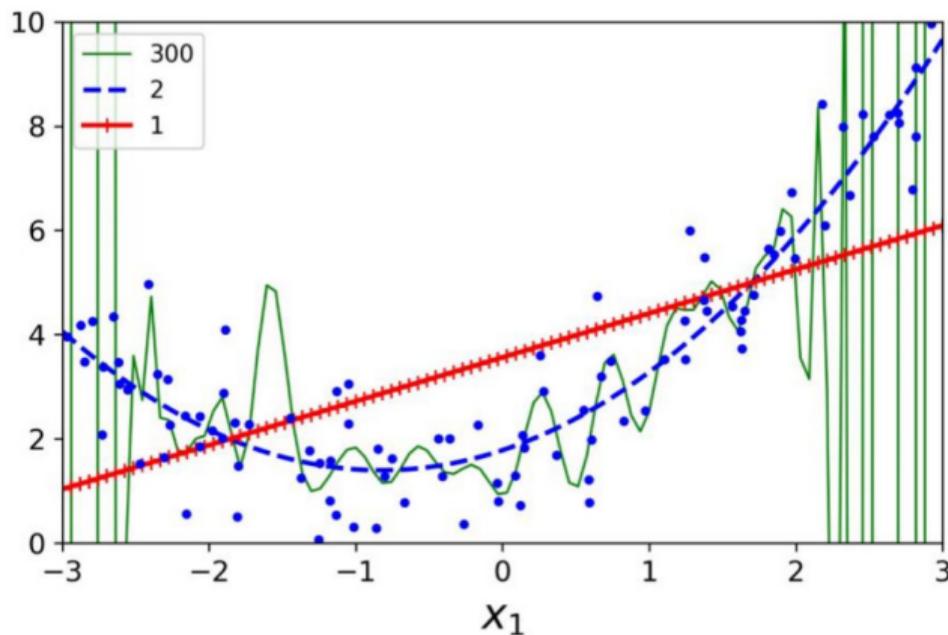
$$x_{i_1} x_{i_2}, x_{i_1} x_{i_3}, x_{i_2} x_{i_3},$$

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



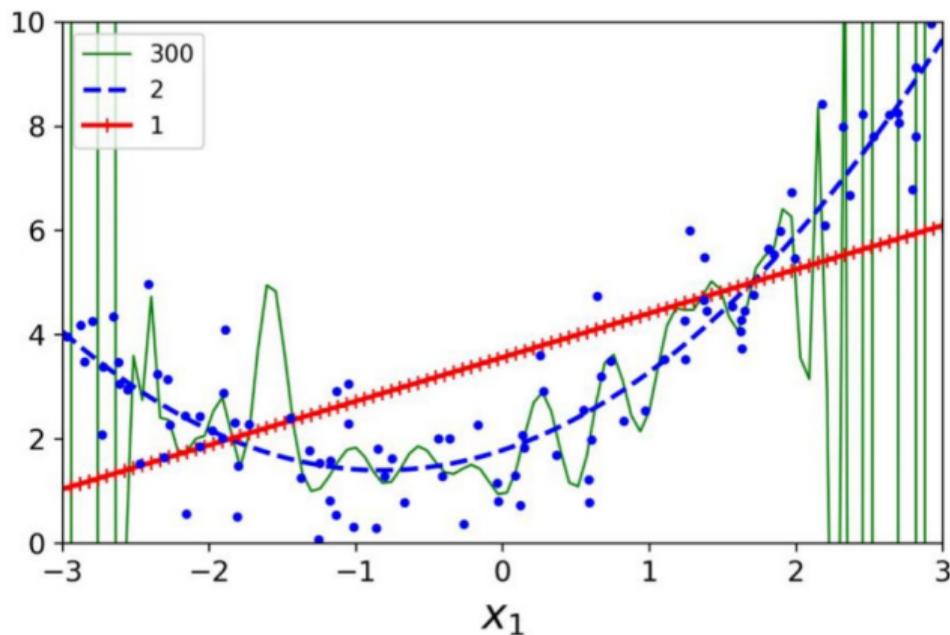
Higher degree polynomials

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



Overfitting

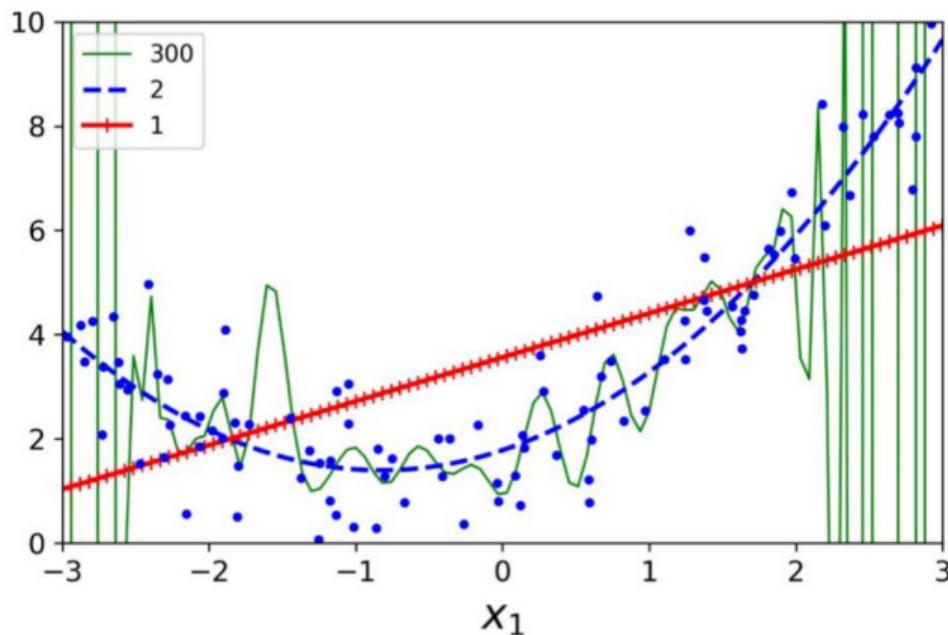
- Need to be careful about adding higher degree terms
- For n training points, can always fit polynomial of degree $(n - 1)$ exactly
- However, such a curve would not generalize well to new data points
- **Overfitting** — model fits training data well, performs poorly on unseen data



Regularization

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

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



Regularization

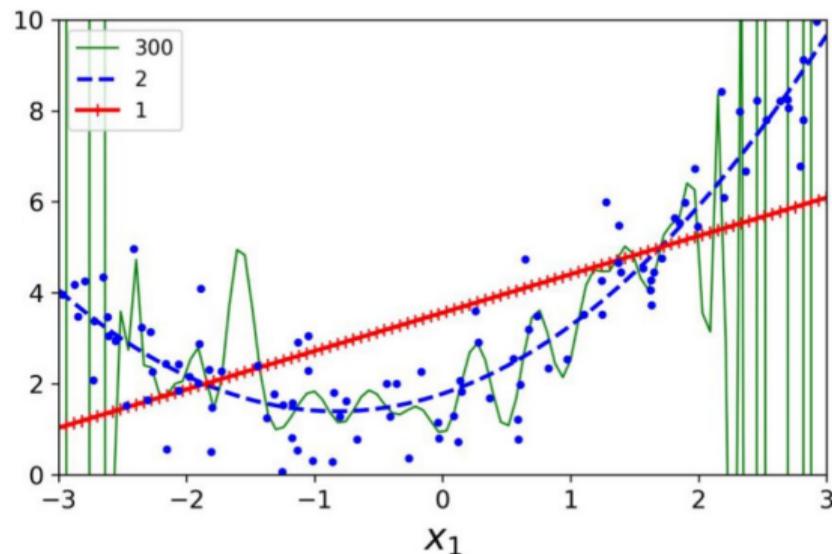
$$\frac{1}{2} \sum_{i=1}^n (z_i - y_i)^2 + \sum_{j=1}^k \theta_j^2$$

- Second term penalizes curve complexity
- Variations on regularization

- Ridge regression: $\sum_{j=1}^k \theta_j^2$

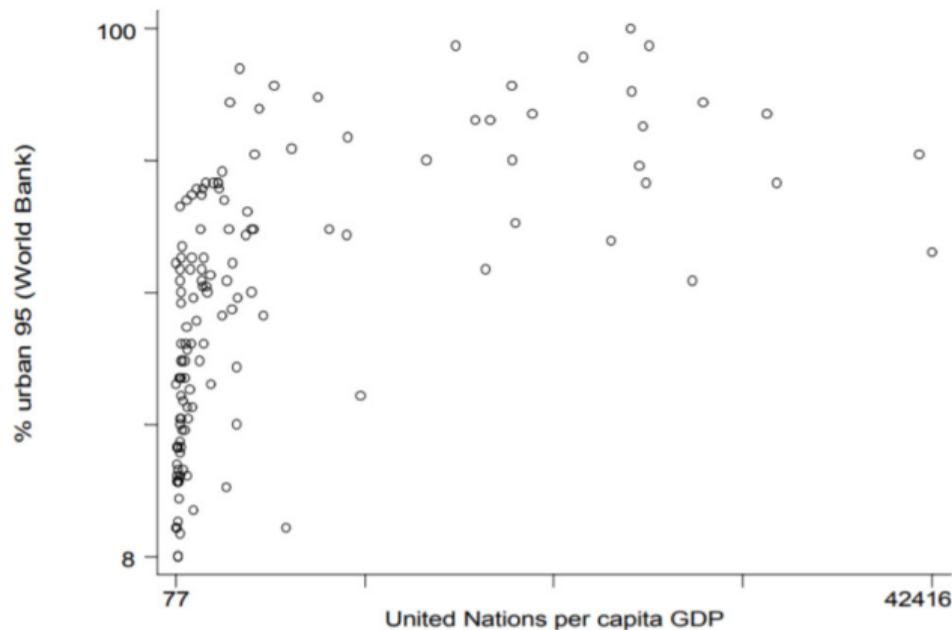
- LASSO regression: $\sum_{j=1}^k |\theta_j|$

- Elastic net regression: $\sum_{j=1}^k \lambda_1 |\theta_j| + \lambda_2 \theta_j^2$



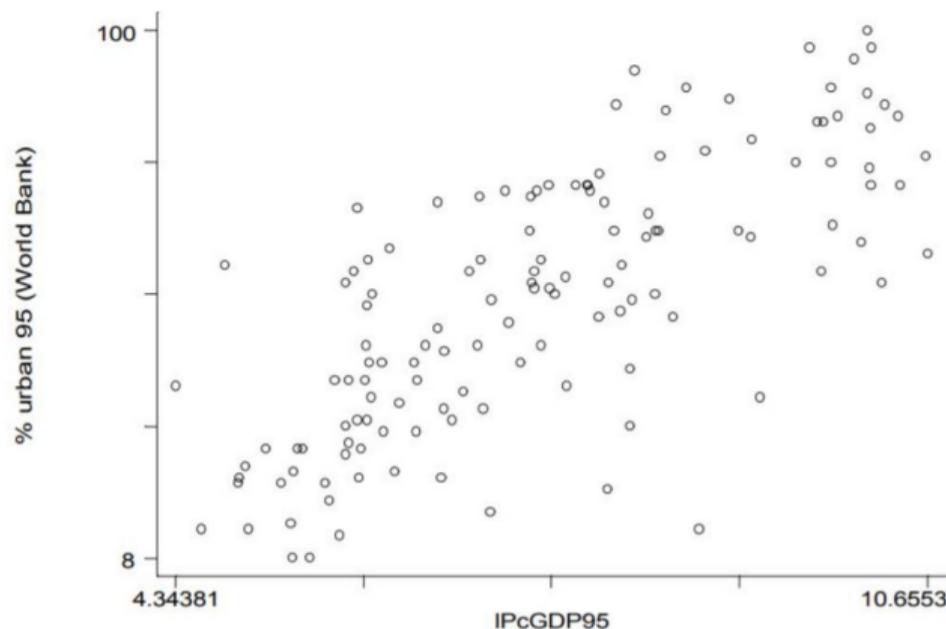
The non-polynomial case

- 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

- 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

