DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 9 2021

Announcements

- HW2 is on Gradescope and Piazza, due on Friday, February 19
- Project resources on Piazza
- Project proposal due on March 4

Outline

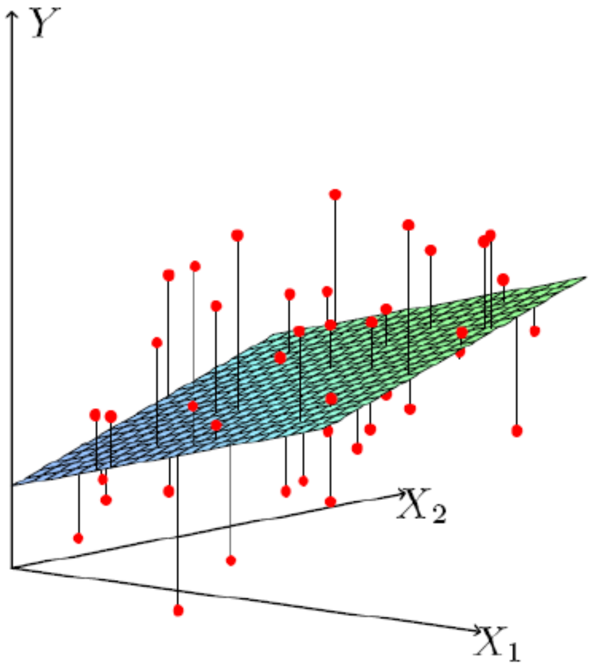
- Gradient Descent comparison with closed-form solution for linear regression
- Regularization
  - Ridge regression and gradient descent update
  - Lasso regression
- Classification
  - K Nearest Neighbors (kNN)
  - Bias-Variance tradeoff

Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{N}\sum (\theta^T x_i - y_i)^2$ Loss / cost

$$\theta = (X^\top X)^{-1} X^\top y$$

MSE is a strictly convex function and has unique minimum



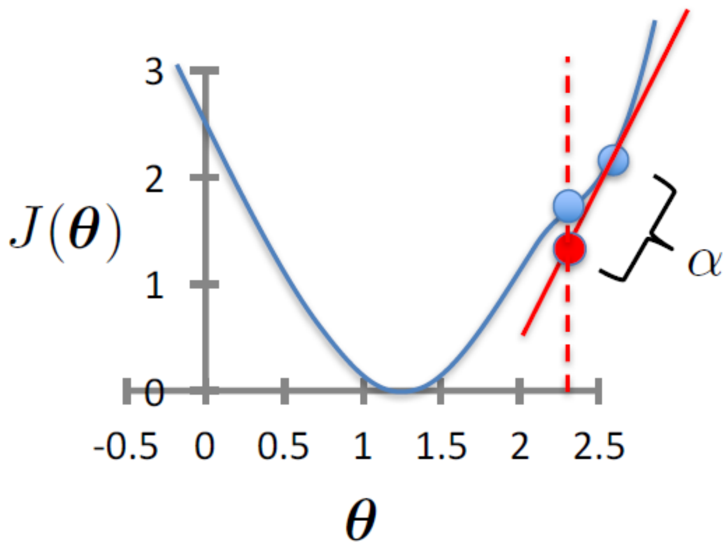
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



Vector update rule: $\theta \leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$

GD for Linear Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence $||\theta_{new} - \theta_{old}|| < \epsilon$ or iterations == MAX_ITER

$$\theta_j \leftarrow \theta_j - \alpha \frac{2}{N} \sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_{ij}$$

simultaneous update for j = 0 ... d

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_\theta(x_i)$
  - Use this stored value in the update step loop
- Assume convergence when $\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$

L_2 norm: $\|\boldsymbol{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$

Gradient Descent in Practice

- Asymptotic complexity
  - N is size of training data, d is feature dimension, and T is number of iterations
- Most popular optimization algorithm in use today
- At the basis of training
  - Linear Regression
  - Logistic regression
  - SVM
  - Neural networks and Deep learning
  - Stochastic Gradient Descent variants

Gradient Descent vs Closed Form

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update for j = 0 ... d

Closed form

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

- **Gradient Descent**

+ Linear increase in d and N

+ Generally applicable

- Need to choose α and stopping conditions
- Might get stuck in local optima

- **Closed Form**

+ No parameter tuning

+ Gives the global optimum

- Not generally applicable
- Slow computation:

Issues with Gradient Descent

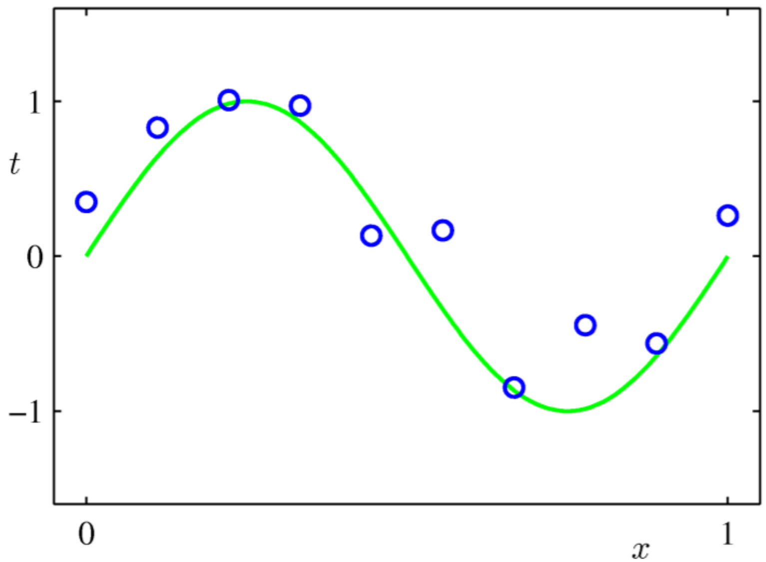
- Might get stuck in local optimum and not converge to global optimum
  - Restart from multiple initial points
- Only works with differentiable loss functions
- Small or large gradients
  - Feature scaling helps
- Tune learning rate
  - Can use line search for determining optimal learning rate

Review Gradient Descent

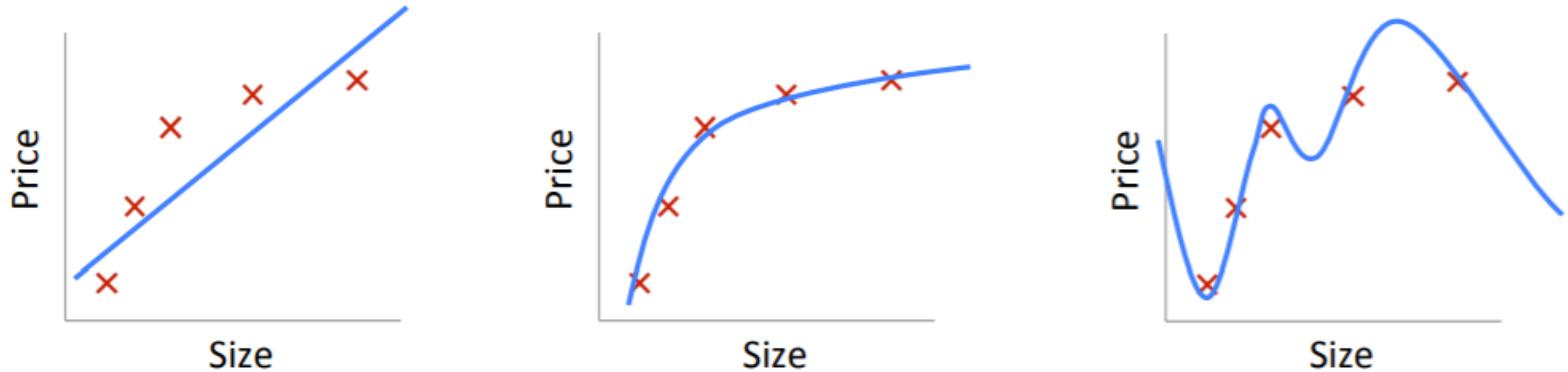
- Gradient descent is an efficient algorithm for optimization and training ML models
  - The most widely used algorithm in ML!
  - Much faster than using closed-form solution for linear regression
  - Main issues with Gradient Descent is convergence and getting stuck in local optima (for neural networks)
- Gradient descent is guaranteed to converge to optimum for strictly convex functions if run long enough

Polynomial Regression

- Polynomial function on single feature
  - $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_p x^p$



Polynomial Regression



$\theta_0 + \theta_1 x$

Underfitting
(high bias)

$\theta_0 + \theta_1 x + \theta_2 x^2$

Correct fit

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfitting
(high variance)

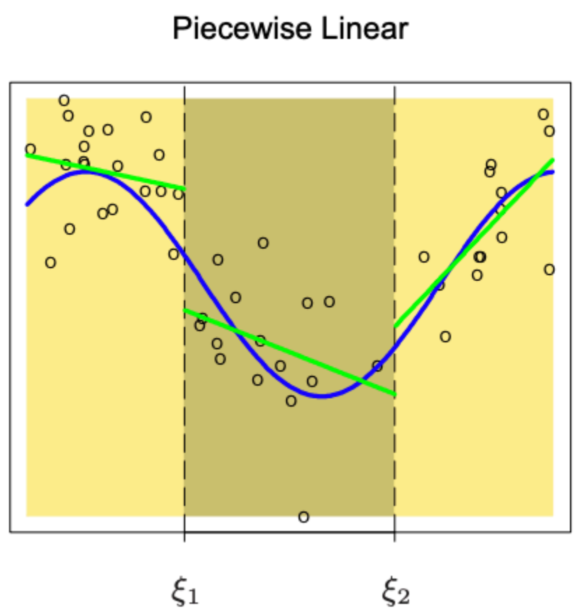
- Typically to avoid overfitting $d \leq 4$

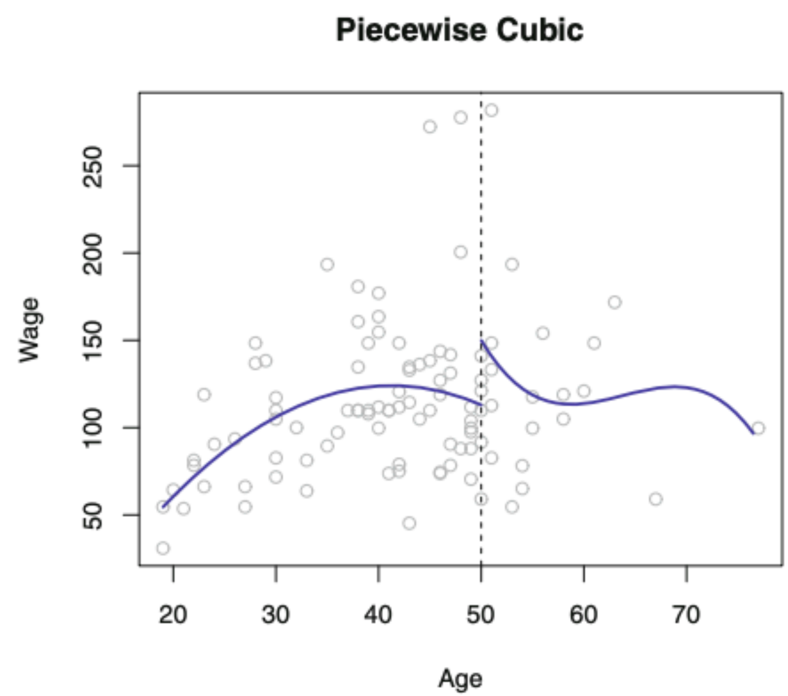
Polynomial Regression Training

- Simple Linear Regression
- $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_p x^p$
- How to train model?

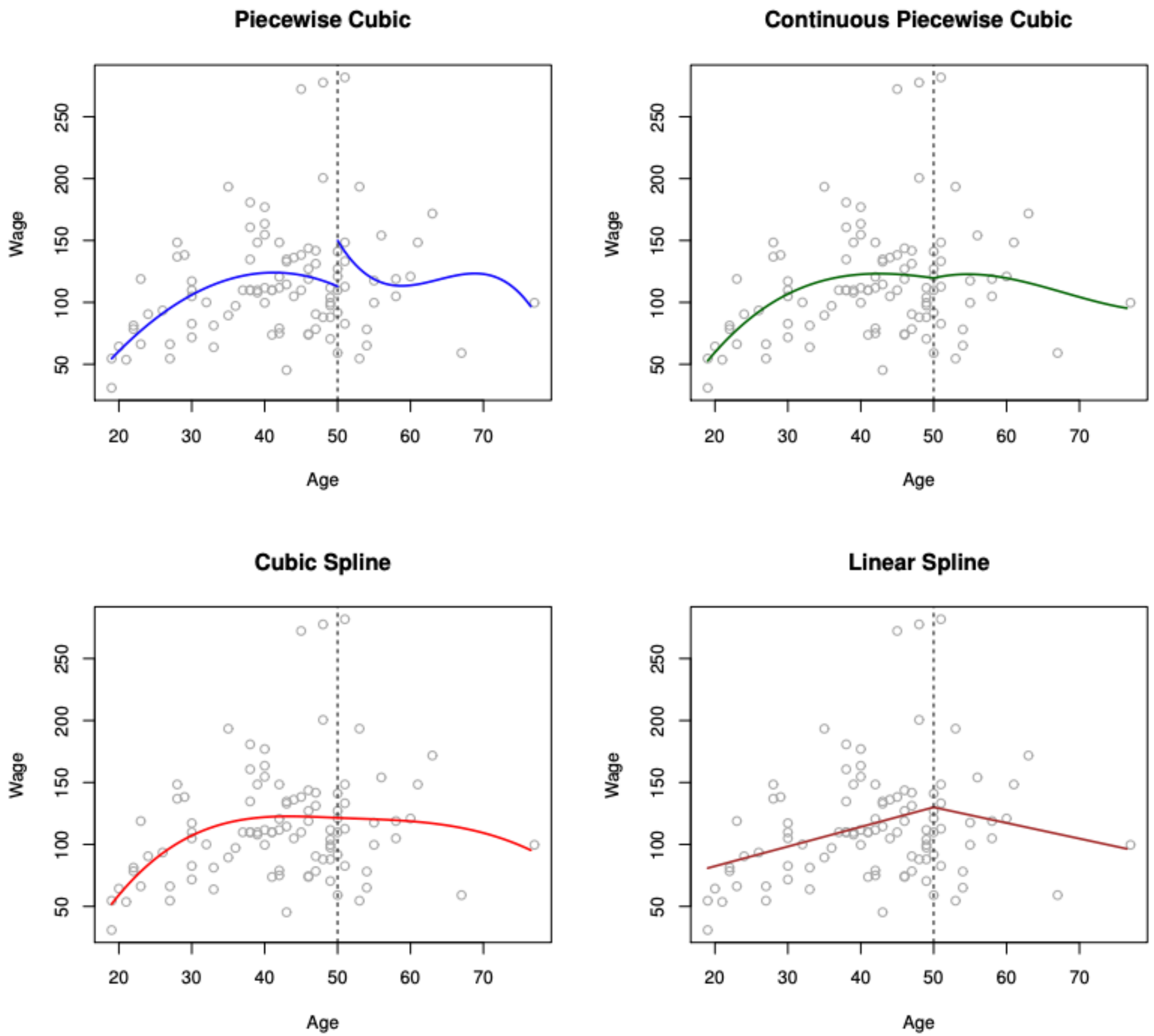
Piecewise Polynomial

- Divide the space into regions
- Polynomial regression on each region
  - Linear piecewise (degree 1), quadratic piecewise (degree 2), cubic piecewise (degree 3)

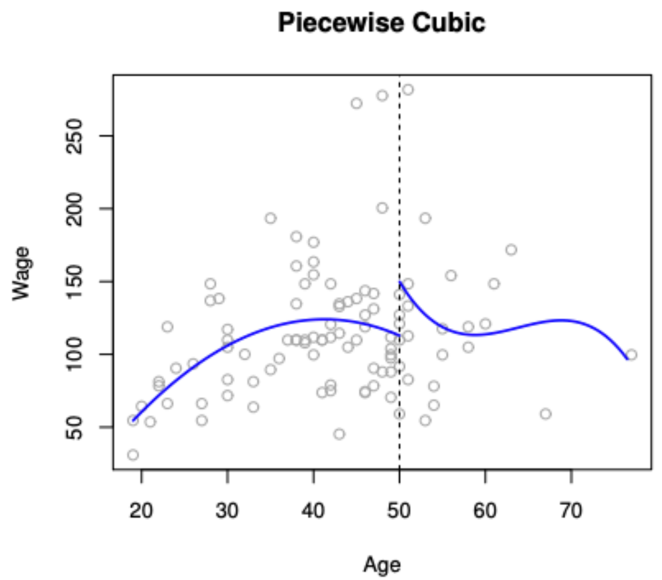


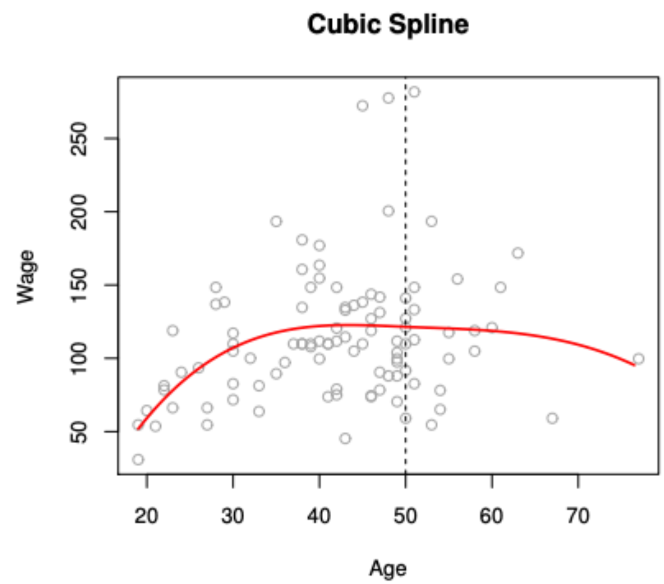


Piecewise and spline regression



Piecewise polynomial vs Regression spline





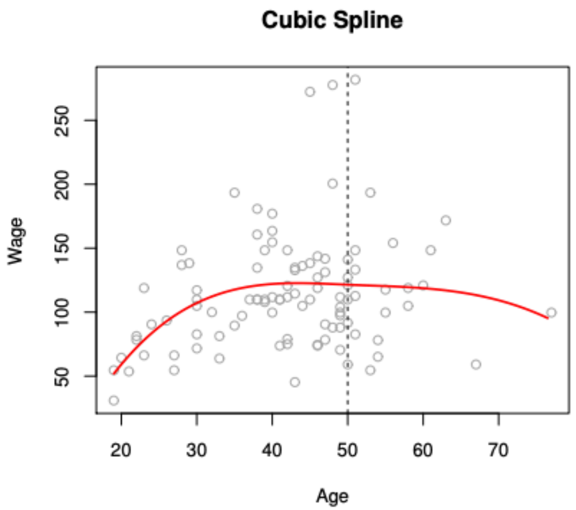
1 break at Age $= 50$

1 knot at Age $= 50$

Definition: Cubic spline

A cubic spline with knots at x-values $\xi_1, \ldots, \xi_K$ is a continuous piecewise cubic polynomial with *continuous derivates* and *continuous second derivatives* at each knot.

Cubic splines



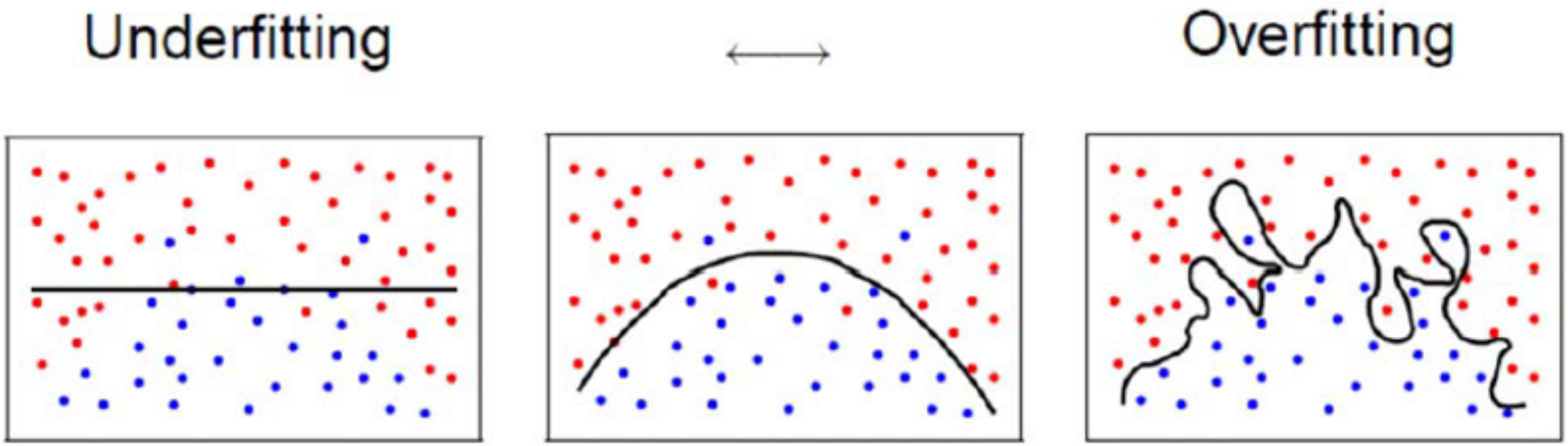
- Turns out, cubic splines are sufficiently flexible to *consistently* estimate smooth regression functions f
- You can use higher-degree splines, but *there's no need to*
- To fit a cubic spline, we just need to pick the knots

A cubic spline with K knots has K+3 free parameters

Additive Models

- Multiple Linear Regression Model
  - $y_i = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$
- Additive Models
  - $y_i = \theta_0 + f_1(x_1) + \cdots + f_d(x_d)$
- Can instantiate functions f with:
  - Linear functions: $f_i(x_i) = \theta_i x_i$
  - Quadratic: $f_i(x_i) = \theta_i^1 x_i + \theta_i^2 x_i^2$
  - Cubic: $f_i(x_i) = \theta_i^1 x_i + \theta_i^2 x_i^2 + \theta_i^3 x_i^3$

Generalization in ML



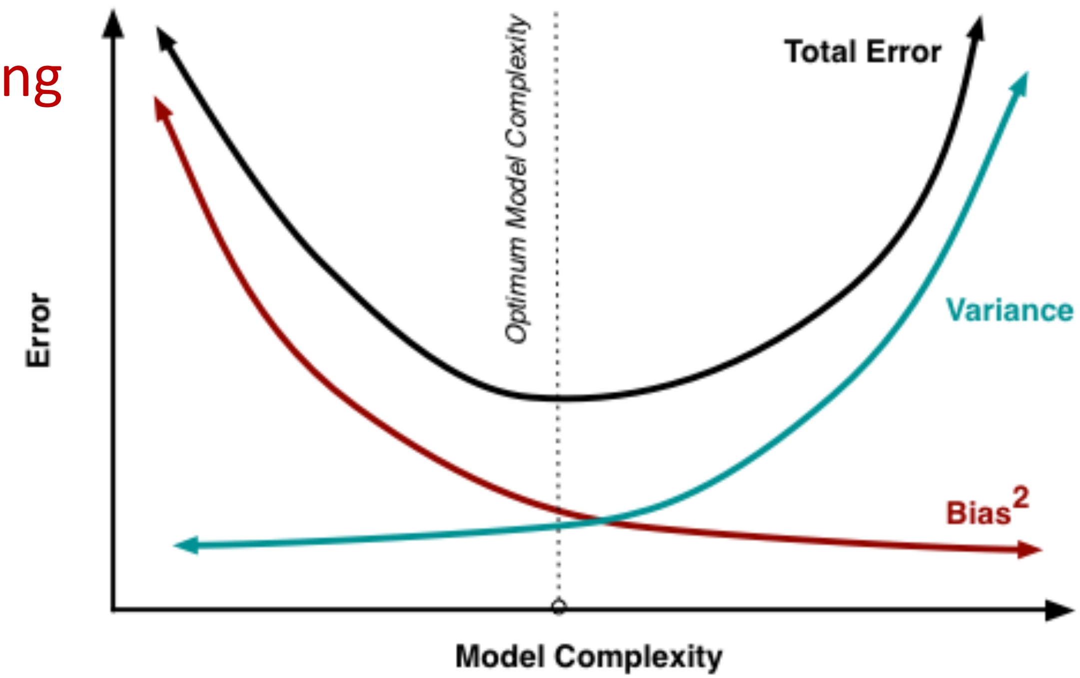
Simple model

Complex model

- Goal is to generalize well on new testing data
- Risk of overfitting to training data

Bias-Variance Tradeoff

Under-fitting



Over-fitting

- Bias = Difference between estimated and true models
- Variance = Model difference on different training sets

MSE is proportional to Bias + Variance

Regularization

- A method for automatically controlling the complexity of the learned hypothesis

- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label

- Can also address overfitting by eliminating features (either manually or via model selection)

Reduce model complexity
Reduce model variance

Ridge regression

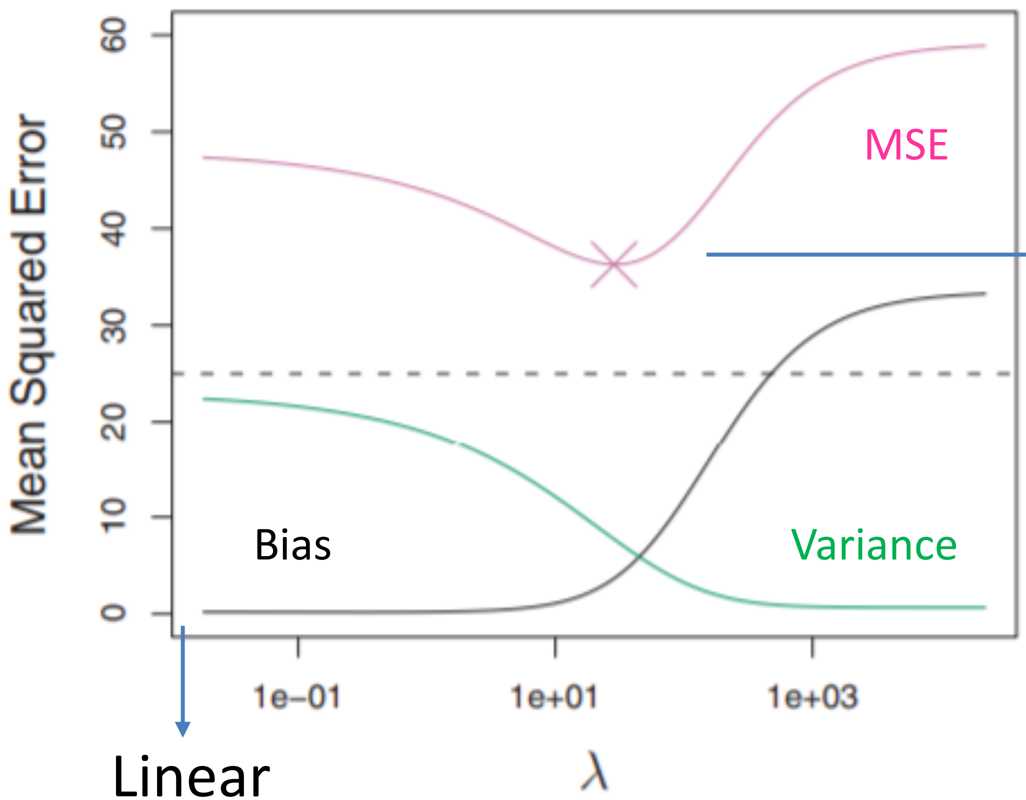
- Linear regression objective function

$$J(\theta) = \underbrace{\frac{1}{2}\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{model fit to data}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{d} \theta_j^2}_{\text{regularization}}$$

  – λ is the regularization parameter ($\lambda \geq 0$)
  – No regularization on θ_0!

    - If λ = 0, we train linear regression
    - If λ is large, the coefficients will shrink close to 0

Bias-Variance Tradeoff

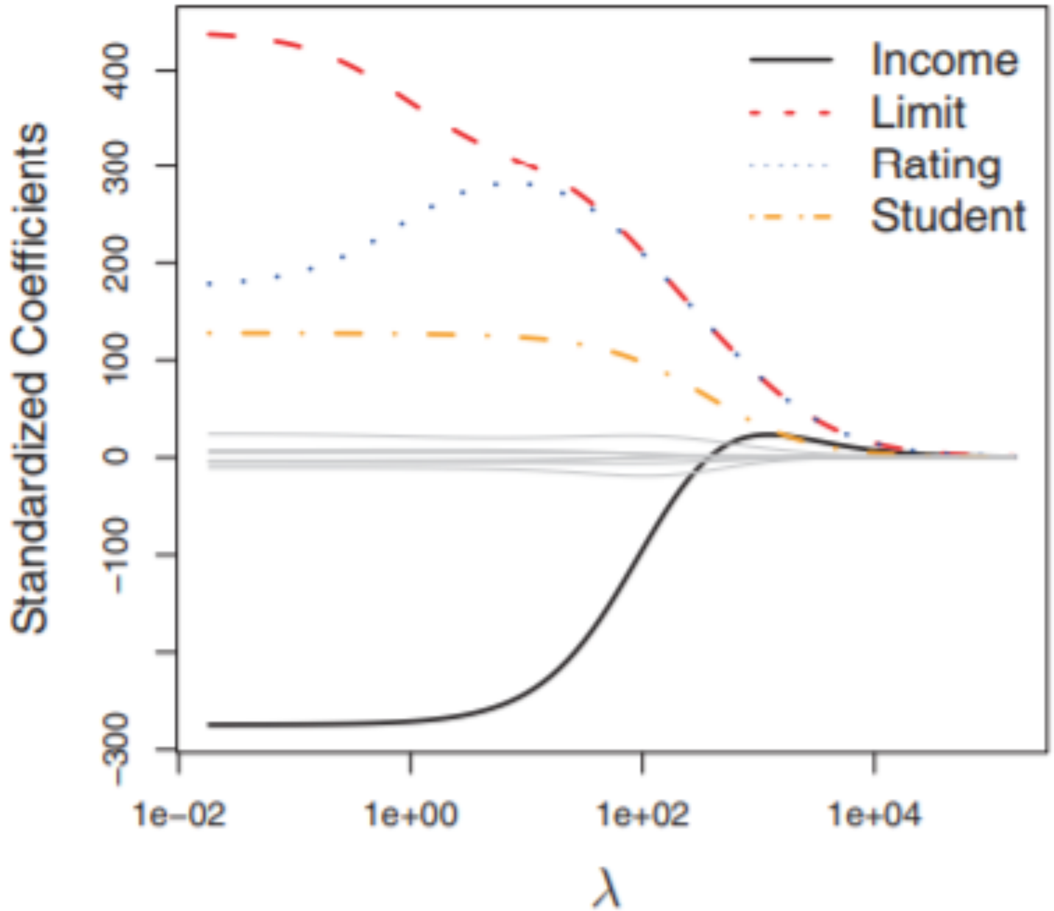


Optimal
Ridge regression

Linear
regression

Reduced model
complexity

Coefficient shrinkage



Predict credit card balance

GD for Ridge Regression

Min MSE

$$J(\theta) = \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2 + \lambda \sum_{j=1}^{d} \theta_i^2$$

Gradient update: $\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{N} (h_\theta(x_i) - y_i)$

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_{ij} - \underbrace{\alpha\lambda\theta_j}_{\text{Regularization}}$$

$$\theta_j \leftarrow \theta_j(1 - \alpha\lambda) - \alpha \sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_{ij}$$

Lasso Regression

$$J(\theta) = \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{Squared Residuals}} + \underbrace{\lambda \sum_{j=1}^{d} |\theta_j|}_{\text{Regularization}}$$

- L1 norm for regularization
- Results in sparse coefficients
- Issue: gradients cannot be computed around 0
- Method of sub-gradient optimization

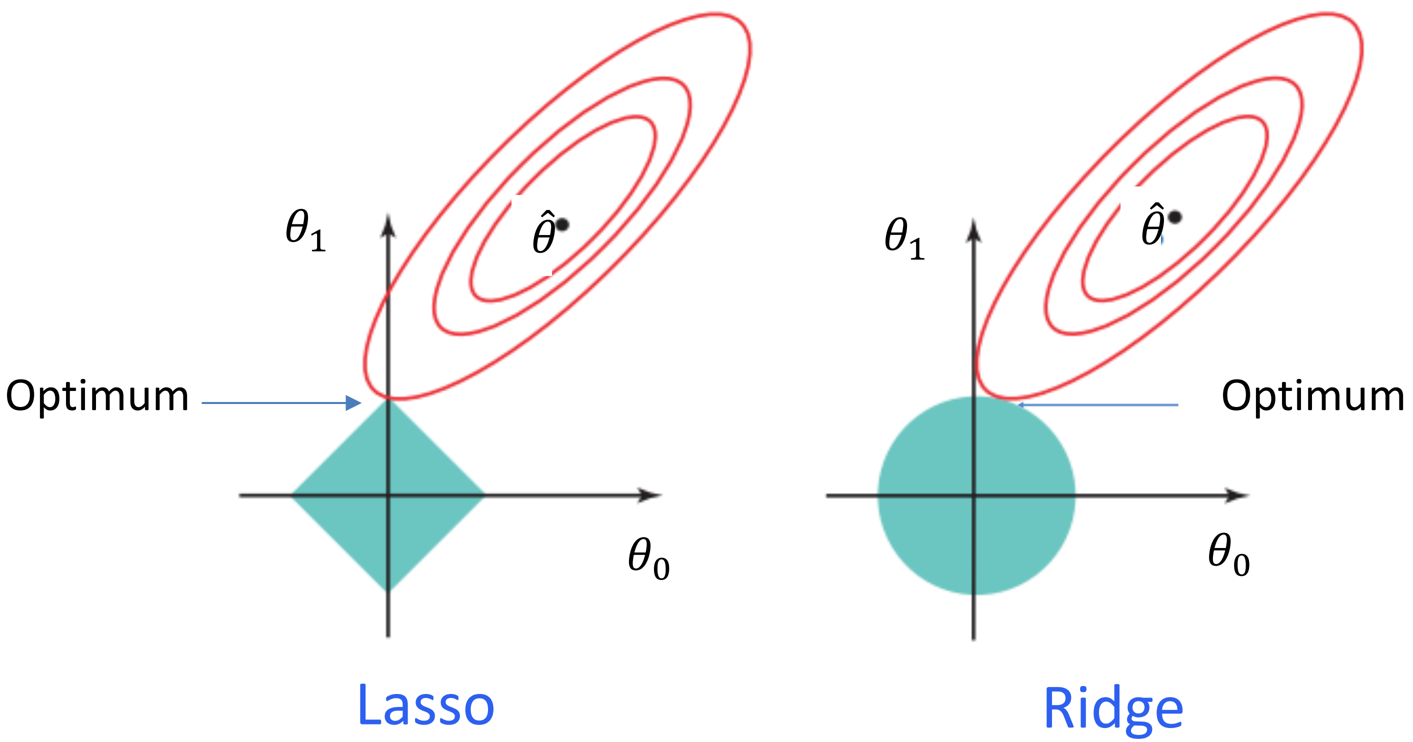
Alternative Formulations

- Ridge
  - L2 Regularization
  - $\min_{\theta} \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2$ subject to $\sum_{j=1}^{d} \left|\theta_j\right|^2 \leq \epsilon$

- Lasso
  - L1 regularization
  - $\min_{\theta} \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2$ subject to $\sum_{j=1}^{d} \left|\theta_j\right| \leq \epsilon$

Lasso vs Ridge

- Ridge shrinks all coefficients
- Lasso sets some coefficients at 0 (sparse solution)
  - Perform feature selection



Ridge vs Lasso

- Both methods can be applied to any loss function (regression or classification)
- In both methods, value of regularization parameter λ needs to be adjusted
- Both reduce model complexity

• Ridge	• Lasso
+ Differentiable objective + Gradient descent converges to global optimum - Shrinks all coefficients	- Gradient descent needs to be adapted + Results in sparse model + Can be used for feature selection in large dimensions