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

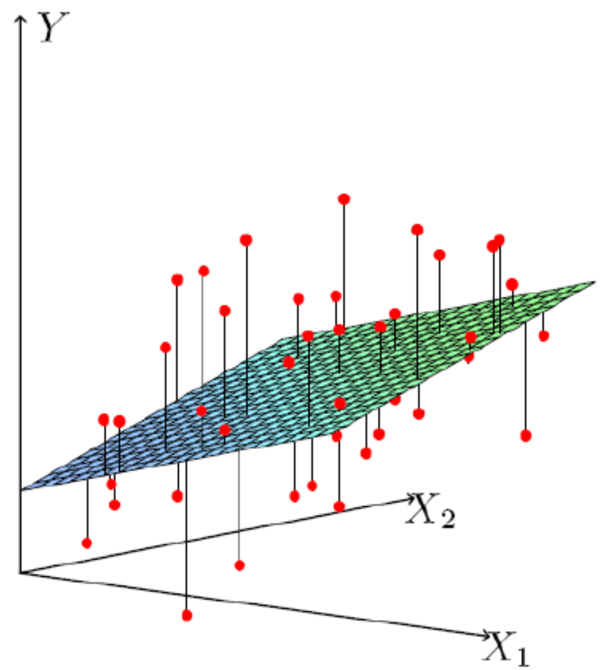
- Gradient Descent comparison with closed-form solution for linear regression
- Regularization
  - Ridge regression and gradient descent update
  - Lasso regression
- Classification
  - K Nearest Neighbors (kNN)
  - Bias-Variance tradeoff

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

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

MSE is a strictly convex function and has unique minimum



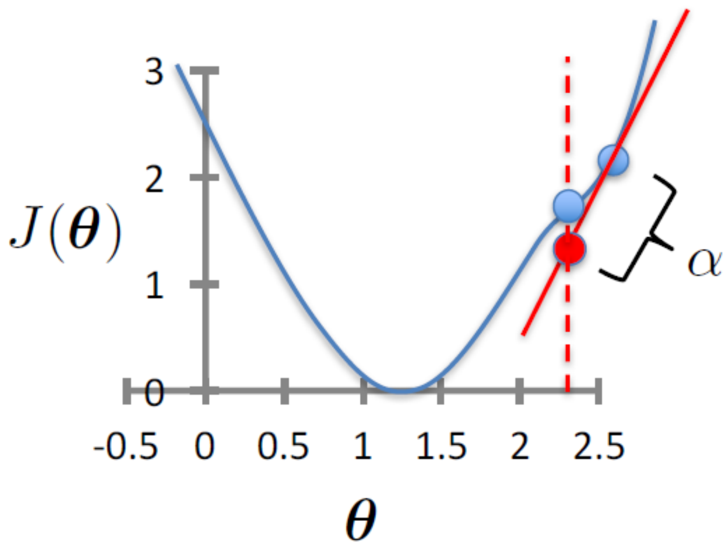
Gradient Descent

- 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 θ
- 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 $O(T \cdot N \cdot d)$
  - *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

$A \cdot B$

$(m, n) \cdot (n, p)$

COMPLEXITY OF MATRIX MULTIPLICATION

$O(m \cdot n \cdot p)$

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

$$\boldsymbol{\theta} = (\underbrace{\boldsymbol{X}^\intercal \boldsymbol{X}}_{O(d^2 N)})^{-1} \underbrace{\boldsymbol{X}^\intercal \boldsymbol{y}}_{(d,N)\cdot(N,1)}$$

$O(d^2 \log d)$

- Gradient Descent

+ FASTER $O(TNd)$

- TUNING FOR α, T, ...

+ GENERAL OPT. METHOD

- LOCAL MIN; NOT CONVERGING

- Closed Form

- SLOW $O(d^2N)+\ldots$

- NOT ALWAYS INVERTIBLE

+ GLOBAL MIN.

Issues with Gradient Descent

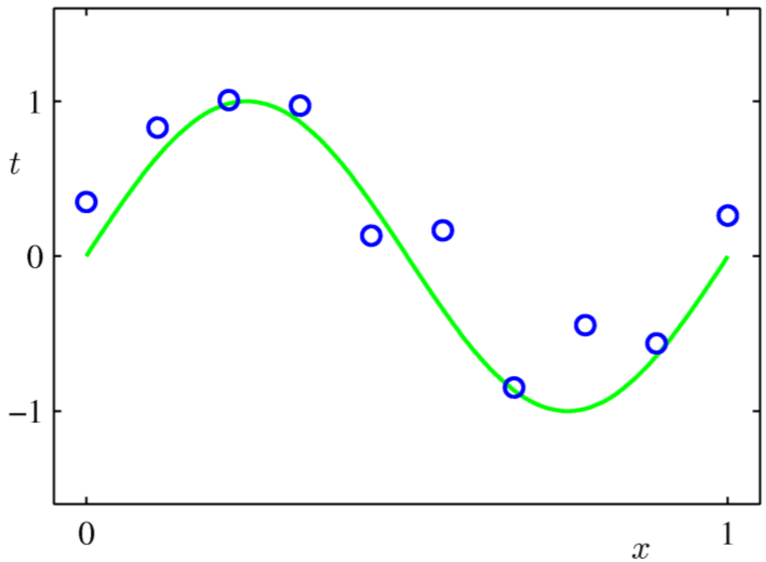
- Might get stuck in local optimum and not converge to global optimum
  - Restart from multiple initial points
- Only works with differentiable loss functions
- Small or large gradients
  - Feature scaling helps
- Tune learning rate
  - Can use line search for determining optimal learning rate

Review Gradient Descent

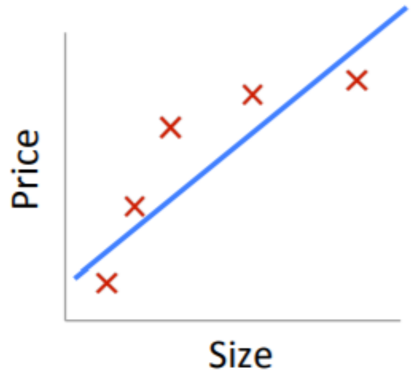
- Gradient descent is an efficient algorithm for optimization and training ML models
  - The most widely used algorithm in ML!
  - Much faster than using closed-form solution for linear regression
  - Main issues with Gradient Descent is convergence and getting stuck in local optima (for neural networks)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

Polynomial Regression

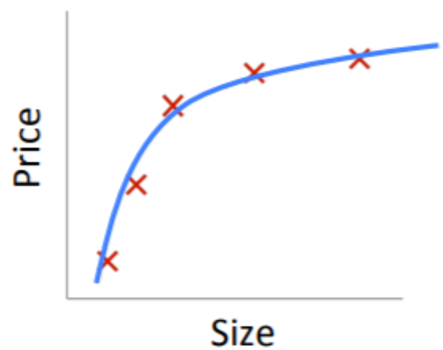
- Polynomial function on single feature $x \in \mathbb{R}, y \in \mathbb{R}$
  - $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_p x^p$ degree p.



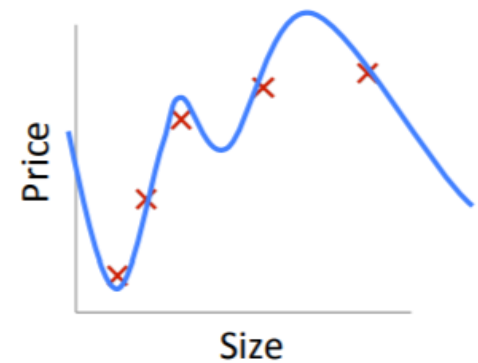
Polynomial Regression



$\theta_0 + \theta_1 x$



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



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

deg 1
LINER REG.
UNDERFIT
HIGH BIAS

deg 2

"BEST" FIT

deg=4
OVERFIT
HIGH VAR

$p \leq 4.$

Polynomial Regression Training

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

$x_1, \ldots, x_N \in \mathbb{R}$

$y_i \in \mathbb{R}$

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & & x_2^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & & x_N^p \end{bmatrix}$$

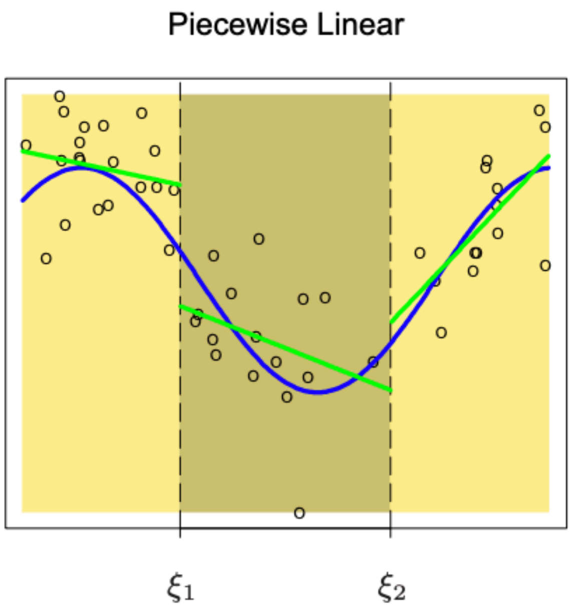
$$y = \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_N \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}$$

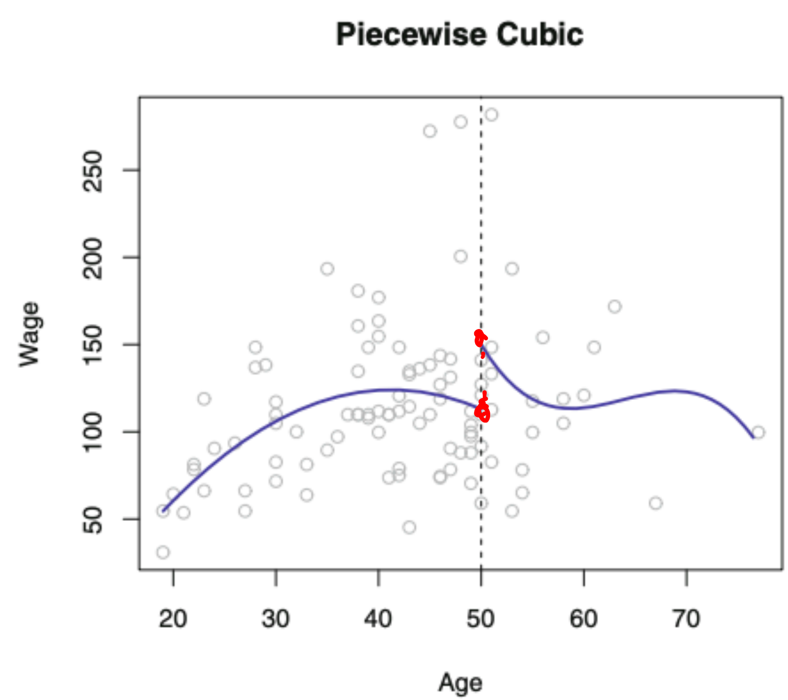
$$\theta = (X^T X)^{-1} X^T y$$

Piecewise Polynomial

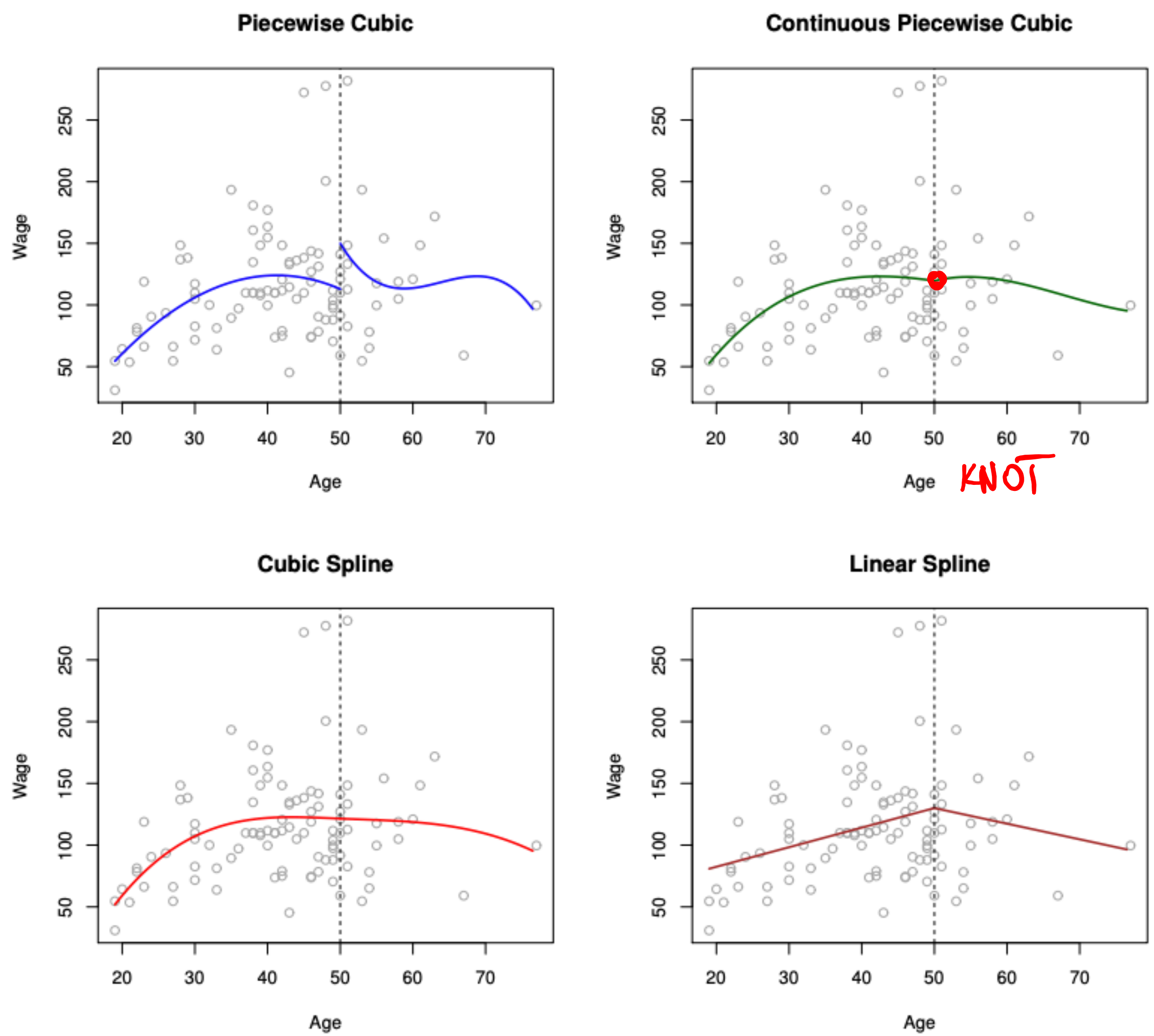
- Divide the space into regions
- Polynomial regression on each region
  - Linear piecewise (degree 1), quadratic piecewise (degree 2), cubic piecewise (degree 3)

KNOT





Piecewise and spline regression



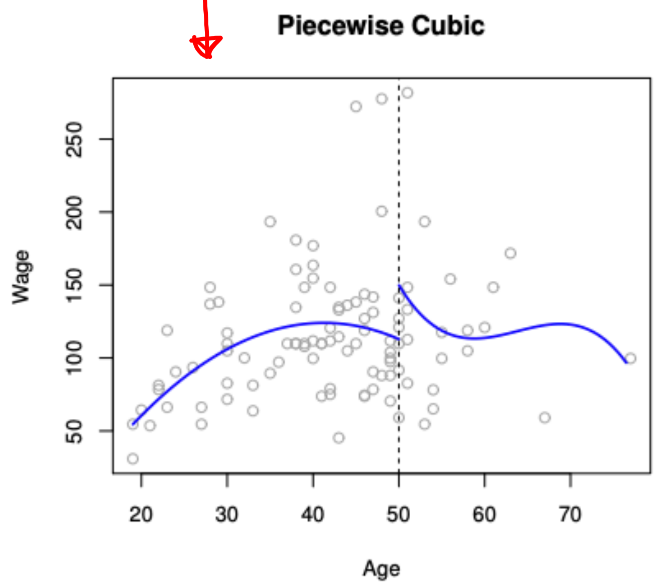
K KNOTS

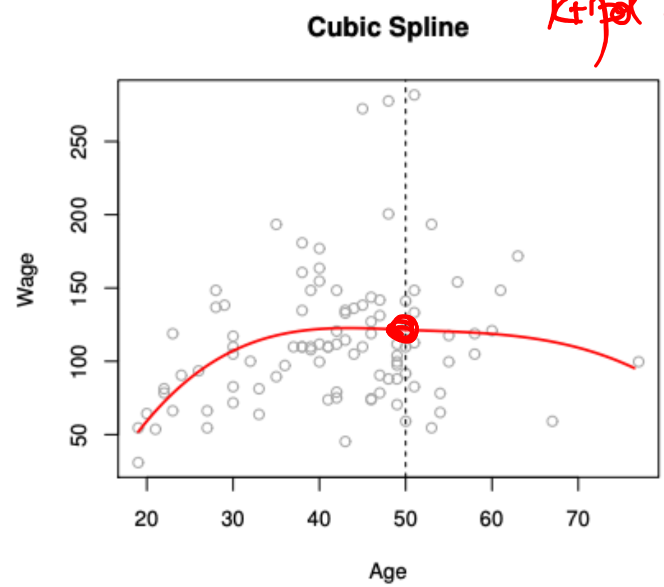
K+1 INTERVALS

Piecewise polynomial vs Regression spline

INDEPENT

1 pol. deg 3 → 4
2 pol deg 3 → 8
⋮
K+1 pol deg 3 → (K+1)4





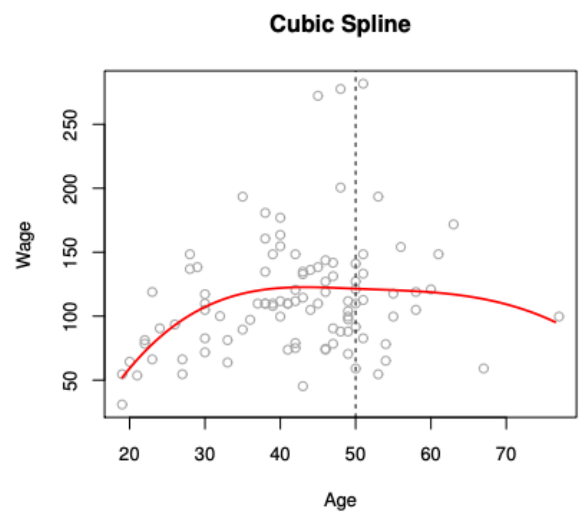
1 break at Age $= 50$

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

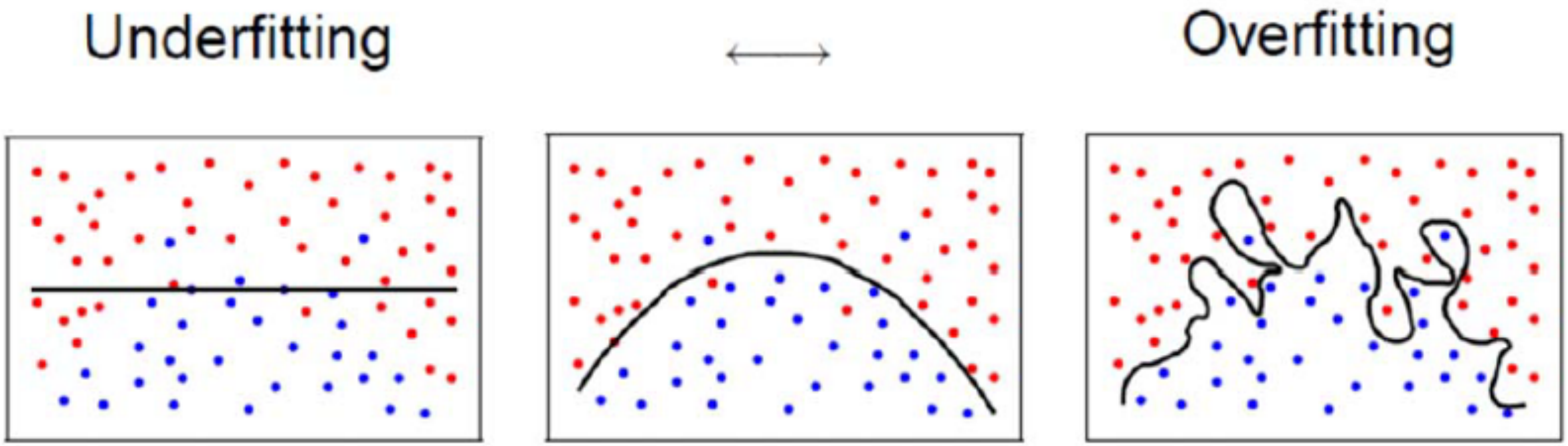
Additive Models

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

CAN OPTIMIZE WITH LEAST SQUARE METHOD

EXPAND SET OF FEATURES

Generalization in ML



Simple model

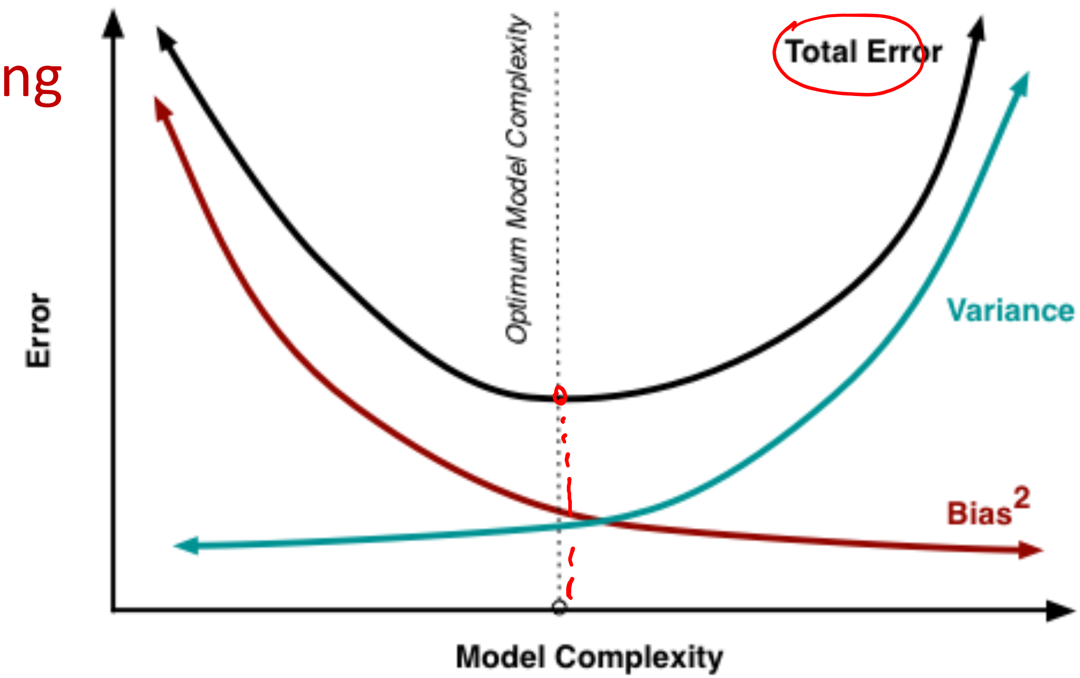
Complex model

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

Bias-Variance Tradeoff

MSE in Testing

Under-fitting



Over-fitting

MSE in TRAINING $\to 0$

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

MSE is proportional to Bias^2 + 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

L2 REGULARIZATION

- Linear regression objective function

$$\min_{\theta} J(\theta) = \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{N MSE}} + \lambda \underbrace{\sum_{j=1}^{d} \theta_j^2}_{\text{REGULARIZATION TERM}}$$

$\|\theta\|_2^2$

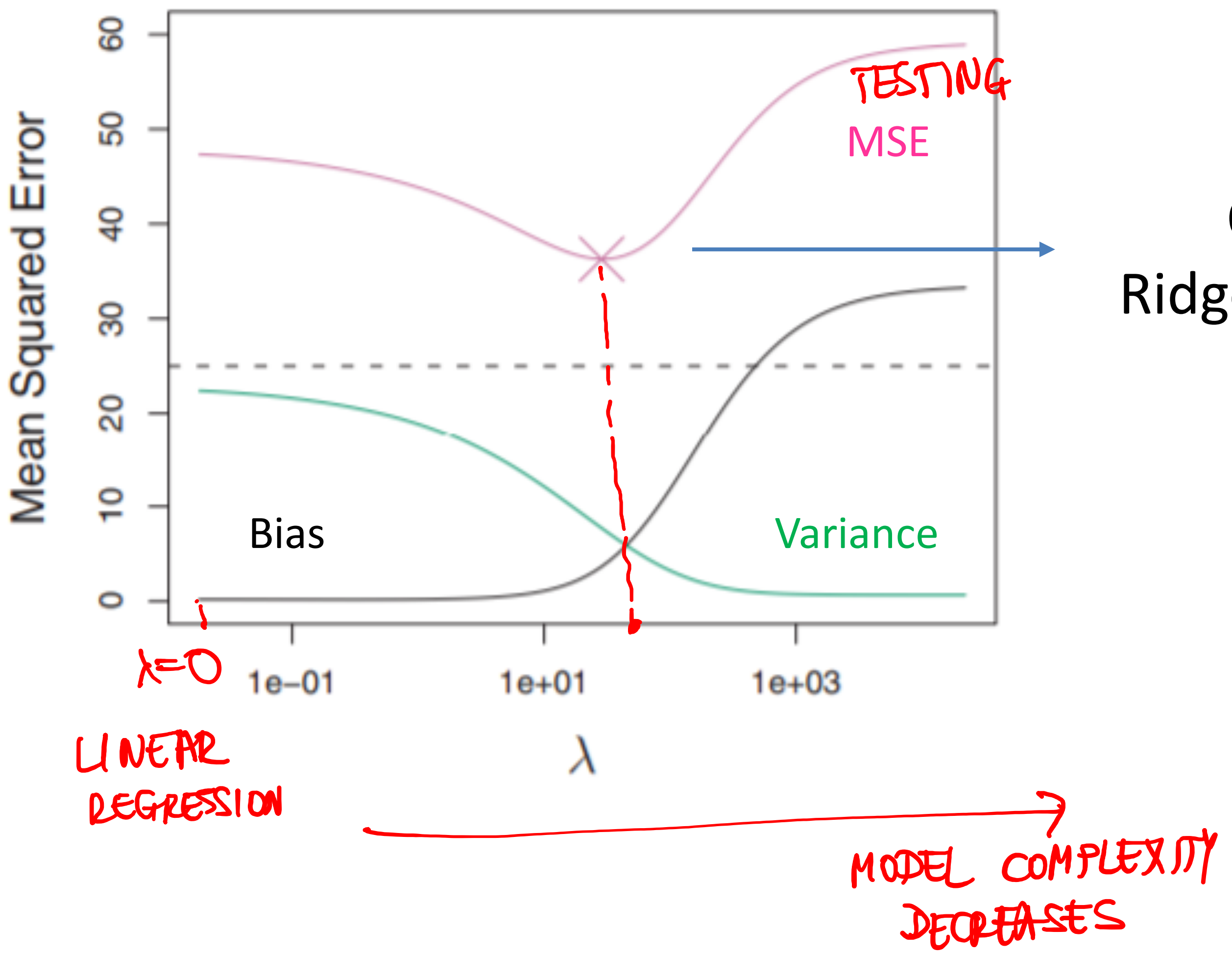
λ = REG PARAM

$\lambda = 0 \Rightarrow$ MLR

AS λ INCREASES $\Rightarrow \|\theta\|^2$ DECREASES

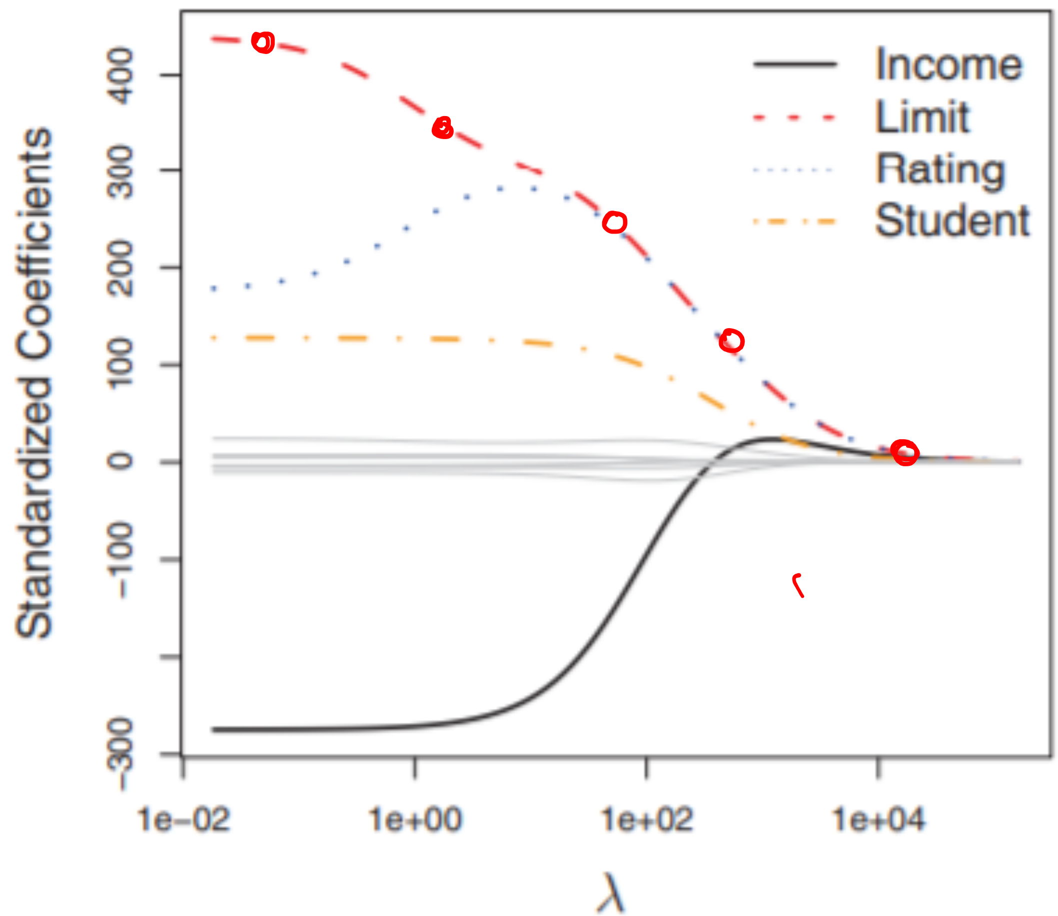
DECREASE MODEL COMPLEXITY

Bias-Variance Tradeoff



Optimal
Ridge regression

Coefficient shrinkage



Predict credit card balance

GD for Ridge Regression

Min MSE

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

$\theta_1^2 + \dots + \theta_j^2 + \dots + \theta_d^2$

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

$J(\theta)$ IS STRICTLY CONVEX IN θ

$$\frac{\partial J(\theta)}{\partial \theta_j} = 2 \sum_{i=1}^{N} [h_\theta(x_i) - y_i] \cdot x_{ij} + 2\lambda\theta_j$$

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

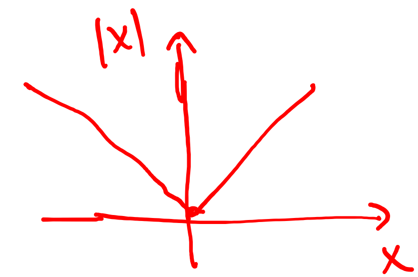
$$= \theta_j (1 - \underbrace{2\alpha\lambda}_{}) - 2\alpha \sum_{i=1}^{N} [h_\theta(x_i) - y_i] x_{ij}$$

Lasso Regression

$$J(\theta) = \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\sim \text{MSE}} + \lambda \underbrace{\sum_{j=1}^{d} |\theta_j|}_{\text{L1 NORM OF } \theta}$$

$\lambda = 0 \Rightarrow$ MLR

AS λ INCREASES $\Rightarrow$ DECREASE θ



- L1 norm for regularization
- Results in sparse coefficients
- Small issue: gradients cannot be computed around 0
  - Can use sub-gradient at 0

Alternative Formulations

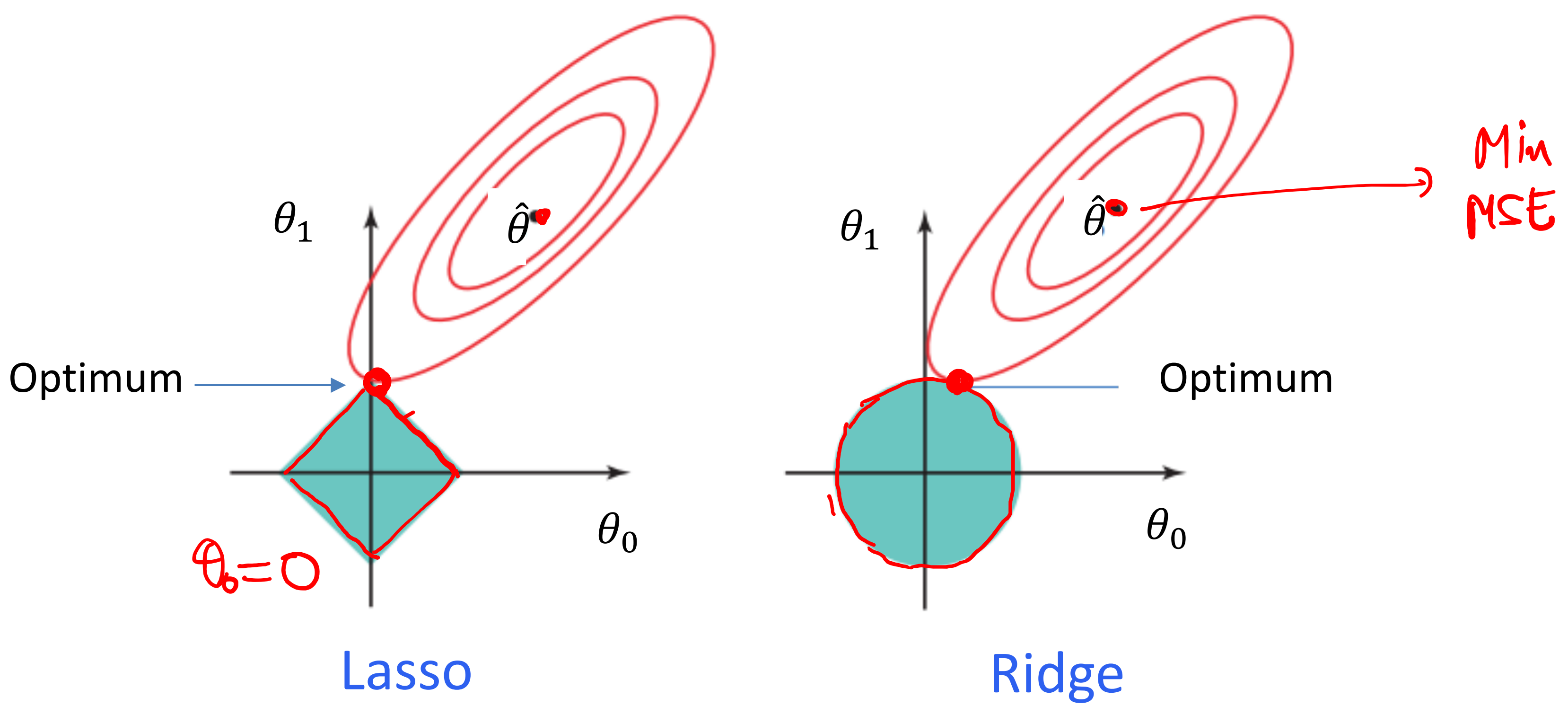
$\theta_0^2 + \theta_1^2 \leq \varepsilon$

- Ridge
  - L2 Regularization
  - $\min_{\theta} \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{MSE}}$ subject to $\underbrace{\sum_{j=1}^{d} |\theta_j|^2}$ $\leq \epsilon$
- Lasso
  - L1 regularization
  - $\min_{\theta} \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{MSE}}$ subject to $\sum_{j=1}^{d} |\theta_j| \leq \epsilon$

$|\theta_0| + |\theta_1| \leq \varepsilon$

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)

• Ridge + SMALL # FEATURES RELEVANT FEATURES + CONVEX, GRADIENT DESCENT + CLOSED FORM; CAN ALWAYS COMPUTE IT	• Lasso + FEATURE SELECTION LARGE DIM DATA − ADAPT GRADIENT DESCENT