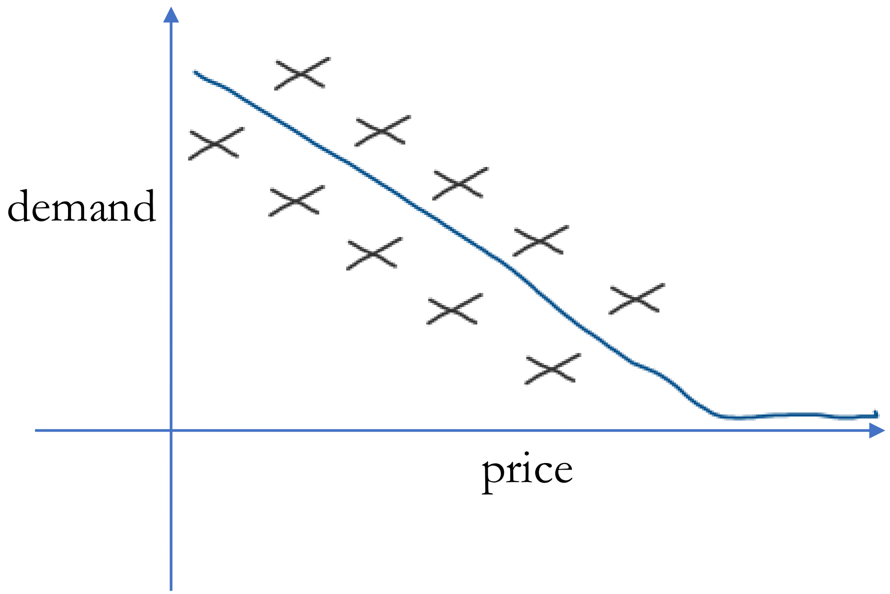
Introduction to Artificial Intelligence

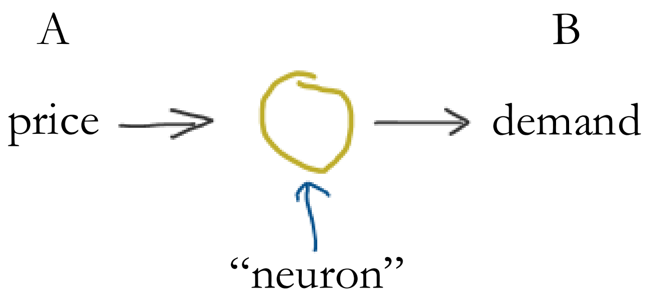
Lecture 7: Backpropagation and generalization

September 25, 2025

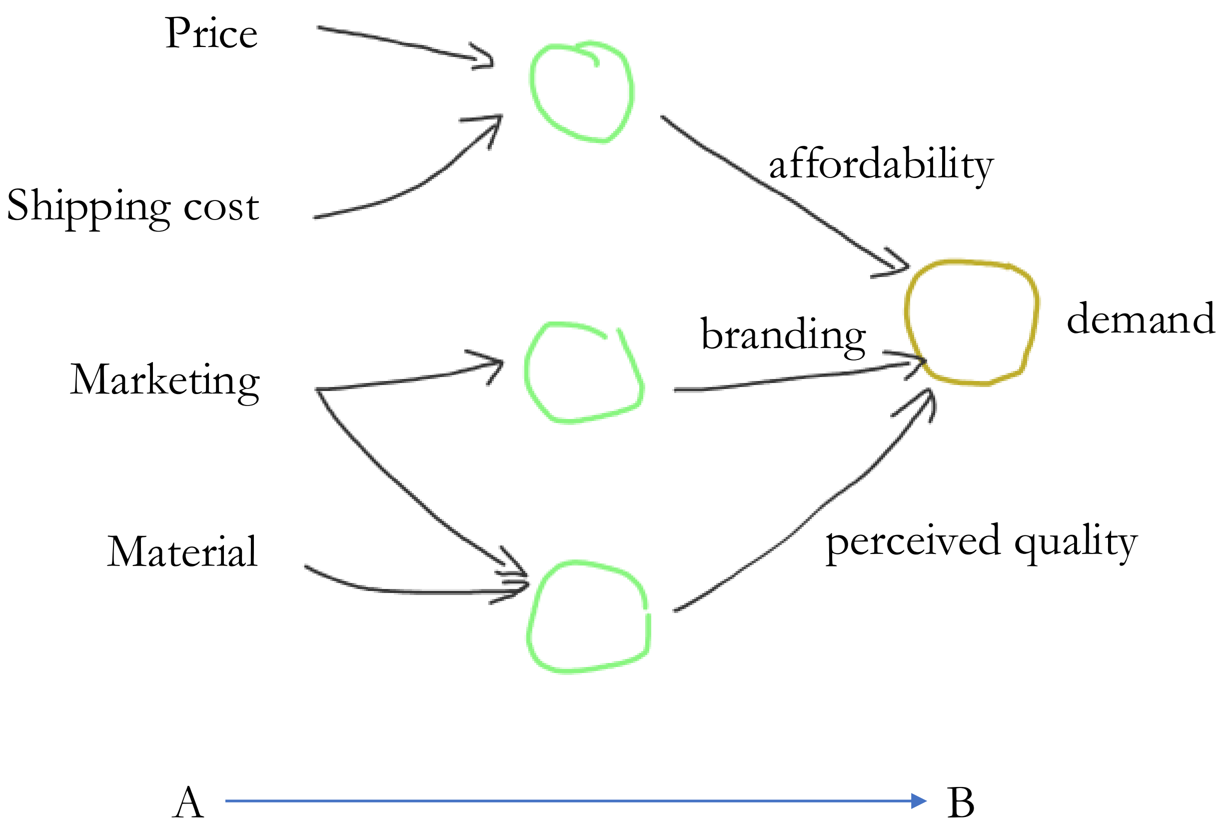
What is a neural network?

- Demand prediction

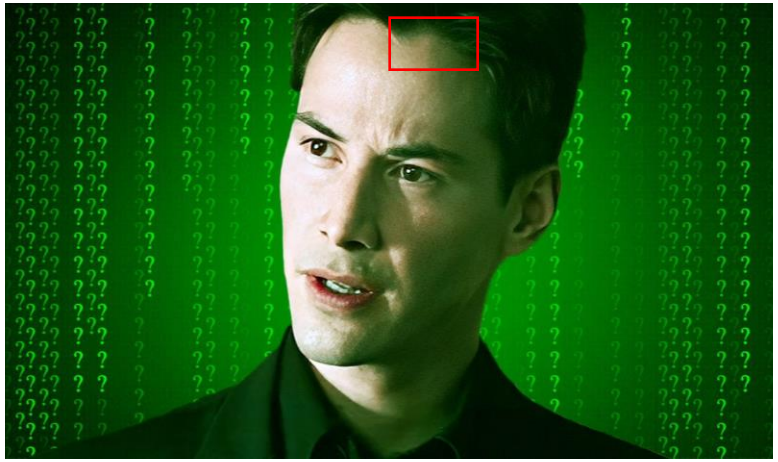




Demand prediction



Face recognition



10	20	30	15	17	7	8	31	8
12	13	345	100	5	0	34	30	1001
230	301	9	0	45	123	0	20	201
31	130	2	9	0	40	2	15	46
1	15	21	42	55	7	1	12	10

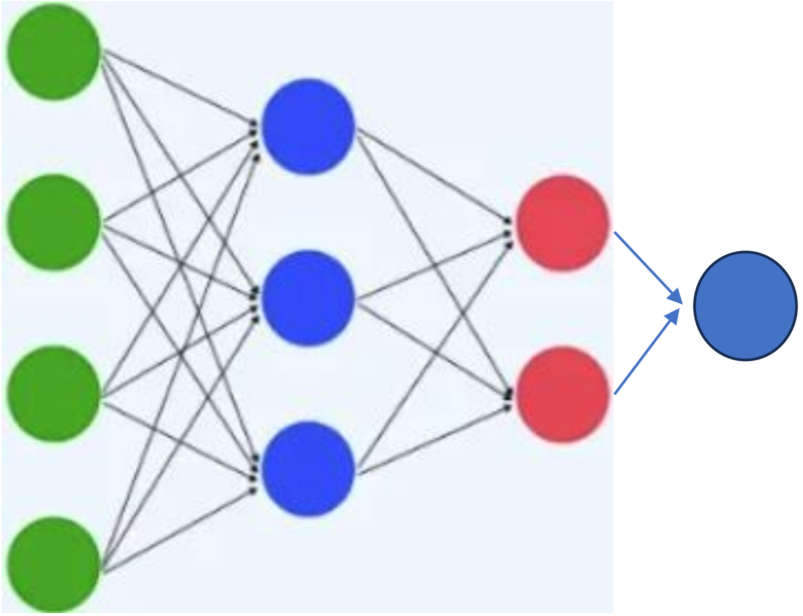
Face recognition

1000

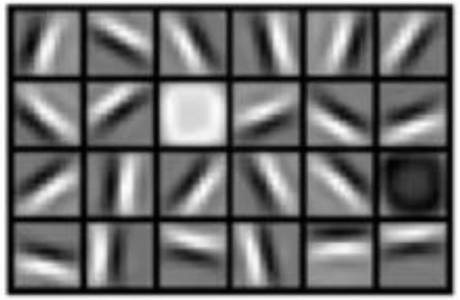
1000

1,000,000 pixel array

3,000,000 colored pixels



identity



AI era

- Any company + deep learning ≠ AI company
- Do things that AI is really good at
  - Strategic data acquisition
  - Unified data warehouse
  - Pervasive automation
  - New roles (e.g., Machine Learning Engineer) and division of labor

AI transformation

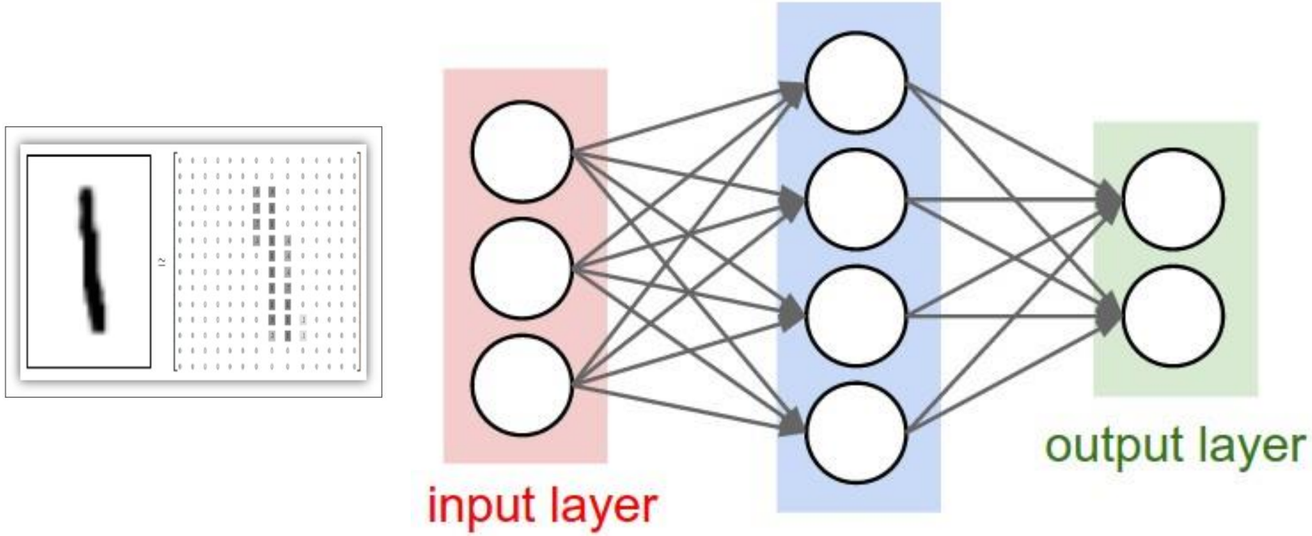
- Execute pilot projects to gain momentum
- Build an in-house AI team
- Provide broad AI training
- Develop an AI strategy
- Develop internal and external communications

Lecture plan

- **Backpropagation**
- Regularization
  - Weight decay
  - Transfer learning

Training loss objective

- Backpropagation algorithm is the workhorse of modern deep networks
- Train parameters W_1, b_1, W_2, b_2 to minimize the cross-entropy loss
- Minimize the cross-entropy loss as the training objective



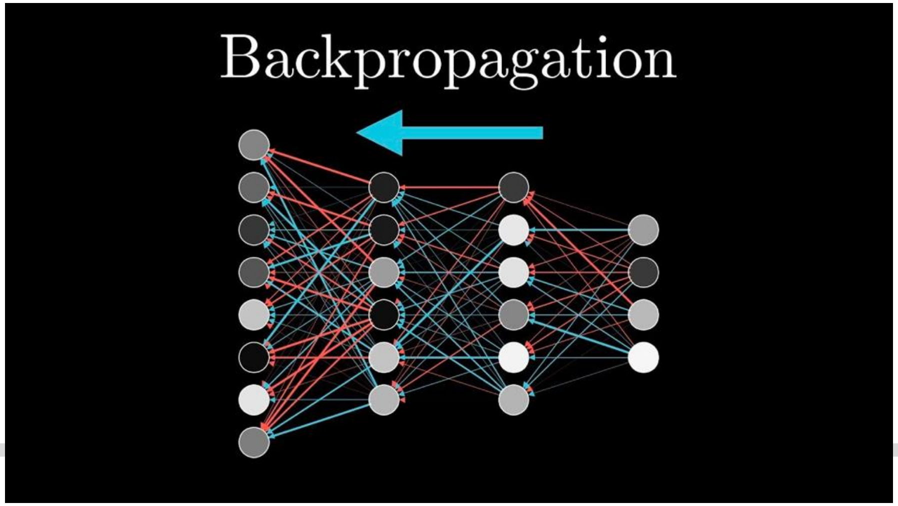
Prediction over {0,**1**,2,3,4,5,6,7,8,9}

The gradient of each layer

- Notations:
  - Suppose x is a data point with label y: Let $\ell(f(x), y)$ be the loss
- Input: x, y
- Output (of backpropagation):
  - Partial derivative of ℓ with respect to W_1, b_1 (layer 1): $\frac{\partial \ell}{\partial W_1}, \frac{\partial \ell}{\partial b_1}$
  - Partial derivative of ℓ with respect to W_2, b_2 (layer 2): $\frac{\partial \ell}{\partial W_2}, \frac{\partial \ell}{\partial b_2}$

How backpropagation works

- Backpropagation consists of two steps
  - **Step 1: Use forward pass** to compute the input to every layer and the output of every layer
  - **Step 2: Use backward pass** to compute the gradient
  - In total, we need to run two passes over the entire neural network to conduct this computation!



Forward pass

- Input: $o_0 = x$
- For $i = 1, 2, \ldots, L$
  - Input to layer i: $z_i = o_{i-1}W_i + b_i$
  - Output of layer i: $o_i = \sigma_i(z_i)$
- Return o_L

- Important takeaway
  - Input to layer i: z_i
  - Output of layer i: o_i

The backward pass

- **Notations**
  - Loss function ℓ
  - i-th trainable layer: weight matrix $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$, bias $b_i \in \mathbb{R}^{d_i}$
  - Activation function: $\sigma_i: \mathbb{R} \to \mathbb{R}$
- **Output**
  - $\frac{\partial \ell}{\partial W_i}$ and $\frac{\partial \ell}{\partial b_i}$ for all $i = 1,2, \dots, L$

Example

- A two-layer linear network with mean squared loss

$$\ell(x, y) = (w_2 w_1 x - y)^2$$

- Output of backpropagation

$$\frac{\partial \ell}{\partial w_2} = 2(w_2 w_1 x - y) w_1 x$$

$$\frac{\partial \ell}{\partial w_1} = 2(w_2 w_1 x - y) w_2 x$$

Example with nonlinear activation

- Nonlinear activation

$$\ell(x, y) = (w_2 \sigma_1(w_1 x) - y)^2$$

- Claims

$$\frac{\partial \ell}{\partial w_2} = 2(w_2 \sigma_1(w_1 x) - y)\sigma_1(w_1 x)$$

$$\frac{\partial \ell}{\partial w_1} = 2(w_2 \sigma(w_1 x) - y) w_2 \sigma_1'(w_1 x) x$$

- Compare with the previous example, we have an additional term which is $\sigma_1{}'(w_1 x)$

Multi-layer network

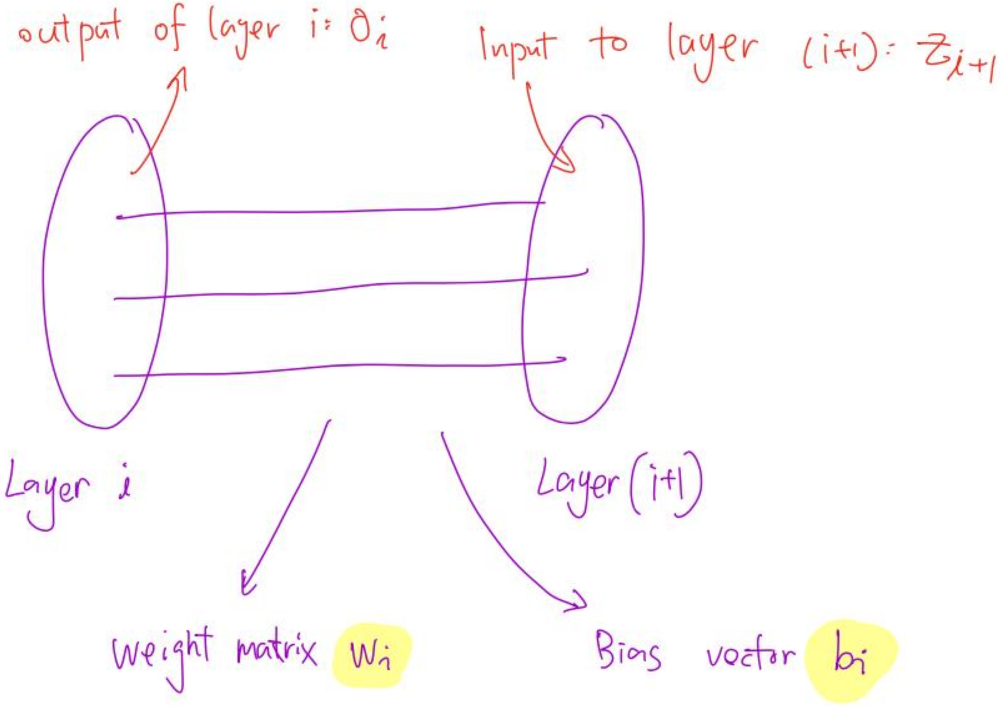
- A multi-layer linear network with squared loss

$$\ell(x, y) = (w_L w_{L-1} \dots w_1 x - y)^2$$

- Back propgating from last layer to first layer
  - $\frac{\partial \ell}{\partial w_L} = 2(w_L w_{L-1} \dots w_1 x - y) w_{L-1} \dots w_1 x$
  - $\frac{\partial \ell}{\partial w_{L-1}} = 2(w_L w_{L-1} \dots w_1 x - y) w_L w_{L-2} \dots w_1 x$
  - …
  - $\frac{\partial \ell}{\partial w_1} = 2(w_L w_{L-1} \dots w_1 x - y) w_L w_{L-1} \dots w_2 x$

Looking at an intermediate layer

- Illustration



Loss function : ℓ

Want : $\frac{\partial \ell}{\partial W_i}$ and $\frac{\partial \ell}{\partial b_i}$

Applying chain rule to tackle nonlinear activation

$$z_{i+1} = w_i o_i = w_i \sigma(z_i)$$

- In this case, we instead have $\frac{\partial z_{i+1}}{\partial z_i} = w_i \sigma'(z_i)$

- Caveat: in this example, we focused on one-dimensional input. For multi-dimensional input, the idea is the same, although the computation is hairier

Summary: The backward pass

- Write $\frac{\partial \ell}{\partial w_i}$ and $\frac{\partial \ell}{\partial b_i}$ based on $\frac{\partial \ell}{\partial w_{i+1}}$ and $\frac{\partial \ell}{\partial b_{i+1}}$
  - Decompose the gradient at this layer back to the gradient of the previous layer

- Find the gradient at every layer by going backward from the final output layer
  - Find out $\frac{\partial \ell}{\partial w_L}$ and $\frac{\partial \ell}{\partial b_L}$
  - Find out $\frac{\partial \ell}{\partial w_{L-1}}$ and $\frac{\partial \ell}{\partial b_{L-1}}$
  - …
  - Find out $\frac{\partial \ell}{\partial w_1}$ and $\frac{\partial \ell}{\partial b_1}$

Lecture plan

- Backpropagation
- **Regularization**
  - Weight decay
  - Transfer learning

Regularization

- **Weight decay:** adding a penalty of $\lambda \sum_i w_i^2$ to the loss function used to train the model
  - Weight decay is equivalent to ℓ_2-regularization or ridge regression (next slide)

- **Dropout:** at each step of training, dropout works by randomly dropping a chosen subset of neurons, and applying backpropagation to a new version of the neural network

Illustrating weight decay in linear models

Linear model: $Y = \beta_0 + X_1\beta_1 + X_2\beta_2 + \cdots + X_p\beta_p + \varepsilon$

- Suppose the number of predictors $p > n$ (e.g., this happens a lot in bioinformatics, such as gene expressions): we have more parameters than observations

- How can we estimate β?

Example

- Predict Boston house prices: Suppose we only have one observation ($n = 1$)

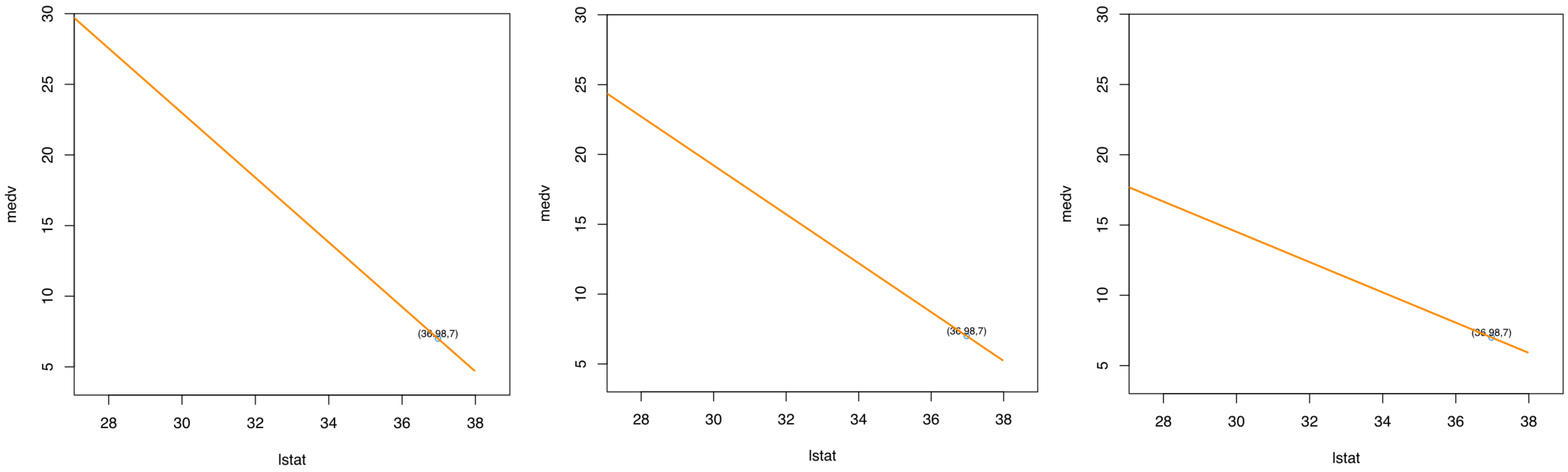
crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
45.7461	0	18.1	0	0.693	4.519	100	1.6582	24	666	20.2	36.98	7

- Suppose we want to estimate the coefficients in simple linear regression:

$$medv = \beta_0 + lstat \cdot \beta_1 + \varepsilon$$

- How can we use one observation to estimate β_0, β_1?

Which β_0 and β_1 should we choose?



All of these are valid solutions!

If we have one more observation…

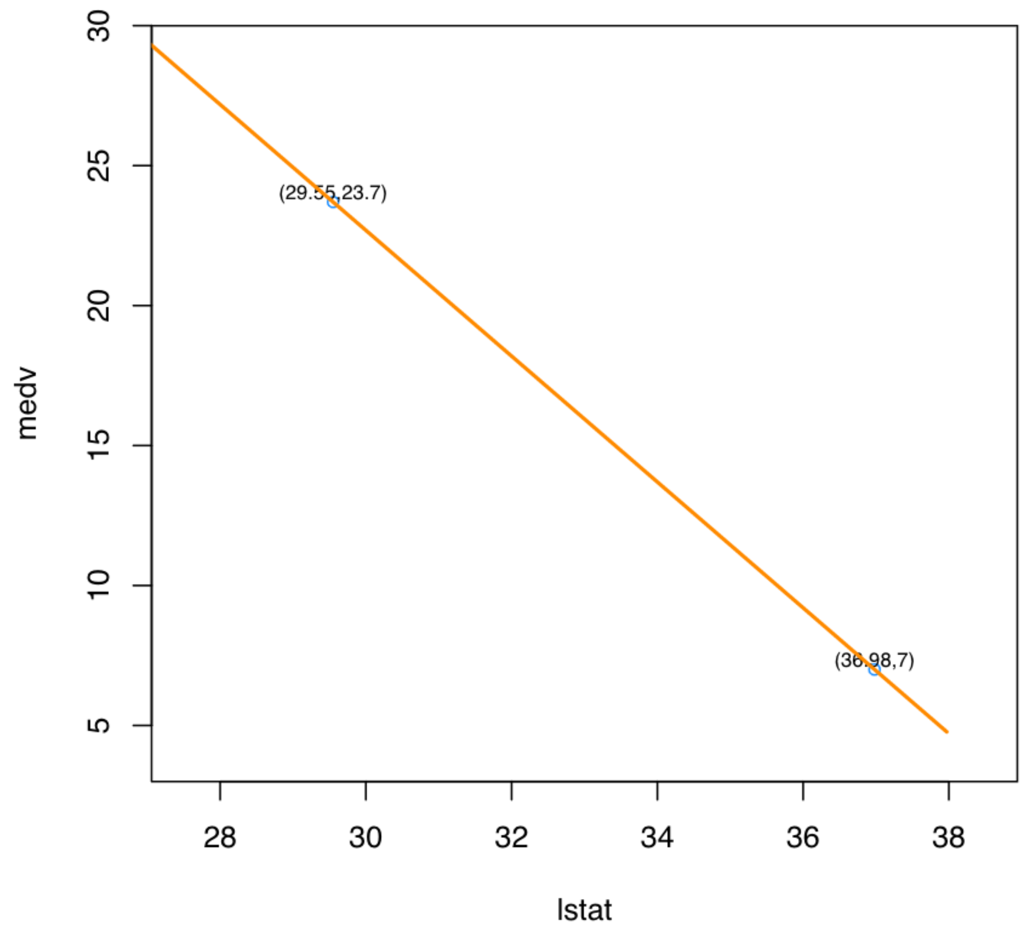
- Suppose we only have two observations ($n = 2$)

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
0.28955	0	10.59	0	0.489	5.412	9.8	3.5875	4	277	18.6	29.55	23.7
45.74610	0	18.10	0	0.693	4.519	100.0	1.6582	24	666	20.2	36.98	7.0

- Let us consider the same model: $medv = \beta_0 + lstat \cdot \beta_1 + \varepsilon$

- We can estimate β_0 and β_1 with two data points (solving a linear system)

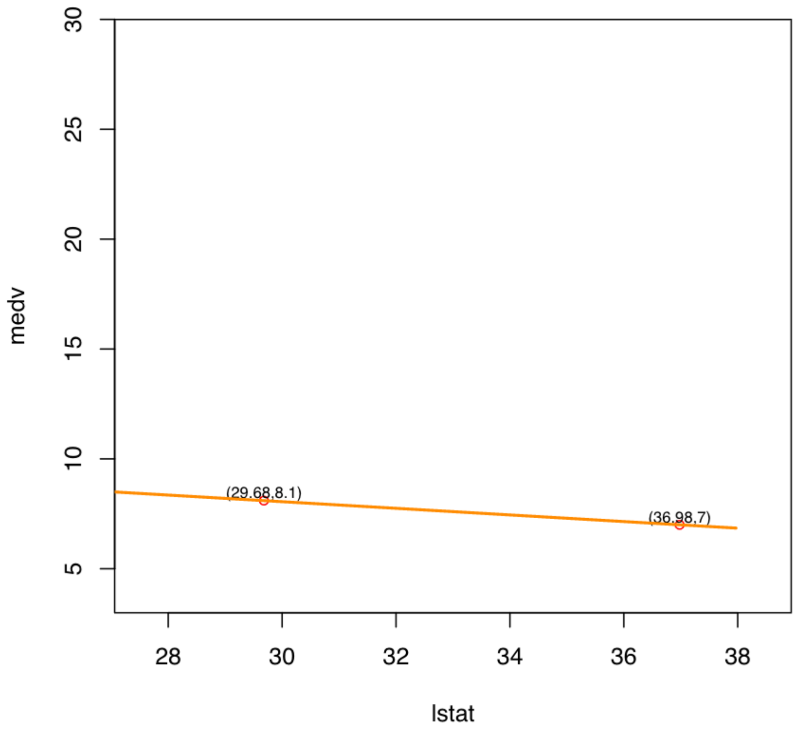
Example



- Problem: The fitted curve is sensitive to the $medv$ of these two observations

Example

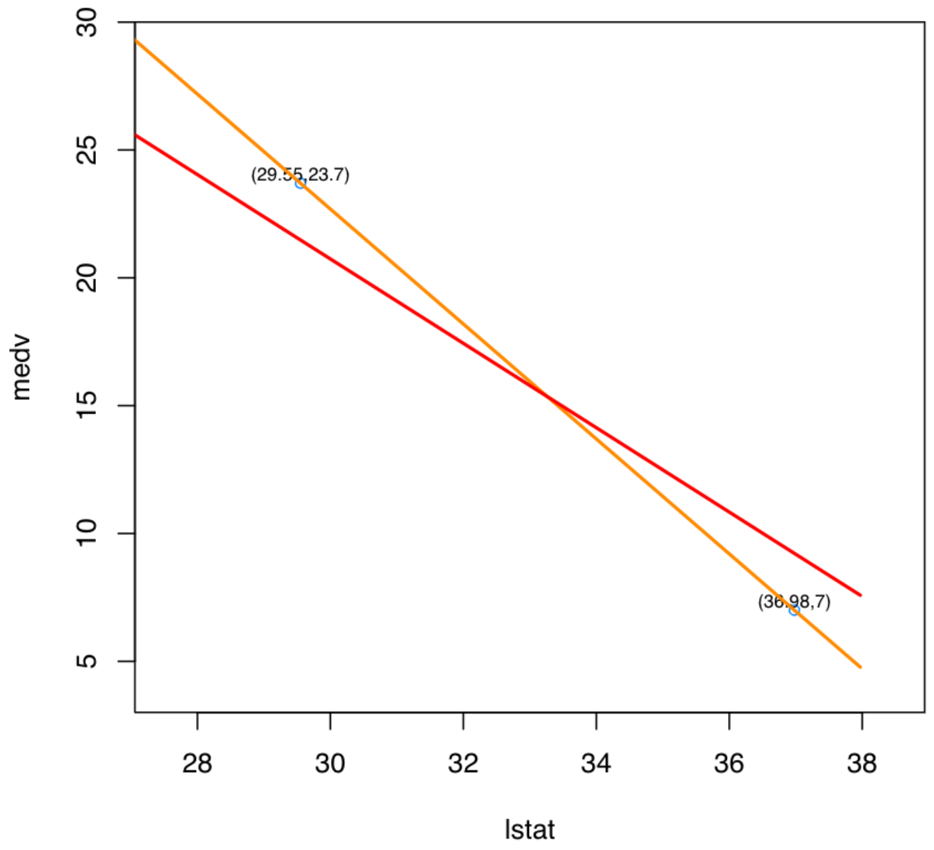
- If one of the two observations changes, we can get a very different fitted curve



- This is an example of overfitting…
- Question: can you think of other examples of overfitting?

Ridge regression

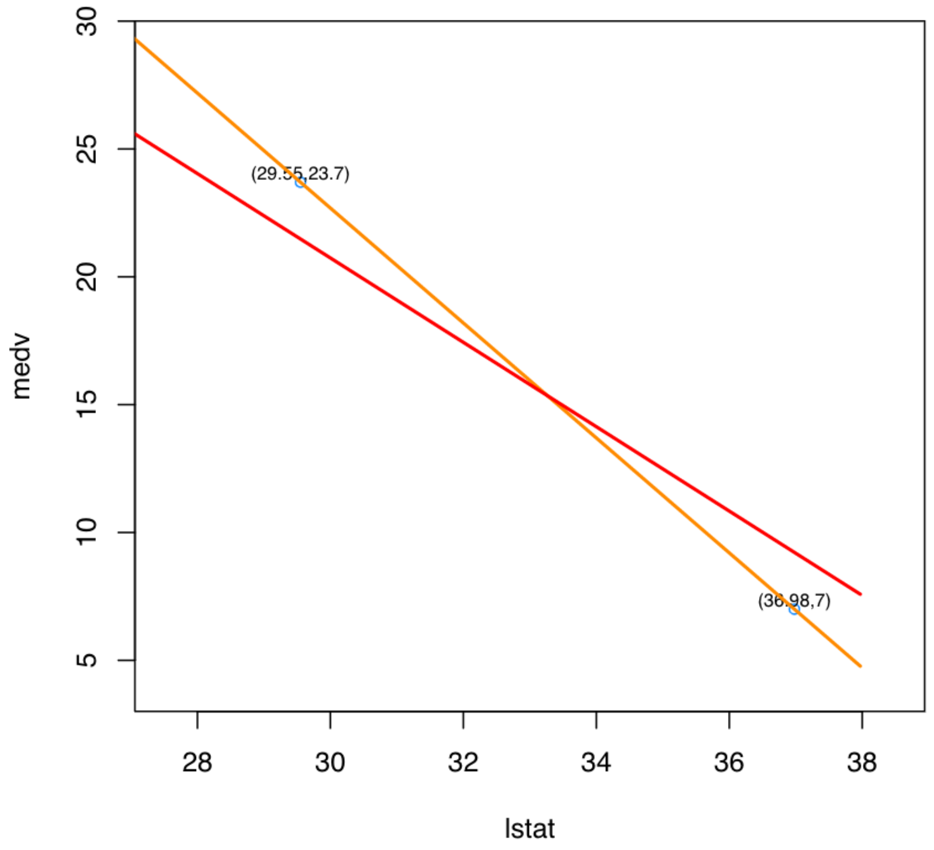
- Find **a new line** that **does not fit** the **training data perfectly**



- Introduce **a small amount of bias** into the fit to data

Ridge regression

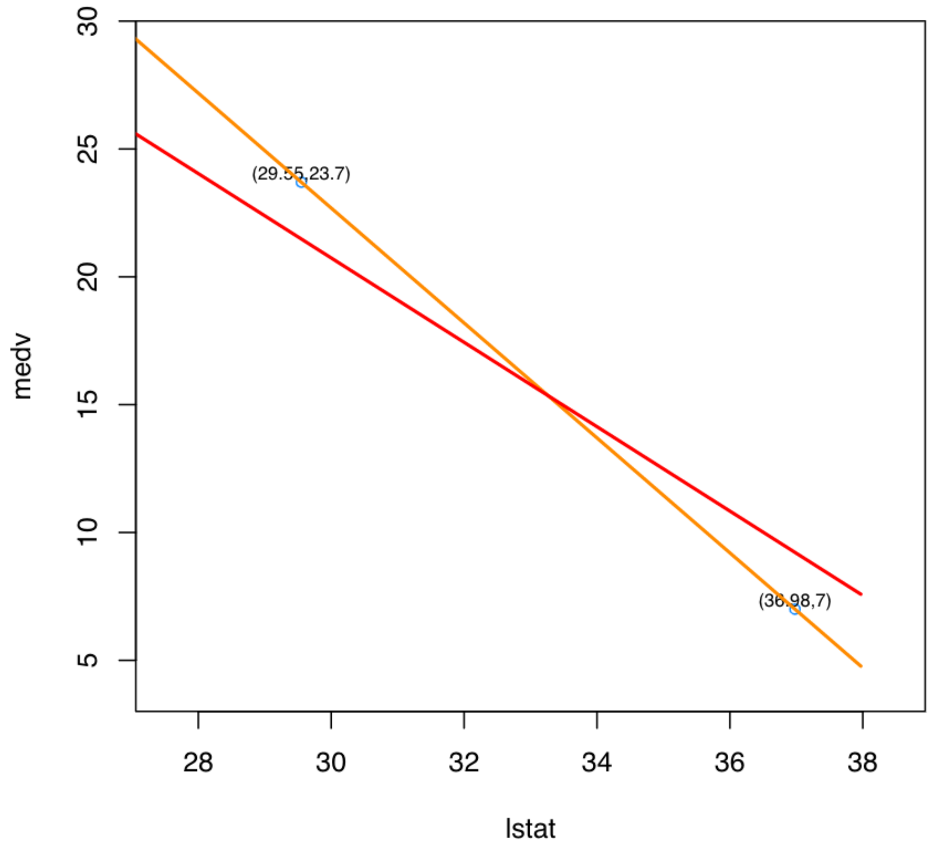
- This can be achieved with ridge regression: by adding **a small amount of bias,** we reduce variance (i.e., the fitted lines are less sensitive to changes with the input)



Fitting ridge regression

- Linear regression minimizes

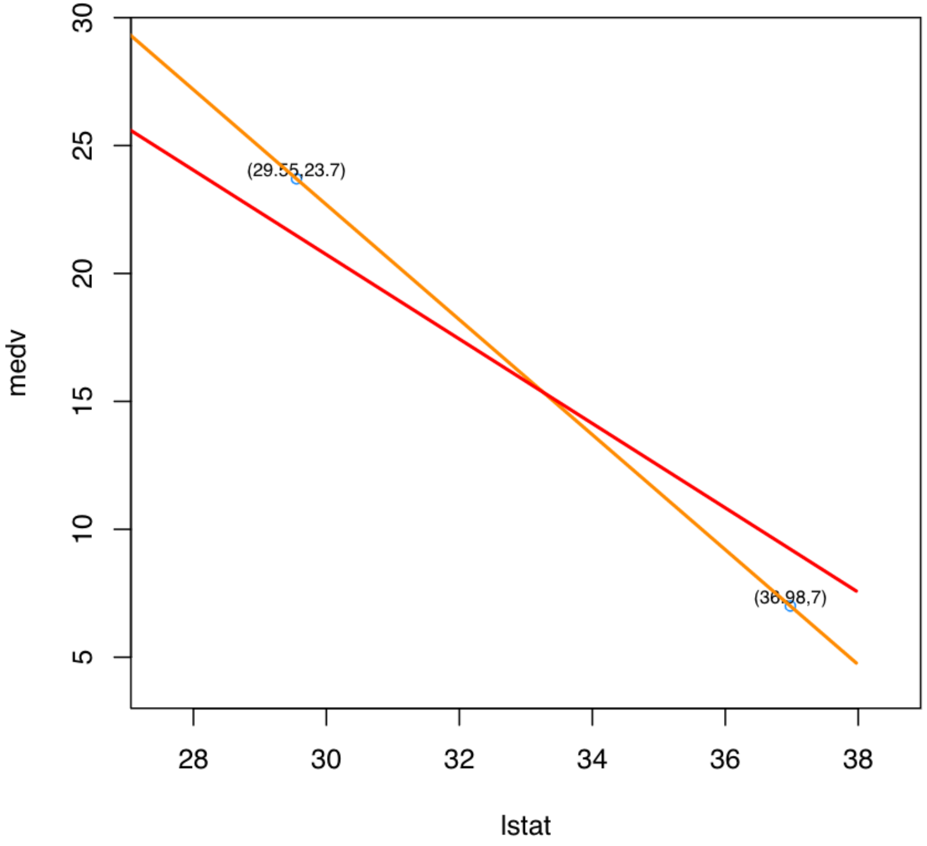
$$MSE = \sum_{i=1}^{n} (medv_i - \beta_0 - lstat \cdot \beta_1)^2$$



- Ridge regression minimizes
  - $\sum_{i=1}^{n} (medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$
  - $\lambda \geq 0$: tuning hyper-parameter

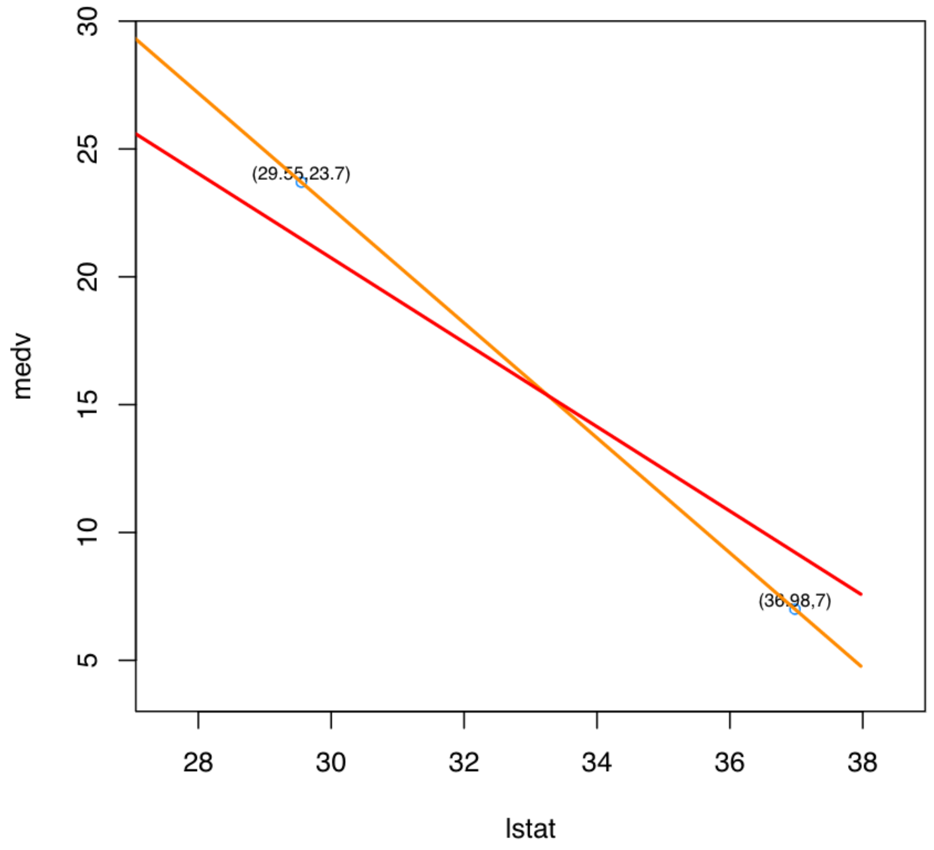
Example

- Suppose $\lambda = 10$
- Linear regression fit: $\widehat{medv} = 90.118 - 2.248 \cdot lstat$
  - $\hat{\beta}_1 = -2.248$
  - $\sum_{i=1}^{n}\left(medv_i - \hat{\beta}_0 - lstat_i \cdot \hat{\beta}_1\right)^2 + \lambda \cdot \hat{\beta}_1^2$
  $= 0 + 10 \cdot 2.248^2 = 50.535$
  - Perfectly fitting the data incurs high loss



Ridge regression

- Suppose $\lambda = 10$
- Ridge regression fit: $\widehat{medv} = 70.234 - 1.650 \cdot lstat$
  - $\hat{\beta}_1^R = -1.650$
  - $\sum_{i=1}^{n}\left(medv_i - \hat{\beta}_0 - lstat_i \cdot \hat{\beta}_1^R\right)^2 + \lambda \cdot \left(\hat{\beta}_1^R\right)^2$
    $= 4.931 + 4.931 + 10 \cdot 1.650^2 = 37.084$
    < 50.535



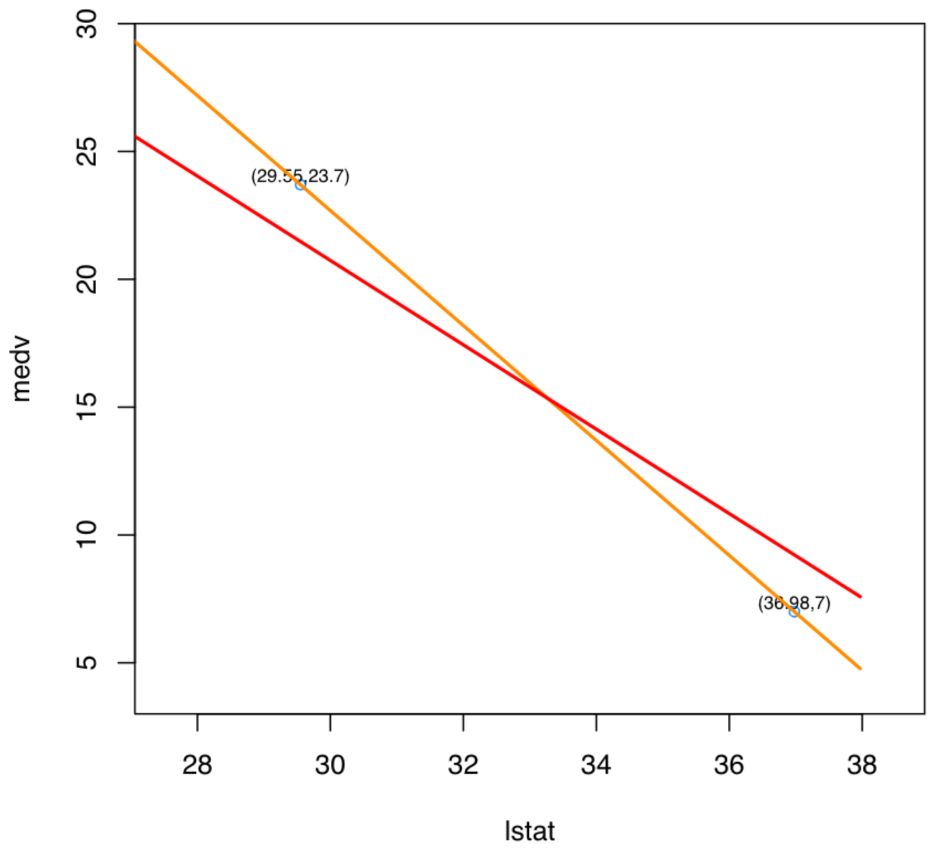
Ridge regression is less sensitive to *lstat*

- Linear regression fit: $\widehat{medv} = 90.118 - 2.248 \cdot lstat$

- One unit change in $lstat$ results in $-\ 2.248$ units change in $medv$

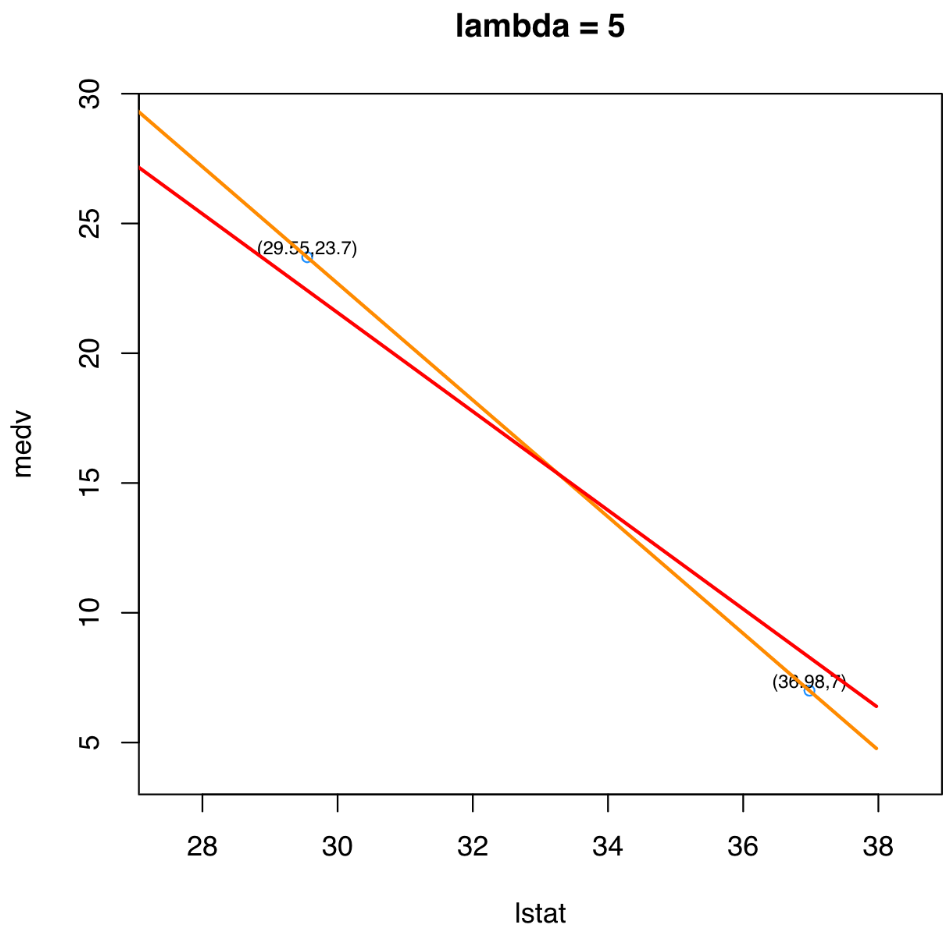
- Ridge regression fit: $\widehat{medv} = 70.234 - 1.650 \cdot lstat$

- One unit change in $lstat$ results in $-\ 1.650$ units change in $medv$



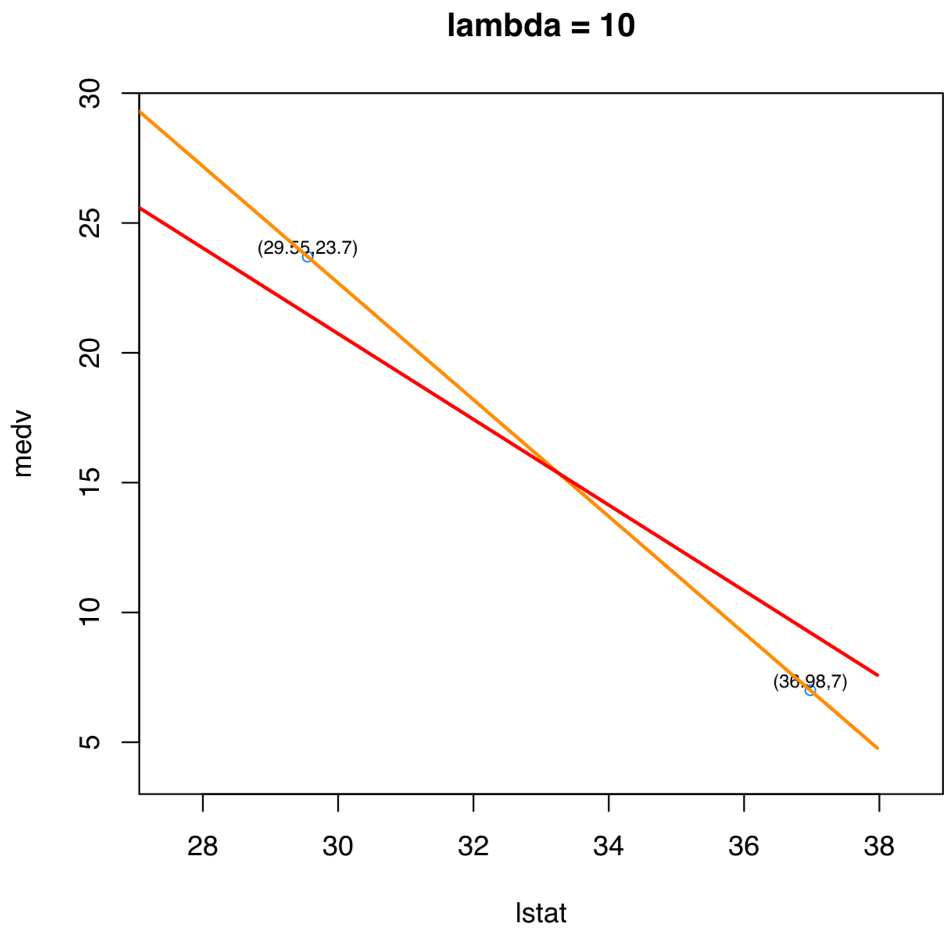
Role of λ in ridge regression

- Ridge regression minimizes
  - $\sum_{i=1}^{n}(medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$
  - $\lambda = 5$



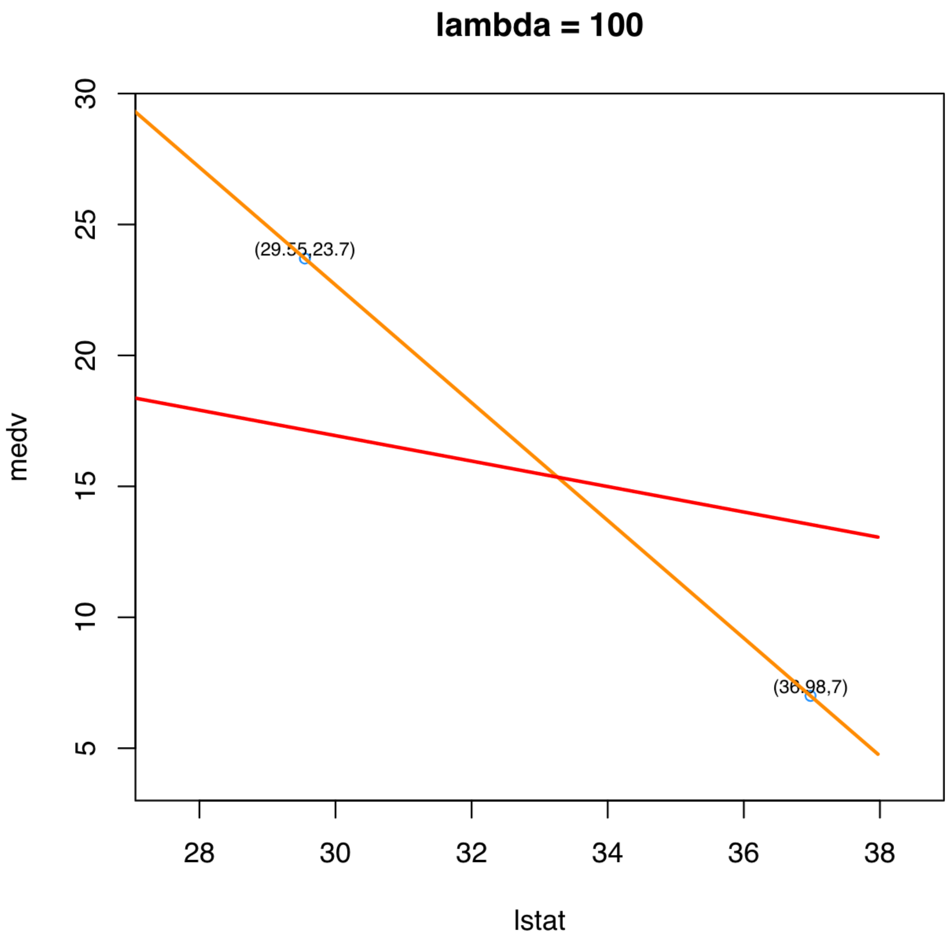
Role of λ in ridge regression

- Ridge regression minimizes
  - $\sum_{i=1}^{n}(medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$
  - $\lambda = 10$



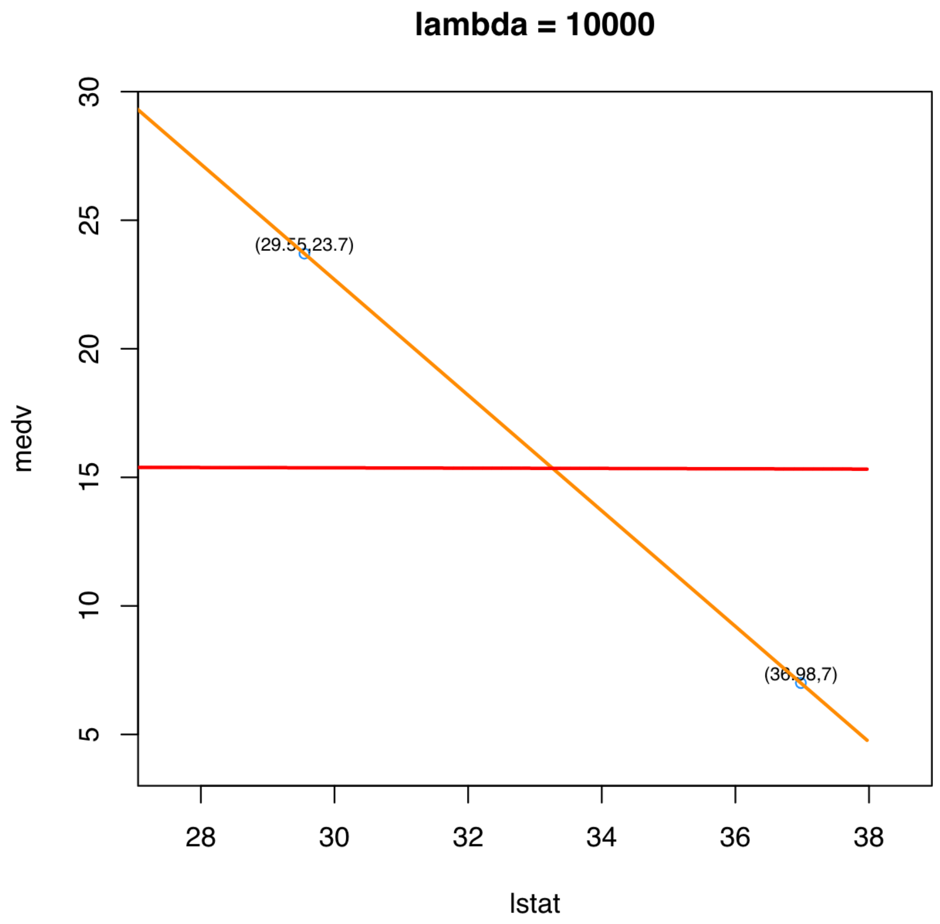
Role of λ in ridge regression

- Ridge regression minimizes
  - $\sum_{i=1}^{n}(medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$
  - $\lambda = 100$



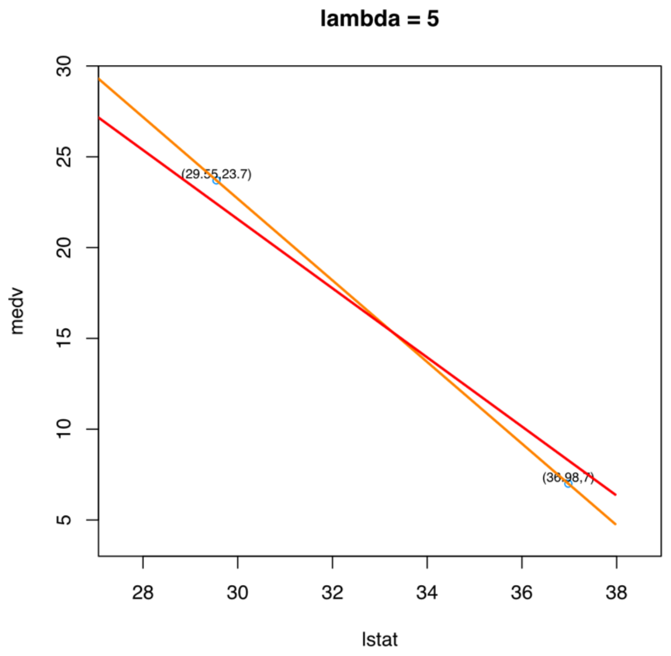
Role of λ in ridge regression

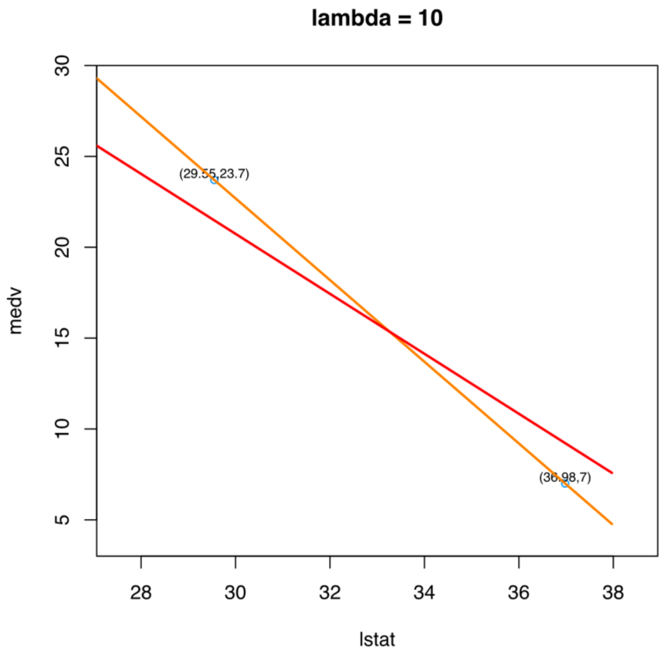
- Ridge regression minimizes
  - $\sum_{i=1}^{n}(medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$
  - $\lambda = 10{,}000$

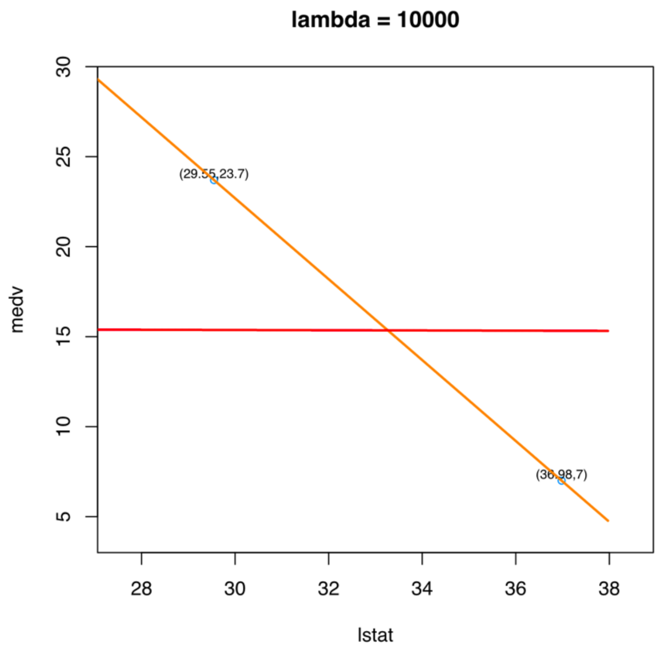


Predictive line is less sensitive to $\Delta lstat$ as λ increases

- Ridge regression minimizes: $\sum_{i=1}^{n}(medv_i - \beta_0 - lstat_i \cdot \beta_1)^2 + \lambda \cdot \beta_1^2$







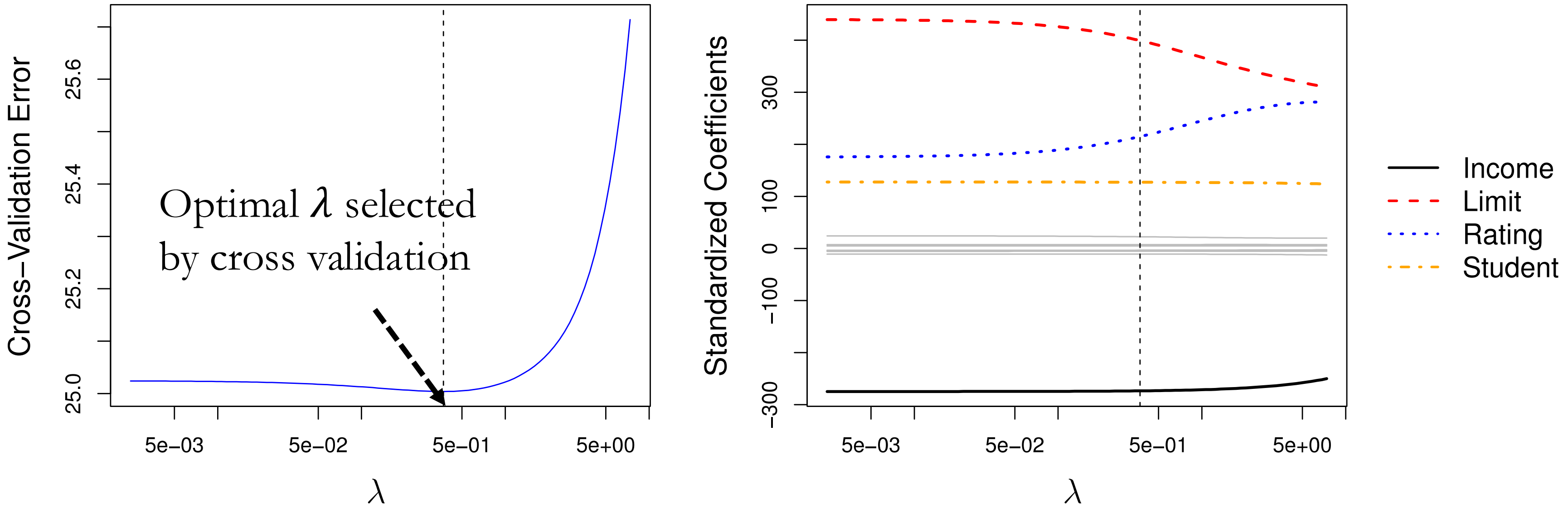
Choose λ by cross-validation

How to choose the optimal λ?

1. Select a grid of λ values
2. Compute the cross-validation error for each λ value
3. Select the λ with the smallest cross-validation error
4. Refit the model using all observations and selected λ

Example: Credit card dataset (ridge regression)

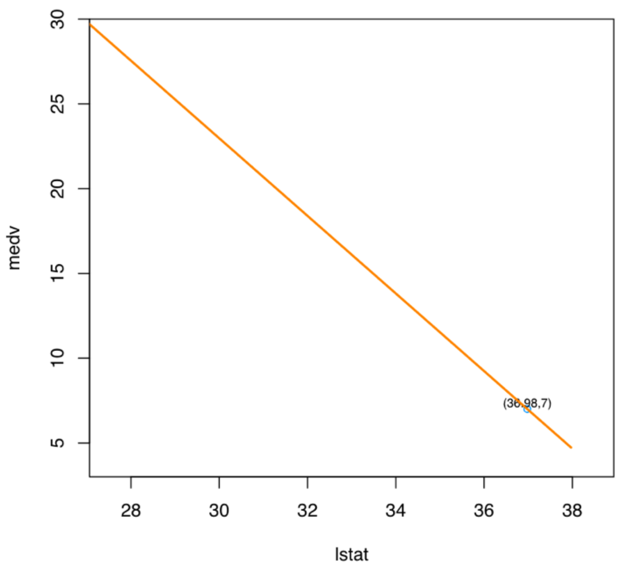
- Cross validation to choose the optimal λ

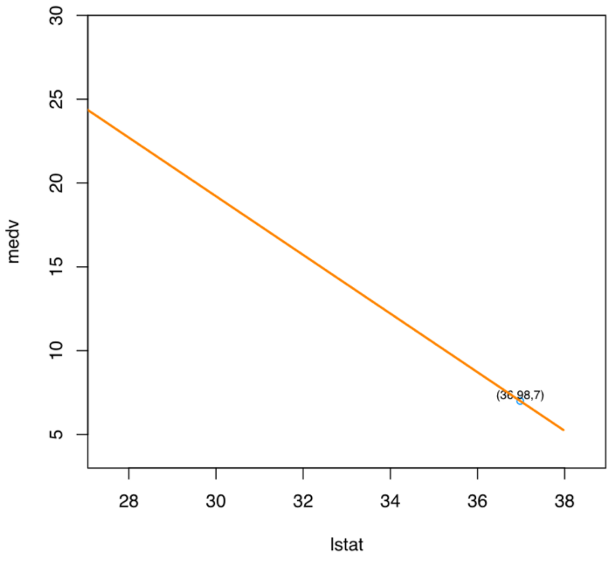


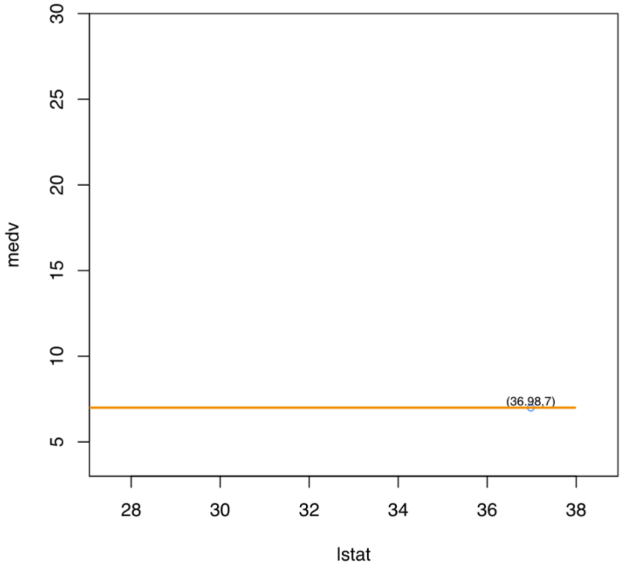
Quiz: Which line is the ridge regression fit?

- One observation ($n = 1$)

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
45.7461	0	18.1	0	0.693	4.519	100	1.6582	24	666	20.2	36.98	7







Transfer learning

- Transfer learning: use the information learned from one task to help learn another task
- Example #1: building a face recognition system from open-source models plus a few hundred labeled examples
- Example #2: fine-tuning a pre-trained language model for solving a downstream text prediction task
- Multitask learning: simultaneously train a multitask learning model on multiple objectives