Utrecht University

Linear regression for data science

Applied Data Analysis and Visualization I

Department of Methodology and Statistics
Javier Garcia-Bernardo

Today

What	When
Model fit and cross validation	Week 3
Linear regression for data science	Week 4
Classification	Week 5
Interactive visualizations with R shiny	Week 6
Tree-based methods	Week 7
...	...

Main points for today

1. Recap: Estimating E(MSE) and cross-validation
2. Linear regression
3. Which variables shall I include in my model?
   - Feature / subset selection
   - Shrinkage / Regularization methods: Lasso and Ridge
4. Conclusions

Recap

How to estimate the generalization error E(MSE) app.wooclap.com/ADAV2024

In a different dataset from the dataset used to train the model! (we use the “out of sample prediction error” to estimate the generalization error of our model)

General framework:

- Training dataset: To train the models
- Validation dataset: To select the best model
- Test dataset: To estimate the generalization error of the best model

Often, the terms “validation set” and “test set” are used interchangeably

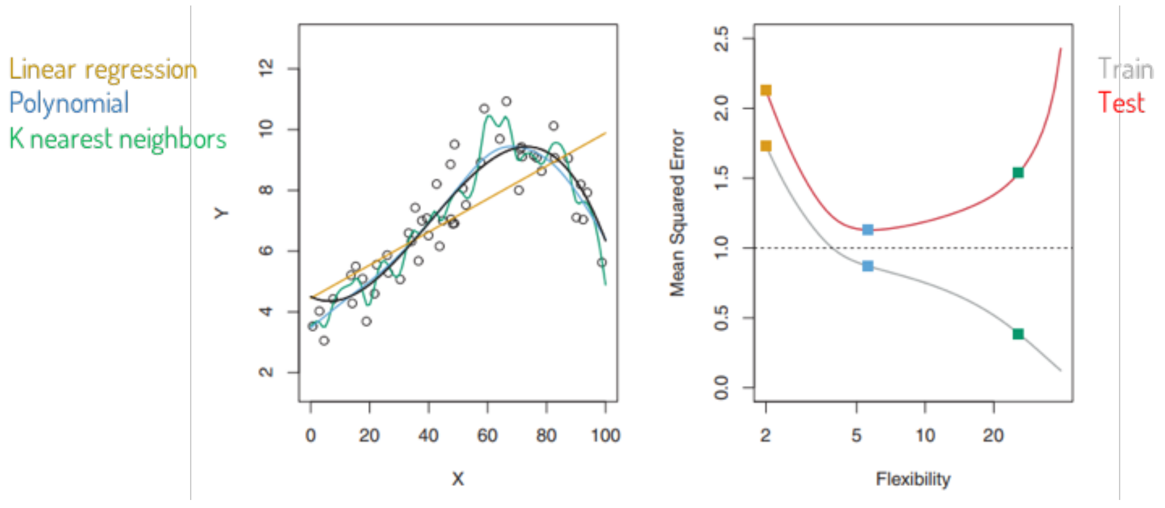
Key points:

- We cannot use the training data to evaluate the generalization error
- If we use some data to compare between models, the MSE of the best model is not an unbiased estimate of the generalization error. We need a new dataset (the test data) to estimate the generalization error.

Why? Mainly to avoid overfitting

> Suppose that you’re a teacher writing an exam for some students [models]. If you want to evaluate their skills, will you give them exercises [observations] that they have already seen [train set], and that they still have on their desks, or new exercises, inspired by what they learned, but different from them? [different set] (jpl, stackoverflow)

Complex models tend to overfit



See: https://javier.science/panel_bias_variance/

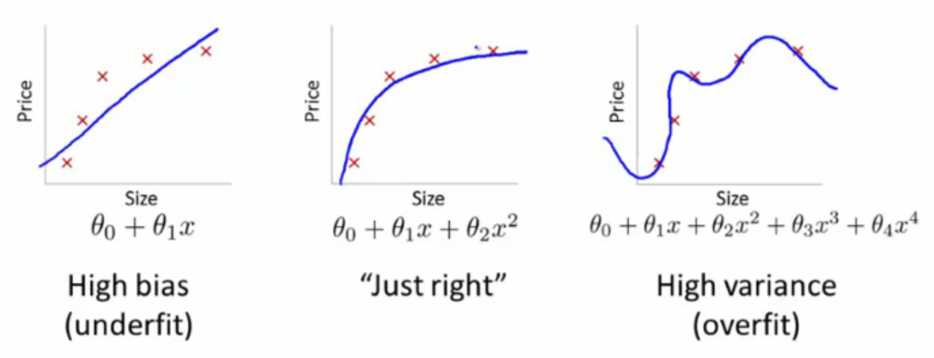
Complex models and bias vs variance

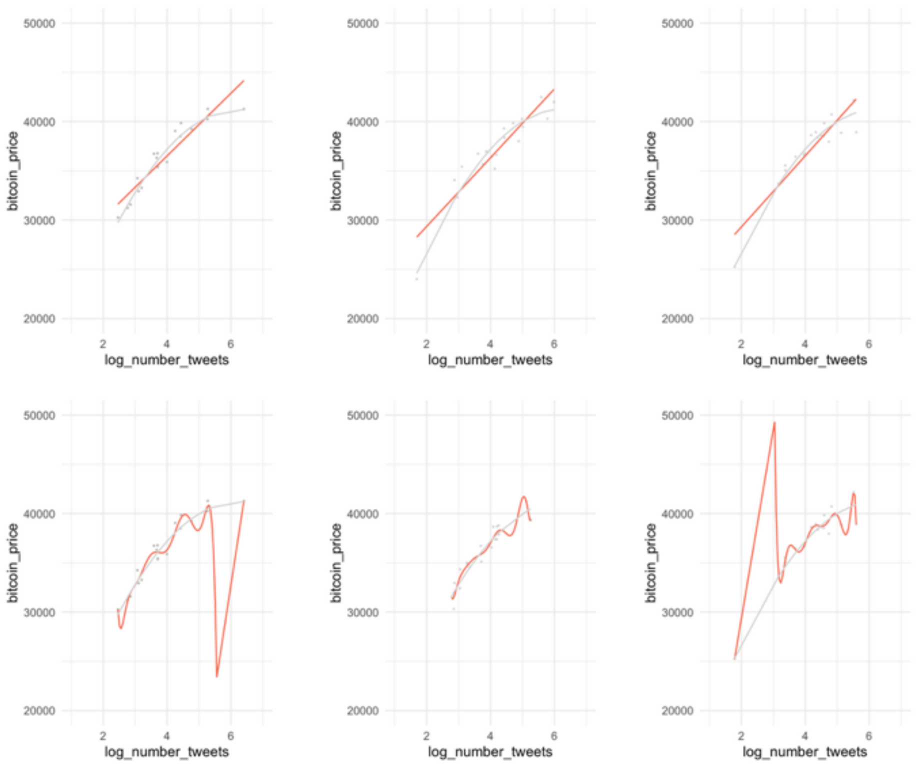
$E(MSE) = Bias^2(model) + Variance(model) + Variance(\epsilon)$

- **Bias**: The error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. High bias often results from underfitting.
- **Variance**: The sensitiviy of our model to small fluctuations in the training dataset. Since the training data are used to fit the statistical learning method, different training data sets will result in a different $\hat{f}$. High variance often comes from overfitting.
- **Variance(**
  ϵ): Irreductible error

Complex models and bias vs variance (cont.)

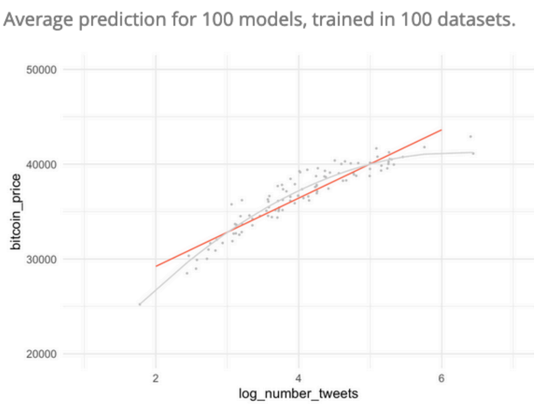
- Complex models will tend to overfit, and have high variance
- Simple models will tend to underfit, and have high bias

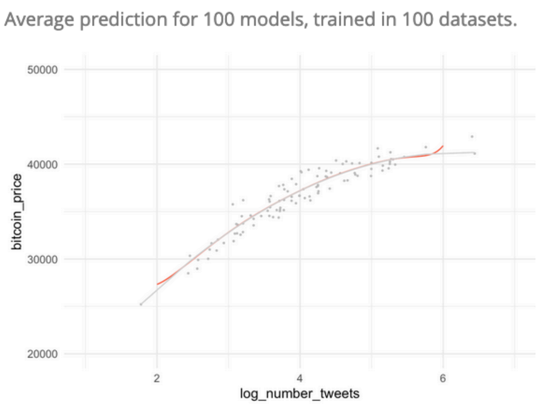




Which model (top/bottom) has high variance?

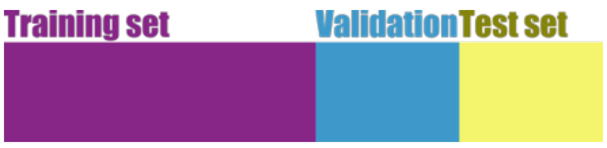
Which model has high bias?





Recap: Two alternatives to select the best model

Training/validation/test



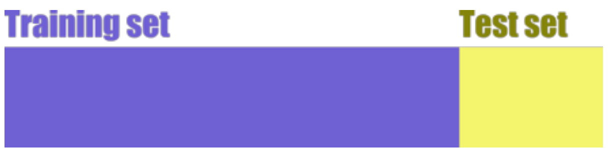
Training set: Train your models

Validation set: Compare the models (i.e. calculate MSE). Often re-used (but not too many times!).

Test set: Don’t touch it until the end!

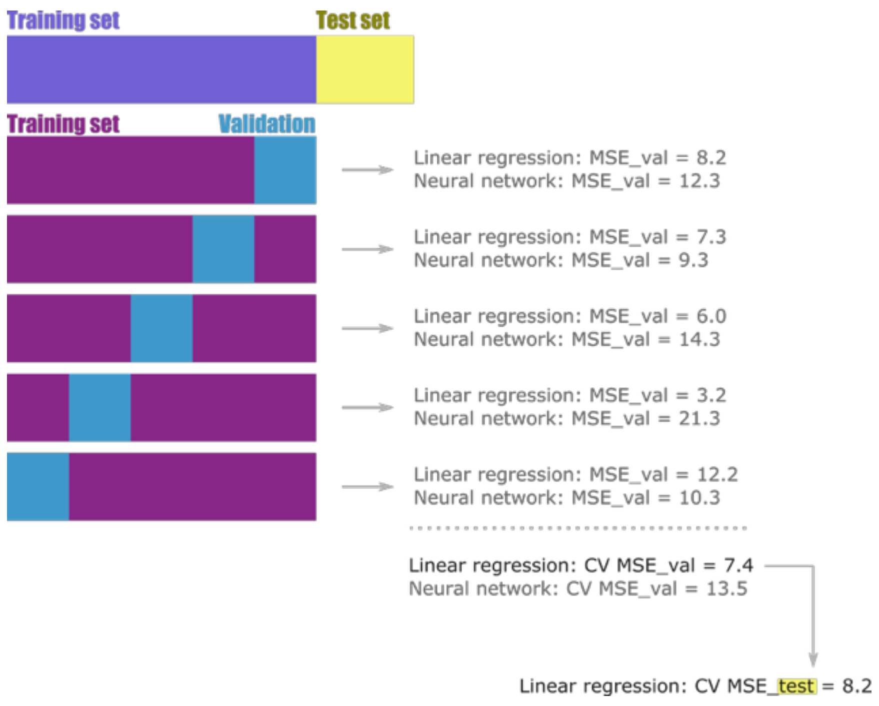
The distributions of values in the different splits should be similar

Cross-validation



Training set: Train and compare your models using cross-validation. Often re-used (but not too many times!).

Test set: Don’t touch it until the end! Estimate E(MSE) in the final model



Linear regression: MSE_val = 8.2
Neural network: MSE_val = 12.3

Linear regression: MSE_val = 7.3
Neural network: MSE_val = 9.3

Linear regression: MSE_val = 6.0
Neural network: MSE_val = 14.3

Linear regression: MSE_val = 3.2
Neural network: MSE_val = 21.3

Linear regression: MSE_val = 12.2
Neural network: MSE_val = 10.3

Linear regression: CV MSE_val = 7.4
Neural network: CV MSE_val = 13.5

Linear regression: CV MSE_test = 8.2

What do we mean with comparing models?

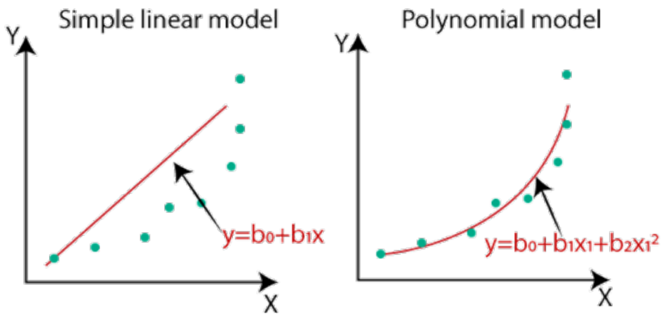
- Comparing statistical methods (e.g. linear regression vs knn regression)
- Comparing models with different predictors included (e.g. linear regression including predictors [X_1 , X_2] vs [X_1 , X_2 , X_3])
- Comparing two models with different hyperparameters (e.g. KNN regression using the closest 3 vs 10 neighbors)

Linear regression

Linear regression

$y = f(x) + \epsilon$

- Data
    - Y: Observed outcomes (dependent variable)
    - $X = x_1, \ldots, x_p$: p predictors (also called features, or independent variables)
    - f: function to estimate



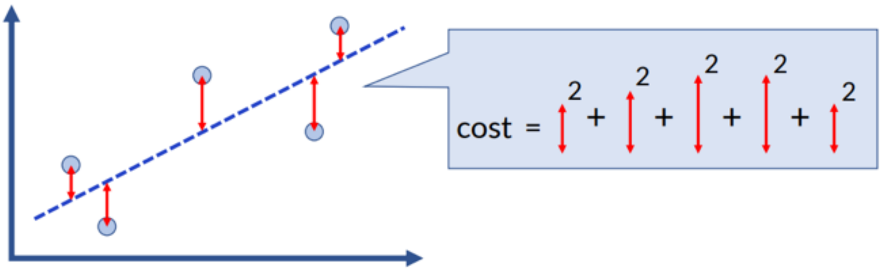
Least squares linear regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \ldots + \theta_d x_{id} = \sum_{j=0}^{d} \theta_j x_{ij}$$

Assume x_{i0} = 1

- Fit model by minimizing sum of squared errors

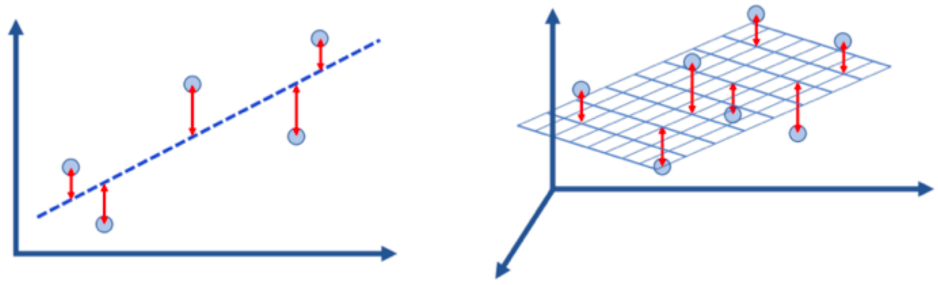


Least squares linear regression

Goal: Find the parameters θ that minimize the loss/cost/MSE

$$\mathcal{L}(\theta) = Cost(\theta) = MSE(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$$

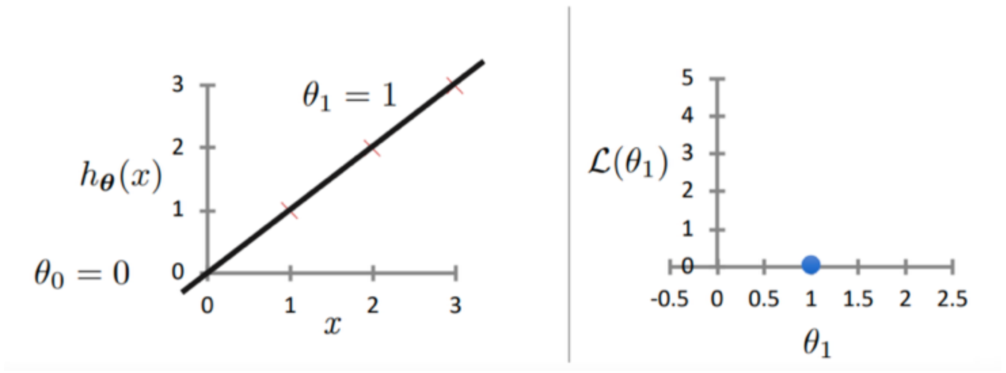
- y_i = observed value i
- $h_\theta(x_i)$ -> Prediction from our model for observation i (also seen as $\hat{f}(x_i)$ or $\hat{y}_i$)



Source: Eric Eaton, Applied Machine Learning course

Cost function: MSE

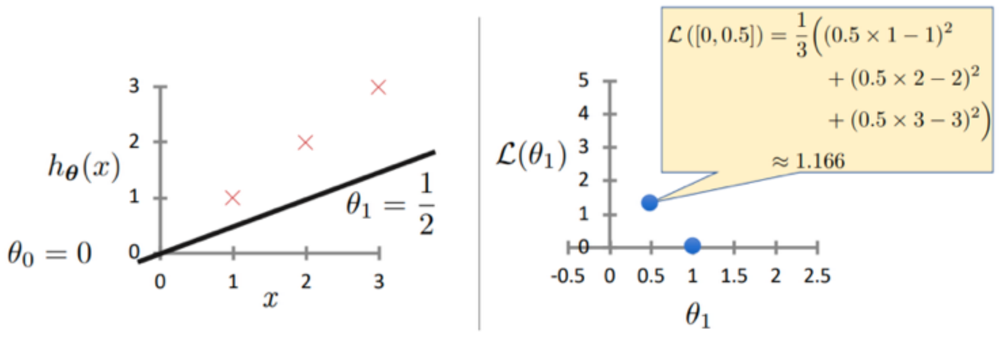
- Goal: Minimize MSE ($\mathcal{L}(\theta)$)
- $\mathcal{L}(\theta) = Cost(\theta) = MSE(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$



Source: Eric Eaton, Applied Machine Learning course

Cost function: MSE

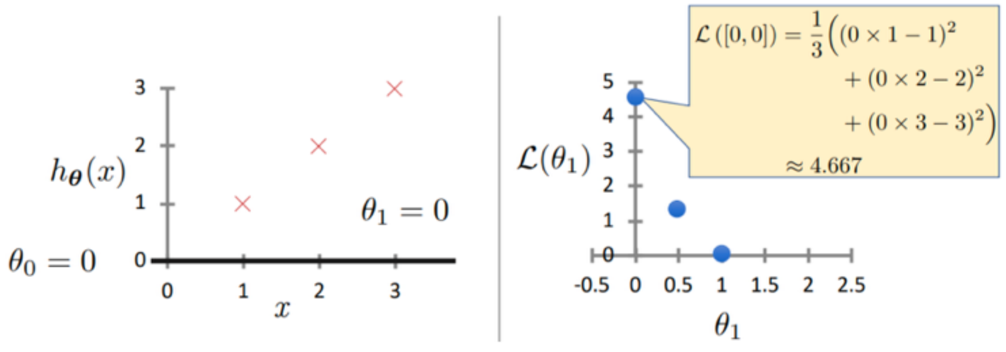
- Goal: Minimize MSE ($\mathcal{L}(\theta)$)
- $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$



Source: Eric Eaton, Applied Machine Learning course

Cost function: MSE

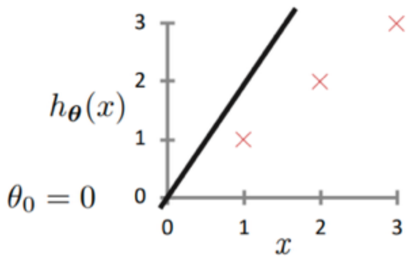
- Goal: Minimize MSE ($\mathcal{L}(\theta)$)
- $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$

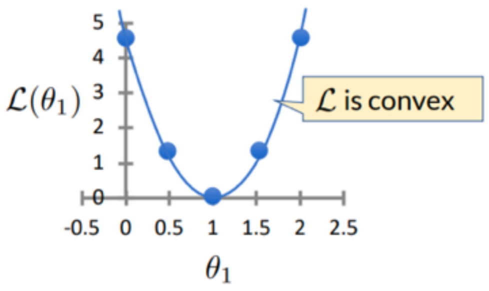


Source: Eric Eaton, Applied Machine Learning course

Cost function: MSE

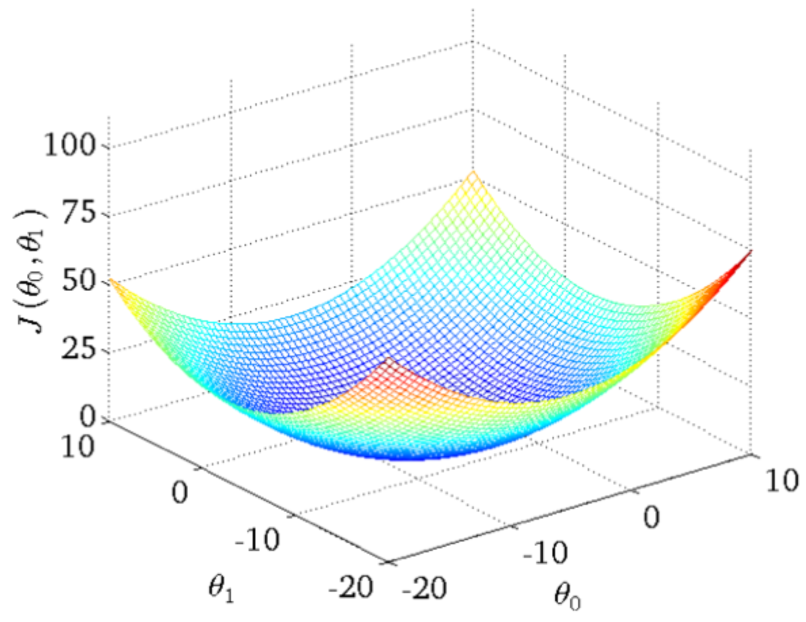
- Goal: Minimize MSE ($\mathcal{L}(\theta)$)
- $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$





Source: Eric Eaton, Applied Machine Learning course

The cost function is convex

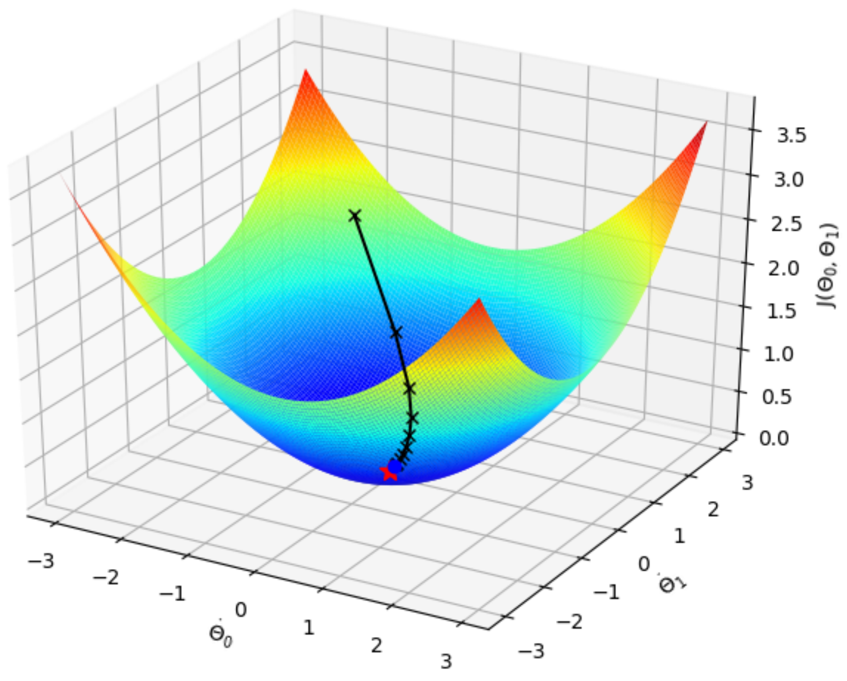


Source: Andrew Ng, Machine Learning course

How to find the optimal θ (the ones that minimize MSE)?

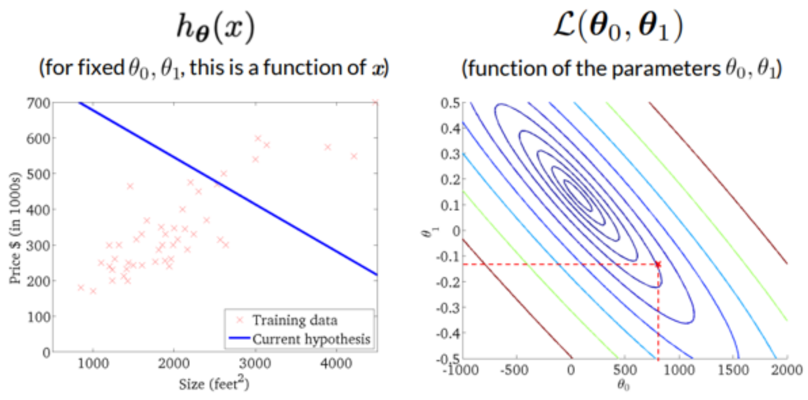
- Option 1: Using algebra
- Option 2: **Using gradient descent –**> optimization technique used in many machine learning methods

Gradient Descent: drop a marble in the curve



Gradient descent

- Initiate parameters θ randomly
- Predict $h_\theta(x)$ (i.e., $\hat{y}$) and calculate $\mathcal{L}(\theta)$
- Move to a lower point in the curve
- Repeat

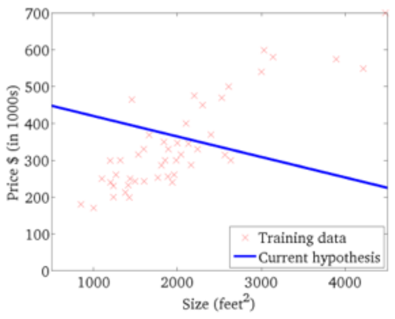


Source: Andrew Ng, Machine Learning course

Cost function

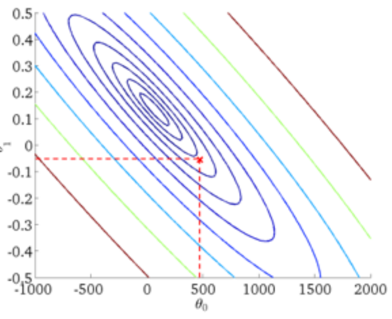
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_0, θ_1, this is a function of x)



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

(function of the parameters θ_0, θ_1)

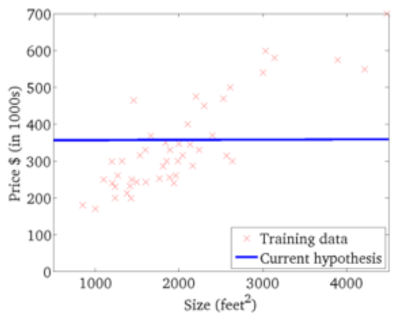


Source: Andrew Ng, Machine Learning course

Cost function

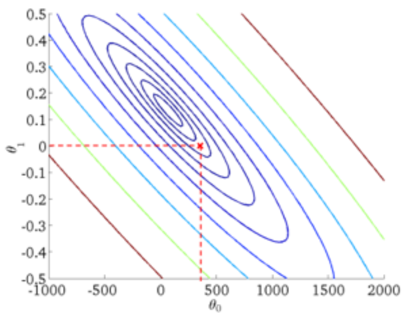
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

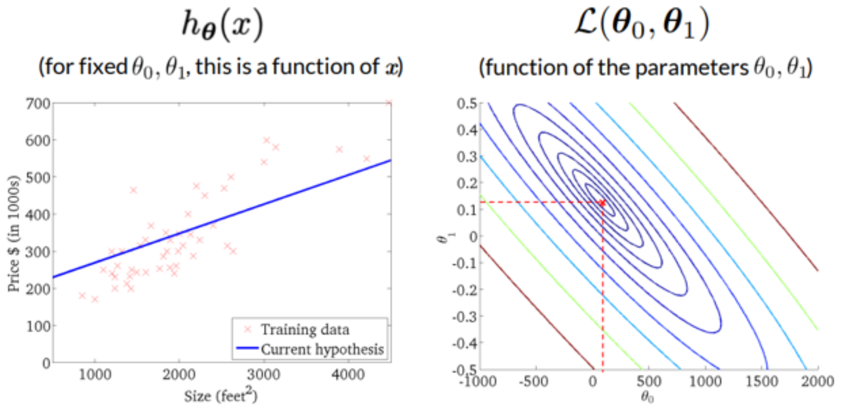
(function of the parameters θ_0, θ_1)



Source: Andrew Ng, Machine Learning course

Cost function

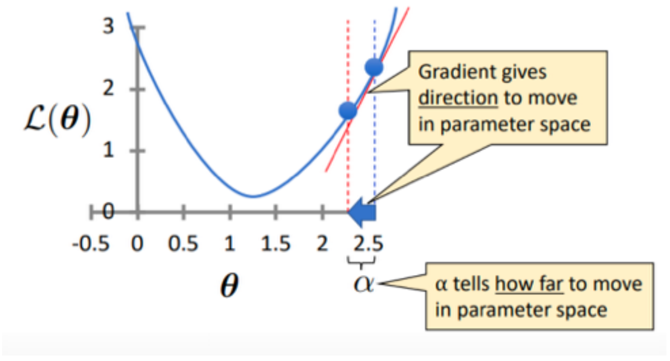
- When we get to the bottom, we stop



Source: Andrew Ng, Machine Learning course

How do we *actually* do this?

- Calculus!
  - We want to know where is “down” in the curve—i.e. the gradient of the curve ($\mathcal{L}(\theta)$) .
  - The gradient is the slope of the tangent line—i.e., the derivative of the MSE ($\mathcal{L}(\theta)$).
- In practice: For each coefficient in the model, θ_j
  - New θ_j = $\theta_j - \alpha \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}$
    - alpha = learning rate (a small value, can be adaptive)



$\mathcal{L}(\boldsymbol{\theta})$
3
2
1
0
-0.5 0 0.5 1 1.5 2 2.5
$\boldsymbol{\theta}$
α
Gradient gives direction to move in parameter space
α tells how far to move in parameter space

Where are we?

- We estimate the generalization error using out-of-sample prediction error (e.g. on a validation or test dataset; or through cross-validation)
- We can fit a linear regression using gradient descent

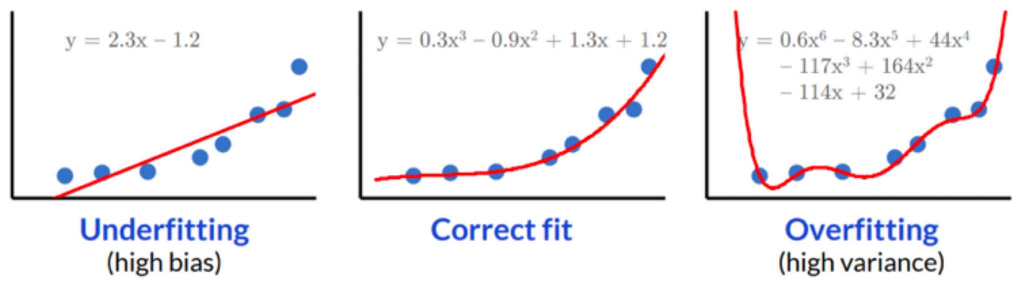
Now:

- Which variables shall I include in my model?
    - Feature (subset) selection
    - Shrinkage / Regularization methods: Lasso and Ridge
- Conclusions

Which variables shall I include in my model?

Quality of fit

- Including too few variables: Model with high bias
- Including too many variables: Model with high variance



Source: Eric Eaton, Applied Machine Learning course

Often we have too many variables

Which ones do we select in the model?

Why do we want to select variables instead of adding all?

https://app.wooclap.com/ADAV2024

A very flexible model (one with many coefficients) is like a kid in candyshop with a platinum credit card: It goes around buying all the coefficients it wants and never stops.

Idea: Tell the model not to go overboard with the complexity. We set up the correct complexity as the one that minimizes MSE in the validation data.

Constraining the complexity of the model

- **Feature selection**:
  - Pick the best p predictors
  - How: Best subset, forward selection, backward selection
- **Regularization**:
  - Constrain the sum of squared cofficients ($L2$) or absolute sum of coefficients ($L1$) to be below s
  - How: Adapt the loss function (e.g. MSE) to penalize including variables
- Dimensionality reduction:
  - Combine variables that correlate (covered in aDAV II)
  - How: Unsupervised learning

10 minute break

General framework of feature selection

- For each level of complexity (number of predictors):
    - Fit x models of equal complexity –> Keep the best using e.g. MSE or R^2
- Estimate E(MSE) for models of different complexity using cross-validation –> Select best model
- Estimate E(MSE) of the best model using test data

Feature selection I: best subset selection

1. Let M_0 denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots, p$ (p = total number of predictors):
   - Fit all $\binom{p}{k}$ models that contain exactly k predictors
     - fit all p models that contain exactly one predictor
     - fit all $\binom{p}{2} = p(p-1)/2$ models that contain exactly two predictors
     - and so forth
   - Pick the best among these $\binom{p}{k}$ models, and call it M_k (using e.g., smallest MSE or highest R2)
3. Select a single best model from among $M_0, \ldots, M_p$ using:
   - A measure that takes into consideration the number of predictors (e.g. AIC, BIC, adjusted-R^2)
   - Or better, cross-validated prediction error, e.g., MSE, or (1-R^2)

Credit data set

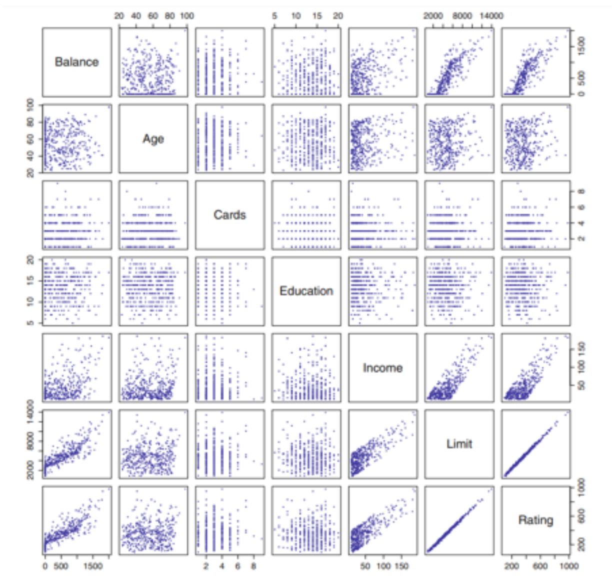
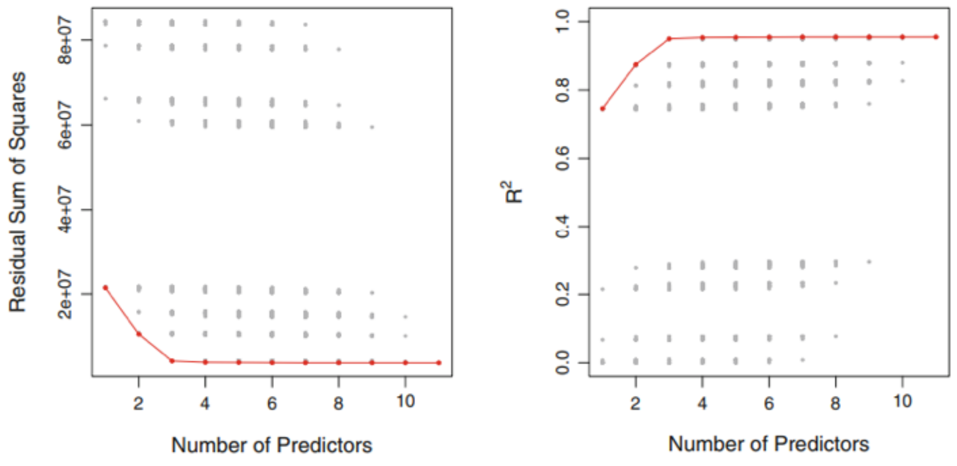


FIGURE 3.6. *The* Credit *data set contains information about* balance, age, cards, education, income, limit, *and* rating *for a number of potential customers.*

Best subset selection in the Credit data set



Train MSE in the credit data set

Careful! Train MSEs decrease with the complexity. We need to select using the validation MSE, not the train MSE!

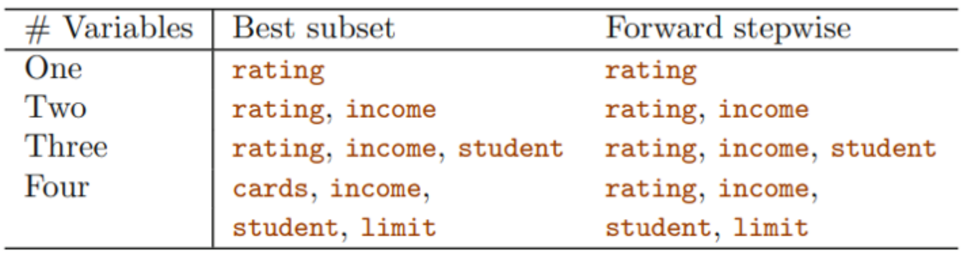
Feature selection II: Forward stepwise

1. Let M_0 denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 0, 1, 2, \ldots, p-1$:
    - Consider all $p - k$ models that augment the predictors in M_k with one additional predictor
    - Pick the best among these $(p - k)$ models, and call it M_{k+1} (using e.g., smallest MSE or highest R2)
3. Select a single best model from among $M_0, \ldots, M_p$ using:
    - A measure that takes into consideration the number of predictors (e.g. AIC, BIC, adjusted-R^2)
    - Or better, cross-validated prediction error, e.g., MSE or (1-R^2)

Best subset selection vs Forward stepwise

- Forward stepwise selection’s has a computational advantage (~$p^2/2$ vs 2^p models)
- Though forward stepwise tends to do well in practice, it is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors.
- For instance, suppose that in a given data set with p = 3 predictors, the best possible one-variable model contains X1, and the best possible two-variable model instead contains X2 and X3. Then forward stepwise selection will fail to select the best possible two-variable model, because M1 will contain X1, so M2 must also contain X1 together with one additional variable.

Best subset selection vs Forward stepwise



# Variables	Best subset	Forward stepwise
One	`rating`	`rating`
Two	`rating`, `income`	`rating`, `income`
Three	`rating`, `income`, `student`	`rating`, `income`, `student`
Four	`cards`, `income`, `student`, `limit`	`rating`, `income`, `student`, `limit`

TABLE 6.1. *The first four selected models for best subset selection and forward stepwise selection on the* `Credit` *data set. The first three models are identical but the fourth models differ.*

Feature selection III: Backward stepwise

1. Let M_p denote the full model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   - Consider all k models that contain all but one of the predictors in M_k, for a total of $k-1$ predictors
   - Pick the best among these k models, and call it M_{k-1} (using e.g., smallest RSS or highest R2)
3. Select a single best model from among $M_0, \ldots, M_p$ using:
   - A measure that takes into consideration the number of predictors (e.g. AIC, BIC, adjusted-R^2)
   - Or better, cross-validated prediction error, e.g., MSE or (1-R^2)

Feature selection - number of models

Number of models fitted at each step, example for a dataset with 20 predictors:

Method	Step 1	Step 2	Step 3	...
Best subset	k = 1 $\binom{20}{1} =$ 20 models	k = 2 $\binom{20}{2} =$ 190 models	k = 3 $\binom{20}{3} =$ 1140 models	$\ldots$ $\ldots$
Forward	k = 0 20 - 0 = 20 models	k = 1 20 - 1 = 19 models	k = 2 20 - 2 = 18 models	$\ldots$ $\ldots$
Backward	k = 20 20 models	k = 19 19 models	k = 18 18 models	$\ldots$ $\ldots$

Feature selection - pros and cons

- **Best subset**
  $+$ Finds the best subset, as advertised – when there is enough data to find it
  $-$ Need to fit 2^p models. With e.g., 20 predictors that is 1,048,576 regressions to run and evaluate. Not even mentioning squares, cubes, products, etc.

- **Forward/backward**
  $+$ Much more efficient, $O(p^2)$ instead of $O(2^p)$, e.g. 211 models for 20 predictors
  $-$ Not guaranteed to find the best subset

Forward vs. backward

- Backward sometimes not even possible (e.g. p > n)
- Forward can be fooled, especially when two variables work together but do nothing alone:
  - Backward considers performance of variable together with others.

- Both backward and forward are well-known to be **bad** at finding 'true' subset of predictors
  → Reveled in several fields (e.g. social science);
- For prediction, we do not care about the 'true' subset.

Cross-validation in subset selection

Consider a regression used to predict an outcome:

1. Starting with 5000 predictors and 500 cases, find the best 10 predictors
2. Fit a linear regression

How do we estimate the test set performance of this classifier?

https://app.wooclap.com/ADAV2024

Answer: You cannot use the same data to find the best predictors and to estimate E(MSE). Once a model has seen the data, you cannot use the same data to estimate E(MSE). First split the data, always.

Penalized (regularized) regression: buying coefficients on a budget

- We want to fit the training data (estimate the weights of the coefficients)
- Make the model behave ‘regularly’ by penalizing the purchase of ‘too many’ coefficients
- Extremely efficient way to approximately solve the best subset problem: Variable selection + regression in one step
- Often yields very good results

- If you are interested in prediction and not inference (i.e. if identifying the relevant features is not a primary goal of the analysis), regularization will usually be better

Regularization: buying coefficients on a budget

Usually, find the θ_j (or sometimes we use the notation β_j) that minimizes

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

Now, find the θ_j that minimizes

$\mathcal{L}(\theta) = MSE + \lambda \cdot Penalty$

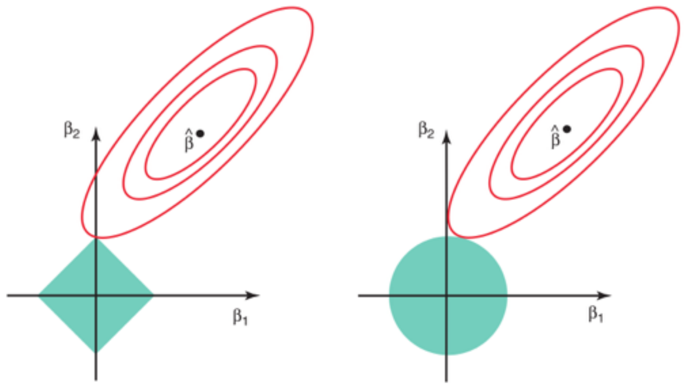
where the penalty is:

- $\sum_{j>0} |\theta_j|$ –> (L1 Lasso) Tends to set some coefficients to zero (great for interpretability)
- $\sum_{j>0} \theta_j^2$ –> (L2, Ridge) Tends to keep all coefficients

Equivalent, minimize MSE subject to:

- $\sum_{j>0} |\theta_j| < s$ –> (L1 Lasso) Tends to set some coefficients to zero (great for interpretability)
- $\sum_{j>0} \theta_j^2 < s$ –> (L2, Ridge) Tends to keep all coefficients

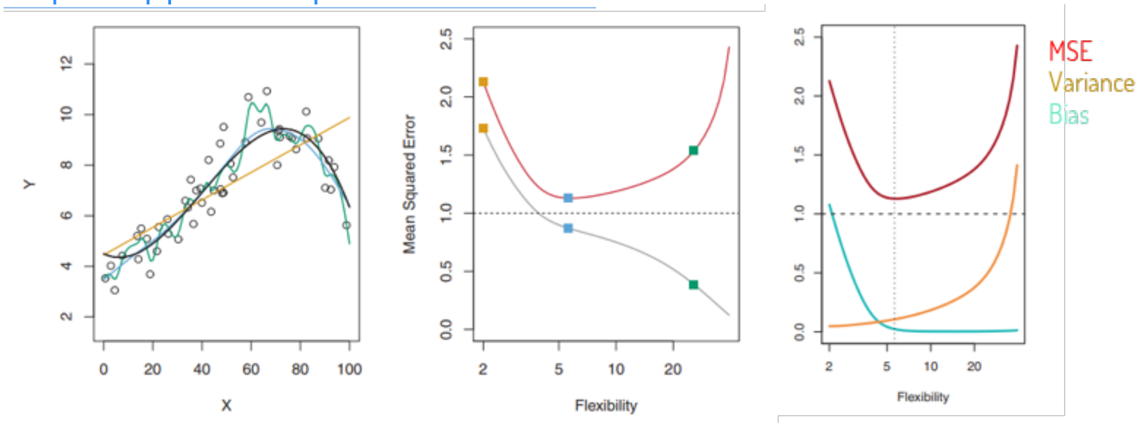
s = budget, “buying” coefficients cannot cost more than that



Increasing λ will

Tip: Think about what happens when λ increases. Do you get more or less predictors in the model?

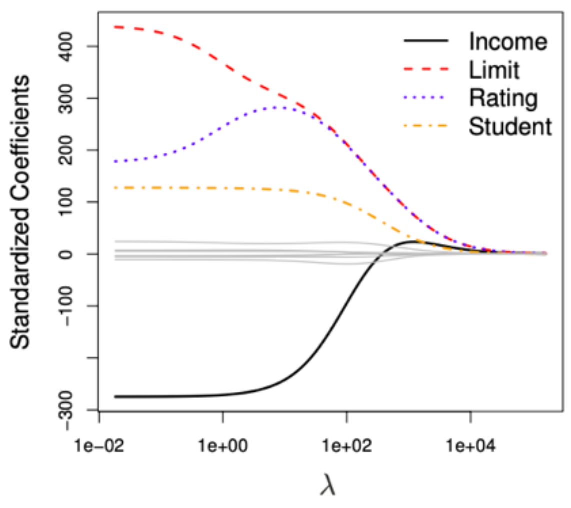
https://app.wooclap.com/ADAV2024



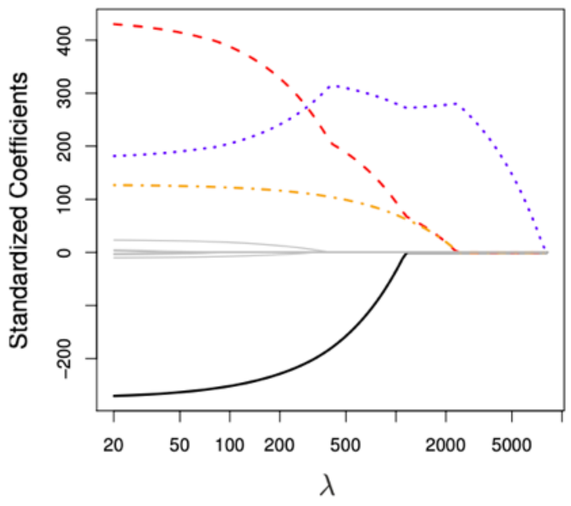
Increasing λ (decreasing s) will reduce the number of coefficients selected: Increase bias, decrease variance

Penalization as 'shrinkage' to zero

Ridge (L2)



Lasso (L1)



How to select λ

Option 1:

- Divide the data into train/val/test
- Create models using different λ, fit them using the train data.
- Calculate MSE in the validation data and select the best model.
- Estimate generalization error for the best model in the test datast.

Option 2 (better):

- Divide the data into train/test
- Use cross-validation, for each k split of train –> train/val:
    - Fit models using different λ.
    - Calculate MSE in the validation dataset
- Select the model with the minimum average MSE.
- Estimate generalization error in the test datast

Standardizing predictors

We want to understand the effect of temperature in humidity.

If we use normal (least squares) regression, would it make a difference if we measure temperature in C or F?

If we use penalized regression, would it make a difference if we measure temperature in C or F?

In penalized regression, adding predictors with a smaller mean (with a larger θ) costs more. It is a good idea to standardize your predictors to have a standard deviation of one.

Penalization in `R`

LASSO:

```r
fit <- glmnet(x, y, alpha = 1, lambda = 1.5)
```

Ridge:

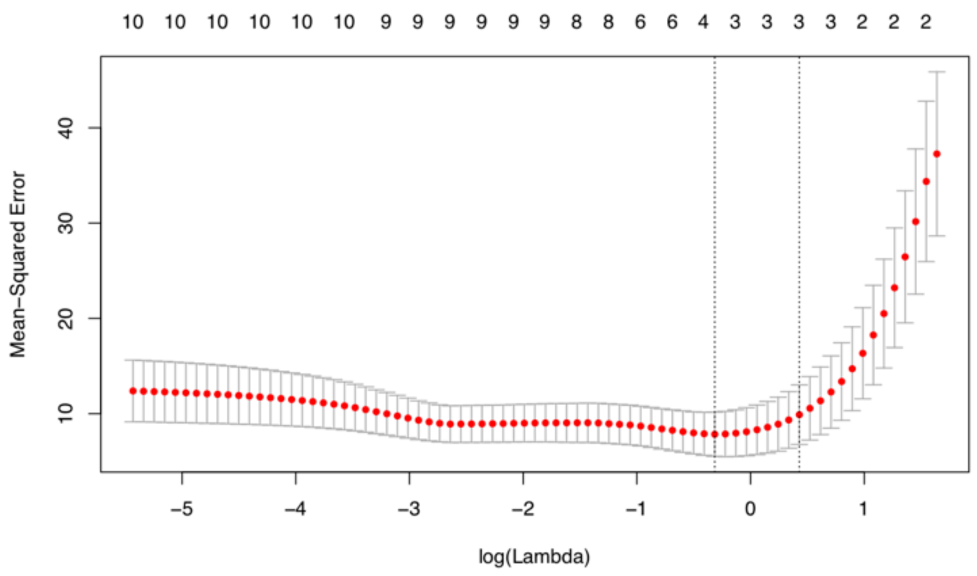
```r
fit <- glmnet(x, y, alpha = 0, lambda = 1.5)
```

- LASSO and ridge have a tuning parameter λ
- The usual least squares is $\lambda = 0$
- Higher λ $\rightarrow$ stricter penalty $\rightarrow$ smaller budget s
- Higher λ 'shrinks' coefficients to 0
- Can select λ using cross-validation (`cv.glmnet()`)

mtcars example: penalization using `glmnet`

	Least squares	LASSO	Ridge
(Intercept)	12.30	33.59	21.20
cyl	-0.11	-0.84	-0.34
disp	0.01	.	-0.01
hp	-0.02	-0.01	-0.01
drat	0.79	.	1.03
wt	-3.72	-2.31	-1.44
qsec	0.82	.	0.19
…	…	…	…

mtcars example: selecting λ with cross-validation



Conclusions

- Gradient descent is a very efficient way to fit/train models
- By using feature selection or regularization, we can obtain better prediction accuracy and model interpretability
- Feature selection includes best subset, forward and backward selection
  - Best subset selection performs best, but it comes at a prize
- Regularization includes LASSO and Ridge
  - LASSO shrinks unimportant parameters to truly zero, while Ridge shrinks them to small values

Next class

Next week:

Supervised learning method: **Classification** (with Dr. Anastasia Giachanou).

Have a nice day!