PSY9511: Seminar 3

Regularization and variable selection

Esten H. Leonardsen

September 8, 2025

UNIVERSITETET
I OSLO

Outline

1. Assignment 2
2. Regularization
    - Variable selection
    - Shrinkage
3. Assignment 3

Assignment 2

UNIVERSITETET
I OSLO

Assignment 2

1. Download the Auto.csv dataset from the ISLP website.
2. Read the Auto.csv-dataset into memory.
3. In the horsepower-column, some values are missing. These are encoded with '?'. Remove these rows from the dataset.
4. Create a new column 'muscle'. This column should contain a 1 for all muscle cars (e.g. cars that have above average horsepower) and 0 for the rest.
5. Split the dataset into a training set and a test set, by randomly drawing 80% of the rows for the former and 20% of the rows for the latter.

http://localhost:8888/notebooks/notebooks%2FAssignment2.ipynb

Assignment 2

1. Fit a simple linear regression model using horsepower as the predictor and mpg as the outcome using the training data.
2. Create a scatter plot with horsepower on the x-axis and mpg on the y-axis using the testing data. Plot the regression line found by the model in the plot.
3. Use the model to generate predictions for the training set. Calculate and report the mean absolute error (MAE) of these predictions.
4. Use the model to generate predictions for the test set. Calculate and report the MAE of the predictions.

http://localhost:8888/notebooks/notebooks%2FAssignment2.ipynb

Assignment 2

Is the training or testing MAE is lower? Does this match your expectation? What would be the general pattern we expect here (e.g. one is lower than the other, they are the same, etc.), and why do we expect that?

Assignment 2

1. Fit a multivariate linear regression model using horsepower, weight, displacement, and year as predictors and mpg as the outcome.
2. Print the intercept and coefficients of the model .
3. Use the model to generate predictions for the training set. Calculate and report the MAE of these predictions.
4. Use the model to generate predictions for the test set. Calculate and report the MAE of the predictions.

http://localhost:8888/notebooks/notebooks%2FAssignment2.ipynb

Assignment 2

Is the training MAE lower or higher than in the simple linear regression model? Does it have to be this way, or could it have been otherwise? What about the testing MAE?

Assignment 2

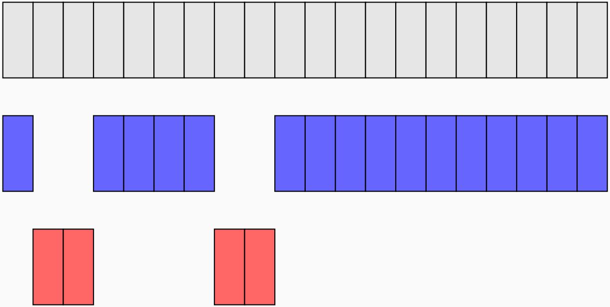
1. Fit a logistic regression model using weight, displacement and year as predictors and our newly created muscle-column as the outcome.
2. Use the model to generate predictions for the training set. Calculate and report the accuracy of these predictions.
3. Use the model to generate predictions for the testing set. Calculate and report the accuracy of these predictions.

http://localhost:8888/notebooks/notebooks%2FAssignment2.ipynb

Assignment 2: Random seeds

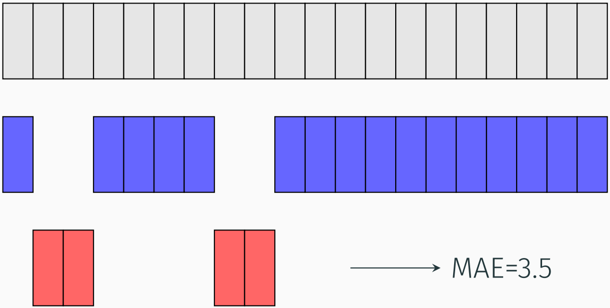
In[1]:

```python
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



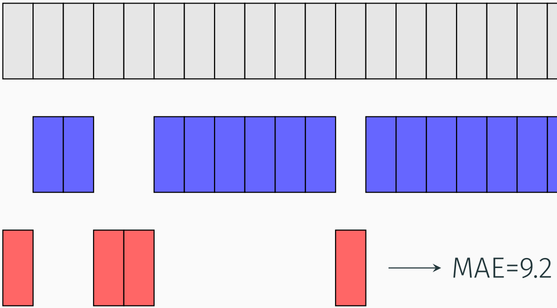
Assignment 2: Random seeds

In[1]:
```python
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



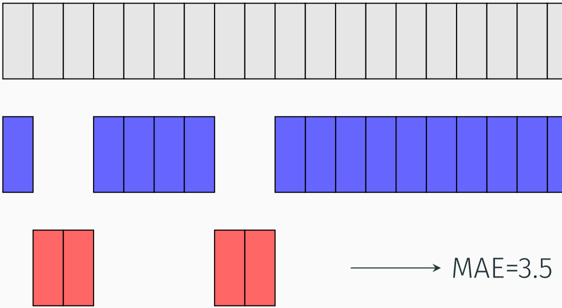
Assignment 2: Random seeds

In[1]:
```python
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



Assignment 2: Random seeds

In[1]:
```python
import pandas as pd
import numpy as np
np.random.seed(42)
df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11 
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223 
```

Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11 
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223 
```

```r
predict(model, df, type="response")
```

```
     1      2      3      4      5      6      7      8      9     10     11 
0.7766 0.8726 0.7314 0.7293 0.7405 0.9853 0.9844 0.9895 0.9233 0.8112 0.8352 
```

Assignment 2: Log-odds vs probability vs class

```
In[1]:  model = LogisticRegression()
        model.fit(df[['year', 'weight']], df['muscle'])
        model.predict(df[['year', 'weight']])

Out[1]: array([0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0])
```

```
In[1]:  model.predict_proba(df[['year', 'weight']])

Out[1]: array([[0.14, 0.86], [0.08, 0.92], [0.17, 0.83], [0.18, 0.82]])
```

Assignment 2: Eye test

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

Assignment 2: Eye test

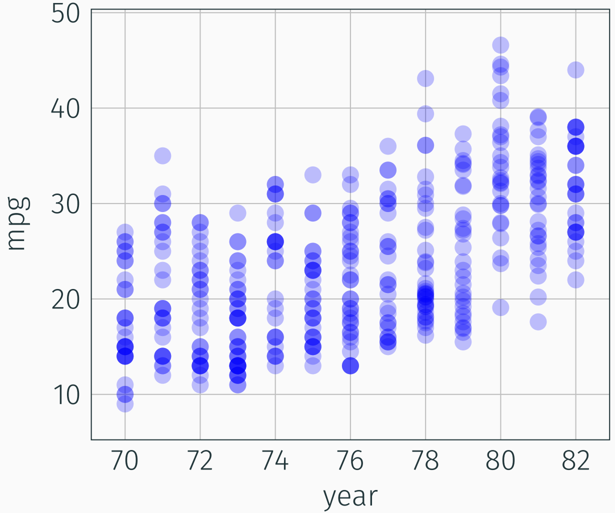
```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

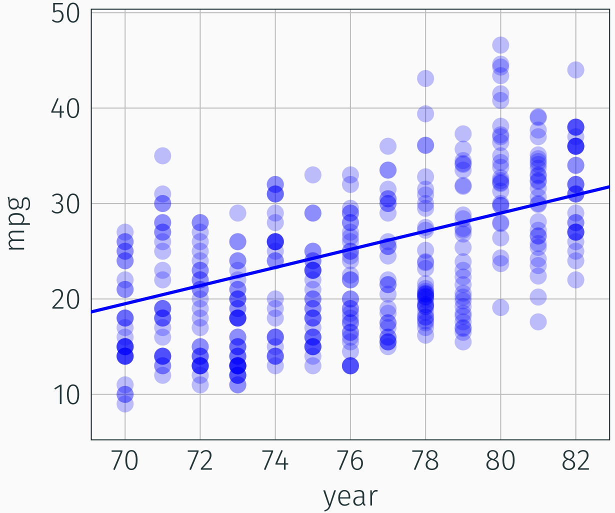
```r
model <- lm(mpg ~ horsepower, df)
summary(model)
```

```
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.59412 0.96187 20.371 < 2e-16 ***
horsepower102 0.40588 4.08087 0.099 0.920840
horsepower103 0.70588 4.08087 0.173 0.862789
horsepower105 0.90588 1.49529 0.606 0.545091
```

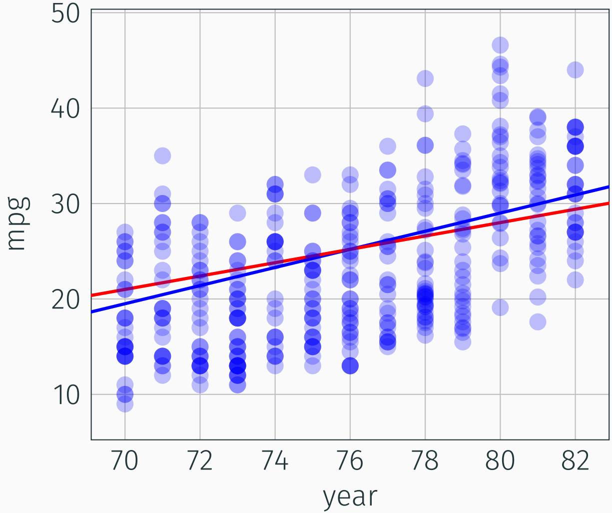
Assignment 2: Interpretation



Assignment 2: Interpretation



Assignment 2: Interpretation



Regularization

Regularization: Background

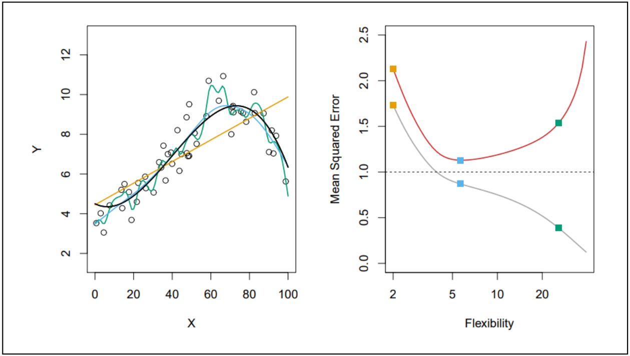
Regularization: A set of techniques to explicitly reduce the complexity of our models (to reduce overfitting), by encouraging them to learn broader and more general patterns in the training data instead of memorizing it

Regularization: Background

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

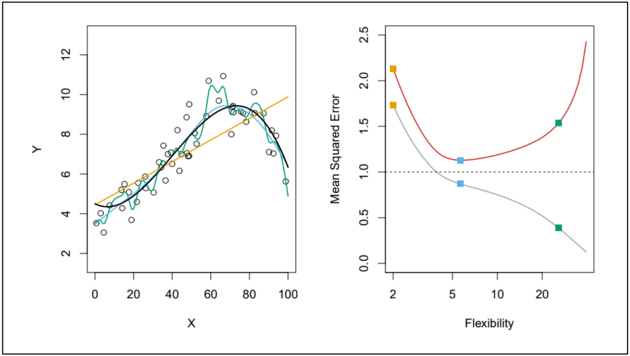
Regularization: Background

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$



Regularization: Background

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 + \beta_4 * x_4$$



Regularization: Background

In[1]:
```python
import pandas as pd


df = pd.read_csv('/Users/esten/Downloads/Auto.csv')
train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]

print(f'Using len(train) samples for training')
print(f'Using len(validation) samples for validation')
```

Out[1]:
```
Using 317 samples for training
Using 80 samples for validation
```

Regularization: Background

1. Variable selection
   a. Best subset selection
   b. Forward stepwise selection
   c. Backward stepwise selection
2. Shrinkage
   a. LASSO
   b. Ridge regression
3. Dimensionality reduction: Lecture 6 and self-study

Regularization: Background

1. Variable selection
    a. Best subset selection
    b. Forward stepwise selection
    c. Backward stepwise selection
2. Shrinkage
    a. LASSO
    b. Ridge regression
3. Dimensionality reduction: Lecture 6 and self-study

Variable selection

The number of predictors we are using in our linear model directly impacts its complexity, allowing us to balance the bias-variance trade off by varying the number of predictors.

Variable selection: Motivation

The number of predictors we are using in our linear model directly impacts its complexity, allowing us to balance the bias-variance trade off by varying the number of predictors.

Fewer predictors also mean that

- We avoid problems due to collinearity.
- There are less coefficients to be interpreted.

Variable selection: Motivation

The number of predictors we are using in our linear model directly impacts its complexity, allowing us to balance the bias-variance trade off by varying the number of predictors.

Fewer predictors also mean that

- We avoid problems due to collinearity.
- There are less coefficients to be interpreted.

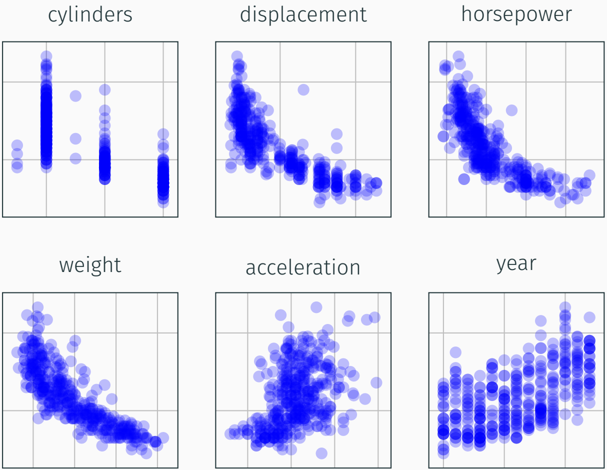
Formal problem specification

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Formal problem specification

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Variable selection: Motivation



Variable selection: Best subset selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Train models on all subsets p and select the best one.

Variable selection: Best subset selection

In[1]:

```
import numpy as np

from itertools import chain, combinations
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

subsets = list(chain.from_iterable(combinations(predictors, r)
               for r in range(len(predictors)+1)))

best = {'mse': float('inf'), 'subset': None}

for subset in subsets:
    if len(subset) == 0:
        continue

    model = LinearRegression()
    model.fit(train[list(subset)], train[target])
    predictions = model.predict(validation[list(subset)])
    mse = mean_squared_error(validation[target], predictions)

    best = best if mse > best['mse'] else {'mse': mse, 'subset': subset}

print(f'MSE: {best["mse"]:.2f}, predictors: {best["subset"]}')
```

Out[1]:

```
MSE: 29.68, predictors: ('cylinders', 'displacement', 'horsepower', 'weight', 'year')
```

Variable selection: Best subset selection

In[1]:

```
import numpy as np

from itertools import chain, combinations
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

subsets = list(chain.from_iterable(combinations(predictors, r)
               for r in range(len(predictors)+1)))

best = {'mse': float('inf'), 'subset': None}

for subset in subsets:
    if len(subset) == 0:
        continue

    model = LinearRegression()
    model.fit(train[list(subset)], train[target])
    predictions = model.predict(validation[list(subset)])
    mse = mean_squared_error(validation[target], predictions)

    best = best if mse > best['mse'] else {'mse': mse, 'subset': subset}

print(f'MSE: {best["mse"]:.2f}, predictors: {best["subset"]}')
```

Out[1]:

```
MSE: 29.68, predictors: ('cylinders', 'displacement', 'horsepower', 'weight', 'year')
```

Variable selection: Best subset selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Train models on all subsets p and select the best one.

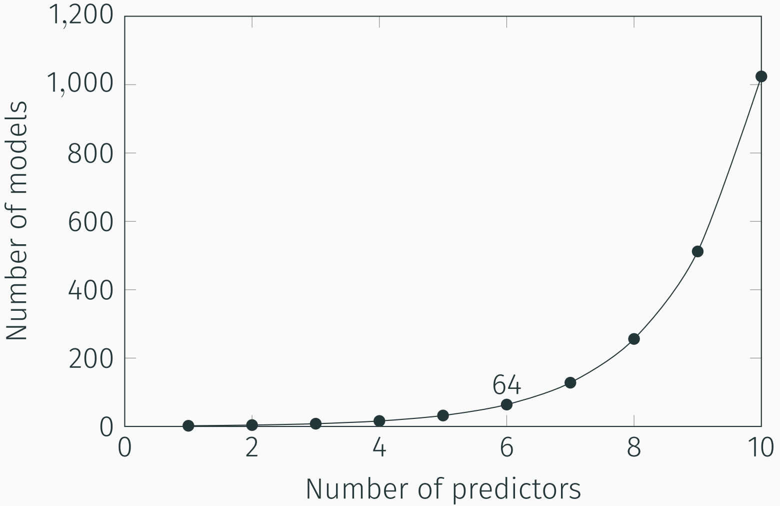
\+ **Positives**

- Guaranteed to find the optimal solution.
- Simple implementation

– **Drawbacks**

- Need to train many ($2^{|P|}$) models.

Variable selection: Best subset selection



Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start by fitting a model with no predictors. Iteratively add the predictor that yields the best model until all are included.

Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

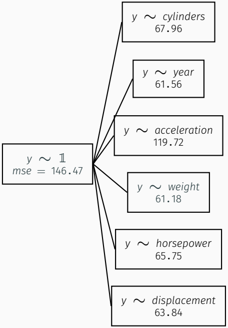
Solution

Start by fitting a model with no predictors. Iteratively add the predictor that yields the best model until all are included.

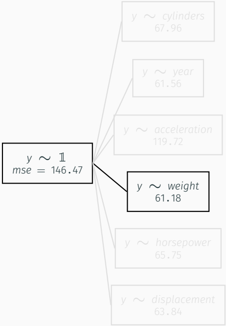
Variable selection: Forward stepwise selection

$$\boxed{\begin{array}{c} y \sim \mathbb{1} \\ mse = 146.47 \end{array}}$$

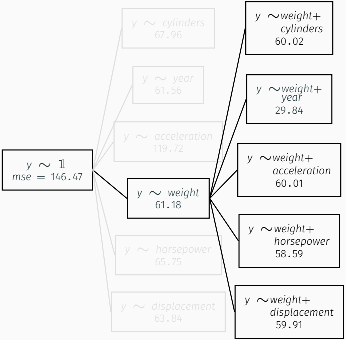
Variable selection: Forward stepwise selection



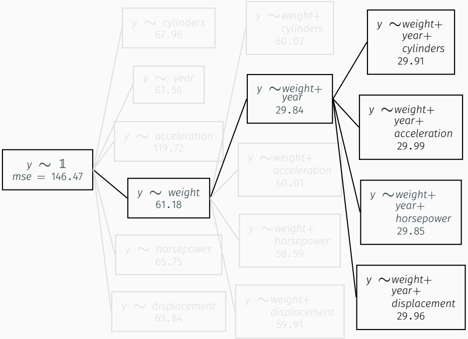
Variable selection: Forward stepwise selection



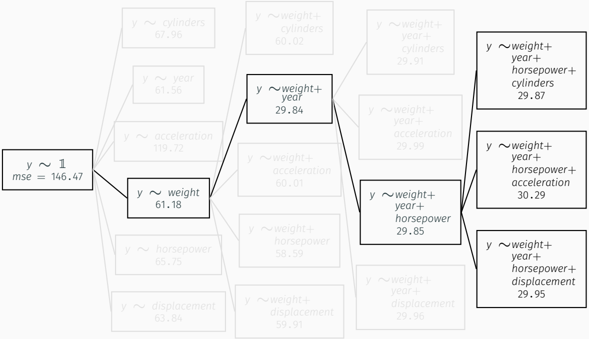
Variable selection: Forward stepwise selection



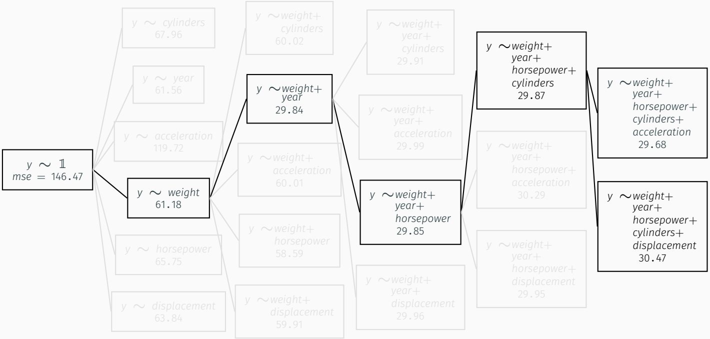
Variable selection: Forward stepwise selection



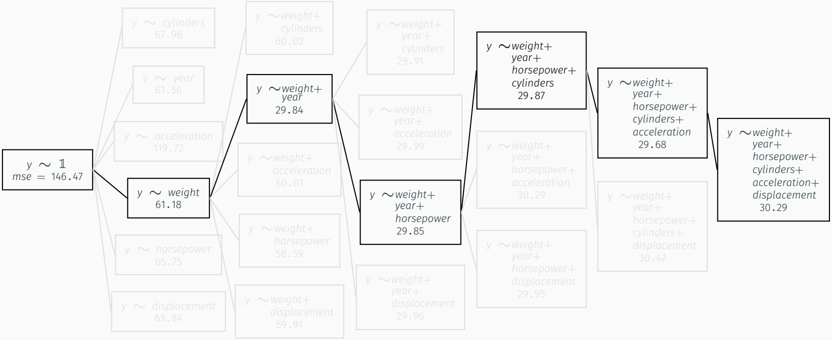
Variable selection: Forward stepwise selection



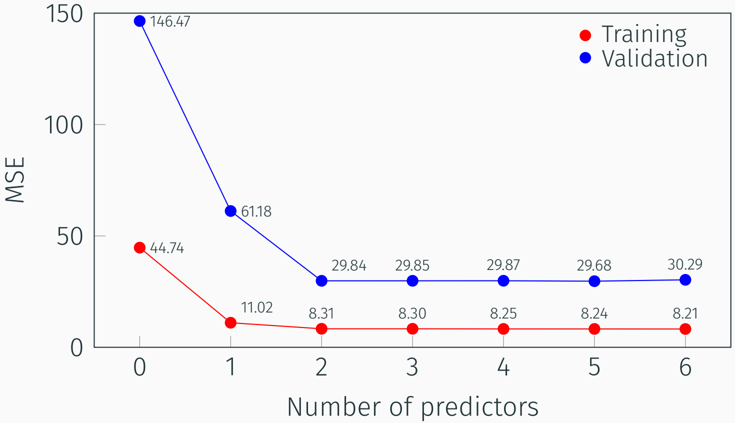
Variable selection: Forward stepwise selection



Variable selection: Forward stepwise selection



Variable selection: Forward stepwise selection



Variable selection: Forward stepwise selection

In[1]:

```
def fit_and_evaluate(train: pd.DataFrame, validation: pd.DataFrame,
                     predictors: List[str], target: str):
    model = LinearRegression()
    model.fit(train[predictors], train[target])

    train_predictions = model.predict(train[predictors])
    validation_predictions = model.predict(validation[predictors])

    return np.mean((train_predictions - train[target]) ** 2),
           np.mean((validation_predictions - validation[target]) ** 2)


predictors = ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year']
target = 'mpg'

train['intercept'] = 1
validation['intercept'] = 1
train_mse, validation_mse = fit_and_evaluate(train, validation, predictors=['intercept'], target=target)
print(f'[]: {validation_mse:.2f} ({train_mse:.2f})')

chosen_predictors = []

while len(chosen_predictors) < len(predictors):
    best_predictor = {'train_mse': None, 'validation_mse': float('inf'),
                      'predictor': None}

    for predictor in set(predictors) - set(chosen_predictors):
        train_mse, validation_mse = fit_and_evaluate(train, validation, predictors=chosen_predictors + [predictor], target=target)

        if validation_mse < best_predictor['validation_mse']:
            best_predictor = {'train_mse': train_mse, 'validation_mse': validation_mse, 'predictor': predictor}

    chosen_predictors.append(best_predictor['predictor'])

    print(f'{chosen_predictors}: {best_predictor["validation_mse"]:.2f} ({best_predictor["train_mse"]:.2f})')
```

Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start by fitting a model with no predictors. Iteratively add the predictor that yields the best model until all are included.

+ Positives

- Need to train fewer models.

- Drawbacks

- Not guaranteed to find the optimal solution.

Variable selection: Backward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start by fitting a model with all predictors. Iteratively remove the predictor that yields the best model until all you have none left.

+ Positives

- Need to train fewer models.

– Drawbacks

- Not guaranteed to find the optimal solution.

Variable selection: Backward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start by fitting a model with all predictors. Iteratively remove the predictor that yields the best model until all you have none left.

+ Positives

- Need to train fewer models.

- Drawbacks

- Not guaranteed to find the optimal solution.

Variable selection: Backward stepwise selection

Does forward and backward stepwise selection yield the same set of predictors?

Shrinkage

UNIVERSITETET
I OSLO

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \rightarrow 0$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space

Shrinkage: Outline

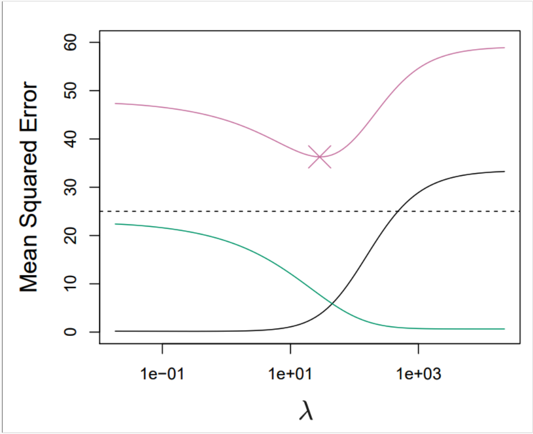
$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space

$$RSS = bias^2 + variance + irreducible\ error$$

Shrinkage: Outline



$$RSS = bias^2 + variance + irreducible\ error$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space
2. A little more bias $\implies$ A lot less variance

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space
2. A little more bias $\implies$ A lot less variance

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

subject to the constraint

$$\sum_{j=1}^{p} |\beta_j| \leq 1$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space
2. A little more bias $\implies$ A lot less variance

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

subject to the constraint

$$\sum_{j=1}^{p} |\beta_j| \leq 1$$

$$\beta_1 = 1, \beta_2 = 0, \beta_3 = 0, \ldots$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \rightarrow 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space
2. A little more bias $\implies$ A lot less variance

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

subject to the constraint

$$\sum_{j=1}^{p} |\beta_j| \leq 1$$

$$\beta_1 = 1, \beta_2 = 0, \beta_3 = 0, \ldots$$

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$$\beta_n \to 0$$

1. $\beta_1 = 0 \implies$
   One less degree of freedom in our function space
2. A little more bias $\implies$ A lot less variance
3. Parameters depend on eachother $\implies$
   Fewer degrees of freedom

Shrinkage: Ridge regression

$$loss_{RSS} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2$$

Shrinkage: Ridge regression

$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Shrinkage: Ridge regression

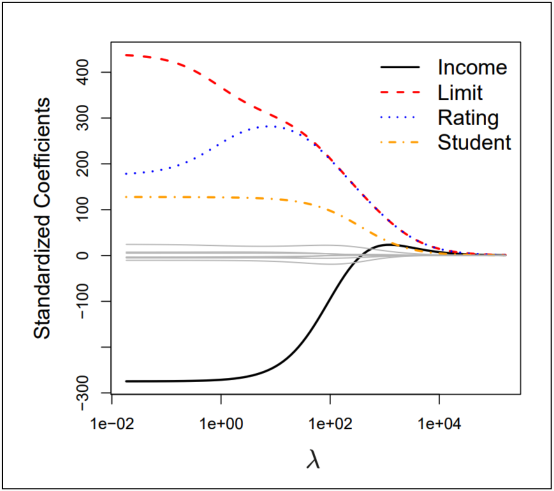
$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 \boxed{+\lambda \sum_{j=0}^{p} \beta_j^2}$$

$$\Downarrow$$

$\lambda \to 0 \Rightarrow \beta$ are free

$\lambda \to \infty \Rightarrow \beta \to 0$

Shrinkage: Ridge regression



Shrinkage: Ridge regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Shrinkage: Ridge regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

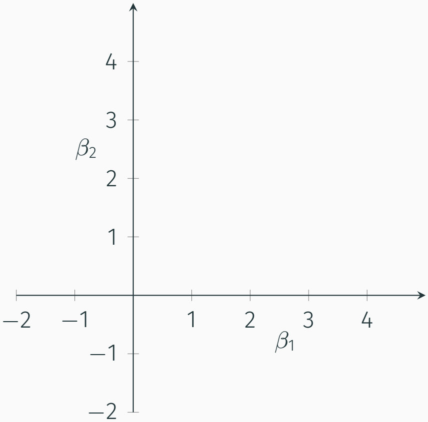
$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Shrinkage: Ridge regression

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

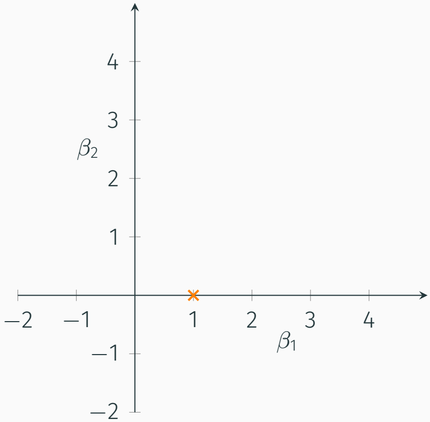
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

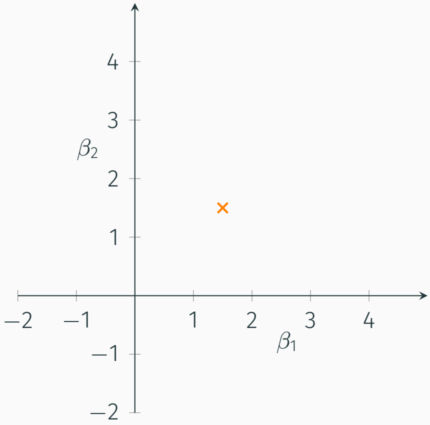
Shrinkage: Ridge regression



$$\hat{y} = 1x_1 + 0x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

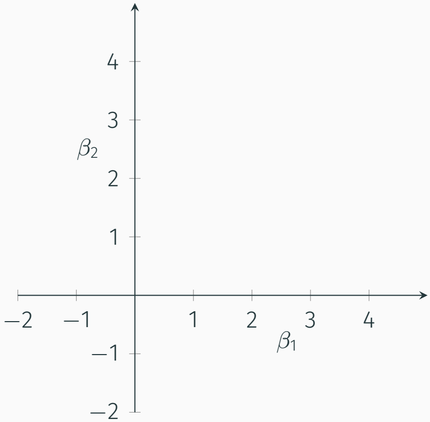
Shrinkage: Ridge regression



$$\hat{y} = 1.5x_1 + 1.5x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

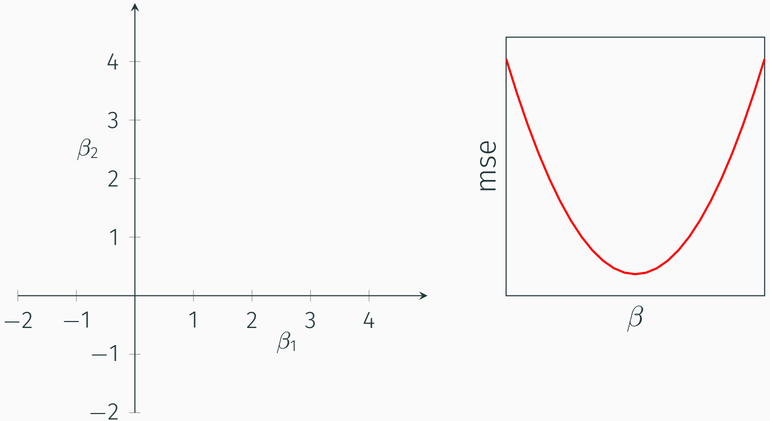
Shrinkage: Ridge regression



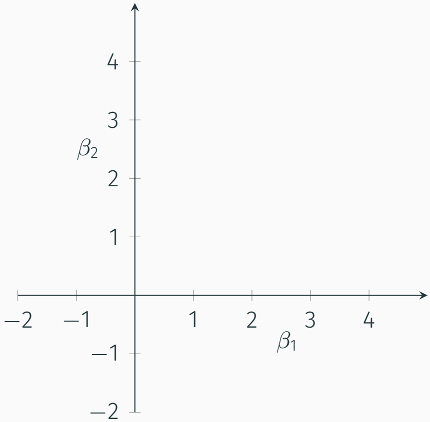
$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

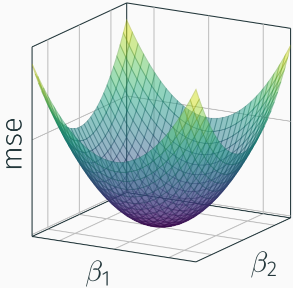
$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

Shrinkage: Ridge regression

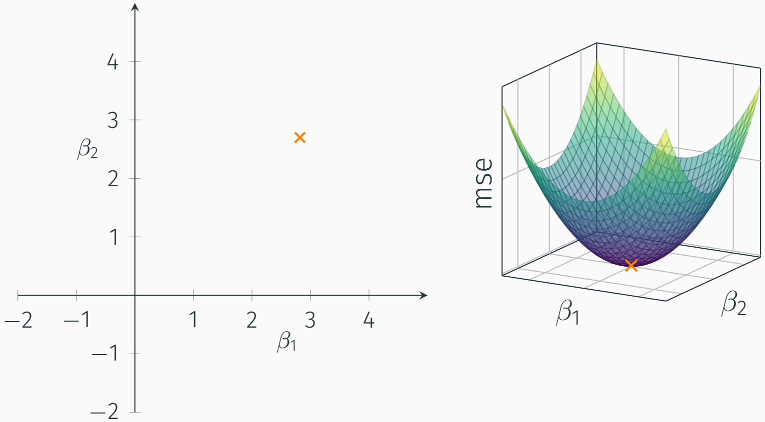


Shrinkage: Ridge regression

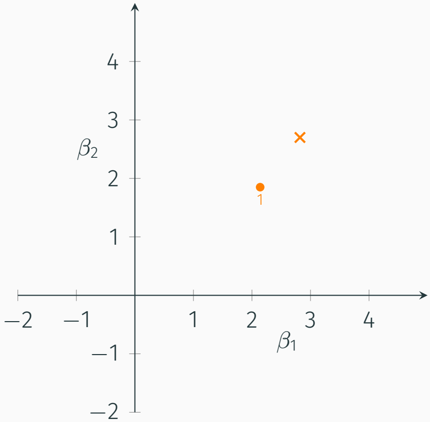


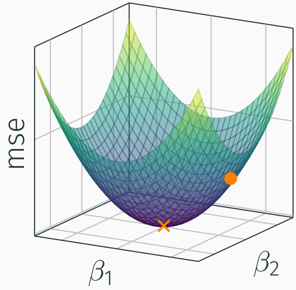


Shrinkage: Ridge regression

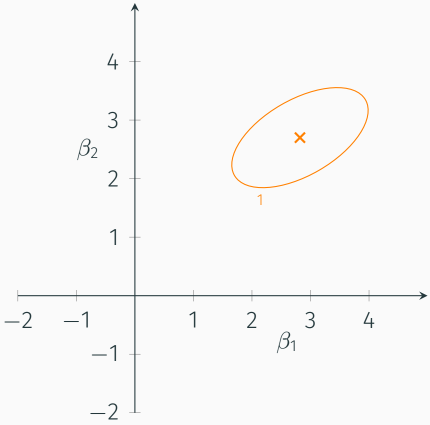


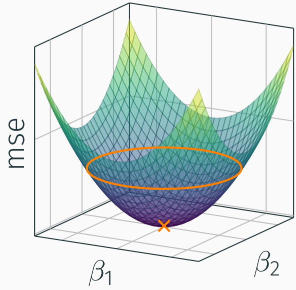
Shrinkage: Ridge regression



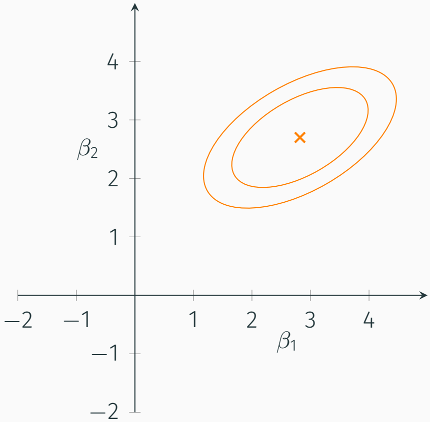


Shrinkage: Ridge regression





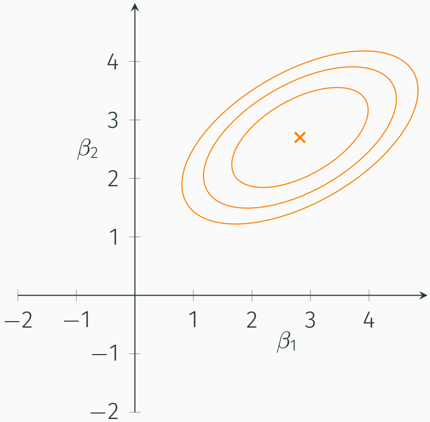
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

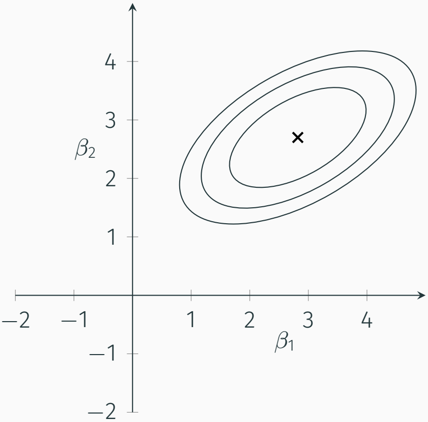
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

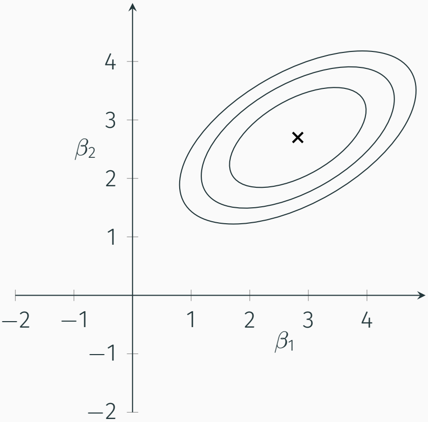
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

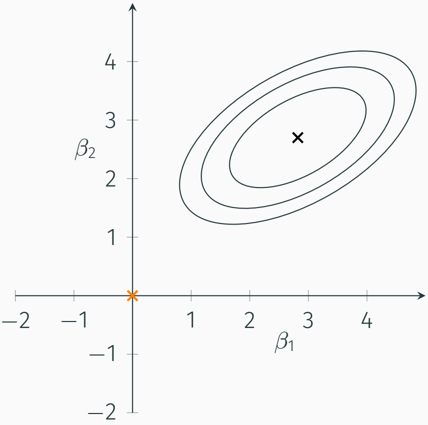
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

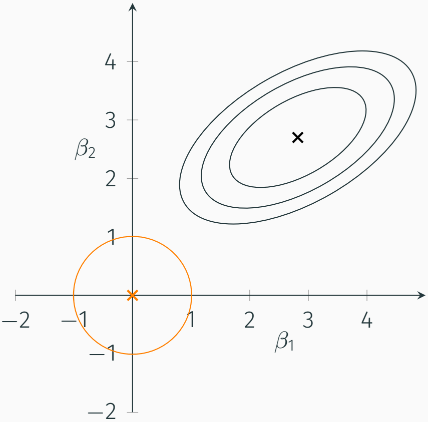
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

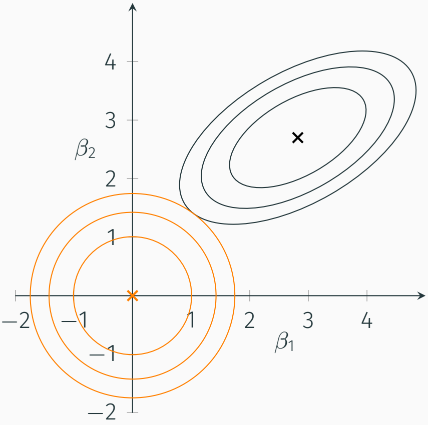
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

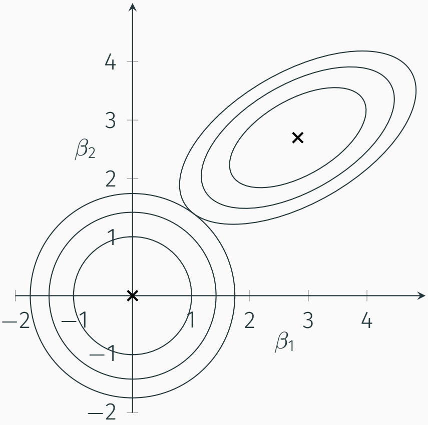
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

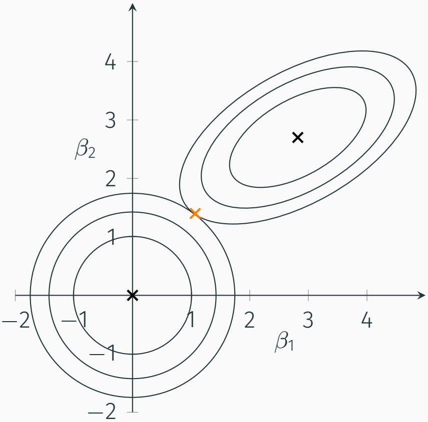
Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

Shrinkage: Ridge regression



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} \beta_j^2$$

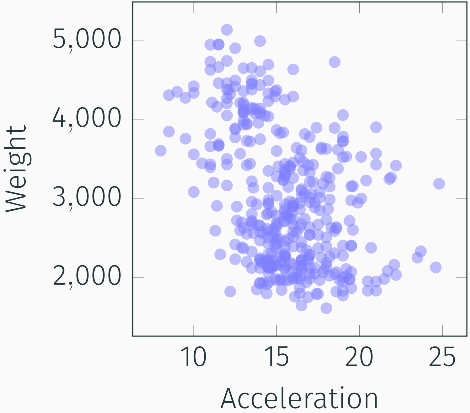
Shrinkage: Feature standardization

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
$$x_1 \in [0, 1], x_2 \in [0, 1000]$$

Shrinkage: Feature standardization

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
$$x_1 \in [0, 1], x_2 \in [0, 1000]$$
$$\Downarrow$$
$$\beta_1 \approx 1000 * \beta_2$$

Shrinkage: Feature standardization



Shrinkage: Feature standardization

z-score standardization

Shrinkage: Feature standardization

z-score standardization

$$x = \frac{x - \mu_x}{\sigma_x}$$

z-score standardization

$$x = \frac{x - \mu_x}{\sigma_x}$$

In[1]:
```python
for col in predictors:
    print(f'{col}: {np.mean(df[col]):.2f} ({np.std(df[col]):.2f})')

# z-score standardization
for col in predictors:
    df[col] = (df[col] - np.mean(df[col])) / np.std(df[col])

for col in predictors:
    print(f'{col} after: {np.mean(df[col]):.2f} ({np.std(df[col]):.2f})')
```

Out[1]:
```
cylinders: 5.47 (1.70)
displacement: 194.41 (104.51)
horsepower: 104.47 (38.44)
cylinders after: -0.00 (1.00)
displacement after: -0.00 (1.00)
horsepower after: -0.00 (1.00)
```

Shrinkage: Ridge regression

$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

http://localhost:8888/notebooks/notebooks/Live%20coding.ipynb

Shrinkage: LASSO

$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

$$loss_{lasso} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |\beta_j|$$

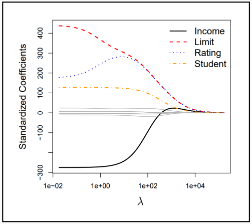
Shrinkage: LASSO

$$loss_{ridge} = \sum_{i=0}^{n}\left(y_i - \sum_{j=0}^{p}\beta_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{p}\beta_j^2$$

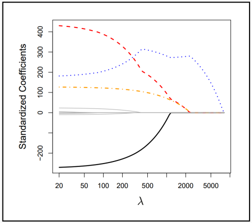
$$loss_{lasso} = \sum_{i=0}^{n}\left(y_i - \sum_{j=0}^{p}\beta_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{p}|\beta_j|$$

Shrinkage: LASSO

Ridge

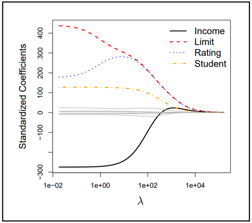


LASSO

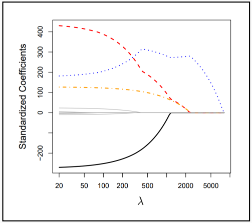


Shrinkage: LASSO

Ridge



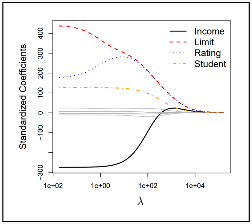
LASSO



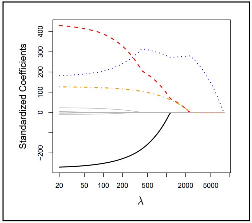
Predictor	Ridge	LASSO
Intercept	23.44	23.44
Weight	-5.59	-4.78
Year	2.75	2.00
Acceleration	0.19	0
Displacement	0.66	0

Shrinkage: LASSO

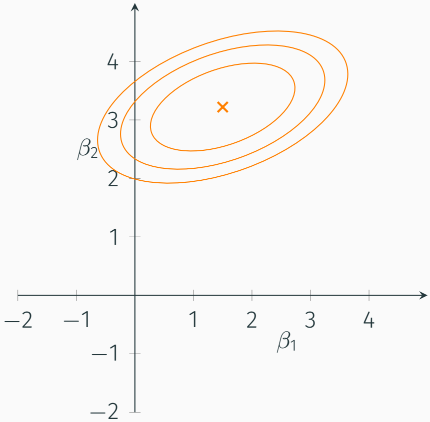
Ridge



LASSO



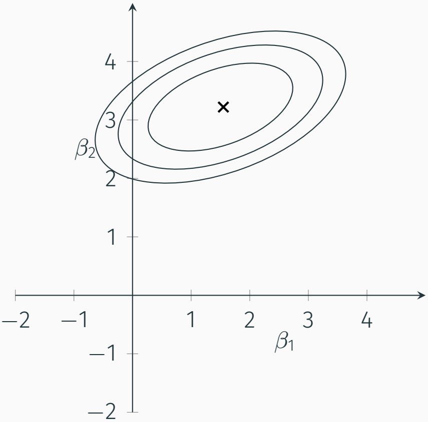
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

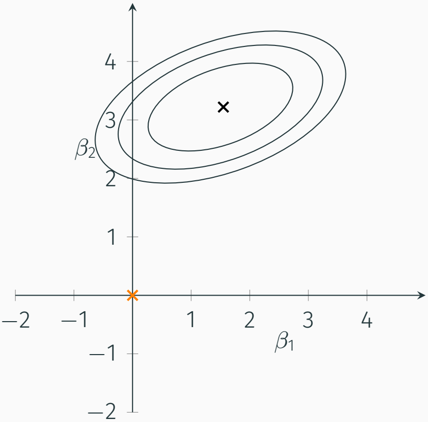
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

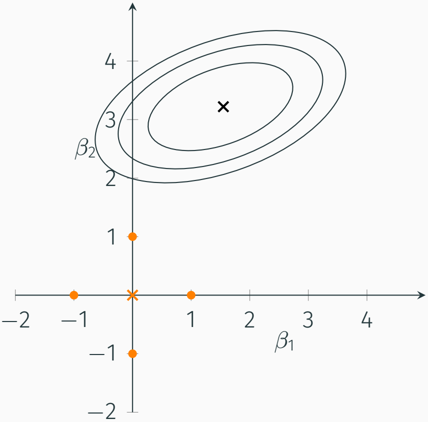
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

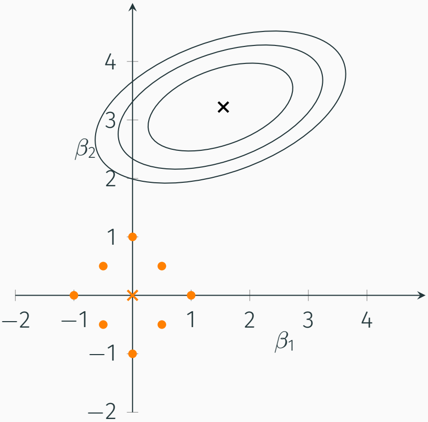
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

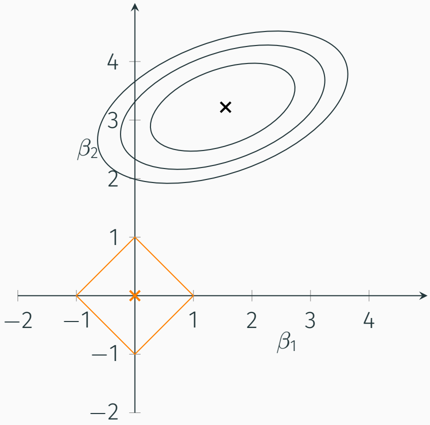
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

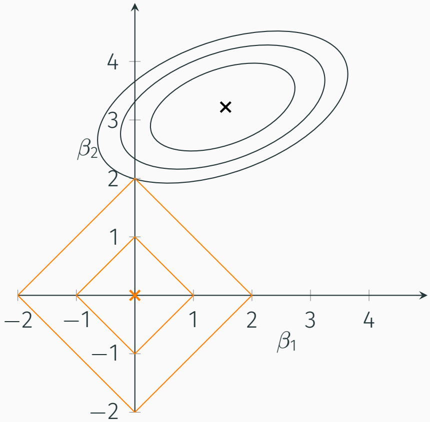
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

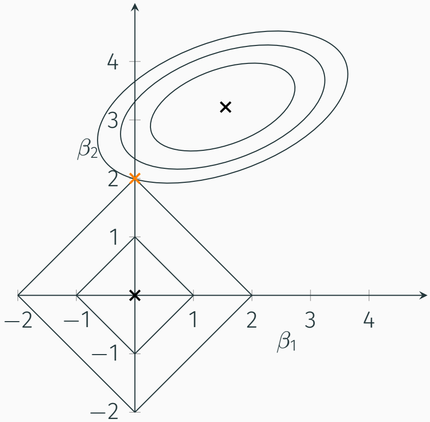
Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

Shrinkage: LASSO



$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

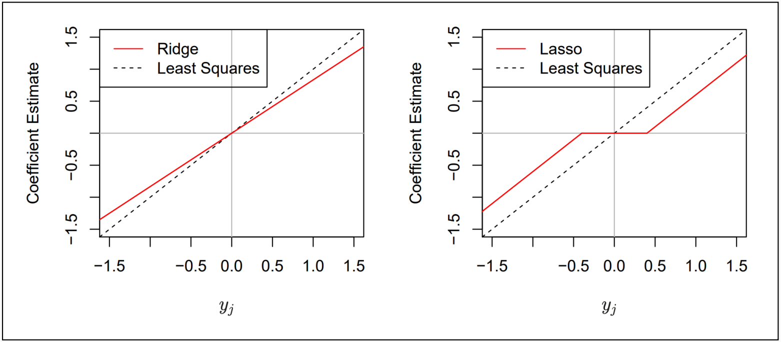
$$\ell = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=0}^{p} |\beta_j|$$

Shrinkage: LASSO

Predictor	Ridge	LASSO
Intercept	23.44	23.44
Weight	-5.59	-4.78
Year	2.75	2.00
Acceleration	0.19	0
Displacement	0.66	0

A coefficient of 0 does not mean the predictor has no association with the outcome! Why?

Shrinkage: LASSO



Shrinkage: Summary

$$loss_{mse} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2$$

Fits the **best** model to the data.

Shrinkage: Summary

$$loss_{mse} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2$$

Fits the **best** model to the data.

$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Fits the **best** model to the data while **shrinking** coefficients towards zero.

Shrinkage: Summary

$$loss_{mse} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2$$

Fits the **best** model to the data.

$$loss_{ridge} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Fits the **best** model to the data while **shrinking** coefficients towards zero.

$$loss_{lasso} = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |\beta_j|$$

Fits the **best** model to the data while **shrinking** coefficients towards zero such that some variables are effectively **removed**.

Assignment 3

UNIVERSITETET
I OSLO

Coding tips: Separation of concerns

In[1]:

```python
# Read and clean data
path = '/Users/esten/Downloads/Auto.csv'
df = pd.read_csv(path)

# Split data
train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]

# Define input and output variables
predictors = ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year']
target = 'mpg'

# Define necessary data structures for state
chosen_predictors = []
mses = []

while len(predictors) > 0:
    best_predictor = {'mse': float('inf'), 'predictor': None}

    for predictor in set(predictors) - set(chosen_predictors):
        potential_predictors = chosen_predictors + [predictor]

        # Fit and evaluate model
        model = LinearRegression()
        model.fit(train[potential_predictors], train[target])
        predictions = model.predict(validation[potential_predictors])
        test_mse = np.mean((validation[target] - predictions) ** 2)

        # Compare model with previous best
        if test_mse < best_predictor['mse']:
            best_predictor = {'mse': test_mse, 'predictor': predictor}

    # Update state
    chosen_predictors.append(best_predictor['predictor'])
    mses.append(best_predictor['mse'])
    predictors = [p for p in predictors if p != best_predictor['predictor']]
```

Coding tips: Separation of concerns

In[1]:

Setup

```
# Read and clean data
path = '/Users/esten/Downloads/Auto.csv'
df = pd.read_csv(path)

# Split data
train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]

# Define input and output variables
predictors = ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year']
target = 'mpg'

# Define necessary data structures for state
chosen_predictors = []
mses = []
```

Selection

```
while len(predictors) > 0:
    best_predictor = {'mse': float('inf'), 'predictor': None}

    for predictor in set(predictors) - set(chosen_predictors):
        potential_predictors = chosen_predictors + [predictor]
```

Modelling

```
        # Fit and evaluate model
        model = LinearRegression()
        model.fit(train[potential_predictors], train[target])
        predictions = model.predict(validation[potential_predictors])
        test_mse = np.mean((validation[target] - predictions) ** 2)
```

```
        # Compare model with previous best
        if test_mse < best_predictor['mse']:
            best_predictor = {'mse': test_mse, 'predictor': predictor}
```

Housekeeping

```
    # Update state
    chosen_predictors.append(best_predictor['predictor'])
    mses.append(best_predictor['mse'])
    predictors = [p for p in predictors if p != best_predictor['predictor']]
```

Coding tips: Separation of concerns

In[1]:

```python
# Read and clean data
path = '/Users/esten/Downloads/Auto.csv'
df = pd.read_csv(path)

# Split data
train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]

# Define input and output variables
predictors = ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year']
target = 'mpg'

# Define necessary data structures for state
chosen_predictors = []
mses = []

def fit_and_evaluate(model: LinearRegression, train: pd.DataFrame, validation: pd.DataFrame, variables: List[str], target: str):
    """ Fit a given model on a training dataset using a given set of variables and return MSE from a validation dataset. """
    model.fit(train[potential_predictors], train[target])
    predictions = model.predict(validation[potential_predictors])

    return np.mean((validation[target] - predictions) ** 2)

while len(predictors) > 0:
    best_predictor = 'mse': float('inf'), 'predictor': None

    for predictor in set(predictors) - set(chosen_predictors):
        potential_predictors = chosen_predictors + [predictor]
        test_mse = fit_and_evaluate(LinearRegression(), train, validation, variables=potential_predictors,target=target)

        # Compare model with previous best
        if test_mse < best_predictor['mse']:
            best_predictor = 'mse': test_mse, 'predictor': predictor

    # Update state
    chosen_predictors.append(best_predictor['predictor'])
    mses.append(best_predictor['mse'])
    predictors = [p for p in predictors if p != best_predictor['predictor']]
```

Modelling

Assignment 3

1. Prepare the Auto-dataset for analysis
2. Find the optimal set of predictors using backward selection
3. Fit multiple LASSO models