Regression

ECON 4810

Summarizing Relationship in Data

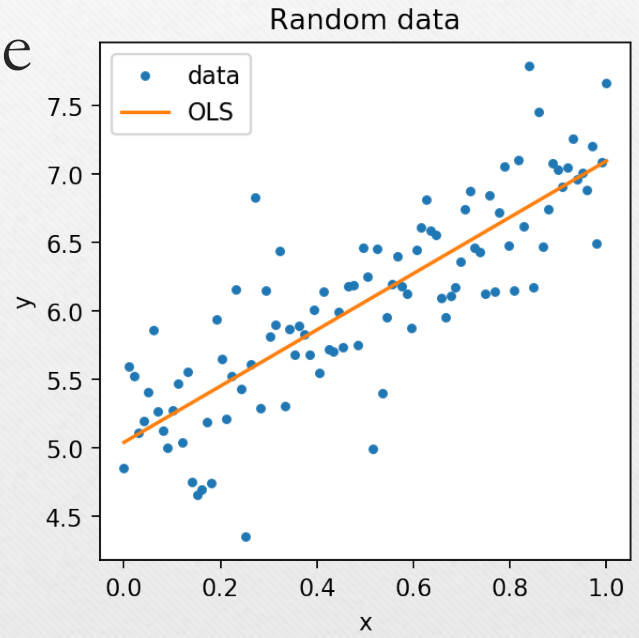
- We have some data X and y, and there appears to be a relationship between them
- How do we summarize this relationship?

Trendline

- Drawing a line through the "center" of the data
- What is a line?

$$y = \beta_0 + \beta_1 x$$

- How to find β_0 and β_1?



Ordinary Least Squares

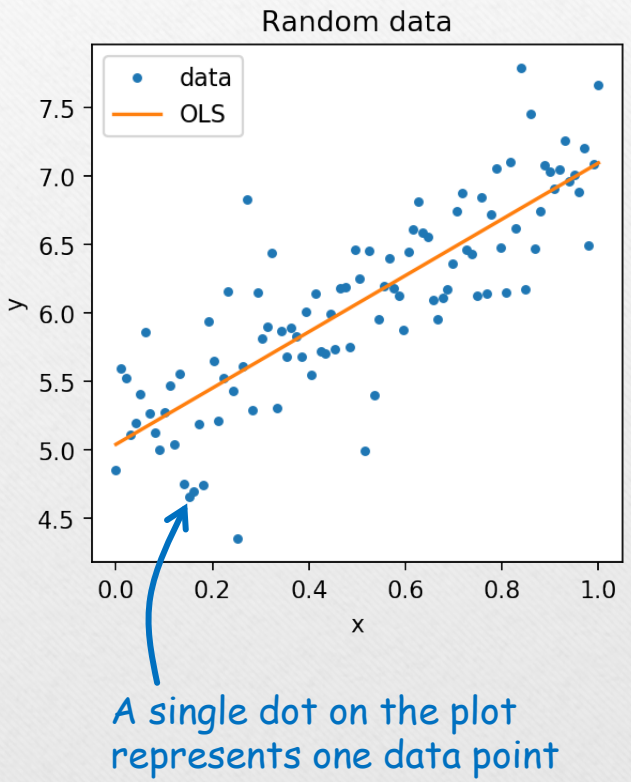
- Ordinary Least Squares (OLS) Regression is the most common technique

$$\min_{\beta_1,\beta_2} \sum_i [y_i - (\beta_0 + \beta_1 x_i)]^2$$

- Minimization problem is usually solved with calculus
- In the case of OLS, however, we can solve it as a linear algebra problem
- But first, some terminology

Terminology

- $\vec{y}$ is the output of our model. In economics it is typically called **dependent variable**, whereas in computer science it is called **target**
  - e.g. profit
- $X = [\vec{x}_1 \dots \vec{x}_k]$ is a matrix of input data**.** In economics this is typically called **independent variables**, whereas in computer science it is called **features**
  - e.g. $\vec{x}_1$ is input price of a raw material**,** $\vec{x}_2$ is wage of workers, etc.
- Each data point/observations/samples contains one value from $\hat{y}$ and a corresponding set of values from X, one from each $\vec{x}_i$



Terminology

- $\vec{\beta}$ is the parameters of the trendline. In economics these are called **coefficients,** whereas in computer science they are called **weights**

OLS Solution

- OLS Regression:

$$\min_{\beta_0 \cdots \beta_k} \sum_i \left[y_i - (\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_k x_{i,k}) \right]^2$$

- Solution to this problem can be represented by the following matrix equation:

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

- OLS is a **linear model**, in the sense that the estimated β's are linear combinations of y's:

$$\beta_j = c_{j,1} y_1 + \cdots + c_{j,n} y_n$$

Regression: Stat vs ML

- Statistics
  - Are the estimated β's **accurate**?
  - Are the estimated β's **statistically significant**? i.e. how do we know their true values are not zero?
- Machine Learning
  - How to make sure the model gives **accurate prediction**?

Regression in Statistics

Are the estimated β's accurate?

There are two possible sources of inaccuracy:

1. **The model is incorrect**

   Real model:

   $$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

   We use instead:

   $$y = \beta_0 + \beta_1 x_1$$

   or

   $$y = \beta_0 + \beta_1 x_1 + \beta_3 x_3$$

   In other words, we used the wrong line

How to Deal with β Inaccuracy

How to make sure your model is correct?

- The real model is almost always unknown
- What you do is try to include in your model variables that *might* be in true model
- This process is called **adding controls**
- The number of control variables that can be added is limited by how many observations you have. Reason:

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

Are the estimated β's accurate?

There are two possible sources of inaccuracy:

2. **The model has random component**

   e.g. $y = \beta_{20} + \beta_1 x_1 + \cdots + \beta x_k + \epsilon$

   and ϵ is random.

Are the estimated β's accurate?

- We need a model that can account for the regression's inaccuracy

  $y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_k x_{i,k} + \epsilon_i$

- Because x's are observables and β's are assumed to be fixed, the only source of inaccuracy comes from ϵ_i
- Usual assumption:
  - $\epsilon_i \sim N(0, \sigma^2)$
  - ϵ_i and ϵ_j are uncorrelated for $i \neq j$

Skip to Slide 20

Are β's statistically significant?

$$y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta x_{i,k} + \epsilon_i$$

- ϵ_i is normally distributed
- x's are observables, i.e. not random
- β—the true effects—are not directly observable, but they are assumed to be fixed
- This means y is normally distributed

Are β's statistically significant?

$$\hat{\beta} = (X^T X)^{-1} X^T \vec{y}$$

- y is normally distributed
- x's are observables
- $\widehat{\boldsymbol{\beta}}$ —the estimated effects—are normally distributed because $\hat{\beta}_j = c_{j,1} y_1 + \cdots + c_{j,n} y_n$
- **We can run the hypothesis tests we learnt last week on $\widehat{\boldsymbol{\beta}}$.** Specifically, we run t-tests.

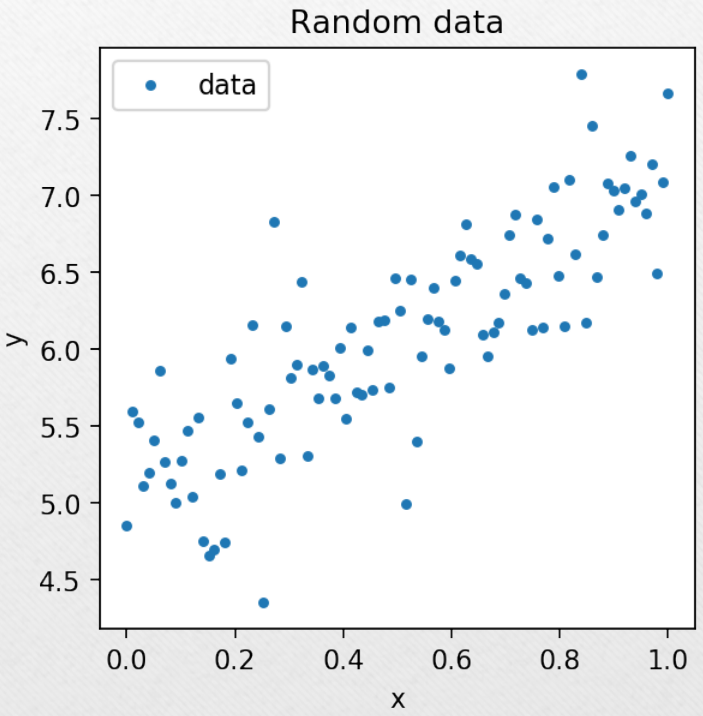
Best Linear Unbiased Estimator

- OLS is theoretically desirable because of the BLUE property:
  - **B**est — Smallest $Var[\hat{\beta}_j]$
  - **L**inear — $\hat{\beta}_j = c_{j,1}y_1 + \cdots + c_{j,n}y_n$
  - **U**nbiased — $E[\hat{\beta}_j] = \beta_j$
  - **E**stimator

Regression in Machine Learning

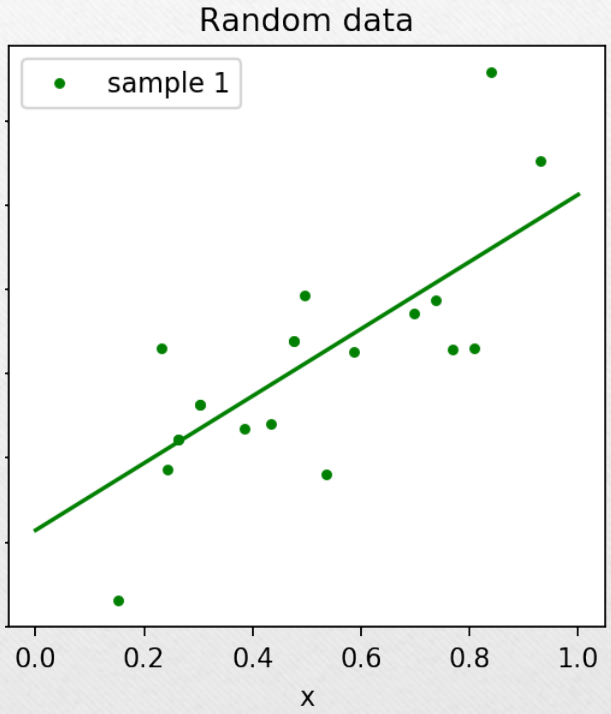
Regression

- OLS is attractive when the objective is coefficient estimation because of its BLUE property
- It is, however, usually not ideal when it comes to making prediction



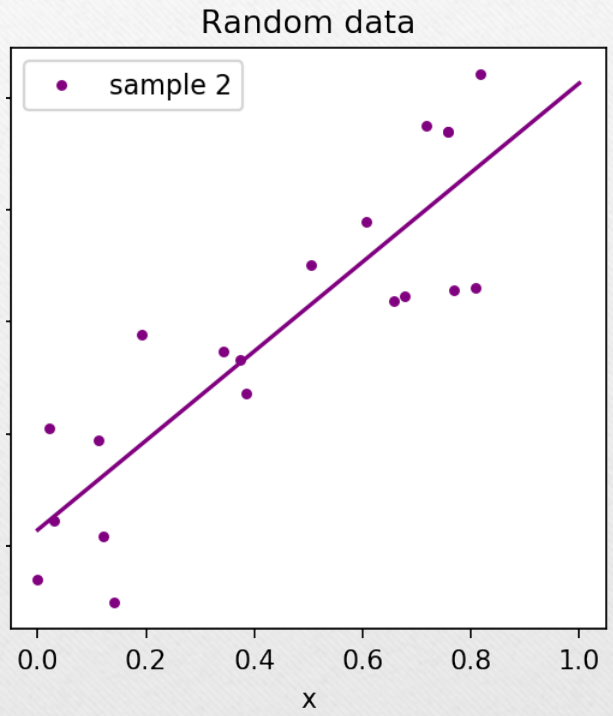
Regression

- OLS is attractive when the objective is coefficient estimation because of its BLUE property
- It is, however, usually not ideal when it comes to making prediction



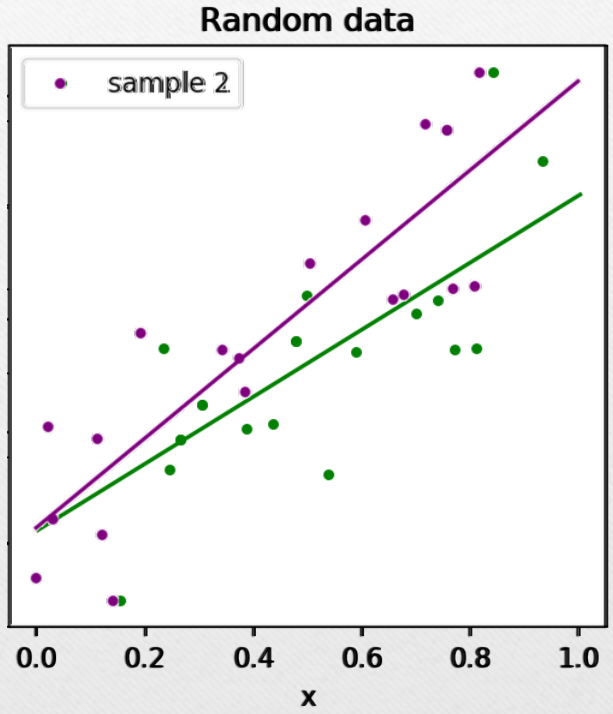
Regression

- OLS is attractive when the objective is coefficient estimation because of its BLUE property
- It is, however, usually not ideal when it comes to making prediction



Regression

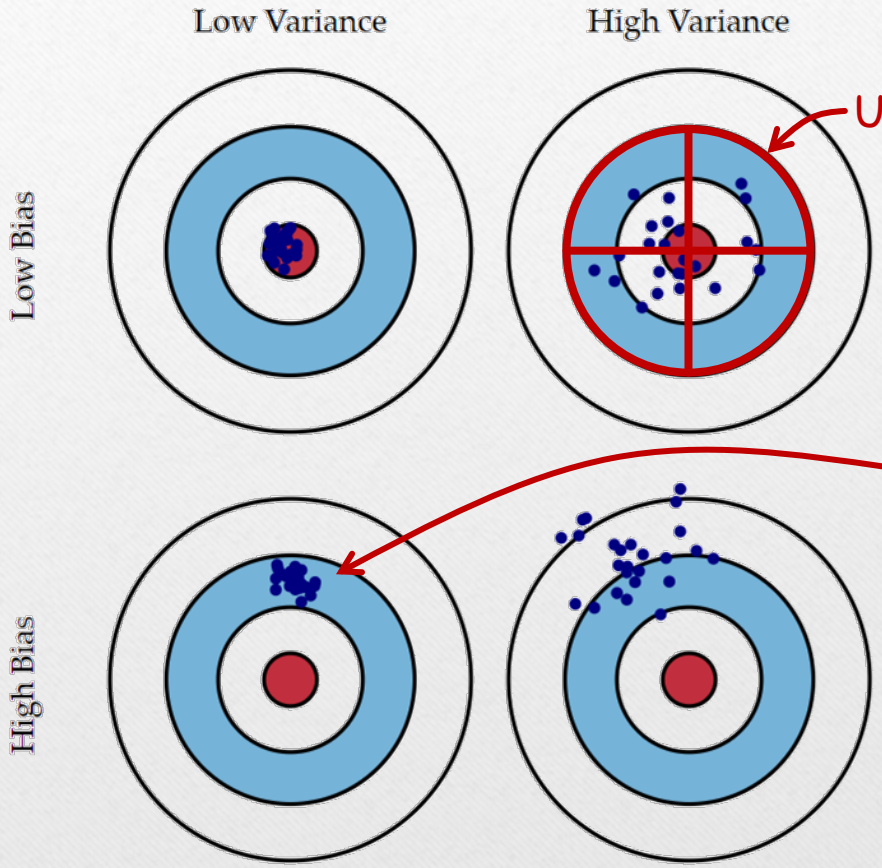
- OLS is attractive when the objective is coefficient estimation because of its BLUE property
- It is, however, usually not ideal when it comes to making prediction



Unbiased Estimator

- The problem with OLS is that we are 100% focused on minimizing bias
- This makes our estimates highly dependent on the particular sample of data we have. In other words, we **overfit**
- How do we deal with this problem?

Variance vs Bias



Unbiased: $E[\hat{\beta}_j] = \beta_j$

OLS has the lowest variance among **unbiased estimators**.

We can further lower variance if we are willing to consider **biased estimators**.

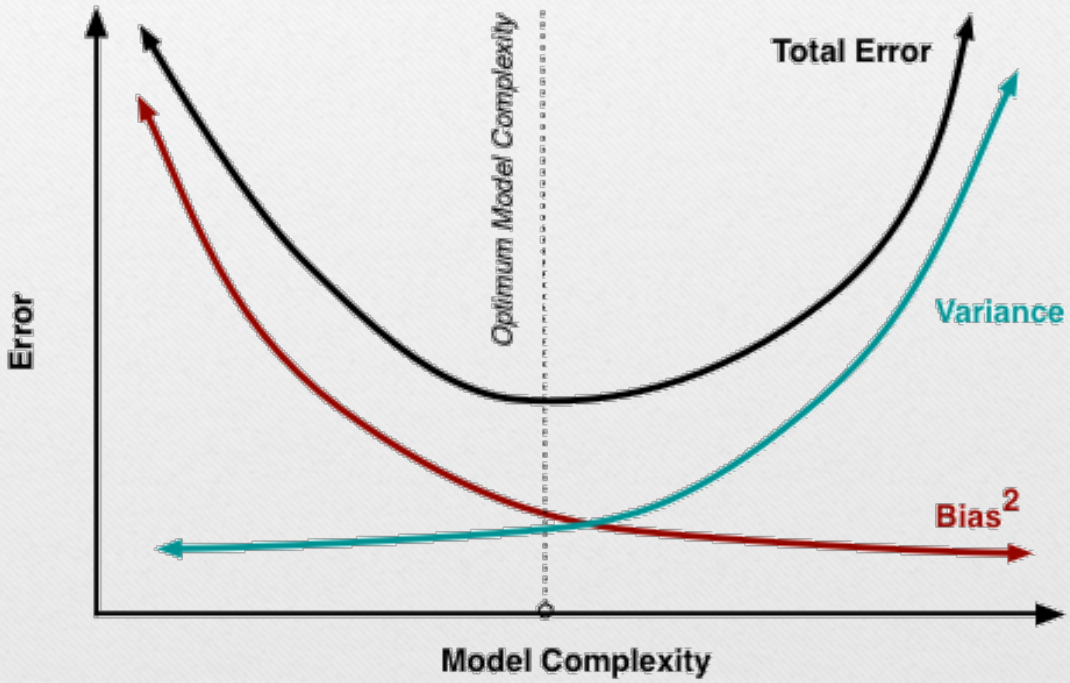
There is a tradeoff between bias and variance

Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Train for Low Variance!

Variance-Bias Tradeoff

- The trade-off between a model's ability to minimize variance and to minimize bias



Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization

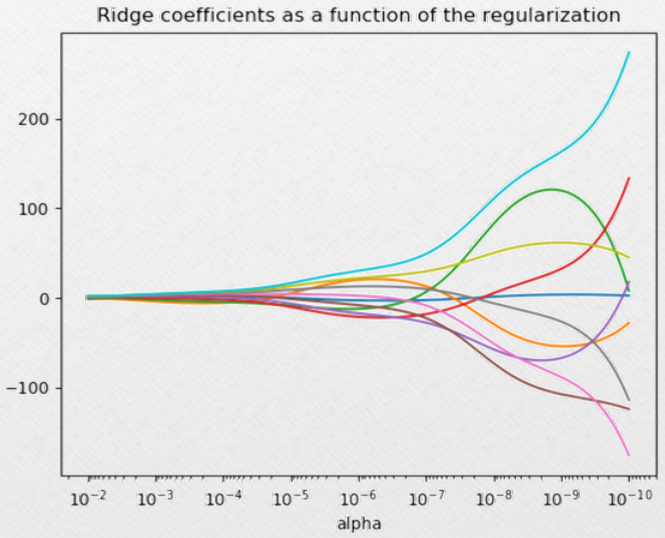
- **Regularization** penalizes large coefficients
- This reduce overfitting by making regression coefficients over different samples more similar to each other
  - In the most extreme case, all coefficients will be zero regardless of sample
- Regularized regression are biased, but they have smaller variance than OLS
- The two most common regularized regressions are **ridge** and **lasso**

Ridge Regression

- Ridge Regression, also called **Least Square with L2-regularization**, have the following objective:

$$\min_{\beta}\left\{\sum_{i}\left(y_i-\vec{\beta}\cdot\vec{x}_i\right)^2+\alpha\sum_{k}\beta_k^2\right\}$$

- Ridge regression pushes **all** coefficients **towards zero**
- Stronger regularization (higher α) leads to smaller coefficients



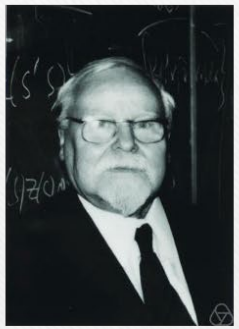
Ridge Regression

- Just like linear regression, ridge regression can be solved as a linear algebra problem:

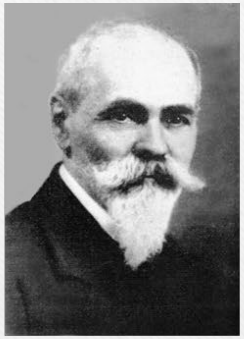
$$\hat{\beta} = (X^T X + \alpha I)^{-1} X^T \vec{y}$$

- What happens as α increases?

Why This Formulation?



Tikhonov, smoothing an ill-posed problem



Zaremba, model complexity minimization

Bayes: priors over parameters

Andrew Ng: need no maths, but it prevents overfitting!

Source: https://towardsdatascience.com/multitask-learning-teach-your-ai-more-to-make-it-better-dde116c2cd40

Lasso Regression

- Lasso Regression, also called **Least Square with L1-regularization**, have the following objective:

$$\min_{\beta} \left\{ \sum_{i} \left(y_i - \vec{\beta} \cdot \vec{x}_i \right)^2 + \alpha \sum_{k} |\beta_k| \right\}$$

- Lasso regression makes **some** coefficients **exactly zero**
- Stronger regularization (higher α) leads to more coefficients becoming zero
- **Lasso can help you select variables**

Python ML Models Workflow

1. Load data
2. Create instance of model class
3. Fit model
4. Check model performance
5. Make prediction

Example: Linear Regression

1. Load data

```
df = pandas.load_csv()
y = df["target"]
X = df[["var1","var2",…]]
```

2. Create instance of model class

```
model = LinearRegression()
```

3. Fit model

```
model.fit(X,y)
```

4. Check model performance

```
model.score(X,y)
```

5. Make prediction

```
model.predict(X)
```

Example: Lasso

1. Load data

```
df = pandas.load_csv()
y = df[“target”]
X = df[[“var1”,“var2”,…]]
```

2. Create instance of model class

```
model = Lasso(alpha=5)
```

3. Fit model

```
model.fit(X,y)
```

4. Check model performance

```
model.score(X,y)
```

5. Make prediction

```
model.predict(X)
```