Modeling Continuous Data

Week 10

Natalie Agus

Learning Objectives

- Write **cost** function of linear regression
- Implement **Gradient Descent algorithm** for **optimisation**
- **Train** linear regression model using gradient descent
- **Evaluate** linear regression model using r^2 and mean-squared-error
- **Plot cost function** over iteration time
- **Plot linear regression**
- **Transform** data for polynomial model

Linear Regression

- Independent vs dependent variable
- Try to model relationship between the two: e.g **linear** relationship

$$y = mx + c$$

- Goal: given arbitrary independent variable, **predict** dependent variable

Hypothesis

A proposed explanation for a phenomenon

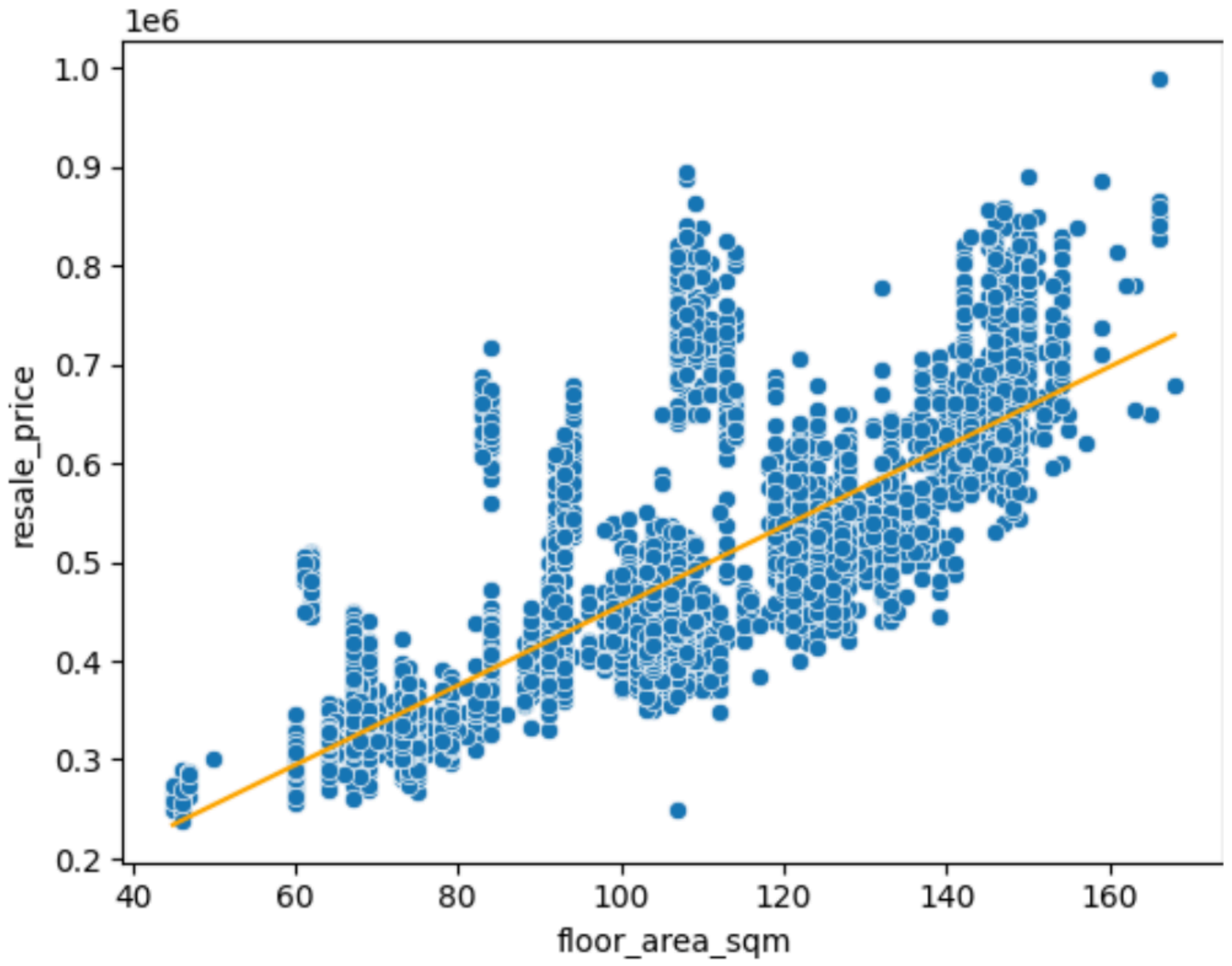
$$y = \beta_0 + \beta_1 x$$

- A hypothesis models the relationship between dependent and independent variables. Also known as **model**
- It contains **parameters**
- We need to **train** our model (hypothesis) to figure out the **value** of the parameters
- These parameters later are used for **prediction**

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Hypothesis

A proposed explanation for a phenomenon



$$\beta_0 = 52643$$

$$\beta_1 = 4030$$

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

Cost Function

- We initially only have a **hypothesis** with unknown **parameters**
- We need to **train** that model to figure out the optimum **parameters**
- How do we know if the training goes well or not?

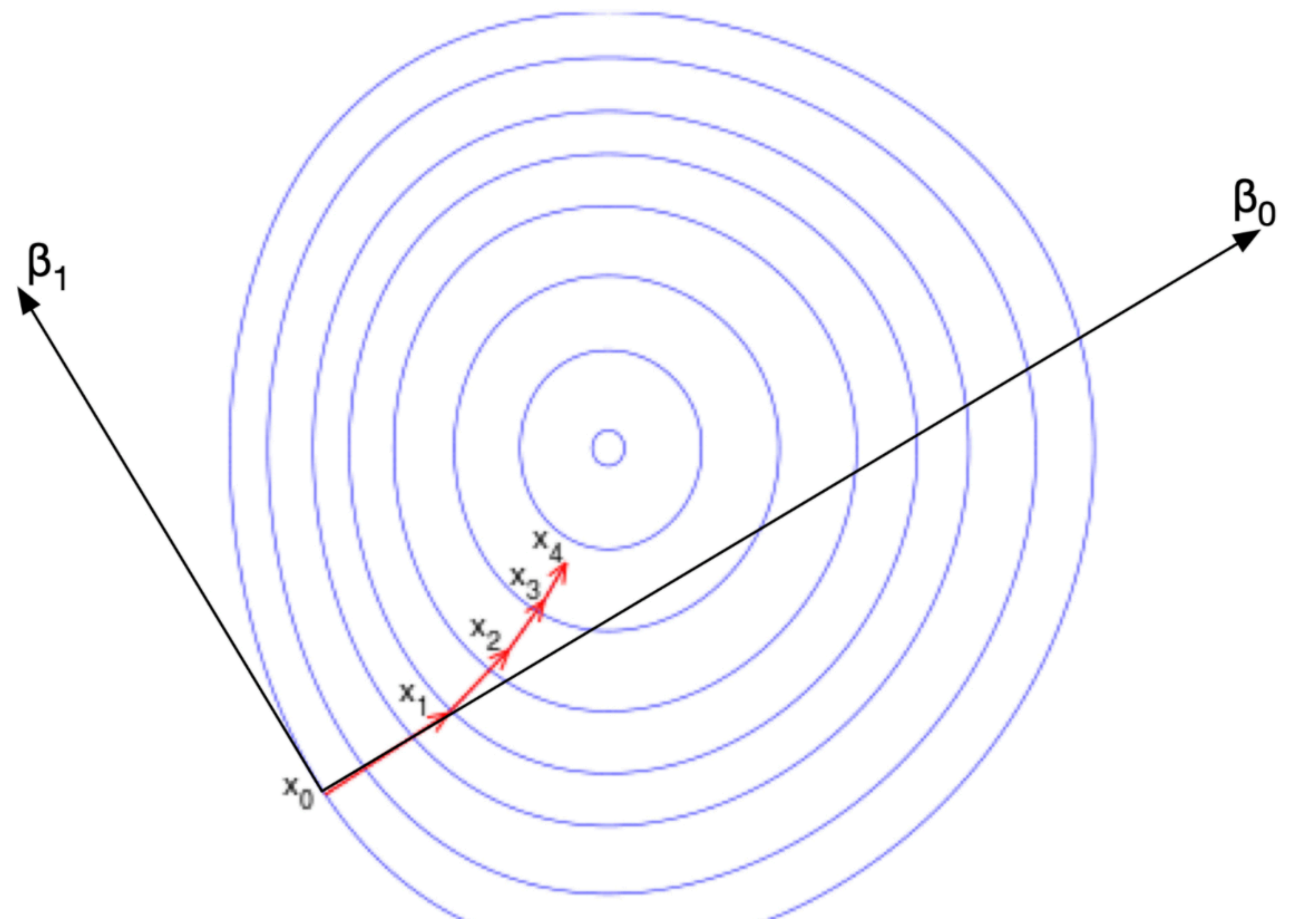
$$RSS = \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right)^2$$

  - Using **cost function**
  - It measures the **difference** between the model's predictions and the actual values
  - Many types of cost function: **Residual Sum Square**, **Mean Squared Error**, Mean Absolute Error, Huber Lost, KL Divergence, Hinge Loss, Poison Loss, etc

Gradient Descent

- Gradient decent is one of the ways to **train a model** and find **optimum** parameters.
    - Other ways: Stochastic GD, simulated annealing, Newton's method, Bayesian optimisation, genetic algorithm, etc
- Goal: **find parameters that minimise cost function**
- **Start:** with initial guess of all parameters
- Then, **compute** the rate of error (gradient of the cost function)
    - Next, **update** the parameters
    - **Repeat**

$$\underset{\hat{\beta}_0, \hat{\beta}_1}{\text{minimize}} \quad J(\hat{\beta}_0, \hat{\beta}_1)$$



$$\hat{\beta}_j = \hat{\beta}_j - \alpha \frac{\partial}{\partial \hat{\beta}_j} J(\hat{\beta}_0, \hat{\beta}_1)$$

Gradient Descent

$$\hat{\beta}_0 = \hat{\beta}_0 - \alpha \frac{\partial}{\partial \hat{\beta}_0} J$$

$$\hat{\beta}_1 = \hat{\beta}_1 - \alpha \frac{\partial}{\partial \hat{\beta}_1} J$$

$$J(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{2m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right)^2$$

$$\frac{\partial}{\partial \hat{\beta}_j} J(\hat{\beta}_0, \hat{\beta}_1) = \frac{\partial}{\partial \hat{\beta}_j} \frac{1}{2m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right)^2$$

Update function:

$$\hat{\beta}_0 = \hat{\beta}_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right)$$

$$\hat{\beta}_1 = \hat{\beta}_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right) x^i$$

Matrix Representation of Hypothesis

- Representing the **hypothesis** as matrices allow us to use matrix operations to clean the data & train the model, **easier** to implement as a program

- Use **NumPy** matrix operations for ease of computation

$$\mathbf{X} = \begin{bmatrix} 1 & x^1 \\ 1 & x^2 \\ \cdots & \cdots \\ 1 & x^m \end{bmatrix} \qquad \hat{\mathbf{b}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} \qquad \hat{\mathbf{y}} = \mathbf{X} \times \hat{\mathbf{b}}$$

Matrix Representation of Cost Function

- Representing the **cost function** as matrices allow us to use matrix operations to clean the data & train the model, **easier to implement**

- Use NumPy matrix operations for ease of computation

- Note: **MSE** is used

$$J(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{2m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right)^2$$

$$J(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{2m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i\right) \times \left(\hat{y}(x^i) - y^i\right)$$

$$J(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{2m} (\hat{\mathbf{y}} - \mathbf{y})^T \times (\hat{\mathbf{y}} - \mathbf{y})$$

Matrix Representation of Gradient Descent Update Function

$$\hat{\beta}_0 = \hat{\beta}_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i \right)$$

$$\hat{\beta}_1 = \hat{\beta}_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}(x^i) - y^i \right) x^i$$

$$\mathbf{X} = \begin{bmatrix} 1 & x^1 \\ 1 & x^2 \\ \dots & \dots \\ 1 & x^m \end{bmatrix}$$

$$\hat{\mathbf{b}} = \hat{\mathbf{b}} - \alpha \frac{1}{m} \mathbf{X}^T \times (\hat{\mathbf{y}} - \mathbf{y})$$

$$\mathbf{X}^T = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x^1 & x^2 & \dots & x^m \end{bmatrix}$$

$$\hat{\mathbf{b}} = \hat{\mathbf{b}} - \alpha \frac{1}{m} \mathbf{X}^T \times (\mathbf{X} \times \hat{\mathbf{b}} - \mathbf{y})$$

Metrics

- Training data set is used to train (build) the hypothesis (model)
- Test data set is used to **evaluate** the model: we compute metrics to determine whether the model is good/bad

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y^i - \hat{y}^i)^2$$

$$r^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{tot} = \Sigma_{i=1}^{n}(y_i - \overline{y})^2$$

$$SS_{res} = \Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\overline{y} = \frac{1}{n}\Sigma_{i=1}^{n} y_i$$

Multiple Linear Regression

- Model the relationship between a **dependent** variable and **multiple** (no longer single) **independent** variable

- SLR: $\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$

- MLR: $\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_n x_n$

Multiple Linear Regression: Matrix Expression

$$\hat{y}(x^1) = \hat{\beta}_0 + \hat{\beta}_1 x_1^1 + \hat{\beta}_2 x_2^1 + \ldots + \hat{\beta}_n x_n^1$$

$$\hat{y}(x^2) = \hat{\beta}_0 + \hat{\beta}_1 x_1^2 + \hat{\beta}_2 x_2^2 + \ldots + \hat{\beta}_n x_n^2$$

$$\ldots$$

$$\hat{y}(x^m) = \hat{\beta}_0 + \hat{\beta}_1 x_1^m + \hat{\beta}_2 x_2^m + \ldots + \hat{\beta}_n x_n^m$$

$$\hat{\mathbf{b}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \ldots \\ \hat{\beta}_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \ldots & x_n^1 \\ 1 & x_1^2 & \ldots & x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_1^m & \ldots & x_n^m \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}$$

$$\boxed{\hat{\mathbf{y}} = \mathbf{X} \times \hat{\mathbf{b}}}$$

Same as SLR

MLR Cost Function: Matrix Expression

$$\boxed{J(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{2m}(\hat{\mathbf{y}} - \mathbf{y})^T \times (\hat{\mathbf{y}} - \mathbf{y})}$$

Same as SLR

MLR Gradient Descent Update Function: Matrix Expression

$$\boxed{\hat{\mathbf{b}} = \hat{\mathbf{b}} - \alpha \frac{1}{m} \mathbf{X}^T \times (\mathbf{X} \times \hat{\mathbf{b}} - \mathbf{y})}$$

Same as SLR

About R^2

- Sometimes it can be **misleading**: R^2 always increases with more features as it measures the **proportion** of variance explained by the variance and adding features generally **reduces** the RSS, making R^2 higher
    - R^2 is good for simple models, easier to interpret
- A high R^2 does **not** necessarily mean a better model—it might just be overfitting.
- Adjusted R2 penalise the inclusion of irrelevant features

$$\text{Adjusted R}^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-k-1}\right)$$

About R^2 vs Adjusted R^2

- Some nuances
- With R^2 = N%, you can say "N% of the **variation** in the application is **explained** by our predictors"
- With Adjusted R^2 = N, you must say "After **accounting** for the number of predictors and **penalizing** any unnecessary ones, the model explains around N% variance in the application"

Polynomial Model

Transform features into polynomial terms

- Plot first see the **relationship** between independent and dependent variable

- Transform the independent variable as necessary, e.g: **quadratic hypothesis**

- **Note:** you **don't always need** the lower order terms if you are confident that only the higher order has predictive power. *Otherwise, it won't "hurt" either, as the params can be near zero if the data doesn't have meaningful relationship with lower order terms*

$$x_1 = x$$

$$x_2 = x^2$$

$$\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2$$

$$\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} & (x^2)^{(1)} \\ 1 & x^{(2)} & (x^2)^{(2)} \\ \dots & \dots & \\ 1 & x^{(m)} & (x^2)^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times 3}$$

Polynomial Model

Considerations

- **Overfitting**: higher degree polynomials can overfit data, capturing **noise** instead of underlying pattern
- **Feature scaling:** polynomial terms can result in large differences in magnitude, requiring normalization or standardization **before** polynomial transformation
- Application: modeling non-linear behaviors like trajectories, forces, market trends

Learning Objectives

- Write **cost** function of linear regression
- Implement **Gradient Descent algorithm** for **optimisation**
- **Train** linear regression model using gradient descent
- **Evaluate** linear regression model using r^2 and mean-squared-error
- **Plot cost function** over iteration time
- **Plot linear regression**
- **Transform** data for polynomial model