COMP5310: Principles of Data Science

W9: Linear Regression & Logistic Regression

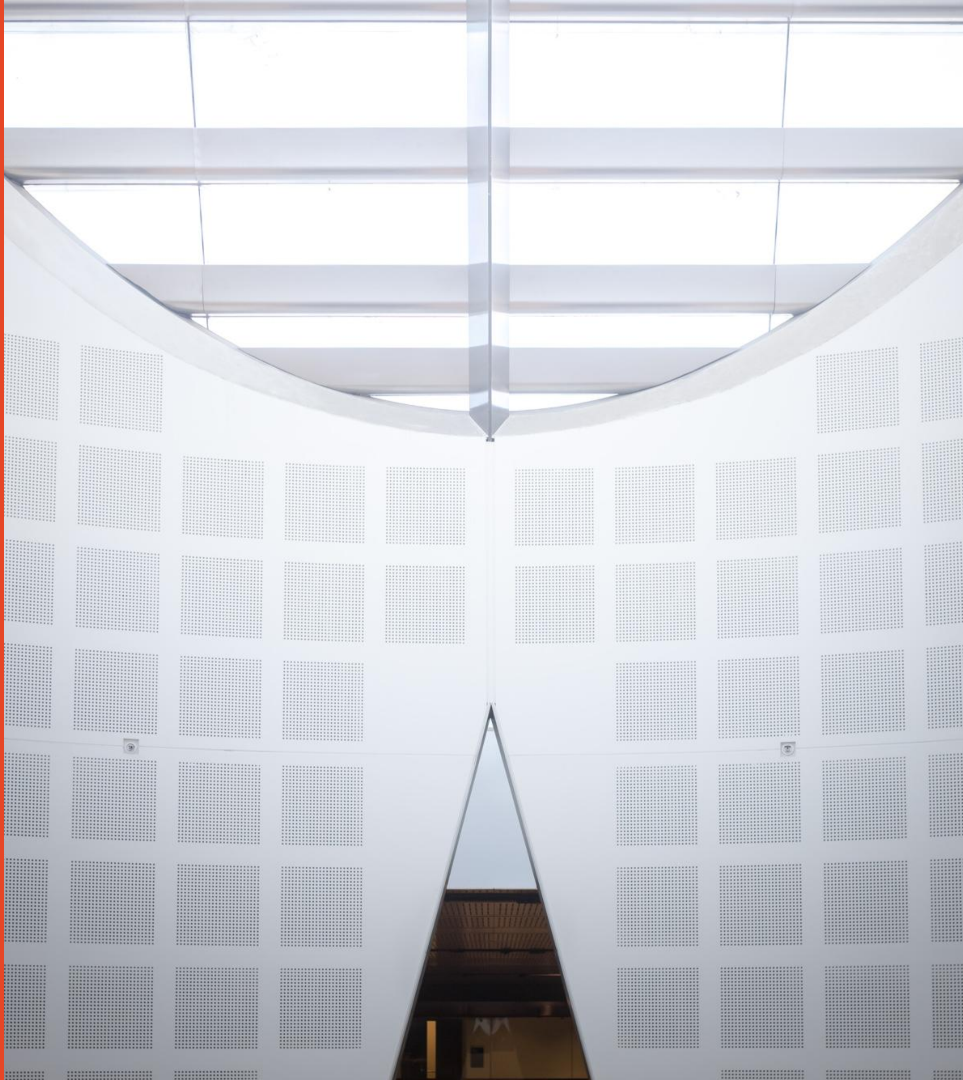
Presented by

Maryam Khanian

School of Computer Science

Based on slides by previous lecturers of this unit of study

THE UNIVERSITY OF SYDNEY



Last week: Clustering and Dimensionality Reduction

Objective

Learn techniques for unsupervised learning, with tools in Python.

Lecture

- Clustering algorithms
- Evaluating clustering
- Choosing k
- Principal Component Analysis

Readings

- Intro to Data Mining, Ch. 7 https://www-users.cse.umn.edu/~kumar001/dmbook/ch7_clustering.pdf
- Data Science from Scratch, Ch. 20

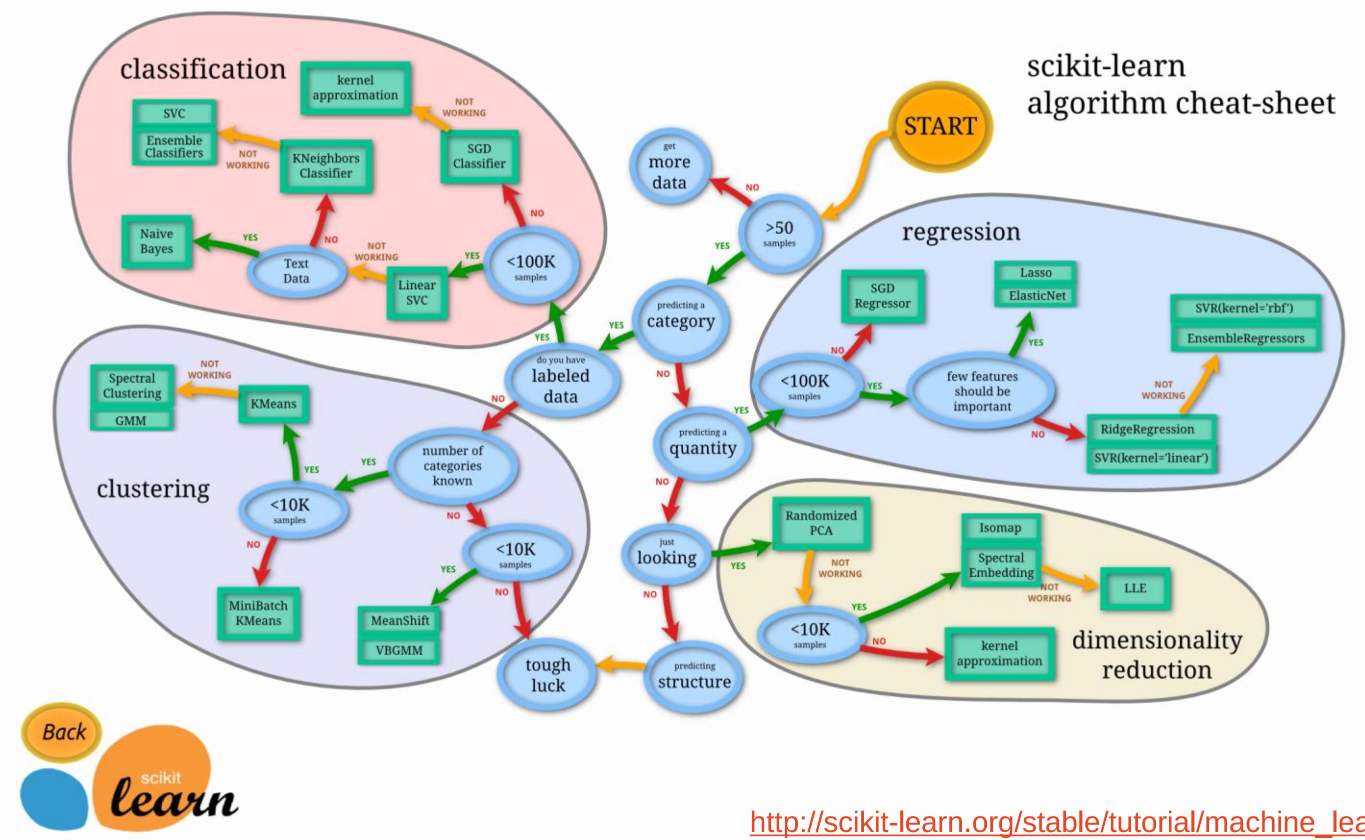
Exercises

- sklearn: clustering and PCA

Supervised Learning:

- We'll now focus on supervised machine learning techniques
  - **Simple linear regression**
  - **Multiple linear regression**
  - **Logistic regression**
  - Decision tree
  - Naïve Bayes

Machine learning map from scikit-learn



http://scikit-learn.org/stable/tutorial/machine_learning_map/

Modelling

Refund	Status	Income	Cheat
Yes	Single	125K	No
No	Married	100K	No
No	Single	70K	No
Yes	Married	120K	No
No	Divorced	95K	Yes
No	Single	85K	Yes
Yes	Single	90K	Yes

Refund	Status	Income	Cheat
No	Married	80K	?

Carbon level(%)	Purity (%)
0.99	90.01
1.02	86.05
1.15	91.43
1.29	93.74
1.46	96.73
1.36	94.45
0.87	87.59

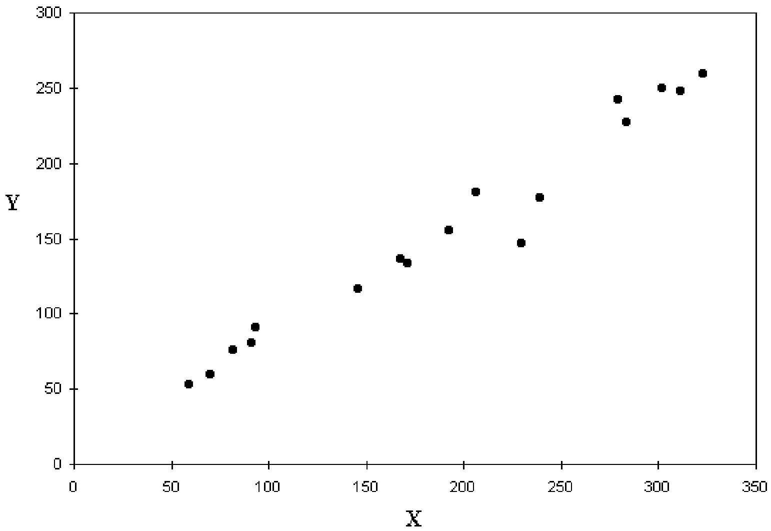
Carbon level(%)	Purity (%)
1.32	?

- A **model** is a specification of a mathematical relationship between different variables
  - Mapping from features (X) to a numeric value or categorical label (Y)

$$Y=f(X)$$

Simple Linear Regression

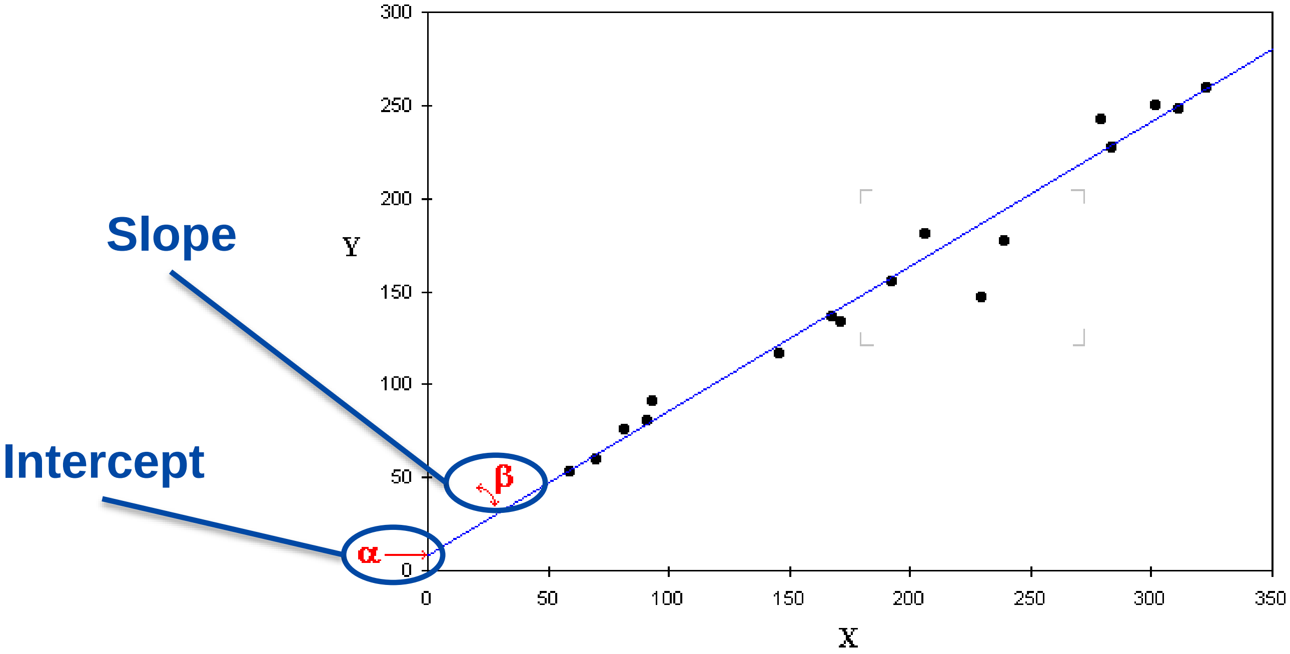
Simple linear regression



- Method for finding the **line of best fit** between the dependent variable Y and the independent variable X
- **Simple**: only one independent variable

$$Y = \alpha + \beta X + \varepsilon$$

What’s the line that explains X↦Y?



Slope

Intercept

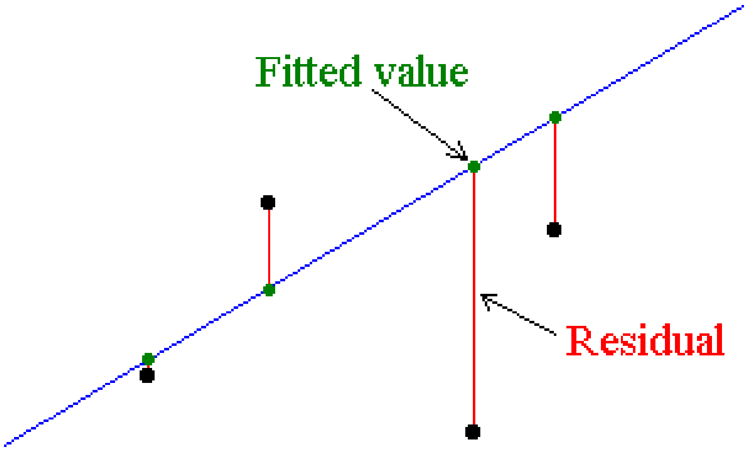
$$Y = \alpha + \beta X$$

Fitting SLR: learn α and β

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

- **α**: Intercept (where the line crosses the y axis)
- **β**: Slope (direction and steepness of the line)
- **ε_i**: Error (error term describing variation of data)
- **Y_i** is the dependent variable (response).
- **X_i** is the independent variable (predictor).

Fitting SLR: Minimize sum of squared errors



- **Error/residual**: difference between the observed value and predicted value $(y_{actual} - y_{predicted})$
- Sum of squared errors:

$$\varepsilon = SSE = \underbrace{\sum}_{\text{Sum}} \underbrace{(y_i - \hat{y}_i)}_{\text{Error}}{}^{\underbrace{2}_{\text{Square}}}$$

http://home.ku.edu.tr/yihlamur/public_html/Bitirme%20Projesi%20-%20Ger%C3%A7ek%20Data%20D%C3%BCzenlenmi%C5%9F/regression/minitab%20regresion%20hakk%C4%B1nda%20g%C3%BCzel%20bilgiler.htm

Ordinary Least Squares (OLS)

- Our goal is to find the optimal value of $\hat{\alpha}$ and $\hat{\beta}$ such that $\frac{1}{n}\sum_{i=1}^{n}\left(\left(\hat{\alpha}+\hat{\beta}x_i\right)-y_i\right)^2$ is the minimum

- Using calculus

$$\hat{\beta}=\frac{\sum\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sum\left(x_i-\bar{x}\right)^2}=\frac{\mathrm{cov}(x,y)}{\mathrm{Var}(x)}=\frac{r(x,y)*sd(y)}{sd(x)}$$

$$\hat{\alpha}=\bar{y}-\hat{\beta}\bar{x}$$

sd is the standard deviation and ***r*** is the correlation

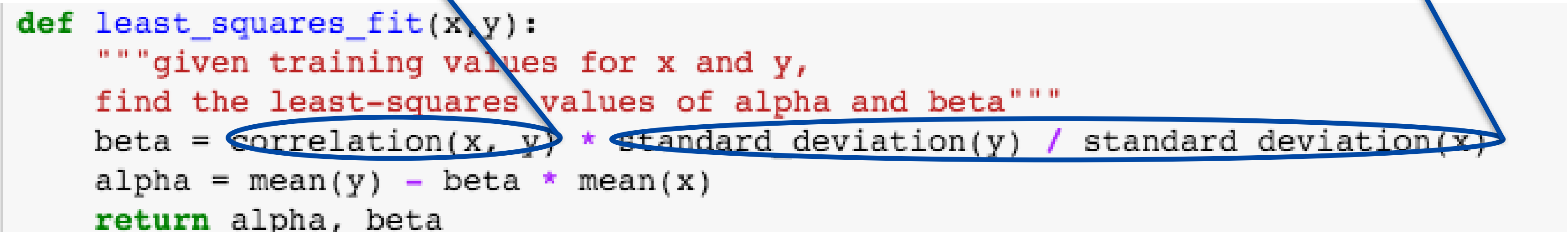
$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$ is the average of x_i values

$\bar{y}=\frac{1}{n}\sum_{i=1}^{n}y_i$ is the average of y_i values

Fitting SLR: Least squares

Slope of standardized data points with mean 0 and stdev 1

Adjust slope for variation in Y and X



```
def least_squares_fit(x,y):
    """given training values for x and y,
    find the least-squares values of alpha and beta"""
    beta = correlation(x, y) * standard_deviation(y) / standard_deviation(x)
    alpha = mean(y) - beta * mean(x)
    return alpha, beta
```

https://en.wikipedia.org/wiki/Simple_linear_regression#Fitting_the_regression_line

Coefficient of determination (R^2)

- **R^2**: ratio of **explained variation in y** to **total variation in y**

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{SSE}{SST}$$

- Ranges from 0 to 1, with higher values indicating better fit for the **linear** model
  - R^2 is square of the correlation coefficient between x and y
- Conveys goodness of fit but not precision

Understanding Model Fit – SSE, SSR, SST

- SSE - Sum of Squared Errors
  - Defined as the **difference between model's predicted value and the observed value**.
  - Smaller the value, the better the model's predictive power and vice versa.

$$SSE = \sum_i (y_i - \hat{y}_i)^2$$

- SSR - Sum of Squared Regression
  - Defined as the **difference between model's predicted value and the mean value of the depended variable**.
  - The smaller the value, the better the model fits the data and vice versa.

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

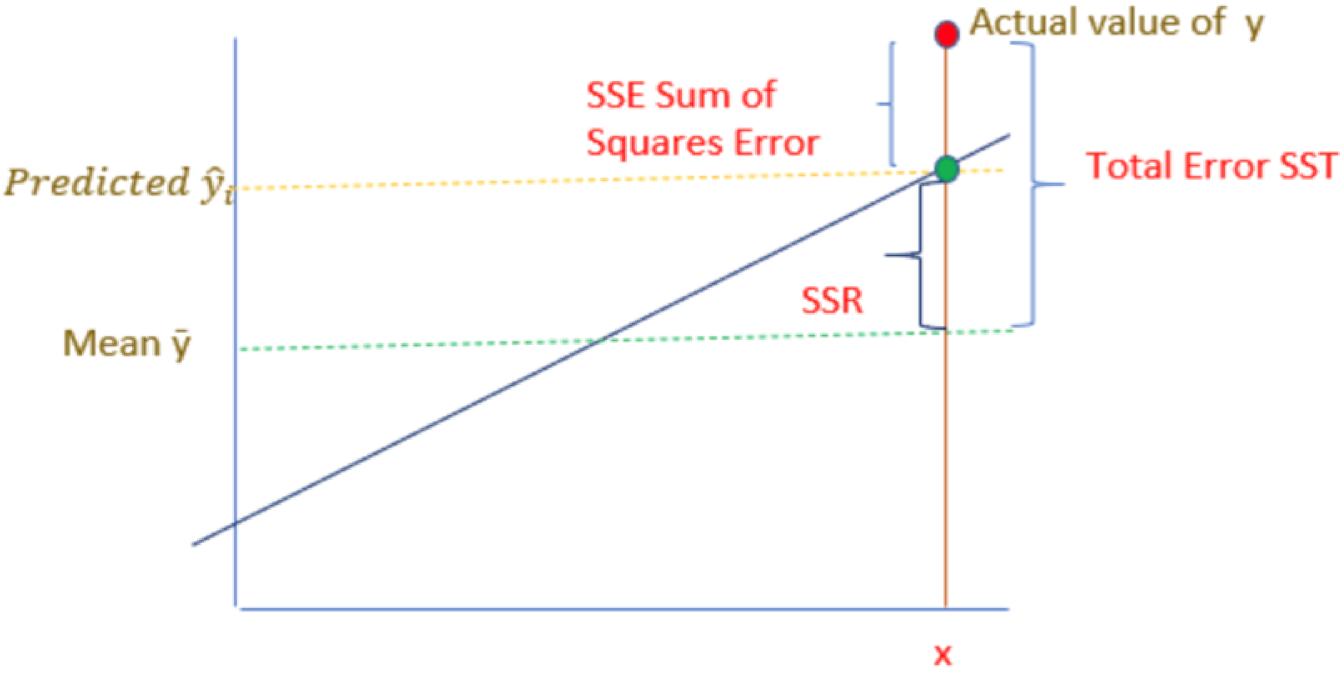
- SST - Sum of Squared Total
  - Defined as SSR + SSE. **It defines the overall dispersion of the points of the *dependent variable* from the mean value of the dependent variable.**
  - Smaller the value, the closer the values of the dependent variable are to the mean of the dependent variable.

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$SSE = \sum_{i} (y_i - \hat{y}_i)^2$$

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$



Source: https://indhumathychelliah.com/2020/10/18/linear-regression-in-python/

Multiple Linear Regression

THE UNIVERSITY OF SYDNEY

Multiple linear regression

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_d X_d$$

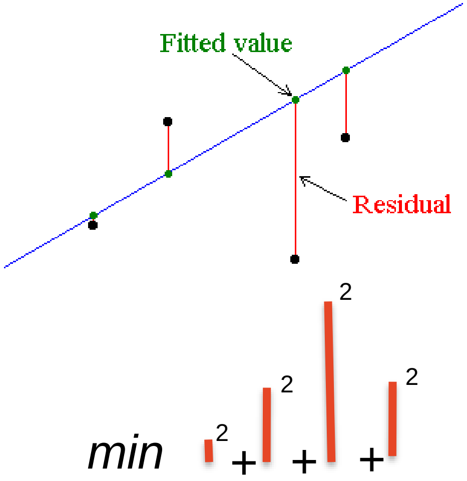
- Explain the relationship between:
  - **two or more** independent variables and
  - one dependent variable

How to learn θ

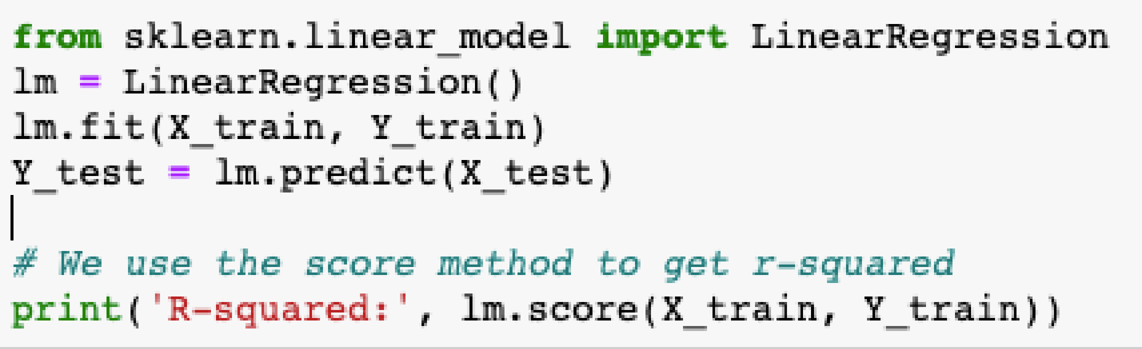
$$Y = h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j = \boldsymbol{\theta}^T \boldsymbol{x}$$

Assume $x_0 = 1$

- Cost function: $J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2$
- Fit model by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
  - There is also a closed-form solution



Machine learning in scikit-learn



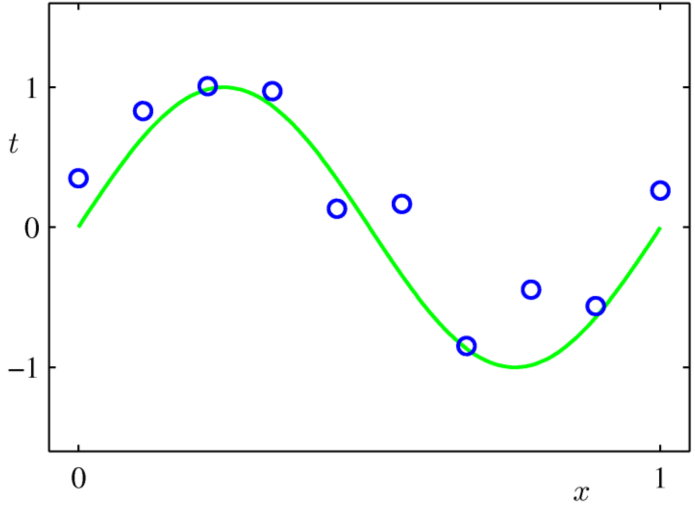
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, Y_train)
Y_test = lm.predict(X_test)

# We use the score method to get r-squared
print('R-squared:', lm.score(X_train, Y_train))
```

- **Fit(X, y)**: fits a model to the training data X, y
  - **X**: feature vectors
  - **y**: labels
- **Predict(T)**: predict labels for new data T

Extending linear regression to more complex models

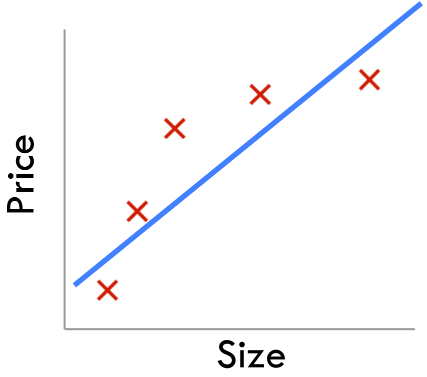
- Polynomial transformation
  - $Y = h_{\boldsymbol{\theta}}(\mathrm{x}) = \theta_0 + \theta_1 \mathrm{x}^1 + \theta_2 \mathrm{x}^2 + \ldots + \theta_d \mathrm{x}^d = \sum_{j=0}^{d} \theta_j \mathrm{x}^j$

- This allows the use of linear regression techniques to fit non-linear datasets.



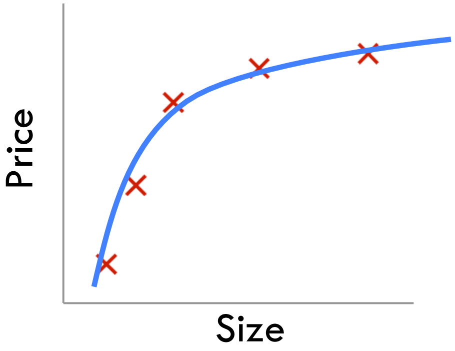
Quality of fit

- Overfitting:
  - The learned model may fit the training set very well
  - but fails to generalize to new examples



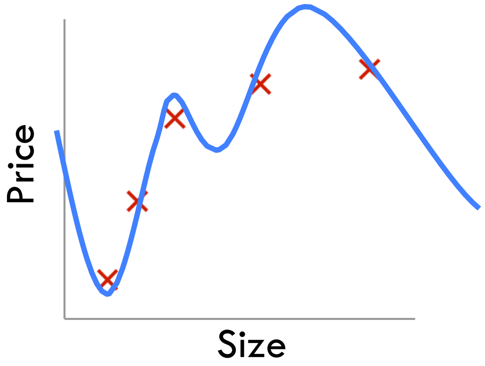
$$\theta_0 + \theta_1 x$$

Under-fitting (high bias)



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Correct fit



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfitting (high variance)

Prevent overfitting with regularization

- A method for automatically controlling the complexity of the learned model
- **Regularization** aims to penalize for large values of coefficients (θ_j)
  - Can incorporate into the cost function
  - The more weight we give to the regularization term, the more we discourage large coefficients
- Can also address **overfitting** by eliminating features (either manually or via model selection)
  - Large feature spaces introduce problems with **overfitting**

Regularization

- Linear regression cost function
  - $J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(\mathrm{h}_{\boldsymbol{\theta}}\left(\mathrm{x}^{(\mathrm{i})}\right) - \mathrm{y}^{(\mathrm{i})}\right)^2}_{\text{Model fit to data}} + \underbrace{\lambda\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$
- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0
- There is also a closed-form solution for minimizing $J(\boldsymbol{\theta})$

Gradient Descent

How to learn θ

- For many other models/cost functions, there is no closed-form solution of θ

- A geneal search procedure for learning θ
  - Choose an initial value for $\boldsymbol{\theta}$
  - Iteratively choose a new value for $\boldsymbol{\theta}$ to reduce $J(\boldsymbol{\theta})$ (e.g., through gradient descent)
    - Until we reach a minimum

Intuition behind cost function

For insight on J(), let's assume $x \in \mathbb{R}$, so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

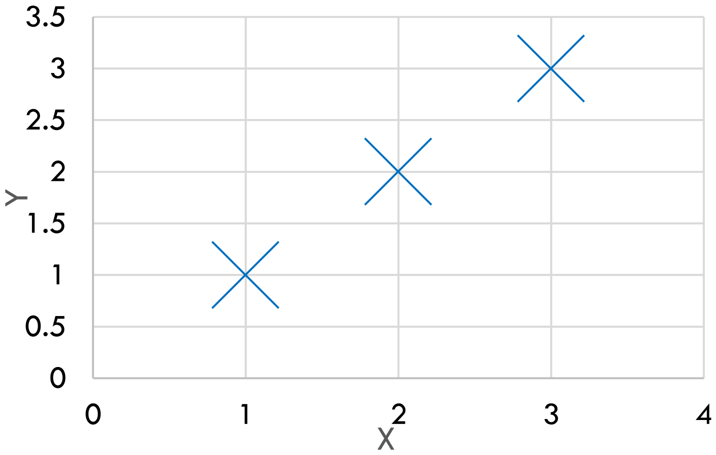
$$Y = h_{\boldsymbol{\theta}}(X) = \underline{\theta_0} + \theta_1 X$$

Let's say we have the following data points:

X	Y
1	1
2	2
3	3

$\underline{\theta_0} = 0$ and $\theta_1 = 1$

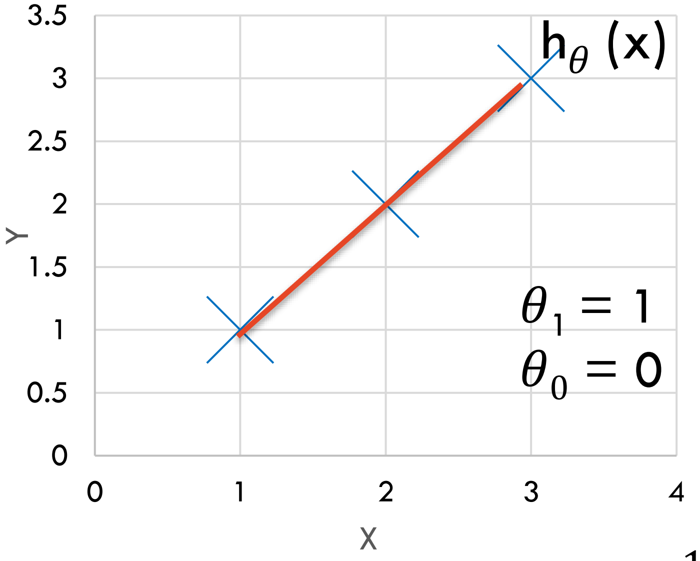
$Y = h_{\theta}(X) = X$

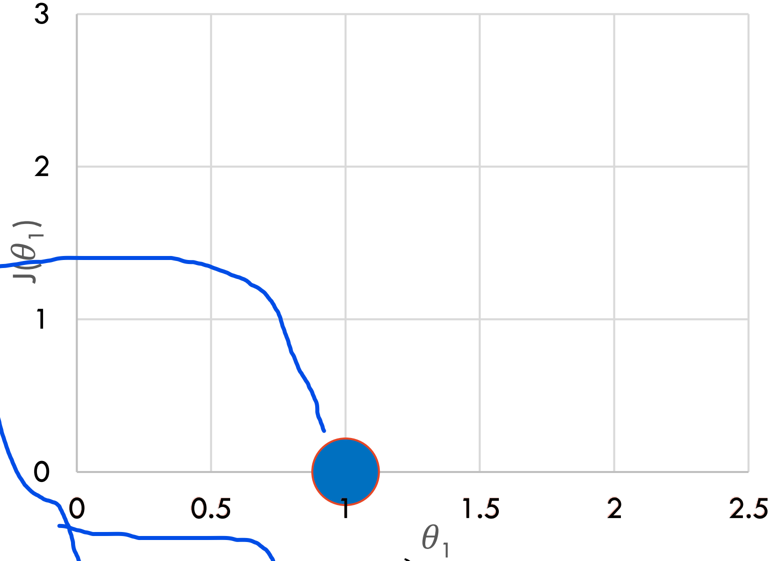


Intuition behind cost function

For insight on J(), let's assume $x \in \mathbb{R}$, so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

$$Y = h_{\boldsymbol{\theta}}(X) = \theta_0 + \theta_1 X$$



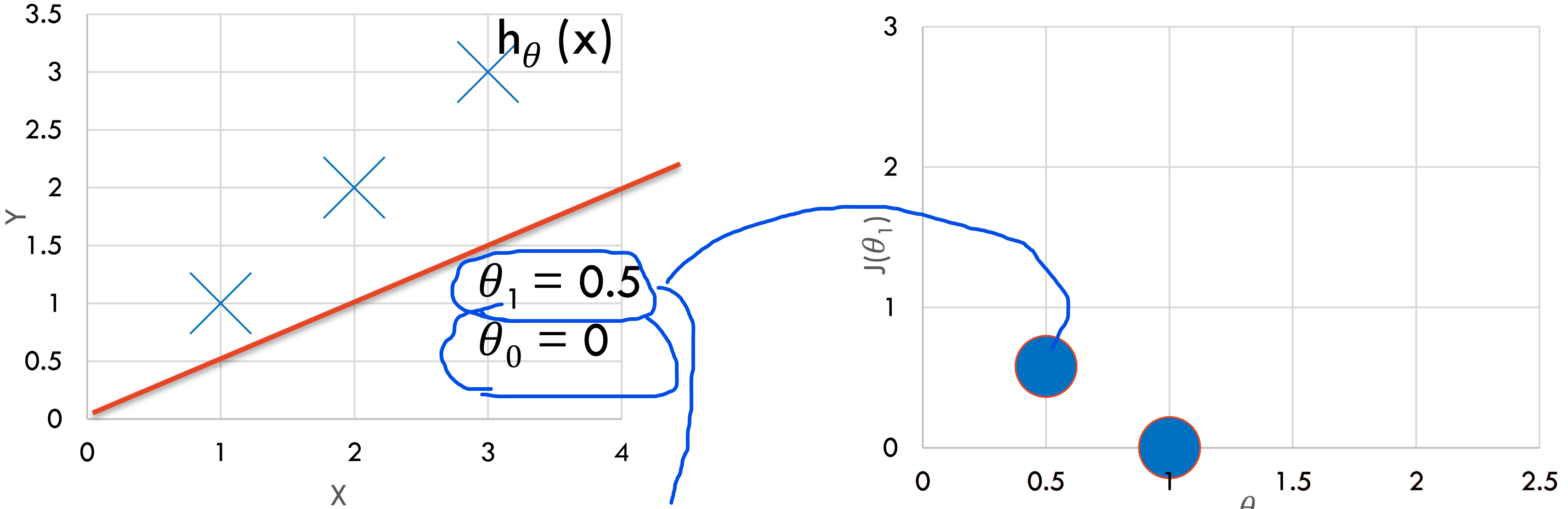


$$J([0,1]) = \frac{1}{2 \times 3}\left((1-1)^2 + (2-2)^2 + (3-3)^2\right) = 0$$

Intuition behind cost function

For insight on J(), let's assume $x \in \mathbb{R}$, so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

$$Y = h_{\boldsymbol{\theta}}(X) = \theta_0 + \theta_1 X$$

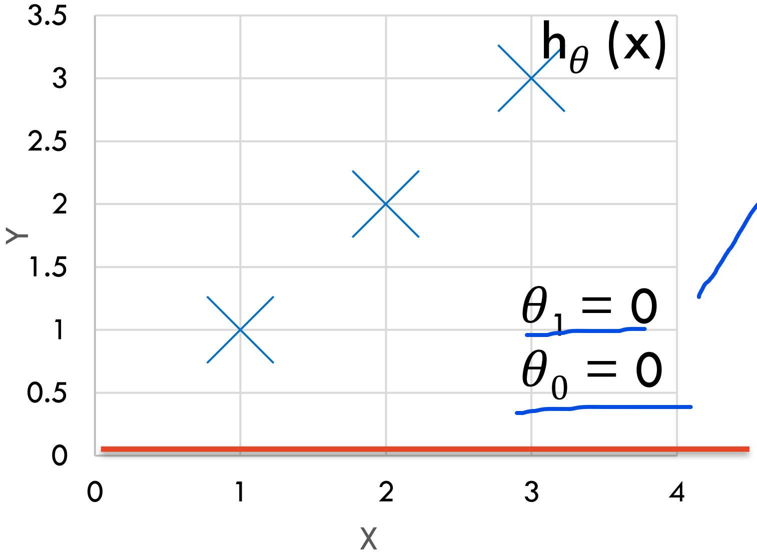


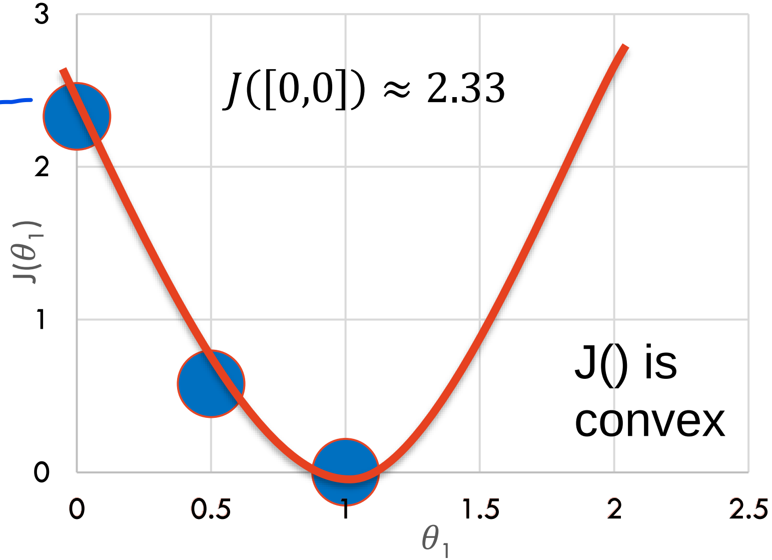
$$J([0,0.5]) = \frac{1}{2 \times 3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \approx 0.58$$

Intuition behind cost function

For insight on J(), let's assume $x \in \mathbb{R}$, so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

$$Y = h_{\boldsymbol{\theta}}(X) = \theta_0 + \theta_1 X$$





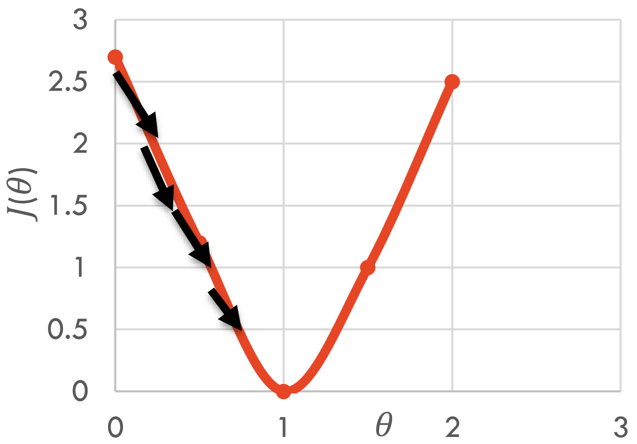
Has only one global minimum

Gradient descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha * \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

α is a learning rate (taking a small value) e.g. $\alpha = 0.05$



$$\theta_j \leftarrow \theta_j - \alpha * \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

simultaneous update for

$$j = 0 \ldots d$$

Gradient descent always converges to the global minimum (assuming α is small)

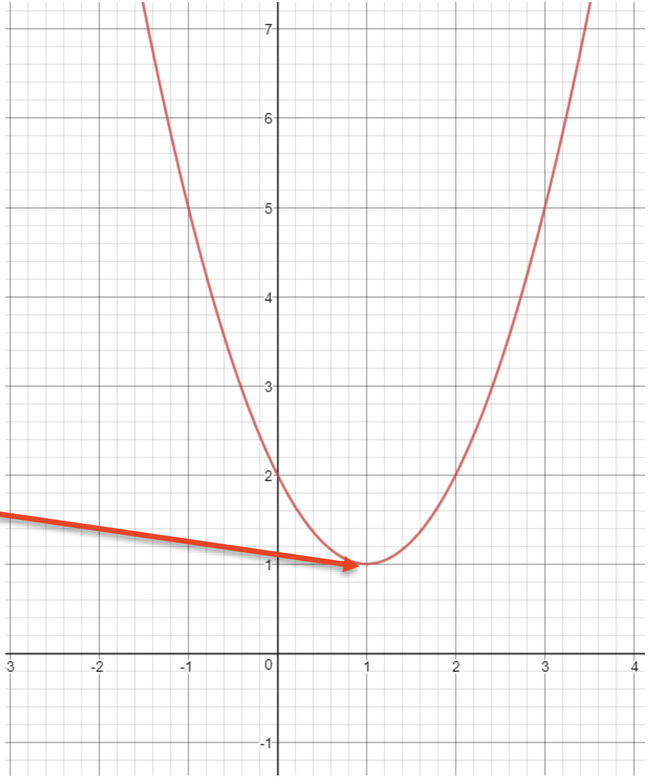
$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k \, x_k^{(i)} - y^{(i)}\right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k \, x_k^{(i)} - y^{(i)}\right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k \, x_k^{(i)} - y^{(i)}\right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k \, x_k^{(i)} - y^{(i)}\right) x_j^{(i)} \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}
\end{aligned}$$

Intuition behind gradient descent

- Given $y(x) = x^2 - 2x + 2$, find the value of x to minimise $y(x)$
- From Calculus, by finding the derivative and set it equal to zero:

$$\frac{dy(x)}{dx} = 2x - 2 = 0 \Longrightarrow x = 1$$

But this is generally hard to solve!



Intuition behind gradient descent

- So, we pick a random number
  - Let $x = 3$, which obviously is wrong
  - Step 1: we take the derivative of the function
    - $\frac{dy(x)}{dx} = 2x - 2$
  - Step 2: we study the derivative at the point we guessed $(x = 3)$
    - $\frac{dy(x)}{dx} = 2 * 3 - 2 = 4$, but the derivative at min should be zero
- Given that the derivative is positive, we know that the value is getting larger
- Therefore, we need to go backword

Intuition behind gradient descent

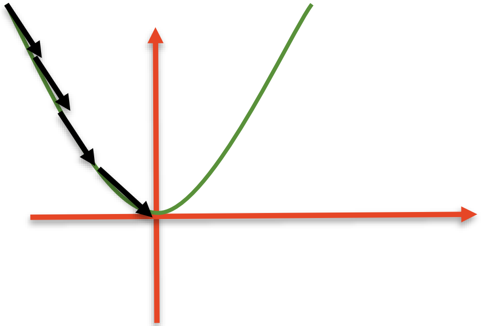
- If we have guessed $x = -1$ instead, the derivative would have been -4
  - then we would know that the function is getting smaller
- By studying the derivative of the current guess, we know if we are getting closer or further away from the minimum
- So here is the equation
  - $x_{i+1} = x_i - \alpha * \frac{dy(x)}{dx}$, α is the learning rate, e.g. $\alpha = 0.2$
- Given our example, we guessed $x_0 = 3$
  - $x_{i+1} = x_i - 0.2 * \frac{dy(x)}{dx}$
  - $x_1 = 3 - 0.2 * 4 = 2.2$

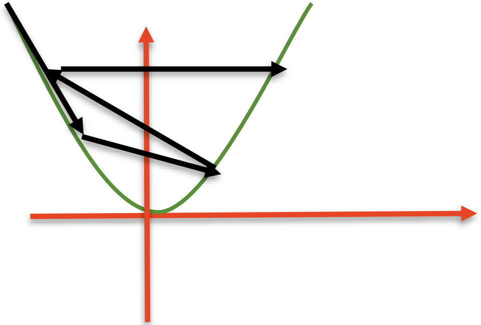
Intuition behind gradient descent

- We repeat this process again at $x_1 = 2.2$
  - $\frac{dy(x)}{dx} = 2x - 2$
  - $\frac{dy(x=2.2)}{dx} = 2 * 2.2 - 2 = 2.4$
  - $x_2 = 2.2 - 0.2 * 2.4 = 1.72$, we moved closer
- At $x_2 = 1.72$
  - $\frac{dy(x)}{dx} = 2x - 2$
  - $\frac{dy(x=1.72)}{dx} = 2 * 1.72 - 2 = 1.44$
  - $x_3 = 1.72 - 0.2 * 1.44 = 1.432$, we moved closer
- If we keep repeating this process, we can find the minimum point of the solution.

Selecting learning rate

- If α is small, gradient descent can be slow
- If α is too large, gradient descent might overshoot the minimum
- One good strategy is gradually shrinking the learning rate





Batch and stochastic gradient descents

- Batch:
  - Repeat until converge $\left\{\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}\right\}$, for every j
  - Accurate but slow:
    - has to scan through the entire training set before taking a single step
    - costly operation if n is large
- Stochastic:

  For $i = 1$ to n

  $$\left\{\theta_j \leftarrow \theta_j - \alpha \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}\right\}, \text{ for every } j$$

  - Fast, start making progress right away.
  - It may not converge to the minimum
- A trade-off between these two extremes is mini-batch

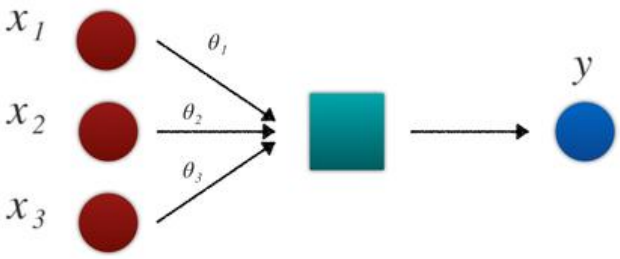
Logistic Regression

Regression vs classification

- Regression assigns a numerical value
- Output is a continuous variable (real value)
- E.g., predict house price

- Classification assigns a class to each example
- Output is a discrete / categorical variable
- E.g., predict whether tumour is harmful or not harmful

Logistic regression



- Predict probability of categorical label
- E.g., probability of defaulting on a loan given
  - Amount of debt
  - Late payment count

http://www.toshistats.net/101-4-logistic-regression/

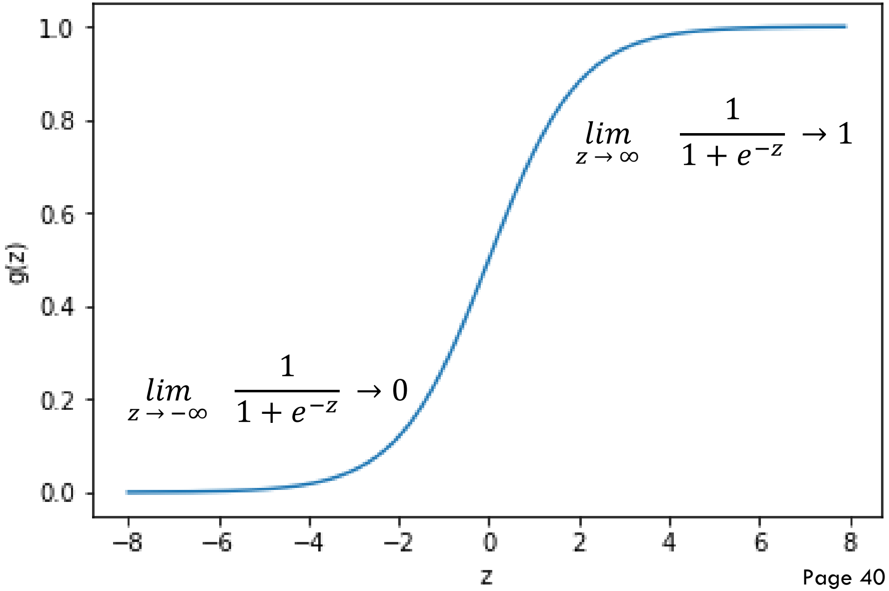
Logistic regression for classification

- Applying linear regression for classification is often not useful
- $h_\theta(x)$ can be a large positive or negative value while y is Yes or No (0 or 1) in the case of binary classification
- Logistic or sigmoid function

$$0 \leq h_\theta\,(x) \leq 1$$

$$h_\theta(x) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

- Here, $g(z) = \frac{1}{1+e^{-z}}$ and



$$\lim_{z \to \infty} \frac{1}{1+e^{-z}} \to 1$$

$$\lim_{z \to -\infty} \frac{1}{1+e^{-z}} \to 0$$

Logistic regression for classification

- A threshold is used to classify
  - If y ≥ 0.5, predict y = 1
  - If y < 0.5 , predict y = 0
- We can see:

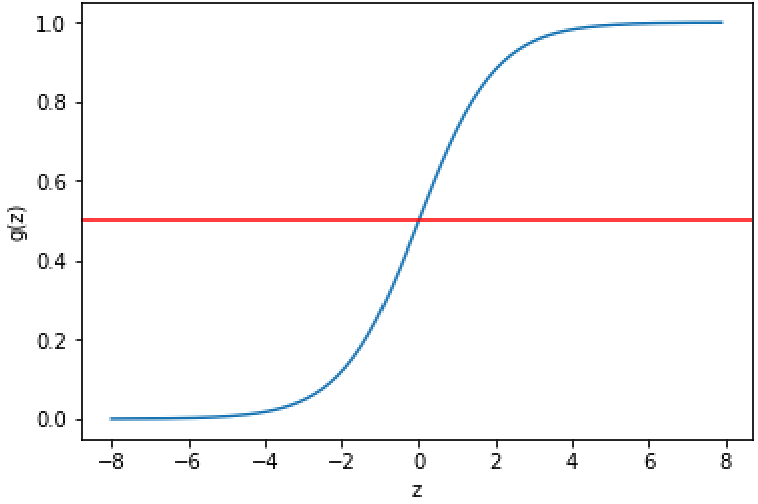
$$g(z) \geq 0.5 \quad \text{if } Z \geq 0$$
$$g(z) \leq 0.5 \quad \text{if } Z < 0$$

- Assume

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

- It can be written as :

$$P(y \mid x; \theta) = \ (h_\theta(x)\ )^y\ (1 - h_\theta(x)\ )^{1-y}$$

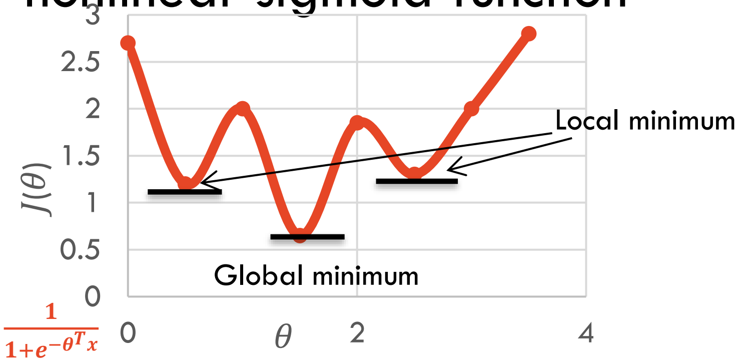


Cost function and optimization

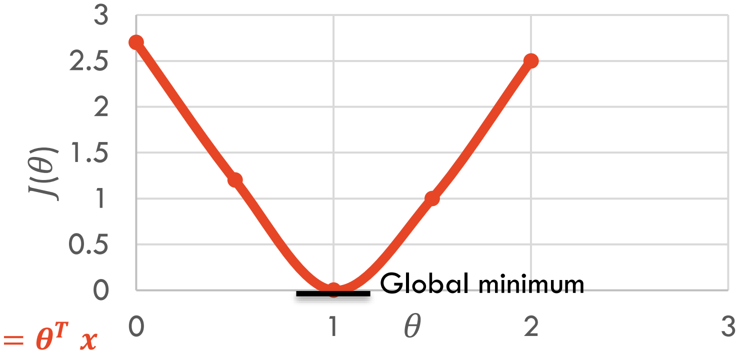
- Linear regression cost function was convex

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(\mathrm{h}_\theta(\mathrm{x}^{(i)}) - \mathrm{y}^{(i)})^2$$

- The same cost function for logistic regression is nonconvex because of nonlinear sigmoid function



$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$



$h_\theta(x) = \theta^T x$

- If our cost function has many local minimums, gradient descent may not find the optimal global minimum.

Convex cost function for logistic regression

- Instead of Mean Squared Error, we use a cost function called Cross-Entropy, also known as Log Loss.
- Cross-entropy loss can be divided into two separate cost functions: one for $y = 1$ and one for $y = 0$.
- We define logistic regression cost function as:

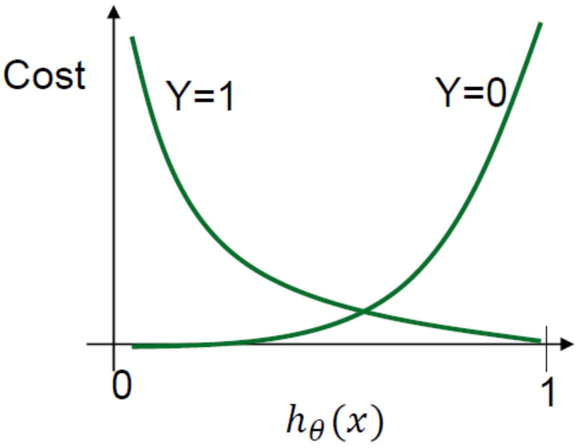
$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n} cost(h_\theta(x^{(i)}), y^{(i)})$$

$$cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

Convex cost function for logistic regression

- The two logistic functions compressed into one

$$J(\theta) = -\frac{1}{2n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h_{\theta}(\mathrm{x}^{(i)})\right) + \left(1 - y^{(i)}\right)\log(1 - h_{\theta}(\mathrm{x}^{(i)}))\right]$$



The objective is to minimize cost

- For class 0, it minimizes $h_{\theta}(x)$
- For class 1, it maximizes $h_{\theta}(x)$

Gradient descent for logistic regression

- To minimize our cost, we can use Gradient Descent

- Given:

$$g(z)' = g(z)(1 - g(z))$$

$$\frac{\partial}{\partial\theta_j} J(\theta) = (\mathrm{g(z)} - \mathrm{y})x_j$$

Multi-class classification

- One-vs-rest strategy
  - We train one logistic regression classifier for each class i to predict the probability that $y = i$
  - For each x, pick the class having highest value of probability
- Or use the softmax function

Regularization parameters in scikit-learn

- Penalty
  - **l1 (lasso)**: estimates sparse coefficients; equivalent to feature selection
  - **l2 (ridge)**: minimizes coefficients; pulls coefficients toward 0
- C
  - Inverse of regularization strength
  - Small values specify stronger regularization

$$J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2n}\sum_{i=1}^{n}(\mathrm{h}_{\boldsymbol{\theta}}(\mathrm{x}^{(i)}) - \mathrm{y}^{(i)})^2}_{\text{Model fit to data}} + \underbrace{\lambda\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$$

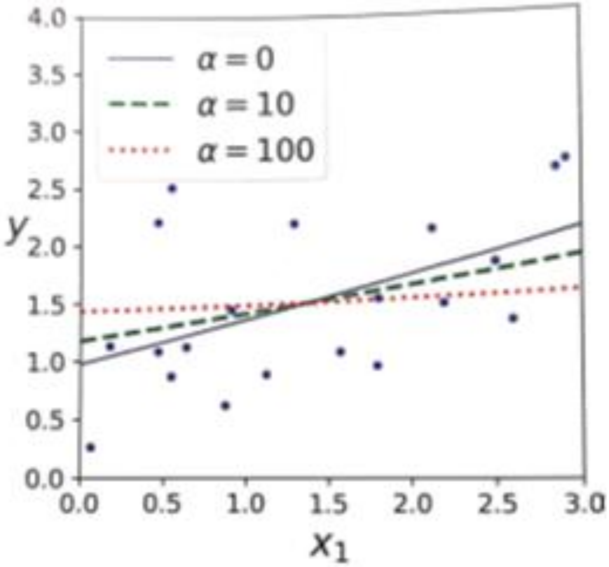


Image from A. Geron, Hands-on ML with Scikit-learn, Keras & TensorFlow

Selecting model parameters with grid search

- Parameters like penalty and regularization strength are not learned from data by default
- Can be set using exhaustive search through combinations of specified possible values
- Perform n-fold cross validation for each combination
- In scikit-learn:
  - from sklearn. model_selection import GridSearchCV

Review

THE UNIVERSITY OF SYDNEY

Tips and tricks

- Compare ML models to the simplest baseline first; Iterate
- Best strategy is often a simpler model with more/better data
- Always test on held-out data that hasn’t been used for training

Tips and tricks

Normalization

- Make sure features are on a similar scale.
- Rescale features to have zero mean and unit variance
  - Let $\bar{x}_j$ be the mean of feature j: $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$
  - Replace each value with:

$$x_j^{(i)} = \frac{x_j^{(i)} - \bar{x}_j}{s_j}$$

  - s_j is the standard deviation of feature j
- After recalling features, it is easier to find a learning rate that works well for all weights.
  - If the features are on vastly different scales, a learning rate that works well for updating one weight might be too large or too small to update the other weight equally well.

W9 review: Linear regression & logistic regression

Objective

Learn techniques for supervised machine learning, with tools in Python.

Lecture

- Simple linear regression
- Multiple linear regression
- Gradient descent
- Logistic regression

Readings

- Data Science from Scratch, Ch. 8, 14, 15, 16

Exercises

- sklearn: regression