COMP5310: Principles of Data Science

W9: Linear Regression & Logistic Regression

Presented by
Ali Anaissi
School of Computer Science
THE UNIVERSITY OF SYDNEY

Overview of Week 9

Today: Linear 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. 14-17

Exercises

- sklearn: regression

Supervised Learning:

– We'll focus on supervised machine learning techniques
  – **Simple linear regression**
  – **Multiple linear regression**
  – **Logistic regression**

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	?

$$y = f(X)$$

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

Modelling: Learn a function that maps X↦Y

Complex decision making: $\boxed{\textit{Object} \rightarrow \textit{Outcome}}$

Entity → Category
Entity → Popularity

input/independent variable $\boxed{X \rightarrow Y}$ prediction (response/dependent variable). Can be qualitative/quantitative (classification/regression).

classifier

$$\boxed{\vec{X} = (x_0, \ldots, x_d) \rightarrow Y}$$

object encoded with features (think DB attributes/ OO member fields of primitive types)
d is the feature dimensionality.

We may know the relation for certain values of X and Y:

$$\boxed{(\vec{x}, y)}$$

In fact, we may know the relation for many $\vec{x}$s and ys:

$$\boxed{\{ (\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(N)}, y^{(N)}) \}}$$

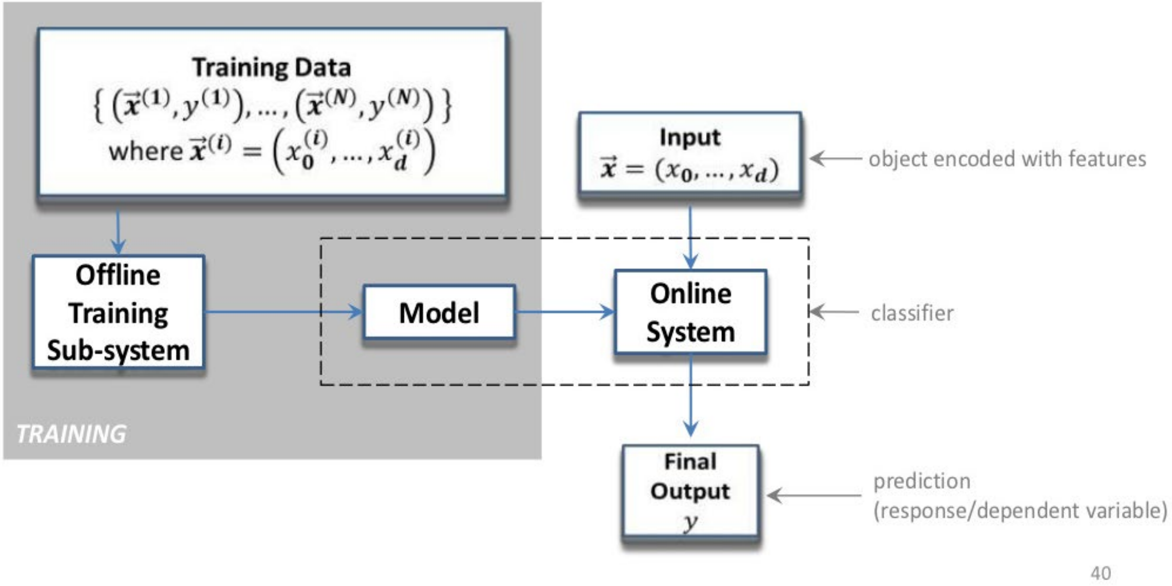
The i-th x is: $\vec{x}^{(i)} = \left(x_0^{(i)}, \ldots, x_d^{(i)}\right)$

38

http://www.slideshare.net/Nicolas_Nicolov/machine-learning-14528792

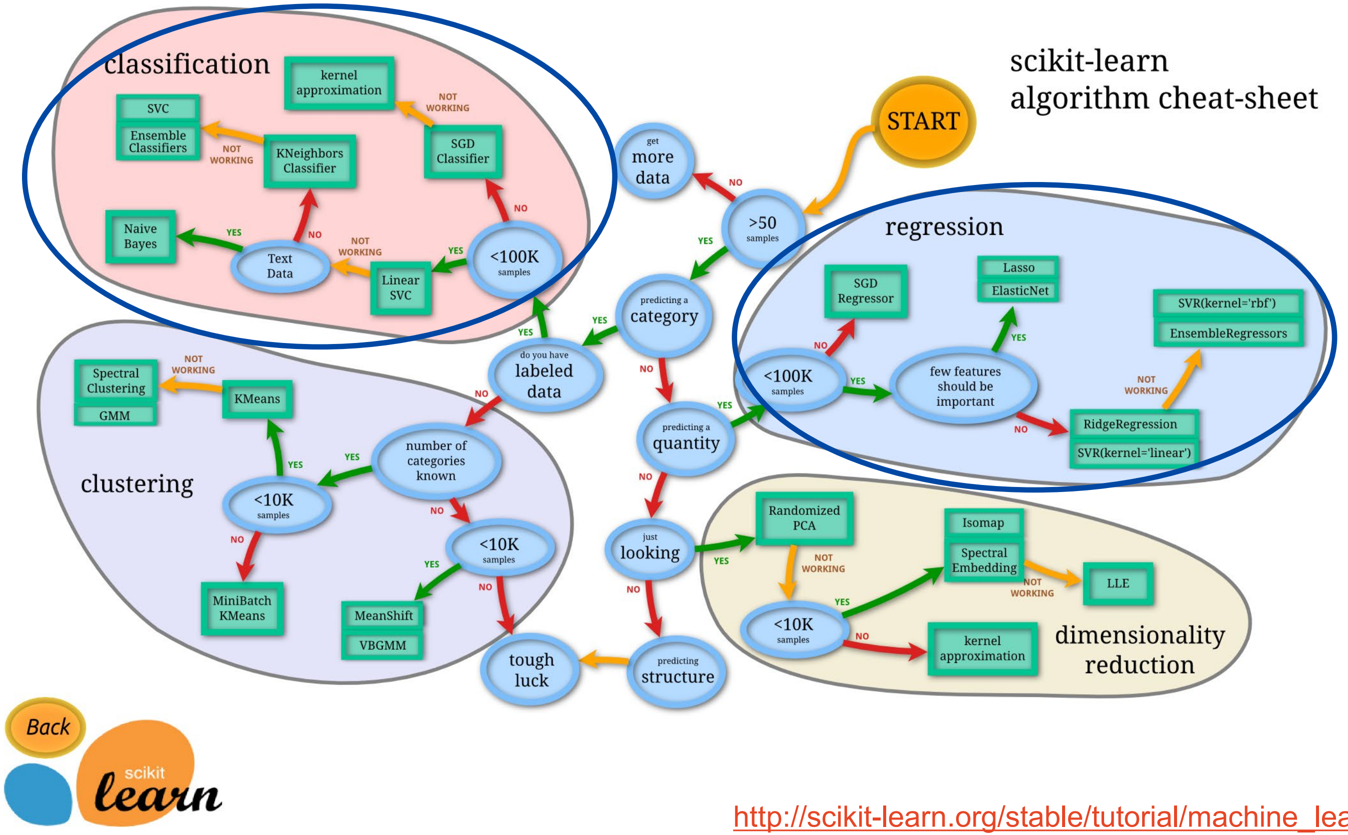
Modelling: Predict label for new feature vectors

$X \to Y$ $\qquad\qquad f(X) = Y$



http://www.slideshare.net/Nicolas_Nicolov/machine-learning-14528792

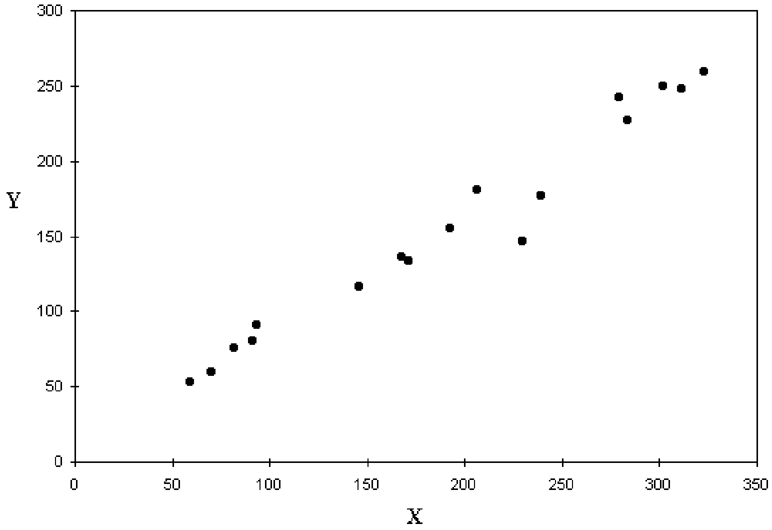
Machine learning map from scikit-learn



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

Simple Linear Regression

What is the relationship between two variables?



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

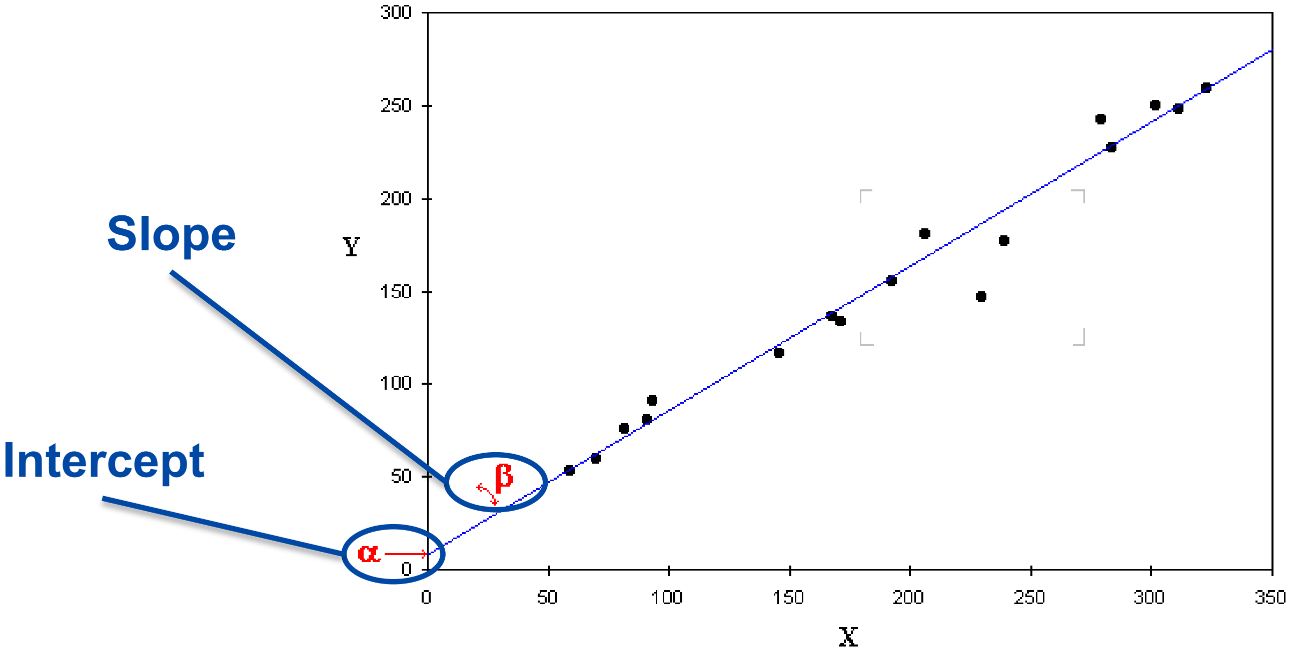
– Correlation measures the strength of the linear relationship
– Often just knowing there's a relationship isn't enough

Simple linear regression

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

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

What's the line that explains X↦Y?

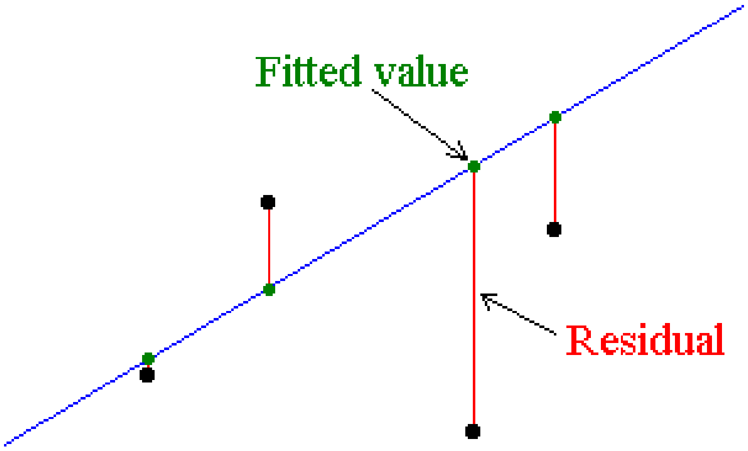


Fitting SLR: learn α and β

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

- 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



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

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

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

Sum Error Square

Ordinary Least Squares (OLS)

- Our goal is to find the optimal value of $\hat{\alpha}$ and $\hat{\beta}$ such that

$$\frac{1}{n}\sum_{i=1}^{n}\left(\left(\hat{\alpha} + \widehat{\beta}\, x_i\right) - y_i\right)^2$$ is the minimum

- Using calculus

$$\hat{\beta} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} = \frac{\operatorname{cov}(x, y)}{Var(x)}$$

$$= \frac{r(x, y) * sd(y)}{sd(x)}$$

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

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

Fitting SLR: Least squares

Slope of standardized data points with mean 0 and stdev 1

Adjust slope for variation in Y and X

```python
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
```

Intercept is the difference between means of observed and predicted y

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**

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

- Ranges from 0 to 1, with higher values indicating better fit
- Conveys goodness of fit but not precision

Standard error (S)

- Square root of the sum of squared errors divided by N

$$S = \sqrt{\frac{SSE}{N}}$$

- Measure of the prediction accuracy
- Expressed in units of the response variable

Model acceptance testing with S

- Suppose we are predicting salary from education level
  - Regression model produces r^2=0.761 and S=$2k
  - Our requirement is that predictions be within $5k
- Calculating a prediction interval from S
  - **Prediction interval:** range that should contain the response value of a new observation
- If sample size is large enough then useful rule-of-thumb:

  approximately 95% of predictions should fall within $\boxed{\hat{y}_i \pm 2 * S}$

- S must be <= $2.5k to produce a sufficiently narrow 95% prediction interval

Exercise: Simple linear regression

- Defining linear algebra and stats functions
  - code cell after “Preliminary maths functions”
  - Compare results to numpy and scipy implementations
- Simple linear regression
  - code cell after “Removing outliers”
  - code cell after “Simiple linear regression”
  - Calculate r-squared and standard error
  - Assess fit and precision
  - Compare results to scipy implementation

Multiple Linear Regression

Multiple linear regression

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

- Explain the relationship between:
  - **two or more** explanatory variables
  - one response variable

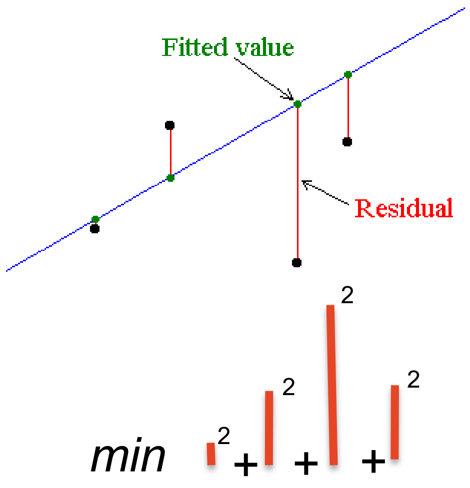
How to learn θ

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

Assume $x_0 = 1$

- Cost function: $J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$
- Fit model by solving $\min_{\theta} J(\theta)$

- Basic search procedure
  - Choose initial value for θ
  - Until we reach a minimum:
    - Choose a new value for θ to reduce $J(\theta)$



Intuition behind cost function

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

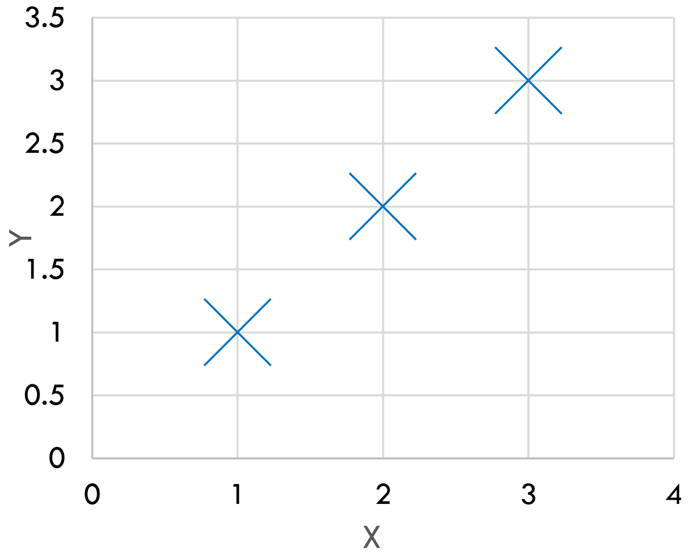
$$Y = h_\theta(x) = \theta_0 + \theta_1 x_1$$

Lets say we have the following data points:

X	Y
1	1
2	2
3	3

$\theta_0 = 0$ and $\theta_1 = 1$

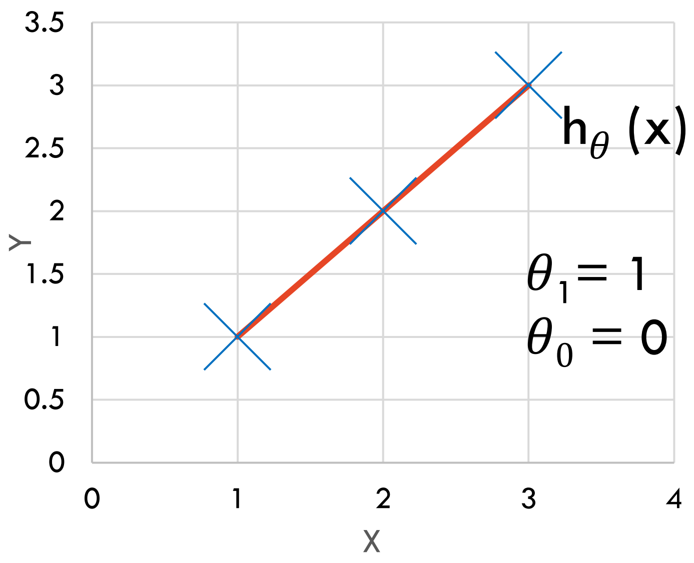
$$Y = h_\theta(x) = X$$

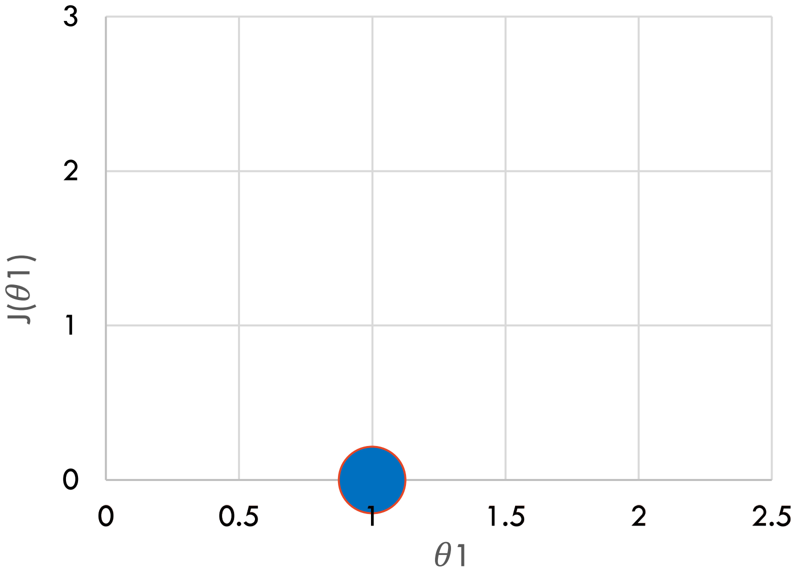


Intuition behind cost function

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

Y = h_θ (x) $= \theta_0 + \theta_1 x_1$



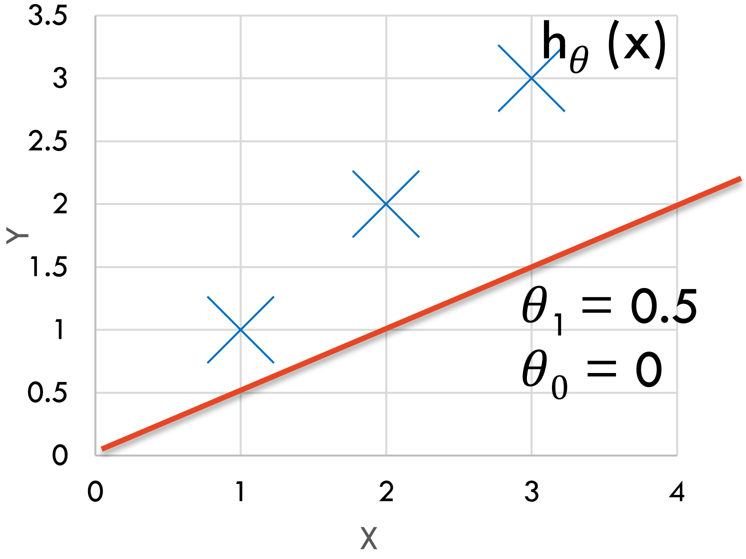


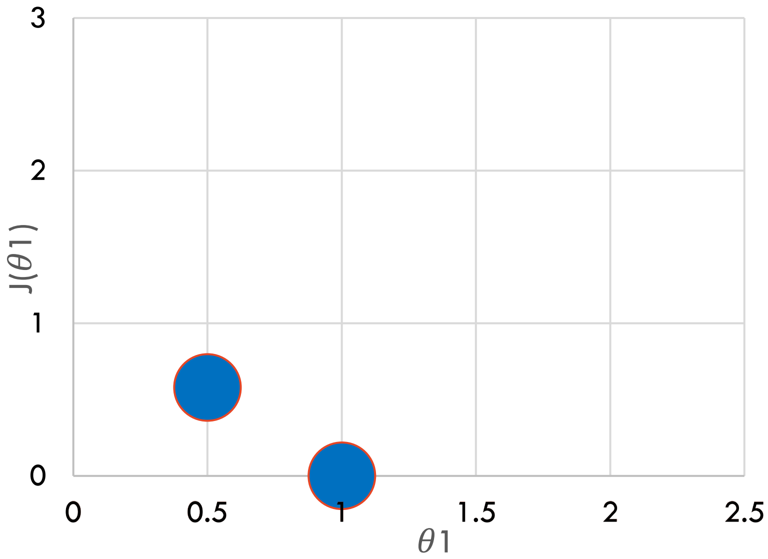
$j([0,1])=1/(2\times 3)\ [(1-1)^2+(2-2)^2+(3-3)^2\ ] = 0$

Intuition behind cost function

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

$$Y = h_\theta(x) = \theta_0 + \theta_1 x_1$$



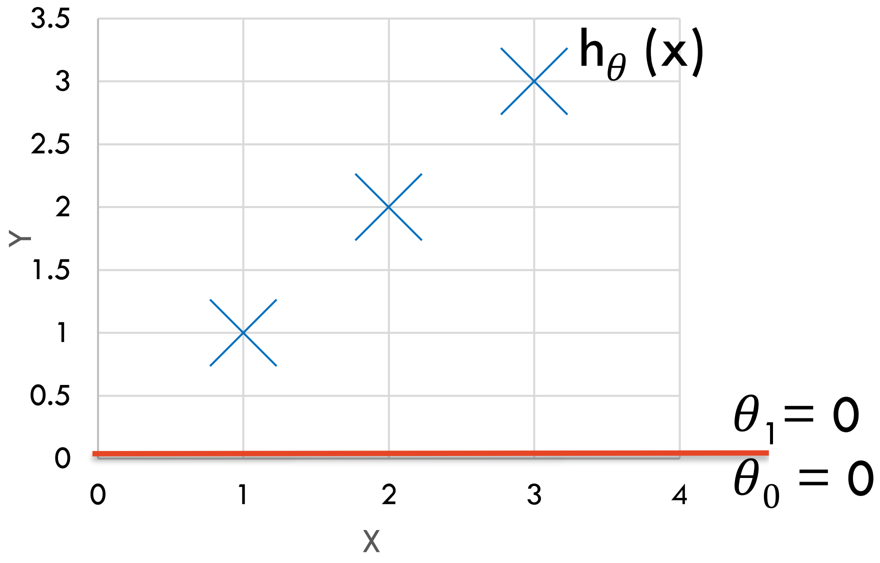


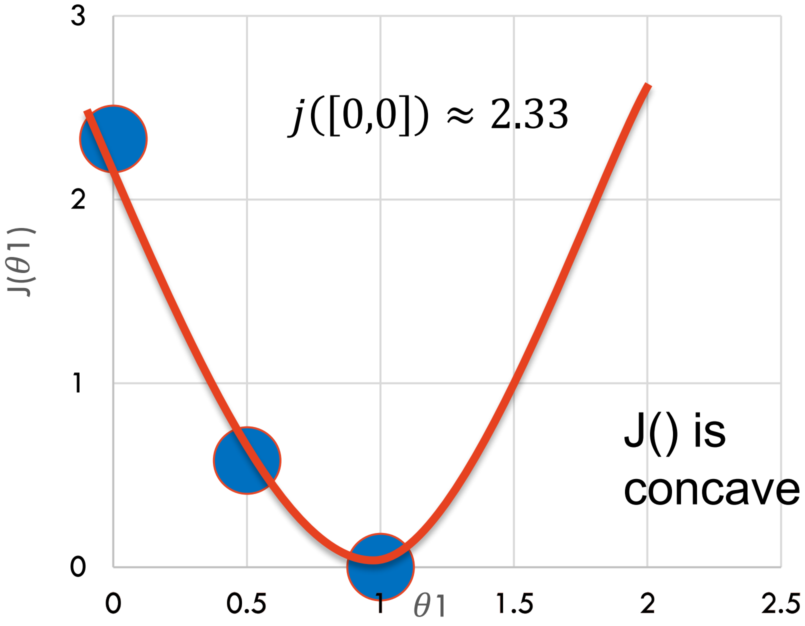
$$j([0,0.5]) = \frac{1}{2 \times 3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \approx 0.58$$

Intuition behind cost function

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

$$Y = h_\theta(x) = \theta_0 + \theta_1 x_1$$





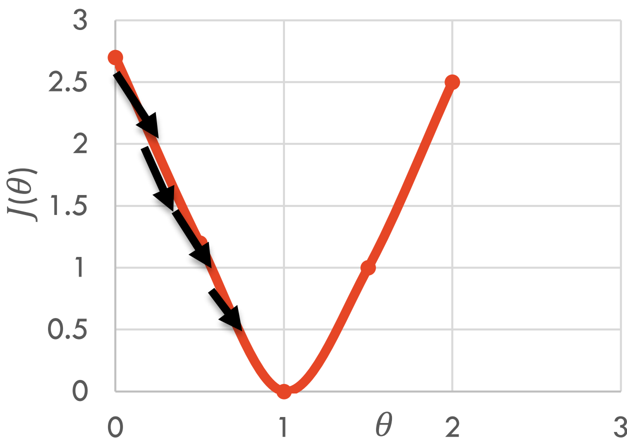
Has only one global minimum

Gradient descent

- Initialize θ
- Repeat until convergence

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

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



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

simultaneous update for
$j = 0 \ldots d$

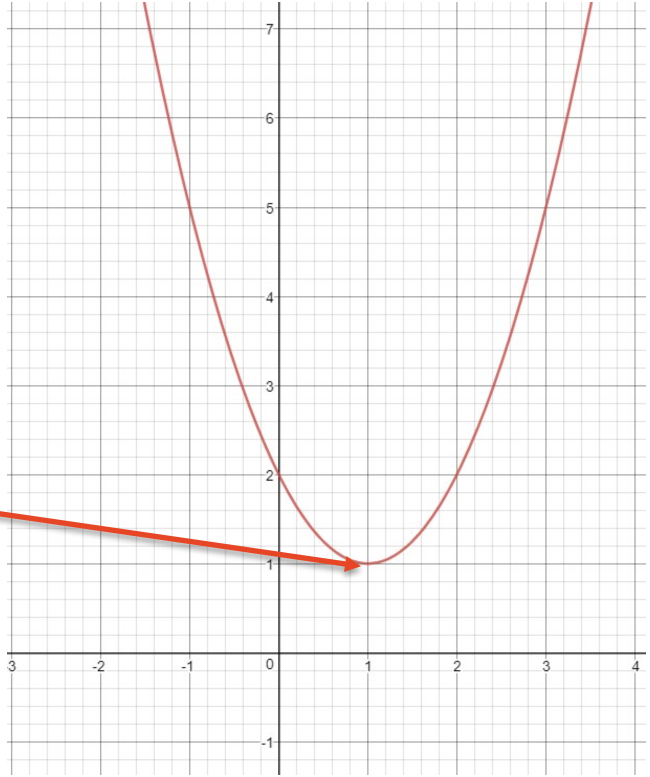
$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_j \, x_j^{(i)} - y^{(i)} \right)^2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

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

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:

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



Intuition behind gradient descent

- With gradient descent, we don't know the optimal value of x,
- 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

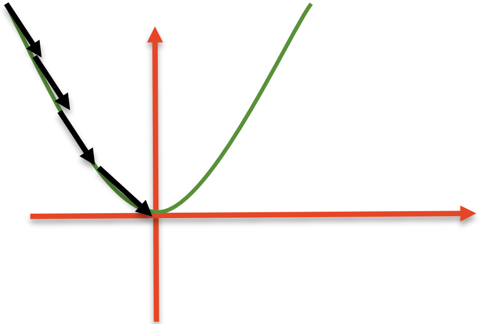
- If we have guessed $x = -1$ instead, the derivative would have been -4
  - then we would know that the function is getting smaller
- By studying the derivative of the current guess, we know if we are getting closer or further away from the minimum
- So here is the equation
  - $x_{i+1} = x_i - \alpha * \frac{dy(x)}{dx}$ # α = 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

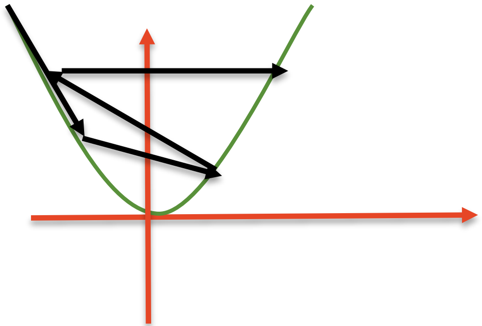
- We repeat this process again at $x_1 = 2.2$
  - $\frac{dy(x)}{dx} = 2x - 2$
  - $\frac{dy(x=2.2)}{dx} = 2 * 2.2 - 2 = 2.4$
  - $x_2 = 2.2 - 0.2 * 2.4 = 1.72$, we moved closer
- At at $x_2 = 1.72$
  - $\frac{dy(x)}{dx} = 2x - 2$
  - $\frac{dy(x=1.72)}{dx} = 2 * 1.72 - 2 = 1.44$
  - $x_3 = 1.72 - 0.2 * 1.44 = 1.432$, 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



Batch and stochastic gradient descents

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

  $For\ i = 1\ to\ n$

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

  - Fast, start making progress right away.
  - It may not converged to the minimum
- When the training set is large, stochastic gradient descent is often preferred over batch gradient descent

Some points before implementation

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

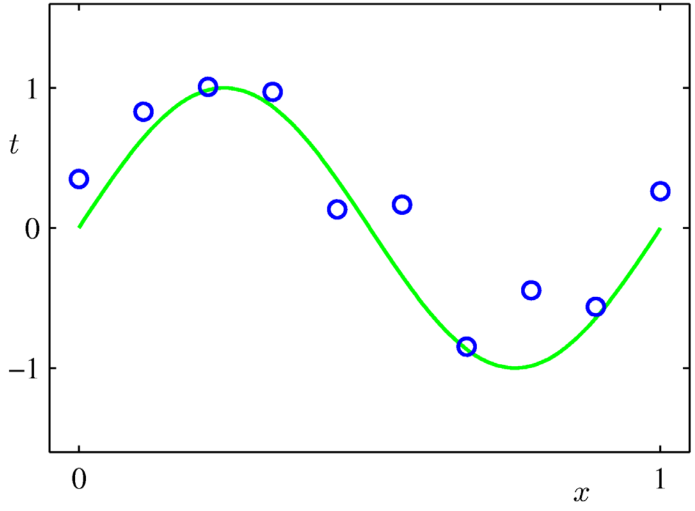
$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{s_j}$$

  - s_j is the standard deviation of feature j

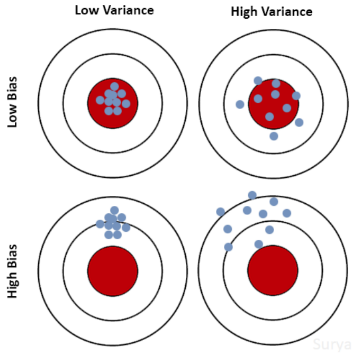
Extending linear regression to more complex models

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

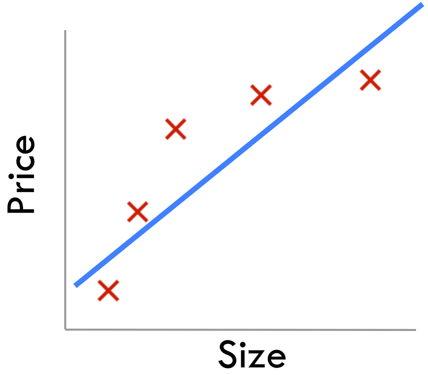
- This allows use of linear regression techniques to fit non-linear datasets.



Quality of fit



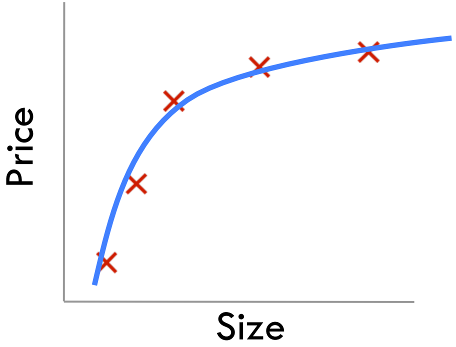
- Overfitting:
  - The learned model may fit the training set very well
  - ...but fails to generalize to new examples



$$\theta_0 + \theta_1 x$$

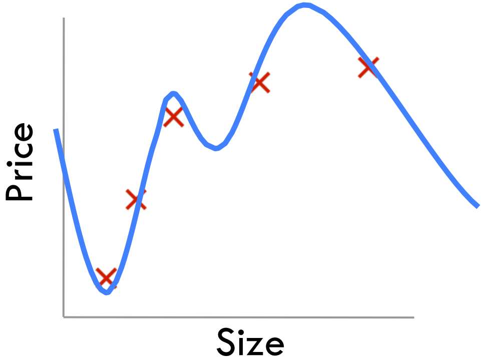
Under-fitting

(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 error 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
  - $$J(\theta) = \underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2}_{\text{Model fit to data}} + \underbrace{\lambda \sum_{j=1}^{d} \theta_j^2}_{\text{regularization}}$$
- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0
- Gradient update:
  - $$\theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)} - \frac{\lambda}{n}\theta_j \qquad j = 1 \ldots d$$

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)
```

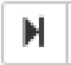
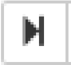
- **Estimator:** a Python object that implements the methods fit(X, y) and predict(T)
- **Fit(X, y):** fits a model to the training data X, y
  - **X:** feature vectors
  - **y:** labels
- **Predict(T):** predict labels for new data T

Assessing fit and standard error

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

# We can also calculate the standard error
stderr = math.sqrt(np.mean((Y_train - lm.predict(X_train))**2))
print('\nStandard error:', stderr)
```

Exercise: Multiple linear regression

- Linear regression in scikit-learn
  -  code cell after “Loading and visualising data”
  -  code cell after “Linear regression in scikit-learn”
- Evaluating linear regression
  - How are are fit and precision?
  - Is a linear model appropriate?

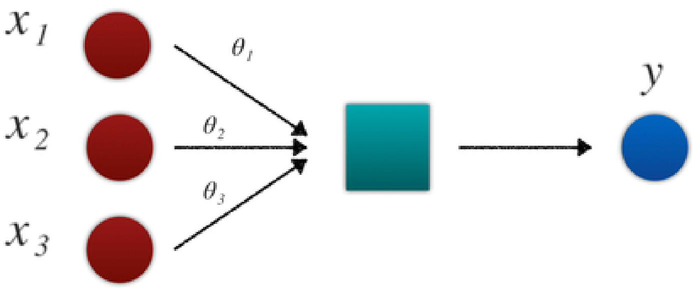
Logistic Regression

Classification vs regression

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

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

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

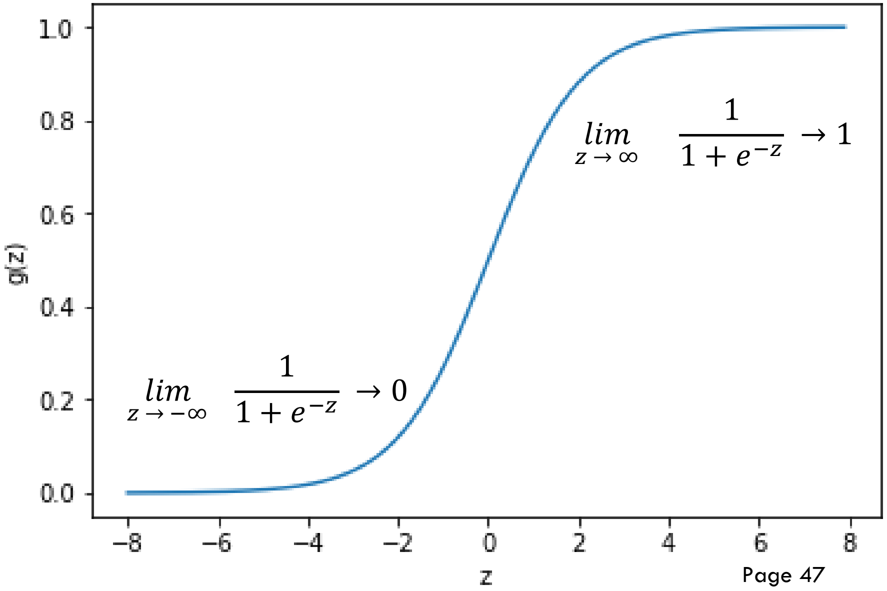
- Applying linear regression for classification is often not useful
- h_θ (x) can be a large positive or negative value while y is Yes or No (0 or 1) in case of binary classification
- Logistic or sigmoid function

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

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

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

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



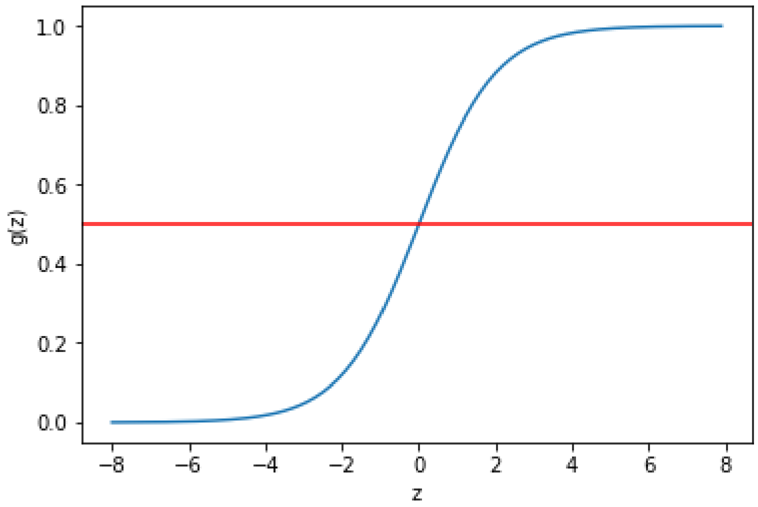
Logistic regression for classification

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

$$g(z) \geq 0.5 \quad if \;\; Z \geq 0$$
$$g(z) \leq 0.5 \;\; if \;\; Z < 0$$

- Assume

$$P(\, y = 1 \,|\, x; \theta \,) = h\theta \; (x)$$
$$P(\, y = 0 \,|\, x; \theta \,) = 1 - h\theta \; (x)$$



- It can be written as :

$$P(\, y \,|\, x; \theta \,) = \; (h\theta \; (x) \;)^{y} \; (1 - h\theta \; (x) \;)^{1-y}$$

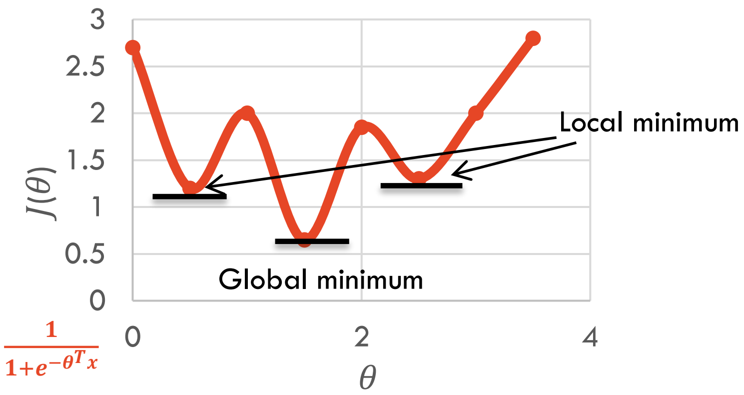
Cost function and optimization

– Linear regression cost function was convex

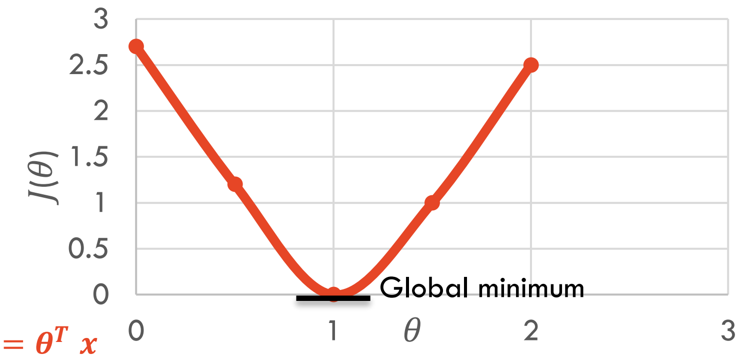
$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

– 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 :

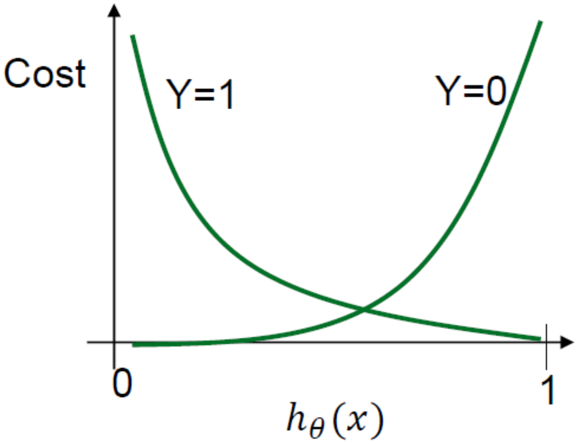
$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n} cost(\mathrm{h}_\theta(\mathrm{x}^{(i)}), \mathrm{y}^{(i)})$$

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

Convex cost function for logistic regression

– The two logistic functions compressed into one

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



- If $h_{\theta}(x)$ goes to zero and Cost also goes to zero, Class 0 is selected
- If h goes to 1 and Cost goes to zero, class 1 is selected

Gradient descent for logistic regression

- To minimize our cost, we use Gradient Descent just like before in Linear Regression.
- Given:

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

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

Multi-class classification

- One-vs-all 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
- One versus one strategy
  - we train binary classifiers corresponding to every combination of two class classifiers.
  - For the test data, we use all the classifiers to classify the data and then count the number of times that the test data was assigned to each class.
  - The final class is the one with the maximum number of wins.

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

Selecting model parameters with grid search

- Parameters like penalty and regularization strength are not learnt 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

Exercise: Logistic regression

- Logistic regression in scikit-learn
  - code cell after “Loading and visualising data”
  - code cell after “Logistic regression in scikit-learn”
- Evaluating logistic regression
  - code cell after “Evaluating classification”
  - Choose C and penalty settings using grid search

Review

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
- More next week…

Project Stage 2: Experiment, Quantify, Report

Objective

Complete a piece of data science work to answer a question or provide an intelligent data-driven tool.

Activities

- Define experimental framework
- Perform analysis or build tool
- Describe evaluation and conclusions

Output

- 4-page report describing framework, analysis and conclusions (plus code)
- Demonstration (2-3/3-4 mins)

Marking

- 20% of overall mark
  - 15% report and code
  - 5% pitch

Suggested timeline for project stage 2

- W7: Define experimental framework
- W8: *Implement approach*
- **W9: Write first page (framework, approach)**
- W10: Evaluate and benchmark approach
- W11: Analyse and characterise results
- W12: Submit full report (W9 + results, analysis, conclusions)
  W12: Deliver demonstration