Classification

Regression

Huiping Cao

Outline

1 Linear regression

2 Solution

3 Logistic Regression

4 Parameter fitting and cost function

Regression problem

- **Regression problem**: predict the value of one or more *continuous target* variables y given the value of a d-dimensional vector $\mathbf{x}$.
    - Output is a *real number*, which is why the method is called *regression*
- **Linear Regression** is one of the most basic and important technique for usually predicting a value of an attribute (y).
    - It is used to fit values in a forecasting or predictive model.

Linear regression – Example

- A collection of observations of the Old Faithful geyser in the USA Yellowstone National Park.

```
> head(faithful)
  eruptions waiting
1     3.600      79
2     1.800      54
3     3.333      74
4     2.283      62
5     4.533      85
6     2.883      55
```

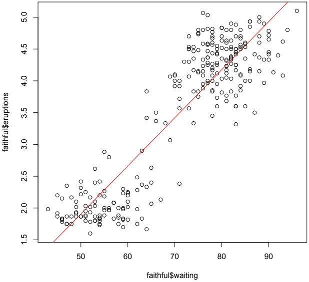
- There are two observation variables in the data set.
  - *eruptions*: the duration of the geyser eruptions.
  - *waiting*: the length of waiting period until the next eruption.
- Predict eruption time given waiting time.

Linear regression – representation

- Training set: $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\}$
  - n is the number of training data points
  - $\mathbf{x}^{(i)}$ is a d-dimensional data point (or vector)
  - $\mathbf{y} = (y^{(1)}, \cdots, y^{(n)})$ is output
  - $(\mathbf{x}, y)$ one training example
  - $(\mathbf{x}^{(i)}, y^{(i)})$ the i-th training example.
- Hypothesis: $h(\cdot)$.
  The learning algorithm learns the hypnosis.
  Using h, we can predict y for any given $\mathbf{x}$.

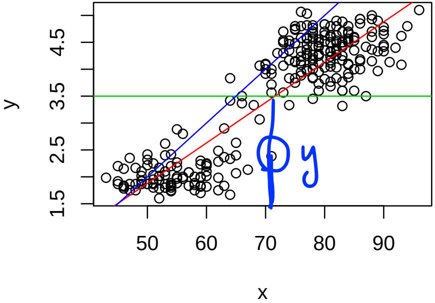
Linear regression – How do we represent h

- Linear regression with one variable (univariate linear regression).
- $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x$
- Or, $h(\mathbf{x}) = \theta_0 + \theta_1 x$



Linear regression – cost function

- Hypothesis $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x$.
- Parameters: θ_is.
- How to choose θ_is?



Linear regression – cost function (cont.)

- Hypothesis: $h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x^{(i)}$.
- Idea: Choose θ_0 and θ_1 such that $h_\theta(\mathbf{x})$ is close to $\mathbf{y}$ for our training examples.
- Intuitively

$$minimize_{\theta_0,\theta_1} \sum_{i=1}^{n} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2 .$$

Linear regression – cost function (cont.)

- Cost function,

$$J(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$$

  - Factor $\frac{1}{n}$ is to average the cost, and the factor $\frac{1}{2}$ is to make the analysis easier.
  - J is called squared error function, or cost function, or squared error cost function.
  - There are other cost functions, but squared error cost function is working well.
- Goal: $minimize_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

Multivariate Linear Regression - motivation and notation

- Example

  $\mathbf{y}$ house prices

  $x_{\cdot 1}$: house square feet

  $x_{\cdot 2}$: number of bedrooms

  $x_{\cdot 3}$: age of home

  etc.

- Notation
  - d: number of features
  - $\mathbf{x}^{(i)}$: the i-th input example
  - $x_j^{(i)}$: the value of the j-th feature of the ith training example.

Multivariate Linear Regression - Hypothesis

λ

- $h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_d x_d^{(i)}$

  E.g., $h_\theta(\mathbf{x}^{(i)}) = 80 + 0.1x_1^{(i)} + 0.01x_2^{(i)} + 3x_3^{(i)} - 2x_4^{(i)}$
- For convenience, denote $x_0^{(i)} = 1$ to define a 0th feature.
- We can define two vectors $\mathbf{x}$ and θ

$$\mathbf{x}^{(i)} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \dots \\ x_d^{(i)} \end{pmatrix} \in \mathbb{R}^{d+1}, \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_d \end{pmatrix} \in \mathbb{R}^{d+1}$$

- $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$

Multivariate Linear Regression - Representation

- Hypothesis: $h_\theta(\mathbf{x}^{(i)}) = \theta^T \mathbf{x}^{(i)}$ where $x_0^{(i)} = 1$
- Parameters: $\theta_0, \theta_1, \theta_2, \cdots, \theta_d$ (or, $\theta \in \mathbb{R}^{d+1}$)
- Cost function:

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$Y = \theta_0 + \theta_1 X_1$

Or

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

- Linear regression models: linear functions of the adjustable parameters.
  - The simplest form of linear regression models are also linear functions of the input variables.

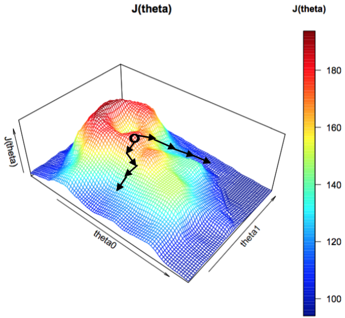
Linear regression model

- Formally,
  - Given (1) a training data set $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\}$ where $\mathbf{x}^{(i)} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \cdots \\ x_d^{(i)} \end{pmatrix}$

    together with (2) corresponding target variables $\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{pmatrix}$
  - Goal: predict the value of y for a new observation $\mathbf{x}$.
  - Linear model $h_\theta(\mathbf{x}^{(i)}) = \theta^T \mathbf{x}^{(i)}$ or $h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_d x_d^{(i)}$
    Decides $d+1$ parameters (or a $(d+1)$-vector) $\theta = (\theta_0, \theta_1, \cdots, \theta_d)$ s.t. $\frac{1}{2n} \sum_{i=1}^{n} (h(\mathbf{x}^{(i)}) - y^{(i)})^2$ is minimal.
- Solution: (1) gradient descent, (2) normal equation (or analytic pseudo-inverse algorithm)

Gradient descent – intuition

(θ_0, θ_1)



Gradient descent algorithm

- Repeat until convergence
  - $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ for $j = 0$ and $j = 1$
- Notations: partial derivative $\frac{\partial}{\partial \theta_0}$; Derivative $\frac{d}{d\theta_1}$
- α: learning rate; large α indicates aggressive learning;
- θ_0 and θ_1 need to be updated simultaneously.
  - CORRECT:
    $temp_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
    $temp_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
    $\theta_0 = temp_0$
    $\theta_1 = temp_1$
  - INCORRECT:
    $temp_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
    $\theta_0 = temp_0$
    $temp_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
    $\theta_1 = temp_1$

Multivariate Linear Regression - Feature Scaling (1)

- Idea: Make sure that features are on a similar scale for the dataset which has multiple features. The gradient descent can converge more quickly.
- For example, if a problem's dataset has two features $x_1^{(i)}$ and $x_2^{(i)}$
  $x_1^{(i)}$ = size (0-2000 feet^2)
  $x_2^{(i)}$ = number of bedrooms (1-5)
- It takes long time to converge.
- A strategy we do is to scale the feature values
  $x_1^{(i)} = \frac{size(feet^2)}{2000}$
  $x_2^{(i)} = \frac{\text{number of bedrooms}}{5}$
  Then, the contour plot of $J(\theta)$ is much less skewed (circle shape).

Multivariate Linear Regression - Feature Scaling (2)

- Idea: make every feature into approximately $-1 \leq x_j^{(i)} \leq 1$ range.
  - $x_0^{(i)} = 1$ fits in the range.
- Generally, as long as the range is similar, it is OK.
  - E.g., $0 \leq x_1^{(i)} \leq 3$ and $-1 \leq x_2^{(i)} \leq 1.5$.
  - Adding two more features ($-200 \leq x_4^{(i)} \leq 200$ and $-0.001 \leq x_4^{(i)} \leq 0.001$) makes the range very different. This is not desirable.

Multivariate Linear Regression - Feature Scaling (3)

- **Mean normalization**: another way of feature scaling.
- Replace $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$ to make features have approximately zero mean (do not apply to $x_0^{(i)}$=1).
  - E.g., $x_1^{(i)} = \frac{size-1000}{2000}$, $x_2^{(i)} = \frac{\#bedrooms-2}{5}$.
    Then, $-0.5 \le x_1^{(i)} \le 0.5$, $-0.5 \le x_2^{(i)} \le 0.5$
- Generally, replace $x_j^{(i)}$ by $\frac{x_j^{(i)}-\mu_j}{\sigma_j}$ where σ_j can be the standard deviation or $max(x_j^{(i)}) - min(x_j^{(i)})$.
  - e.g., $0 \le x_1^{(i)} \le 3$ and $-1 \le x_2^{(i)} \le 1.5$.
  - Adding two more features ($-200 \le x_4^{(i)} \le 200$ and $-0.001 \le x_4^{(i)} \le 0.001$) makes the range very different. This is not desirable.

Multivariate Linear Regression - Normal Equation

- **Normal equation**: method to solve for θ analytically

- Normal equation method has some advantages and disadvantages, which will be discussed later.

Review

- Given the value of a d-dimensional vector $\mathbf{x}$ and corresponding y values
- Regression problem: predict the value of one or more continuous target variables y .

$$h(\mathbf{x}) = \theta^T \mathbf{x}$$

- Classification problem: output is *binary*.

$$h(\mathbf{x}) = sign(\theta^T \mathbf{x})$$

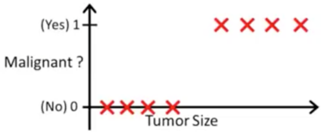
- Define a cost function.
- Solution: decides θ such that cost function is minimal.

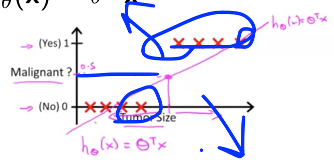
Logistic regression - Motivation

- Binary classification problem: $y \in \{0, 1\}$
  - 0: Negative class (e.g., benign tutors, normal emails, normal transactions)
  - 1: Positive class (e.g., malignant tumors, spam emails, fraudulent transactions)
- Multiclass classification problem: $y \in \{0, 1, 2, 3, 4\}$

Logistic regression - Motivation (1)

- Given $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}$ and corresponding $(y^{(1)}, \cdots, y^{(n)})$.
- Can utilize linear regression: $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$

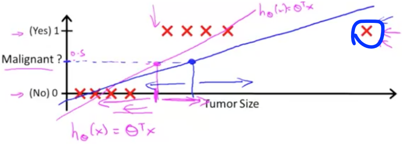




- Threshold classifier output
  - If $h_\theta(\mathbf{x}) \geq 0.5$, predict $y = 1$
  - If $h_\theta(\mathbf{x}) < 0.5$, predict $y = 0$

Logistic regression - Motivation (2)

- Issue with this approach. Example (add one extra non-critical point)

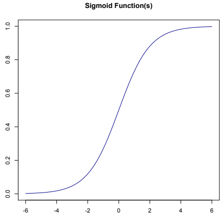


- If we run linear regression, the line will be different. Everything to the right of a point, we predictive it to be positive.
- Directly applying linear regression to do classification generally does not work well.

Logistic regression - Hypothesis

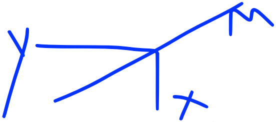
- Logistic regression, generate output in [0, 1]: $0 \leq h_\theta(\mathbf{x}) \leq 1$
- Define $h_\theta(\mathbf{x})$ to be $g(\theta^T \mathbf{x})$
- Utilize a logistic function (or sigmoid function) $g(z) = \frac{e^z}{1+e^z}$ (or, rewritten as $\frac{1}{1+e^{-z}}$), get the hypothesis

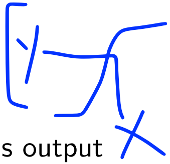
$$h(\mathbf{x}) = g(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$



- Sigmoid function

Logistic regression - Hypothesis (cont.)

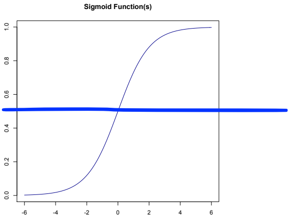




- Interpretation of hypothesis output
  - $h_\theta(\mathbf{x})$: for the input $\mathbf{x}$, the estimated probability that $y = 1$.
- Example: if $y = 1$ means a tumor is malignant, $h_\theta(\mathbf{x}) = 0.7$ tells that 70% chance the tumor is malignant.
- $h_\theta(\mathbf{x}) = P(y = 1|\mathbf{x}; \theta)$
  Probability that $y = 1$ given $\mathbf{x}$, parameterized by θ.
  - $P(y = 0|\mathbf{x}; \theta) + P(y = 1|\mathbf{x}; \theta) = 1$
  - $P(y = 0|\mathbf{x}; \theta) = 1 - P(y = 1|\mathbf{x}; \theta)$

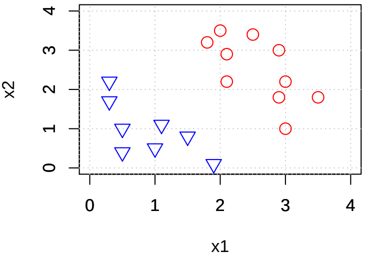
Logistic regression - Decision boundary

- Hypothesis: $h_\theta(\mathbf{x}) = g(\theta^T\mathbf{x})$ where $g(z) = \frac{e^z}{1+e^z}$
- When will we predict $y = 0$ or $y = 1$?
- Suppose that
  - we predict $y = 1$ if $h_\theta(\mathbf{x}) \geq 0.5$
  - we predict $y = 0$ if $h_\theta(\mathbf{x}) < 0.5$



- Re-examine the sigmoid function
  - when $z \geq 0$, $g(z) \geq 0.5$; in this case, we predict $y = 1$
    Equivalently, when $\theta^T\mathbf{x} \geq 0$, $h_\theta(\mathbf{x}) = g(\theta^T\mathbf{x}) \geq 0.5$
  - when $z < 0$, $g(z) < 0.5$; in this case, we predict $y = 1$
    Equivalently, when $\theta^T\mathbf{x} < 0$, $h_\theta(\mathbf{x}) = g(\theta^T\mathbf{x}) < 0.5$

Logistic regression - Decision boundary (cont.)



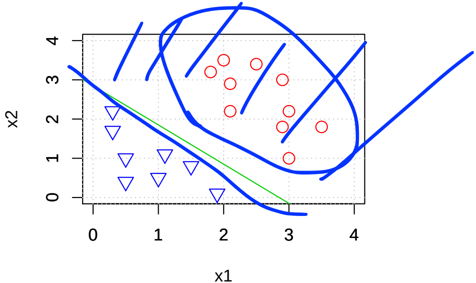
- Training data: red circle (class 1), blue triangle (class -1)
- $h_\theta(\mathbf{x}^{(i)}) = g(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)})$

$\theta^T X$

- Suppose that we have the hypothesis parameter $\theta = \begin{pmatrix} -3 \\ 1 \\ 1 \end{pmatrix}$
- How do we make prediction?
  Predict $y = 1$ if $-3 + x_1 + x_2 \geq 0$ (i.e., $x_1 + x_2 \geq 3$)

Logistic regression - Decision boundary (cont.)

- $h_\theta(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$



- If we draw a line $x_1 + x_2 = 3$, the points above this line are predicted to be $y = 1$; the points below this line are predicted to be $y = 0$.
- Line $x_1 + x_2 = 3$ is called **decision boundary**, which separates the regions for prediction of $y = 0$ and $y = 1$.

Logistic regression - Fit the parameters θ (1)

- Training set: $\{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(n)}, y^{(n)})\}$ with n examples
  - $\mathbf{x} = \begin{pmatrix} x_{\cdot,0} \\ x_{\cdot,1} \\ \cdots \\ x_{\cdot,d} \end{pmatrix}$ where $x_{\cdot,0} = 1$
  - $y \in \{0, 1\}$
- How to choose parameters θ?

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

$\theta^T x$

Logistic regression - Fit the parameters θ (cont.)

- Cost function for linear regression:

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

- Define

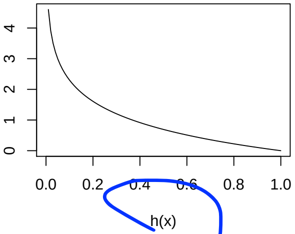
$$\mathrm{cost}(h_\theta(\mathbf{x}^{(i)}), y^{(i)}) = \frac{1}{2}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$$

- Logistic regression

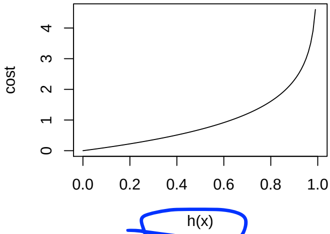
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{cost}(h_\theta(\mathbf{x}^{(i)}), y^{(i)})$$

Logistic regression - log function

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



$-\log(h_\theta(\mathbf{x}))$



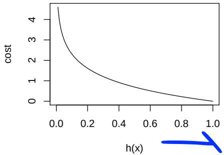
$-\log(1 - h_\theta(\mathbf{x}))$

Logistic regression - Cost function (cont.)

- The cost function for logistic regression is defined as follows:

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

- What does this cost function look like when $y = 1$?



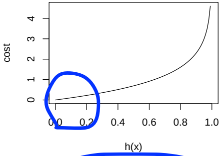
h_θ x

- When $y = 1$, this cost function has many good properties.
  - If $y = 1$ and $h_\theta(\mathbf{x}) = 1$, then Cost $= 0$.
  - If $y = 1$ and $h_\theta(\mathbf{x}) \to 0$, Cost $\to \infty$.
  - Captures intuition: if $y = 1$ (actual class) and $h_\theta(\mathbf{x}) = 0$ (predict $P(y = 1|\mathbf{x}; \theta) = 0$; absolutely impossible), we'll penalize the learning algorithm by a very large cost.

Logistic regression - Cost function (cont.)

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

- What does this cost function look like when $y = 0$?



- When $y = 0$, this cost function has many good properties.
  - If $y = 0$ and $h_\theta(\mathbf{x}) = 0$, then Cost $=0$.
  - If $y = 0$ and $h_\theta(\mathbf{x}) \to 1$, Cost $\to \infty$.
  - Captures intuition that
    $h_\theta(\mathbf{x}) = 1$ (predict $P(y = 1|\mathbf{x}; \theta) = 1$), absolutely impossible

Logistic regression - Cost function - rewriting

- Cost function

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

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

where $y = 0$ or 1

- The cost function can be rewritten as

$$\text{cost}(h_\theta(\mathbf{x}), y) = -y\log(h_\theta(\mathbf{x})) - (1-y)\log(1 - h_\theta(\mathbf{x}))$$

Logistic regression - Cost function - rewriting (cont.)

- Cost function

$$\begin{aligned} J(\theta) &= \frac{1}{n}\sum_{i=1}^{n} \text{cost}(h_\theta(\mathbf{x}^{(i)}), y^{(i)}) \\ &= -\frac{1}{n}(\sum_{i=1}^{n} y^{(i)}\log(h_\theta(\mathbf{x}^{(i)})) + (1-y^{(i)})\log(1-h_\theta(\mathbf{x}^{(i)}))) \end{aligned}$$

- To fit parameters θ:

$$min_\theta J(\theta)$$

- To make a prediction given new $\mathbf{x}$:
  output

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

  The meaning is $p(y=1|\mathbf{x};\theta)$

Logistic regression - Gradient Descent

- Cost function

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

- Goal

$$min_\theta J(\theta)$$

- Algorithm

  Repeat {

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

  (simultaneously update all θ_j)

  }
- To fit parameters θ:

$$min_\theta J(\theta)$$

Logistic regression - Gradient Descent

- Repeat {

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

  (simultaneously update all θ_j)

  }
- Since $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}^{(i)}$, we get
- Repeat {

  $$\theta_j = \theta_j - \alpha \sum_{i=1}^{n} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}^{(i)}$$

  (simultaneously update all θ_j)

  }
- Algorithm looks identical to linear regression!
- The difference is the definition of $h_\theta(\mathbf{x}^{(i)})$.

Python - logistic regression

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
```

References

- Chapter 4: Introduction to Data Mining (2nd Edition) by Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar
- Logistic Regression: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`