Artificial Intelligence

Taro Sekiyama

National Institute of Informatics (NII)
sekiyama@nii.ac.jp

Notice

- Next week is for programming
  - The room will be changed to:
    E-room, 3rd floor, Building 63
  - You need to bring your laptop
- (Common) Programming weeks are for making opportunities to receive questions
  - No new information will be provided
  - No attendance will be taken

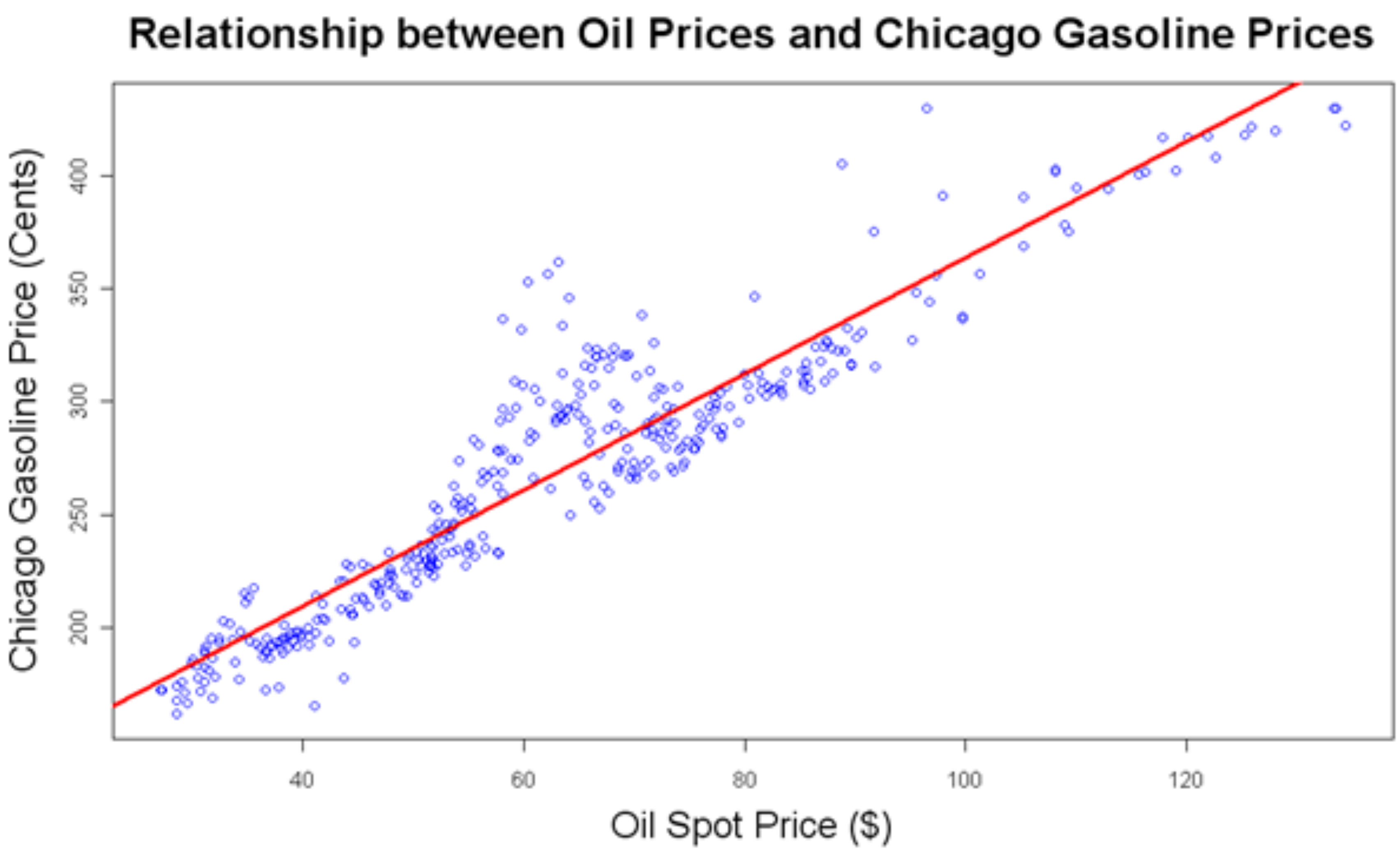
Agenda

- Linear regression: an ML algorithm for solving regression problems

Regression

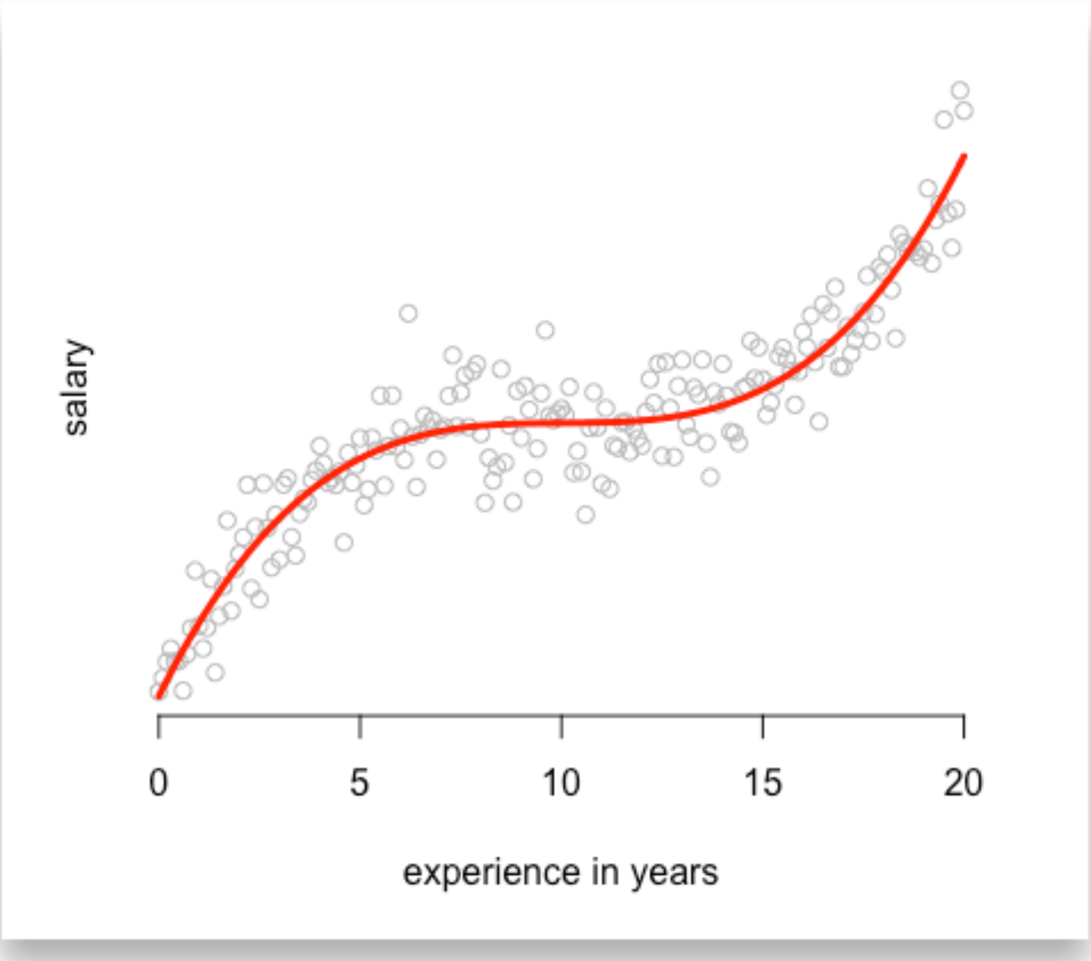
- Finding a relationship between properties of data points
  - Many relationships can be approximated by functions
- Goal: identifying functions approximating the relationship properly

Example



http://www.calculatinginvestor.com/2011/03/10/oil-and-gasoline-prices/

Example



https://www.freecodecamp.org/news/learn-how-to-improve-your-linear-models-8294bfa8a731/

Linear regression

Approximation by ***polynomial (linear)*** functions

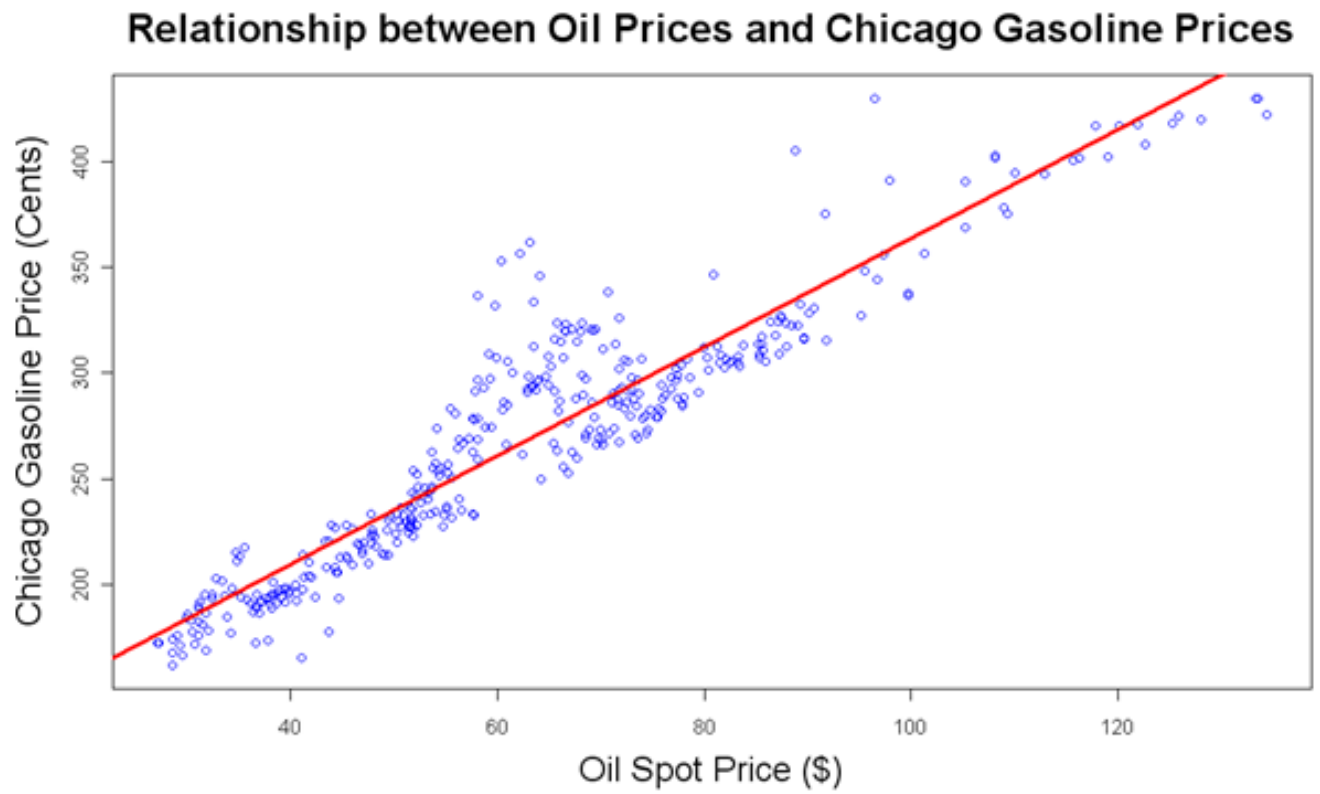
- Approximate function f over real numbers is expressed by:

$$f(x) = ax + b$$

  where a and b are parameters learned from a training dataset

  - a is called a *weight* or *coefficient*
  - b is called a *bias* parameter

Example



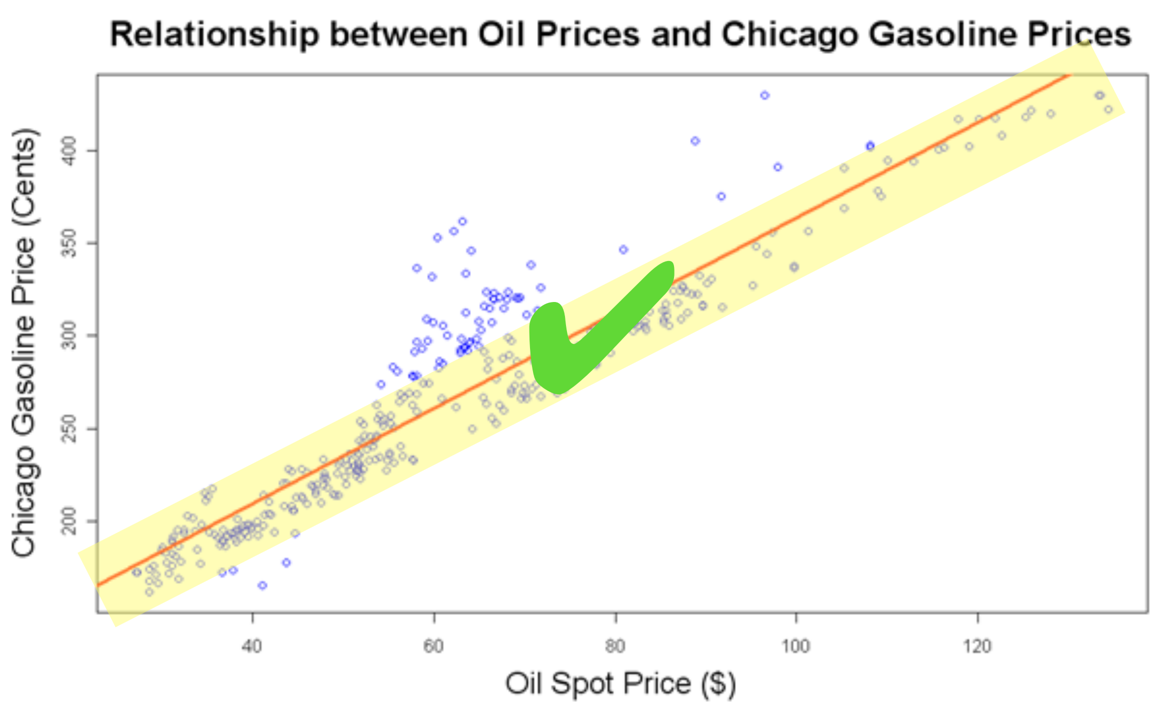
http://www.calculatinginvestor.com/2011/03/10/oil-and-gasoline-prices/

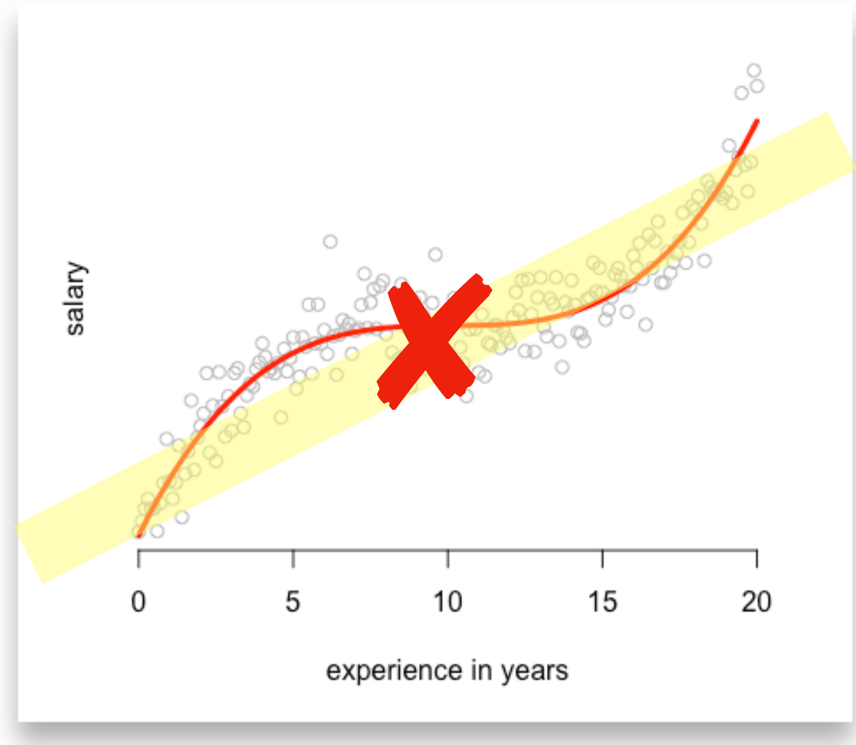
- Prediction (the red line) of the gasoline price for oil price x is approximated by

$$f(x) = 2.57x + 106.7$$

When is linear regression useful?

- It works more well as outputs are more likely to be proportional to inputs





- Important to take a careful look at data points

Extension to multiple features

- If an input is multiple features $(x_1, \ldots, x_n)$, then:

$$f(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + b$$

- Shorthand

$$f(\mathbf{x}) = \mathbf{a}\mathbf{x}^T + b$$

  - $\mathbf{x} = (x_1, \ldots, x_n)$
  - $\mathbf{a} = (a_1, \ldots, a_n)$
  - $(\ -\ )^T$: the transpose of vectors

Linear regression in Python

- scikit-learn provides a linear regression model

  `sklearn.linear_model.LinearRegression`

  https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Learning in simple linear regression

- Finding optimal values for parameters a and b of approximate function $f(x) = ax + b$
- Values $\hat{a}$ and $\hat{b}$ are optimal if $\hat{a}x_i + \hat{b}$ is as close to y_i as possible for training data points (x_i, y_i)
  - In other words: optimal values $\hat{a}$ and $\hat{b}$ make $(\hat{a}x_i + \hat{b}) - y_i$ as small as possible

Cost function

- Mathematical description of the difference between given and optimal parameters
- Major means: the mean squared error (MSE)

$$C(a, b) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathbf{T}} ((a x_i + b) - y_i)^2$$

where $\mathbf{T}$ is a set of n training data points (x_i, y_i)

- Values $\hat{a}$ and $\hat{b}$ are optimal if they minimize C

$$(\hat{a}, \hat{b}) = \arg \min_{(a,b) \in \mathbb{R}^2} C(a, b)$$

Optimization

- Finding $\hat{a}, \hat{b}$ s.t. $(\hat{a}, \hat{b}) = \arg\ \min_{(a,b)\in\mathbb{R}^2}\ C(a, b)$
- Visualization tells what $\hat{a}, \hat{b}$ minimize $C(a, b)$

Preprocess for visualization

- Expanding $C(a, b)$, which is equivalent to:

$$\frac{1}{n}\sum_{(x_i,y_i)}^{n}((ax_i + b) - y_i)^2 =$$

$$\frac{1}{n}\sum_{(x_i,y_i)}^{n}(a^2x_i^2 + 2abx_i + b^2 - 2ax_iy_i - 2by_i + y_i^2)$$

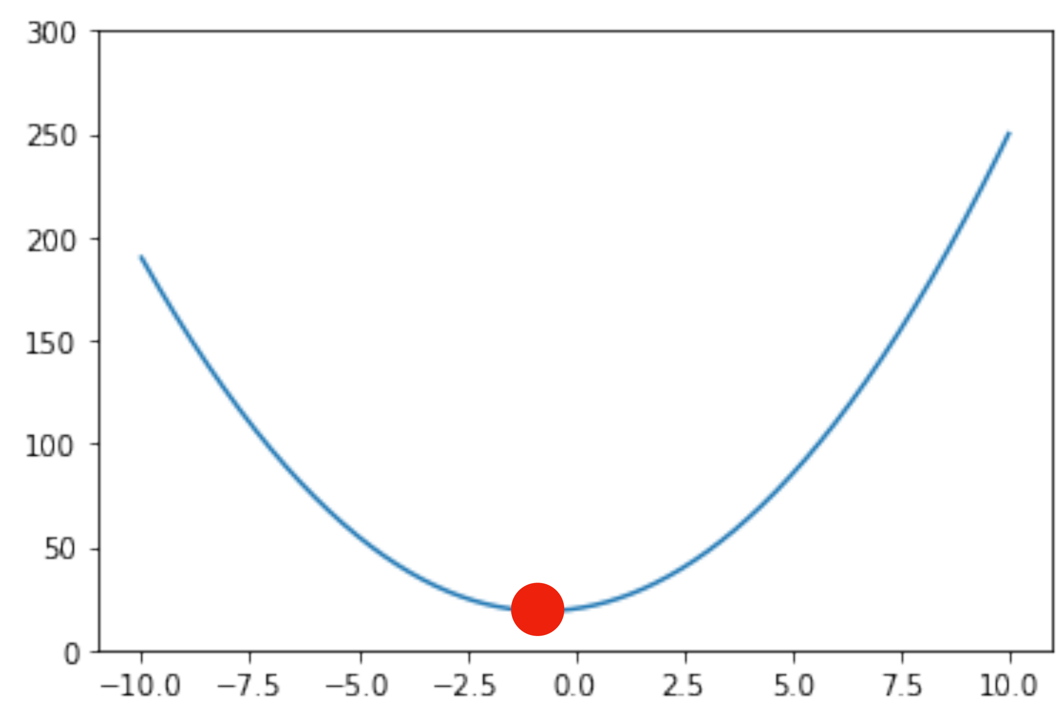
- Viewing $C(a, b)$ as a function with one parameter by fixing either of a and b
  - $C^b(a) \equiv C(a, b)$ (only a is varying; b is fixed)
  - $C^a(b) \equiv C(a, b)$ (only b is varying; a is fixed)

Characteristics of $C^b(a)$, $C^a(b)$

- $C^b(a)$ and $C^a(b)$ are:
  - Quadratic: the highest degrees are 2
  - Convex: the coefficients of a^2 and $b^2 \geq 0$

$$C(a,b) = \frac{1}{n} \sum_{(x_i,y_i)}^{n} (a^2 x_i^2 + 2abx_i + b^2 - 2ax_iy_i - 2by_i + y_i^2)$$

- The parameters that minimizes $C^b(a)$, $C^a(b)$ are found by looking at their ***derivatives***



Derivatives for optimization

- For a convex quadratic function $F(x)$,
  $\hat{x} = \arg\ \min_x\ F(x)$ if and only if $d\dfrac{F(\hat{x})}{dx} = 0$
- $\hat{a}$ and $\hat{b}$ are optimal if and only if:
  - $d\dfrac{C^{\hat{b}}}{da}(\hat{a}) = 0$ and
  - $d\dfrac{C^{\hat{a}}}{db}(\hat{b}) = 0$

Methods

Two methods to find $\hat{a}$ and $\hat{b}$ satisfying

$d\dfrac{C^{\hat{b}}}{da}(\hat{a}) = 0$ and $d\dfrac{C^{\hat{a}}}{db}(\hat{b}) = 0$

1. Solving as simultaneous equation problem
2. Gradient descent

1. Simultaneous equation solving

(1) $d\frac{C^{\hat{b}}}{da}(\hat{a}) = 0$

(2) $d\frac{C^{\hat{a}}}{db}(\hat{b}) = 0$

$$C(a, b) = \frac{1}{n} \sum_{(x_i, y_i)}^{n} (a^2 x_i^2 + 2abx_i + b^2 - 2ax_i y_i - 2by_i + y_i^2)$$

1. Simultaneous equation solving

(1) $d\frac{C^{\hat{b}}}{da}(\hat{a}) = 0$

(2) $\hat{b} = \frac{1}{n}\sum_{(x_i,y_i)}^{n} y_i - \frac{\hat{a}}{n}\sum_{(x_i,y_i)}^{n} x_i$

$$C(a,b) = \frac{1}{n}\sum_{(x_i,y_i)}^{n}(a^2x_i^2 + 2abx_i + b^2 - 2ax_iy_i - 2by_i + y_i^2)$$

1. Simultaneous equation solving

(1) $d\dfrac{C^{\hat{b}}}{da}(\hat{a}) = 0$

(2) $\hat{b} = \overline{y} - \hat{a}\overline{x} \quad (\overline{x} = \dfrac{1}{n}\Sigma^n x_i, \overline{y} = \dfrac{1}{n}\Sigma^n y_i)$

$$C(a, b) = \frac{1}{n}\sum_{(x_i, y_i)}^{n}(a^2x_i^2 + 2abx_i + b^2 - 2ax_iy_i - 2by_i + y_i^2)$$

1. Simultaneous equation solving

(1) $\hat{a} = \dfrac{\Sigma^n (x_i y_i) - n\overline{x}\overline{y}}{\Sigma^n x_i^2 - n\overline{x}^2}$

(2) $\hat{b} = \overline{y} - \hat{a}\overline{x} \quad (\overline{x} = \dfrac{1}{n}\Sigma^n x_i, \overline{y} = \dfrac{1}{n}\Sigma^n y_i)$

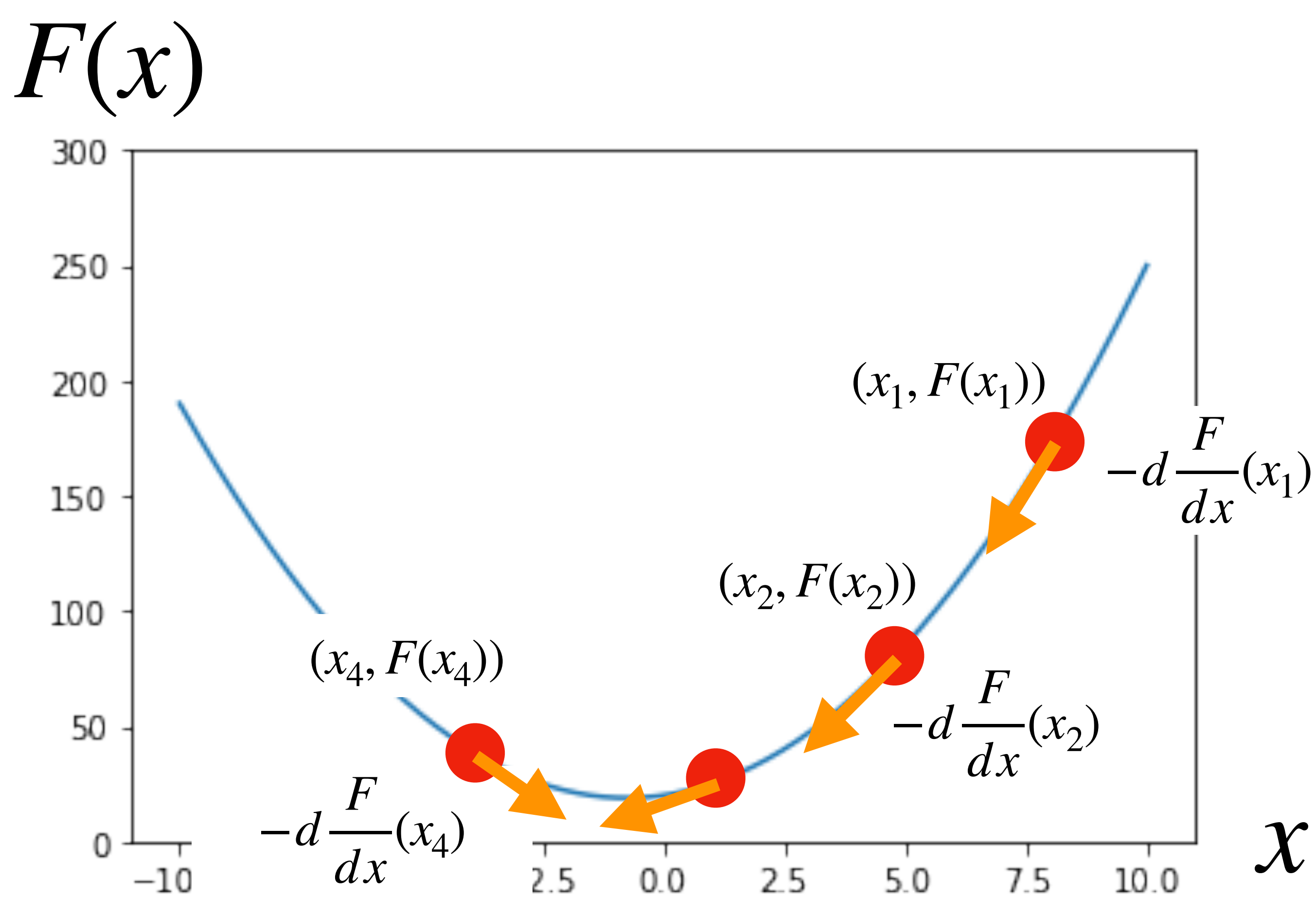
$$C(a, b) = \frac{1}{n} \sum_{(x_i, y_i)}^{n} (a^2 x_i^2 + 2abx_i + b^2 - 2ax_i y_i - 2by_i + y_i^2)$$

1. Simultaneous equation solving

- **Strong point**
  - Production of the exactly optimal values
- **Weak point**
  - Poor scalability, especially w.r.t. # of features
    - Extension to multiple features gives rise to operations on matrices, whose run-time cost depends on # of features

2. Gradient descent

- Approaching optimal values gradually



2. Gradient descent

- Input
  - Initial parameters $\hat{a}$, $\hat{b}$
  - Learning rate $\eta \in \mathbb{R}$
    - Determining how parameters change by one time update
  - The number n of updating the parameters
- Algorithm: repeat the following update n times
  - $\hat{a} := \hat{a} - \eta \cdot d\dfrac{C^{\hat{b}}}{da}(\hat{a})$ $\left(d\dfrac{C^{\hat{b}}}{da}(\hat{a}) = \dfrac{1}{n}\Sigma^n x_i(ax_i + b - y_i) \right)$
  - $\hat{b} := \hat{b} - \eta \cdot d\dfrac{C^{\hat{a}}}{db}(\hat{b})$ $\left(d\dfrac{C^{\hat{a}}}{db}(\hat{b}) = \dfrac{1}{n}\Sigma^n (ax_i + b - y_i) \right)$

2. Gradient descent

- **Strong point**
  - Extensible to an algorithm (relatively) scalable w.r.t. the numbers of features
    - Known as *stochastic gradient descent*
  - Applicable to non-convex cost functions (if they are differentiable)
    - Ex: cost functions of neural networks
- **Weak point**
  - Goodness of $\hat{a}$ and $\hat{b}$ depends on the choice of learning rate η and the number of updates n

Evaluation

- Goodness of estimated $\hat{a}$ and $\hat{b}$ are evaluated by the cost function

$$C(a,b) = \frac{1}{n} \sum_{(x_i,y_i)\in\mathbf{E}} ((ax_i + b) - y_i)^2$$

for test dataset $\mathbf{E}$

Learning in linear regression with multiple features

- Cost function

$$C(a,b) = \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{T}} ((\mathbf{a}\mathbf{x}_i^T + b) - y_i)^2$$

- Both of (1) simultaneous equations solving and (2) gradient descent can be extended

Programming assignment (common)

- Two kinds of assignment
  - Mandatory: need to be solved to achieve the full score
  - Optional: NOT need to be solved, but evaluated positively
- Submitted by MyWaseda
  - ***I'll respond if I accept the submission***

1st programming assignment

- Deadline: 11/14
- Topics
  - K-nearest neighbors
  - Linear regression

1st programming assignment

1. Implement K-nearest neighbors classifier

- Submission: implementation code and test
- Template of implementation is found at https://github.com/skymountain/waseda-AI-lecture/blob/master/programming1/knn.py
  - You may fill the unimplemented parts there
  - It contains the minimum test
- Minimum requirements
  - The test in the script above has to be passed
  - DO NOT USE `sklearn.neighbors.KNeighborsClassifier`

1st programming assignment

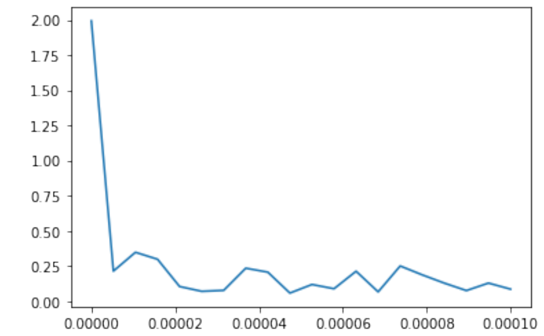
1 (optional). Implement K-nearest neighbors ***regression***

- Submission: implementation code and test
- Two choices on prediction of continuous values
  - The average of the outputs of the K nearest neighbors
  - The weighted average of the outputs of the K nearest neighbors
    - Weights are the inverses of the distances

1st programming assignment

2. Plot the relationship between learning rate η and the goodness of the estimated $\hat{a}$ and $\hat{b}$ by gradient descent

- Submission
  - Graph
  - Code used to generate test data points and plot the graph
  - Remark: Jupyter-notebook is fine
    - VSCode is not tested in my laptop



Example of graphs

1st programming assignment

2. Plot the relationship between learning rate η and the goodness of the estimated $\hat{a}$ and $\hat{b}$ by gradient descent

- Goodness: $C(\hat{a}, \hat{b}) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathbf{E}} ((\hat{a} x_i + \hat{b}) - y_i)^2$
- Minimum requirements
  - The number of updates n is $\geq$ 10,000
  - Try all the learning rates produced by `` `numpy.linspace(0., 0.0005., 100)` ``
- Implementation of gradient descent is found at: https://github.com/skymountain/waseda-AI-lecture/blob/master/lecture3/simple%20linear%20regression.ipynb
  - Code to generate data points is also found

1st programming assignment

2 (optional). Extend linear regression to an arbitrary number of features

- Submission: implementation code and test
- Available dataset:
  `sklearn.datasets.load_boston`
- Either of
  (1) simultaneous equations solving and
  (2) gradient discent
  is fine