Introduction to Machine Learning Linear Regression

Prof. Chang-Chieh Cheng
Information Technology Service Center
National Chiao Tung University

Overview

- The main concept of error-based models
  - Given a set of data instances $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ and each data instance $\mathbf{x}_i$ has a target y_i
  - Design an error-based model *M* with a set of parameters **w**.
  - *w* satisfies the following equation

$$\mathbf{w} = \arg\min_{\mathbf{w}'} \sum_{i=1}^{m} D(y_i, M_{\mathbf{w}'}(\mathbf{x}_i))$$

    - where *D(y, y_m)* is the error of real target *y* and predicted target y_m.

  - Therefore, we can predict a query **q** by the model M with *w*.

$$y_q = M_{\mathbf{w}}(\mathbf{q})$$

Overview

- How to choose a model to fit the training data?
  - Line
  - Polynomial
  - Nonlinear model
- How to decide *w*?
  - Gradient descent

Linear Model

- Single-variable linear equation
  - $\mathbf{x} = \{x\}$

$$y = w_0 + w_1 x$$

- Multi-variable linear equation
  - $\mathbf{x} = \{x_0, x_1, x_2, \ldots, x_n\}$, where $x_0 = 1$

$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

$$= \sum_{j=0}^{n} w_j x_j$$

$$= \mathbf{w} \cdot \mathbf{x}$$

Linear Model

- Example
  - A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.
    - Size(x) → Rental price(y)

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

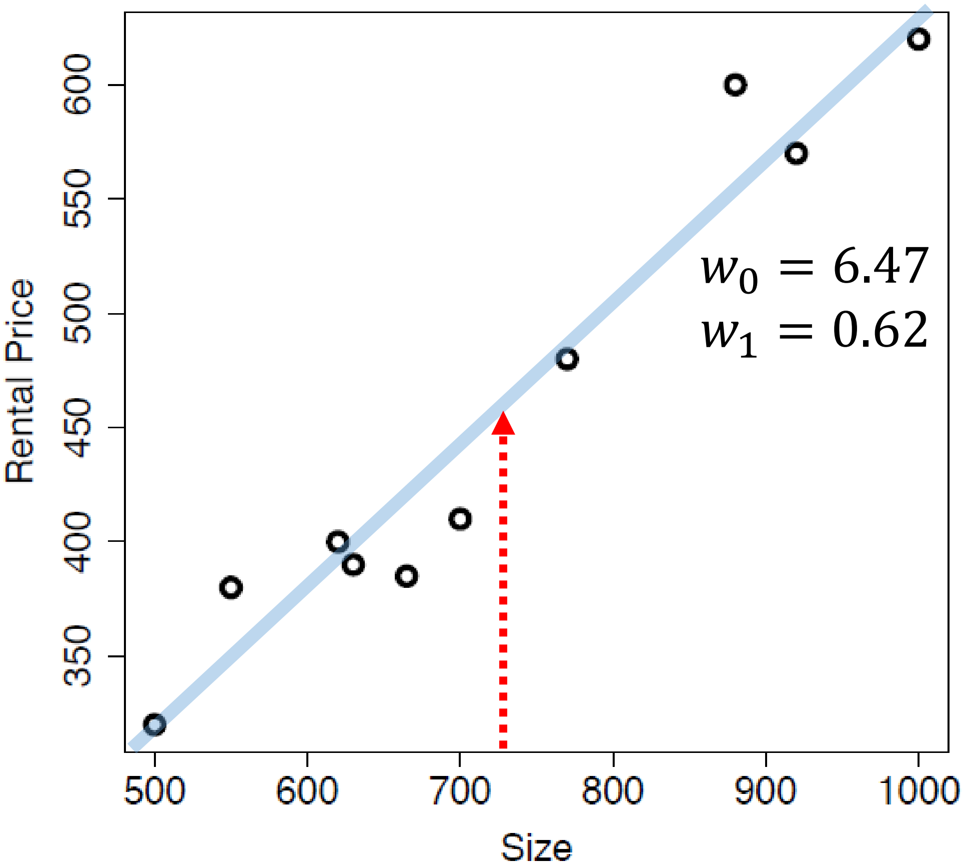
Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Model

- Example
  - Size(x) - Rental price(y)

$$y = w_0 + w_1 x$$
$$w_0 = ?$$
$$w_1 = ?$$

- Query: x = 730, y = ?

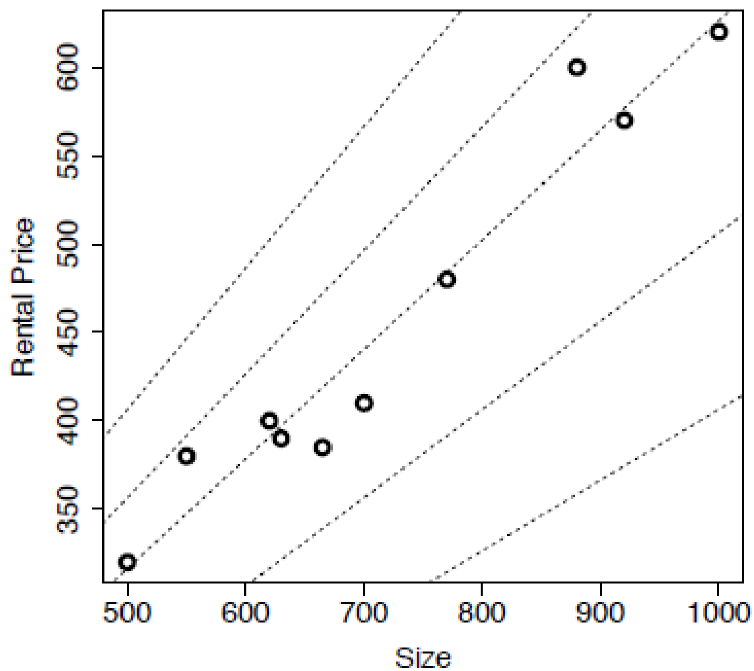


Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Model

- L_2-norm minimization
- Let $\mathbf{w} = [w_0, w_1]$

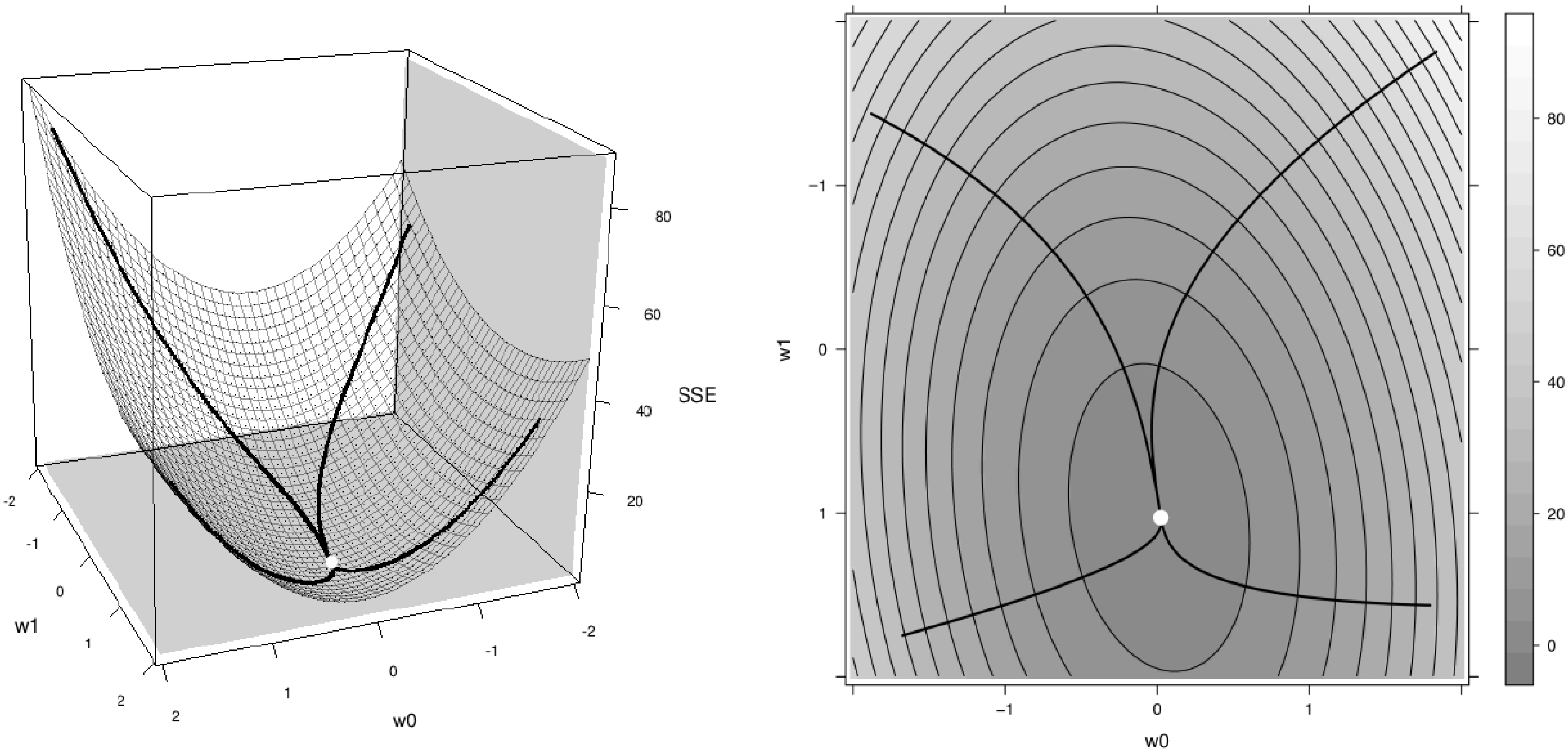
$$\mathbf{w} = \arg\min_{\mathbf{w}'} \frac{1}{2} \sum_{i=1}^{m} (y_i - (w'_0 + w'_1 x_i))^2$$



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Model

- Error surface



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Finding a **w** such that its error is minimum

Linear Model

- Least square method

$$\mathrm{Ax} = \mathrm{b},$$

A is an *m* x *n* matrix
x is a n-d vector
b is a m-d vector
If A and b are known, what is x?

$$\mathrm{Ax} = \mathrm{b}$$
$$\mathrm{A^T\,Ax} = \mathrm{A^T b}$$

$\mathrm{A^T\,A}$ is an *n* x *n* matrix and is invertible.
Then,

$$(\mathrm{A^T\,A})^{-1}(\mathrm{A^T\,A})\mathrm{x} = (\mathrm{A^T\,A})^{-1}\mathrm{A^T b}$$
$$\mathrm{x} = (\mathrm{A^T\,A})^{-1}\mathrm{A^T b}$$

Linear Model

- Least square method
- `numpy.linalg.lstsq(A, b)`
  - Returns:
    - `x`
    - residuals : `|b - A*x|`^{2}
    - rank : `int`
    - Singular values of `A`

```
import numpy as np
A = np.matrix([[1, 2], [2, 3], [3, 4]])  # 3 x 2 matrix
b = np.array([1, 5, 6])          # 3 x 1 array
R = np.linalg.lstsq(A, b, rcond = None)
x = R[0]
print(x)         # [ 3.5 -1. ]
print(A * x.reshape(2, 1))
'''
    [[1.5]
     [4. ]
     [6.5]]
'''
```

Linear Model

- Given a function $f(x) = w_0 + w_1 x$
  - if $f(2) = 10; f(4) = 50; f(8) = 75; f(10) = 20$
  - What are w_0 and w_1?

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 8 \\ 1 & 10 \end{bmatrix} \quad \mathrm{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathrm{y} = \begin{bmatrix} 10 \\ 50 \\ 75 \\ 20 \end{bmatrix}$$

$$\mathrm{Aw} = \mathrm{y}$$

Linear Model

- Given a function $f(x) = aw_2{}^2 + bw_1 + w_0$
- if $f(2) = 10; f(4) = 50; f(8) = 75; f(10) = 20$
  - What are w_0, w_1, and w_2?

Linear Model

- If the error surface has the minimum, then

$$\mathbf{w} = \arg\min_{\mathbf{w}'} \frac{1}{2}\sum_{i=1}^{m}(y_i - (w'_0 + w'_1 x_i))^2$$

$$\Rightarrow \frac{d\frac{1}{2}\sum_{i=1}^{m}(y_i - (w_0 + w_1 x_i))^2}{d\mathbf{w}} = \mathbf{0}$$

Linear Model

- Multi-variable linear model
  - $\mathbf{x} = \{x_0, x_1, x_2, \dots, x_n\}$, where $x_0 = 1$
  - $\mathbf{w} = \{w_0, w_1, w_2, \dots, w_n\}$

$$\mathbf{w} = \arg\min_{\mathbf{w}'} \frac{1}{2} \sum_{i=1}^{m} (y_i - \mathbf{w}' \cdot \mathbf{x})^2$$

$$\Rightarrow \frac{d \frac{1}{2} \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x})^2}{d\mathbf{w}} = \mathbf{0}$$

Gradient Descent

$$\frac{\partial \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}\cdot\mathbf{x})^2}{\partial w_j}$$

$$\frac{\partial f(x,y)g(x,y)}{\partial x}$$

$$= \frac{\partial f(x,y)}{\partial x} g(x,y) + f(x,y)\frac{\partial g(x,y)}{\partial x}$$

- If $m = 1$

$$\frac{\partial \frac{1}{2}(y - \mathbf{w}\cdot\mathbf{x})^2}{\partial w_j} = (y - \mathbf{w}\cdot\mathbf{x})\frac{\partial (y - \mathbf{w}\cdot\mathbf{x})}{\partial w_j}$$

$$= (y - \mathbf{w}\cdot\mathbf{x})\frac{\partial\left(y - w_0x_0 - w_1x_1 - \cdots - w_jx_j - \cdots - w_nx_n\right)}{\partial w_j}$$

$$= -x_j(y - \mathbf{w}\cdot\mathbf{x})$$

Gradient Descent

- For $m > 1$

$$\frac{\partial \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}\cdot\mathbf{x})^2}{\partial w_j}$$

$$= \sum_{i=1}^{m} \frac{\partial \frac{1}{2}(y_i - \mathbf{w}\cdot\mathbf{x})^2}{\partial w_j}$$

$$= \sum_{i=1}^{m} -x_{ij}(y_i - \mathbf{w}\cdot\mathbf{x}_i)$$

Gradient Descent

- To minimize error

$$w_j' = -\frac{\partial \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}\cdot\mathbf{x})^2}{\partial w_j}$$

- ➔ $$w_j' = \sum_{i=1}^{m} x_{ij}(y_i - \mathbf{w}\cdot\mathbf{x}_i)$$

- Therefore, increasing w_j by w'_j can let total error to approach zero
  - where α is an adjustment factor called **learning rate**

$$w_j = w_j + \alpha w_j'$$

Gradient Descent

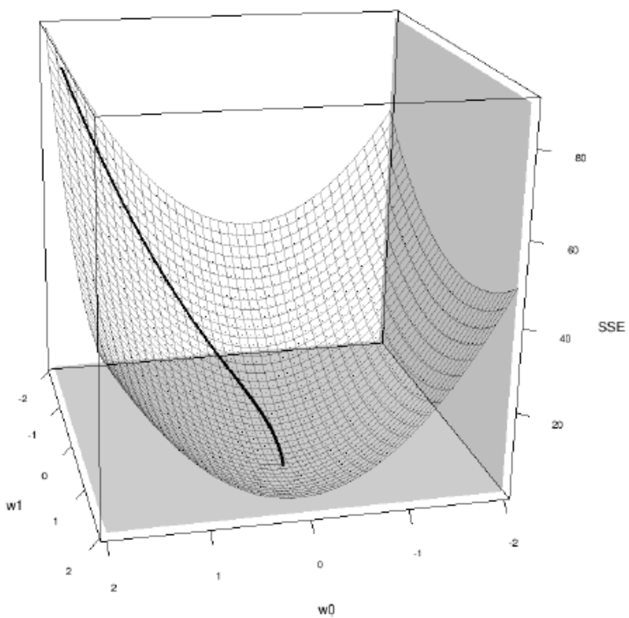
- Algorithm
  - **Data set:** $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$
    - each $\mathbf{x}_i = \{x_{i0}, x_{i1}, x_{i2}, \ldots, x_{in}\}$, where $x_{i0} = 1$
  - $\mathbf{w} = \{w_0, w_1, w_2, \ldots, w_n\}$

1. Initializing **w**
2. *k* = 0
3. do
4. for *j* = 0 to *n*
5. $$w'_j = \sum_{i=1}^{m} x_{ij}(y_i - \mathbf{w} \cdot \mathbf{x}_i) = 0$$
6. $$w_j = w_j + \alpha w'_j$$
7. *k* = *k* + 1
8. until all w'_j < a small value or *k* ≥ the maximum iterations

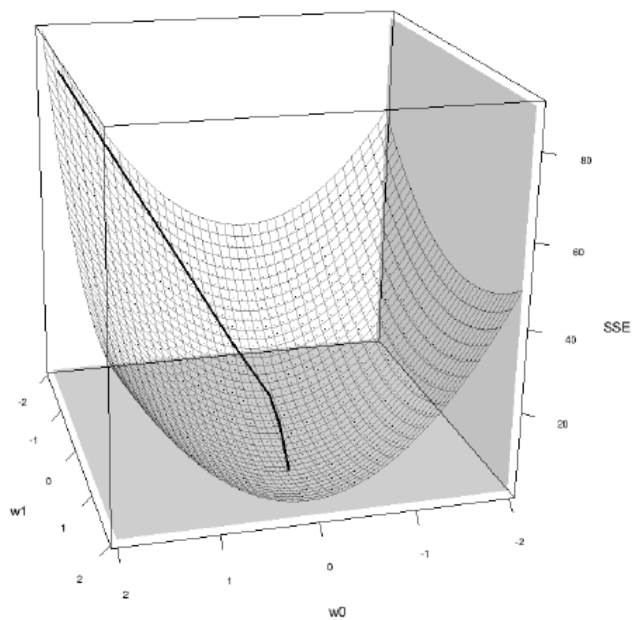
Gradient Descent

- Learning rate α is difficult to choose

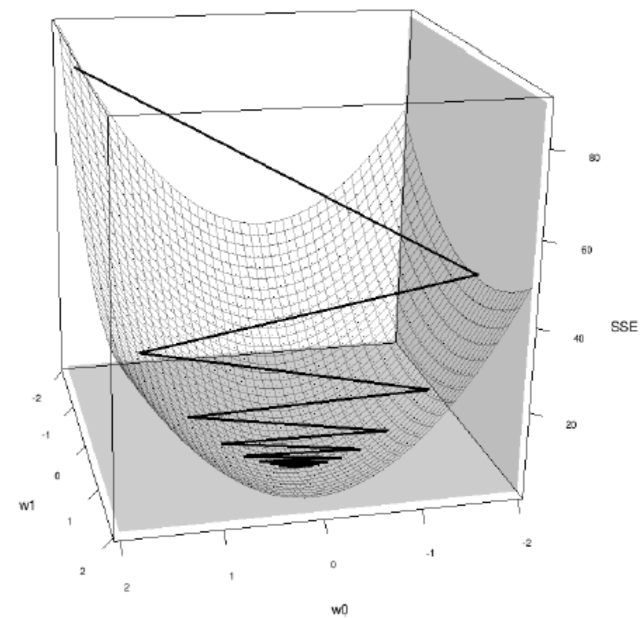
α is too small



Appropriate α



α is too large



- A typical range of α is [0.00001, 10]

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Gradient Descent

- Initial value of $\mathbf{w}$
  - choosing random value uniformly from the range [-0.2, 0.2]

Gradient Descent

- Example
  - A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.
    - x_1: Size, x_2: Floor, x_3: Broadband rate,
    - $w_0 + w_1x_1 + w_2x_2 + w_3x_3$ = Rental price

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Gradient Descent

- Example
  - α = 0.00000002
  - $\mathbf{w} = [-0.146 \quad 0.185 \quad -0.044 \quad 0.119]$
  - SSE = 1067571.59

$$w_1' = \sum_{i=1}^{m} x_{i1}(y_i - \mathbf{w} \cdot \mathbf{x}_i) = 2412074$$

$$\alpha w_1' = 0.23324148$$

$$w_1 = w_1 + \alpha w_j' = 0.185 + 0.23324148 = 0.233$$

Gradient Descent

- Example
  - α = 0.00000002
  - $\mathbf{w} = [-0.146 \quad \mathbf{0.233} \quad -0.044 \quad 0.119]$
  - SSE = 886723.04

$$w_1' = \sum_{i=1}^{m} x_{i1}(y_i - \mathbf{w} \cdot \mathbf{x}_i) = 2195616.08$$

$$w_1 = w_1 + \alpha w_j' = 0.27691232$$

  - ….
  - After 100 iterations
    - $\mathbf{w} = [-0.1513 \quad 0.6270 \quad -0.1781 \quad 0.0714]$
    - SSE = 2913.5

Gradient Descent

- Which one is good?
  - Each iteration only updates w_j
  - Each iteration updates **w**

Initializing **w**

$k = 0$, $\mathbf{w}' = [0]$

do

  for $j = 0$ to n

$$w_j' = \sum_{i=1}^{m} x_{ij}(y_i - \mathbf{w} \cdot \mathbf{x}_i) = 0$$

$$\mathbf{w} = \mathbf{w} + \alpha \mathbf{w}'$$

  $k = k + 1$

until $|\mathbf{w}'|$ < a small value or $k \geq$ the maximum iterations

Gradient Descent

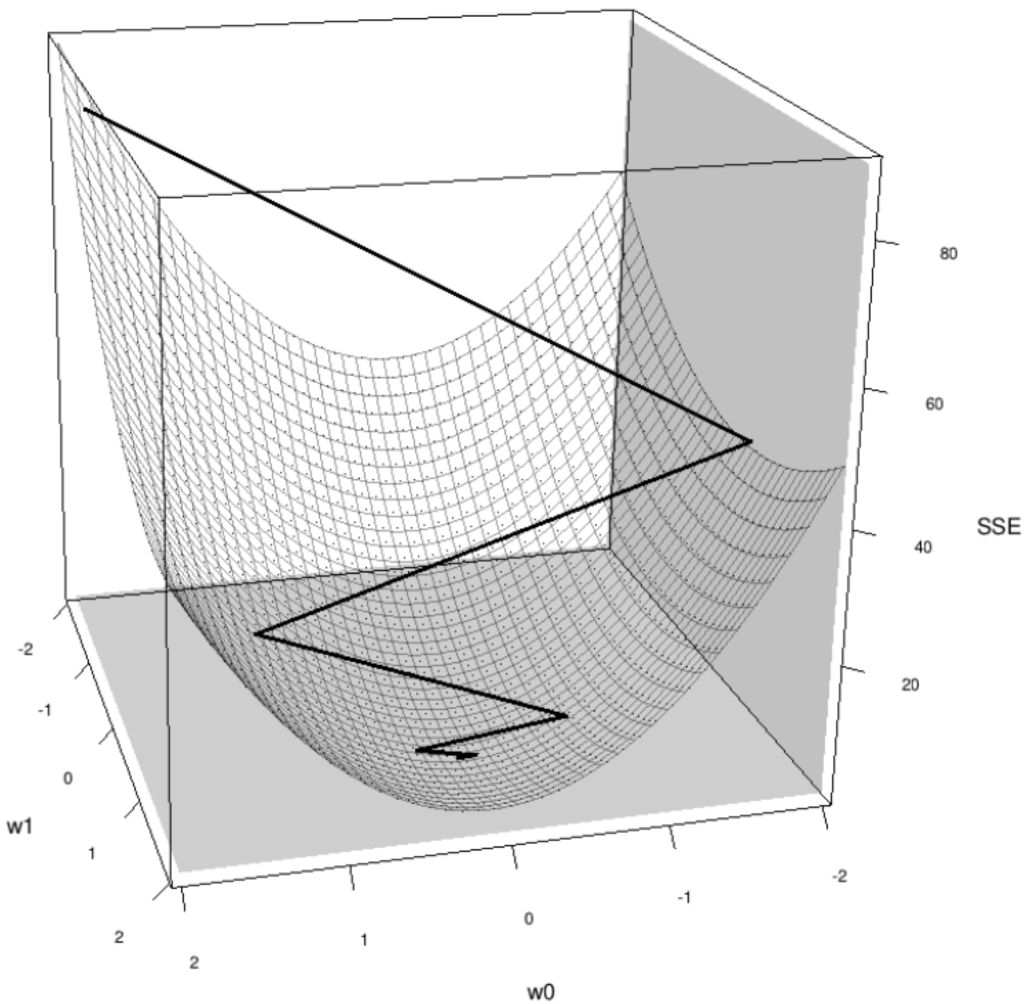
- Learning rate decay

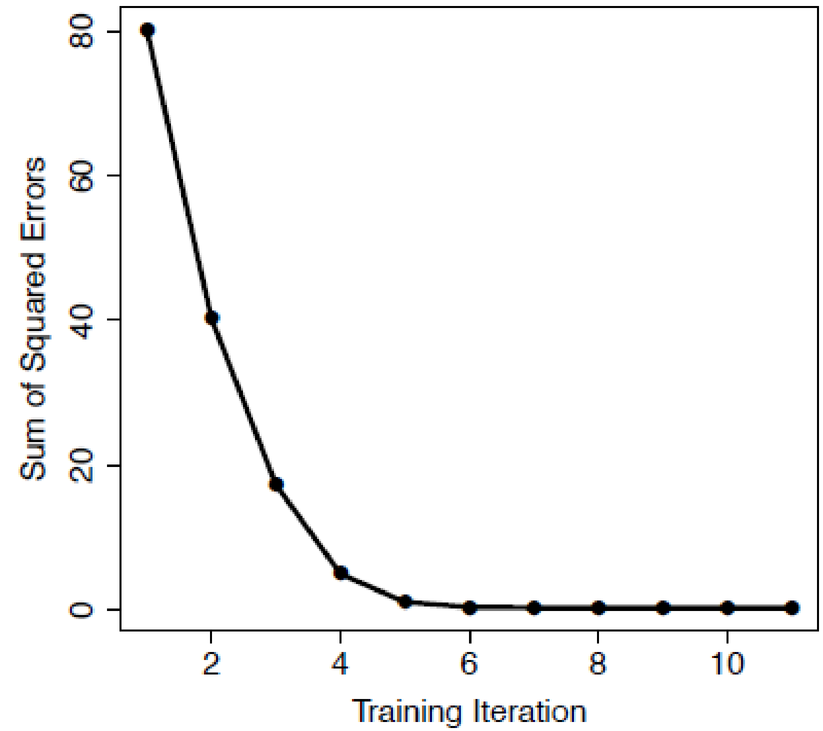
$$\alpha_k = \alpha_0 \frac{c}{c + k}$$

- where *k* is the number of iteration and *c* is a constant number

Gradient Descent

- Learning rate decay
  - $\alpha_0 = 0.18$, $C = 10$

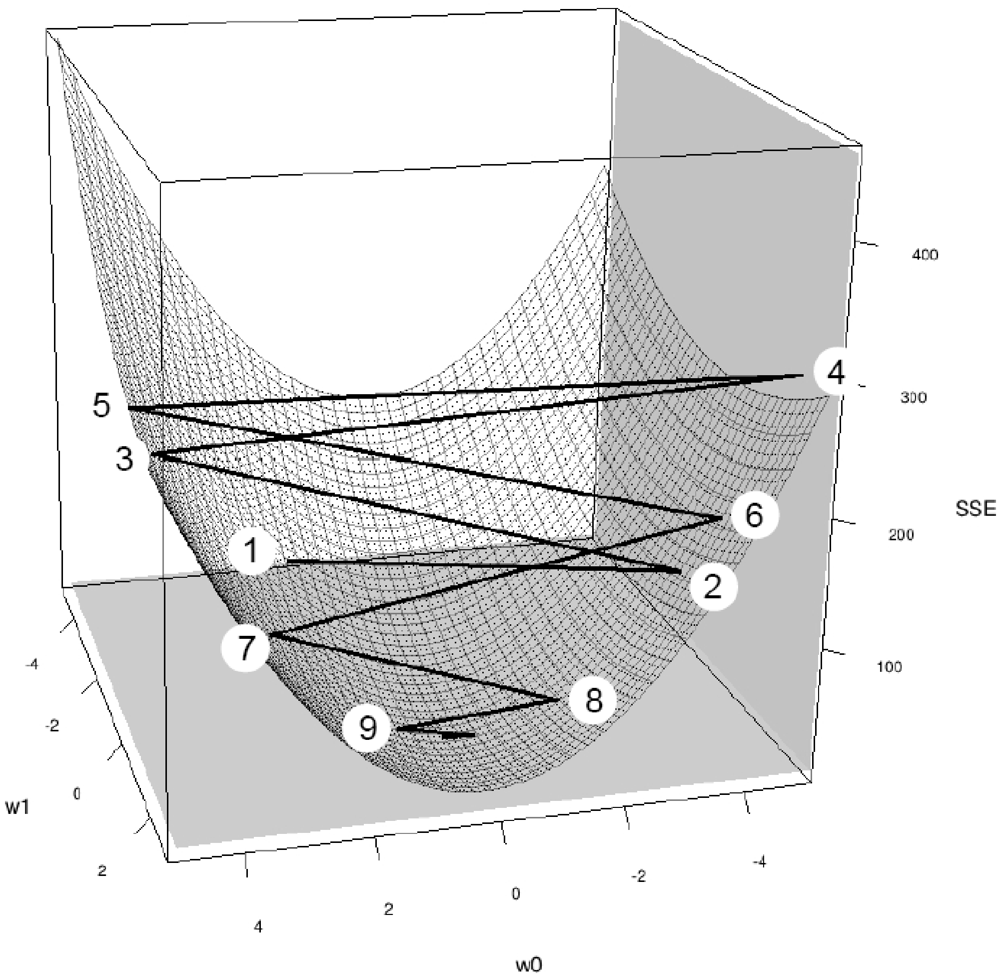


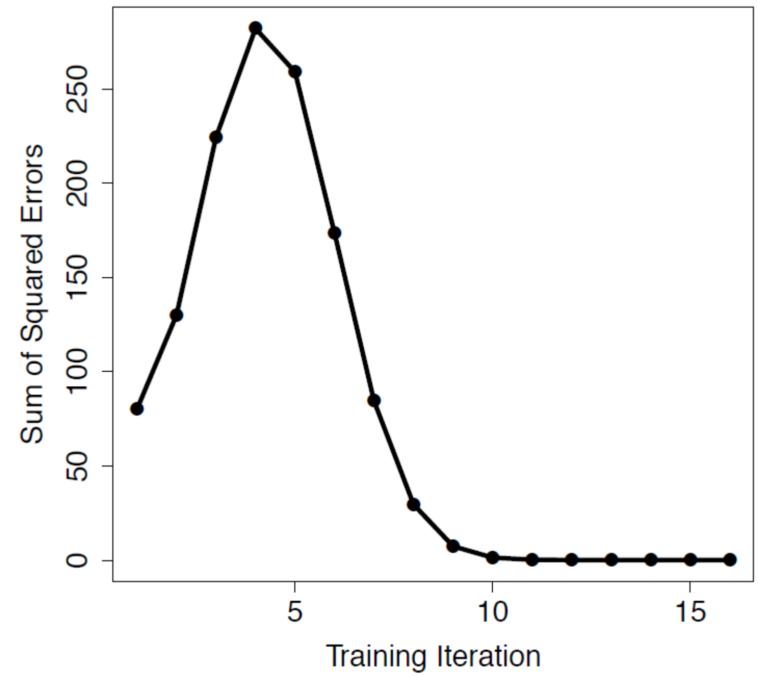


Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Gradient Descent

- Learning rate decay
  - $\alpha_0 = 0.25$, $C = 100$





Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Gradient Descent

- Momentum

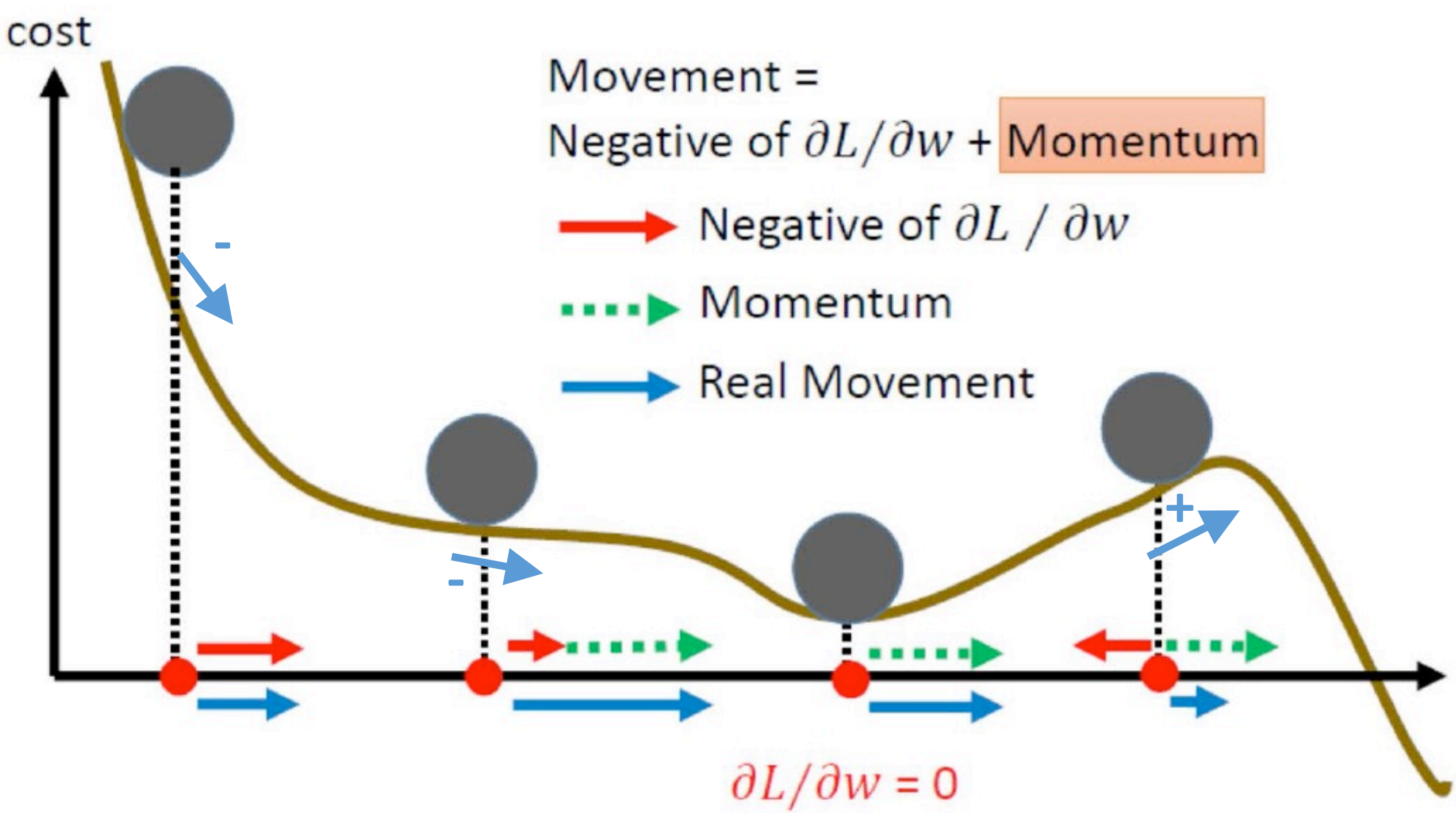


Figure designed by Xinxin Zuo

Gradient Descent

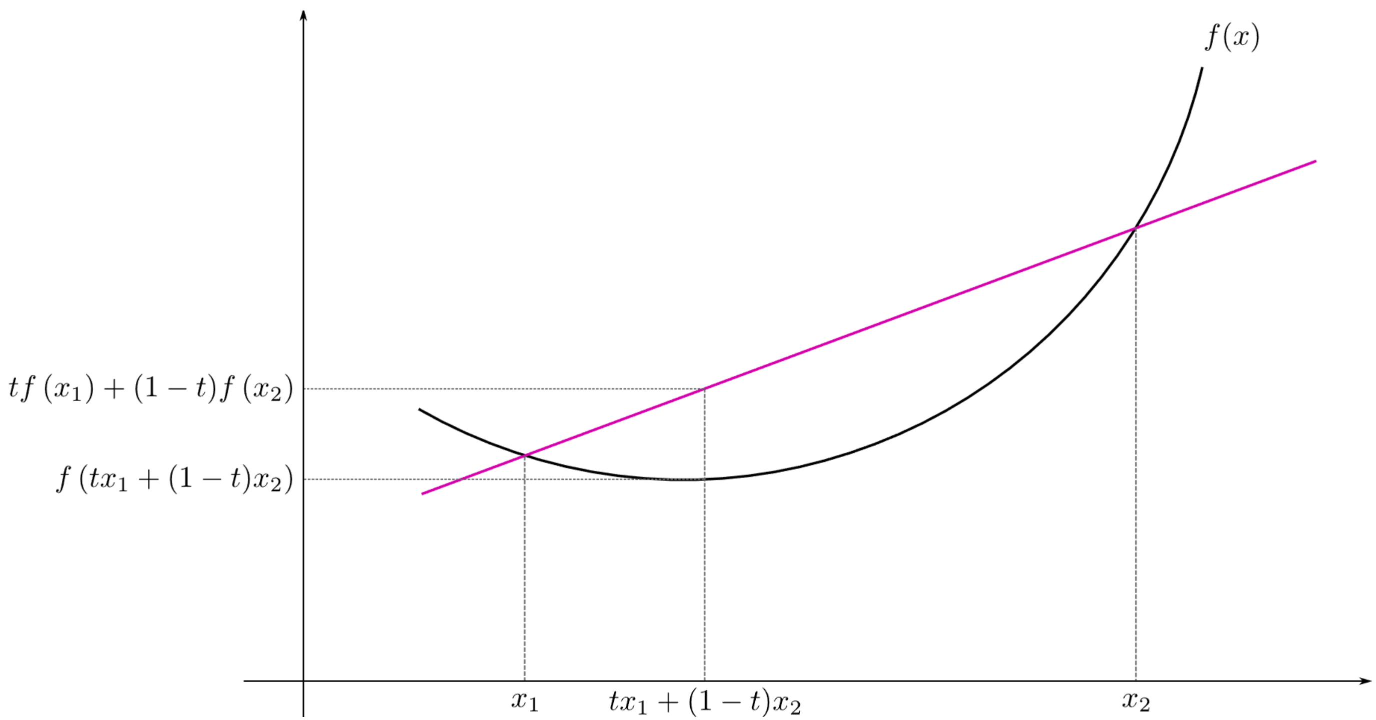
- Adam: A Method for Stochastic Optimization
  - D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 2015.
- Algorithm

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

$\alpha = 0.001$,
$\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Gradient Descent

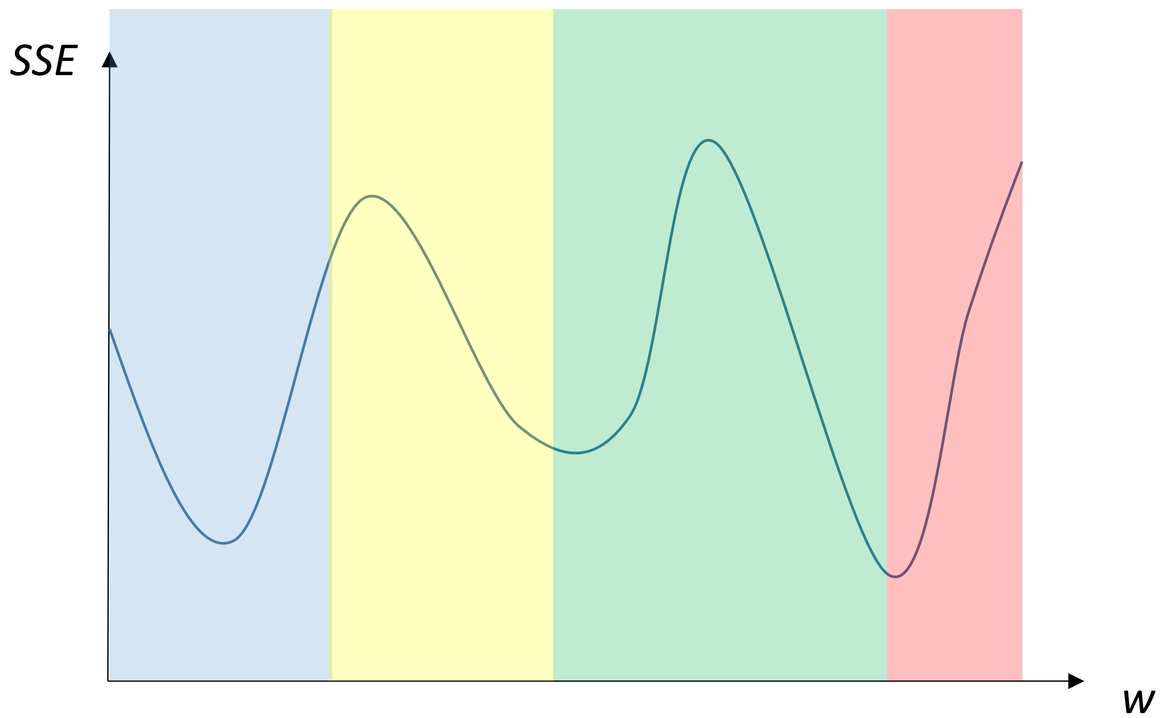
- Convex
  - A real-valued function defined on an n-dimensional interval is called convex if the line segment between any two points on the graph of the function lies above or on the graph, in a Euclidean space of at least two dimensions.



https://en.wikipedia.org/wiki/Convex_function

Gradient Descent

- If the error surface is convex, the gradient descent can find the unique answer
- Otherwise, the answer may not be the best.



Gradient Descent

- Categorical Descriptive Features
  - Gradient descent requires that all value must be continuous
    - ➔Differentiable
  - For example, ENERGY RATING has three levels: A, B, and C
    - ➔ Create three binary features to stand for three levels

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING A	ENERGY RATING B	ENERGY RATING C	RENTAL PRICE
1	500	4	8	0	0	1	320
2	550	7	50	1	0	0	380
3	620	9	7	1	0	0	400
4	630	5	24	0	1	0	390
5	665	8	100	0	0	1	385
6	700	4	8	0	1	0	410
7	770	10	7	0	1	0	480
8	880	12	50	1	0	0	600
9	920	14	8	0	0	1	570
10	1 000	9	24	0	1	0	620

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Error functions

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2$$

$$\text{mean squared error} = \frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}$$

$$\text{root mean squared error} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}}$$

$$\text{mean absolute error} = \frac{\sum_{i=1}^{n} abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n}$$

Error functions

- Continuous targets

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}}$$

$$\text{total sum of squares} = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2$$

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2$$

Linear Model in Python

- Linear regression in sklearn
  - https://github.com/jameschengcs/ml/blob/master/linear_regr.py

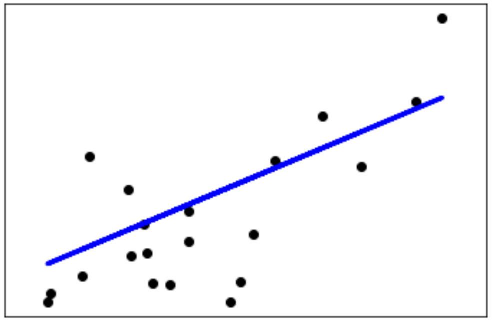
```
import numpy as np
from sklearn import datasets, linear_model

diabetes = datasets.load_diabetes()   # Load the diabetes dataset
diabetes_X = diabetes.data[:, np.newaxis, 2]
                      # Get the data[2] as an n×1 matrix

diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)

print('Coefficients: \n', regr.coef_, regr.intercept_ )
print(diabetes_X_test[0], diabetes_y_pred[0])
```



Linear Model in Python

- Linear model in tensorflow
  - Example:
    - https://github.com/jameschengcs/ml/blob/master/gd_tf.py