Deep learning
텐서플로로 시작하는 딥러닝 1장

Ha-Jin Yu, Dept. of Computer Science, University of Seoul
서울시립대학교 컴퓨터과학부 유하진

2019.

HJYU@UOS.AC.KR

CHAPTER 1 텐서플로 입문

1.1.1 머신러닝의 개념

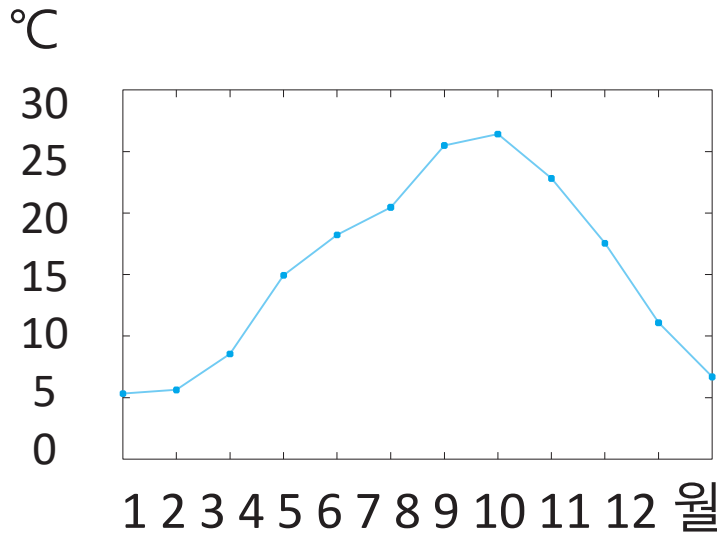


그림 1-3 월별 평균 기온 데이터

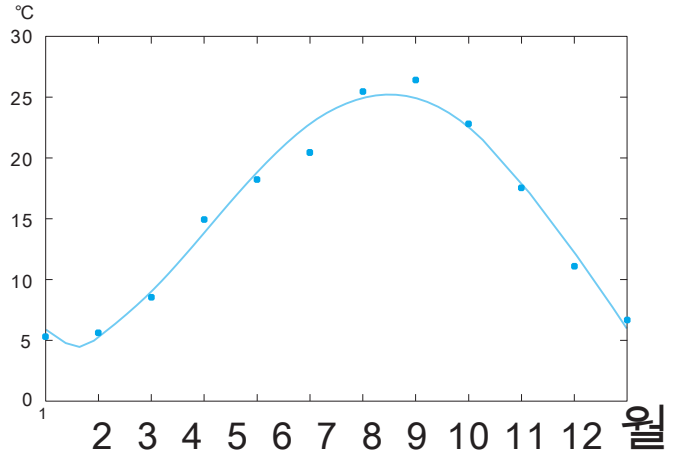


그림 1-4 완만한 곡선으로 예측한 평균 기온

- 데이터의 모델화 (modeling) : 주어진 데이터 안에 있는 원리를 생각하자 …

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

$$E = \frac{1}{2}\sum_i (y_i - t_i)^2$$

머신러닝 모델의 3단계

❶ 주어진 데이터를 기반으로 해서 미지의 데이터를 예측하는 식을 생각.

– 주어진 데이터의 속에 있는 구조를 특정 수식으로 모델화하는 것.

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \quad \text{(식 1.1)}$$

❷ 식에 포함된 파라미터의 좋고 나쁨을 판단하는 오차 함수를 준비한다

$$E = \frac{1}{2}\sum_i (y_i - t_i)^2 \quad \text{(식 1.2)}$$

❸ 오차 함수를 최소화할 수 있도록 파라미터값을 결정한다.

$$w_0 , w_1, w_2, w_3, w_4$$

❶ ❷ → 사람
❸ → computer

1.1.2 신경망의 필요성

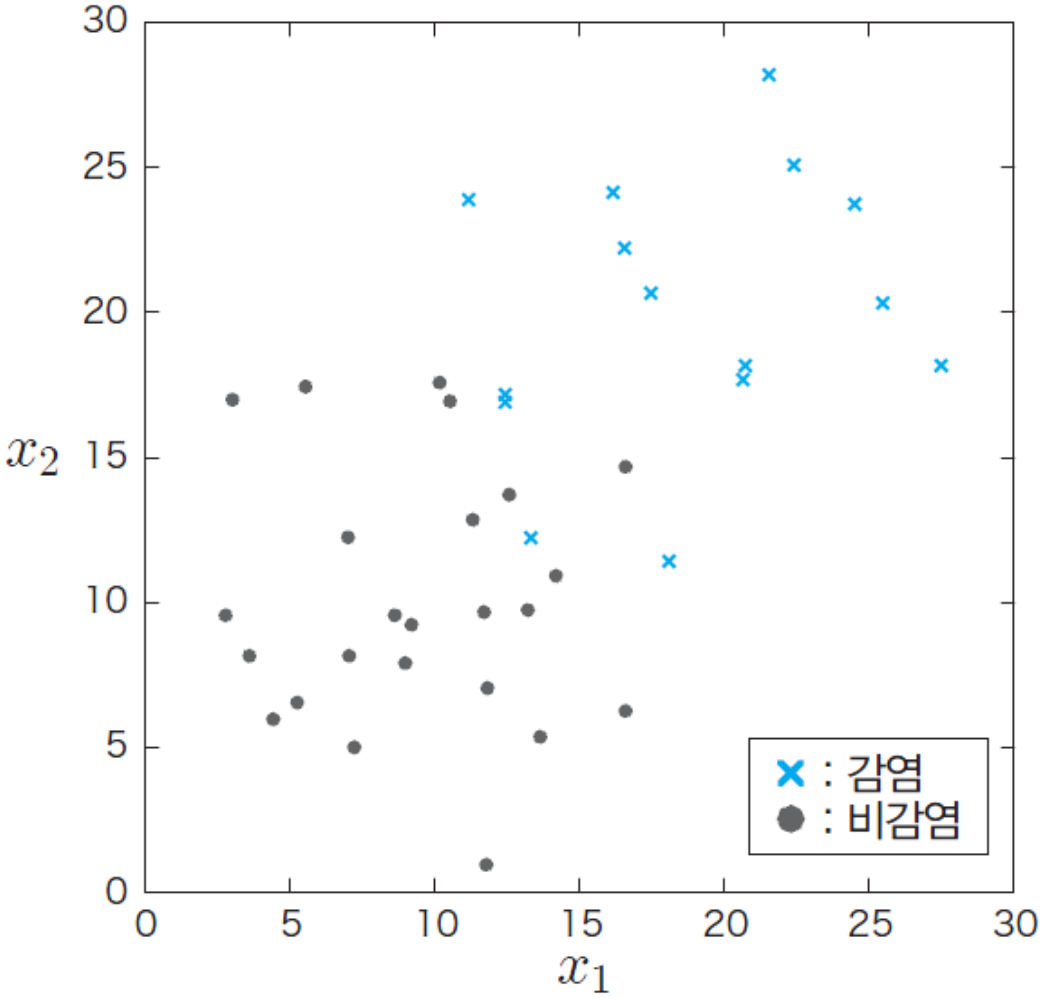
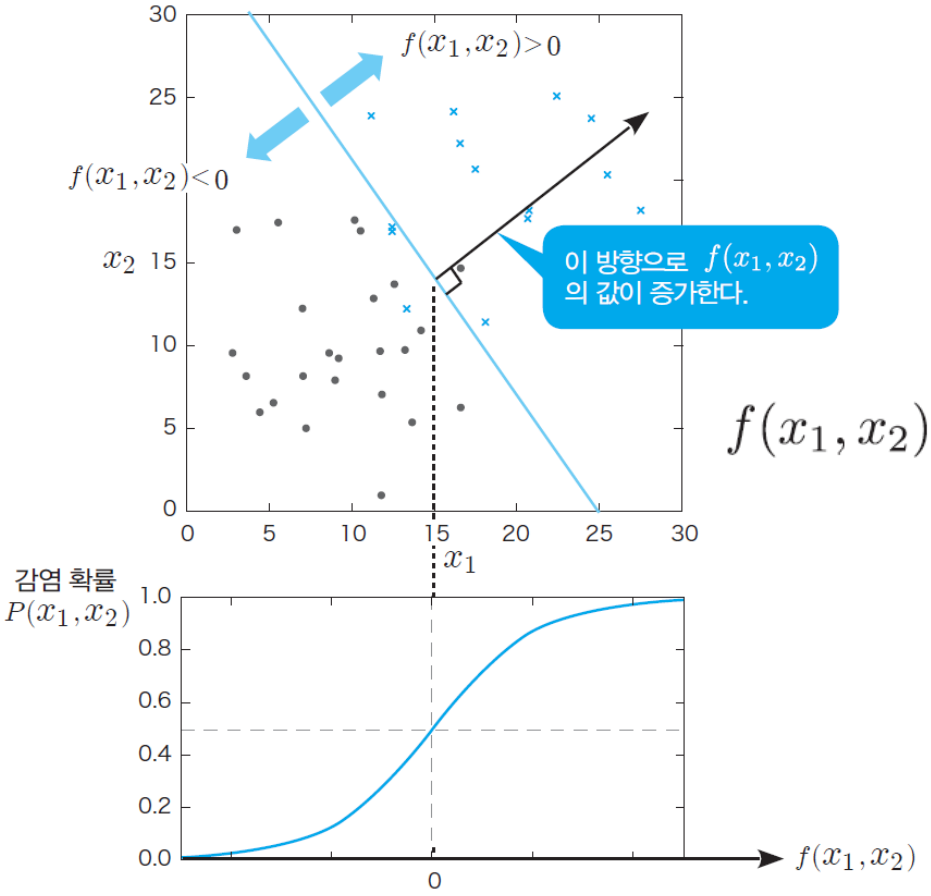


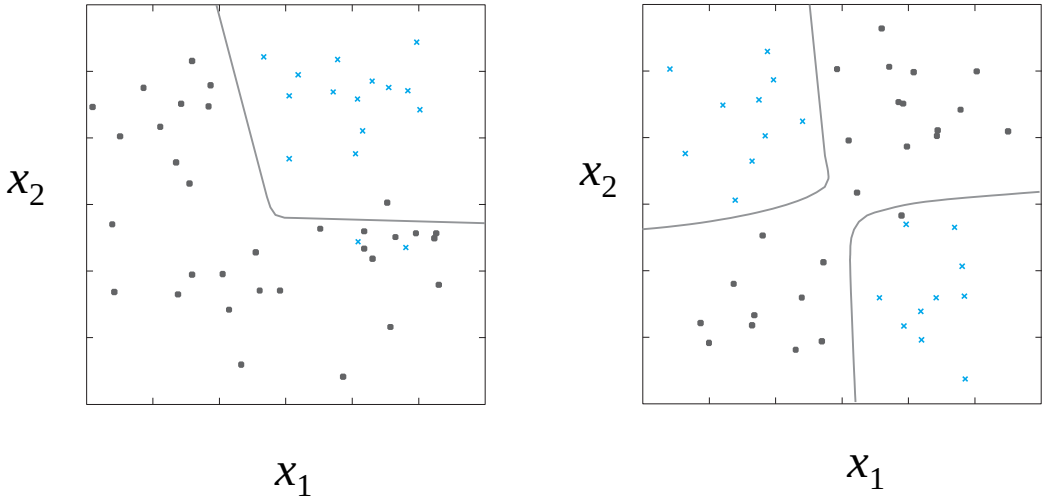
그림 1 – 5 예비 검사의 결과와 실제 감염 상황을 나타낸 데이터

그림 1-6 직선을 이용한 분류와 감염 확률로의 변환



$$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 = 0$$

그림 1-7 보다 복잡한 데이터 배치의 예



1.1.4 텐서플로를 이용한 파라미터 최적화

❶단계: 주어진 데이터의 속에 있는 구조를 특정 수식으로 모델화

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$ 식 1.1,5

❷단계: 파라미터의 좋고 나쁨을 평가하는 기준

$$E = \frac{1}{2}\sum_{n=1}^{12}(y_n - t_n)^2$$ 식 1.2,6

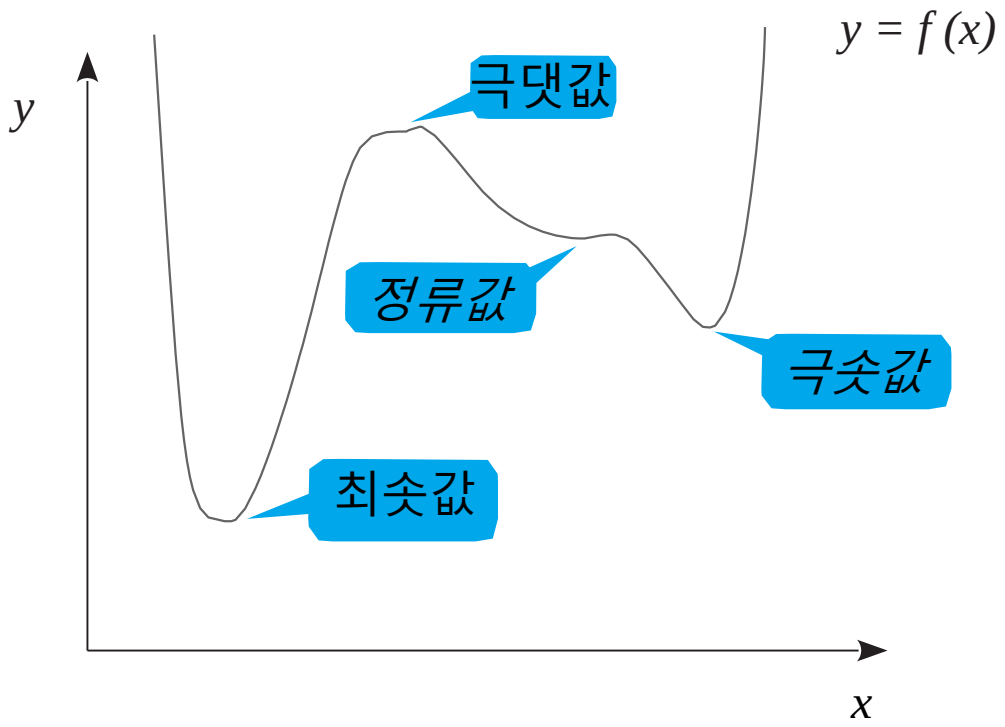
$$y_n = w_0 + w_1 n + w_2 n^2 + w_3 n^3 + w_4 n^4 = \sum_{m=0}^{4} w_m n^m$$ 식 1.7

$$E(w_0, w_1, w_2, w_3, w_4) = \frac{1}{2}\sum_{n=1}^{12}\left(\sum_{m=0}^{4} w_m n^m - t_n\right)^2$$ 식 1.8

❸단계로서 식 1.8의 값을 최소로 하는 w 를 결정

$$\frac{\partial E}{\partial w_m}(w_0, w_1, w_2, w_3, w_4) = 0 \ (m=0,\cdots,4)$$ 식 1.9

그림 1-13 그래프의 기울기가 0이 되는 지점



기울기 벡터(gradient vector)

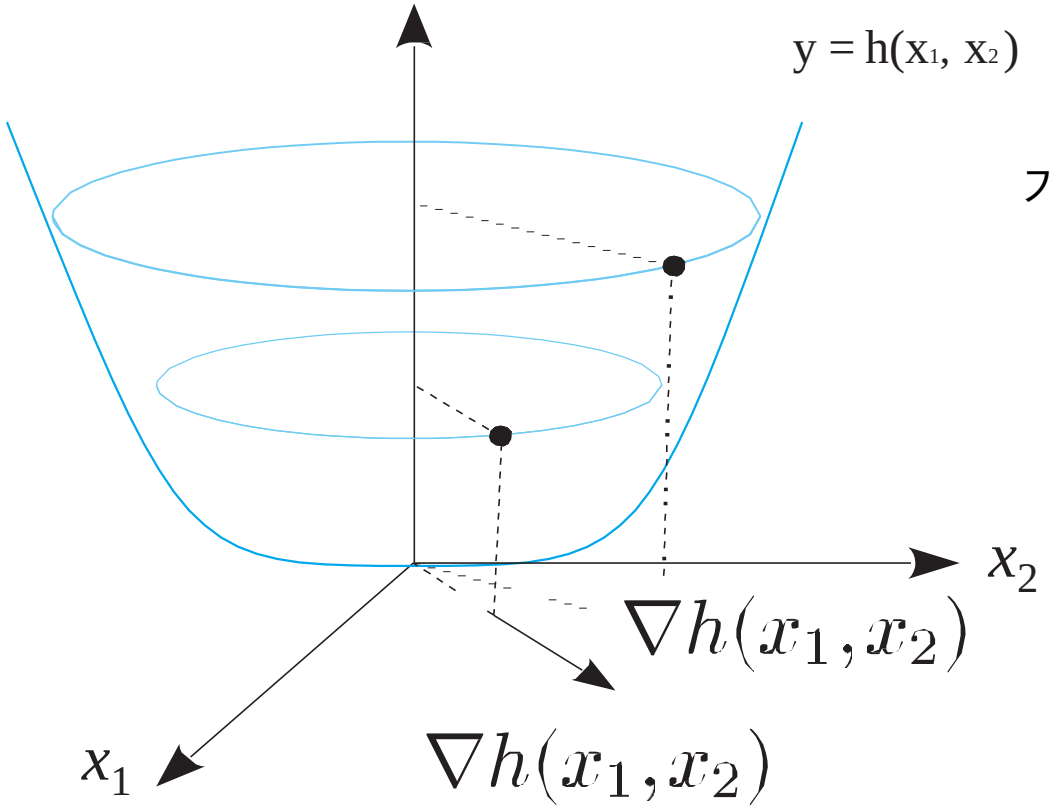
- 예)

$$h(x_1, x_2) = \frac{1}{4}\left(x_1^2 + x_2^2\right)$$

$$\frac{\partial h}{\partial x_1}(x_1, x_2) = \frac{1}{2}x_1, \quad \frac{\partial h}{\partial x_2}(x_1, x_2) = \frac{1}{2}x_2$$

$$\nabla h(x_1, x_2) = \begin{pmatrix} \frac{1}{2}x_1 \\ \\ \frac{1}{2}x_2 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

그림 1-14 2변수 함수의 기울기 벡터



기울기 벡터(gradient vector)

$$\nabla h(x_1, x_2)$$

$$\nabla E(\mathbf{w}) = \begin{pmatrix} \frac{\partial E}{\partial w_0}(\mathbf{w}) \\ \vdots \\ \frac{\partial E}{\partial w_4}(\mathbf{w}) \end{pmatrix}$$

$$\mathbf{x}^{\text{new}} = \mathbf{x} - \nabla h$$

경사 하강법 (gradient descent)

$$\mathbf{x}^{\text{new}} = \mathbf{x} - \nabla h$$

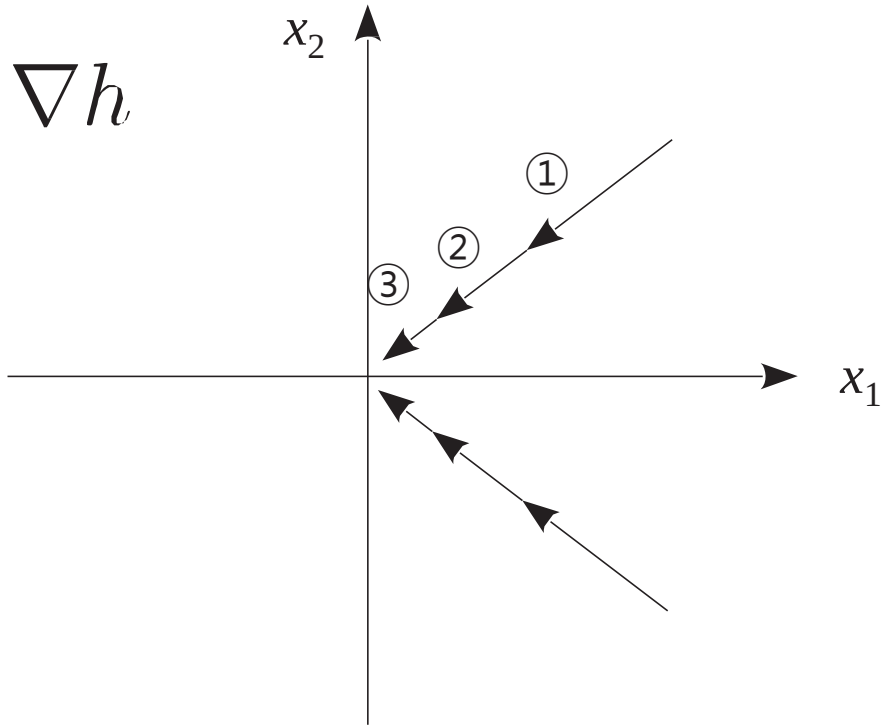
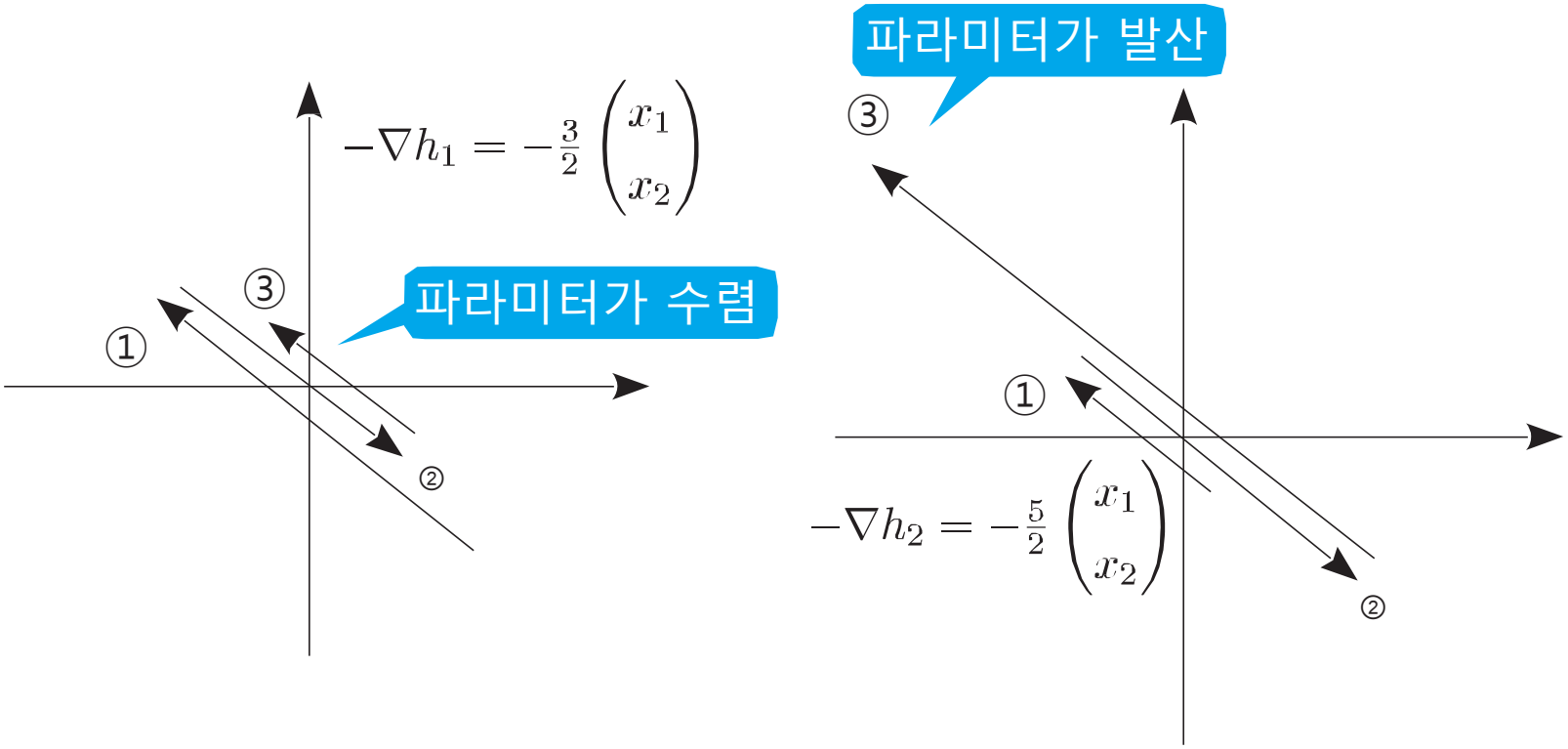


그림 1-15 경사 하강법으로 최솟값에 가까워지는 모습

그림 1-16 경사 하강법에 의한 두 종류의 이동 예



학습률 (learning rate, ϵ)

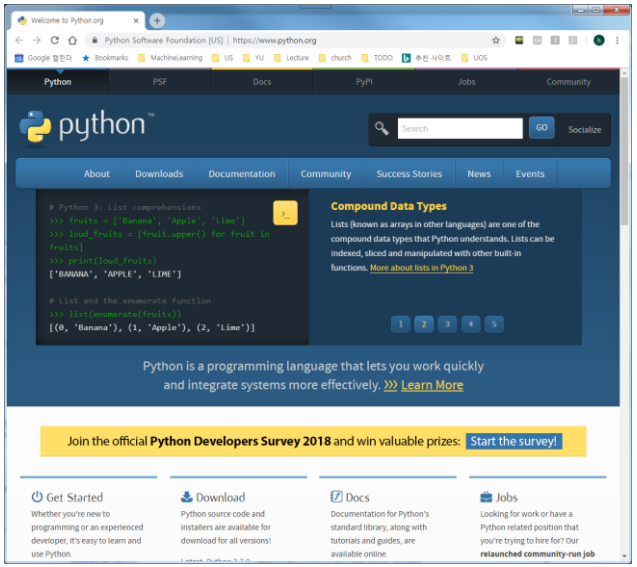
$$\mathbf{x}^{\text{new}} = \mathbf{x} - \epsilon \nabla h$$ 식 1.19

$$\mathbf{w}^{\text{new}} = \mathbf{w} - \epsilon \nabla E(\mathbf{w})$$

$$\nabla E(\mathbf{w}) = \begin{pmatrix} \frac{\partial E}{\partial w_0}(\mathbf{w}) \\ \vdots \\ \frac{\partial E}{\partial w_4}(\mathbf{w}) \end{pmatrix}$$ 기울기 벡터(gradient vector)

1.2 환경 준비

- https://www.tensorflow.org/install/



- Windows
  - Install Python 3.6 (64bit)
  - 명령창(관리자 권한)에서 다음 명령 실행

    pip install --upgrade tensorflow

- Google Colab
  - https://colab.research.google.com

- Docker : 애플리케이션 실행에 필요한 파일을 모아서 container image를 생성하고 리눅스 컨테이너 환경에서 애플리케애션을 실행하는 소프트웨어

Deep learning 도구

- Python : 쉽고 간편한 프로그래밍 언어
- Numpy : Package for scientific computing
- TensorFlow
- Keras: Deep Learning library for TensorFlow

1.3 텐서플로 훑어보기

1.3.1 다차원 배열을 이용한 모델 표현
1.3.2 텐서플로 코드를 이용한 표현
1.3.3 세션을 이용한 트레이닝 실행

예제 코드:

https://github.com/Jpub/TensorflowDeeplearning

Tensor

- 다차원 배열로 표현되는 값

$$y_n = (n^0, n^1, n^2, n^3, n^4) \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \qquad \mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{12} \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 1^0 & 1^1 & 1^2 & 1^3 & 1^4 \\ 2^0 & 2^1 & 2^2 & 2^3 & 2^4 \\ \vdots & & & & \\ 12^0 & 12^1 & 12^2 & 12^3 & 12^4 \end{pmatrix}, \ \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

1.3.2 텐서플로 코드를 이용한 표현

- 실행에 필요한 모듈 가져오기

```
# import modules
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Package for scientific computing

Graphic image library

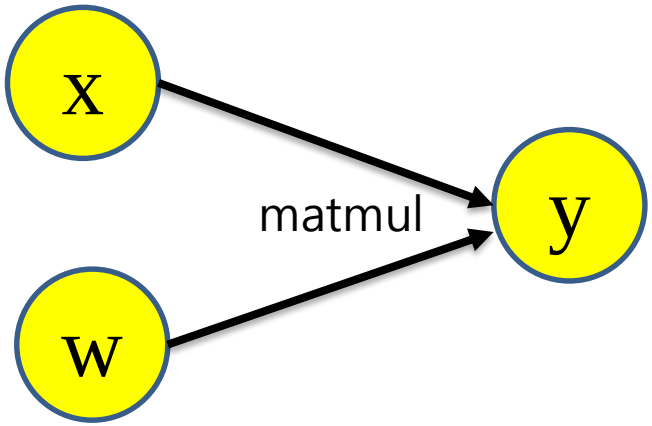
연산 정의

```
x = tf.placeholder(tf.float32, [None, 5])
w = tf.Variable(tf.zeros([5, 1]))
y = tf.matmul(x, w)
```

Placeholder : training data set

Variable : parameters to be optimized

None : 데이터의 개수. 미지정



$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{12} \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1^0 & 1^1 & 1^2 & 1^3 & 1^4 \\ 2^0 & 2^1 & 2^2 & 2^3 & 2^4 \\ \vdots & & & & \\ 12^0 & 12^1 & 12^2 & 12^3 & 12^4 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

$$y_n = (n^0, n^1, n^2, n^3, n^4) \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

Eq 24

12 x 1

None (not fixed) x 5

5 x 1

Objective function 정의

t = tf.placeholder(tf.float32, [None, 1])

loss = tf.reduce_sum(tf.square(y-t))

$$E = \sum_{i=1}^{N}(y_i - t_i)^2$$

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_{12} \end{pmatrix}$$ Target value

$v = (v_1, v_2, \ldots v_N)^T$

$$\text{square}(\mathbf{v}) = \begin{pmatrix} v_1^2 \\ v_2^2 \\ \vdots \\ v_N^2 \end{pmatrix}$$

$$\text{reduce\_sum}(\mathbf{v}) = \sum_{i=1}^{N} v_i$$

$$E = \frac{1}{2}\text{reduce\_sum}\left(\text{square}(\mathbf{y} - \mathbf{t})\right)$$

Table 2. Tensorflow 의 주요 자료형

자료형	설명
tf.float32	32비트 부동 소수점
tf.float64	64비트 부동 소수점
tf.int8	8비트 부호 있는 정수
tf.int16	16비트 부호 있는 정수
tf.int32	32비트 부호 있는 정수
tf.int64	64비트 부호 있는 정수
tf.string	문자열(가변 길이 바이트 배열)
tf.bool	부울값
tf.complex64	복소수(32비트 부동 소수점의 실수부와 허수부를 갖는다)

학습 방법 선택

- train_step = tf.train.AdamOptimizer().minimize(loss)

The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
Initialize 1st and 2nd moment variables $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{0}$
Initialize time step $t = 0$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
$t \leftarrow t + 1$
Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

https://www.deeplearningbook.org/

Optimizers

- The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables.
  - tf.train.Optimizer
  - tf.train.GradientDescentOptimizer
  - tf.train.AdadeltaOptimizer
  - tf.train.AdagradOptimizer
  - tf.train.AdagradDAOptimizer
  - tf.train.MomentumOptimizer
  - tf.train.AdamOptimizer
  - tf.train.FtrlOptimizer
  - tf.train.ProximalGradientDescentOptimizer
  - tf.train.ProximalAdagradOptimizer
  - tf.train.RMSPropOptimizer

https://www.tensorflow.org/api_guides/python/train

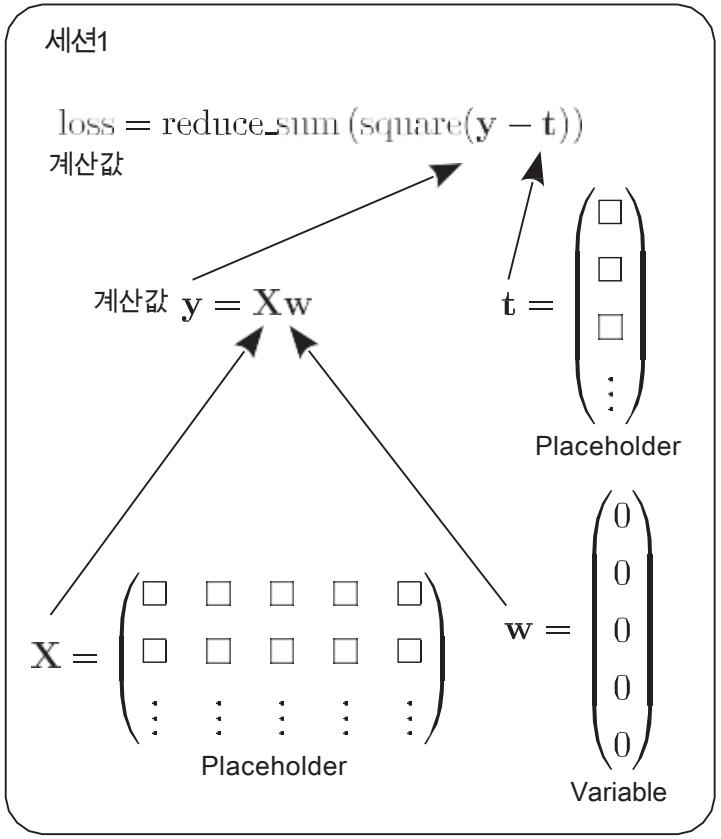
1.3.3 세션을 이용한 트레이닝 실행

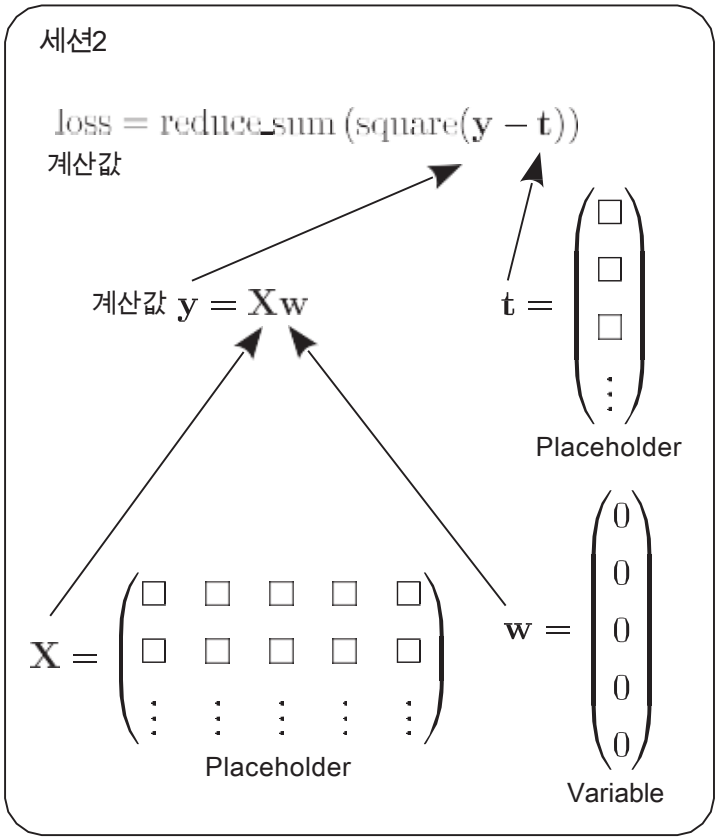
- sess = tf.Session()
- Training algorithm의 실행 환경
- 복수의 session, 개별 계산 가능
- sess.run(tf.global_variables_initializer()) 초기화

그림 1 - 25 세션 내에서 변숫값을 관리하는 구조

```
sess1 = tf.Session()
sess1.run(tf.global_variables_initializer())
```

```
sess2 = tf.Session()
sess2.run(tf.global_variables_initializer())
```

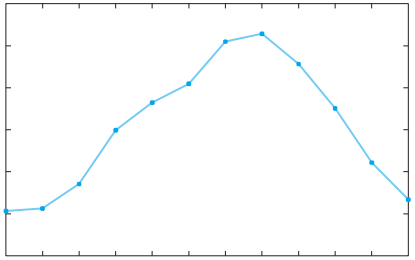




트레이닝 세트 데이터 준비

```
train_t = np.array([5.2, 5.7, 8.6, 14.9, 18.2, 20.4, 25.5, 26.4, 22.8, 17.5, 11.1, 6.6])
train_t = train_t.reshape([12,1])

train_x = np.zeros([12, 5])
for row, month in enumerate(range(1, 13)):
    for col, n in enumerate(range(0, 5)):
        train_x[row][col] = month**n
```



Array object : python의 list에 수치 계산에 편리한 기능을 추가한 wrapper

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{12} \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1^0 & 1^1 & 1^2 & 1^3 & 1^4 \\ 2^0 & 2^1 & 2^2 & 2^3 & 2^4 \\ \vdots & & & & \\ 12^0 & 12^1 & 12^2 & 12^3 & 12^4 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

enumerate()

enumerate(thing),
thing is either an iterator or a sequence,
returns a iterator that will return
(0, *thing*[0]), (1, *thing*[1]), (2, *thing*[2]), and so forth.
A common idiom to change every element of a list looks like this:

```
for i in range(len(L)):
    item = L[i]
    # ... compute some result based on item ...
    L[i] = result
```

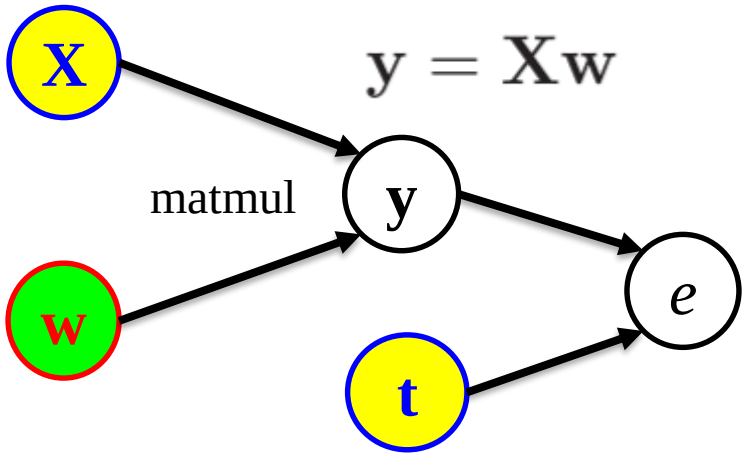
This can be rewritten using enumerate() as:

```
for i, item in enumerate(L):
    # ... compute some result based on item ...
    L[i] = result
```

https://docs.python.org/2.3/whatsnew/section-enumerate.html

목적함수 계산 및 최적화

```
x = tf.placeholder(tf.float32, [None, 5])
w = tf.Variable(tf.zeros([5, 1]))
y = tf.matmul(x, w)
t = tf.placeholder(tf.float32, [None, 1])
e = tf.reduce_sum(tf.square(y-t))
…
train_step = tf.train.GradientDescentOptimizer().minimize(e)
```



$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{12} \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1^0 & 1^1 & 1^2 & 1^3 & 1^4 \\ 2^0 & 2^1 & 2^2 & 2^3 & 2^4 \\ \vdots & & & & \\ 12^0 & 12^1 & 12^2 & 12^3 & 12^4 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

$$y_n = (n^0, n^1, n^2, n^3, n^4) \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

None (not fixed) x 5

5 x 1

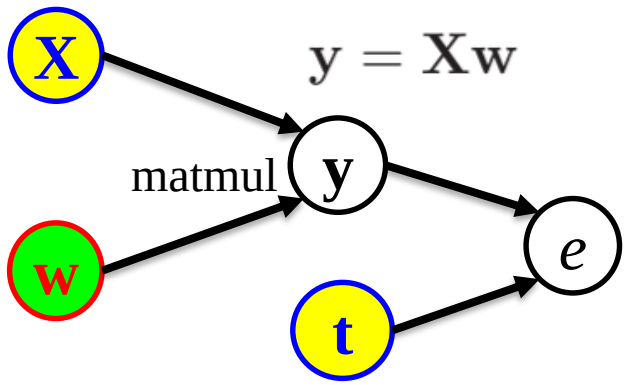
경사 하강법을 이용한 파라미터 최적화

```
i = 0
for _ in range(100000):
    i += 1
    sess.run(train_step, feed_dict={x:train_x, t:train_t})
    if i % 10000 == 0:
        loss_val = sess.run(loss, feed_dict={x:train_x, t:train_t})
        print ('Step: %d, Loss: %f' % (i, loss_val))
```

Variable w를 수정

Evaluate loss

tf.placeholder로 정의한 변수는 반드시
feed_dict option으로 지정. (dictionary)
그렇지 않을 경우 Invalid Argument Error



Test

```
w_val = sess.run(w)

def predict(x):
    result = 0.0
    for n in range(0, 5):
        result += w_val[n][0] * x**n
    return result
```

Evaluate Variable w

$$y(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 = \sum_{m=0}^{4} w_m x^m$$

그래프 그리기

```
1. fig = plt.figure()
2. subplot = fig.add_subplot(1,1,1)
3. subplot.set_xlim(1,12)
4. subplot.scatter(range(1,13), train_t)
5. linex = np.linspace(1,12,100)
6. liney = predict(linex)
7. subplot.plot(linex, liney)
```

[y,x,n] = [세로에 y개, 가로에 x개, n 번째 영역]

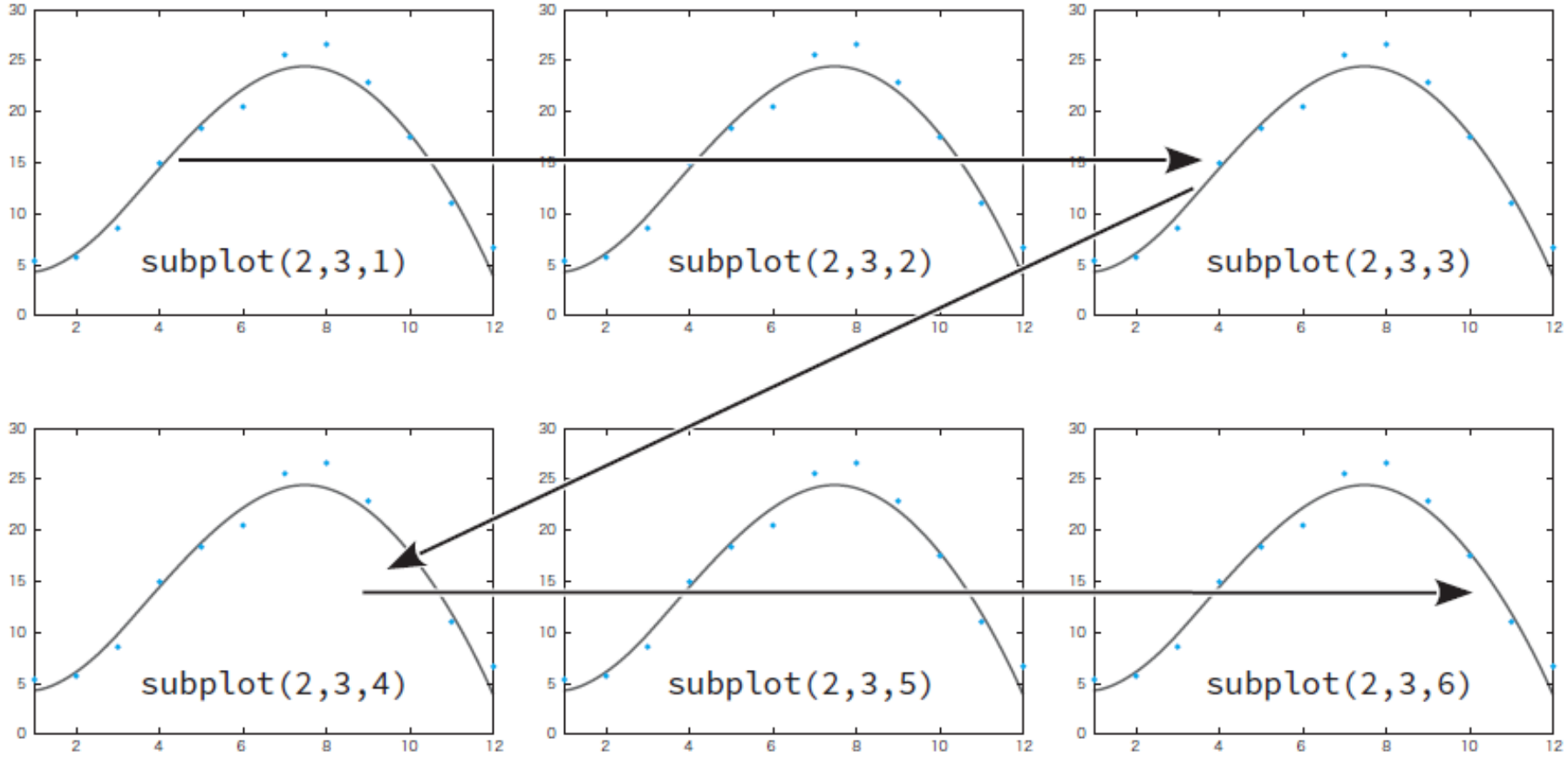
X축의 표시 범위

X축 방향, y축 방향 데이터의 list

1~12까지의 범위를 같은 간격으로 나눈 100개의 값 리스트

linex로 지정된 x축상의 100개의 위치에 해당하는 점을 연결해서 부드러운 곡선을 표현

그림 1 - 26 그래프 영역(subplot)의 순번



그래프 그리기 코드 설명

- 4행은 트레이닝 세트 데이터, 즉 실제로 관측된 월별 평균 기온을 산포도로 나타낸다.
- pyplot 모듈로 그래프를 그릴 때는 일반적으로 축 방향 데이터의 리스트와 y 축 방향 데이터의 리스트를 인수로 지정.

- array 오브젝트에 단일 값(스칼라)을 대입해야 하는 함수에 array 오브젝트를 대입하면 각 값의 함숫값을 다시 array 오브젝트로 얻을 수 있다.

결과 모델