(수치해석 과제 #6)

5장, P167~p225

(In and Out Practic)

소프트웨어학과 20162820 김영민

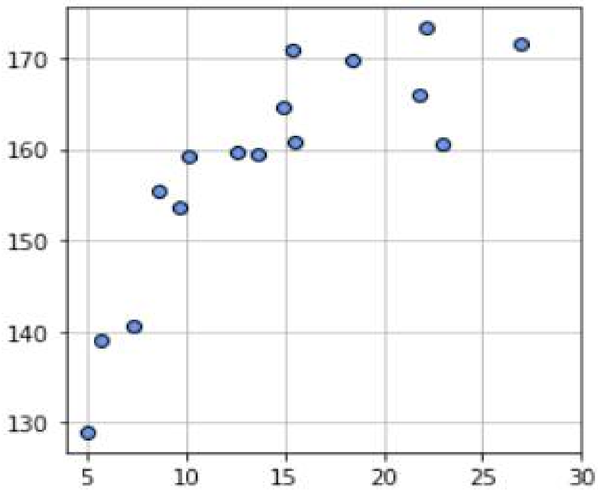
P168 ~ 169

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(seed=1)
X_min = 4
X_max = 30
X_n = 16
X = 5 + 25 * np.random.rand(X_n)
Prm_c = [170, 108, 0.2]
T = Prm_c[0] - Prm_c[1] * np.exp(-Prm_c[2] * X) ₩
+ 4 * np.random.randn(X_n)
np.savez('ch5_data.npz', X=X, X_min=X_min, X_max=X_max, X_n=X_n, T=T)

#리스트 5-1-(2)
print(X)

#리스트 5-1-(3)
print(np.round(X, 2))

#리스트 5-1-(4)
print(np.round(T, 2))
```

```
[15.42555012 23.00811234  5.00285937 12.55831432  8.66889727  7.30846487
  9.65650528 13.63901818 14.91918686 18.47041835 15.47986286 22.13048751
 10.11130624 26.95293591  5.68468983 21.76168775]
[15.43 23.01  5.   12.56  8.67  7.31  9.66 13.64 14.92 18.47 15.48 22.13
 10.11 26.95  5.68 21.76]
[170.91 160.68 129.   159.7  155.46 140.56 153.65 159.43 164.7  169.65
 160.71 173.29 159.31 171.52 138.96 165.87]
```

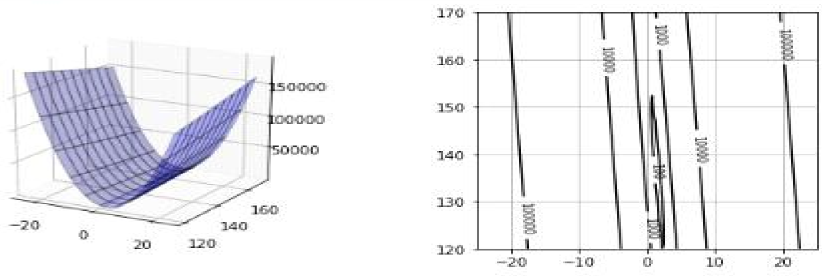
P170

In [7]:
```python
plt.figure(figsize=(4, 4))
plt.plot(X, T, marker='o', linestyle='None',
         markeredgecolor='black', color='cornflowerblue')
plt.xlim(X_min, X_max)
plt.grid(True)
plt.show()
```



P172 ~ 173

```python
from mpl_toolkits.mplot3d import Axes3D
def mse_line(x, t, w):
    y = w[0] * x + w[1]
    mse = np.mean((y - t)**2)
    return mse

xn = 100
w0_range = [-25, 25]
w1_range = [120, 170]
x0 = np.linspace(w0_range[0], w0_range[1], xn)
x1 = np.linspace(w1_range[0], w1_range[1], xn)
xx0, xx1 = np.meshgrid(x0, x1)
J = np.zeros((len(x0), len(x1)))
for i0 in range(xn):
    for i1 in range(xn):
        J[i1, i0] = mse_line(X, T, (x0[i0], x1[i1]))

plt.figure(figsize=(9.5, 4))
plt.subplots_adjust(wspace=0.5)

ax = plt.subplot(1, 2, 1, projection='3d')
ax.plot_surface(xx0, xx1, J, rstride=10, cstride=10, alpha=0.3, color='blue', edgecolor='black')
ax.set_xticks([-20, 0, 20])
ax.set_yticks([120, 140, 160])
ax.view_init(20, -60)

plt.subplot(1, 2, 2)
cont = plt.contour(xx0, xx1, J, 30, colors='black',levels=[100, 1000, 10000, 100000])
cont.clabel(fmt='%1.0f', fontsize=8)
plt.grid(True)
plt.show()
```



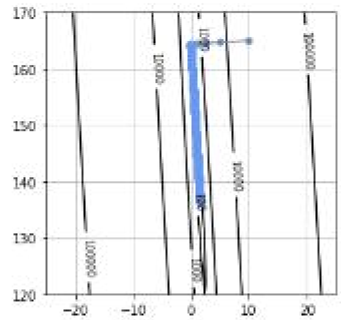
P177 ~ 180

```
def dmse_line(x, t, w):
    y = w[0] * x + w[1]
    d_w0 = 2 * np.mean((y - t) * x)
    d_w1 = 2 * np.mean(y - t)
    return d_w0, d_w1

d_w = dmse_line(X, T, [10, 165])
print(np.round(d_w, 1))
```

```
[5046.3  301.8]
```

```
def fit_line_num(x, t):
    w_init = [10.0, 165.0]
    alpha = 0.001
    i_max = 100000
    eps = 0.1
    w_i = np.zeros([i_max, 2])
    w_i[0, :] = w_init
    for i in range(1, i_max):
        dmse = dmse_line(x, t, w_i[i - 1])
        w_i[i, 0] = w_i[i - 1, 0] - alpha * dmse[0]
        w_i[i, 1] = w_i[i - 1, 1] - alpha * dmse[1]
        if max(np.absolute(dmse)) < eps:
            break
    w0 = w_i[i, 0]
    w1 = w_i[i, 1]
    w_i = w_i[:i, :]
    return w0, w1, dmse, w_i

plt.figure(figsize=(4, 4))
xn = 100
w0_range = [-25, 25]
w1_range = [120, 170]
x0 = np.linspace(w0_range[0], w0_range[1], xn)
x1 = np.linspace(w1_range[0], w1_range[1], xn)
xx0, xx1 = np.meshgrid(x0, x1)
J = np.zeros((len(x0), len(x1)))
for i0 in range(xn):
    for i1 in range(xn):
        J[i1, i0] = mse_line(X, T, (x0[i0], x1[i1]))
cont = plt.contour(xx0, xx1, J, 30, colors='black',
                   levels=(100, 1000, 10000, 100000))
cont.clabel(fmt='%1.0f', fontsize=8)
plt.grid(True)
W0, W1, dMSE, W_history = fit_line_num(X, T)
print('반복 횟수 {0}'.format(W_history.shape[0]))
print('W=[{0:.6f}, {1:.6f}]'.format(W0, W1))
print('dMSE=[{0:.6f}, {1:.6f}]'.format(dMSE[0], dMSE[1]))
print('MSE={0:.6f}'.format(mse_line(X, T, [W0, W1])))
plt.plot(W_history[:, 0], W_history[:, 1], '.-',
         color='gray', markersize=10, markeredgecolor='cornflowerblue')
plt.show()
```

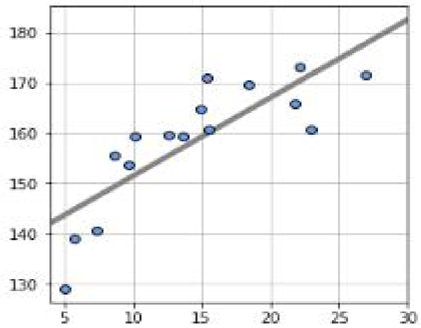
```
반복 횟수 13820
W=[1.539947, 136.176160]
dMSE=[-0.005794, 0.099991]
MSE=49.027452
```



P180

```
:  def fit_line(x, t):
       mx  = np.mean(x)
       mt  = np.mean(t)
       mtx = np.mean(t  * x)
       mxx = np.mean(x  * x)
       w0  = (mtx - mt  * mx) / (mxx - mx**2)
       w1  = mt  - w0  * mx
       return np.array([w0, w1])

   W  = fit_line(X, T)
   print("w0={0:.3f}, w1={1:.3f}".format(W[0], W[1]))
   mse = mse_line(X, T, W)
   print("SD={0:.3f} cm".format(np.sqrt(mse)))
   plt.figure(figsize=(4, 4))
   show_line(W)
   plt.plot(X, T, marker='o', linestyle='None',
            color='cornflowerblue', markeredgecolor='black')
   plt.xlim(X_min, X_max)
   plt.grid(True)
   plt.show()
```

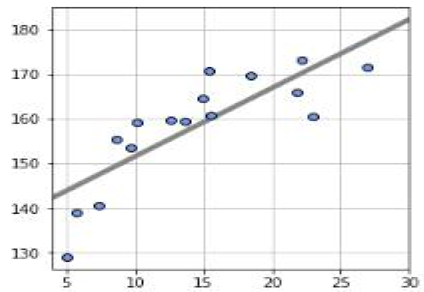
```
w0=1.558, w1=135.872
SD=7.001 cm
```



P185

```
[10]:  def show_line(w):
           xb  = np.linspace(X_min, X_max, 100)
           y = w[0] * xb + w[1]
           plt.plot(xb, y, color=(.5, .5, .5), linewidth=4)

           plt.figure(figsize=(4, 4))
       W=np.array([W0, W1])
       mse = mse_line(X, T, W)
       print("w0={0:.3f}, w1={1:.3f}".format(W0, W1))
       print("SD={0:.3f} cm".format(np.sqrt(mse)))
       show_line(W)
       plt.plot(X, T, marker='o', linestyle='None',
                color='cornflowerblue', markeredgecolor='black')
       plt.xlim(X_min, X_max)
       plt.grid(True)
       plt.show()
```

```
w0=1.540, w1=136.176
SD=7.002 cm
```

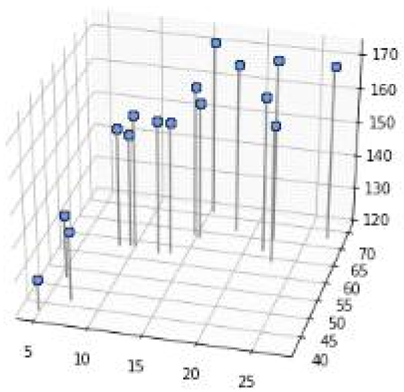


P186 ~ 187

In [9]:
```
X0 = X
X0_min = 5
X0_max = 30
np.random.seed(seed=1)
X1 = 23 * (T / 100)**2 + 2 * np.random.randn(X_n)
X1_min = 40
X1_max = 75

print(np.round(X0, 2))
print(np.round(X1, 2))
print(np.round(T, 2))
```

```
[15.43 23.01  5.   12.56  8.67  7.31  9.66 13.64 14.92 18.47 15.48 22.13
 10.11 26.95  5.68 21.76]
[70.43 58.15 37.22 56.51 57.32 40.84 57.79 56.94 63.03 65.69 62.33 64.95
 57.73 66.89 46.68 61.08]
[170.91 160.68 129.   159.7  155.46 140.56 153.65 159.43 164.7  169.65
 160.71 173.29 159.31 171.52 138.96 165.87]
```

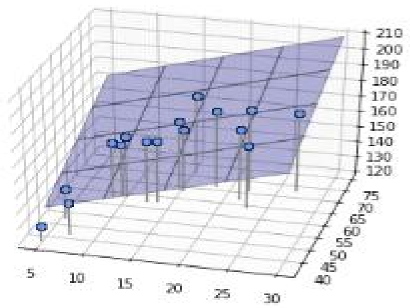
n [14]:
```
def show_data2(ax, x0, x1, t):
    for i in range(len(x0)):
        ax.plot([x0[i], x0[i]], [x1[i], x1[i]],
                [120, t[i]], color='gray')
        ax.plot(x0, x1, t, 'o',
                color='cornflowerblue', markeredgecolor='black',
                markersize=6, markeredgewidth=0.5)
        ax.view_init(elev=35, azim=-75)

plt.figure(figsize=(6, 5))
ax = plt.subplot(1,1,1,projection='3d')
show_data2(ax, X0, X1, T)
plt.show()
```



P189 ~ 190

```
def show_plane(ax, w):
    px0 = np.linspace(X0_min, X0_max, 5)
    px1 = np.linspace(X1_min, X1_max, 5)
    px0, px1 = np.meshgrid(px0, px1)
    y = w[0]*px0 + w[1] * px1 + w[2]
    ax.plot_surface(px0, px1, y, rstride=1, cstride=1, alpha=0.3,
                    color='blue', edgecolor='black')

def mse_plane(x0, x1, t, w):
    y = w[0] * x0 + w[1] * x1 + w[2]
    mse = np.mean((y - t)**2)
    return mse

plt.figure(figsize=(6, 5))
ax = plt.subplot(1, 1, 1, projection='3d')
W = [1.5, 1, 90]
show_plane(ax, W)
show_data2(ax, X0, X1, T)
mse = mse_plane(X0, X1, T, W)
print("SD={0:.3f} cm".format(np.sqrt(mse)))
plt.show()
```

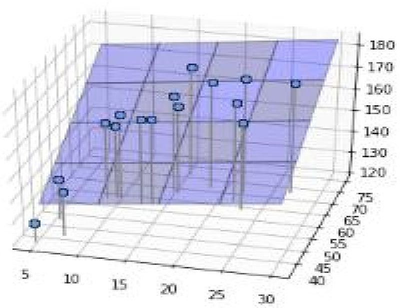
SD=12.876 cm



P192

```
def fit_plane(x0, x1, t):
    c_tx0 = np.mean(t * x0) - np.mean(t) * np.mean(x0)
    c_tx1 = np.mean(t * x1) - np.mean(t) * np.mean(x1)
    c_x0x1 = np.mean(x0 * x1) - np.mean(x0) * np.mean(x1)
    v_x0 = np.var(x0)
    v_x1 = np.var(x1)
    w0 = (c_tx1 * c_x0x1 - v_x1 * c_tx0) / (c_x0x1**2 - v_x0 * v_x1)
    w1 = (c_tx0 * c_x0x1 - v_x0 * c_tx1) / (c_x0x1**2 - v_x0 * v_x1)
    w2 = -w0 * np.mean(x0) - w1 * np.mean(x1) + np.mean(t)
    return np.array([w0, w1, w2])

plt.figure(figsize=(6, 5))
ax = plt.subplot(1, 1, 1, projection='3d')
W = fit_plane(X0, X1, T)
print("w0={0:.1f}, w1={1:.1f}, w2={2:.1f}".format(W[0], W[1], W[2]))
show_plane(ax, W)
show_data2(ax, X0, X1, T)
mse = mse_plane(X0, X1, T, W)
print("SD={0:.3f} cm".format(np.sqrt(mse)))
plt.show()
```

w0=0.5, w1=1.1, w2=89.0
SD=2.546 cm

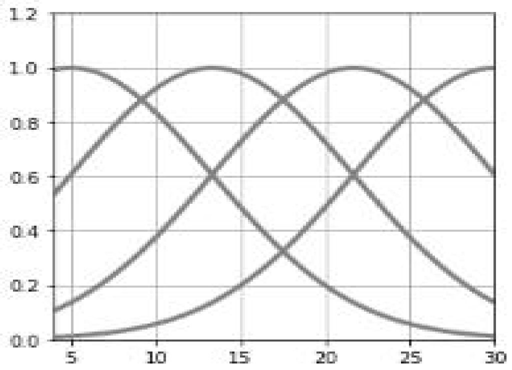


P201 ~ 202

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

outfile = np.load('ch5_data.npz')
X = outfile['X']
X_min = outfile['X_min']
X_max = outfile['X_max']
X_n = outfile['X_n']
T = outfile['T']
```

```
def gauss(x, mu, s):
    return np.exp(-(x - mu)**2 / (2 * s**2))
```

```
M = 4
plt.figure(figsize=(4, 4))
mu = np.linspace(5, 30, M)
s = mu[1] - mu[0]
xb = np.linspace(X_min, X_max, 100)
for j in range(M):
    y = gauss(xb, mu[j], s)
    plt.plot(xb, y, color='gray', linewidth=3)
plt.grid(True)
plt.xlim(X_min, X_max)
plt.ylim(0, 1.2)
plt.show()
```



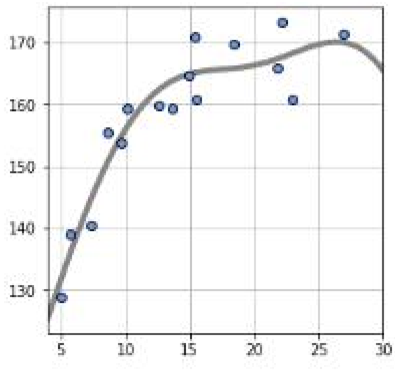
P207

```
def mse_gauss_func(x, t, w):
    y = gauss_func(w, x)
    mse = np.mean((y - t)**2)
    return mse
```

```
def fit_gauss_func(x, t, m):
    mu = np.linspace(5, 30, m)
    s = mu[1] - mu[0]
    n = x.shape[0]
    psi = np.ones((n, m+1))
    for j in range(m):
        psi[:, j] = gauss(x, mu[j], s)
    psi_T = np.transpose(psi)

    b = np.linalg.inv(psi_T.dot(psi))
    c = b.dot(psi_T)
    w = c.dot(t)
    return w
```

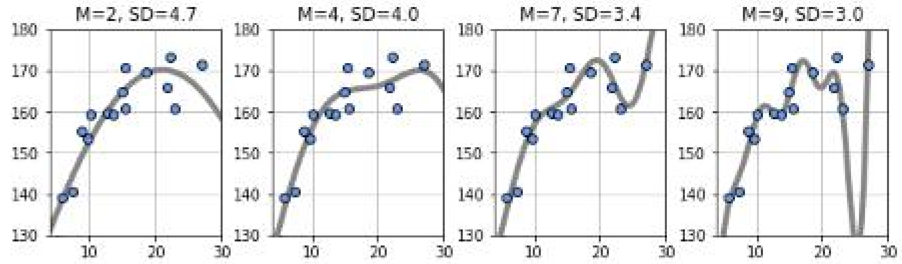
```
def show_gauss_func(w):
    xb = np.linspace(X_min, X_max, 100)
    y = gauss_func(w, xb)
    plt.plot(xb, y, c=[.5, .5, .5], lw=4)

plt.figure(figsize=(4, 4))
M = 4
W = fit_gauss_func(X, T, M)
show_gauss_func(W)
plt.plot(X, T, marker='o', linestyle='None',color='cornflowerblue', markeredgecolor='black')
plt.xlim(X_min, X_max)
plt.grid(True)
mse = mse_gauss_func(X, T, W)
print('W='+ str(np.round(W,1)))
print("SD={0:.2f} cm".format(np.sqrt(mse)))
plt.show()
```

```
W=[29.4 75.7  2.9 98.3 54.9]
SD=3.98 cm
```

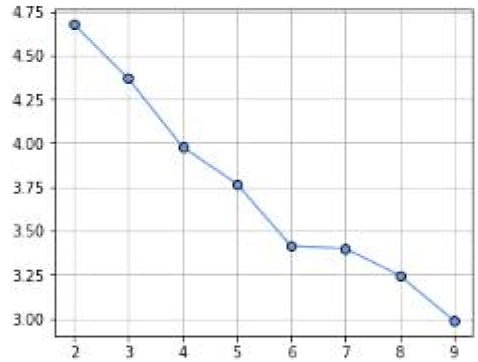


P 207~209

```python
plt.figure(figsize=(10, 2.5))
plt.subplots_adjust(wspace=0.3)
M = [2, 4, 7, 9]
for i in range(len(M)):
    plt.subplot(1, len(M), i + 1)
    W = fit_gauss_func(X, T, M[i])
    show_gauss_func(W)
    plt.plot(X, T, marker='o', linestyle='None',
             color='cornflowerblue', markeredgecolor='black')
    plt.xlim(X_min, X_max)
    plt.grid(True)
    plt.ylim(130, 180)
    mse = mse_gauss_func(X, T, W)

    plt.title("M={0:d}, SD={1:.1f}".format(M[i], np.sqrt(mse)))
plt.show()
```

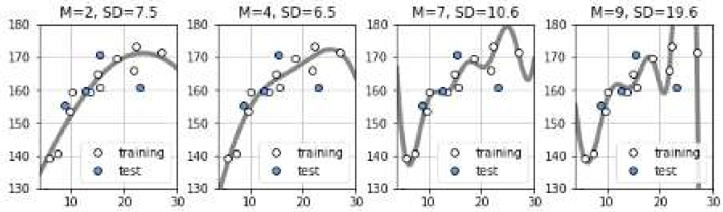



```python
plt.figure(figsize=(5, 4))
M = range(2, 10)
mse2 = np.zeros(len(M))
for i in range(len(M)):
    W = fit_gauss_func(X, T, M[i])
    mse2[i] = np.sqrt(mse_gauss_func(X, T, W))
plt.plot(M, mse2, marker='o',
         color='cornflowerblue', markeredgecolor='black')
plt.grid(True)
plt.show()
```

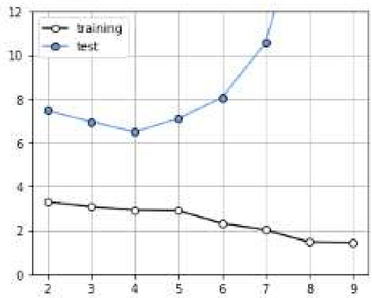


P 210 ~ 213

```
X_test = X[:int(X_n / 4 + 1)]
T_test = T[:int(X_n / 4 + 1)]
X_train = X[int(X_n / 4 + 1):]
T_train = T[int(X_n / 4 + 1):]

plt.figure(figsize=(10, 2.5))
plt.subplots_adjust(wspace=0.3)
M = [2, 4, 7, 9]
for i in range(len(M)):
    plt.subplot(1, len(M), i + 1)
    W = fit_gauss_func(X_train, T_train, M[i])
    show_gauss_func(W)
    plt.plot(X_train, T_train, marker='o',linestyle='None', color='white',markeredgecolor='black', label='training')
    plt.plot(X_test, T_test, marker='o', linestyle='None',color='cornflowerblue', markeredgecolor='black', label='test')
    plt.legend(loc='lower right', fontsize=10, numpoints=1)
    plt.xlim(X_min, X_max)
    plt.ylim(130, 180)
    plt.grid(True)
    mse = mse_gauss_func(X_test, T_test, W)
    plt.title("M={0:d}, SD={1:.1f}".format(M[i], np.sqrt(mse)))
plt.show()
```




```
plt.figure(figsize=(5, 4))
M = range(2, 10)
mse_train = np.zeros(len(M))
mse_test = np.zeros(len(M))
for i in range(len(M)):
    W = fit_gauss_func(X_train, T_train, M[i])
    mse_train[i] = np.sqrt(mse_gauss_func(X_train, T_train, W))
    mse_test[i] = np.sqrt(mse_gauss_func(X_test, T_test, W))
plt.plot(M, mse_train, marker='o', linestyle='-', markerfacecolor='white', markeredgecolor='black', color='black', label='training')
plt.plot(M, mse_test, marker='o', linestyle='-', color='cornflowerblue', markeredgecolor='black', label='test')
plt.legend(loc='upper left', fontsize=10)
plt.ylim(0, 12)
plt.grid(True)
plt.show()
```



P215

```python
def kfold_gauss_func(x, t, m, k):
    n = x.shape[0]
    mse_train = np.zeros(k)
    mse_test  = np.zeros(k)
    for i in range(0, k):
        x_train = x[np.fmod(range(n), k) != i] # (A)
        t_train = t[np.fmod(range(n), k) != i] # (A)
        x_test  = x[np.fmod(range(n), k) == i] # (A)
        t_test  = t[np.fmod(range(n), k) == i] # (A)
        wm = fit_gauss_func(x_train, t_train, m)
        mse_train[i] = mse_gauss_func(x_train, t_train, wm)
        mse_test[i] = mse_gauss_func(x_test, t_test, wm)
    return mse_train, mse_test
```

```python
np.fmod(range(10),5)
```

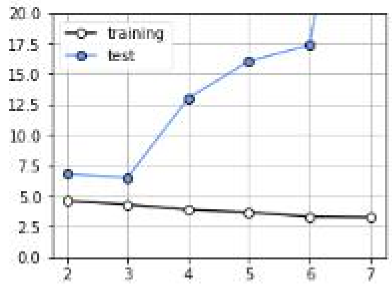
```
array([0, 1, 2, 3, 4, 0, 1, 2, 3, 4], dtype=int32)
```

```python
M = 4
K = 4
kfold_gauss_func(X, T, M, K)
```

```
(array([12.87927851,  9.81768697, 17.2615696 , 12.92270498]),
 array([ 39.65348229, 734.70782012,  18.30921743,  47.52459642]))
```

P216

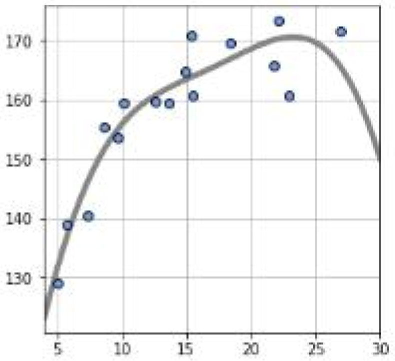
```python
M = range(2, 8)
K = 16
Cv_Gauss_train = np.zeros((K, len(M)))
Cv_Gauss_test  = np.zeros((K, len(M)))
for i in range(0, len(M)):
    Cv_Gauss_train[:, i], Cv_Gauss_test[:, i] =₩
                    kfold_gauss_func(X, T, M[i], K)
mean_Gauss_train = np.sqrt(np.mean(Cv_Gauss_train, axis=0))
mean_Gauss_test  = np.sqrt(np.mean(Cv_Gauss_test, axis=0))


plt.figure(figsize=(4, 3))
plt.plot(M, mean_Gauss_train, marker='o', linestyle='-',
         color='k', markerfacecolor='w', label='training')
plt.plot(M, mean_Gauss_test, marker='o', linestyle='-',
         color='cornflowerblue', markeredgecolor='black', label='test')
plt.legend(loc='upper left', fontsize=10)
plt.ylim(0, 20)
plt.grid(True)
plt.show()
```



P217

```
M = 3
plt.figure(figsize=(4, 4))
W = fit_gauss_func(X, T, M)
show_gauss_func(W)
plt.plot(X, T, marker='o', linestyle='None',
         color='cornflowerblue', markeredgecolor='black')
plt.xlim([X_min, X_max])
plt.grid(True)
mse = mse_gauss_func(X, T, W)
print("SD={0:.2f} cm".format(np.sqrt(mse)))
plt.show()
```

```
SD=4.37 cm
```



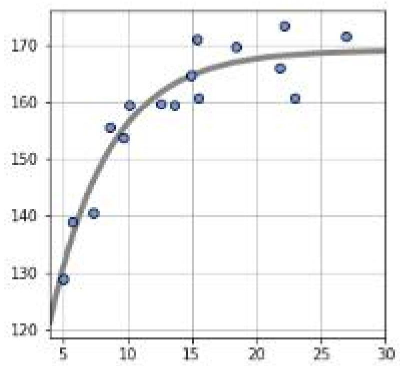
P219 ~ 221

```python
# 리스트 5-2-(17)
# 모델 A ------------------------------------
def model_A(x, w):
    y = w[0] - w[1] * np.exp(-w[2] * x)
    return y


# 모델 A 표시 -------------------------------
def show_model_A(w):
    xb = np.linspace(X_min, X_max, 100)
    y = model_A(xb, w)
    plt.plot(xb, y, c=[.5, .5, .5], lw=4)


# 모델 A의 MSE -----------------------------
def mse_model_A(w, x, t):
    y = model_A(x, w)
    mse = np.mean((y - t)**2)
    return mse
```

```python
# 리스트 5-2-(18)
from scipy.optimize import minimize


# 모델 A의 매개 변수 최적화 ------------------
def fit_model_A(w_init, x, t):
    res1 = minimize(mse_model_A, w_init, args=(x, t), method="powell")
    return res1.x
```

```python
# 리스트 5-2-(19)
# 메인 ------------------------------------
plt.figure(figsize=(4, 4))
W_init=[100, 0, 0]
W = fit_model_A(W_init, X, T)
print("w0={0:.1f}, w1={1:.1f}, w2={2:.1f}".format(W[0], W[1], W[2]))
show_model_A(W)
plt.plot(X, T, marker='o', linestyle='None',
         color='cornflowerblue',markeredgecolor='black')
plt.xlim(X_min, X_max)
plt.grid(True)
mse = mse_model_A(W, X, T)
print("SD={0:.2f} cm".format(np.sqrt(mse)))
plt.show()
```

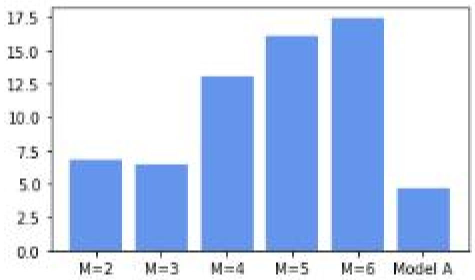
```
w0=169.0, w1=113.7, w2=0.2
SD=3.86 cm
```



P222

```
def kfold_model_A(x, t, k):
    n = len(x)
    mse_train = np.zeros(k)
    mse_test = np.zeros(k)
    for i in range(0, k):
        x_train = x[np.fmod(range(n), k) != i]
        t_train = t[np.fmod(range(n), k) != i]
        x_test = x[np.fmod(range(n), k) == i]
        t_test = t[np.fmod(range(n), k) == i]
        wm = fit_model_A(np.array([169, 113, 0.2]), x_train, t_train)
        mse_train[i] = mse_model_A(wm, x_train, t_train)
        mse_test[i] = mse_model_A(wm, x_test, t_test)
    return mse_train, mse_test

K = 16
Cv_A_train, Cv_A_test = kfold_model_A(X, T, K)
mean_A_test = np.sqrt(np.mean(Cv_A_test))
print("Gauss(M=3) SD={0:.2f} cm".format(mean_Gauss_test[1]))
print("Model A SD={0:.2f} cm".format(mean_A_test))
SD = np.append(mean_Gauss_test[0:5], mean_A_test)
M = range(6)
label = ["M=2", "M=3", "M=4", "M=5", "M=6", "Model A"]
plt.figure(figsize=(5, 3))
plt.bar(M, SD, tick_label=label, align="center",
facecolor="cornflowerblue")
plt.show()
```

```
Gauss(M=3) SD=6.51 cm
Model A SD=4.72 cm
```



<소감>

머신러닝에대해 본격적으로 실습할 수 있는 구간 이었다. 특히 회귀와 분류의 상관관계를 알 수 있었고 5장에서는 회귀에대해서 본격으로 배울 수 있었다. 회귀란 입력에 대해 연속적인 값을 대응시키는 문제이다. 처음엔 1차원 입력 직선 모델로 인공데이터를 실습하였고 소수점 이하의 표시가 너무 길다면 반올림하는 np.round를 통해 깔끔하게 표시할 수 있음을 알았다. 직선 모델의 방정식또한 회귀를 통해 나타낼수 있었다. 제곱오차함수를 통해 평균 제곱 오차를 계산할 수 있었다. 특히 P174에서 경사 하강법을 보았는데 이것은 지형 위의 한 지점에 대응하여 기울기를 확인 하고 지형위의 한지점이 가장 감소하는 방향으로 w0과 w1을 조금만 진행하면 계속 반복을 통해 최종적으로 지형위의 한지점이 가장 작아지는 그릇의 바닥인 w0과 w1에 도착할 수 있었다. 특히 여기서 기울기를 계산하는 함수 dmse_line()을 통해 만들어줄 수 있었다. 그래서 경사하강법 fit_line_num(x,t)를 리스트에서 구해주면 mse_line을 최소화하는 w를 돌려주어 기울기 w를 갱신하며 갱신 단계의 폭이 되는 학습비율을 계산해 주었다.이처럼 J의 기울기만 구할 수 있다면, 최소 제곱법으로 극소 값을 구할 수 있다. 여기서 주의할점을 알았는ㄴ데 일반적으로 경사 하강법으로 구해지는 해는 어디까지나 극소값이며 전체의 최소값은 아니라는 것이다. 만약 J가 곳곳에 움푹 들어간 모양을 하고 있으면 최소 제곱법으로 초기값 근처의 함몰 지점 극솟값에 수렴하게 된다. 이렇듯 가장 J가 작아진 지점을 최솟값으로 채용하는 근사적인 방법을 생각할 수 있다.
그리고 2차원 입력면 모델을 만들 수 있다.난수를 고정시켜 리스트에 표시를하면 3차원 플롯의 그래프로 그릴 수 있다. 예를 들어 나이와 몸무게와 키의 인공데이터를 3차원 플롯 형태로 만들 수 있음을 알았다.데이터에 따라 면을 대응시키는 방법에서는

show_plane(ax, w)를 준비하여 ax라는 인수를 그래프의 id로 지정하여 평균제곱오차를 계산하는 함수 mse_plane()도 만들 수 있다. 그보다 큰 차원인 D차원의 선형회귀 모델을 나타내기위해 가로벡터를 계산하면 된다는 것도 알 았다. 선형 기저 함수모델은 직선보다 곡선 쪽이 데이터에 더 어 울린다. 곡선을 사용하면ㄴ 예측의 오차도 작아진다는 것을 알았다. 직선면에서는 범용성이 높은 선형 기저 함수 모델이 유용하다 는 것을 알았다.
문제점으로 오버피팅의 문제가 있었다. 최적의 M을 찾는 방법은 평균제곱오차와 그 제곱근인 SD는 M이 증가하면 점점 감소하는 경향이 있기 때문에 최적의 M을 찾는 기준이 안된다는 것을 알았다. 그래서 진전한 목적인 새 데이터에 대한 예측의 정확도를 생각하여 해결해준다. 어떤 비율로 테스트 데이터와 훈련데이터를 나누었냐에 따라 결과도 조금 달라지겠지만 홀드 아웃 검증을 통해 해결해나가는 것을 배웠다. 이렇게 경사하강법을 사용하여 수치적으로 w를 구하는 방법과 해석적으로 도출하는 방법을 이 챕터에서 배웠다. 함수의 최솟값 또는 최댓값을 구하는 최적화 문제,를 배웠고, 파이썬의 minimize함수를 사용하여 최적 매개변수를 구하는 것도 알았다.