Assignment1

April 9, 2019

기계학습 숙제 1

1. [linear algebra; 10p] 훈련집합이

$$x_1 = (2,1)^T, x_2 = (2,4)^T, x_3 = (4,1)^T, x_4 = (4,3)^T$$

이다.

A. 훈련집합을 축

$$(0,1)^T$$

과 축

$$(1,1)^T$$

으로 투영하고, 각 축에 투영된 훈련집합의 분산들을 구하세요.

sol> if projection b onto a is p,

$$p = \frac{aa^T}{a^Ta}b$$

- 훈련집합 $X = \{x_1, x_2, x_3, x_4\}$를 축 $(0,1)^T$ 으로 투영한 훈련집합은 $P = \{p_1, p_2, p_3, p_4\}$이라 위의 식으로 계산하면 다음과 같다.

$$p = \frac{(0,1)(0,1)^T}{(0,1)^T(0,1)}x$$

$$a = (0,1)(0,1)^T = \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}$$

$$b = (0,1)^T(0,1) = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = 1$$

$$\frac{a}{b} = \frac{(0,1)(0,1)^T}{(0,1)^T(0,1)} = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}$$

$$p_1 = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}x_1 = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}$$

$$p_2 = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}x_2 = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}2\\4\end{pmatrix} = \begin{pmatrix}0\\4\end{pmatrix}$$

$$p_3 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} x_3 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$p_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} x_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

위의 투영된 벡터를 기반으로 분산을 구하기 위해 공분산 행렬을 구하면
sol>

$$\mathrm{Cov}[X, X] = \mathrm{E}[(x - \mu)(x - \mu)^T]$$

- 훈련집합

$$X = \{p_1, p_2, p_3, p_4\}$$

의 평균 벡터를 구하면

$$\mu = \frac{1}{4}(p_1 + p_2 + p_3 + p_4)$$

$$a = p_1 + p_2 + p_3 + p_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 4 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 9 \end{pmatrix}$$

$$\therefore \mu = \frac{1}{4} \begin{pmatrix} 0 \\ 9 \end{pmatrix} = \begin{pmatrix} 0 \\ 9/4 \end{pmatrix}$$

- 위의 평균 벡터를 가지고 공분산 벡터를 구하려면

$$(p - \mu)(p - \mu)^T$$

의 계산을 아래와 같이 해야한다.

$$(p_1 - \mu) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 0 \\ -5/4 \end{pmatrix}$$

$$\therefore (p_1 - \mu)(p_1 - \mu)^T = \begin{pmatrix} 0 \\ -5/4 \end{pmatrix} \begin{pmatrix} 0 & -5/4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & (-5/4)^2 \end{pmatrix}$$

$$(p_2 - \mu) = \begin{pmatrix} 0 \\ 4 \end{pmatrix} - \begin{pmatrix} 0 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 7/4 \end{pmatrix}$$

$$\therefore (p_2 - \mu)(p_2 - \mu)^T = \begin{pmatrix} 0 \\ 7/4 \end{pmatrix} \begin{pmatrix} 0 & 7/4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & (7/4)^2 \end{pmatrix}$$

$$(p_3 - \mu) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 0 \\ -5/4 \end{pmatrix}$$

$$\therefore (p_3 - \mu)(p_3 - \mu)^T = \begin{pmatrix} 0 \\ -5/4 \end{pmatrix} \begin{pmatrix} 0 & -5/4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & (-5/4)^2 \end{pmatrix}$$

$$(p_4 - \mu) = \begin{pmatrix} 0 \\ 3 \end{pmatrix} - \begin{pmatrix} 0 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 3/4 \end{pmatrix}$$

$$\therefore (p_4-\mu)(p_4-\mu)^T = \begin{pmatrix}0\\3/4\end{pmatrix}\begin{pmatrix}0 & 3/4\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & (3/4)^2\end{pmatrix}$$

$$\mathrm{Cov}[X,X] = \frac{1}{3}\sum_{i=1}^{4}(p_i-\mu)(p_i-\mu)^T$$

$$a = \sum_{i=1}^{4}(p_i-\mu)(p_i-\mu)^T = \begin{pmatrix}0 & 0\\0 & (-5/4)^2\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & (7/4)^2\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & (-5/4)^2\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & (3/4)^2\end{pmatrix}$$

$$= \begin{pmatrix}0 & 0\\0 & 108/16\end{pmatrix}$$

$$\therefore \mathrm{Cov}[X,X] = \frac{1}{3}a = \begin{pmatrix}0 & 0\\0 & 9/4\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 2.25\end{pmatrix}$$

위의 시그마 연산 후에 4 개의 sample 공분산을 계산을 하는데 3으로 하는 이유는 모수 기대값은 모르기 때문에 그것으로 계산을 한다.

하지만 모수의 기대값을 알고 있다면 공분산을 구하면 아래와 같다

$$\therefore \mathrm{Cov}[X,X] = \frac{1}{4}a = \begin{pmatrix}0 & 0\\0 & 27/16\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 1.6875\end{pmatrix}$$

covariance wikipedia의 Calculating the sampel covariance 참조

위의 공분산 행렬 값을 Numpy로 구하면 다음과 같다.

```
In [1]: import numpy as np

        p1 = np.array([0,1])
        p2 = np.array([0,4])
        p3 = np.array([0,1])
        p4 = np.array([0,3])

        mean_vector = (p1+p2+p3+p4)/4

        print("mean vector projected onto (1,1)^T: \n{}".format(mean_vector))
        total_vector = np.array([p1, p2, p3, p4])
        t = total_vector.T
        print("transpose:\n{}".format(total_vector.T))
        cov = np.cov(t)
        print("covariance vector projected onto (1,1)^T: \n{}".format(cov))
```

```
mean vector projected onto (1,1)^T:
[0.   2.25]
transpose:
[[0 0 0 0]
 [1 4 1 3]]
covariance vector projected onto (1,1)^T:
[[0.   0.  ]
 [0.   2.25]]
```

reply) (0,1)^T의 축으로 투영한 벡터들은 위의 공분산 결과에서 확인 할수 있듯이 (x, y)의 이차원 벡터라고 할때, x축의 정보는 사라지고 y 축 정보만 남아 y축으로만 분산이 형성된다.

- 훈련집합 $X = \{x_1, x_2, x_3, x_4\}$를 축 $ (1,1)^T $ 으로 투영한 훈련집합은 $P = \{p_1, p_2, p_3, p_4\}$ 이라 위의 식으로 계산하면 다음과 같다.

$$p = \frac{(1,1)(1,1)^T}{(1,1)^T(1,1)}x$$

$$a = (1,1)(1,1)^T = \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix} = \begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}$$

$$b = (1,1)^T(1,1) = \begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix} = 2$$

$$\frac{a}{b} = \frac{(1,1)(1,1)^T}{(1,1)^T(1,1)} = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}$$

$$p_1 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}x_1 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}3\\3\end{pmatrix}$$

$$p_2 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}x_2 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\begin{pmatrix}2\\4\end{pmatrix} = \frac{1}{2}\begin{pmatrix}6\\6\end{pmatrix}$$

$$p_3 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}x_3 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\begin{pmatrix}4\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}5\\5\end{pmatrix}$$

$$p_4 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}x_4 = \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\begin{pmatrix}4\\3\end{pmatrix} = \frac{1}{2}\begin{pmatrix}7\\7\end{pmatrix}$$

위의 투영된 벡터를 기반으로 분산을 구하기 위해 공분산 행렬을 구하면
sol>

$$\mathrm{Cov}[X, X] = \mathrm{E}[(x-\mu)(x-\mu)^T]$$

- 훈련집합 $X = \{p_1, p_2, p_3, p_4\}$의 평균 벡터를 구하면

$$\mu = \frac{1}{4}(p_1 + p_2 + p_3 + p_4)$$

$$a = p_1 + p_2 + p_3 + p_4 = \frac{1}{2}\begin{pmatrix}3\\3\end{pmatrix} + \frac{1}{2}\begin{pmatrix}6\\6\end{pmatrix} + \frac{1}{2}\begin{pmatrix}5\\5\end{pmatrix} + \frac{1}{2}\begin{pmatrix}7\\7\end{pmatrix} = \frac{1}{2}\begin{pmatrix}21\\21\end{pmatrix}$$

$$\therefore \mu = \frac{1}{4} \times \frac{1}{2}\begin{pmatrix}21\\21\end{pmatrix} = \frac{1}{8}\begin{pmatrix}21\\21\end{pmatrix}$$

- 위의 평균 벡터를 가지고 공분산 벡터를 구하려면

$$(p-\mu)(p-\mu)^T$$

의 계산을 아래와 같이 해야한다.

$$(p_1 - \mu) = \frac{1}{2}\begin{pmatrix}3\\3\end{pmatrix} - \frac{1}{8}\begin{pmatrix}21\\21\end{pmatrix} = \begin{pmatrix}-9/8\\-9/8\end{pmatrix}$$

$$\therefore (p_1 - \mu)(p_1 - \mu)^T = \begin{pmatrix}-9/8\\-9/8\end{pmatrix}\begin{pmatrix}-9/8 & -9/8\end{pmatrix} = \begin{pmatrix}(-9/8)^2 & (-9/8)^2\\(-9/8)^2 & (-9/8)^2\end{pmatrix}$$

$$(p_2 - \mu) = \frac{1}{2}\begin{pmatrix}6\\6\end{pmatrix} - \frac{1}{8}\begin{pmatrix}21\\21\end{pmatrix} = \begin{pmatrix}3/8\\3/8\end{pmatrix}$$

$$\therefore (p_2 - \mu)(p_2 - \mu)^T = \begin{pmatrix}3/8\\3/8\end{pmatrix}\begin{pmatrix}3/8 & 3/8\end{pmatrix} = \begin{pmatrix}(3/8)^2 & (3/8)^2\\(3/8)^2 & (3/8)^2\end{pmatrix}$$

$$(p_3 - \mu) = \frac{1}{2}\begin{pmatrix}5\\5\end{pmatrix} - \frac{1}{8}\begin{pmatrix}21\\21\end{pmatrix} = \begin{pmatrix}-1/8\\-1/8\end{pmatrix}$$

$$\therefore (p_3 - \mu)(p_3 - \mu)^T = \begin{pmatrix}-1/8\\-1/8\end{pmatrix}\begin{pmatrix}-1/8 & -1/8\end{pmatrix} = \begin{pmatrix}(-1/8)^2 & (-1/8)^2\\(-1/8)^2 & (-1/8)^2\end{pmatrix}$$

$$(p_4 - \mu) = \frac{1}{2}\begin{pmatrix}7\\7\end{pmatrix} - \frac{1}{8}\begin{pmatrix}21\\21\end{pmatrix} = \begin{pmatrix}7/8\\7/8\end{pmatrix}$$

$$\therefore (p_4 - \mu)(p_4 - \mu)^T = \begin{pmatrix}7/8\\7/8\end{pmatrix}\begin{pmatrix}7/8 & 7/8\end{pmatrix} = \begin{pmatrix}(7/8)^2 & (7/8)^2\\(7/8)^2 & (7/8)^2\end{pmatrix}$$

$$\mathrm{Cov}[X, X] = \frac{1}{3}\sum_{i=1}^{4}(p_i - \mu)(p_i - \mu)^T$$

$$a = \sum_{i=1}^{4}(p_i - \mu)(p_i - \mu)^T =$$

$$\begin{pmatrix}(-9/8)^2 & (-9/8)^2\\(-9/8)^2 & (-9/8)^2\end{pmatrix} + \begin{pmatrix}(3/8)^2 & (3/8)^2\\(3/8)^2 & (3/8)^2\end{pmatrix} + \begin{pmatrix}(-1/8)^2 & (-1/8)^2\\(-1/8)^2 & (-1/8)^2\end{pmatrix} + \begin{pmatrix}(7/8)^2 & (7/8)^2\\(7/8)^2 & (7/8)^2\end{pmatrix}$$

$$= \begin{pmatrix}140/64 & 140/64\\140/64 & 140/64\end{pmatrix}$$

$$\therefore \mathrm{Cov}[X, X] = \frac{1}{3}a = \begin{pmatrix}35/48 & 35/48\\35/48 & 35/48\end{pmatrix} = \begin{pmatrix}0.7291666 & 0.7291666\\0.7291666 & 0.7291666\end{pmatrix}$$

위의 시그마 연산 후에 4 개의 sample 공분산을 계산을 하는데 3으로 하는 이유는 모수 기대값은 모르기 때문에 그것으로 계산을 한다.

하지만 모수의 기대값을 알고 있다면 공분산을 구하면 아래와 같다

$$\therefore \mathrm{Cov}[X, X] = \frac{1}{4}a = \begin{pmatrix}35/64 & 35/64\\35/64 & 35/64\end{pmatrix} = \begin{pmatrix}0.546875 & 0.546875\\0.546875 & 0.546875\end{pmatrix}$$

covariance wikipedia의 Calculating the sampel covariance 참조

위의 공분산 행렬 값을 Numpy로 구하면 다음과 같다.

```
In [2]: import numpy as np

        p1 = np.array([3/2,3/2])
        p2 = np.array([3,3])
        p3 = np.array([5/2,5/2])
        p4 = np.array([7/2,7/2])

        mean_vector = (p1+p2+p3+p4)/4

        print("mean vector projected onto (1,1)^T: \n{}".format(mean_vector))
        total_vector = np.array([p1, p2, p3, p4])
        t = total_vector.T
        print("transpose:\n{}".format(total_vector.T))
        cov = np.cov(t)
        print("covariance vector projected onto (1,1)^T: \n{}".format(cov))
```

```
mean vector projected onto (1,1)^T:
[2.625 2.625]
transpose:
[[1.5 3.  2.5 3.5]
 [1.5 3.  2.5 3.5]]
covariance vector projected onto (1,1)^T:
[[0.72916667 0.72916667]
 [0.72916667 0.72916667]]
```

reply) 위의 공분산 결과에서 확인 할수 있듯이 (x, y) 의 이차원 벡터라고 할때, x=y의 선인 (1,1)^T 로 투영한 벡터들은 일직선상에 존재하기 때문에 분산 값이 일정하다

B. 해당 훈련집합의 공분산 행렬을 구하세요.

sol>

$$\mathrm{Cov}[X, X] = \mathrm{E}[(x-\mu)(x-\mu)^T]$$

- 훈련집합 $X = \{x_1, x_2, x_3, x_4\}$ 의 평균 벡터를 구하면

$$\mu = \frac{1}{4}(x_1 + x_2 + x_3 + x_4)$$

$$a = x_1 + x_2 + x_3 + x_4 = \begin{pmatrix}2\\1\end{pmatrix} + \begin{pmatrix}2\\4\end{pmatrix} + \begin{pmatrix}4\\1\end{pmatrix} + \begin{pmatrix}4\\3\end{pmatrix} = \begin{pmatrix}12\\9\end{pmatrix}$$

$$\therefore \mu = \frac{1}{4}\begin{pmatrix}12\\9\end{pmatrix} = \begin{pmatrix}3\\9/4\end{pmatrix}$$

- 위의 평균 벡터를 가지고 공분산 벡터를 구하려면

$$(x-\mu)(x-\mu)^T$$

의 계산을 아래와 같이 해야한다.

$$(x_1 - \mu) = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 9/4 \end{pmatrix} = \begin{pmatrix} -1 \\ -5/4 \end{pmatrix}$$

$$\therefore (x_1 - \mu)(x_1 - \mu)^T = \begin{pmatrix} -1 \\ -5/4 \end{pmatrix} \begin{pmatrix} -1 & -5/4 \end{pmatrix} = \begin{pmatrix} 1 & 5/4 \\ 5/4 & (5/4)^2 \end{pmatrix}$$

$$(x_2 - \mu) = \begin{pmatrix} 2 \\ 4 \end{pmatrix} - \begin{pmatrix} 3 \\ 9/4 \end{pmatrix} = \begin{pmatrix} -1 \\ 7/4 \end{pmatrix}$$

$$\therefore (x_2 - \mu)(x_2 - \mu)^T = \begin{pmatrix} -1 \\ 7/4 \end{pmatrix} \begin{pmatrix} -1 & 7/4 \end{pmatrix} = \begin{pmatrix} 1 & -7/4 \\ -7/4 & (7/4)^2 \end{pmatrix}$$

$$(x_3 - \mu) = \begin{pmatrix} 4 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 1 \\ -5/4 \end{pmatrix}$$

$$\therefore (x_3 - \vec{\mu})(x_3 - \mu)^T = \begin{pmatrix} 1 \\ -5/4 \end{pmatrix} \begin{pmatrix} 1 & -5/4 \end{pmatrix} = \begin{pmatrix} 1 & -5/4 \\ -5/4 & (-5/4)^2 \end{pmatrix}$$

$$(x_4 - \mu) = \begin{pmatrix} 4 \\ 3 \end{pmatrix} - \begin{pmatrix} 3 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3/4 \end{pmatrix}$$

$$\therefore (x_4 - \mu)(x_4 - \mu)^T = \begin{pmatrix} 1 \\ 3/4 \end{pmatrix} \begin{pmatrix} 1 & 3/4 \end{pmatrix} = \begin{pmatrix} 1 & 3/4 \\ 3/4 & (3/4)^2 \end{pmatrix}$$

$$\mathrm{Cov}[X, X] = \frac{1}{3} \sum_{i=1}^{4} (x_i - \mu)(x_i - \mu)^T$$

$$a = \sum_{i=1}^{4} (x_i - \mu)(x_i - \mu)^T = \begin{pmatrix} 1 & 5/4 \\ 5/4 & (5/4)^2 \end{pmatrix} + \begin{pmatrix} 1 & -7/4 \\ -7/4 & (7/4)^2 \end{pmatrix} + \begin{pmatrix} 1 & -5/4 \\ -5/4 & (-5/4)^2 \end{pmatrix} + \begin{pmatrix} 1 & 3/4 \\ 3/4 & (3/4)^2 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & -1 \\ -1 & 108/16 \end{pmatrix}$$

$$\therefore \mathrm{Cov}[X, X] = \frac{1}{3} a = \begin{pmatrix} 4/3 & -1/3 \\ -1/3 & 9/4 \end{pmatrix} = \begin{pmatrix} 1.3333333 & -0.33333 \\ -0.333333 & 2.25 \end{pmatrix}$$

위의 시그마 연산 후에 4 개의 sample 공분산을 계산을 하는데 3으로 하는 이유는 모수 기대값은 모르기 때문에 그것으로 계산을 한다.

하지만 모수의 기대값을 알고 있다면 공분산을 구하면 아래와 같다

$$\therefore \mathrm{Cov}[X, X] = \frac{1}{4} a = \begin{pmatrix} 1 & -1/4 \\ -1/4 & 27/16 \end{pmatrix} = \begin{pmatrix} 1 & -0.25 \\ -0.25 & 1.6875 \end{pmatrix}$$

covariance wikipedia의 Calculating the sampel covariance 참조

위의 공분산 행렬 값을 Numpy로 구하면 다음과 같다.

```
In [3]: import numpy as np

        p1 = np.array([2,1])
        p2 = np.array([2,4])
        p3 = np.array([4,1])
        p4 = np.array([4,3])

        mean_vector = (p1+p2+p3+p4)/4

        print("mean vector: \n{}".format(mean_vector))
        total_vector = np.array([p1, p2, p3, p4])
        t = total_vector.T
        print("transpose:\n{}".format(total_vector.T))
        cov = np.cov(t)
        print("covariance vector:\n{}".format(cov))
```

```
mean vector:
[3.   2.25]
transpose:
[[2 2 4 4]
 [1 4 1 3]]
covariance vector:
[[ 1.33333333 -0.33333333]
 [-0.33333333  2.25      ]]
```

reply) 훈련집합이

$$x_1 = (2,1)^T, x_2 = (2,4)^T, x_3 = (4,1)^T, x_4 = (4,3)^T$$

의 공분산 행렬로부터 분포는 x축으로 분포보다 y축으로 넓게 분포하고 x=y 의 선상에서 적게 분포되어 있다.

C. 구한 공분산 행렬을 고유분해를 구하세요.

sol>

$$Ax - \lambda x = (A - \lambda I)x = 0$$

- 위에서 구한 모수의 기대값을 알고 있는 공분산 행렬를 이용하여 고유벡터(eigenvector)와 고유값(eigenvalue)를 계산해보자.(we don't know the population but we only know Sample mean and covariance - wikipedia)

$$\begin{pmatrix} 1 & -1/4 \\ -1/4 & 27/16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -1/4 \\ -1/4 & 27/16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

$$\left(\begin{pmatrix} 1 & -1/4 \\ -1/4 & 27/16 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

$$\begin{pmatrix} 1-\lambda & -1/4 \\ -1/4 & 27/16-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

위의

$$A - \lambda I$$

인

2차 행렬의 determinant식을 세우면 아래와 같다.

$$\begin{vmatrix} 1-\lambda & -1/4 \\ -1/4 & 27/16-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)(27/16-\lambda) - (-1/4)(-1/4) = 0$$

위의 식을 전개하면

$$\lambda^2 - 43/16\lambda + 26/16 = 0$$

위의 식 양변에 16을 곱하면

$$16\lambda^2 - 43\lambda + 26 = 0$$

위의 식에서 람다를 위해 아래의 근의 공식(quadratic formula)으로 구하면

$$quadratic = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\lambda = \frac{43 \pm \sqrt{43^2 - 4 * 16 * 26}}{2 * 16} = \frac{43 \pm \sqrt{1849 - 1664}}{2 * 16} = \frac{43 \pm \sqrt{185}}{32}$$

$$\therefore \lambda_1 = \frac{43 + \sqrt{185}}{32}, \lambda_2 = \frac{43 - \sqrt{185}}{32}$$

이 고유값으로 다시

$$A - \lambda I$$

에 대하여

$$\lambda_1$$

을 대입한다.

$$\lambda_1$$

의 경우에는 아래와 같다

$$\begin{pmatrix} 1-\lambda_1 & -1/4 \\ -1/4 & 27/16-\lambda_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-\frac{43+\sqrt{185}}{32} & -1/4 \\ -1/4 & 27/16-\frac{43+\sqrt{185}}{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{11+\sqrt{185}}{32} & -1/4 \\ -1/4 & \frac{11+\sqrt{185}}{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

$$\lambda_2$$

의 경우에는 아래와 같다

$$\begin{pmatrix} 1-\lambda_2 & -1/4 \\ -1/4 & 27/16-\lambda_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-\frac{43-\sqrt{185}}{32} & -1/4 \\ -1/4 & 27/16-\frac{43-\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$= \begin{pmatrix} -\frac{11-\sqrt{185}}{32} & -1/4 \\ -1/4 & \frac{11-\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

- $$\lambda_1$$

  인 경우로 계산을 해보면

$$\begin{pmatrix} 1-\lambda_1 & -1/4 \\ -1/4 & 27/16-\lambda_1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-\frac{43+\sqrt{185}}{32} & -1/4 \\ -1/4 & 27/16-\frac{43+\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$= \begin{pmatrix} -\frac{11+\sqrt{185}}{32} & -1/4 \\ -1/4 & \frac{11+\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

  양면에 먼저 -4를 곱한다.

$$\begin{pmatrix} \frac{11+\sqrt{185}}{8} & 1 \\ 1 & -\frac{11+\sqrt{185}}{8} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

  위의 행렬의 Null space를 구하기 위해 가우스 소거(Gauss Elimination)을 하면

$$row2 = row2 - \frac{8}{11+\sqrt{185}}row1$$

를 적용하면

$$\begin{pmatrix} 1 & \frac{8}{11+\sqrt{185}} \\ 0 & \frac{185+11\sqrt{185}}{44+4\sqrt{185}} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$
$$x_1 + \frac{8}{11+\sqrt{185}}x_2 = 0$$

  역시

$$x_2$$

는 free variable이므로 임의값 1을 넣으면

$$\lambda_1$$

에 대한 고유(eigenvevtor)는 아래와 같다.

$$X_{\lambda_1} = \begin{pmatrix} -\frac{8}{11+\sqrt{185}} \\ 1 \end{pmatrix}$$

- $$\lambda_2$$

  인 경우에도 동일한 계산 과정이 나온다.

$$\begin{pmatrix} 1-\lambda_2 & -1/4 \\ -1/4 & 27/16-\lambda_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-\frac{43-\sqrt{185}}{32} & -1/4 \\ -1/4 & 27/16-\frac{43-\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$= \begin{pmatrix} -\frac{11-\sqrt{185}}{32} & -1/4 \\ -1/4 & \frac{11-\sqrt{185}}{32} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

  양면에 먼저 -4를 곱한다.

$$\begin{pmatrix} \frac{11-\sqrt{185}}{8} & \frac{8}{11-\sqrt{185}} \\ 1 & -\frac{11-\sqrt{185}}{8} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$

  위의 행렬의 Null space를 구하기 위해 가우스 소거(Gauss Elimination)을 하면

$$row2 = row2 - \frac{8}{11-\sqrt{185}}row1$$

를 적용하면

$$\begin{pmatrix} 1 & 1 \\ 0 & \frac{185-11\sqrt{185}}{44-4\sqrt{185}} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$$
$$x_1 + \frac{8}{11-\sqrt{185}}x_2 = 0$$

  역시

$$X_2$$

는 free variable이므로 임의값 1을 넣으면

$$\lambda_1$$

에 대한 고유(eigenvevtor)는 아래와 같다.

$$X_{\lambda_2} = \begin{pmatrix} -\frac{8}{11-\sqrt{185}} \\ 1 \end{pmatrix}$$

$$\therefore \mathrm{Cov}[X, X]$$

의 고유분해를 하면

$$\mathrm{Cov}[X, X] = \left(X_{\lambda_1}, X_{\lambda_2}\right)\begin{pmatrix} \frac{11+\sqrt{185}}{8} & 0 \\ 0 & \frac{11-\sqrt{185}}{8} \end{pmatrix}\left(X_{\lambda_1}, X_{\lambda_2}\right)^T$$

$$\mathrm{Cov}[X, X] = \begin{pmatrix} -\frac{8}{11+\sqrt{185}} & -\frac{8}{11-\sqrt{185}} \\ 1 & 1 \end{pmatrix}\begin{pmatrix} \frac{11+\sqrt{185}}{8} & 0 \\ 0 & \frac{11-\sqrt{185}}{8} \end{pmatrix}\begin{pmatrix} -\frac{8}{11+\sqrt{185}} & 1 \\ -\frac{8}{11-\sqrt{185}} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} -1 & -1 \\ \frac{11+\sqrt{185}}{8} & \frac{11-\sqrt{185}}{8} \end{pmatrix}\begin{pmatrix} -\frac{8}{11+\sqrt{185}} & 1 \\ -\frac{8}{11-\sqrt{185}} & 1 \end{pmatrix} = \begin{pmatrix} -11/4 & -2 \\ -2 & 11/4 \end{pmatrix}$$

- 위의 과정 처럼 고유분해를 하게 되면 복잡하여 지금부터는 python의 Numpy를 이용한다.

```
In [4]: import numpy as np

        cov = np.array([[1, -1/4],[-1/4, 27/16]])
        print("Cov(X,X): \n{}".format(cov))

        w, v  = np.linalg.eig(cov)
        print("After decomposition into eigenvalue and eigenvector")
        print("eigenvalue_1 : {}, eigenvalue_2: {}".format(w[0], w[1]))
        print("eigenvector_1: \n{}\neigenvector_2: \n{}".format(v[0], v[1]))
```

```
Cov(X,X):
[[ 1.     -0.25  ]
 [-0.25    1.6875]]
After decomposition into eigenvalue and eigenvector
eigenvalue_1 : 0.9187040466020173, eigenvalue_2: 1.7687959533979827
eigenvector_1:
[-0.95098267  0.30924417]
eigenvector_2:
[-0.30924417 -0.95098267]
```

D. 고유분해의 가장 큰 고유값을 가지는 고유벡터를 축으로 훈련집합을 투영한 결과를 그리고, 투영된 훈련 집합의 분산을 구하세요.

```
In [5]: x1 = np.array([2,1])
        x2 = np.array([2,4])
        x3 = np.array([4,1])
        x4 = np.array([4,3])

        mean_vector = (x1+x2+x3+x4)/4

        print("mean vector: \n{}".format(mean_vector))
        total_vector = np.array([p1, p2, p3, p4])
        t = total_vector.T
        print("transpose:\n{}".format(total_vector.T))
        cov = np.cov(t)
        print("covariance vector:\n{}".format(cov))

        bigone = None

        if w[0] > w[1]:
            bigone = 0
        else:
            bigone = 1

        print("the eigenvector which has the bigest eigenvalue: {}".format(bigone))
        print(v[bigone])
```

```
mean vector:
[3.   2.25]
transpose:
[[2 2 4 4]
 [1 4 1 3]]
covariance vector:
[[ 1.33333333 -0.33333333]
 [-0.33333333  2.25      ]]
the eigenvector which has the bigest eigenvalue: 1
[-0.30924417 -0.95098267]
```

a,p는 colum vector

$$p = \frac{aa^T}{a^Ta}b$$

위의 식을 계산하기 위한 projection matrix

$$\frac{aa^T}{a^Ta}$$

을 구하면 아래와 같다.

```
In [6]: outer = np.outer(v[bigone],v[bigone])
        inner = np.dot(v[bigone],v[bigone])

        print("outer:\n{}".format(outer))
        print("inner:\n{}".format(inner))

        fraction = outer/inner
        print("Projection matrix:\n{}".format(fraction))
```

```
outer:
[[0.09563196 0.29408585]
 [0.29408585 0.90436804]]
inner:
1.0000000000000002
Projection matrix:
[[0.09563196 0.29408585]
 [0.29408585 0.90436804]]
```

- 훈련집합 $X = \{x_1, x_2, x_3, x_4\}$를 축 eigenvalue가 가장 큰 eigenvector로 투영한 훈련집합은 $P = \{p_1, p_2, p_3, p_4\}$이라 위의 식으로 계산하면 다음과 같다.

```
In [7]: p1 = np.dot(fraction, x1)
        p2 = np.dot(fraction, x2)
        p3 = np.dot(fraction, x3)
        p4 = np.dot(fraction, x4)
```

```
        print("p1:\n{}".format(p1))
        print("p2:\n{}".format(p2))
        print("p3:\n{}".format(p3))
        print("p4:\n{}".format(p4))
```

```
p1:
[0.48534976 1.49253974]
p2:
[1.36760731 4.20564387]
p3:
[0.67661368 2.08071144]
p4:
[1.26478538 3.88944752]
```

```
In [8]: ## 위의 투영된 공분산의 벡터를 구하면
        mean_vector = (p1+p2+p3+p4)/4

        print("mean vector: \n{}".format(mean_vector))
        total_vector = np.array([p1, p2, p3, p4])
        t = total_vector.T
        print("transpose:\n{}".format(total_vector.T))
        cov = np.cov(t)
        print("covariance vector: \n{}".format(cov))
```

```
mean vector:
[0.94858903 2.91708564]
transpose:
[[0.48534976 1.36760731 0.67661368 1.26478538]
 [1.49253974 4.20564387 2.08071144 3.88944752]]
covariance vector:
[[0.18803922 0.57825516]
 [0.57825516 1.77824092]]
```

E. 위의 A번과 D번에서 구한 분산을 비교하세요
reply) A번의 $(0,1)^T$로 투영한 훈련집합 분산(numpy cov)은

$$\begin{pmatrix} 0 & 0 \\ 0 & 2.25 \end{pmatrix}$$

와 $(1,1)^T$로 투영한 훈련집합 분산(numpy cov)은

$$\begin{pmatrix} 0.7291666 & 0.7291666 \\ 0.7291666 & 0.7291666 \end{pmatrix}$$

이고 D에서 구한 분산(numpy cov)

$$\begin{pmatrix} 0.18803922 & 0.57825516 \\ 0.57825516 & 1.77824092 \end{pmatrix}$$

이다. D에서 구한 공분산 행렬은 전체적으로 분산이 크게 유지하지만 A에서 구한 분산은 한쪽으로 치우치거나 분산이 비슷하게 분포하는 경향이 있어 데이터의 분포를 설명하기에는 D보다 낮다.

2. [probability; 10p] 직원이 A 제조사로 부터 1000개의 직접회로 (IC)를, B제조사로부터 2000개의 IC를, C제조사로부터 3000개의 IC를 구매했다. IC의 불량 검사 결과, A사로부터 구매한 불량 확률은 0.05, B사로부터 구매한 IC의 불량 확률은 0.10, C사로부터 구매한 IC의 불량 확률은 0.01이었다.

A. 만약 3개의 제조사로부터 구매한 IC가 섞여 있는 경우, 임의로 선택한 IC가 불량일 확률은 얼마인가?

sol> A 제조사 1000개중 불량은 50개, B 제조사 2000개중 불량품 200개, C 제조사 3000개중 불량품은 300

총 불량품은 550개이고 전체 구매한 IC는 6000개이므로

P(IC가 불량)= 550/6000 = 0.0916666

B. 임의로 선택한 IC가 불량인 경우, 그것이 제조사 A로부터 만들어질 확률은 얼마인가>

sol> P(A|불량) = P(A∩불량)/P(불량)= (P(불량|A)×p(A))/P(불량) 이므로

$0.05 \times \frac{1}{6} \times \frac{1}{0.09166} = 0.0909$

3. [probability; 10p] K 대학은 대학원생보다 2배의 학부생이 재학중이다. 대학원생의 25%가 기숙사에 살고 있고, 학부생의 10%가 기숙사에 살고 있다.

A. 한 학생을 임의로 선정한 경우,그 학생이 기숙사에 살고 있는 학부생일 확률은 얼마인가?

대학원생의 수를 a라고 하면, 학부생의 수는 2a이다.

대학원생이면서 기숙사생활을 하는 대학원생 수는0.25a이고

학부생이면서 기숙사생활을 하는 학부생는0.2a이라고 하면

P(기숙사에 살고 있는 학부생일 확률) = 0.2a/3a = 1/15 = 0.0666

B. 기숙사에 살고 있는 한 학생을 임의로 선정한 경우, 그 학생이 대학원생일 확률은 얼마인가?

P(기숙사 살고 있는 대학원 기숙사생) = 0.25a/0.45a = 5/9 = 0.55

4. [numerical computation; 10p] 다음의 선형 최소 제곱 함수 f(x)= $\frac{1}{2}||Ax-b||_2^2$의 최소값을 Newton method를 이용하여 구하여라.

일반적으로 극점에 최소값을 가지므로 아래의 Newton method를 이용하여 구하면

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$$

$$f'(x) = (Ax-b)A$$
$$f''(x) = AA$$

$$X_{t+1} = X_t - \frac{f\prime(x_t)}{f''(x_t)}$$

$$X_{t+1} = X_t - \frac{Ax_t - b}{A}$$

위의 연산 결과에서

$X_t = 0$ 일때, $X_{t+1} = \frac{b}{A}$

$X_t = 1$ 일때, $X_{t+1} = 1 - \frac{(A-b)}{A} = \frac{b}{A}$

$\therefore X = \frac{b}{A}$가 **최소점**이므로 이를 통해 최소값을 구하면

$f(\frac{b}{A}) = \frac{1}{2}||A\frac{b}{A} - b||_2^2 = 0$ 이므로 **최소값은** 0 이다.

5. [numerical computation; 10p] 함수

$$f(x,y) = x^2 - 4x + y^2 + 2y$$

의 다음을 구하세요.

A. 함수의 최소점과 최솟값을 분석적으로 구하세요.
최소점과 최소값은 극점인 미분이해서 $f'(x,y) = 0$ 인 곳에서 나타나므로
f(x,y) 에대하여 각각 ∂x과 ∂y로 미분을 하여 0인 값을 구하면 아래와 같다.

$$\frac{\partial f(x,y)}{\partial x} = 2x - 4 = 0, \therefore x = 2$$

$$\frac{\partial f(x,y)}{\partial y} = 2y + 2 = 0, \therefore y = -1$$

이므로,
최소점을 나타내는 좌표는 $(2,-1)$ 이고, 최소값은 $f(2,-1) = 4 - 8 + 1 - 2 = -5$
B. 이동 비율이 0.1인 경사 하강법을 통해 초기점 (x,y) $= (5,5)^T$ 에서 이동 할 때 얻어지는 점들을 구하세요.(최소 3개 이상의 경로점)

$$\frac{\partial f(x,y)}{\partial x} = 2x - 4 = 0$$

$$\frac{\partial f(x,y)}{\partial y} = 2y + 2 = 0$$

이므로,

$$W_0 = (5,5)^T$$

에서

$$W_{a+1} = W_a - 0.1 * (2x - 4, 2y + 2)$$

이므로

$$W_1 = (5,5)^T - 0.1 * (6,12)^T = (4.4, 3.8)$$

$$W_2 = (4.4, 3.8) - 0.1 * (4.8, 9.6) = (3.92, 2.84)$$

$$W_3 = (3.92, 2.84) - 0.1 * (3.84, 7.68) = (3.536, 2.072)$$

C. 구한 경로점들이 분석적으로 구한 최소값으로 이동하는지 확인하세요.
이동을 추가적으로 더 구하면

$$W_4 = (3.536, 2.072) - 0.1 * (3.072, 6.144) = (3.2288, 1.4576)$$

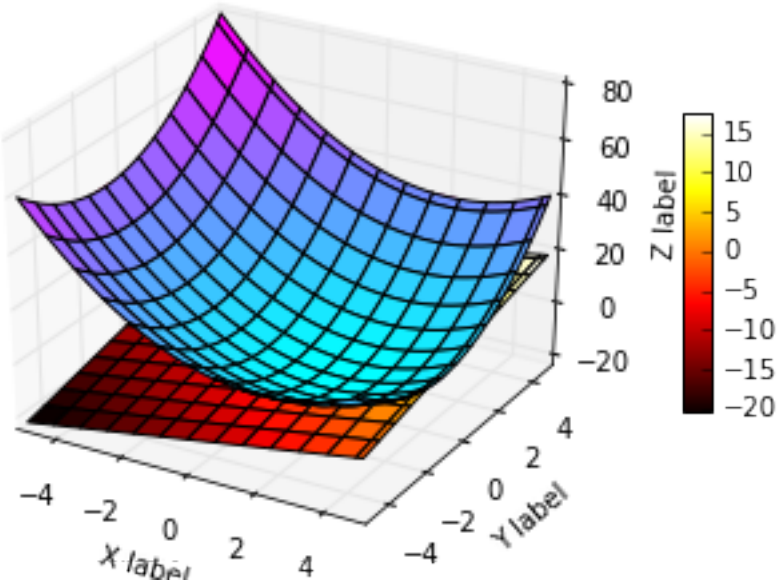
$$W_5 = (3.2288, 1.4576) - 0.1 * (2.4576, 4.9152) = (2.98304, 0.96608)$$

위의 보는 거와 같이 점차 x값은 5에서 2으로 y 값은 5에서 0로 떨어지면서 확인한 결과 점차 최소값으로 다가가고 있다.
D. 함수를 등고선 형태로 그리고, 함수의 미분 경사도 그리세요.

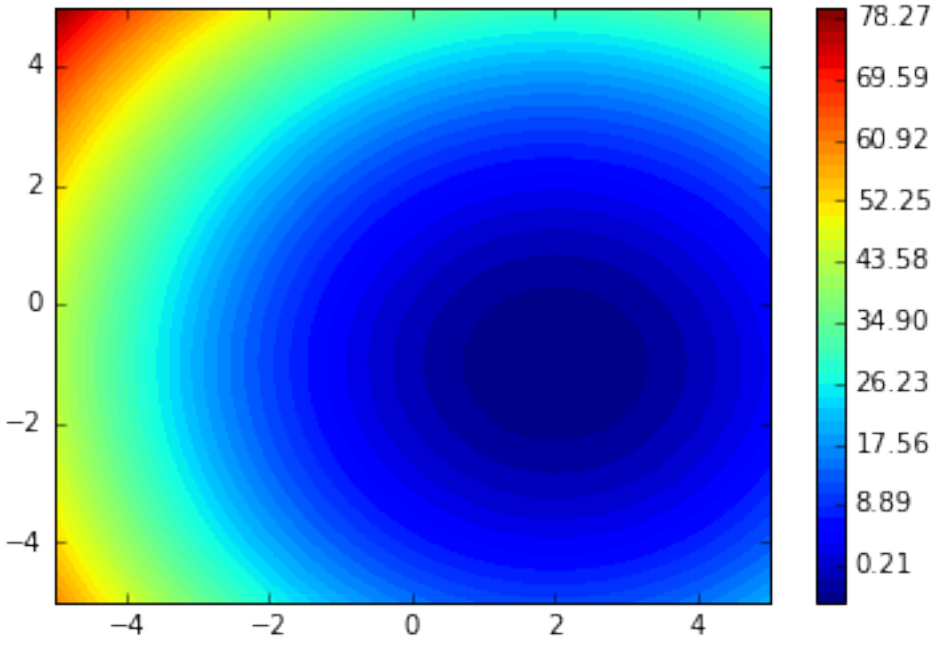
In [9]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

def f(x, y):
    return x**2 - 4*x +y**2 +2*y
def f_prime(x, y):
    return 2*x-4+2*y+2

x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x,y)

z = f(X, Y)
z_prime = f_prime(X,Y)


fig = plt.figure()
ax = fig.gca(projection="3d")
surf = ax.plot_surface(X, Y, z,
                       rstride = 4,
                       cstride = 4,
                       cmap = cm.cool,
                       linewidth = 1,
                       antialiased =True)

surf = ax.plot_surface(X, Y, z_prime,
                       rstride = 4,
                       cstride = 4,
                       cmap = cm.hot,
                       linewidth = 1,
                       antialiased =True)

ax.set_xlabel("X label")
ax.set_ylabel("Y label")
ax.set_zlabel("Z label")
fig.colorbar(surf, shrink=0.5, aspect=10)
cset = ax.contour(X,Y,z,100, cmap=cm.magma, LineWidth=2)
ax.clabel(cset, fontsize=10, inline=10)
plt.show()

cp = plt.contourf(X, Y, z, levels = np.linspace(z.reshape(-1,1).min(), z.reshape(-1,1)
plt.colorbar(cp)
```

```
/home/hyunyoung2/.local/lib/python3.5/site-packages/numpy/ma/core.py:6442: MaskedArrayFutureWa
  return self.reduce(a)
```

```
/home/hyunyoung2/.local/lib/python3.5/site-packages/numpy/ma/core.py:6442: MaskedArrayFutureWar
  return self.reduce(a)
```



Out[9]: <matplotlib.colorbar.Colorbar at 0x7facc425c2e8>



6. [numerical computation; 5p]

$$g_0 = w_0 + 2w_1 + 1, g_1 = 2w_0 + 3_1 1$$

인

$$f = (g_0 + 2g_1 1)^2 \text{의}$$

$$w_0, w_1$$

에 대한 편미분을 구하세요.

$$f = (g_0 + 2g_1 - 1)^2 = (5W_0 + 8W_1 - 2)^2$$

이므로

$$\frac{\partial f}{\partial W_0} = 10(5W_0 + 8W_1 - 2)$$

$$\frac{\partial f}{\partial W_1} = 16(5W_0 + 8W_1 - 2)$$

7. [PLA; 15p] PLA 가중치 갱신 법칙 w(t + 1) = w(t) + y(t)x(t)를 보고 다음 문제의 답을 보이세요.

A. $y(t)w^T(y)x(y) < 0$ 임을 보이세요. (Hint: x(t)는 w(t)에 의해 오분류 됨)

Perceptron Learning algrorithm에서
$y(t) = w^T(t)x(t)$ 에서
$y(t) > 0$ 일때, $w^T(t)x(t) > 0$이므로 $y(t)w^T(t)x(t) > 0$ 이다.
하지만 이를 x(t)가 w(t)의 오분류되었다고 하면 원래 $w^T(t)x(t) > 0$가 $w^T(t)x(t) < 0$로 변하여
$y(t)w^T(t)x(t) < 0$ 가 된다.
$y(t) < 0$ 일때, $w^T(t)x(t) < 0$이므로 $y(t)w^T(t)x(t) > 0$ 이다.
하지만 이를 x(t)가 w(t)의 오분류되었다고 하면 원래 $w^T(t)x(t) < 0$가 $w^T(t)x(t) > 0$로 변하여
$y(t)w^T(t)x(t) < 0$가 된다.
따라서 오분류되는 데이터에 대하여 $y(t)w^T(y)x(y)$ 는 0보다 작다
B.

$$y(t)w(t+1)x(t) > y(t)w(t)x(t)$$

임을 보이세요. (Hint: w(t +1) = w(t)+ y(t)x(t) 이용)
$w(t+1) = w(t) + y(t)x(t)$ 이므로
$y(t)w(t+1)x(t) > y(t)w(t)x(t)$ 의 식은
아래와 같이 변경될 수 있다.
$y(t)(w(t) + y(t)x(t))^T x(t) > y(t)w(t)x(t)$는
Transpose property 중에서
$(A+B)^T = A^T + B^T$ 와 $(AB)^T = B^T A^T$에 의해
위의 식은
$y(t)(w^T(t) + x^T(t)y^T(t))x(t) > y(t)w(t)x(t)$
$y(t)w^T(t)x(t) + y(t)x^T(t)y^T(t)x(t) > y(t)w(t)x(t)$
양변에 $y(t)w^T(t)x(t)$를 빼면
$y(t)x^T(t)y^T(t)x(t) > 0$이다.
그리고 PLA가 정상분류되었다면

$y(t)x^T(t) > 0$ 이고 $y^T(t)x(t) > 0$이므로
$y(t)x^T(t)y^T(t)x(t)$ 항상 0 크다.

$$\therefore y(t)w(t+1)x(t) > y(t)w(t)x(t)$$

은 성립한다.

C. w(t)에서 w(t + 1)로 이동하는 것이 x(t)를 분류하는데 올바른 방향으로 이동함을 설명하세요.
$y(t)w(t+1)x(t) > y(t)w(t)x(t)$ 에서
양변에 $y(t)w(t)x(t)$를 빼면
$w(t+1) > w(t)$이고
위의 식에서 $w(t)$를 양변에 빼면
$w(t+1) - w(t) > 0$이다.
위의 식에 $w(t+1) = w(t) + y(t)x(t)$ 대입하면
$y(t)x(t) > 0$이다.
$y(t)x(t) > 0$가 성립하기 위해서는 y(t)와 x(t)은 항상 같은 방향이여야 하므로
y(t)와 x(t)가 다른 방향 일때,
w(t+1)는 y(t)를 x(t)와 같은 방향으로 업데이트를 하기 때문에
w(t)에서 w(t + 1)로 이동하는 것이 x(t)를 분류하는 것은 올바른 방향으로 이동한다.

D. $w = [w_0, w_1, w_2]^T$이고, $x = [1, x_1, x_2]^T$인 $h(x) = sign(w^T x)$ 일 때, $h(x) = 1$와 $h(x) = 1$는 결정 직선$x_2 = ax_1 + b$에 의해 구분된다. 결정 직선의 기울기 a와 절편 b를 가중치 w_0, w_1, w_2에 의해 설명하세요.
$w^T = w_0 + w_1x_1 + w_2x_2$ 이므로
$h(x) = sign(w_0 + w_1x_1 + w_2x_2)$에서
$h(x) = 1$이면 w_0+w_1x_1 + w_2x_2 < 0 이고
$h(x) = 1$이면 w_0+w_1x_1 + w_2x_2 > 0 이므로
결정직선은 w_0+w_1x_1 + w_2x_2 = 0 이다.
위에서 결정직선 $x_2 = ax_1 + b$에 의해 $h(x) = 1$와 $h(x) = 1$는 구분되므로
∴ 기울기는 $a = \frac{-w_1}{w_2}$, 절편 $b = \frac{-w_0}{w_2}$ 이다.

8. [PLA; 15p] PLA는 결과적으로 선형적으로 분리를 하는 최적 가중치 w 에 얻게 된다. 다음 문제의 답을 보이세요. (w(0) = 0임을 가정)

A. $p = \min_{1<=n<=N} y_n(w^*x_n)$ 일 때, p > 0임을 보이세요.
$y(t) > 0$ 일때, $w^T(t)x(t) > 0$이므로 $y(t)w^T(t)x(t) > 0$ 이고
$y(t) < 0$ 일때, $w^T(t)x(t) < 0$이므로 $y(t)w^T(t)x(t) > 0$ 이다.
w는 최적 가중치로 위의 두개의 식을 만족해야 하므로
$y_n(w^*x_n) > 0$ 크다
$\therefore p > 0$

B. $w^T(t)w^* >= w^T(t-1)w^* + p$임을 보이고, 결과적으로 $w^T(t)w^* >= tp$임을 보이세요. (Hint: 귀납법 이용)
$w^T(t)w^* >= w^T(t-1)w^* + p$에서
양변에 $w^T(t-1)w^* + p$를 빼면
$(w^T(t) - w^T(t-1))w^* - p0$이고
$y(t-1)w^T(t)x(t-1) > y(t-1)w^T(t-1)x(t-1)$에 의해
$w^T(t) - w^T(t-1) > 0$이고
$p = \min_{1<=n<=N} y_n(w^*x_n)$ 일때, $p > 0$ 크다.
$p = \min_{1<=n<=N} y_n(w^*x_n)$ 일때, $p <= y(t-1)w^*x(t-1)$ 이므로 $w^* > 0$이다.
따라서 $(w^T(t) - w^T(t-1))w^* - p >= 0$는 성립한다.

$w^T(t)w^* - w^T(t-1)w^* - p >= 0$ 에서
$w^T(t)w^* - w^T(t-1)w^* >= p$ 이므로 $w^T(t)w^* tp$는 성립한다.
C. $||w(t)||^2 <= ||w(t-1)||^2 + ||x(t-1)||^2$ 임을 보이세요. (Hint: $x(t-1)$ 는 $w(t-1)$ 에 오분류 되어 $y(t1)(w^T(t-1)x(t-1)) <= 0$ 임을 활용)
$||w(t)||^2 > ||w(t-1)||^2 + ||x(t-1)||^2$ 이라고 하자.
$||w(t)||^2$에 $w(t) = w(t-1) + y(t-1)x(t-1)$를 대입하고
$||w(t-1) + y(t-1)x(t-1)||^2$이고, 이를 전개하면
$||w(t-1)||^2 + 2||w(t-1)y(t-1)x(t-1)|| + ||y(t-1)x(t-1)||^2$이다.
PLA에서 $y(t) = 1$ 또는 $y(t) = -1$ 이므로
$||w(t-1)||^2 + 2||w(t-1)y(t-1)x(t-1)|| + ||y(t-1)x(t-1)||^2$에
$y(t) = 1$ 또는 $y(t) = -1$를 대입하면
$||w(t)||^2 = ||w(t-1)||^2 + 2||w(t-1)x(t-1)|| + ||x(t-1)||^2$이다.
이를 $||w(t)||^2 > ||w(t-1)||^2 + ||x(t-1)||^2$에 대입하면
$||w(t-1)||^2 + 2||w(t-1)x(t-1)|| + ||x(t-1)||^2 > ||w(t-1)||^2 + ||x(t-1)||^2$ 이고
양변에 $||w(t-1)||^2 + ||x(t-1)||^2$를 빼면
$2||w(t-1)x(t-1)|| > 0$이다. 하지만
$x(t-1)$가 $w(t-1)$에 의해 오분류되면
$y(t-1)(w^T(t-1)x(t-1)) <= 0$이므로
$||w(t-1)x(t-1)|| > 0$ 모순이 되므로
$||w(t)||^2 <= ||w(t-1)||^2 + ||x(t-1)||^2$이다.

9. [LR; 15p] UCI의 wine quality 데이터를 활용하여 다음의 분석을 수행하세요.

(참고: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/)
A. 와인의 데이터의 일부를 출력하여 확인하세요.

In [10]:
```
import pandas as pd

red = pd.read_csv("winequality-red.csv", sep=';')
white = pd.read_csv("winequality-white.csv", sep=';')
data = pd.concat([red, white], axis=0)
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.head.ht
data.head(n=7)
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	
5	7.4	0.66	0.00	1.8	0.075	
6	7.9	0.60	0.06	1.6	0.069	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.9978	3.51	0.56	
1	25.0	67.0	0.9968	3.20	0.68	
2	15.0	54.0	0.9970	3.26	0.65	
3	17.0	60.0	0.9980	3.16	0.58	

```
4                11.0                  34.0   0.9978  3.51       0.56
5                13.0                  40.0   0.9978  3.51       0.56
6                15.0                  59.0   0.9964  3.30       0.46

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
5      9.4        5
6      9.4        5
```

B. 와인의 각 속성의 통계적 특성들 (평균, 분산, 최대값, 최소값 등)을 확인하세요.

```
In [11]: # https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describ
         data.describe()
```

Out[11]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	6497.000000	6497.000000	6497.000000	6497.000000	
mean	7.215307	0.339666	0.318633	5.443235	
std	1.296434	0.164636	0.145318	4.757804	
min	3.800000	0.080000	0.000000	0.600000	
25%	6.400000	0.230000	0.250000	1.800000	
50%	7.000000	0.290000	0.310000	3.000000	
75%	7.700000	0.400000	0.390000	8.100000	
max	15.900000	1.580000	1.660000	65.800000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	6497.000000	6497.000000	6497.000000	6497.000000	
mean	0.056034	30.525319	115.744574	0.994697	
std	0.035034	17.749400	56.521855	0.002999	
min	0.009000	1.000000	6.000000	0.987110	
25%	0.038000	17.000000	77.000000	0.992340	
50%	0.047000	29.000000	118.000000	0.994890	
75%	0.065000	41.000000	156.000000	0.996990	
max	0.611000	289.000000	440.000000	1.038980	

	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000
mean	3.218501	0.531268	10.491801	5.818378
std	0.160787	0.148806	1.192712	0.873255
min	2.720000	0.220000	8.000000	3.000000
25%	3.110000	0.430000	9.500000	5.000000
50%	3.210000	0.510000	10.300000	6.000000
75%	3.320000	0.600000	11.300000	6.000000
max	4.010000	2.000000	14.900000	9.000000

C. 와인의 12가지 속성 간의 상관관계를 heatmap 형태로 그려 분석하세요.

In [12]:
```
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.ht
data.corr() # for correlation
```

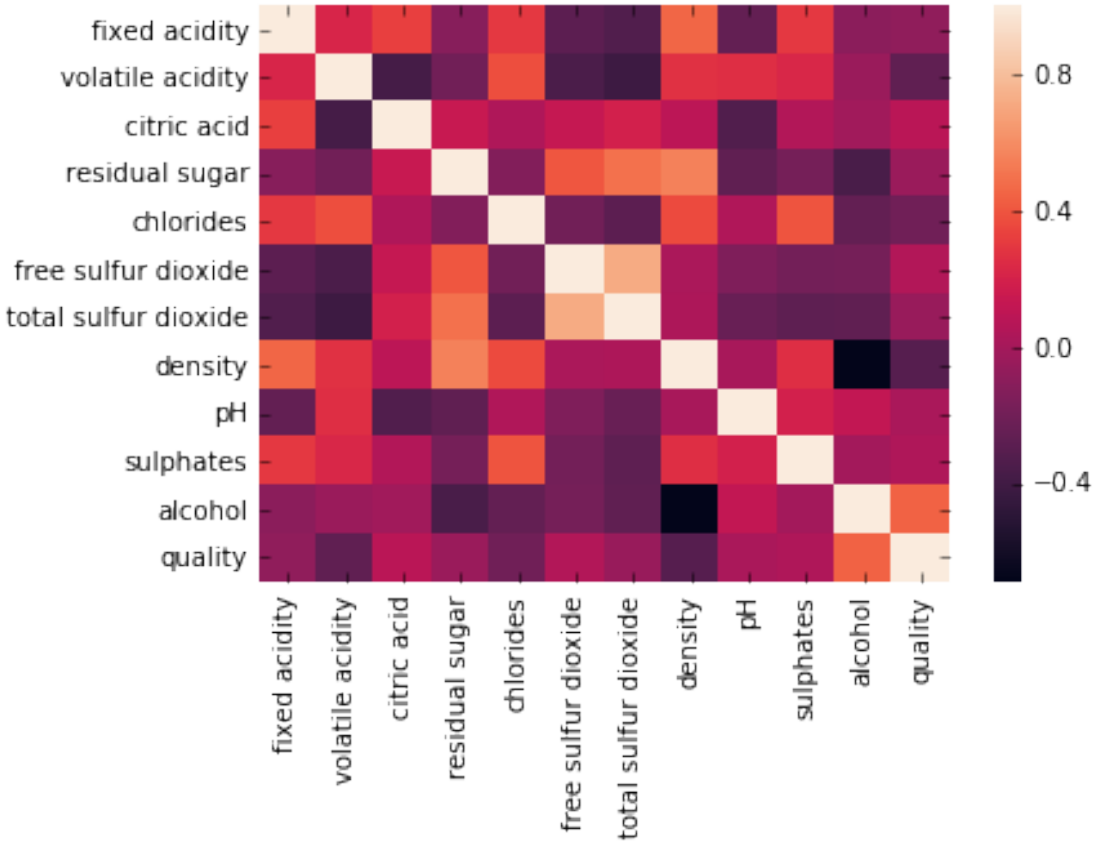
Out[12]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	0.219008	0.324436	-0.111981	0.298195	-0.282735	-0.329054	0.458910	-0.252700	0.299568	-0.095452	-0.076743
volatile acidity	0.219008	1.000000	-0.377981	-0.196011	0.377124	-0.352557	-0.414476	0.271296	0.261454	0.225984	-0.037640	-0.265699
citric acid	0.324436	-0.377981	1.000000	0.142451	0.038998	0.133126	0.195242	0.096154	-0.329808	0.056197	-0.010493	0.085532
residual sugar	-0.111981	-0.196011	0.142451	1.000000	-0.128940	0.402871	0.495482	0.552517	-0.267320	-0.185927	-0.359415	-0.036980
chlorides	0.298195	0.377124	0.038998	-0.128940	1.000000	-0.195045	-0.279630	0.362615	0.044708	0.395593	-0.256916	-0.200666
free sulfur dioxide	-0.282735	-0.352557	0.133126	0.402871	-0.195045	1.000000	0.720934	0.025717	-0.145854	-0.188457	-0.179838	0.055463
total sulfur dioxide	-0.329054	-0.414476	0.195242	0.495482	-0.279630	0.720934	1.000000	0.032395	-0.238413	-0.275727	-0.265740	-0.041385
density	0.458910	0.271296	0.096154	0.552517	0.362615	0.025717	0.032395	1.000000	0.011686	0.259478	-0.686745	-0.305858
pH	-0.252700	0.261454	-0.329808	-0.267320	0.044708	-0.145854	-0.238413	0.011686	1.000000	0.192123	0.121248	0.019506
sulphates	0.299568	0.225984	0.056197	-0.185927	0.395593	-0.188457	-0.275727	0.259478	0.192123	1.000000	-0.003029	0.038485
alcohol	-0.095452	-0.037640	-0.010493	-0.359415	-0.256916	-0.179838	-0.265740	-0.686745	0.121248	-0.003029	1.000000	0.476166
quality	-0.076743	-0.265699	0.085532	-0.036980	-0.200666	0.055463	-0.041385	-0.305858	0.019506	0.038485	0.476166	1.000000

```
citric acid           -0.010493  0.085532
residual sugar        -0.359415 -0.036980
chlorides             -0.256916 -0.200666
free sulfur dioxide   -0.179838  0.055463
total sulfur dioxide  -0.265740 -0.041385
density               -0.686745 -0.305858
pH                     0.121248  0.019506
sulphates             -0.003029  0.038485
alcohol                1.000000  0.444319
quality                0.444319  1.000000
```

In [13]:
```
# https://datascienceschool.net/view-notebook/4c2d5ff1caab4b21a708cc662137bc65/
# http://seaborn.pydata.org/generated/seaborn.heatmap.html
import seaborn as sns
%matplotlib inline
corr = data.corr()
sns.heatmap(corr,
            xticklabels=corr.columns,
            yticklabels=corr.columns)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7facb0ccf9b0>
```



```
In [14]: data.corr().head()
```

```
Out[14]:                      fixed acidity  volatile acidity  citric acid  \
         fixed acidity             1.000000          0.219008     0.324436
         volatile acidity          0.219008          1.000000    -0.377981
         citric acid               0.324436         -0.377981     1.000000
         residual sugar           -0.111981         -0.196011     0.142451
         chlorides                 0.298195          0.377124     0.038998

                                residual sugar  chlorides  free sulfur dioxide  \
         fixed acidity               -0.111981   0.298195            -0.282735
         volatile acidity            -0.196011   0.377124            -0.352557
         citric acid                  0.142451   0.038998             0.133126
         residual sugar               1.000000  -0.128940             0.402871
         chlorides                   -0.128940   1.000000            -0.195045

                                total sulfur dioxide   density        pH  sulphates  \
         fixed acidity                     -0.329054  0.458910 -0.252700   0.299568
         volatile acidity                  -0.414476  0.271296  0.261454   0.225984
         citric acid                        0.195242  0.096154 -0.329808   0.056197
         residual sugar                     0.495482  0.552517 -0.267320  -0.185927
         chlorides                         -0.279630  0.362615  0.044708   0.395593

                                 alcohol   quality
         fixed acidity         -0.095452 -0.076743
         volatile acidity      -0.037640 -0.265699
         citric acid           -0.010493  0.085532
         residual sugar        -0.359415 -0.036980
         chlorides             -0.256916 -0.200666
```

위의 heatmap을 보면 상관관계를 수치적으로 보여주는데 밝은색일수록 양의 상관관계를 가지고 어두운 색일수록 음의 상관관계를 보여준다. 두 개의 변수간의 상관관계를 보여주기 때문에 heatmap은 대각선을 기준으로 대칭이다. 위의 다시 상관관계 표의 상위 다섯 줄을 보면 "fixed acidity"는 "volatile acidity","citric acid", "chlorides", "density", "sulphates"와 양의 상관관계를 보여주고 "residual sugar", "free sulfur dioxide", "total sulfur dioxide", "pH", , "alcohol", "quality"과는 음의 상관관계를 보여주고 있다.

D. Z-변환을 적용하여 데이터 전처리를 수행하고, 위 B번의 통계적 특성의 변화를 확인하세요.

```
In [15]: import numpy as np
         print("Before Z-transform")
         data.describe()
```

```
Before Z-transform
```

```
Out[15]:        fixed acidity  volatile acidity  citric acid  residual sugar  \
         count    6497.000000       6497.000000  6497.000000     6497.000000
         mean        7.215307          0.339666     0.318633        5.443235
         std         1.296434          0.164636     0.145318        4.757804
```

```
min         3.800000          0.080000     0.000000         0.600000
25%         6.400000          0.230000     0.250000         1.800000
50%         7.000000          0.290000     0.310000         3.000000
75%         7.700000          0.400000     0.390000         8.100000
max        15.900000          1.580000     1.660000        65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.056034            30.525319            115.744574     0.994697
std       0.035034            17.749400             56.521855     0.002999
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992340
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.996990
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  6497.000000  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801     5.818378
std       0.160787     0.148806     1.192712     0.873255
min       2.720000     0.220000     8.000000     3.000000
25%       3.110000     0.430000     9.500000     5.000000
50%       3.210000     0.510000    10.300000     6.000000
75%       3.320000     0.600000    11.300000     6.000000
max       4.010000     2.000000    14.900000     9.000000
```

In [16]:
```
z = (data - np.mean(data, axis=0))/np.std(data, axis=0)
print("After Z-transform")
z.describe()
```

```
After Z-transform
```

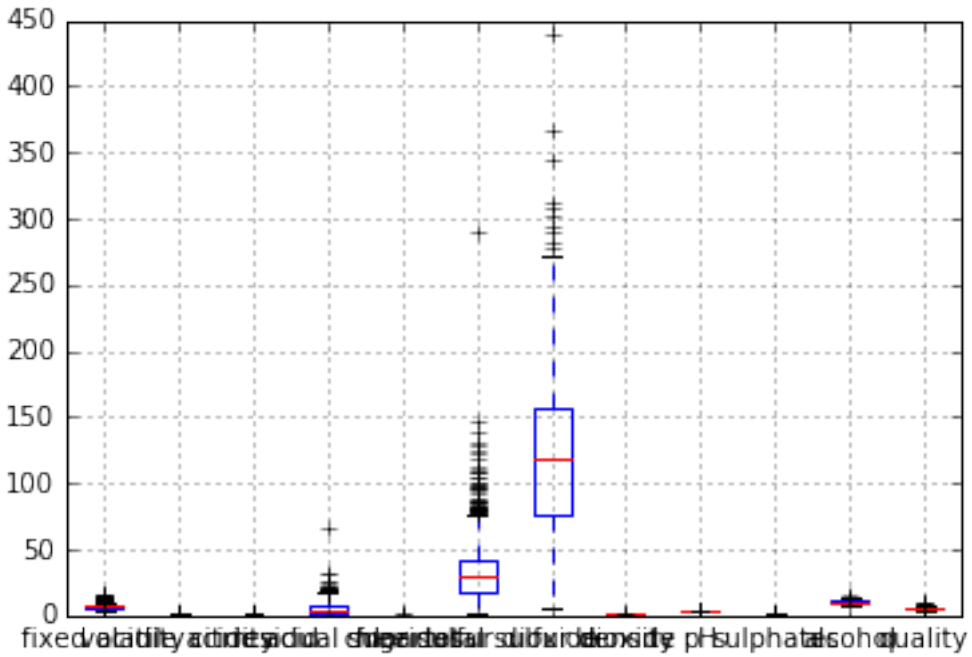
Out[16]:
```
       fixed acidity  volatile acidity   citric acid  residual sugar  \
count   6.497000e+03      6.497000e+03  6.497000e+03    6.497000e+03
mean    3.009718e-15     -2.754242e-14  4.815986e-14   -1.242383e-15
std     1.000077e+00      1.000077e+00  1.000077e+00    1.000077e+00
min    -2.634589e+00     -1.577330e+00 -2.192833e+00   -1.018034e+00
25%    -6.289329e-01     -6.661613e-01 -4.723335e-01   -7.657978e-01
50%    -1.660892e-01     -3.016939e-01 -5.941375e-02   -5.135612e-01
75%     3.738951e-01      3.664962e-01  4.911459e-01    5.584445e-01
max     6.699425e+00      7.534354e+00  9.231281e+00    1.268682e+01

          chlorides  free sulfur dioxide  total sulfur dioxide       density  \
count  6.497000e+03         6.497000e+03          6.497000e+03  6.497000e+03
mean   1.410368e-14        -8.749179e-17         -6.999344e-17  2.181030e-12
std    1.000077e+00         1.000077e+00          1.000077e+00  1.000077e+00
min   -1.342639e+00        -1.663583e+00         -1.941780e+00 -2.530192e+00
```

```
25%   -5.147986e-01        -7.620742e-01         -6.855323e-01 -7.859527e-01
50%   -2.578826e-01        -8.594301e-02          3.990667e-02  6.448888e-02
75%    2.559494e-01         5.901882e-01          7.122647e-01  7.648525e-01
max    1.584219e+01         1.456357e+01          5.737257e+00  1.476879e+01

                 pH     sulphates       alcohol       quality
count  6.497000e+03  6.497000e+03  6.497000e+03  6.497000e+03
mean  -3.321189e-14 -6.491891e-15 -1.240634e-14 -3.018467e-16
std    1.000077e+00  1.000077e+00  1.000077e+00  1.000077e+00
min   -3.100615e+00 -2.091935e+00 -2.089350e+00 -3.227687e+00
25%   -6.748622e-01 -6.805919e-01 -8.316152e-01 -9.372296e-01
50%   -5.287424e-02 -1.429373e-01 -1.608231e-01  2.079990e-01
75%    6.313125e-01  4.619241e-01  6.776670e-01  2.079990e-01
max    4.923029e+00  9.870879e+00  3.696231e+00  3.643685e+00
```

In [17]:
```
#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.
print("Data boxplot")
data.boxplot()
```

```
Data boxplot
```

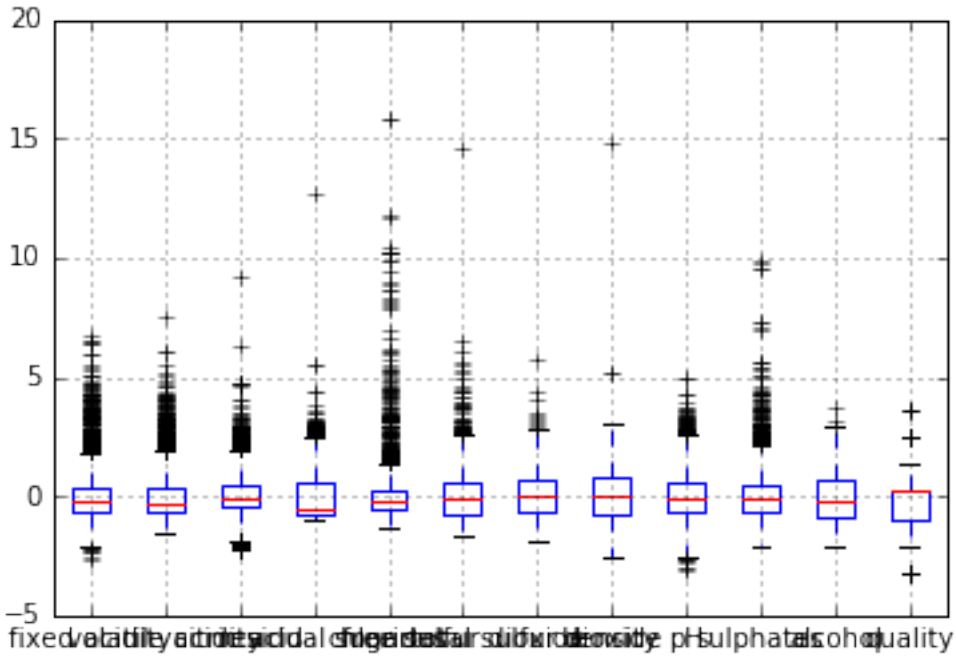
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7facb0c77b70>



In [18]:
```
print("Z-transform boxplot")
z.boxplot()
```

Z-transform boxplot

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7facb0a43128>



위의 결과를 z-transform 평균과 표준편차를 정규분포의 평균과 분산인 $(0, 1)$로 변환되었다.

E. 훈련 집합과 시험 집합을 7:3으로 구분하여 선형회귀 모델을 적용하여 와인의 질을 예측 분석해 보세요.

```
In [19]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score
         x = data.drop(["quality"], axis=1)
         y = pd.DataFrame(data["quality"])
         print("x.shape: {}".format(x.shape))
         x.head()
```

```
x.shape: (6497, 11)
```

Out[19]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	

```
   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

```
In [20]: print("y.shape: {}".format(y.shape))
         y.head()

y.shape: (6497, 1)
```

```
Out[20]:    quality
         0        5
         1        5
         2        5
         3        6
         4        5
```

```
In [21]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
         print("x_train.shape: {}".format(x_train.shape))
         x_train.head()

x_train.shape: (4547, 11)
```

```
Out[21]:       fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
         4452            6.3              0.39         0.22            2.80      0.048
         4666            6.4              0.31         0.53            8.80      0.057
         2333            6.8              0.37         0.28            1.90      0.024
         2473            6.0              0.29         0.21           15.55      0.043
         840             7.6              0.33         0.35            6.30      0.036

               free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
         4452                 53.0                 173.0  0.99304  3.24       0.45
         4666                 36.0                 221.0  0.99642  3.17       0.44
         2333                 64.0                 106.0  0.98993  3.45       0.60
         2473                 20.0                 142.0  0.99658  3.11       0.54
         840                  12.0                 126.0  0.99240  3.16       0.39

               alcohol
```

```
4452      9.8
4666      9.1
2333     12.6
2473     10.1
840      12.0
```

```
In [22]: print("x_test.shape: {}".format(x_test.shape))
         x_test.head()
```

```
x_test.shape: (1950, 11)
```

Out[22]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
122	7.3	0.695	0.00	2.5	0.075	3.0	13.0	0.99800	3.49	0.52	9.2
4141	6.8	0.280	0.29	11.9	0.052	51.0	149.0	0.99544	3.02	0.58	10.4
1275	6.2	0.220	0.27	1.5	0.064	20.0	132.0	0.99380	3.22	0.46	9.2
275	6.6	0.190	0.41	8.9	0.046	51.0	169.0	0.99540	3.14	0.57	9.8
1046	7.6	0.780	0.00	1.7	0.076	33.0	45.0	0.99612	3.31	0.62	10.7

```
In [23]: print("y_train.shape: {}".format(y_train.shape))
         y_train.head()
```

```
y_train.shape: (4547, 1)
```

Out[23]:

	quality
4452	5
4666	5
2333	8
2473	6
840	7

```
In [24]: print("y_test.shape: {}".format(y_test.shape))
         y_test.head()
```

```
y_test.shape: (1950, 1)
```

```
Out[24]:       quality
         122         5
         4141        6
         1275        6
         275         6
         1046        6
```

```
In [25]: #https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegress
         model  = LinearRegression()
         model.fit(x_train, y_train)
         y_pred = model.predict(x_test)
         mse = mean_squared_error(y_test, y_pred)
         r2 = r2_score(y_test, y_pred)
         print("mse:{}".format(mse))
         print("r2_score: {}".format(r2))
```

```
mse:0.5526169612102436
r2_score: 0.2680114929789291
```

data set 7:3으로 나누어 Linear regression model를 quality 값을 예측한 결과는 위와 같다.

1 Reference

- Korean ver.
- projection matrix and subspace
- Newton's method on darkprogrammer
- English ver.
- Mean Vector and Covariance Matrix
- Calculating the sampel covariance on wikipedia
- Covariance on wikipedia
- Newton's method on wikipedia
- Perceptron Learning Algorithm: A Graphical Explanation Of why It works