독립과 종속

- 간략히 말해서, 사건 E와 F가 독립이라는 것은, 우리가 E의 사건이 발생했는지에 대한 정보를 알고 있을 때, 이것이 F에 대한 정보를 주지 못하면 사건 E와 F는 독립이라 한다.
- $P(E, F) = P(E)P(F)$

조건부 확률

- 만약 사건 E와 F가 독립이 아닐 수 있으면, 다음과 같이 조건부 확률을 정의

$$P(E|F)P(E|F) = \frac{P(E,F)}{P(F)}$$

- 사건 F가 발생하였을 때, E가 발생할 확률

Example

- 두 자녀가 있는 가족을 생각해 보자.
- 각 자녀는 같은 확률로 남자 혹은 여자일 수 있다.
- 둘째의 성별은 첫째의 성별과 독립적이다.
- 첫째가 딸(G)일 때, 두 자녀가 모두 딸(B)일 확률?

$$P(B|G) = \frac{P(B,G)}{P(G)} = \frac{B}{G} = \frac{1}{2}$$

- 적어도 하나의 아이가 딸일 때(L), 두 아이가 모두 딸(B)일 확률?

$$P(B|L) = \frac{P(B,L)}{P(L)} = \frac{P(B)}{P(L)} = \frac{1}{3}$$

랜덤 가족을 만들어 결과 확인

In [1]:

```
from __future__ import division
import math, random

def random_kid():
    return random.choice(["boy", "girl"])

both_girls, older_girl, either_girl = 0, 0, 0
random.seed(0)

for _ in range(10000):
        younger = random_kid()
        older = random_kid()
        if older == "girl":
            older_girl += 1
        if older == "girl" and younger == "girl":
            both_girls += 1
        if older == "girl" or younger == "girl":
            either_girl += 1

print "P(both | older):", both_girls / older_girl
print "P(both | either): ", both_girls / either_girl
```

```
  File "<ipython-input-1-48304b48288b>", line 20
    print "P(both | older):", both_girls / older_girl
                          ^
SyntaxError: invalid syntax
```

Bayes's theorem

- $P(F|E)$를 알 때 $P(E|F)$를 계산♂

$$P(E|F) = \frac{P(E,F)}{P(F)} = \frac{P(F|E)P(E)}{P(F)}$$

- 사건 F는 F,E와 F,E^c로 나눌 수 있음

$$P(E|F) = \frac{P(F|E)P(E)}{P(F|E)P(E) + P(F|E^c)P(E^c)}$$

예제: Bayes's theorem

- 10,000명당 1명꼴로 발생하는 질병
- 99% 정확성을 지니는 질병 테스트
- 질병에 대한 랜덤 테스트를 진행했을 때, 양성 반응의 의미
  - D : 질병 감염, T : 양성 반응.

$$P(D|T) = \frac{P(T|D)P(D)}{P(T|D)P(D) + P(T|D^c)P(D^c)}$$

  - $P(T|D) = 0.99, P(T|D^c) = 0.01$
  - $P(D) = 0.0001, P(D^c) = 0.9999$
- ⇒ $P(D|T) = 0.98\%$
- 0.98%의 양성 반응자만이 실제로 질병에 걸림을 알 수 있음
- 실제 테스트에서는 증상이 있는 환자가 검사를 받기 때문에,
- 양성반응+증상이 있음을 조건으로 병에 걸렸을 확률을 계산하면, 확률은 보다 상승

확률 변수

- 확률 변수는 특정 확률 분포를 따르며 랜덤 값을 가지는 변수를 말함
  - 동전의 앞이 나오면 1, 뒷면이 나오면 0
  - 동전을 9번 던졌을 때 앞면의 수
    - range(10)에서 고를 수 있는 임의의 숫자
- 기댓값 : 발생 가능한 확률 변수 값에 확률을 곱한 뒤 합하여 얻는 값
  - 동전 던져서 나온 앞면의 수 : 0.5
  - 주사위의 값 : 3.5

연속 분포 – 균등 분포

- 예: 0과 1사이의 균등 분포 (uniform distribution)
- 확률 밀도 함수(pdf)를 통해 확률 변수의 분포 정도를 표현
- 0과 1사이의 균등 분포에 대한 pdf

In [17]:

```
def uniform_pdf(x):
   return 1 if x>=0 and x<1 else 0
```

- random.random()을 이용하여 균등 분포를 지니는 확률 변수 생성
- 0과 1사이 균등 분포의 확률 누적 함수(cdf):

In [18]:

```
def uniform_cdf(x):
    if x<0: return 0
    elif x<1: return x
    else: return 1
```

연속 분포 – 정규 분포

- 벨 모양의 정규분포는 평균 μ와 표준 편차 σ를 이용하여 표현
- 정규 분포의 pdf:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

In [19]:

```
def normal_pdf(x, mu=0, sigma=1):
    sqrt_two_pi = math.sqrt(2 * math.pi)
    return (math.exp(-(x - mu) ** 2 /2 /sigma ** 2)/(sqrt_two_pi * sigma))
```

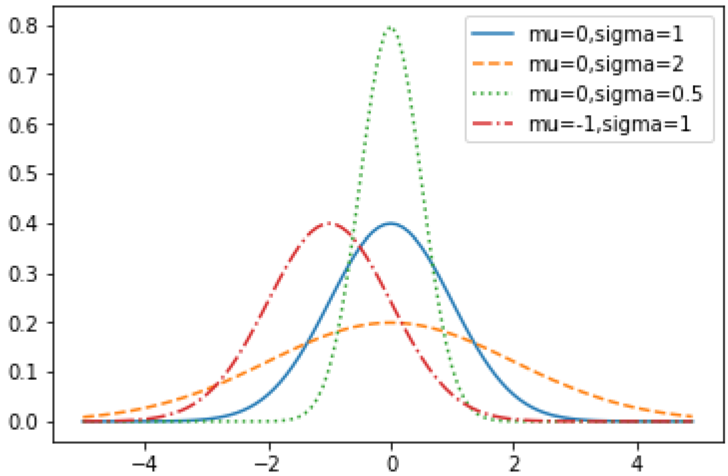
정규분포 pdf 그리기

In [20]:

```
%matplotlib inline
```

In [21]:

```
from matplotlib import pyplot as plt
import math
xs = [x / 10.0 for x in range(-50, 50)]
plt.plot(xs,[normal_pdf(x,sigma=1) for x in xs], '-', label='mu=0,sigma=1')
plt.plot(xs,[normal_pdf(x,sigma=2) for x in xs], '--', label='mu=0,sigma=2')
plt.plot(xs,[normal_pdf(x,sigma=0.5) for x in xs], ':', label='mu=0,sigma=0.5')
plt.plot(xs,[normal_pdf(x,mu=-1)   for x in xs], '-.', label='mu=-1,sigma=1')
plt.legend()
plt.show()
```



정규분포의 cdf

- 정규 분포의 cdf:

$$F(x) = \frac{1}{2}\left\{1 + erf\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right\}$$

- erf는 에러 함수라고 불리며 math.erf를 통해 계산

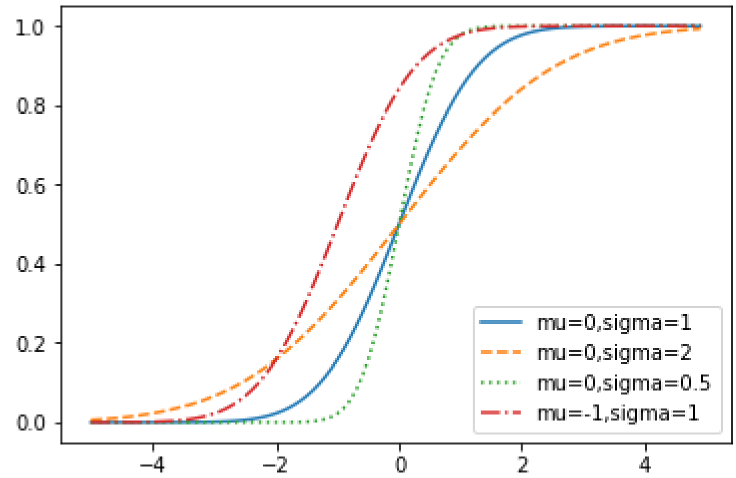
In [22]:

```
def normal_cdf(x, mu=0, sigma=1):
    return (1+math.erf((x-mu)/math.sqrt(2)/sigma))/2
```

정규분포 cdf 그리기

In [23]:

```
xs = [x / 10.0 for x in range(-50, 50)]
plt.plot(xs,[normal_cdf(x,sigma=1) for x in xs], '-', label='mu=0,sigma=1')
plt.plot(xs,[normal_cdf(x,sigma=2) for x in xs], '--', label='mu=0,sigma=2')
plt.plot(xs,[normal_cdf(x,sigma=0.5) for x in xs], ':', label='mu=0,sigma=0.5')
plt.plot(xs,[normal_cdf(x,mu=-1) for x in xs], '-.', label='mu=-1,sigma=1')
plt.legend(loc=4) # bottom right
plt.show()
```



정규분포 cdf의 역함수

- 주어진 p에 대해, p=F(z)를 만족하는 z를 찾는다.
- 이진 탐색을 이용한 방법
  - 먼저 탐색 도메인을 (low_z, high_z)으로 설정
  - 이 도메인의 중앙값인 mid_z에서 F(mid_z)를 계산
    - 만약 F(mid_z)>p 이면 탐색 도메인을 (low_z, mid_z)으로 재설정
    - 만약 F(mid_z)<p 이면 탐색 도메인을 (mid_z, high_z)으로 재설정
    - 만약 F(mid_z)≈p 이면 탐색 종료

정규분포 cdf의 역함수

In [24]:

```
def inverse_normal_cdf(p, mu=0, sigma=1, tolerance=0.00001):
    if mu != 0 or sigma != 1:
        return mu + sigma * inverse_normal_cdf(p, tolerance=tolerance)

    low_z, low_p = -10.0, 0                  # normal_cdf(-10)은 0과 매우 가까움
    hi_z,  hi_p  =  10.0, 1                  # normal_cdf(10)은 1과 매우 가까움
    while hi_p - low_p > tolerance:
        mid_z = (low_z + hi_z) / 2           # 중앙값
        mid_p = normal_cdf(mid_z)            # 중앙값의 cdf
        if mid_p < p:                        # 중앙값이 아직 작으면 윗쪽을 탐색
            low_z, low_p = mid_z, mid_p
        elif mid_p > p:                      # 중앙값이 아직 크면, 아랫쪽을 탐색
            hi_z, hi_p = mid_z, mid_p
        else:
            break
    return mid_z
```

중심 극한 정리

- 같은 분포를 지니는 독립 확률 변수들의 합 혹은 평균은 정규 분포와 가까워 진다.
- $X_1, \cdots, X_n$ 을 평균 μ, 표준 편차 σ를 가질 때, n이 충분히 크면

$$\frac{X_1 + \cdots + X_n - n\mu}{\sigma\sqrt{n}}$$

이 평균 0, 표준편차 1인 정규 분포에 근사함.

베르누이 분포와 이항 분포

- 베르누이 분포 : p의 확률로 성공 시 1, 실패 시 0
- 이항 분포 : n회 독립 베르누이 시행 시 총 성공 횟수

In [25]:

```
def bernoulli_trial(p):
    return 1 if random.random() < p else 0
```

In [26]:

```
def binomial(n, p):
    return sum(bernoulli_trial(p) for _ in range(n))
```

이항 분포의 평균 : np, 표준편차 : $\sqrt{np(1-p)}$

이항분포와 중심 극한 정리

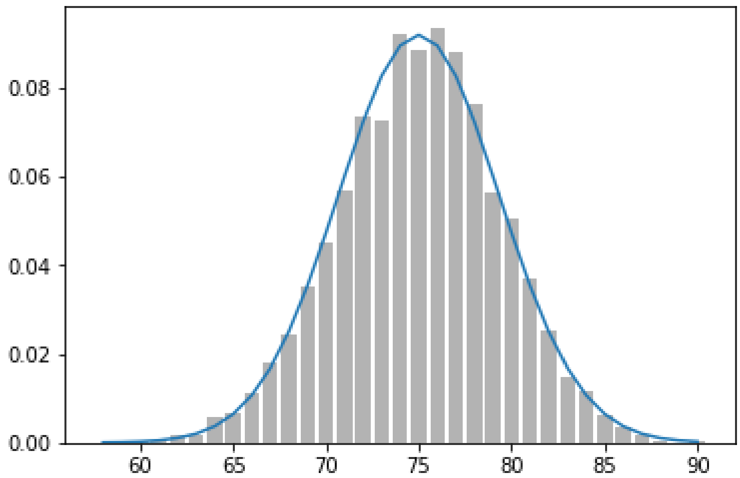
In [30]:

```
from collections import Counter
import random
def make_hist(n, p, num_points):
    data = [binomial(n, p) for _ in range(num_points)]
    # 이항 분포 샘플을 히스토그램으로
    histogram = Counter(data)                    #from collections import Counter
    plt.bar([x for x in histogram.keys()],
            [v / num_points for v in histogram.values()], 0.8, color='0.70')

    mu, sigma = p * n, math.sqrt(n * p * (1 - p))
    # 정규 분포 그리기
    xs = range(min(data), max(data) + 1)
    ys = [normal_cdf(i + 0.5, mu, sigma) - normal_cdf(i - 0.5, mu, sigma)
           for i in xs]
    #ys = [normal_pdf(i) for i in xs]
    plt.plot(xs,ys)
    plt.show()
```

In [31]:

```
make_hist(100, 0.75, 10000)
```



numpy의 기능을 이용한 확률, 통계 함수

In [32]:

```
import numpy as np
x = np.random.rand(10)                       # 10개의 uniform 확률 변수 array
y = np.random.randint(0, 10, 20)             # 0이상 10미만의 20개의 random 정수
z = np.random.randn(10)                      # 10개의 표준정규 확률 변수 array
b = np.random.binomial(n=10, p=0.4, size=10) # n, p의 모수를 지니는 이항분포를 따르는 10개의
 확률 변수
t = np.random.standard_t(df=5, size=10)      # 자유도 nu의 t분포를 따르는 10개의 확률 변수
```

numpy.random.rand (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand.html)
numpy.random.randint (https://docs.scipy.org/doc/numpy-1.12.0/reference/generated/numpy.random.randint.html)
numpy.random.randn (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randn.html)
numpy.random.binomial (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.binomial.html)
numpy.random.standard_t (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.standard_t.html)

numpy의 기능을 이용한 통계량

In [33]:

```
x = np.random.normal(3, 2, 10000)     #평균 3, 표준편차 2인 정규분포
```

다음 중의 방법으로 array의 평균, 표준편차를 구함

In [34]:

```
x.mean()
np.mean(x)
```

Out[34]:

```
2.9965694056980925
```

In [35]:

```
x.std()
np.std(x)
```

Out[35]:

```
1.9979139823920404
```

상관계수 구하는 예제

In [36]:

```
y = np.random.normal(1, 1.5, 10000)
np.corrcoef(x, y)
```

Out[36]:

```
array([[1.        , 0.00676595],
       [0.00676595, 1.        ]])
```

정규 모집단 소표본

In [37]:

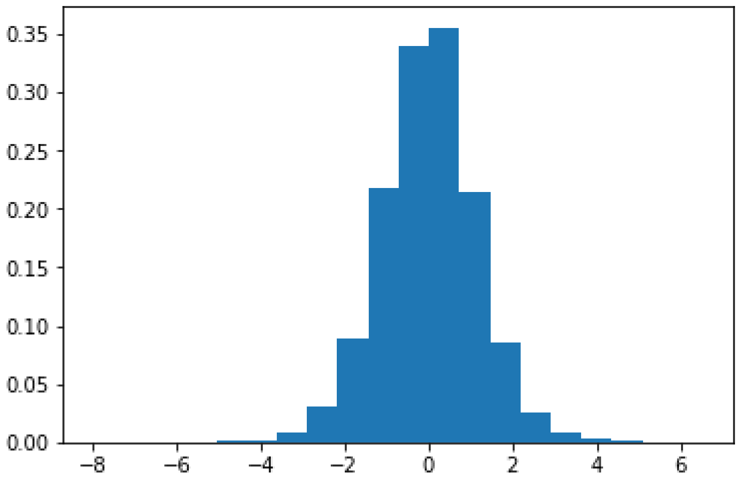
```
import numpy as np
import matplotlib.pyplot as plt

N = 10000
mean_x = np.zeros(N)
std_x = np.zeros(N)
ts = np.zeros(N)

mu, sigma, n = 5, 3, 10      #모평균, 모표준편차, 표본 개수

for i in range(N):
    x = np.random.normal(mu, sigma, n)
    mean_x[i], std_x[i] = x.mean(), x.std(ddof=1)         # 표본 평균, 표본 표준편차
    ts[i] = (mean_x[i] - mu)/(std_x[i]/np.sqrt(n))    # t 통계량

plt.hist(ts, bins=20, normed=True)
plt.show()
```



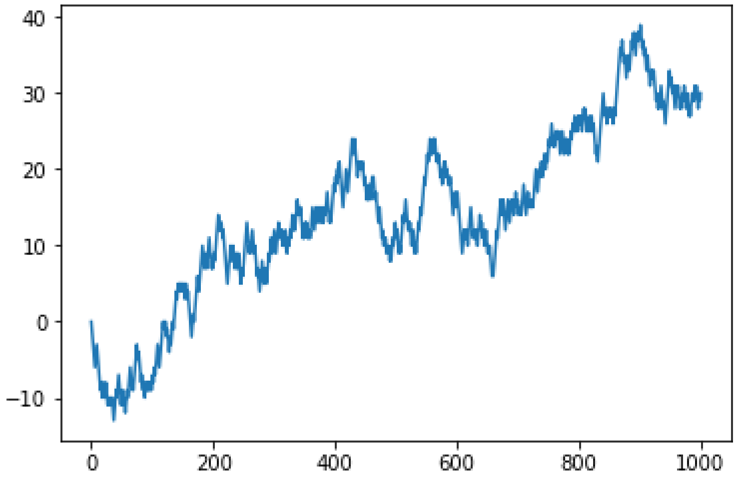
random walks

In [38]:

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

position = 0
walk = [position]
steps = 1000
for _ in range(steps):
    if np.random.randint(0,2): step=1
    else: step=-1
    position += step
    walk.append(position)

plt.plot(range(steps+1), walk)
plt.show()
```



브라운 운동 - Brownian motion

In [39]:

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

position = 0
walk = [position]
steps = 1000
for _ in range(steps):
    step = np.random.normal(loc=0, scale=1)
    position += step
    walk.append(position)

plt.plot(range(steps+1), walk)
plt.show()
```