1. Variational Bayes in Laplace

Given the random variable x obey standard normal distribution，$y = \frac{1}{1+e^{-(Xw^T)}}$, which is the real distribution of y

1.1. generation of data

1.1.1. generation of x

修改：将原来的都服从相同的正态分布改成服从均值为0，方差为1

```
########构造x#################
from scipy.stats import multivariate_normal
import numpy as np
def gen_x(var,n_samples,p_samples):
    np.random.seed(9)####设置随机种子
    cov = np.identity(p_samples)*var
    #print(cov)####输出生成的协方差矩阵
    center =np.zeros(10)
    #print(center)###输出生成的均值array
   # global data_x
    data_x = multivariate_normal.rvs(mean=center, cov=cov, size=n_samples)
    return (data_x)
data_x = gen_x(2,1000,10)
```

1.1.2. generation of w and y

```
######构造w和y#######
def gen_w_y(sigma,center,p_samples,data_x):
    w = np.random.normal(center, sigma, p_samples)
    data_y= 1/(1+np.exp(-np.dot(data_x,w)))
    #print(data_y[1:10])
    y_prob = data_y
    #maxprob = max(y_prob)
    y = np.zeros(1000)
    for i in range(y_prob.shape[0]):
        if (y_prob[i]>0.5):
            y[i] = 1
        else:
            y[i] = 0
    return w,y
```

the *coef_* (the $\hat{w}$) with logistic regression model from sklearn keeps away from the real w , what we need to do is to get the distribution $p(w|X,y)$.
According to the Gaussian mixture model(GMM), which can approximate any distribution. Let the $g(w;\mu,\Sigma) \approx p(w|X,y)$

大量的模型种存在着各种未知的参数，传统的机器学习方法是用自变量和因变量之间的真实联系转换为数学关系（模型），求得参数。
贝叶斯采用的是概率的思想，以现有的样本来推测最大可能下的参数值。$p(w|X,y)$就成为了重点关注对象。如何找到一组w使得$p(w|X,y)$最小。

- 首先根据高斯混合模型可以知道，任何分布都可以用几个高斯分布叠加而成，因此虽然不知道$p(w|X,y)$的具体表达式，但是可以用一个高斯混合分布来近似表示它$g(w;\mu,\Sigma)$
- 对$log(g(w;\mu,\Sigma))$进行泰勒展开(在一阶导为0处展开)，得到最后表达式，minimize该表达式时发现所需要的w就是在一阶导为0的w，它只是要求高斯分布的mode和目标分布的mode位置相同，方法就是把目标分布在mode处做泰勒级数展开到第二阶，然后用对应的高斯分布去代替，就是把未知系数给凑出来
- 因为$log(g(w;\mu,\Sigma))$近似于$log(p(w|X,y))$，所以可以根据最基本的贝叶斯公式将$log(g(w;\mu,\Sigma))$的表达式给写出，以此计算w

1.2. Kullback–Leibler divergence(K.L.)

$$KL = E[log\frac{g(w;\mu,\Sigma)}{p(w|X,y)}] = E[log\frac{g(w;\mu,\Sigma)}{\frac{p(y|X,w)p(w)}{p(y|x)}}] = E[log\frac{g(w;\mu,\Sigma)}{p(y|X,w)p(w)} + logp(y|X)] \quad (1)$$
$$= E[logg(w;\mu,\Sigma) - logp(y|X,w) - logp(w)] + logp(y|X)$$

Let the $J = E[logg(w;\mu,\Sigma) - logp(y|X,w) - logp(w)]$, what we need to do is to minimize the J

$$E(logp(y|X,w)) = log[p(y_1|X,w)p(y_2|X,w)p(y_3|X,w)\cdots p(y_N|X,w)] \quad (2)$$
$$= \sum_{n=1}^{N} logp(y_n|X,w) = \sum_{n=1}^{N} log\frac{1}{1+e^{-X^T w}}$$

1.3. Laplace approximate

$$log(g(w;\mu,\Sigma)) \propto log(g(\hat{w};\mu,\Sigma)) + \frac{\delta log^{'}(g(w;\mu,\Sigma))}{\delta w}|_{w=\hat{w}}\frac{(w-\hat{w})}{1!} + \quad (3)$$
$$\frac{\delta log^{''}(g(w;\mu,\Sigma))}{\delta w}|_{w=\hat{w}}\frac{(w-\hat{w})^2}{2!}$$

when $w=\hat{w}$, the First derivative is 0. $\hat{w}$ is the value of Newton's method, So:

$$g(w;\mu,\Sigma) \propto log(g(\hat{w};\mu,\Sigma)) + \frac{\delta log^{''}(g(w;\mu,\Sigma))}{\delta w}|_{w=\hat{w}}\frac{(w-\hat{w})^2}{2!} \quad (4)$$

Let $K = g(\hat{w};\mu,\Sigma), \mu = \hat{w}, \sigma^2 = \frac{1}{v}, v = -\frac{\delta log^{''}(g(w;\mu,\Sigma))}{\delta w}|_{w=\hat{w}}$

the g is transformed into :

$$log(g(w;\mu,\Sigma)) \propto logK - \frac{(w-\mu)^2}{2\sigma^2} \quad (5)$$

As a result, the next step is to solve the $\hat{w}$

1.3.1. $\hat{w}$ of solution

As we all know , $p(w|x,y) = \frac{p(y|x,w)p(w)}{p(y|x)}$,then define the $log[g(w;x,y,\sigma^2)] = log[p(y|x,w)p(w|\sigma^2)]$. Obviously, the $\frac{log[g(w;x,y,\sigma^2)]}{p(w|x,y)} = C$, the C is a constant, so if the maxmum of $log(g)$ is solved, the $\hat{w}$ is what we need.

- **First, we need to transform the function $log(g)$:**

$$log[g(w;x,y,\sigma^2)] = log[p(y|x,w)] + log[p(w|\sigma^2)] \tag{6}$$
$$= log[p(y=y_1|x_1,w)\cdot(y=y_2|x_2,w)\cdots p(y=y_n|x_n,w)] + log[p(w|\sigma^2)]$$
$$= \sum_{n=1}^{N} log[p(y=y_n|x_n,w)] + log[p(w|\sigma^2)]$$
$$= \sum_{n=1}^{N} log[(\frac{1}{1+exp(-w^T x_n)})^{y_n} \cdot (\frac{exp(-w^T x_n)}{1+exp(-w^T x_n)})^{1-y_n}] + log[p(w|\sigma^2)]$$

- **Let $P_n = P(y_n = 1|w, x_n)$, the formula can be transformed as follows:**

$$log[g(w;x,y,\sigma^2)] = log[p(w|\sigma^2)] + \sum_{n=1}^{N} log[P_n^{y_n} \cdot (1-P_n)^{1-y_n}] \tag{7}$$
$$= log[p(w|\sigma^2)] + \sum_{n=1}^{N}(y_n \cdot logP_n + (1-y_n)\cdot log(1-P_n))$$

- **D is defined as the dimension of w, the formula can be transformed as follows:（注意：这里的w是表示向量）**

$$log[g(w;x,y,\sigma^2)] = -\frac{D}{2}log2\pi - Dlog\sigma - \frac{1}{2\sigma^2}w^T w + \sum_{n=1}^{N}(y_n \cdot logP_n + (1-y_n)\cdot log(1-P_n)) \tag{8}$$

- **the next step is to solve for the first derivative，注意这里是对所有的w求导**:

$$\frac{\delta P_n}{\delta w} = P_n(1-P_n)x_n \tag{9}$$

$$\frac{\delta log[g(w;x,y,\sigma^2)]}{\delta w} = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N}(\frac{y_n}{P_n}\frac{\delta P_n}{\delta w} - \frac{1-y_n}{1-P_n}\frac{\delta P_n}{\delta w}) = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N} x_n(y_n - P_n) \tag{10}$$

- **the next step is to solve for the second derivative**:

$$\frac{\delta^2 log[g(w;x,y,\sigma^2)]}{\delta w^2} = -\frac{1}{\sigma^2} - \sum_{n=1}^{N}(x_n^2 P_n(1-P_n)) < 0 \tag{11}$$

Because the second derivative is less than 0, the function must have a maximum. As a result, Newton method is used to solve the zero of the first derivative. Assume that $f(w) = \frac{\delta log[g(w;x,y,\sigma^2)]}{\delta w}$, $f^{(2)}(w) = \frac{\delta^2 log[g(w;x,y,\sigma^2)]}{\delta w^2}$,

$$w_{n+1} = w_n - \frac{f(w_n)}{f^{(1)}(w_n)} \tag{12}$$

1.3.2. Gradient Descent

$$\frac{\delta log[g(w;x,y,\sigma^2)]}{\delta w} = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N}(\frac{y_n}{P_n}\frac{\delta P_n}{\delta w} - \frac{1-y_n}{1-P_n}\frac{\delta P_n}{\delta w}) = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N} x_n(y_n - P_n) \tag{13}$$

$$w_{n+1} = w_n - \eta f^{'}(w_n) \tag{14}$$

```
#求一阶导
def delta1(w,sigma):   #w代表初始的随机生成的w,var代表先验的方差
    initial_w = w
    derivative1 = -(1/(sigma**2))*initial_w
    temp = np.zeros(data_x.shape)
    for i in range(0,data_x.shape[0]):
        temp[i,] = data_x[i,]*(y[i]-1/(1+np.exp(-np.dot(data_x[i,],initial_w))))
    derivative1 = temp.sum(axis = 0)+derivative1
    return (derivative1)
```

```
#求二阶导
def delta2(w,sigma):
    initial_w = w
    derivative2 = -(1/(sigma**2))
    temp = np.zeros(data_x.shape)
    for i in range(0,data_x.shape[0]):
        temp[i,] = data_x[i,]*data_x[i,]*(1-1/(1+np.exp(-
np.dot(data_x[i,],initial_w))))*1/(1+np.exp(-np.dot(data_x[i,],initial_w)))
    derivative2 = temp.sum(axis = 0)-derivative2
    return(derivative2)
```

```
#计算目标函数的值
def cal(w,sigma):
    D = w.shape[0]
    tempsum = 0
    for i in range(data_x.shape[0]):
        p = 1/(1+np.exp(-np.dot(data_x[i,],w)))
        tempsum = tempsum + y[i] * np.log(p) + (1-y[i]) * (1-np.log(p))
    res = tempsum- (D/2)*np.log(2*np.pi) - D * np.log(sigma) - 1/(2*
(sigma**2))*np.dot(w.T,w)
    #print(tempsum-res)
    return (res)
```

$$log[g(w;x,y,\sigma^2)] = -\frac{D}{2}log2\pi - Dlog\sigma - \frac{1}{2\sigma^2}w^Tw + \sum_{n=1}^{N}(y_n \cdot logP_n + (1-y_n)\cdot log(1-P_n)) \quad (15)$$

1.3.3. Gradient Descent

$$\frac{\delta log[g(w;x,y,\sigma^2)]}{\delta w} = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N}(\frac{y_n}{P_n}\frac{\delta P_n}{\delta w} - \frac{1-y_n}{1-P_n}\frac{\delta P_n}{\delta w}) = -\frac{1}{\sigma^2}w + \sum_{n=1}^{N}x_n(y_n - P_n) \quad (16)$$

$$w_{n+1} = w_n - \eta f^{'}(w_n) \quad (17)$$

```
# 调用上述函数求得目标函数一阶导为0的点
if __name__ == '__main__':
    eta = 0.01####学习率
    p = 10 #生成w的维度
    np.random.seed(0)
    sigma0 = 1 #设置w的先验参数信息
    initial_w = np.random.normal(1, sigma0, p)####利用w的先验随机生成w，此处均值为0
    w0 = initial_w
    epsilon = 1
    i = 0
    rec = []
```

```
    while(epsilon>0.001): #梯度下降
        w1 = w0 + eta * delta1(w0,sigma0)
        epsilon = abs(cal(w1,sigma0) - cal(w0,sigma0))
        rec.append(epsilon)
        i = i+1
        w0 = w1
    mu = w0
    v = -delta2(w0,sigma0)
    sigma = 1/v
```

```
print("真正的w为： ")
print(w)
print("\n Laplace的结果为： ")
print(mu)
```

```
真正的w为：
[ 0.43213071  4.36282052  2.28311318  0.36502505  1.3315897   1.00102298
  4.48223722 -0.61547479  0.9392031  -2.56228722]

 Laplace的结果为：
[ 0.43383022  4.30980619  2.30844433  0.33010292  1.45460567  1.18068878
  4.64460481 -0.70787317  0.93355798 -2.67787504]
```