In [1]:
```python
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from scipy.stats import beta
%matplotlib inline
```

Question 1

CS 118B HW2 Brian Barry

1) $J(u) = u^T S u + \lambda(1 - u^T u) = u^T S u + \lambda - \lambda u^T u$

$$\frac{\partial J(u)}{\partial u} = \underbrace{(S+S^T)u}_{\text{Eq. 73}} + \underbrace{0}_{\text{constant}} - \lambda \underbrace{(u)}_{\text{eq 61}}$$

$$= u(S + S^T - \lambda)$$

Question 2

2) $$p(\underset{\text{data set}}{D} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{ND/2} |\Sigma|^{N/2}} \exp\left\{-\frac{1}{2}\sum (x^{(n)} - \mu)^T \Sigma^{-1} (x^{(n)} - \mu)\right\}$$

$$\ln p(D \mid \mu, \Sigma) = \ln\left(\frac{1}{(2\pi)^{ND/2}}\right) + \ln\left(\frac{1}{|\Sigma|^{N/2}}\right) - \frac{1}{2}\sum (x^{(n)} - \mu)^T \Sigma^{-1} (x^{(n)} - \mu)$$

$$= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum (x^{(n)} - \mu)^T \Sigma^{-1} (x^{(n)} - \mu)$$

$$\partial/\partial\mu \ln(D \mid \mu, \Sigma) = \sum \Sigma^{-1}(x_n - \mu)$$

$$0 = \text{'' ''} \rightarrow N \cdot \mu = \sum x_n, \quad \mu_{ML} = \frac{1}{N}\sum^{N} x_n$$

Question 3

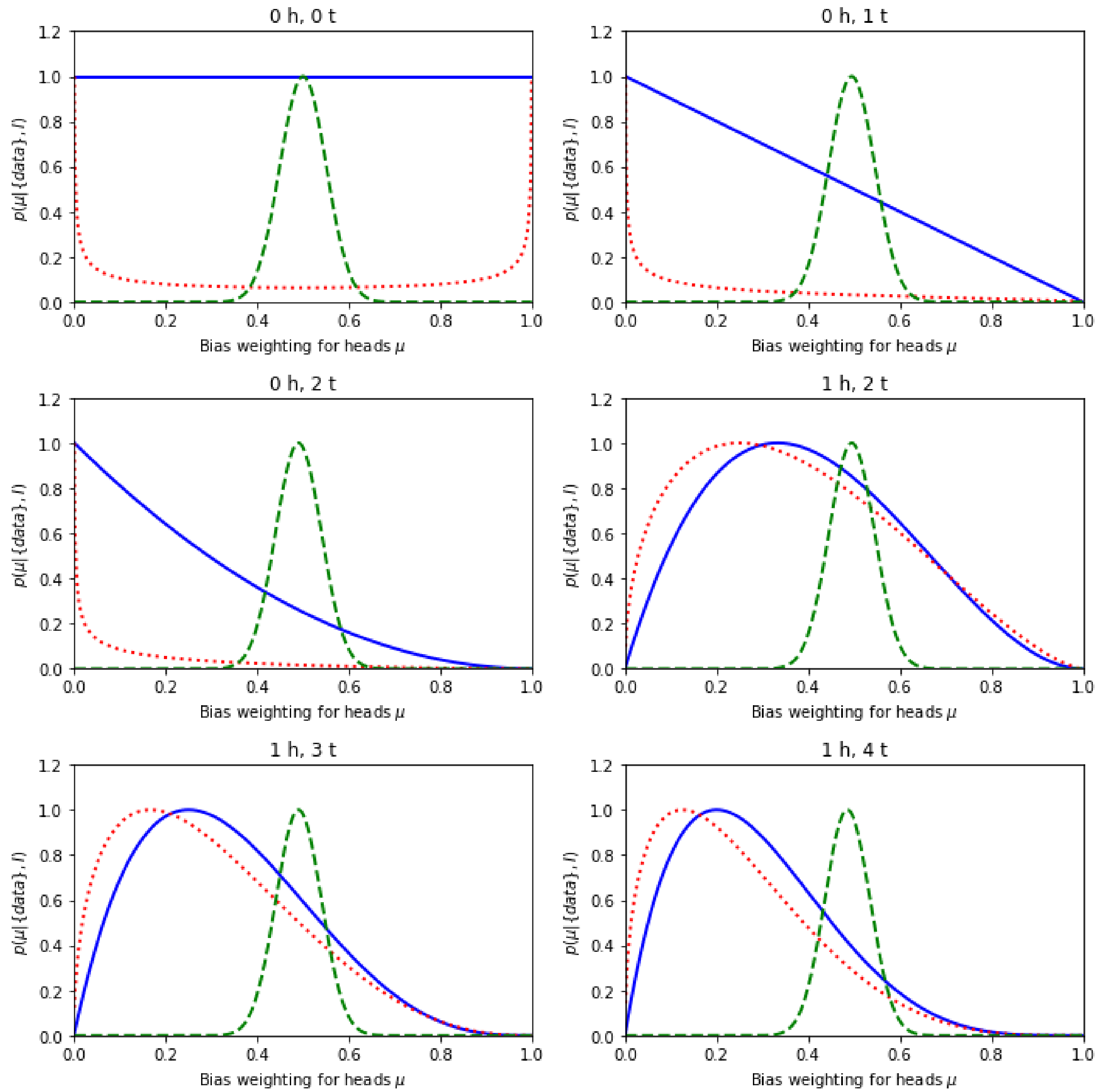
In [2]:
```
def plotbetapdfs(ab, sp_idx, tally):
    # ab is a 3-by-2 matrix containing the a,b parameters for the
    # priors/posteriors
    # Before the first flip: ab = [[1, 1], [0.5, 0.5], [50, 50]]
    #
    # sp_idx is a 3-tuple that specfies in which subplot to plot the curren
    # distributions specified by the (a,b) pairs in ab.
    #
    # tally is a 2-tuple (# heads, # tails) containing a running count of t
    # observed number of heads and tails.
    # Before the first flip: tally=(0,0)

    num_rows = np.shape(ab)[0]
    mark = ['b-','r:','g--'];

    if 'axes' not in globals():
        global fig
        global axes
        fig, axes = plt.subplots(sp_idx[0], sp_idx[1])
        fig.set_figheight(10)
        fig.set_figwidth(10)
    elif np.shape(axes)[0] != sp_idx[:2][0] or np.shape(axes)[1] != sp_idx[
        print(sp_idx[:2])
        print(list(np.shape(axes)))
        fig, axes = plt.subplots(sp_idx[0], sp_idx[1])
        fig.set_figheight(10)
        fig.set_figwidth(10)

    for row in range(num_rows):
        a = ab[row][0]
        b = ab[row][1]

        x = np.linspace(0.001, 0.999, num=999)
        y = beta.pdf(x, a, b)
        norm_y = y / max(y)

        marker = mark[row]
        ax = axes[sp_idx[2]//sp_idx[1], sp_idx[2]%sp_idx[1]]

        ax.plot(x, norm_y, mark[row], lw=2)
        ax.set_xlim([0, 1])
        ax.set_ylim([0, 1.2])
        ax.set_title(str(tally[0])+' h, '+str(tally[1])+' t')
        ax.set_xlabel('Bias weighting for heads $\mu$')
        ax.set_ylabel('$p(\mu|\{data\},I)$')

    fig.tight_layout()
    plt.close()
    return fig
```

5 Flips

In [7]:
```
mu = 0.25
tallies = [np.array([0,0])]
ab = [[ np.array([1, 1]), np.array([0.5, 0.5]), np.array([50, 50]) ] ]

for i in np.arange(5):

    tally = np.random.multinomial(1, [mu, 1-mu], size=1)[0]
    tallies.append(tallies[i]+tally)



    ab_update = [x+tally for x in ab[i]]
    ab.append(ab_update)
```

In [15]:
```
plotbetapdfs(ab[5], [3,2,5], tallies[5])
```

Out[15]:



2048 flips

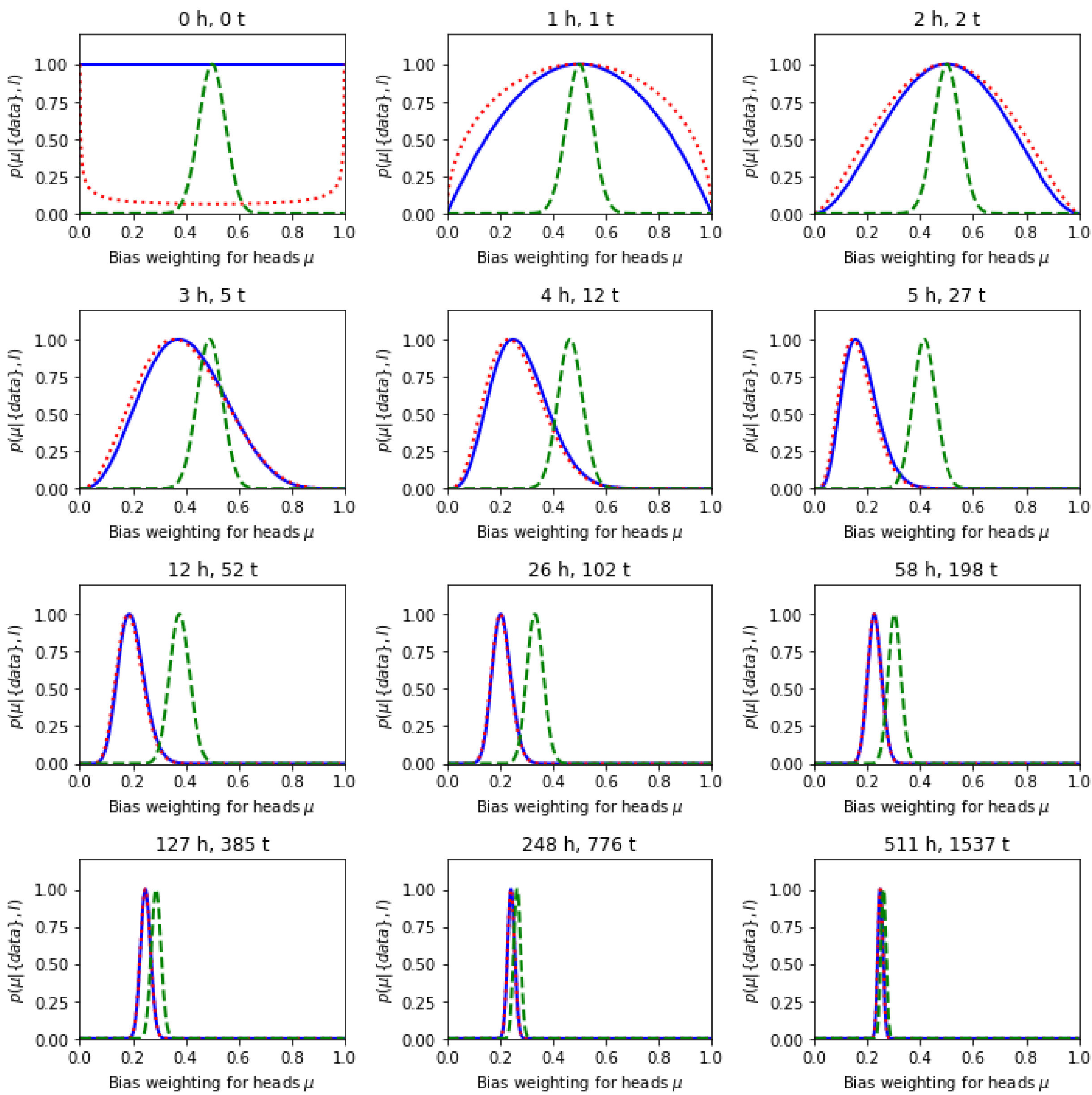
In [16]:

```
mu = 0.25
tallies = [np.array([0,0])]
ab = [[ np.array([1, 1]), np.array([0.5, 0.5]), np.array([50, 50]) ] ]

for i in np.arange(2048):

    tally = np.random.multinomial(1, [mu, 1-mu], size=1)[0]
    tallies.append(tallies[i]+tally)



    ab_update = [x+tally for x in ab[i]]
    ab.append(ab_update)
```

In [29]:

```
plotbetapdfs(ab[2048], [4,3,11], tallies[2048])
```

Out[29]:



C

The Beta(a=50, b=50) distribution is more robust to the updating tallies as it continues to be centered around the mean of an unbiased coin. So here it is clear that the prior is fake data as it does not support the idea that the pribability is in fact 0.25 for heads.

D

At first the Beta(a=50, b=50) is robust to change but its mean eventually converges towards the real mean along with the other smaller priors.

Question 4

In [31]:
```
def plotCurrent(X, Rnk, Kmus):
    N, D = np.shape(X)
    K = np.shape(Kmus)[0]

    InitColorMat = np.matrix([[1, 0, 0],
                              [0, 1, 0],
                              [0, 0, 1],
                              [0, 0, 0],
                              [1, 1, 0],
                              [1, 0, 1],
                              [0, 1, 1]])

    KColorMat = InitColorMat[0:K]
    colorVec = Rnk.dot(KColorMat)
    muColorVec = np.eye(K).dot(KColorMat)

    plt.scatter(X[:,0], X[:,1], edgecolors=colorVec, marker='o', facecolors
    plt.scatter(Kmus[:,0], Kmus[:,1], c=muColorVec, marker='D', s=50);
```

In [32]:
```
data = np.loadtxt('faithful.txt')
k1 = 4
N1 = np.shape(data)[0]
D1 = np.shape(data)[1]

    # Allocate space for the K mu vectors
Kmus1 = np.zeros((k1, D1))
rndinds1 = np.random.permutation(N1)
Kmus1 = data[rndinds1[:k1]];
```

In [33]:
```
def calcSqDistances(X, Kmus):

    sqDmat = []

    for d in X:
        dists = []

        for mu in Kmus:
            dist = np.linalg.norm(d-mu)**2
            dists.append(dist)

        sqDmat.append(dists)

    return np.array(sqDmat)
```

In [34]:
```
def determineRnk(sqDmat):
    Rnks = []

    for i in np.arange(sqDmat.shape[0]):
        Rnks.append(np.zeros((sqDmat.shape[1],), dtype=int))

        nk = np.where(sqDmat[i] == min(sqDmat[i]))[0][0]
        Rnks[i][nk - 1] = 1

    return np.array(Rnks)
```

In [35]:
```
def recalcMus(X, Rnk):
    # Recalculate mu values based on cluster assignments as per Bishop (9.4

    clusters = []

    for k in np.arange(Rnk.shape[1]):
        cluster_k = []

        for i in np.arange(X.shape[0]):
            if Rnk[i][k] == 1:
                cluster_k.append(X[i])

        clusters.append(cluster_k)

    Kmus = []
    for n in clusters:
        Kmus.append( np.average(n, axis=0))

    return np.array(Kmus)
```

In [36]:
```
def runKMeans(K, fileString):
    # Load data file specified by fileStringfrom Bishop book
    X = np.loadtxt(fileString)

    # Determine and store data set information
    N = np.shape(X)[0]
    D = np.shape(X)[1]

    # Allocate space for the K mu vectors
    Kmus = np.zeros((K, D))

    # Initialize cluster centers by randomly picking points from the data
    rndinds = np.random.permutation(N)
    Kmus = X[rndinds[:K]];

    # Specify the maximum number of iterations to allow
    maxiters = 1000;

    for iter in range(maxiters):
        # Assign each data vector to closest mu vector as per Bishop (9.2)
        # Do this by first calculating a squared distance matrix where the
        # contains the squared distance from the nth data vector to the kth

        # sqDmat will be an N-by-K matrix with the n,k entry as specfied ab
        sqDmat = calcSqDistances(X, Kmus);

        # given the matrix of squared distances, determine the closest clus
        # center for each data vector

        # R is the "responsibility" matrix
        # R will be an N-by-K matrix of binary values whose n,k entry is se
        # per Bishop (9.2)
        # Specifically, the n,k entry is 1 if point n is closest to cluster
        # and is 0 otherwise
        Rnk = determineRnk(sqDmat)

        KmusOld = Kmus
        plotCurrent(X, Rnk, Kmus)
        plt.show()

        # Recalculate mu values based on cluster assignments as per Bishop
        Kmus = recalcMus(X, Rnk)

        # Check to see if the cluster centers have converged.  If so, break
        if sum(abs(KmusOld.flatten() - Kmus.flatten())) < 1e-6:
            break

    plotCurrent(X,Rnk,Kmus)
```

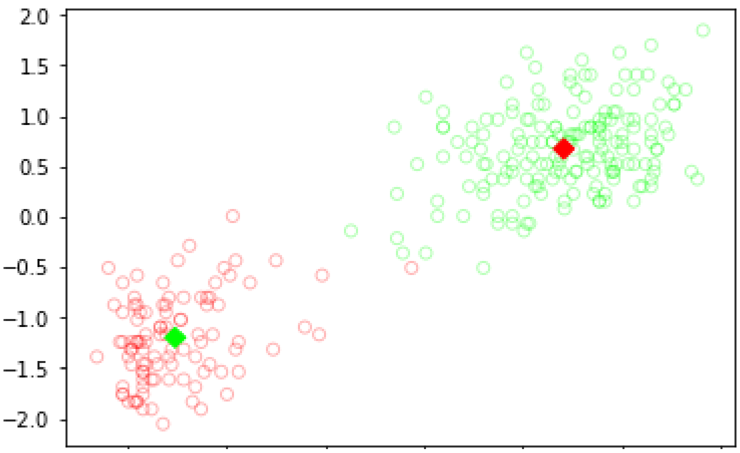
In [ ]:

In [ ]:

In [ ]:

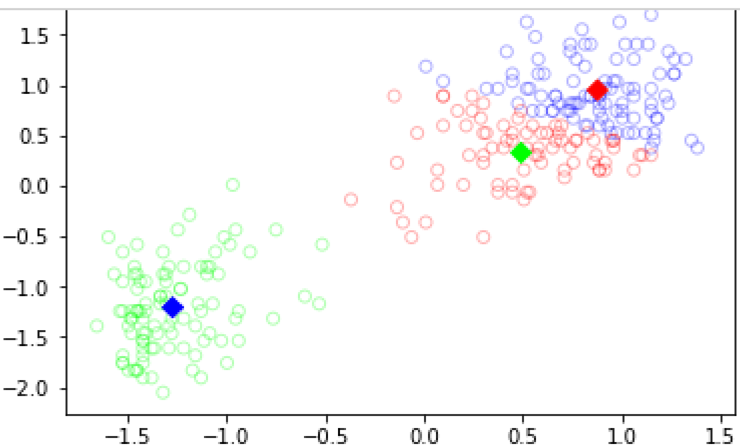
K = 2

In [37]:
```
runKMeans(2, 'scaledfaithful.txt')
```



K = 3

In [38]:
```
runKMeans(3, 'scaledfaithful.txt')
```

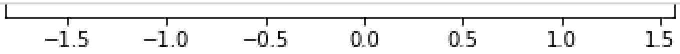


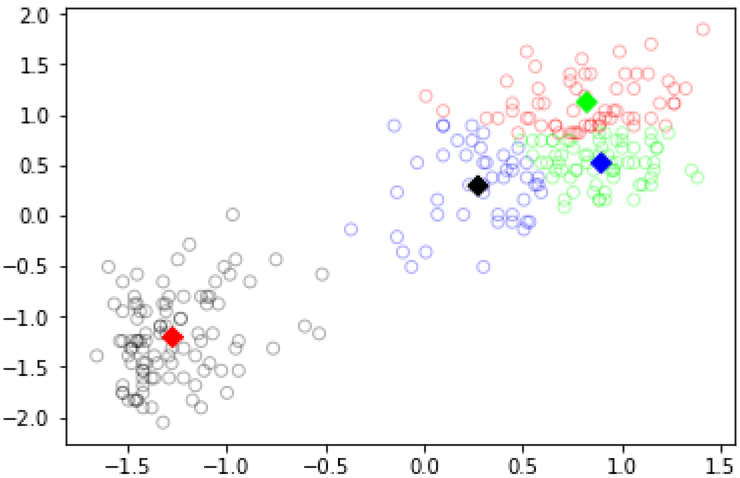
In [ ]:

In [ ]:

K = 4

In [39]:
```
runKMeans(4, 'scaledfaithful.txt')
```





Question 5

5) a) $\int_{-\infty}^{\infty} x e^{-\frac{(x-3)^2}{2}} dx = \int \sqrt{2\pi \cdot 1} \cdot x p(x) dx = \sqrt{2\pi} \cdot E(x) = \boxed{3\sqrt{2\pi}}$

$\mu = 3, \quad \sigma^2 = 1$

b) $\int_{-\infty}^{\infty} (x-3)^2 e^{-\frac{(x-3)^2}{2}} dx \quad \mu = 3, \ \sigma^2 = 1 \qquad (x-3)^2 = x^2 - 6x + 9$

$$= \underbrace{\int x^2 e^{-\frac{(x-3)^2}{2}}}_{\sqrt{2\pi} \cdot E(x^2)} - 6 \underbrace{\int x e^{-\frac{(x-3)^2}{2}}}_{-6 \cdot \sqrt{2\pi} \cdot E(x)} + 9 \int e^{-\frac{(x-3)^2}{2}}$$

$$\sqrt{2\pi} \cdot E(x^2) \quad -6 \cdot \sqrt{2\pi} \cdot E(x) \quad + 9 \cdot \sqrt{2\pi} \longrightarrow$$

$$E(x^2) = Var(x) + \left(E(x)\right)^2 = 1 + 9 = 10$$

$$\int_{-\infty}^{\infty} (x-3)^2 e^{-(x-3)^2/2}\,dx = \sqrt{2\pi}\cdot 10 - \left(6\cdot\sqrt{2\pi}\cdot 3\right) + 9\sqrt{2\pi}$$

$$= \sqrt{2\pi}$$

$$\int_{-\infty}^{\infty} x^2 e^{-(x-3)^2/2}\,dx = \sqrt{2\pi}\cdot E(x^2)$$

$$= \boxed{10\sqrt{2\pi}}$$ according to above work

Question 6

6.) $E[P] = E[x] + E[a] = \boxed{m+a}$

$$Var[P] = E[(X+a)^2] - E[P]^2$$

$$= E[X^2 + 2aX + a^2] - (m+a)^2$$

$$= E[X^2] + 2am + a^2 - m^2 - 2am - a^2, \quad E[X^2] = \sigma^2 + m^2$$

$$= \sigma^2 + \cancel{m^2} + \cancel{2am} + \cancel{a^2} - \cancel{m^2} - \cancel{2am} - a^2 = \boxed{\sigma^2}$$

$$E[Q] = E[bX] = bm$$

$$Var[Q] = E[(bx)^2] - E[Q]^2 = b^2E[x^2] - b^2m^2$$

$$= b^2(\sigma^2 + m^2) - b^2m^2$$

$$= b^2\sigma^2 + \cancel{b^2m^2} - \cancel{b^2m^2}$$

$$= \boxed{b^2\sigma^2}$$