2.1) $p(D|\theta) = \theta \times \theta \times (1-\theta)$

$= \theta^2(1-\theta)$

$= \theta^2 - \theta^3$

2.2) In a general case, if we need 'h' number of heads out of n tosses. The probability is:

$$\binom{n}{h} \theta^h \cdot (1-\theta)^{n-h}$$

In case we want 2 heads out of 3

$$\binom{3}{2} \theta^2 (1-\theta)$$

$$= 3\theta^2(1-\theta)$$

2.3 In case of n_h heads and n_t tails. The likelihood is given by:

$$p(D|\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

where θ is probability of heads.

2.4 The log-likelihood is given by:

$$\log(p(D|\theta)) = n_h \log\theta + n_t \log(1-\theta)$$

Differentiating, w.r.t θ

$$\Rightarrow \frac{\partial}{\partial\theta}(n_h \log\theta) + \frac{\partial}{\partial\theta}(n_t \log(1-\theta)) = 0$$

$$\Rightarrow n_h(1/\theta) + n_t\left(\frac{1}{1-\theta}\right)(-1) = 0$$

$$\Rightarrow n_h(1-\theta) + n_t(-\theta) = 0$$

$$\Rightarrow \theta = \frac{n_h}{n_t + n_h}$$

3.1)

$$p(\theta|D) \propto p(\theta)\, p(D|\theta)$$

$$= \theta^{h-1}(1-\theta)^{t-1}\, \theta^{n_h}(1-\theta)^{n_t}$$

$$= \theta^{h-1+n_h}(1-\theta)^{t-1+n_t}$$

This is equivalent to $Beta(h+n_h, t+n_t)$

3.2

Using the results from the ~~[illegible]~~ question,

$$\hat{\theta}_{MLE} = \frac{n_h}{n_h + n_t}$$

$$\hat{\theta}_{MAP} = \frac{h + n_h - 1}{h + n_h + t + n_t - 2}$$

$$\hat{\theta}_{POSTERIOR\ mean} = \frac{h + n_h}{h + n_h + t + n_t}$$

3.3

As the number of coin flips approach infinity, $\hat{\theta}_{MLE}$, $\hat{\theta}_{MAP}$ and $\hat{\theta}_{Posterior\ mean}$ all approach θ. The effect of the prior reduces as the number of samples increase.

3.4

MLE is the only one of the three to give an unbiased estimate.

MAP is also unbiased, except in the case $h=t=1$

Posterior mean is asymptotically unbiased as $h, t \to 0$.

3.5

I would use posterior mean, and I will take beta-distribution as my prior.

4.1.1

$$p(D_i|\theta_i) = \theta_i^{x_i}(1-\theta_i)^{n_i-x_i}$$

4.1.2

Since $Beta(\theta_i; a, b)$ is a probability density function, it must integrate to 1.

Therefore, $\int \theta_i^{a-1}(1-\theta_i)^{b-1}\,d\theta = B(a,b)$

4.1.3

$$p(\theta_i|D_i) \propto p(\theta_i)\,p(D_i|\theta_i)$$

$$= \frac{1}{B(a,b)}\,\theta_i^{a-1}(1-\theta_i)^{b-1}\cdot\theta_i^{x_i}(1-\theta_i)^{n_i-x_i}$$

$$= \frac{1}{B(a,b)}\,\theta_i^{a_i+x_i-1}(1-\theta_i)^{b+n_i-x_i-1}$$

Therefore, $p(\theta_i|D_i) = \frac{1}{B(a+x_i, b+n_i-x_i)}\left[\theta_i^{(a+x_i)}(1-\theta)^{(b+n_i-x_i)}\right]$

4.1.4

$$p(D_i) = \int p(D_i|\theta_i) \cdot p(\theta_i) \cdot d\theta_i$$

$$= \frac{1}{B(a,b)} \int \theta_i^{a-1}(1-\theta_i)^{b-1} \cdot \theta_i^{x_i}(1-\theta_i)^{n_i-x_i} \cdot d\theta_i$$

$$= \frac{1}{B(a,b)} B(a+x_i, b+n_i-x_i)$$

4.1.5

If we consider $p(D_i) = \int p(D_i|\theta_i) \cdot p(\theta_i) \cdot d\theta_i$ as essentially taking a weighted average over the values of $p(D_i|\theta_i)$, where weight is determined by $p(\theta_i)$. We know that $p(D_i|\theta_i)$ is maximised at (x_i/n_i).

$$p(D_i) = \int p(D_i|\theta_i) \, dP(\theta_i)$$ {P is any general prior}

$$\leq \int p_{MLE}(D_i) \cdot dP(\theta_i)$$ {averaging is worse than putting all weight at x_i/n_i}

$$= p_{MLE}(D_i)$$

4.1.6

Since,

$$p(D_i) = \int p(D_i|\theta_i) \cdot dP(\theta_i)$$

$$p(D_i|\theta_i = x_i/n_i) \geq p(D_i|\theta) \quad \forall \theta \in [0,1]$$

Now, it is clear that if the prior distribution P is as close of MLE, the value of $p(D_i)$ will increase. As P gets closed to MLE, $p(D_i)$ will increase more.

In this case, we can just keep on considering priors such that the ~~mean~~ expected value is x_i/n_i and variance $\to 0$, and we can get larger values of likelihood.

4.2 Empirical Bayes Using All App Data

In the previous section, we considered working with data from a single app. With a fixed prior, such as Beta(3,400), our Bayesian estimates for θ_i seem more reasonable to me[3] than the MLE when our sample size n_i is small. The fact that these estimates seem reasonable is an immediate consequence of the fact that I chose the prior to give high probability to estimates that seem reasonable to me, before ever seeing the data. Our earlier attempt to use empirical Bayes (ML-2) to choose the prior in a data-driven way was not successful. With only a single app, we were essentially overfitting the prior to the data we have. In this section, we'll consider using the data from all the apps, in which case empirical Bayes makes more sense.

1. Let $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_d)$ be the data from all the apps. Give an expression for $p(\mathcal{D} \mid a, b)$, the **marginal likelihood** of $\mathcal{D}$. Expression should be in terms of a, b, x_i, n_i for $i = 1, \ldots, d$. Assume data from different apps are independent. (Hint: This problem should be easy, based on a problem from the previous section.)

   $p(D) = \prod_{i=1}^{d} p(d_i)$

   $\therefore p(\mathcal{D} \mid a, b) = \prod_{i=1}^{d} \frac{B(a+x_i, b+n_i-x_i)}{B(a,b)}$

2. Explain why $p(\theta_i \mid \mathcal{D}) = p(\theta_i \mid \mathcal{D}_i)$, according to our model. In other words, once we choose values for parameters a and b, information about one app does not give any information about other apps.

   In the given problem, all $\theta_i's$ are independent and D_i only depends on θ_i. So any other D_k, where $k \neq i$ doesn't matter.

[3] I say "to me", since I am the one who chose the prior. You may have an entirely different prior, and think that my estimates are terrible.

3. Suppose we have data from 6 apps. 3 of the apps have a fair number of impressions, and 3 have relatively few. Suppose we observe the following:

	Num Clicks	Num Impressions
App 1	50	10000
App 2	160	20000
App 3	180	60000
App 4	0	100
App 5	0	5
App 6	1	2

Compute the empirical Bayes estimates for a and b. (Recall, this amounts to computing $(\hat{a}, \hat{b}) = \arg\max_{(a,b)\in\mathbf{R}^{>0}\times\mathbf{R}^{>0}} p(\mathcal{D} \mid a, b)$.) This will require solving an optimization problem, for which you are free to use any optimization software you like (perhaps scipy.optimize would be useful). The empirical Bayes prior is then $\text{Beta}(\hat{a}, \hat{b})$, where $\hat{a}$ and $\hat{b}$ are our ML-2 estimates. Give the corresponding prior mean and standard deviation for this prior.

I have used scipy.optimize to get the Bayes estimates for a and b.

$\hat{a} = 6.46897515$
$\hat{b} = 1181.43728276$

Prior mean is = .54% and prior SD = .21%

```
import scipy
from scipy.optimize import minimize
from scipy.misc import comb
from scipy.special import beta,betaln
import numpy as np

num_clicks = [50,160,180,0,0,1]
num_impressions = [10000,20000,60000,100,5,2]

def optimize_beta(prior):
    res = 0
    for i in range(len(num_clicks)):

        res = res + (betaln(num_clicks[i] + prior[0], num_impressions[i] - num_clic

    res = res * (-1)
    return res

prior = [10,1200]
print(minimize(fun = optimize_beta, x0 = prior, method = 'Nelder-Mead'))
```

4. Complete the following table:

	NumClicks	NumImpressions	MLE	MAP	PosteriorMean	PosteriorSD
App 1	50	10000	0.5%	0.5%	0.5%	0.07%
App 2	160	20000	0.8%	0.78%	0.79%	0.06%
App 3	180	60000	0.3%	0.3%	0.3%	0.02%
App 4	0	100	0%	0.43%	0.5%	0.2%
App 5	0	5	0%	0.46%	0.54%	0.21%
App 6	1	2	50%	0.54%	0.23%	0.43%

Make sure to take a look at the PosteriorSD values and note which are big and which are small.

4.3.1

$$v = \frac{ab}{(a+b)^2(a+b+1)}$$

$$m = \frac{a}{a+b}$$

Since, $\eta = a+b$

$$m = \frac{a}{\eta}$$

and, $1-m = \frac{b}{\eta}$

Substituting in expression of v,

$$v = \frac{(m\eta)(\eta(1-m))}{\eta^2(\eta+1)}$$

$$= \frac{m(1-m)}{\eta+1}$$

$$\therefore \eta = \frac{m(1-m)}{v} - 1$$

So,

$$Beta(\theta; m, v) = \frac{1}{B(m\eta, (1-m)\eta)} (\theta_i^{m\eta-1})(1-\theta_i)^{(1-m)\eta-1}$$

where, $\eta = a + b = \frac{m(1-m)}{v} - 1$

4.3.2 We will use a Beta prior as $m \in (0,1)$. Since $v \in (0, \infty)$ we would use Gamma Distribution.

4.3.3 $$p(\theta_1, \ldots \theta_d \mid D) = \int_0^\infty \int_0^1 p(\theta_1, \ldots \theta_d \mid m, v) p(m, v \mid D) \, dm \, dv$$

If we use the approximation,

$$p(\theta_1, \ldots, \theta_d \mid D) \approx p(\theta_1, \ldots, \theta_d \mid m_{MAP}, v_{MAP})$$

5 Bayesian Linear Regression

In this problem, we will implement Bayesian Gaussian linear regression, essentially reproducing the example from lecture, which in turn is based on the example in Figure 3.7 of Bishop's *Pattern Recognition and Machine Learning* (page 155). We've provided plotting functionality in "support_code.py". Your task is to complete "problem.py". The implementation uses np.matrix objects, and you are welcome to use[5] the np.matrix.getI method.

1. Implement likelihoodFunc.

```
def likelihoodFunc(W, x, y_train, likelihood_var):

    likelihood = 1.0
    fixed_term = 1.0/(np.sqrt(2 * np.pi * likelihood_var))

    for i in range(len(x)):

        likelihood = likelihood * fixed_term * np.exp(-1 * ((np.square(y_train[i] -

    return likelihood
```

[5] However, in practice we are usually interested in computing the product of a matrix inverse and a vector, i.e. $X^{-1}b$. In this case, it's usually faster and more accurate to use a library's algorithms for solving a system of linear equations. Note that $y = X^{-1}b$ is just the solution to the linear system $Xy = b$. See for example John Cook's blog post for discussion.

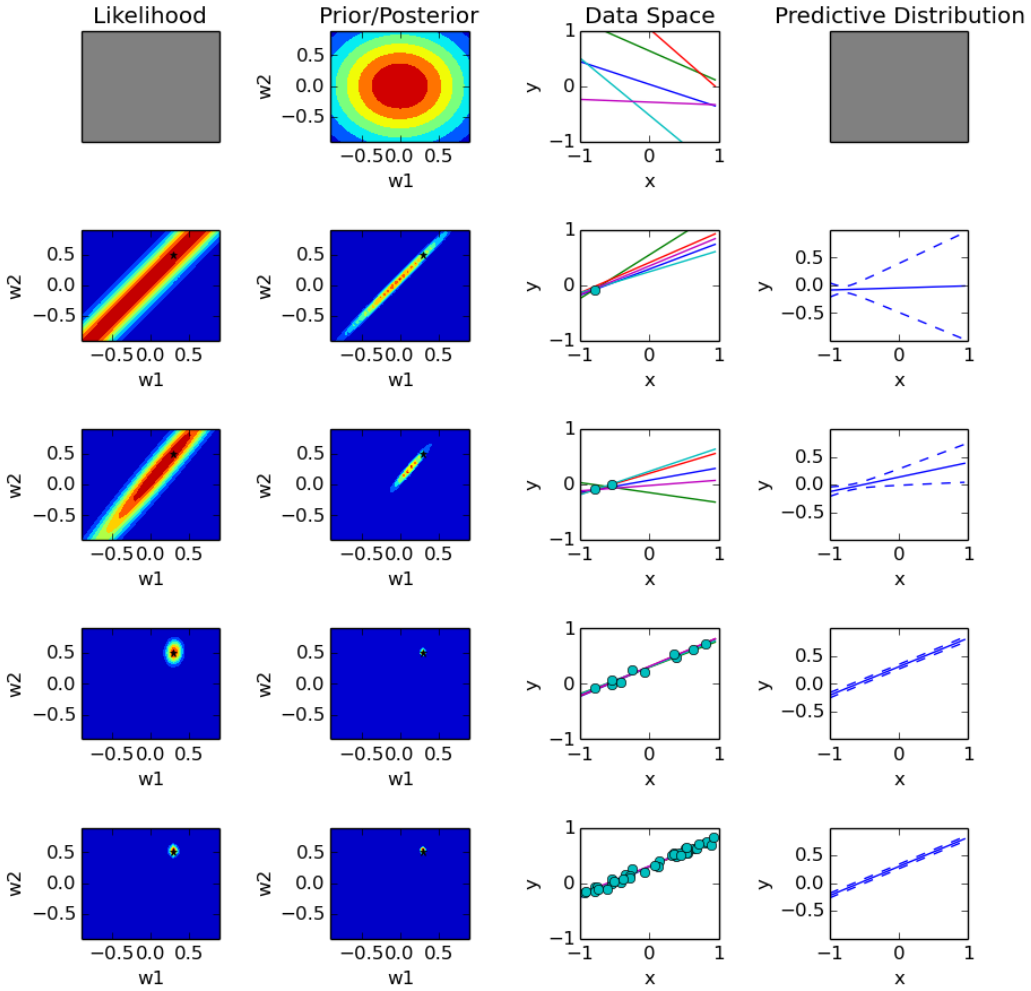
2. Implement getPosteriorParams.

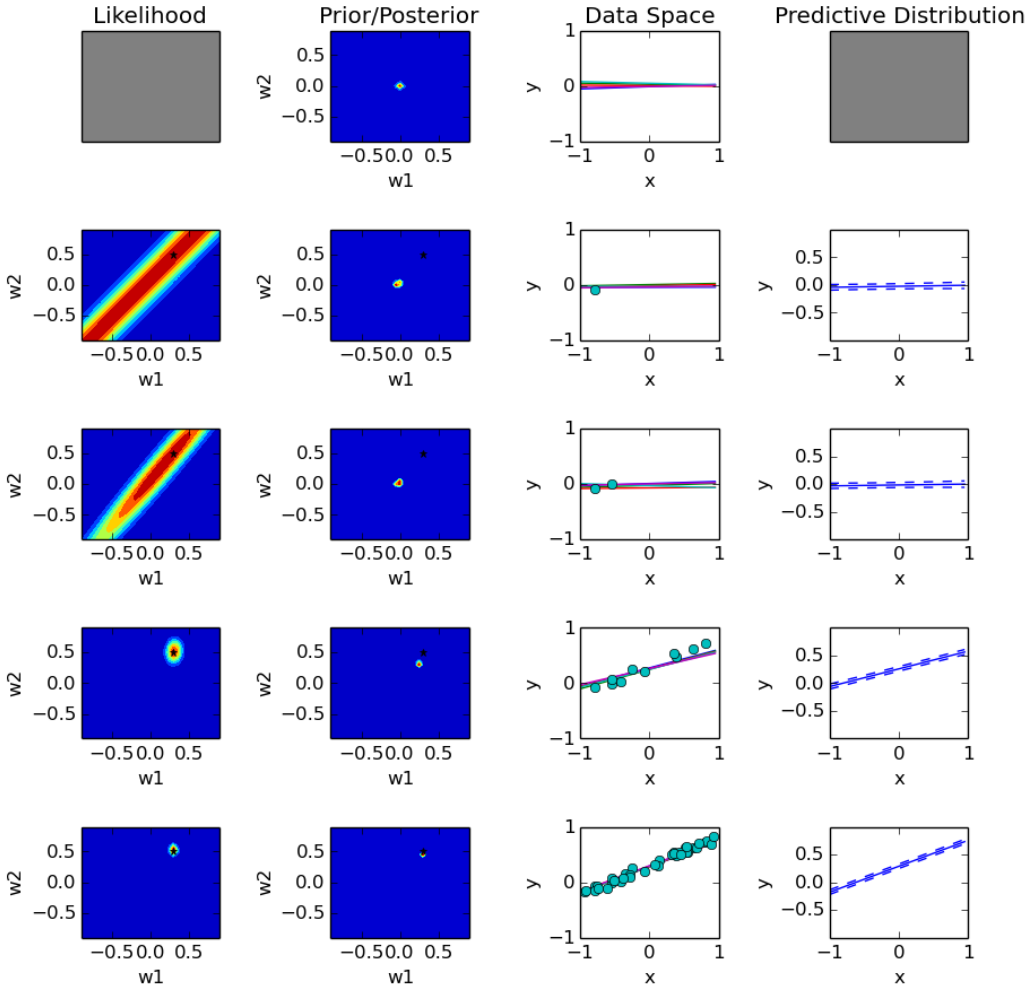
```
def getPosteriorParams(x, y_train, prior, likelihood_var = 0.2**2):

    x_square = np.dot(x.T,x)
    first_term_mean = np.matrix.getI(x_square + (likelihood_var**2) * np.matrix.get
    second_term_mean = np.dot(x.T, y_train)
    postMean = np.dot(first_term_mean, second_term_mean)

    first_term_var = x_square/(likelihood_var**2)
    second_term_var = np.matrix.getI(prior['var'])
    postVar = np.matrix.getI(first_term_var + second_term_var)

    return postMean, postVar
```

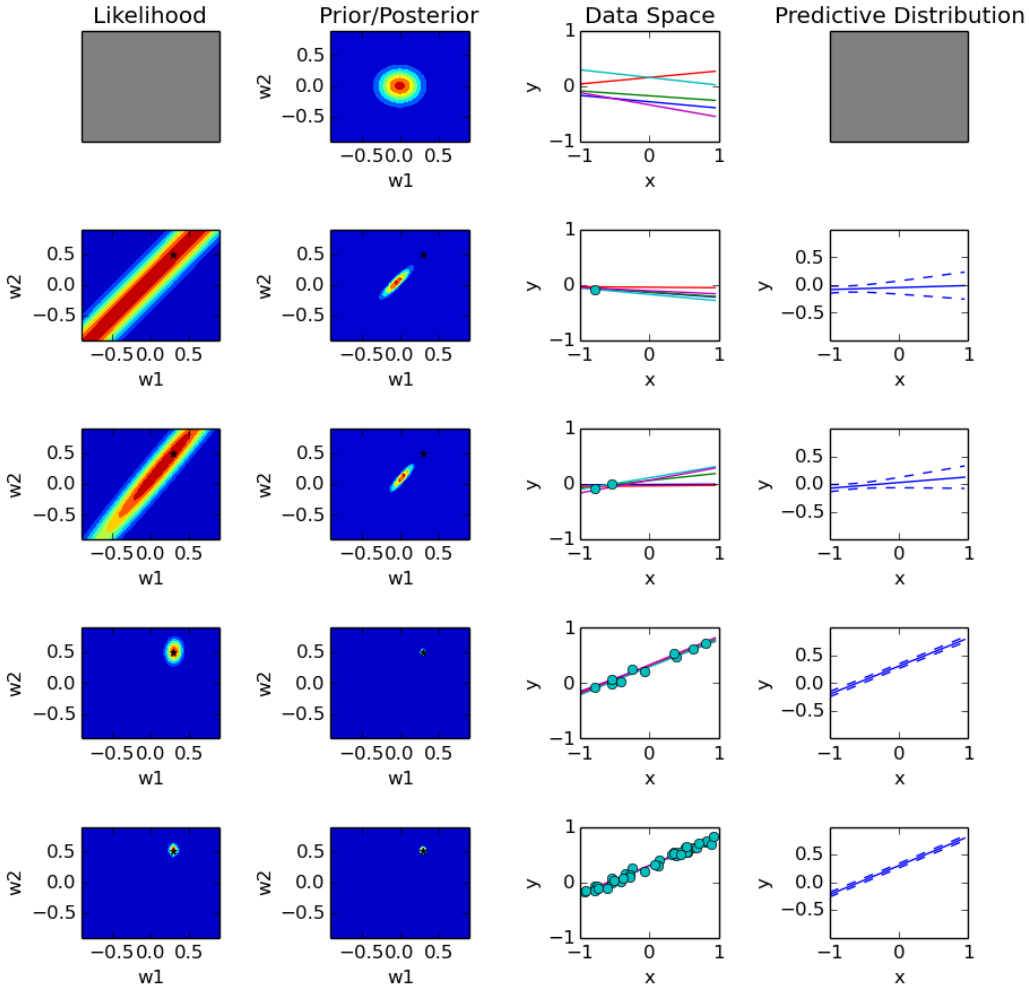
3. Implement getPredictiveParams.

```
def getPredictiveParams(x_new, postMean, postVar, likelihood_var = 0.2**2):

    predMean = np.dot(postMean.T, x_new)
    prod = np.dot(x_new.T, postVar)
    predVar = np.dot(prod, x_new) + (likelihood_var**2)

    return predMean, predVar
```

4. Run 'python problem.py' from inside the Bayesian Regression directory to generate the plots. Note we suggest you change the default behavior in support_code.make_plots from plt.show, to saving the plots for inclusion in your homework submission.




Likelihood
Prior/Posterior
Data Space
Predictive Distribution
w1
w2
x
y


Likelihood
Prior/Posterior
Data Space
Predictive Distribution
w1
w2
x
y

5. Comment on your results. In particular, discuss how each of the following change with sample size and with the strength of the prior: (i) the likelihood function, (ii) the posterior distribution, and (iii) the posterior predictive distribution.

   (i) Likelihood is dependent on the data, so it converges as the sample size increases. It is not affected by the strength of prior.

   (ii) In case of posterior, increase in sample size will result in the improvement of the posterior and it would converge towards to the mode. Also, if we increase the strength of the prior it would lead to the posterior being closer to prior distribution. Now, if the prior is close to the real distribution, we will get to it quickly, but if it is away from the real distribution, it would take more time to converge.

   (iii) In case of predictive distribution, increase in the sample size would decrease the variance of prediction as posterior improves. Increase in the strength of the prior would also lead to the predictive distribution having less variance in prediction in the first few steps. Similar, to the case of the posterior, if prior is away from the true distribution the predictive distribution takes more time to converge.

6. [Optional] Our work above was very much "full Bayes", in that rather than coming up with a single prediction function, we have a whole distribution over posterior prediction functions. However, sometimes we want a single prediction function, and a common approach is to use the MAP estimate. As we discussed in class, for this setting, we can get the MAP estimate using ridge regression. Use ridge regression to get the MAP prediction function corresponding to the first prior covariance. What value did you use for the regularization coefficient? Why?