ISTA 421/521 – Homework 3

Due: Thursday, November 10, 5pm

25 pts total for Undergrads, 30 pts total for Grads

Shuo Yang

Graduate

Instructions

In this assignment you are required to modify/write 3 scripts in python. Details of what you are to do are specified in problems 1, 3, and 7.

Included in the homework 3 release are following sample scripts:

- `approx_expected_value.py` - This script demonstrates how to approximate an expected value through sampling. You will modify this code and submit your solution for problem 1.
- `gauss_surf.py` - This is provided for fun – it is not required for any problem here. It generates a 2d multivariate Gaussian and plots it as both a contour and surface plot.
- `predictive_variance_example.py` - This script demonstrates (a) generating and plotting error bars (predictive variance) and (b) sampling of model parameters from the $\text{cov}\{\widehat{\mathbf{w}}\}$ estimated from data. You will run this script in problem 2, and then use it as the basis for a script in problem 3.
- `w_variation_demo.py` - This script is also provided for fun and is not required for the assignment. (It also provides more example python code!) This implements the simulated experiment demonstrating the theoretical and empirical bias in the estimate of variance, $\widehat{\sigma^2}$, of the model variance, σ^2, as a function of the sample size used for estimation.

All problems require that you provide some "written" answer (in some cases also figures), so you will also submit a .pdf of your written answers. (You can use LaTeX or any other system (including handwritten; plots, of course, must be program-generated) as long as the final version is in PDF.)

The final submission will include (minimally) the three scripts you need to write for problems 1, 3 and 7, and a PDF version of your written part of the assignment. You are required to create either a .zip or tarball (.tar.gz / .tgz) archive of all of the files for your submission and submit your archive to the d2l dropbox by the date/time deadline above.

NOTE: Problems 5 is required for Graduate students only; Undergraduates may complete this problem for extra credit equal to the point value.

(FCMA refers to the course text: Rogers and Girolami (2012), *A First Course in Machine Learning*. For general notes on using LaTeX to typeset math, see: `http://en.wikibooks.org/wiki/LaTeX/Mathematics`)

1. [3 points] Adapted from **Exercise 2.4** of FCMA p.90:

   Let X be a random variable with uniform density, $p(x) = \mathcal{U}(a, b)$. Derive $\mathbf{E}_{p(x)}\{60 + 0.1x + 0.5x^3 + 0.05x^4\}$. Work out analytically $\mathbf{E}_{p(x)}\left\{60 + 0.1x + 0.5x^3 + 0.05x^4\right\}$ for $a = -10$, $b = 5$ (show the steps).

   The script `approx_expected_value.py` demonstrates how you use random samples to approximate an expectation, as described in Section 2.5.1 of the book. The script estimates the expectation of the function y^2 when $Y \sim \mathcal{U}(0, 1)$ (that is, y is uniformly distributed between 0 and 1). This script shows a plot of how the estimation improves as larger samples are considered, up to 100 samples.

   Modify the script `approx_expected_value.py` to compufte a sample-based approximation to the expectation of the function $60 + 0.1x + 0.5x^3 + 0.05x^4$ when $X \sim \mathcal{U}(-10, 5)$ and observe how the approximation improves with the number of samples drawn. Include a plot showing the evolution of the approximation, relative to the true value, over 5,000 samples.

   Solution. Analytical solution:

$$\begin{aligned}
\mathbf{E}_{p(x)}\{60 + 0.1x + 0.5x^3 + 0.05x^4\} &= \int_a^b (60 + 0.1x + 0.5x^3 + 0.05x^4)p(x)dx \\
&= \int_a^b (60 + 0.1x + 0.5x^3 + 0.05x^4)\frac{1}{b-a}dx \\
&= \frac{1}{b-a}(60x + \frac{1}{20}x^2 + \frac{1}{8}x^4 + \frac{1}{100}x^5)|_a^b \\
&= \frac{1}{b-a}((60b + \frac{1}{20}b^2 + \frac{1}{8}b^4 + \frac{1}{100}b^5) - (60a + \frac{1}{20}a^2 + \frac{1}{8}a^4 + \frac{1}{100}a^5)) \\
&= 50.375
\end{aligned}$$

   Approximation solution:
   python program `approx_expected_value.py` output:

```
dhcp-10-134-228-163:hw3 shuoyang$ python approx_expected_value.py
Sample-based approximation: 50.334998
```

   Plot showing the evolution of the approximation (every 100 samples), relative to the true value, over 5000 samples, is shown in Figure 1.

2. [3 points] Adapted from **Exercise 2.12** of FCMA p.91:

   Familiarize yourself with the provided script `predictive_variance_example.py`. When you run it, it will generate a dataset and then remove all values for which $-2 \leq x \leq 2$. Observe the effect this has on the predictive variance in this range. Plot (a) the data, (b) the error bar plots for model orders 1, 3, 5 and 9, and (c) the sampled functions for model orders 1, 3, 5 and 9. You will plot a total of 9 figures. There is no code for you to write for this part, just run the script and it will generate the plots. However, you must include the plots in your submission and do the following: Include a caption for each figure that qualitatively describes what the figure shows; contrast the figures within group (b) with each other; do the same for group (c). Also, clearly explain what removing the points has done in contrast to when they're left in.

   Solution.

   Plot of the data is shown in Figure 2.
   Plots of the error bar is shown in Figure 3.
   Plots of the sampled functions is shown in Figure 4.

   The impact of removing the points

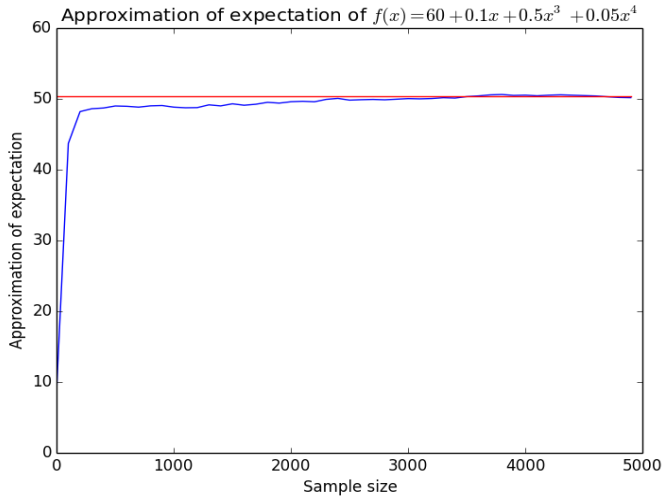


Figure 1: Problem 1 - the evolution of the approximation

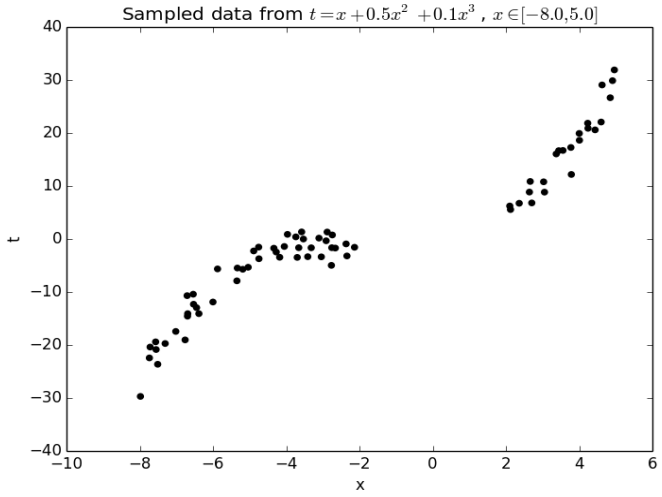


Figure 2: Problem 2 - (a)

Removing the points within the area of [-2, 2] creates a gap of the data. In general, the model is less confident in areas where it has less data. This is why we see that the lenghts of error bars keep increasing in the area of [-2 2] as the model order increases in Figure 3, and the lines of sampled functions are far from each other in the area of [-2 2] for the order 5 and 9.

Comparison of figures within the group (b)

Figure-3 (a) shows that the linear model has relatively high predictive variance compared to other three higher order models except for the area ([-2, 2]) where data points were removed. It's unable to model the deterministic trend in the data very well. The 3rd order model showing in Figure-3 (b) is better to model the trend with much more confident predications since 3rd order is the correct order, just the predictive variances are a little bit high due to removal of data within [-2, 2]. The 5th order model showing in Figure-3 (c) fits the data well except the removal area, but the predictive variances are higher towards the edge of the data than the 3rd model plot. The 9th order model is overly complex, it has too much freedom and can fit the data well for quite a large range of parameter values. This increased uncertainty in $\hat{w}$ leads to increased predictive variability.

Comparison of figures within the group (c)

The plot for linear model shows very little gradient (w_1) change across samples. The effect of removing the data points between [-2, 2] is magnified as the model order increases. The increased variability of functions caused by the increase in parameter uncertainty is clear for 5th and 9th order model.

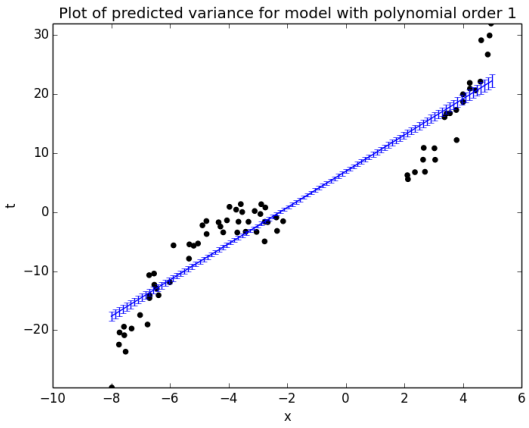
3. [5 points]

In this exercise, you will create a simple demonstration of how model bias impacts variance, similar to the demonstration in Lecture 9 (starting slide 13). Using the same true model in the script `predictive_variance_example.py`, that is $t = x+0.5x^2+0.1x^3$, generate 20 data sets, each consisting of 25 samples from the true function (using the same range of $x \in [-8.0, 5.0]$). Then, create a separate plot for each of the model polynomial orders 1, 3, 5 and 9, in which you plot the true function in red and each of the best fit functions of that model order to the 20 data sets. You will therefore produce four plots. The first will be for model order 1 and will include the true model plotted in red and then 20 curves, one each for an order 1 best fit model for each of the 20 data set, for all data sets. The second plot will repeat this for model order 3, and so on. You can use any of the code in the script `predictive_variance_example.py` as a guide. Describe what happens to the variance in the functions as the model order is changed. (TIPS: plot the true function curve **last**, so it is plotted on top of the others; also, use `linewidth=3` in the plot fn for the true model to increase the line width to make the curve stand out more.)

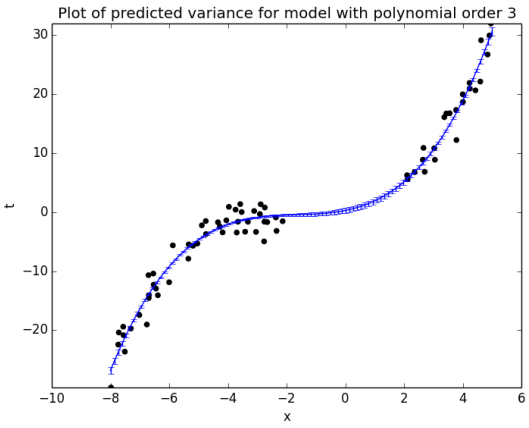
Solution.

The plots for each of model order are shown in Figure 5.

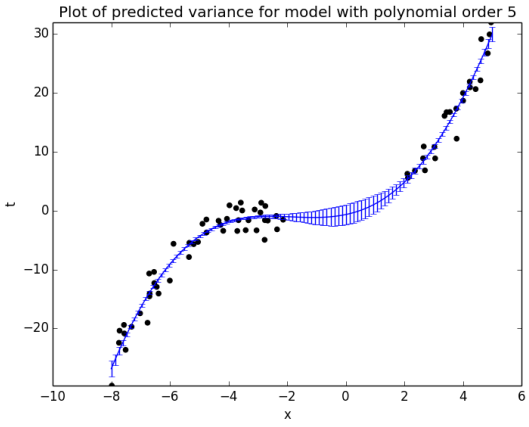
As we can see from the figure, the variance of the function didn't change much from order-1 to order-3, but increase significantly from order-3 to order-5, and from order-5 to order-9. In general, more flexible model has higher variance. The linear model is relatively inflexible and has low variance. The 3rd order model also has low variance since it's the correct order. As order increases to 5 and 9, it becomes more flexible, thus with higher variance as shown in the figure.



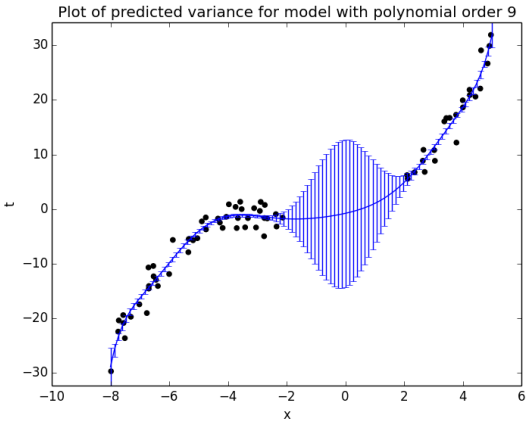
(a) order-1



(b) order-3



(c) order-5



(d) order-9

Figure 3: Problem 2 - group (b)

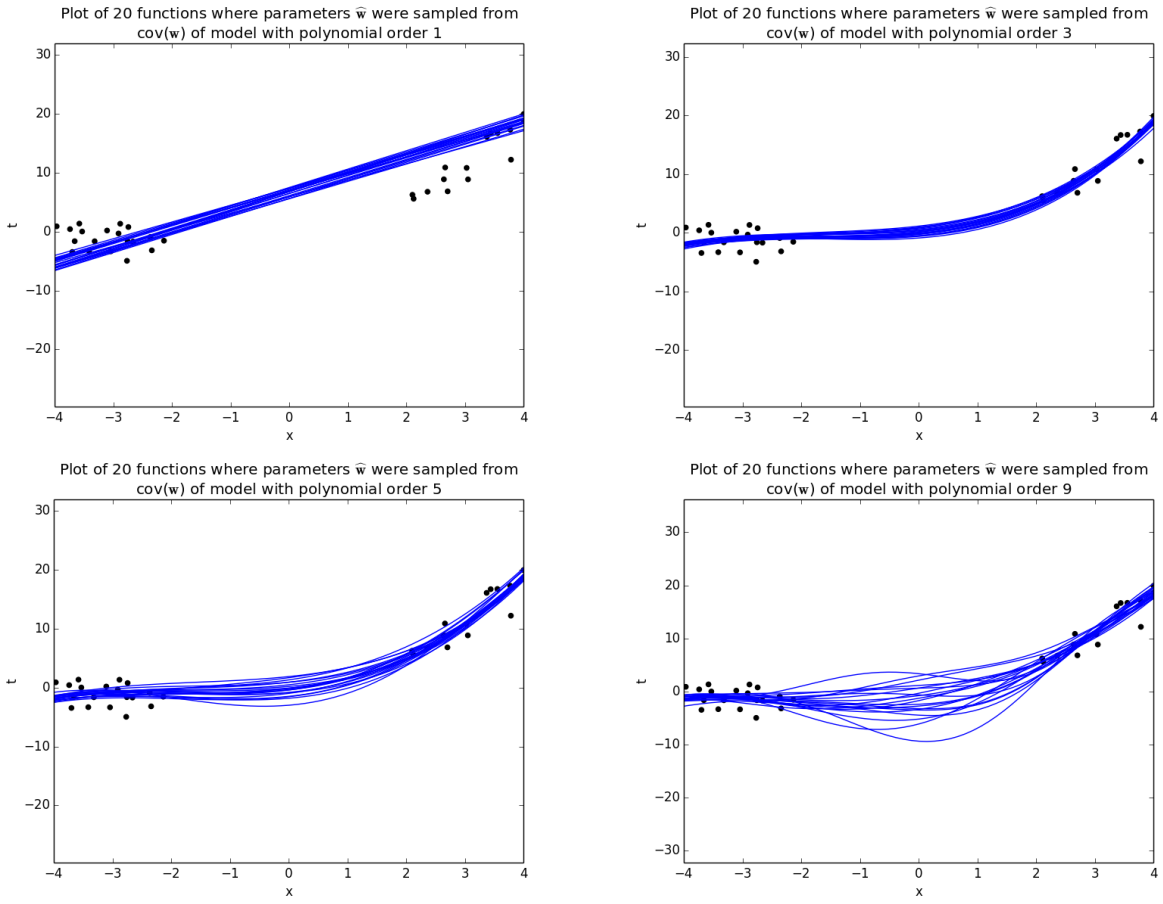


Figure 4: Problem 2 - group (c)

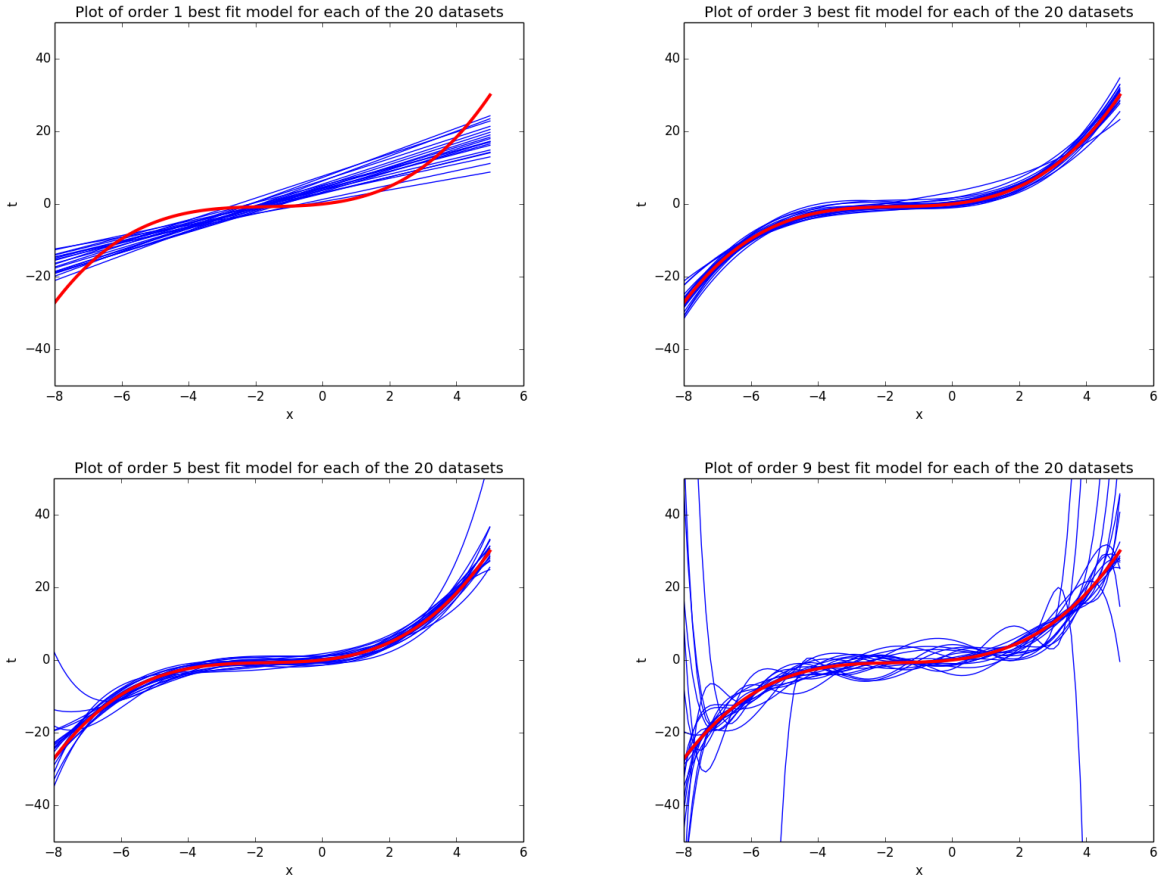


Figure 5: Problem 3

4. [4 points] Adapted from **Exercise 3.5** of FCMA p.134:

   If a random variable R has a beta density

   $$p(r) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1},$$

   derive an expression for the expected value of r, $\mathbb{E}_{p(r)}\{r\}$. You will need the following identity for the gamma function:

   $$\Gamma(n+1) = n\Gamma(n).$$

   Hint: use the fact that

   $$\int_{r=0}^{r=1} r^{\alpha-1}(1-r)^{b-1}\, \mathrm{d}r = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

   Solution. By definition, we have:

   $$\begin{aligned}
   \mathbb{E}_{p(r)}\{r\} &= \int_{r=0}^{r=1} r p(r) dr \\
   &= \int_{r=0}^{r=1} r \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1} dr \\
   &= \int_{r=0}^{r=1} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha+1-1}(1-r)^{\beta-1} dr \\
   &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_{r=0}^{r=1} r^{\alpha+1-1}(1-r)^{\beta-1} dr \\
   &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} \\
   &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\alpha\Gamma(\alpha)\Gamma(\beta)}{(\alpha+\beta)\Gamma(\alpha+\beta)} \\
   &= \frac{\alpha}{\alpha+\beta}
   \end{aligned}$$

5. [5 points; **Required only for Graduates**] Adapted from **Exercise 3.12** of FCMA p.135:

   When performing a Bayesian analysis of the Olympics data, we assumed that σ^2 was known. If instead we assume that $\mathbf{w}$ is known and an inverse Gamma prior is placed on σ^2

   $$p(\sigma^2|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} (\sigma^2)^{-\alpha-1} \exp\left\{-\frac{\beta}{\sigma^2}\right\},$$

   the posterior over σ^2 will also be inverse Gamma. Derive the posterior parameters.

   Solution.

   According to the Bayes rule, we know the following is true for the posterior distribution over σ^2:

   $$p(\sigma^2|\mathbf{t},\mathbf{X},\mathbf{w}) \propto p(\mathbf{t}|\sigma^2,\mathbf{X},\mathbf{w}) p(\sigma^2|\alpha,\beta)$$

The likelihood $p(\mathbf{t}|\sigma^2, \mathbf{X}, \mathbf{w})$ is the Gaussian density with mean $\mathbf{Xw}$ and variance $\sigma^2\mathbf{I}$. Therefore,

$$\begin{aligned}
p(\sigma^2|\mathbf{t}, \mathbf{X}, \mathbf{w}) &\propto \frac{1}{(2\pi)^{N/2}|\sigma^2\mathbf{I}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{t}-\mathbf{Xw})^\top(\sigma^2\mathbf{I})^{-1}(\mathbf{t}-\mathbf{Xw})\right\} p(\sigma^2|\alpha,\beta) \\
&\propto \frac{1}{(2\pi)^{N/2}|\sigma^2\mathbf{I}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{t}-\mathbf{Xw})^\top(\sigma^2\mathbf{I})^{-1}(\mathbf{t}-\mathbf{Xw})\right\} \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1}\exp\left\{-\frac{\beta}{\sigma^2}\right\} \\
&\propto \frac{\beta^\alpha(\sigma^2)^{-\alpha-1}}{\Gamma(\alpha)\sigma^N} \exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(t_n - x_n w)^2 - \frac{\beta}{\sigma^2}\right\} \\
&\propto \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-(\frac{N}{2}+\alpha)-1} \exp\left\{-\frac{\frac{1}{2}\sum_{n=1}^{N}(t_n - x_n w)^2 + \beta}{\sigma^2}\right\} \\
&\propto \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha'-1}\exp\left\{-\frac{\beta'}{\sigma^2}\right\}
\end{aligned}$$

Therefore, the posterior over σ^2 is also an inverse Gamma with the parameters $\alpha' = \frac{N}{2} + \alpha$ and $\beta' = \frac{1}{2}\sum_{n=1}^{N}(t_n - x_n w)^2 + \beta$.

6. [6 points] Adapted from **Exercise 4.2** of FCMA p.165-166:

   In Chapter 3, we computed the posterior density over r, the probability of a coin giving heads, using a beta prior and a binomial likelihood. Recalling that the beta prior, with parameters α and β, is given by

$$p(r|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1}$$

   and the binomial likelihood, assuming y heads in N throws, is given by

$$p(y|r,N) = \binom{N}{y} r^y (1-r)^{N-y}$$

   compute the Laplace approximation to the posterior. (Note, you should be able to obtain a closed-form solution for the MAP value, $\hat{r}$, by getting the log posterior, differentiating (with respect to r), equating to zero and solving for r.)

   Solution.

   By definition of posterior, we have:

$$\begin{aligned}
p(r|y,N) &\propto p(y|r,N)p(r|\alpha,\beta) \\
&\propto \binom{N}{y} r^y(1-r)^{N-y} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1} \\
&\propto \binom{N}{y} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{y+\alpha-1}(1-r)^{N-y+\beta-1}
\end{aligned}$$

   Next, we find the mode $\hat{r}$ by setting the log posterior to zero, differentiating, equating to zero and solving for r.

$$\begin{aligned}
\log p(r|y,N) &= \log \binom{N}{y} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{y+\alpha-1}(1-r)^{N-y+\beta-1} \\
&= \log \binom{N}{y} + \log \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} + \log r^{y+\alpha-1} + \log(1-r)^{N-y+\beta-1} \\
&= \log \binom{N}{y} + \log \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} + (y+\alpha-1)\log r + (N-y+\beta-1)\log(1-r) \\
\frac{\partial \log p(r|y,N)}{\partial r} &= \frac{y+\alpha-1}{r} + \frac{N-y+\beta-1}{r-1}
\end{aligned}$$

setting to zero:

$$\frac{y+\alpha-1}{r} + \frac{N-y+\beta-1}{r-1} = 0$$

we get:

$$\hat{r} = \frac{y+\alpha-1}{N+\beta+\alpha-2}$$

Then we need to calculate the variance σ^2 of the approximated Gaussian. To do that, we need to take second derivative:

$$\begin{aligned}
\frac{\partial \log p(r|y,N)}{\partial r} &= \frac{y+\alpha-1}{r} + \frac{N-y+\beta-1}{r-1} \\
\frac{\partial^2 \log p(r|y,N)}{\partial r^2} &= -\frac{r^2(N-y+\beta-1) + (r-1)^2(y+\alpha-1)}{r^2(r-1)^2}
\end{aligned}$$

The variance is equal to the inverse of the above second derivative when $r = \hat{r}$. Therefore,

$$\sigma^2 = \frac{\hat{r}^2(\hat{r}-1)^2}{\hat{r}^2(N-y+\beta-1) + (\hat{r}-1)^2(y+\alpha-1)}$$

Therfore, the Laplace approximation to the posterior is $p(r|y,N) \approx \mathcal{N}(\hat{r}, \sigma^2)$.

7. [4 points] Adapted from **Exercise 4.3** of FCMA p.166:

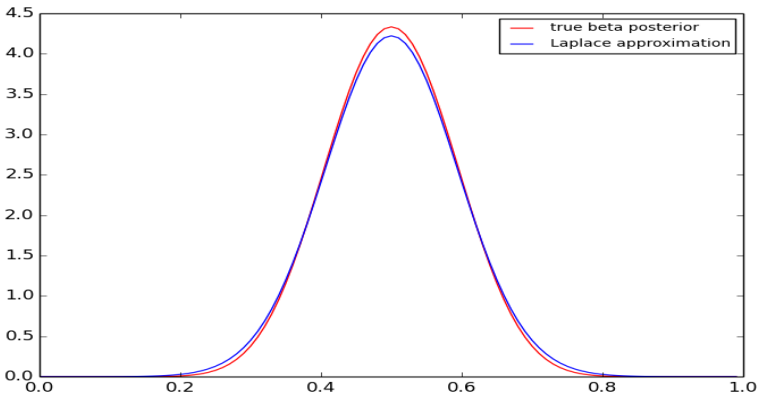
   In the previous exercise you computed the Laplace approximation to the true beta posterior. In this problem, plot both the true beta posterior and the Laplace approximation for the three sets of values:

   1. $\alpha = 5$, $\beta = 5$, $N = 20$, and $y = 10$,
   2. $\alpha = 3$, $\beta = 15$, $N = 10$, and $y = 3$,
   3. $\alpha = 1$, $\beta = 30$, $N = 10$, and $y = 3$.

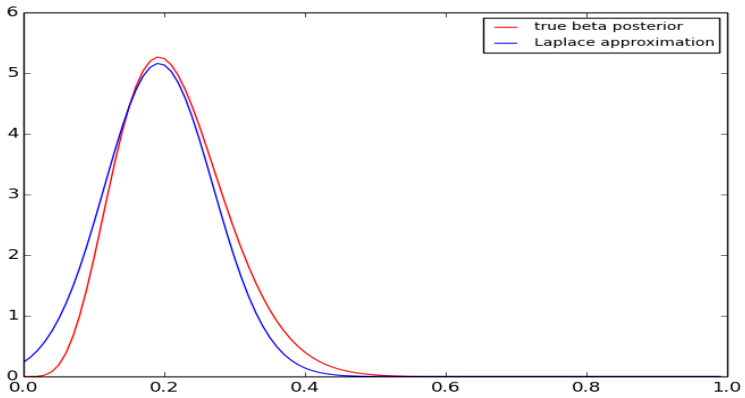
   Be sure to clearly indicate the values in your plot captions. Include how the two distributions (the true beta posterior and the Laplace approximation) compare in each case. Include the python script you use to generate these plots; the script should be named `plot_laplace_approx.py`. **Suggestion**: for plotting the beta and Gaussian (Normal) distributions, you can use `scipy.stats.beta` and `scipy.stats.normal` to create the beta and Gaussian random variables, and use the `pdf(x)` method for each to generate the curves. Note that for `scipy.stats.normal`, the mean is the location (`loc`) parameter, and the sigma is the `scale` parameter. Also, `scipy.stats.normal` expects the scale parameter to be the standard deviation (i.e., take the square root: `math.sqrt(x)`) of the variance you'll compute for the Laplace approximation.

Solution. Plots for the three sets of values are shown in Figure 6.

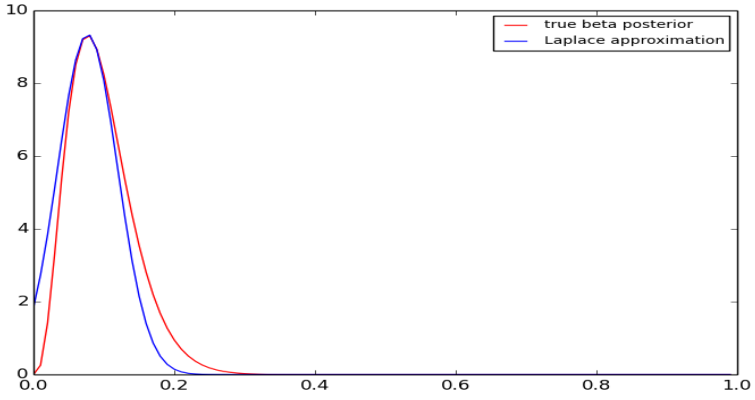
Figure 6 (a) shows that for the first value set, the laplace method approximated the true beta posterior very well. Figure 6 (b) shows that for the second value set, the laplace approximation works well for the x values centered around the mean, but as x values spread out, it loses some precision. This effect is magnified in Figure 6 (c) as the value of α decreases while β increases.



(a) $\alpha = 5$, $\beta = 5$, $N = 20$, and $y = 10$



(b) $\alpha = 3$, $\beta = 15$, $N = 10$, and $y = 3$



(c) $\alpha = 1$, $\beta = 30$, $N = 10$, and $y = 3$

Figure 6: Problem 7