ISTA 421/521 – Homework 3

Due: Friday, October 5, 5pm

20 pts total for Undergrads, 25 pts total for Grads

Ken Youens-Clark

Graduate

N.B.: I consulted with graduate students Kai Blumberg and Matthew Miller.

1. [2 points] Adapted from **Exercise 2.3** of FCML:

   Let Y be a random variable that can take any non-negative integer value. The likelihood of these outcomes is given by the Poisson pmf (probability mass function):

   $$p(y) = \frac{\lambda^y}{y!} e^{-\lambda} \tag{1}$$

   By using the fact that for a discrete random variable the pmf gives the probabilities of the individual events occurring and the probabilities are additive...

   (a) Compute the probability that $Y \geq 5$ and $Y \leq 10$ for $\lambda = 7$, i.e., $P(5 \leq Y \leq 10)$. Write a (very!) short python script to compute this value, and include a listing of the code in your solution.

   (b) Using the result of (a) and the fact that one outcome has to happen, compute the probability that $Y < 5$ or $Y > 10$.

   Solution.

```
#!/usr/bin/env python3
#
# INFO521 Homework 3 Problem #1 (Poisson pmf)
# Author: Ken Youens-Clark
# Sept 29, 2018
#

import numpy as np

lam = 7
tot = 0
for y in range(5,11):
    r = (np.power(lam, y) / np.math.factorial(y)) * np.exp(-1 * lam)
    print('y({:2}, {}) = {:.02f}'.format(y, lam, r))
    tot += r

print('--------   ----'.format(tot))
print('total    = {:.02f}'.format(tot))

$ ./poisson.py
y( 5, 7) = 0.13
y( 6, 7) = 0.15
y( 7, 7) = 0.15
y( 8, 7) = 0.13
```

```
y( 9, 7) = 0.10
y(10, 7) = 0.07
--------   ----
total    = 0.73
```

$P(5 \leq Y \leq 10) = 0.73$, therefore $P(Y < 5 \text{ or } Y > 10) = (1 - 0.73) = 0.27$

2. [3 points] Adapted from **Exercise 2.4** of FCML:

   Let X be a random variable with uniform density, $p(x) = \mathcal{U}(a, b)$.
   Work out analytically $\mathbf{E}_{p(x)}\left\{35 + 3x - 0.5x^3 + 0.05x^4\right\}$ for $a = -4$, $b = 10$ (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 FCML. 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 1000 samples.

   Modify the script `approx_expected_value.py` to compute a sample-based approximation to the expectation of the function $35+3x-0.5x^3+0.05x^4$ when $X \sim \mathcal{U}(-4, 10)$ 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.

$$
\begin{aligned}
p(x) = \mathcal{U}(a, b) &= \frac{1}{b-a} = \frac{1}{10-(-4)} = \frac{1}{14} \\
\mathbf{E}_{p(x)}\left\{35 + 3x - 0.5x^3 + 0.05x^4\right\} &= \int_{x=-4}^{10} 35 + 3x - 0.5x^3 + 0.05x^4 p(x) dx \\
&= \int_{x=-4}^{10} \frac{35 + 3x - 0.5x^3 + 0.05x^4}{14} dx \\
&= \left[\frac{35x + \frac{3}{2}x^2 - \frac{0.5}{4}x^4 + \frac{0.05}{5}x^5}{14}\right]_{-4}^{10} \\
&= \frac{((35 * 10) + (1.5 * 100) - (0.125 * 10000) + (0.01 * 100000))}{14} \\
&\quad - \frac{((35 * -4) + (1.5 * 16) - (0.125 * 256) + (0.01 * -1024))}{14} \\
&= \frac{(350 + 150 - 1250 + 1000)}{14} - \frac{(-140 + 24 - 32 - 10.24)}{14} \\
&= \frac{250 + 158.24}{14} \\
&= \frac{408.24}{14} \\
&= 29.16
\end{aligned}
$$

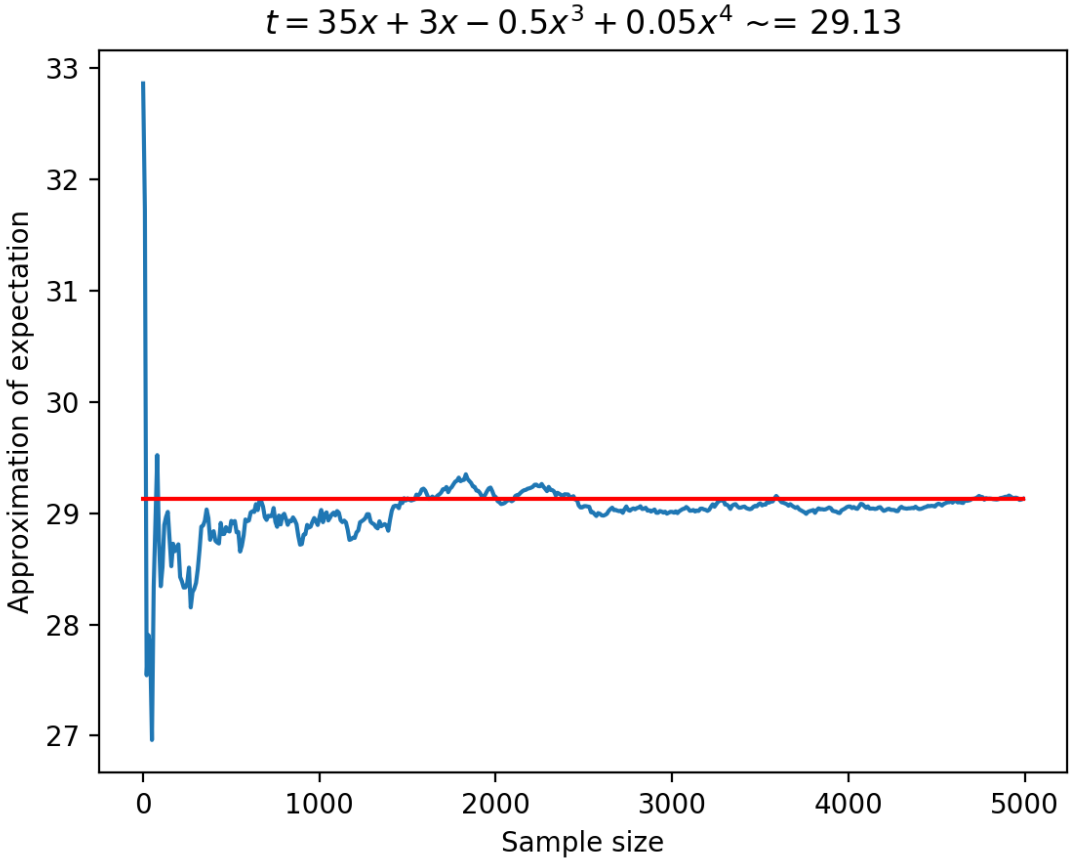

Figure 1: Sample-based approximation

3. [2 points; **Required only for Graduates**] Adapted from **Exercise 2.6** of FCMA:

   Assume that $p(\mathbf{w})$ is the Gaussian pdf for a D-dimensional vector $\mathbf{w}$ given in

$$p(\mathbf{w}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}. \tag{2}$$

   Suppose we use a diagonal covariance matrix with different elements on the diagonal, i.e.,

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_D^2 \end{bmatrix}$$

   Does this assume independence of the D elements of $\mathbf{w}$? If so, show how by expanding the vector notation of Eqn. 3 and re-arranging. You will need to be aware that the determinant of a matrix that only has entries on the diagonal is the product of the diagonal values and that the inverse of the same matrix is constructed by simply inverting each element on the diagonal. (Hint, a product of exponentials can be expressed as an exponential of a sum. Also, just a reminder that $\exp\{x\}$ is e^x.)

Solution.

Yes, the elements of $\mathbf{w}$ are independent.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_D^2 \end{bmatrix}$$

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_D^2} \end{bmatrix}$$

$$p(\mathbf{w}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}.$$

$$= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{1}{\sigma_d^2}(w_d-\mu_d)^2\right\}$$

$$= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2\sigma_d^2}\sum_{d=1}^{D}(w_d-\mu_d)^2\right\}$$

$$= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \prod_{d=1}^{D}\exp\left\{-\frac{1}{2\sigma_d^2}(w_d-\mu_d)^2\right\}$$

$$p(\mathbf{w}) = \prod_{d=1}^{D}\frac{1}{(2\pi\sigma_d^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma_d^2}(w_d-\mu_d)^2\right\}$$

4. [4 points] Adapted from **Exercise 2.9** of FCML:

   Assume that a dataset of N binary values, $x_1, ..., x_n$, was sampled from a Bernoulli distribution, and each sample x_i is independent of any other sample. Explain why this is *not* a Binomial distribution. Derive the maximum likelihood estimate for the Bernoulli parameter.

   Solution.

   A Bernoulli is concerned with a particular sequence, so "HHTHH" which produces an outcome of one T.

   A Binomial considers all the possibilities of getting a particular outcome (e.g., one T): "THHHH", "HTHHH" ... "HHHHT."

   If each sample is independent, then we're not thinking about all the different ways we could have gotten each particular sequence and therefore all the combinations of sequences that would produce the end results. The result would be:

$$Bern(x|\mu) = \mu^x(1-\mu)^{(1-x)}$$

$$Bern(x_1) * Bern(x_2) * Bern(x_3) * ...Bern(x_n) = \prod_{n}^{N} Bern(x_n)$$

Whereas a Binomial distribution incorporates an additional function to consider combinations (N choose m):

$$Bin(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{(N-m)}$$

MLE:

$$\begin{aligned}
P(X=x) &= q^x(1-q)^{1-x} \\
\mathcal{L} &= \prod_n^N q^{x_n}(1-q)^{1-x_n} \\
\log\mathcal{L} &= \sum_n^N \log\left(q^{x_n}(1-q)^{1-x_n}\right) \\
&= \sum_n^N x_n \log q + \sum_n^N \log(1-q)^{1-x_n} \\
&= \log q \sum_n^N x_n + \log(1-q)^{\sum_n^N 1-x_n} \\
\frac{\partial \log\mathcal{L}}{\partial q} &= \frac{\sum_n^N x_n}{q} + \frac{\sum_n^N 1-x_n}{1-q} = 0 \\
\frac{\sum_n^N x_n}{q} &= -\frac{\sum_n^N 1-x_n}{1-q} \\
(1-q)\sum_n^N x_n &= -q\sum_n^N (1-x_n) \\
(1-q)\sum_n^N x_n &= -q(N-\sum_n^N x_n) \\
\sum_n^N x_n - q\sum_n^N x_n &= -qN + q\sum_n^N x_n \\
\sum_n^N x_n &= -qN + 2q\sum_n^N x_n \\
\sum_n^N x_n &= q(2\sum_n^N x_n - N) \\
q &= \frac{\sum_n^N x_n}{2\sum_n^N x_n - N}
\end{aligned}$$

5. [6 points] Adapted from **Exercise 2.12** of FCML:

   Familiarize yourself with the provided script `predictive_variance.py`. It is mostly implemented, but you will have to fill in the details for two functions:

   – `calculate_prediction_variance`, which calculates the *variance* for a prediction at x_{new} given the design matrix, $\mathbf{X}$, the estimated parameters, $\mathbf{w}$, and target responses, $\mathbf{t}$.

– `calculate_cov_w`, which calculates the estimated covariance of $\mathbf{w}$ given the design matrix, $\mathbf{X}$, the estimated parameters, $\mathbf{w}$, and target responses, $\mathbf{t}$.

Once implemented, then you can run the script.

When you run the script, it will generate a dataset based on a function (implemented in `true_function`) and then remove all values for which $-2 \leq x \leq 2$. Three groups of plots will be generated:

(a) First is a plot of the data (this will be generated by Part 5a of the script, starting on line 70).

(b) Next, the script will plot the error bar plots for predictions of values for model orders 1, 3, 5 and 9 (this will be generated by Part 5b of the script, starting on line 145).

(c) Finally, in Part 5c (starting line 197), the script samples model parameters $\mathbf{w}$ from the covariance $\text{cov}(w)$ and plots the resulting functions (again, for model orders 1, 3, 5 and 9).

In total, you will plot 9 figures. 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.

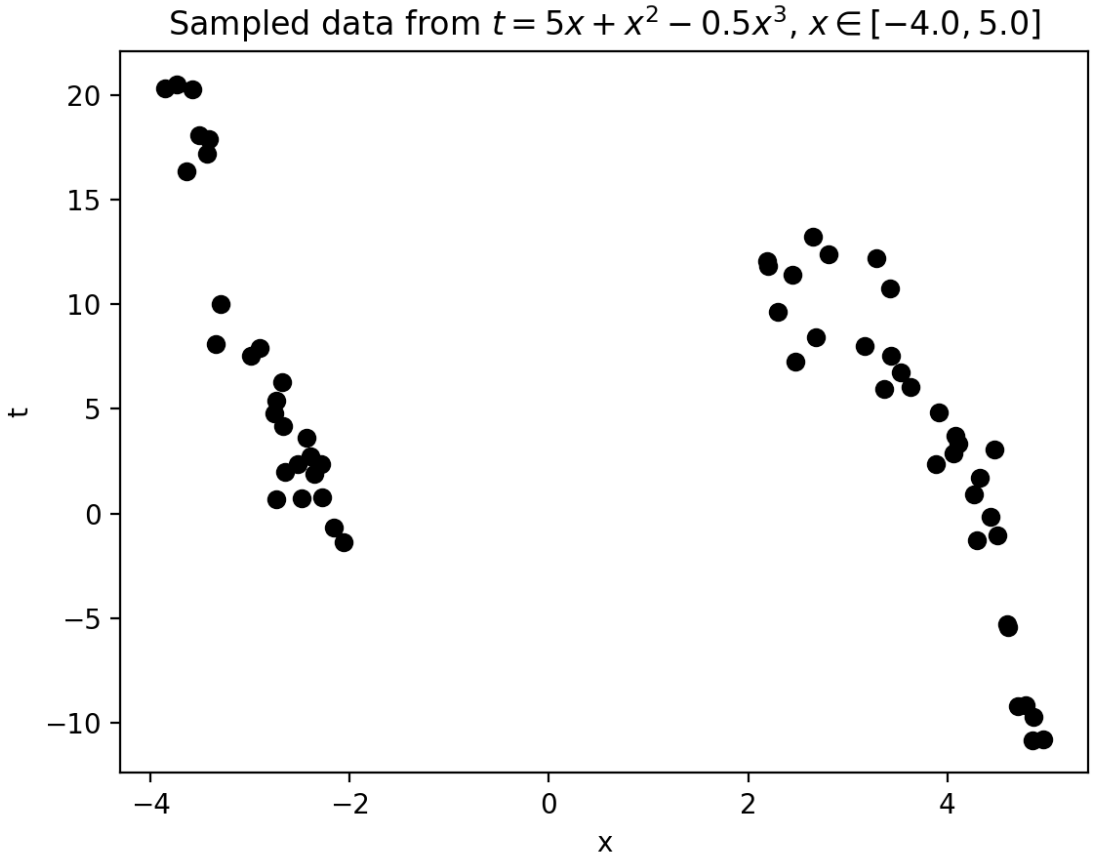

Figure 2: This is a sample of the data generated by our true function with a gap introduced by deleting the values between -2 and 2.

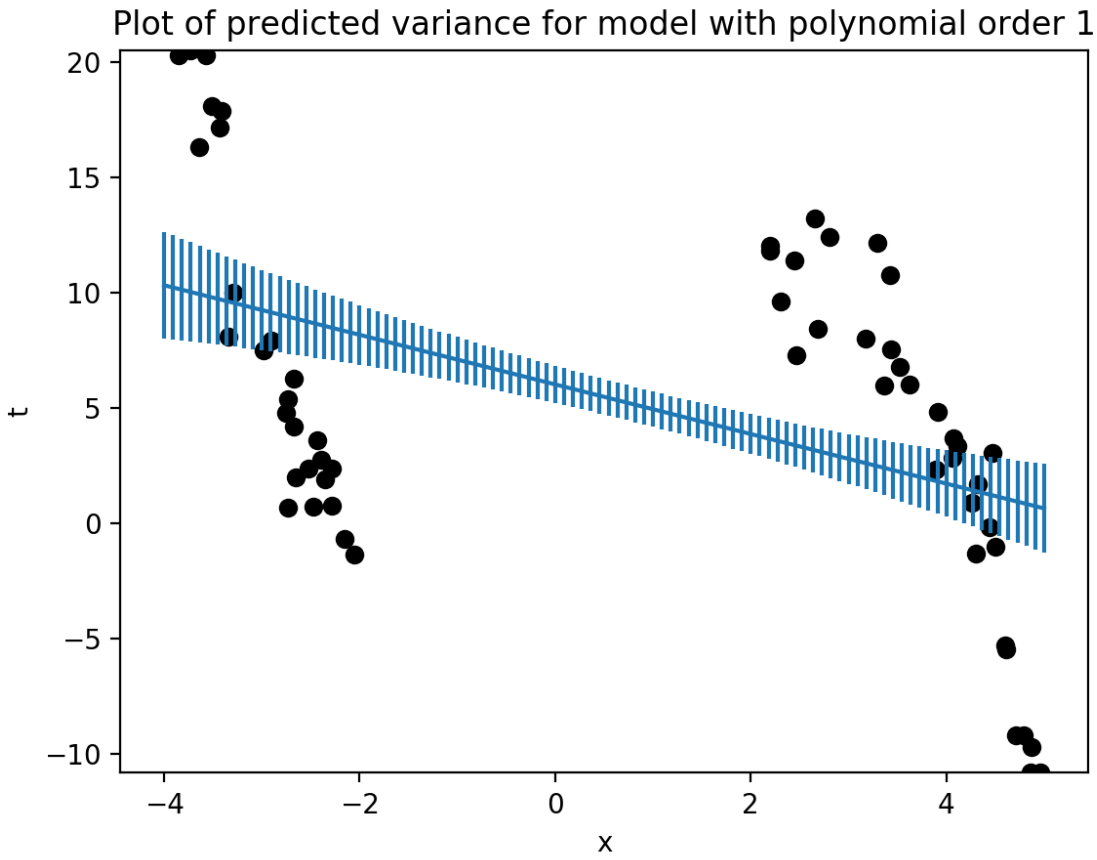

Figure 3: Predicted error model 1: Limited to a straight line, the 1st order model can do little better than estimate a slope through the means of the two clusters.

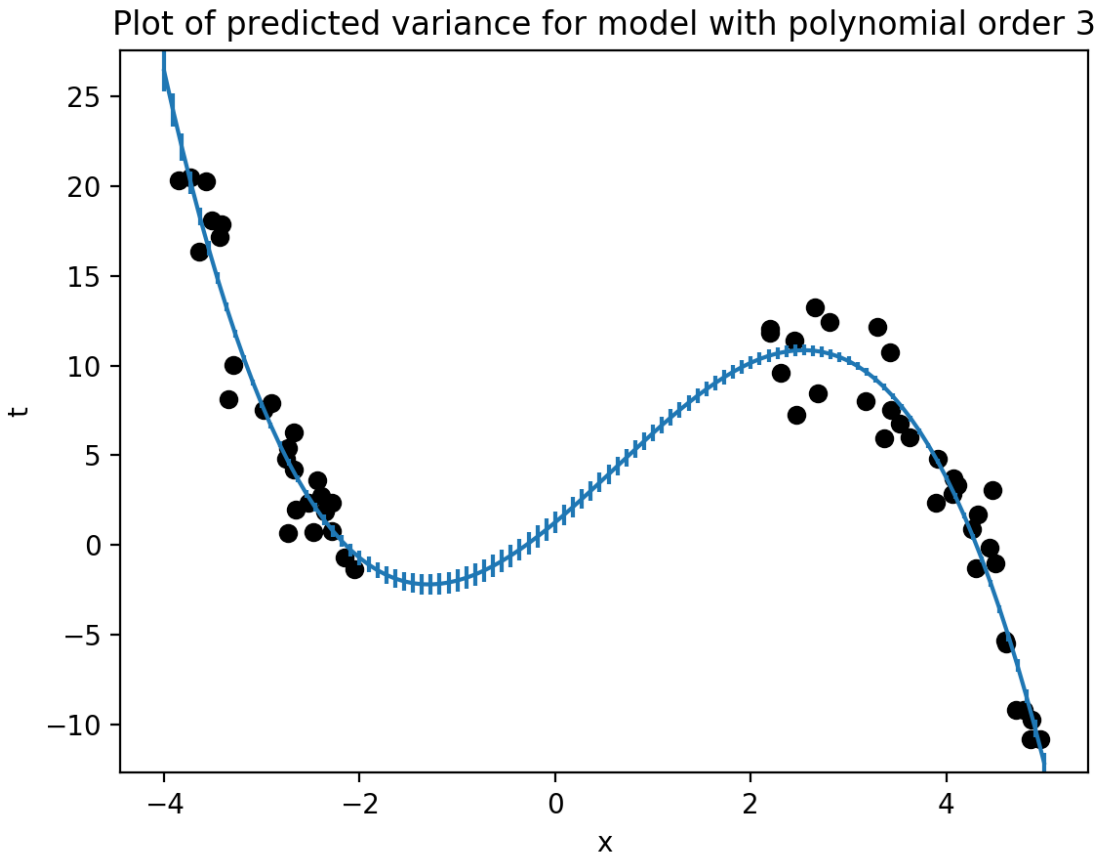

Figure 4: Predicted error model 3: The third order includes enough parameters to fit a curve that closely resembles the original function. The error bars, however, indicate there is some uncertainty in the gap, but it's not much because we are really close to the true function.

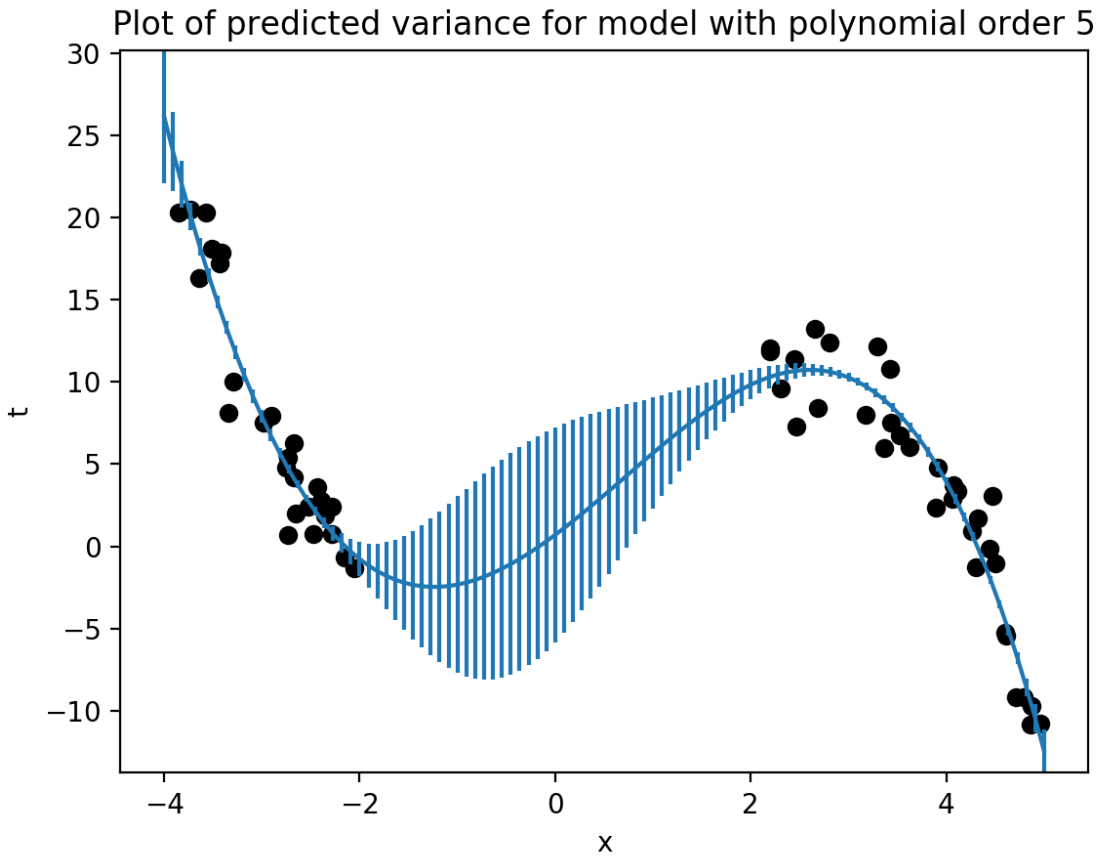

Figure 5: Predicted error model 5: While the fifth order is more flexible through the provided data points, this model is much more uncertain in the gaps as indicated by the wider error bars.

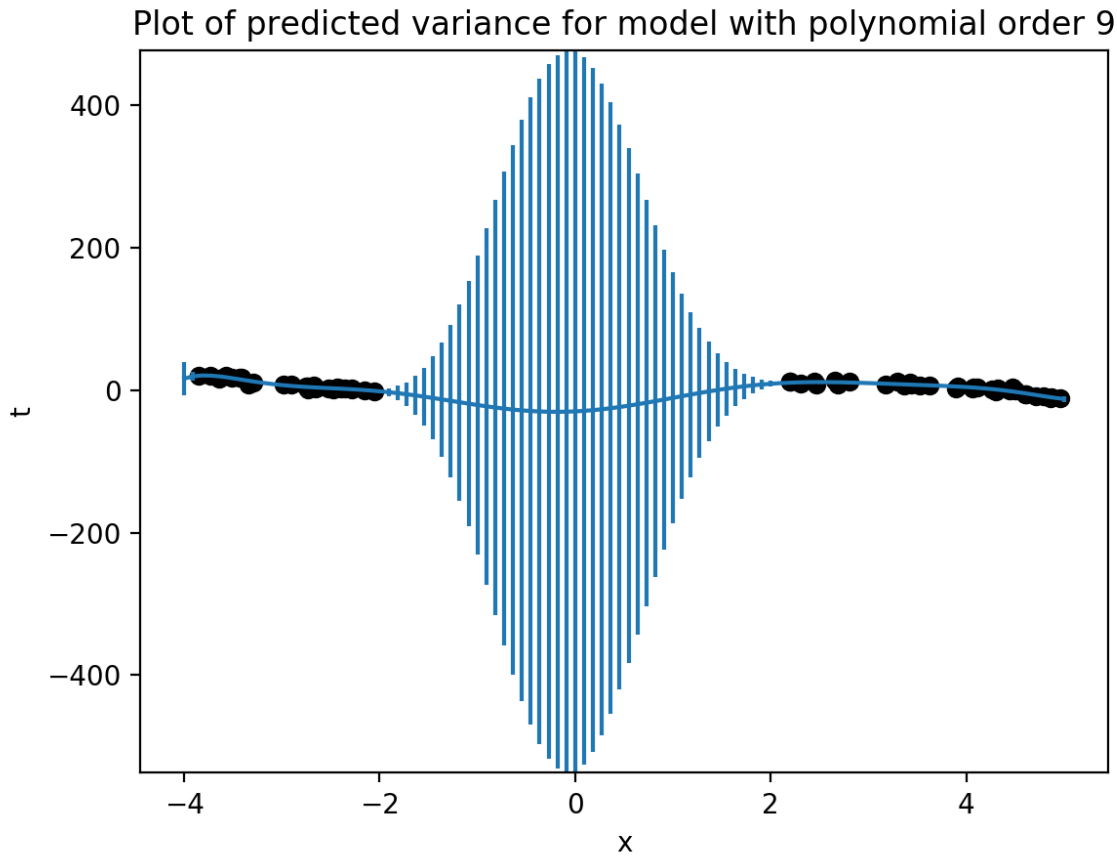

Figure 6: Predicted error model 9: The ninth order can hit almost every provided data point, but the error explodes in the gap, pushing the curve into almost a line to accommodate the wide error margins.

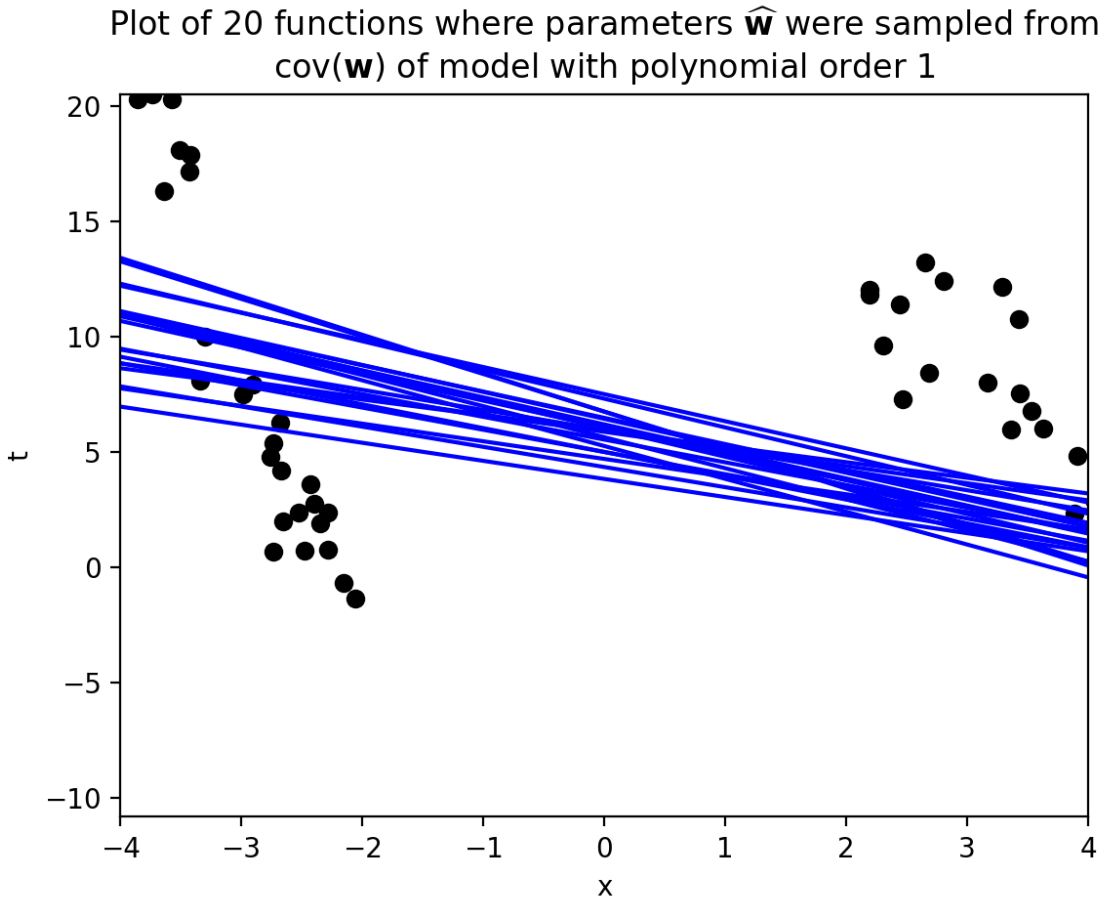

Figure 7: Sampled covariance model 1: The 20 first-order models roughly capture the same slope through the gap, but there is a good deal of variability from line to line.

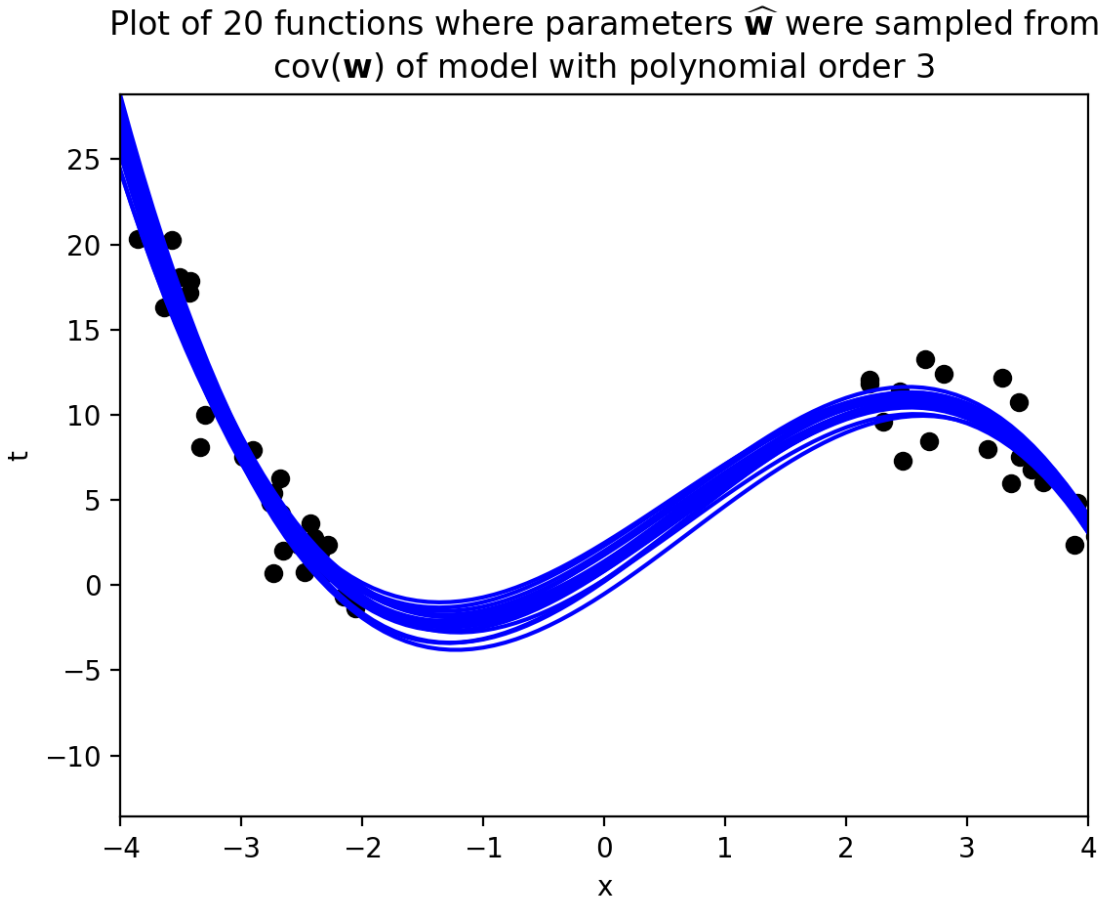

Figure 8: Sampled covariance model 3: The third order, being of the same order as the true function, form almost a single ribbon due to the agreement in the samples.

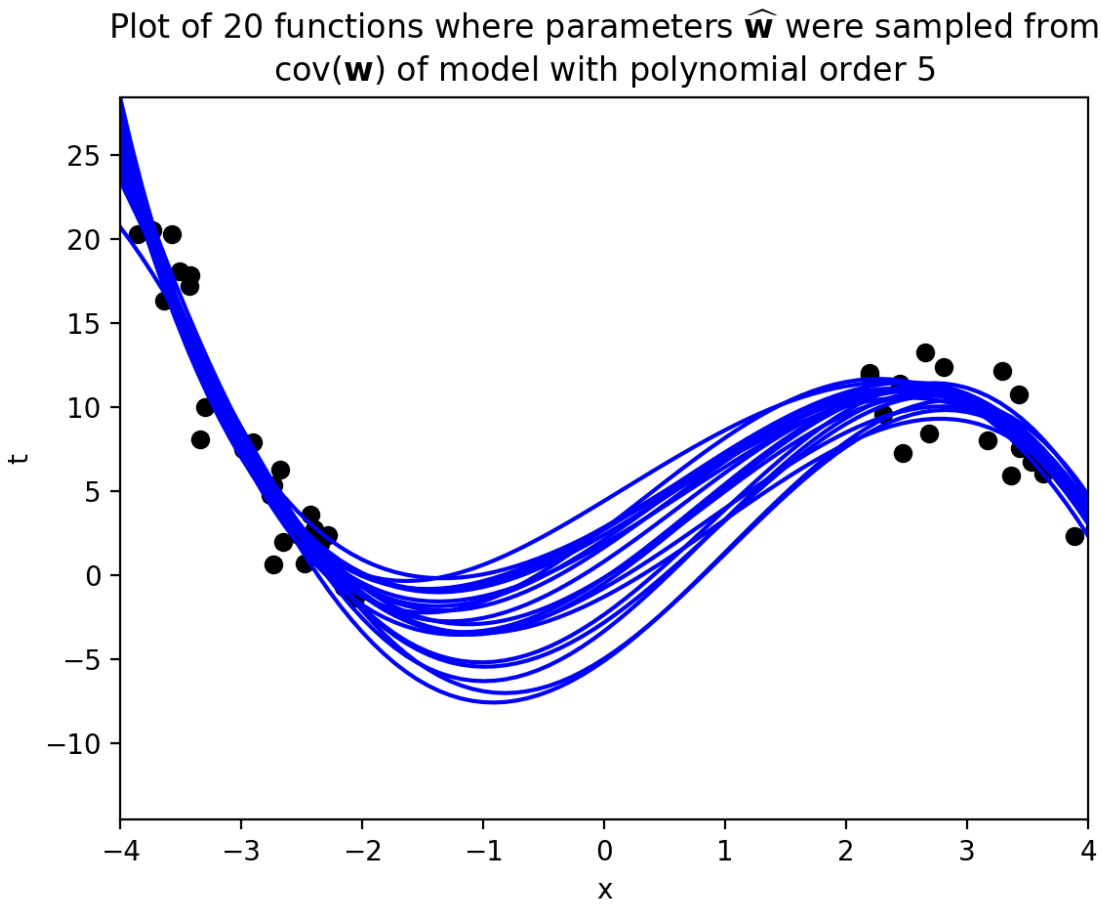

Figure 9: Sampled covariance model 5: The fifth-order curves do well in the places where there is supporting data but begin to vary considerably in how they best estimate how to cross the gap.

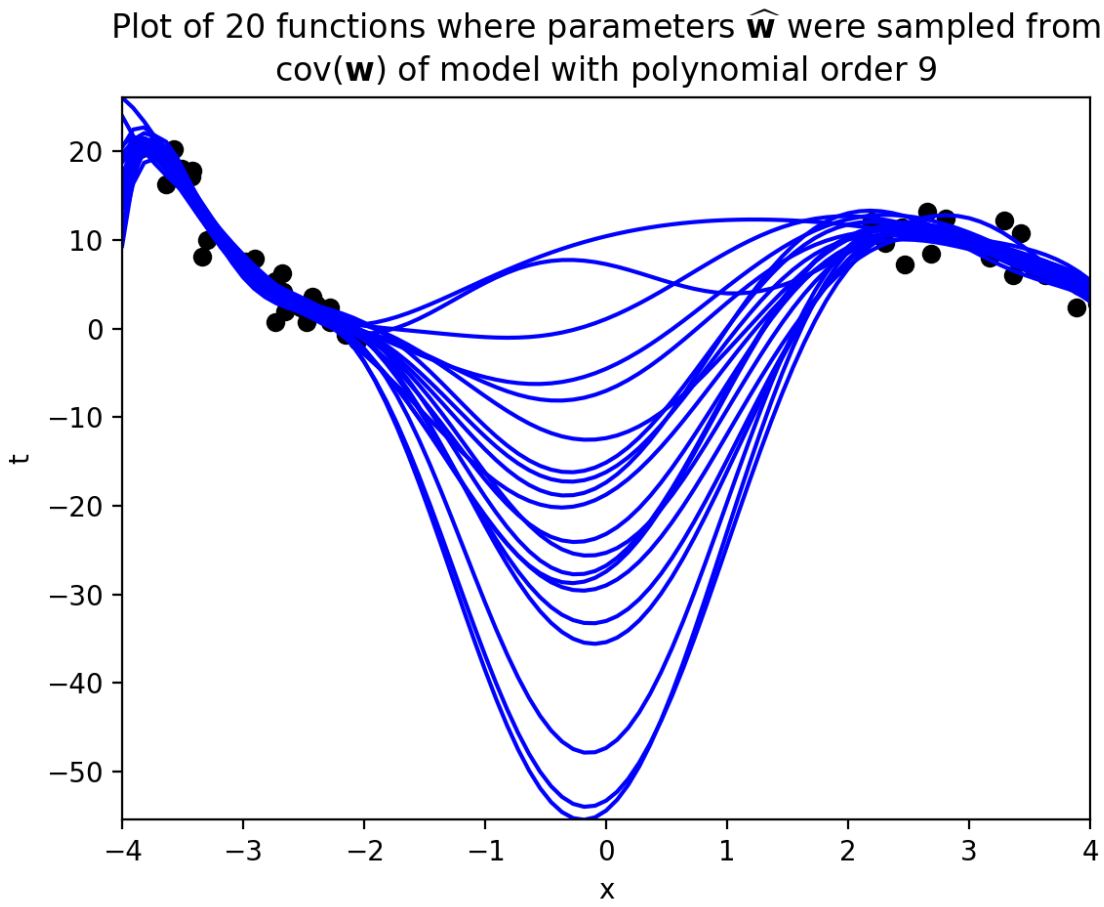

Figure 10: Sampled covariance model 9: The ninth-order model is solid over the provided data but goes fairly wobbly in the gap providing almost unusable data.

When the -2/2 data points are not removed, each successive model over 1 looks pretty good and has very little error.

6. [5 points]

   In this exercise, you will create a new script that demonstrates how model bias impacts variance, similar to the demonstration in Lecture 9 (starting slide 28). In your submission, you will name the file for your script `model_bias_variance.py`. You will copy the functions `true_function` and `sample_from_function` from `predictive_variance.py`. Using `true_function` (which computes $t = 5x + x^2 - 0.5x^3$), generate 20 data sets, each consisting of 25 samples from the true function (using the same range of $x \in [-4.0, 5.0]$ and `noise_var` $= 6$), using the function `sample_from_function`. 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 (in blue) of that model order to each of 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 blue 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.py` as a guide. In your written answer, describe what happens to the variance in the functions in the plots as the model order is changed. (tips: plot

the red true function curve last, so it is plotted on top of the others; also, use `linewidth=3` in the plot fn to increase the line width to make the red true model curve stand out more.)

Solution.

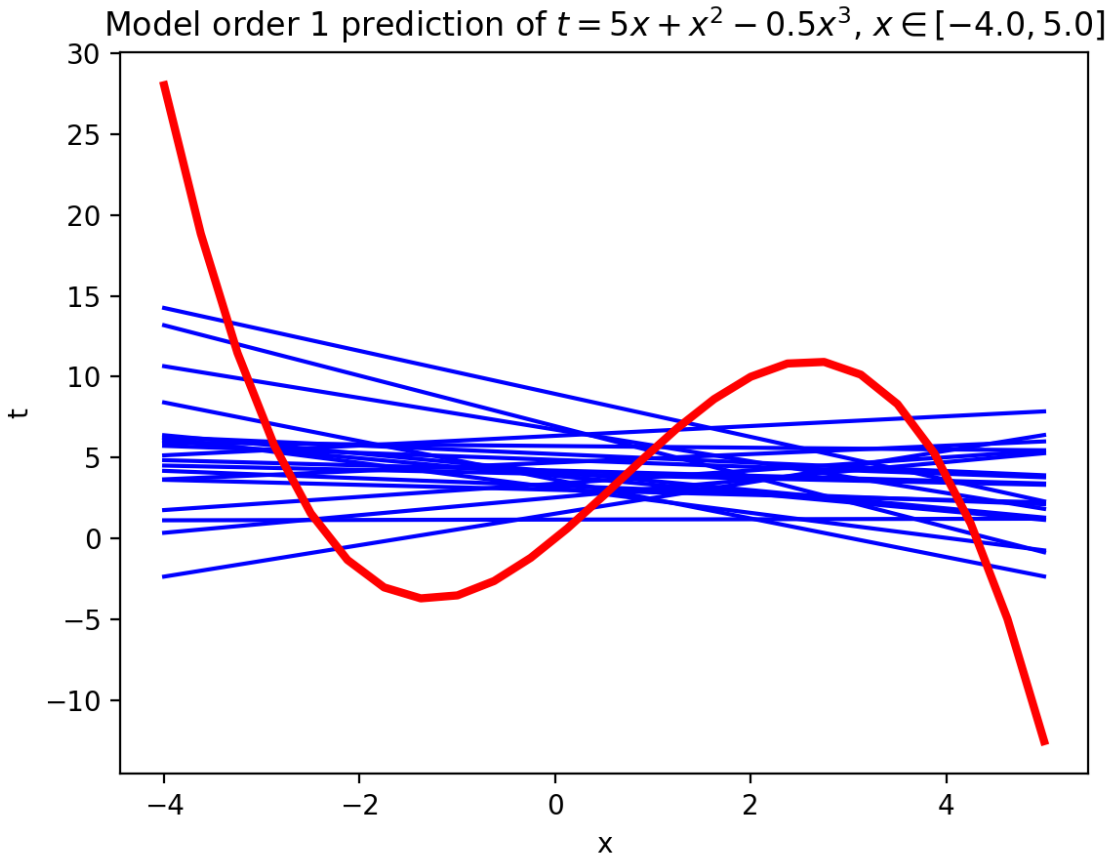

Figure 11: Prediction of function with model order 1: This model contains too little flexibility and so basically predicts a flat line.

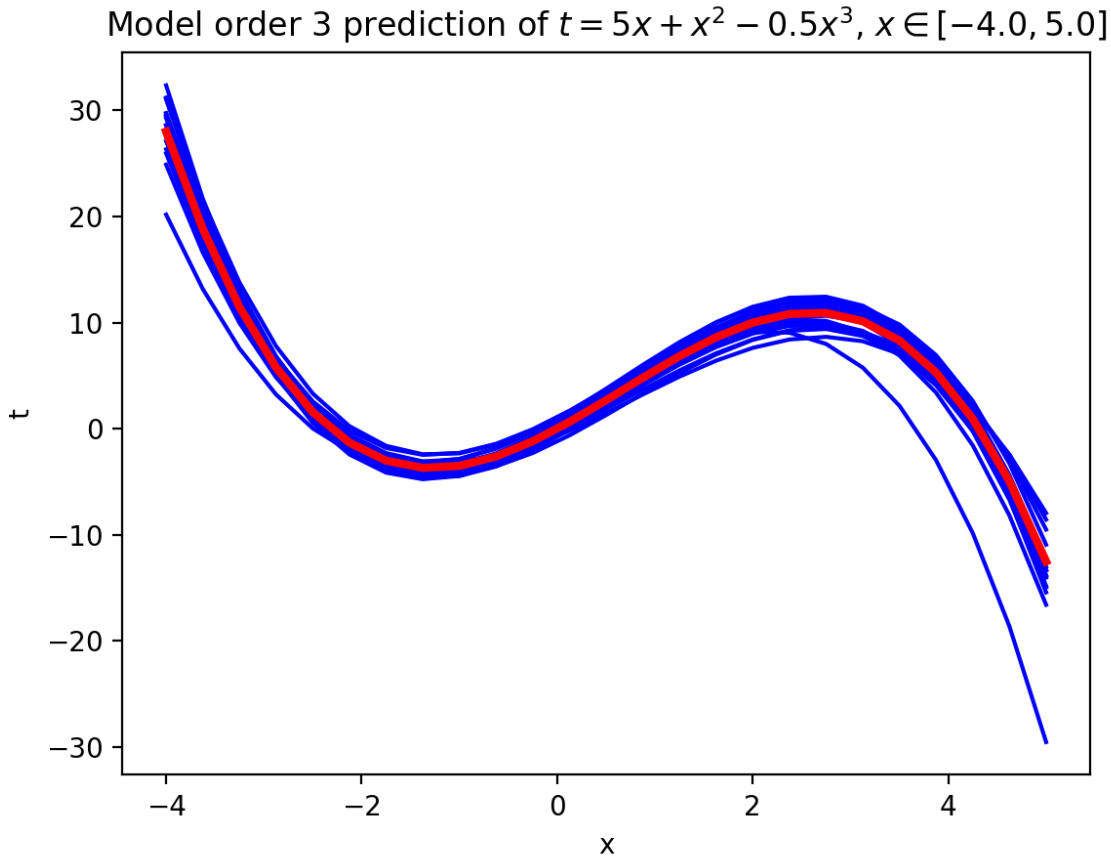

Figure 12: Prediction of function with model order 3: This model is of the same order as the true function and so the lines are almost indistinguishable from each other.

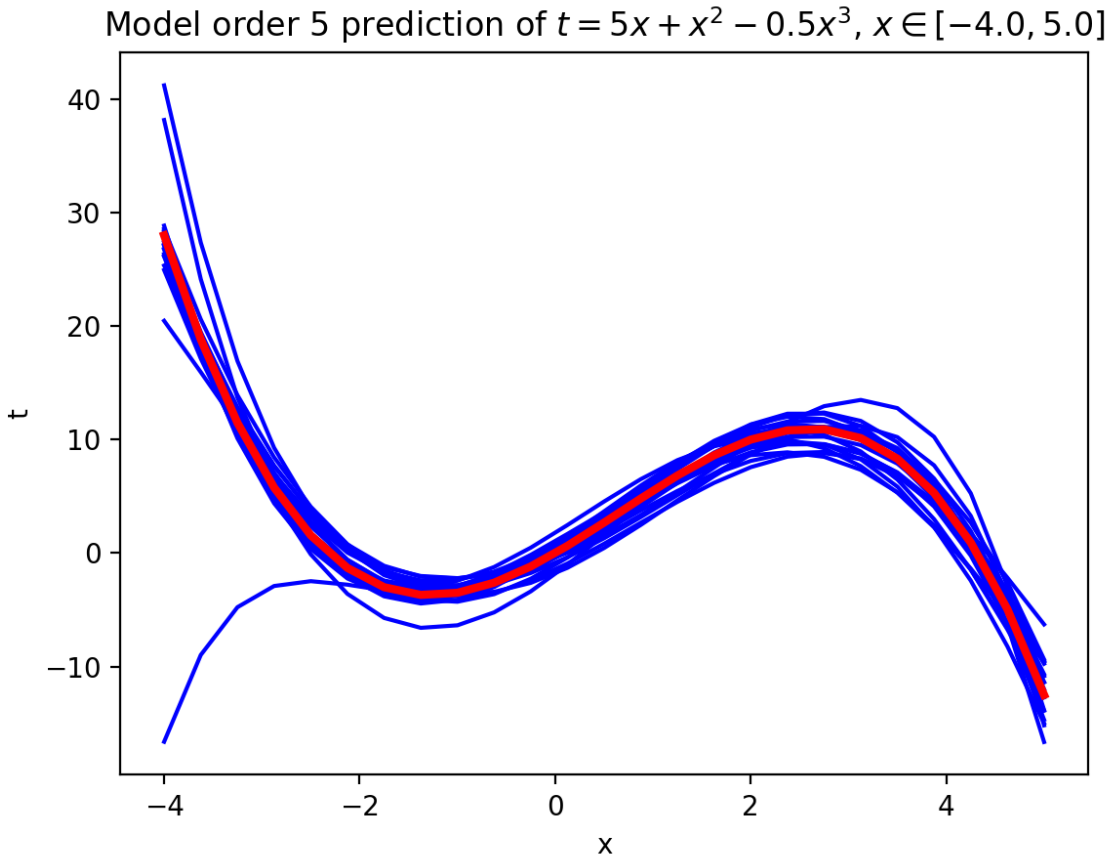

Figure 13: Prediction of function with model order 5: The bias introduced by this more flexible model is shown by how the blue lines are more spread out from each other.

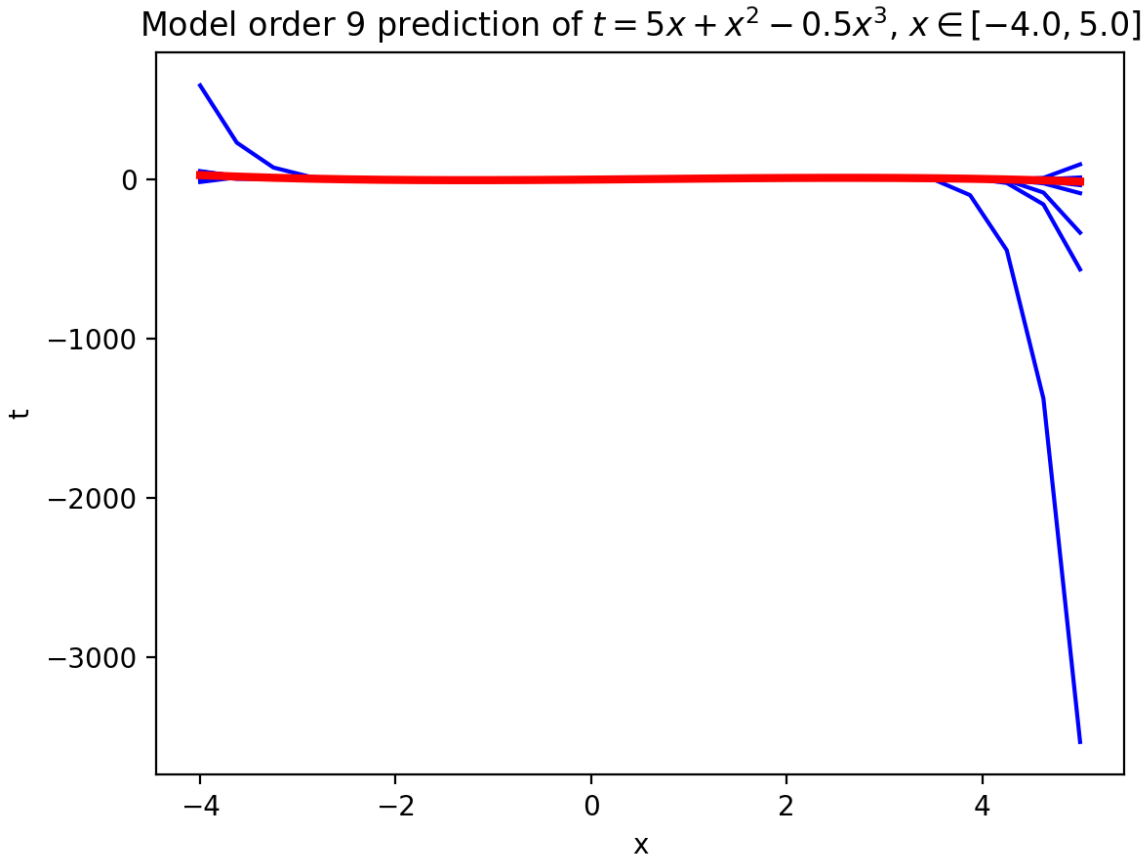

Figure 14: Prediction of function with model order 9: From run to run, this model looks wildly different because of the extremity of the model's order.

The variance increases with model order, introducing greater variability in the predictions.

7. [3 points; **Required only for Graduates**] Adapted from **Exercise 2.13** of FCML:

   Compute the Fisher Information Matrix for the parameter of a Bernoulli distribution.

   Solution.

$$
\begin{aligned}
P(x|q) &= q^x(1-q)^{1-x} \\
\mathcal{L} &= \prod_n^N q^{x_n}(1-q)^{1-x_n} \\
\log \mathcal{L} &= \sum_n^N \log\left(q^{x_n}(1-q)^{1-x_n}\right) \\
\frac{\partial \log \mathcal{L}}{\partial q} &= \frac{\sum_n^N x_n}{q} + \frac{\sum_n^N 1-x_n}{1-q} = q^{-1}\sum_n^N x_n + (1-q)^{-1}\sum_n^N 1-x_n \\
\frac{\partial^2 \log \mathcal{L}}{\partial q} &= -q^{-2}\sum_n^N x_n - (1-q)^{-2}\sum_n^N 1-x_n \\
\mathcal{I} &= \mathrm{E}_{P(x|q)}\{-\frac{\partial^2 \log \mathcal{L}}{\partial q}\} \\
\mathrm{E}_{P(x)}\{f(X)\} &= \sum_x f(x)P(x) \\
\mathcal{I} &= \sum_x -\frac{\partial^2 \log \mathcal{L}}{\partial q}P(x|q) \\
P(x|q) &= (1*q) + (0*(1-q)) = q \\
\mathcal{I} &= \sum_x -\frac{\partial^2 \log \mathcal{L}}{\partial q}q \\
&= -q\sum_x \frac{\partial^2 \log \mathcal{L}}{\partial q}
\end{aligned}
$$

I think this is probably the answer. I hope I've sufficiently proven $P(x|q) = q$. I confess I do not understand how this is a matrix. Won't this be a scalar value in the end?

I had done some additional distribution of terms which I'm not sure is necessary:

$$
\begin{aligned}
\mathcal{I} &= \sum_{x=n}^N \left[-\left(-q^{-2}\sum_n^N x_n - (1-q)^{-2}\sum_n^N 1-x_n \right) q \right] \\
&= \sum_{x=n}^N \left[q\left(\frac{\sum_n^N x_n}{q^2} + \frac{\sum_n^N 1-x_n}{(1-q)^2} \right) \right] \\
&= \sum_{x=n}^N \left[q\left(\frac{\sum_n^N x_n}{q^2} + \frac{N-\sum_n^N x_n}{(1-q)^2} \right) \right] \\
&= \sum_{x=n}^N \left[\frac{q\sum_n^N x_n}{q^2} + \frac{qN - q\sum_n^N x_n}{(1-q)^2} \right] \\
&= \sum_{x=n}^N \left[\frac{\sum_n^N x_n}{q} + \frac{qN - q\sum_n^N x_n}{(1-q)^2} \right]
\end{aligned}
$$

I have no idea how to handle the outer summation over the inner summation or if I should be worried about using different subscripts.