1.a

We are given an exponential family distribution where the natural parameter, η, is scalar and we assume the sufficient statistic to take the form $T(y) = y$, yielding

$$p(y; n) = b(y) \exp(\eta y - a(\eta)).$$

We seek to derive an expression for the mean of the distribution. Given that y is a continuous random variable, the mean (or expected value) is defined as

$$\mathbb{E}[y; \eta] = \int y p(y; \eta) dy = \int y b(y) \exp(\eta y - a(\eta)) dy. \tag{1}$$

We can explore simplifying the integral by taking the partial derivative of the probability density function wrt η,

$$\frac{\partial}{\partial \eta} p(y; \eta) = b(y) \exp(\eta y - a(\eta))(y - \frac{\partial}{\partial \eta} a(\eta)). \tag{2}$$

Looking at the expansion above, we can substitute (1) into (2) once we take the integral,

$$\begin{aligned}
\int \frac{\partial}{\partial \eta} p(y; \eta) dy &= \int b(y) \exp(\eta y - a(\eta))(y - \frac{\partial}{\partial \eta} a(\eta)) dy \\
&= \mathbb{E}[y; \eta] - \int \frac{\partial}{\partial \eta} a(\eta) p(y; \eta) dy \\
&= \mathbb{E}[y; \eta] - \frac{\partial}{\partial \eta} a(\eta) \int p(y; \eta) dy.
\end{aligned}$$

We note that the improper integral of a PDF is equal to 1 according to its properties. Rearranging, this yields the expected value

$$\begin{aligned}
\mathbb{E}[y; \eta] &= \frac{\partial}{\partial \eta} \int p(y; \eta) dy + \frac{\partial}{\partial \eta} a(\eta) \int p(y; \eta) dy \\
&= \frac{\partial}{\partial \eta} (1) + \frac{\partial}{\partial \eta} a(\eta)(1) \\
&= \frac{\partial}{\partial \eta} a(\eta).
\end{aligned}$$

1.b

The variance for the distribution is given by

$$Var(Y;\eta) = \mathbb{E}[(Y - \mathbb{E}(Y))^2] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2.$$

In (1.a) we found that $\mathbb{E}[y;\eta] = \frac{\partial}{\partial \eta} a(\eta)$. Substituting the expected value into the above form and using the definition, we have

$$\begin{aligned}
\mathbb{E}[Y^2] - \mathbb{E}[Y]^2 &= \int y^2 p(y;\eta) dy - (\frac{\partial}{\partial \eta} a(\eta))(\frac{\partial}{\partial \eta} a(\eta)) \\
&= \int y^2 p(y;\eta) dy - (\frac{\partial}{\partial \eta} a(\eta)) \int y p(y;\eta) dy \\
&= \int (y - \frac{\partial}{\partial \eta} a(\eta)) y p(y;\eta) dy \\
&= \int (y - \frac{\partial}{\partial \eta} a(\eta)) y b(y) \exp(\eta y - a(\eta)) dy.
\end{aligned}$$

Looking at this distributed form, we can observe a likeness in taking the derivative of the exponential family with respect to the natural parameter,

$$\frac{\partial}{\partial \eta} y p(y;\eta) = y b(y) \exp(\eta y - a(\eta))(y - \frac{\partial}{\partial \eta} a(\eta)).$$

Substituting into integral above, we reduce to a shortened expression for the variance

$$\begin{aligned}
\mathbb{E}[Y^2] - \mathbb{E}[Y]^2 &= \int \frac{\partial}{\partial \eta} y p(y;\eta) dy \\
&= \frac{\partial}{\partial \eta} \int y p(y;\eta) dy \\
&= \frac{\partial}{\partial \eta} (\frac{\partial}{\partial \eta} a(\eta)) \\
&= \frac{\partial^2}{\partial \eta^2} a(\eta).
\end{aligned}$$

1.c

The loss function is given by

$$L(\theta) = -p(y;\eta) = -p(y|x;\theta).$$

We get the NLL by taking the natural logarithm of the loss function:

$$\begin{aligned} l(\theta) = ln(L(\theta)) &= ln(-b(y)\exp(\eta y - a(\eta))) \\ &= -ln(b(y)) - ln(\exp(\eta y - a(\eta))) \\ &= -ln(b(y)) - \eta y + a(\eta) \\ &= -ln(b(y)) - \theta^T xy + a(\theta^T x). \end{aligned}$$

Here, it is the case that we can write the Hessian of the loss wrt θ as $\nabla^2_\theta l(\theta) = \nabla_\theta(\nabla_\theta l(\theta))^T$. Taking the partial derivatives, we get

$$\begin{aligned} \nabla^2_\theta l(\theta) &= \nabla_\theta(-xy + \frac{\partial}{\partial\theta}a(\theta^T x)x)^T \\ &= \frac{\partial^2}{\partial\theta^2}a(\theta^T x)xx^T. \end{aligned}$$

Since we assumed $\eta = \theta^T x$, we can say that $\frac{\partial^2}{\partial\theta^2}a(\theta^T x)$ is equivalently $Var(Y;\eta)$ that we found in (1.b). This means that the Hessian is simply

$$\nabla^2_\theta l(\theta) = Var(Y;\eta)xx^T.$$

We know that the variance of any probability distribution is non-negative. We also know that xx^T is non-negative since this is equivalently the square vector of all entries in x, squared. Therefore, $\nabla^2_\theta l(\theta) \geq 0$ and is PSD. We conclude that NLL loss of GLM is convex.

2.a

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\theta^T \hat{x}^{(i)} - y^{(i)})^2$$

Differentiating this objective, we get:

$$\nabla_\theta J(\theta) = \sum_{i=1}^{n}(\theta^T \hat{x}^{(i)} - y^{(i)})(\hat{x}^{(i)})$$

The gradient descent update rule is

$$\theta := \theta - \lambda \nabla_\theta J(\theta)$$

which reduces here to:

$$\theta := \theta - \lambda \sum_{i=1}^{n}(\theta^T \hat{x}^{(i)} - y^{(i)})(\hat{x}^{(i)})$$

2.c

Increasing the degree of the polynomial fitting function essentially increases the sensitivity of the feature map to the data. We see that lower values of k yield linear fitting functions that capture the general decreasing trend in the data. As k increases, the prediction improves by capturing the sinusoidal trend. We see that particularly high values of k such as $k = 20$ are sensitive to small perturbations in the data and will attempt to fit outliers.

2.e

The addition of $sin(x)$ to the feature map yields more accurate fitting functions at lower values of k. This is because the polynomial features are dominated by sine for $k < 10$. We still see fitting instability at higher values of k as perturbations in the data affect the prediction.

2.g

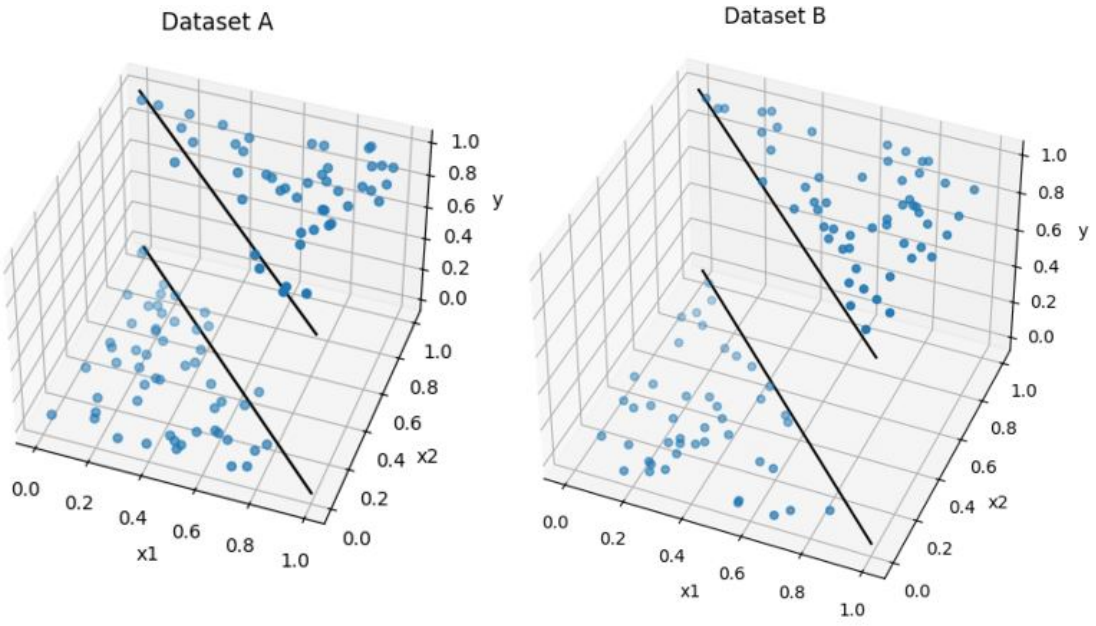
The plots demonstrate that "overfitting" generally yields well-fitting models within the range of the data provided, but would fail to predict extrapolated data values outside the given range. At higher values of k, the model diverges for $x \notin [-4, 6]$. The exercise demonstrates the importance of understanding the scope of the model and moderating the number of features according to the training data.

3.a

The most notable difference in training the logistic regression model on datasets A and B is that, as the problem states, the training algorithm converges for A and never converges for B. That is, the logistic regression algorithm is able to find a set of parameters, θ, for which the loss function is minimized to a small tolerance for dataset A only. Both use the same constant learning rate, $\alpha = 0.1$.

3.b

After inspecting the differences between the two logistic regression algorithms within the code, I found that the parameters continue to grow without bound for dataset B. This is because as the algorithm iterates, the gradient of the loss function fails to converge. I generated scatter plots of both datasets, which revealed a key difference in the two datasets. Namely, that A has data that cross the linear decision boundary and B does not. This is seen in the figure below.



The total separation across the linear decision boundary for dataset B is likely the cause of the non-convergence, because the algorithm may be hunting for a non-unique solution, or it may be numerically unstable.

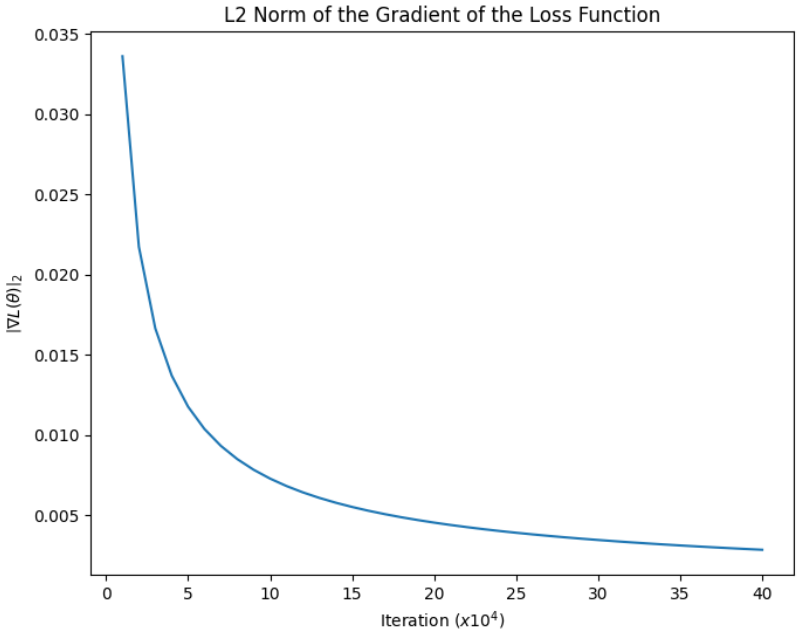
I then examined the gradient of the loss function as the training loop iterated for dataset B, and took the L2 norm depicted in the plot below.

We see that as the training algorithm continues to iterate, the gradient of the loss function flattens out. I believe that this means the algorithm is stuck at a saddle point of the loss function, which means that the gradient never approaches 0 and the method fails to converge.

These explanations do not apply to dataset A because the gradient norm converges to near machine precision in roughly 30,000 iterations and the dataset contains data that cross the linear decision boundary.

3.c

i. Using a different constant learning rate - No; changing the learning rate will not overcome the fundamental issue of linearly separable data.
ii. Decreasing the learning rate over time - No; same as above.
iii. Linear scaling of the input features - No; again, scaling the input features does not address linear separability about the decision boundary.
iv. Adding a regularization term $\|\theta\|_2^2$ to the loss function - Yes; in this case, I observed that the parameters diverge slowly. A regularization term could help minimize the fitting coefficients.
v. Adding zero-mean Gaussian noise to the training data or labels - Probably; adding noise would most likely address the issue of linear separability, especially for a dataset such as B where a few data points are extremely close to the decision boundary.



Documentation: I did not collaborate with anyone on this assignment.