This handout includes space for every question that requires a written response. Please feel free to use it to handwrite your solutions (legibly, please). If you choose to typeset your solutions, the `README.md` for this assignment includes instructions to regenerate this handout with your typeset LaTeX solutions.

1.a

Since

$$\int\limits_{-\infty}^{\infty} p(y;\eta)dy = 1$$

then

$$\frac{\partial}{\partial\eta}\int\limits_{-\infty}^{\infty} p(y;\eta)dy = \int\limits_{-\infty}^{\infty} \frac{\partial}{\partial\eta}p(y;\eta)dy = 0$$

The probability mass function looks as follows:

$$p(y;\eta) = b(y)e^{\eta y - a(\eta)}$$

Its partial derivative with respect to η:

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

Let's put it into the integral mentioned above:

$$\int\limits_{-\infty}^{\infty}\frac{\partial}{\partial\eta}p(y;\eta)dy = \int\limits_{-\infty}^{\infty}p(y;\eta)(y-\frac{\partial a(\eta)}{\partial\eta})dy = \int\limits_{-\infty}^{\infty}p(y;\eta)ydy - \int\limits_{-\infty}^{\infty}\frac{\partial a(\eta)}{\partial\eta}dy$$

since the first part is an expectation and $a(\eta)$ does not depend on y, then

$$\int\limits_{-\infty}^{\infty}p(y;\eta)ydy - \int\limits_{-\infty}^{\infty}\frac{\partial a(\eta)}{\partial\eta}dy = \mathbb{E}[Y;\eta] - \frac{\partial a(\eta)}{\partial\eta} = 0$$

therefore

$$\mathbb{E}[Y;\eta] = \frac{\partial a(\eta)}{\partial\eta}$$

1.b

Let us recall the following:

$$\frac{\partial}{\partial\eta}\int\limits_{-\infty}^{\infty} p(y;\eta)dy = \int\limits_{-\infty}^{\infty} \frac{\partial}{\partial\eta}p(y;\eta)dy = 0$$

This is also valid:

$$\frac{\partial^2}{\partial\eta^2}\int\limits_{-\infty}^{\infty} p(y;\eta)dy = \int\limits_{-\infty}^{\infty} \frac{\partial^2}{\partial\eta^2}p(y;\eta)dy = 0$$

From the previous task we know that:

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

Let us calculate the second derivative of p with respect to η:

$$\frac{\partial^2}{\partial\eta^2}p(y;\eta) = \frac{\partial}{\partial\eta}(p(y;\eta)(y - \frac{\partial a(\eta)}{\partial\eta})) = (y - \frac{\partial a(\eta)}{\partial\eta})\frac{\partial p(y;\eta)}{\partial\eta} - p(y;\eta)\frac{\partial^2 a(\eta)}{\partial\eta^2} = (y - \frac{\partial a(\eta)}{\partial\eta})^2 p(y;\eta) - p(y;\eta)\frac{\partial^2 a(\eta)}{\partial\eta^2}$$

Therefore:

$$\int\limits_{-\infty}^{\infty} \frac{\partial^2}{\partial\eta^2}p(y;\eta)dy = \int\limits_{-\infty}^{\infty} (y - \frac{\partial a(\eta)}{\partial\eta})^2 p(y;\eta) - p(y;\eta)\frac{\partial^2 a(\eta)}{\partial\eta^2}dy = 0$$

and

$$\int\limits_{-\infty}^{\infty} (y - \frac{\partial a(\eta)}{\partial\eta})^2 p(y;\eta)dy = \int\limits_{-\infty}^{\infty} p(y;\eta)\frac{\partial^2 a(\eta)}{\partial\eta^2}dy = \frac{\partial^2 a(\eta)}{\partial\eta^2}$$

since we know that $\mathbb{E}[Y;\eta] = \frac{\partial a(\eta)}{\partial\eta}$, then let's rewrite the expression above taking this into account:

$$\int\limits_{-\infty}^{\infty} (y - \mathbb{E}[Y;\eta])^2 p(y;\eta)dy = \frac{\partial^2 a(\eta)}{\partial\eta^2}$$

The integral is actually variance, since by definition:

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

Therefore:

$$Var[Y;\eta] = \frac{\partial^2 a(\eta)}{\partial\eta^2}$$

1.c

Starting with that probability mass function looks as follows:

$$p(y;\eta) = b(y)e^{\eta y - a(\eta)}$$

For x_i, y_i and $\eta_i = \theta^T x_i$ the likelihood will be:

$$\log p(y_i \mid \eta_i) = \log p(y_i \mid x_i;\theta) = \log b(y_i) + \eta_i y_i - a(\eta_i)$$

Since $b(y_i)$ is independent from θ, the negative log loss:

$$\ell(\theta) = \sum_{i=1}^{n} \left[a(\theta^\top x_i) - y_i\,\theta^\top x_i\right]$$

The gradient:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \left(\frac{da(\eta_i)}{d\theta} x_i - y_i x_i\right) = \sum_{i=1}^{n} \left(\frac{da(\eta_i)}{d\eta_i}\frac{d\eta_i}{d\theta} - y_i x_i\right) = \sum_{i=1}^{n} \left(\frac{da(\eta_i)}{d\eta_i} - y_i\right) x_i$$

The Hessian:

$$\nabla^2_\theta \ell(\theta) = \sum_{i=1}^{n} \frac{d^2 a(\eta_i)}{d\eta_i^2}\, x_i x_i^\top$$

we know from reasoning in the previous tasks that $\frac{d^2 a(\eta_i)}{d\eta_i^2}$ is variance and it's a non-negative value. $x_i x_i^\top$ is always PSD because for any vector $z \in \mathbb{R}^d$

$$z^\top (x_i x_i^\top) z = (z^\top x_i)(x_i^\top z) = (x_i^\top z)^2 \geq 0$$

Therefore

$$\nabla^2_\theta \ell(\theta) = \sum_{i=1}^{n} \frac{d^2 a(\eta_i)}{d\eta_i^2}\, x_i x_i^\top \geq 0$$

2.a

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

Differentiating this objective, we get:

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

The gradient descent update rule is

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

which reduces here to:

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

2.c

For value k=1 the regression becomes the line $1 + \theta_1 x$. It is underfitting.

For value k=2 the regression becomes the function $1 + \theta_1 x + \theta_2 x^2$. Here we can see that the line is almost the same as the function above. This means that the calculated value of θ_2 is small compared to θ_1. The loss function is comparable with the function above. The regression underfits.

For value k=3 the regression contains cubic components and is still underfitting, but starts to fit better due to presence of cubic component.

For value k=5 and k=10 it seems the regression has the best fit and generalizes well.

For value k=20 the regression is overfitting and starts catching the fluctuations of input data.

2.e

For values k=1, 2, 3, 5 the regression fits well. For value k=10 the regression has the best fit. For value k=20 the regression is overfitting and starts catching the fluctuations of input data.

The difference is that the fitted model is different and the sinusoid function is used instead of polynome. The sinusoid fits better from the start compared to polynome.

2.g

There is lack of input points, however, the logic remains the same - for k=1,2,3 we see that the polynome line is not going through the points and not fitting well. For k = 5 it fits the best and for higher values it's overfitting. Definitely it's required to add more data for the model to predict better. Otherwise, the model will perform well on training and much worse on test data.

For sinusoidal function it goes a bit better from the start but starts to overfit from k=5.

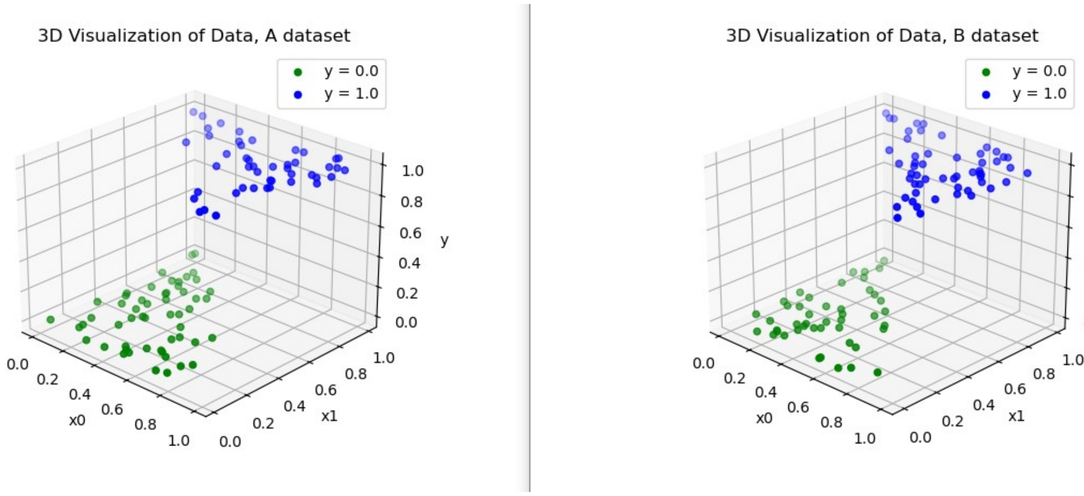

Figure 1: A vs B dataset

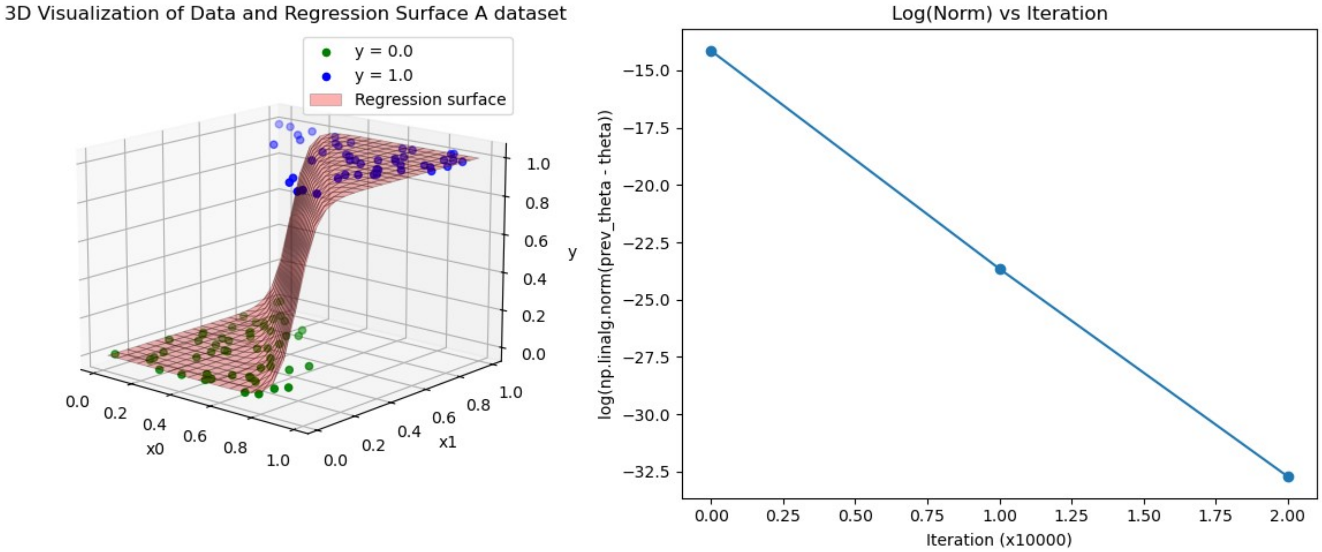

Figure 2: Dataset A

3.a

In A dataset the data for y=1.0 and y=0.0 is overlapping on x0, x1 scale. It means that there is a zone in x0, x1 plane where the points have randomly y=1.0 and y=0.0 values.

The highest difference is that regression doesn't converge that quick on dataset B as it converges on dataset A.

3.b

The specifics of the dataset A implies some fluctuation of the points, that do not allow the loss function to closely adapt to the points. Finally, the change of the loss function becomes negligible with respect to iteration. See figure 2.

For dataset A:
Theta 10K iterations: [-20.81394174 21.45250215 19.85155266]
Theta 20K iterations: [-20.81437785 21.45295156 19.85198173]
Theta 30K iterations: [-20.81437788 21.45295159 19.85198176]

Dataset B is clearly separable. On dataset B the model catches well the separation. However θ parameter increases to really high values, due to the specifics of dataset B. This leads to situation, when each new calculated value of vector

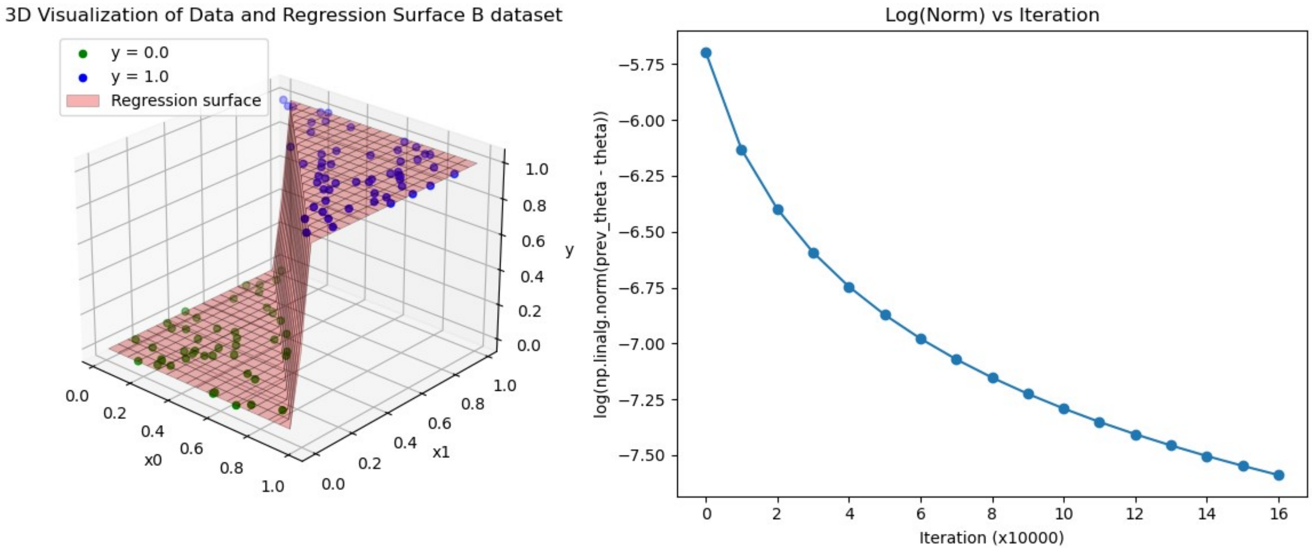

Figure 3: Dataset B

of θ values hugely differs from the previous value. Each module of θ is higher than the previous one. Nevertheless, the regression converges, but very slowly - see figure 3.

For dataset B:
Theta 140K iterations: [-136.0833065 136.45725847 136.09754179]
Theta 150K iterations: [-139.33136783 139.73026941 139.34109305]
Theta 160K iterations: [-142.43475235 142.85857494 142.43994361]
Theta 170K iterations: [-145.40867865 145.85731947 145.40934321]

3.c

Using different constant learning rate won't help as it won't fix the behaviour of logistic regression on the data such as dataset B. The decreasing of learning rate will result in fact that the regression on dataset A will converge much slower.

Decreasing learning rate over time could help. However it also slows down the convergence if it becomes small from the very beginning (even for A type of dataset) and by this logic won't allow logistic regression to learn quickly from the data. In this case we will get a degradation of learning procedure and bad fit of θ parameters to the data.

Linear scaling of the input features won't help. Because it doesn't fix the problem - on B type of datasets. Log regression will try to learn high values of θ for perfect separation.

Adding a regularization term to the loss function would help as it will help to avoid large weights and regression will converge.

Adding zero-mean Gaussian noise could help. Because the data won't be linearly separable. However, it won't be stable on the same data for each session of learning, because of the probability of Gaussian noise.