Scott McHaffie - ASSIGNMENT I

Regression and Classification

SH2150 Machine Learning in Physics

October 2025

You are using a very fast detector to measure the energy spectrum of gamma photons emitted by an unknown radioactive nuclide. After some initial experiments you conclude that this nuclide emits two gamma rays with energies E_A (first photon) and E_B (second photon) in very rapid succession, with a time difference on the order of a picosecond. The photons are emitted in random directions, and therefore you only detect one of them in most cases. However, in some rare cases you manage to measure both and determine the order they arrive in so that you can label them A and B.

The photons interact with the detector through compton scattering and you measure the energy E and scattering angle θ for each photon. Some photons also photointeract in the detector, but their energy is outside the detector's range of validity ($E > 650$ keV) so you cannot get a trustworthy energy measurement from these, and therefore you discard them.

Your mission is to:

- classify the photon events into two categories, namely A and B,
- estimate the incident photon energies E_A and E_B,
- and finally identify the nuclide.

1. Logistic Regression Model

The logistic regression model for classification is given by

$$\hat{y} = \sigma(\mathbf{W}\vec{x} + b) = \frac{1}{1 + e^{-(\mathbf{W}\vec{x}+b)}}$$

where we assume that the target variables $y^{(i)}$ are independently Bernoulli distributed with parameters $\hat{y}^{(i)} = \sigma(\mathbf{W}\vec{x}^{(i)} + b)$ for $i = 1, \ldots, n$. Observe that $\hat{y} \in (0, 1)$ and can therefore be interpreted as a probability.

(a) (2p) Show that fitting the model by maximizing the likelihood $L(\mathbf{W}, b|\vec{x}, y)$ of the data $(\vec{x}^{(i)}, y^{(i)})_{i=1}^{n}$ under the model w.r.t. $\mathbf{W}, b$, is equivalent to minimizing the cross-entropy (CE) loss defined by

$$L_{CE}(\mathbf{W}, b) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log\left(1 - \hat{y}^{(i)}\right)$$

(Hint: write out the log-likelihood of the Bernoulli distribution and simplify).

Answer: We know from class that when we have:

$$\begin{cases} y^{(i)} = 1 \text{ with probability } \hat{y}^{(i)}, \text{ and} \\ y^{(i)} = 0 \text{ with probability } 1 - \hat{y}^{(i)} \end{cases}$$

then we can write the Bernoulli distribution as:

$$p(\mathbf{y} \mid \hat{\mathbf{y}}) = \prod_{i=1}^{n} \hat{y}^{(i)\, y^{(i)}} \left(1 - \hat{y}^{(i)}\right)^{1-y^{(i)}}. \tag{1}$$

Taking the log of this distribution gives:

$$\begin{aligned}
\log\left(p(\mathbf{y} \mid \hat{\mathbf{y}})\right) &= \log\left(\prod_{i=1}^{n} \hat{y}^{(i)\, y^{(i)}} \left(1 - \hat{y}^{(i)}\right)^{1-y^{(i)}}\right) \\
&= \sum_{i=1}^{n} \log\left[\hat{y}^{(i)\, y^{(i)}}\right] + \log\left[\left(1 - \hat{y}^{(i)}\right)^{1-y^{(i)}}\right] \\
&= \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log\left(1 - \hat{y}^{(i)}\right)
\end{aligned}$$

Normalizing based on the size of the dataset gives:

$$\boxed{\log\left(p(\mathbf{y} \mid \hat{\mathbf{y}})\right) = \frac{1}{n} \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log\left(1 - \hat{y}^{(i)}\right) \equiv -L_{CE}(\mathbf{W}, b)}. \tag{2}$$

Since the log of the Bernoulli distribution is equal to the negative of $L_{CE}(\mathbf{W}, b)$ then we can see that maximizing Equation 2 is equivalent to minimizing $L_{CE}(\mathbf{W}, b)$.

(b) (2p) Derive an expression for the decision boundary when we have two predictor variables. That is, if $\vec{x} = (x_1, x_2)^\top$, for what $\vec{x}$ is the model $\hat{y} = \sigma(\mathbf{W}\vec{x} + b)$ most uncertain as to which class $\vec{x}$ belongs to?

Answer: The model will be most uncertain when $\hat{y} = 0.5$. Let's plug this in and solve for the corresponding $\vec{x}$ in terms of $\mathbf{W}$ and b.

$$\begin{aligned}
\hat{y} &= \sigma(\mathbf{W}\vec{x} + b) \\
0.5 &= \frac{1}{1 + e^{-(\mathbf{W}\vec{x}+b)}} \\
0.5\left(1 + e^{-(\mathbf{W}\vec{x}+b)}\right) &= 1 \\
e^{-(\mathbf{W}\vec{x}+b)} &= 1 \\
\mathbf{W}\vec{x} &= -b \\
\Rightarrow w_1 x_1 + w_2 x_2 &= -b,
\end{aligned}$$

which gives

$$\boxed{x_2 = \frac{-b - w_1 x_1}{w_2}}. \tag{3}$$

(c) (2p) Fit the logistic model to the `measuredData.mat` dataset by minimizing the CE loss, taking the scattering angle and measured energy as input variables $x = (\theta, V)$. Use the Adam optimizer with learning rate 10^{-2} and run 10 000 epochs. Using the

formula found in 2b) plot the found decision boundary on top of the data, and report the found coefficients W, b.

If you did not find a formula in 2b) for the boundary you can use the approximate formula $x_2 = -115 + 810x_1$.
(Hint: In PyTorch you can use the Adam optimizer to minimize the loss function via `torch.optim.Adam`)

Answer: Using Equation 3 I fit the logistic model to find the decision boundary shown in Figure 1. I obtained weight values of $w_1 = 61.2942$ and $w_2 = -0.0796$, and a bias of $b = -7.5269$.

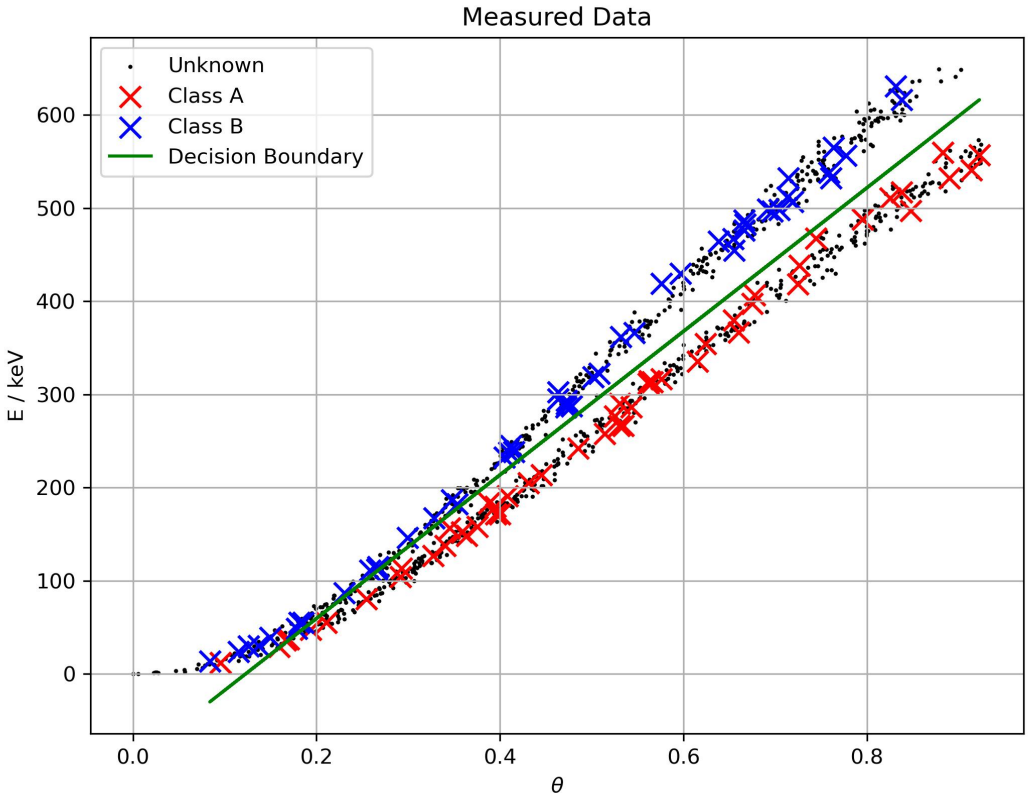

Figure 1: The decision boundary from the logistic model plotted overtop of the gamma photon scattering data.

2. Neural Network Model

The logistic model can be interpreted as a simple neural network that first runs the input x through an affine function $\vec{x} \to \mathbf{W}\vec{x} + b$ and then spits out a single sigmoid activation $\hat{y} = \sigma(\mathbf{W}\vec{x}+b) \in [0, 1]$. We can expand this into an arbitrary feed-forward neural network by adding more (hidden) layers with more nodes, and play with the choice of activation

functions[1] to get the model

$$y = \sigma_l(\mathbf{W}_l \cdots \sigma_2(\mathbf{W}_2\sigma_1(\mathbf{W}_1\vec{x} + b_1) + b_2) \cdots + b_l)$$

where σ is a choice of activation function properly chosen for the case at hand.

(a) (2p) Instead of the logistics model above, fit a neural network with two hidden layers of dimension 32 followed by 16, with tanh activations, and a single output node with sigmoid activation, using the CE loss and the Adam optimizer. Train it for 7 500 epochs with learning rate 10^{-3}. Make sure you get a reasonable fit, if not, run the training again. Plot the decision boundary on top of the data and comment on its shape in comparison to the one found in 1(c). Are there any differences?

Answer: I then fit the data using a neural networks with the parameters given above. The decision boundary from the neural network is shown in Figure 2.

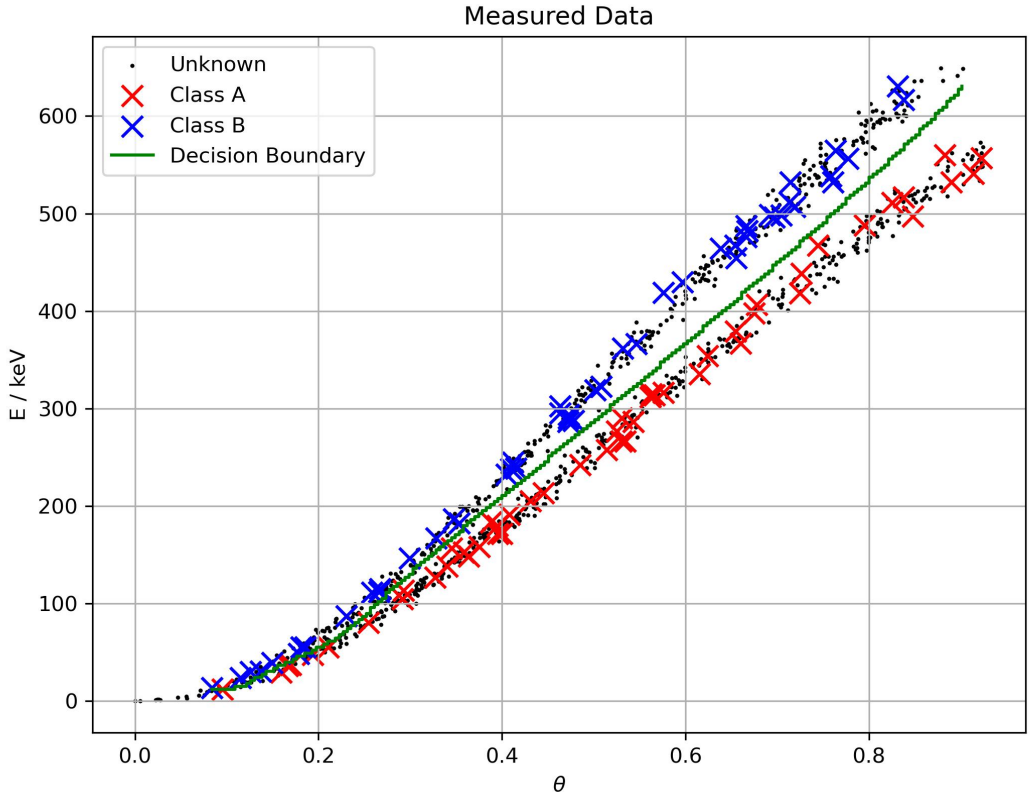

Figure 2: The decision boundary from the neural network plotted overtop of the gamma photon scattering data, for two hidden layers with 32, and 16 nodes.

The shape of this decision boundary is slightly different than in part 1 (a) as it is not entirely linear and seems to flatten out towards higher values of θ.

(b) (1p) Experiment a bit with the architecture of your neural network by adjusting the number of hidden layers and the number of nodes per layer. Try to achieve the

[1] Activation functions are applied component-wise for vector outputs

highest possible training accuracy while keeping the network as simple as possible. Is this a good classifier, how would it perform on unseen data?

Answer: I modified the architecture of my neural network and fit the data using three different models. First, I added a third hidden layer with 8 nodes. The decision boundary is shown in Figure 3, and in this plot we can see that there is some overfitting occuring, especially near the bottom of the dataset. This could hurt the effectiveness of the model on unseen data.

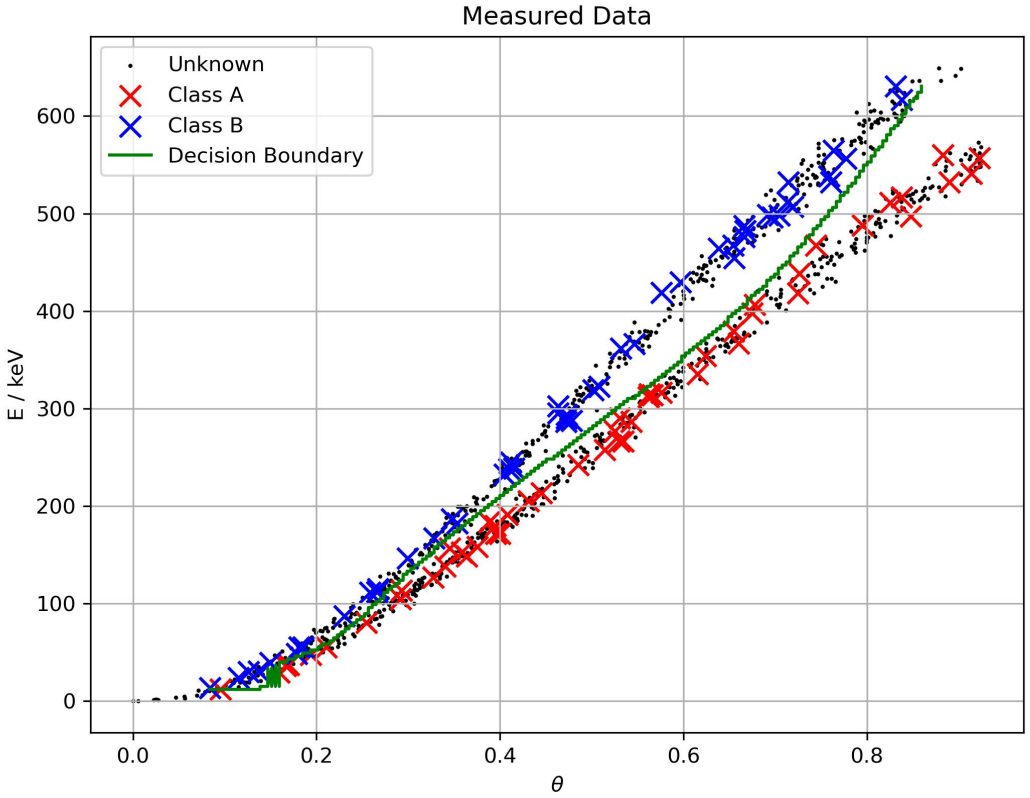

Figure 3: The decision boundary from the neural network plotted overtop of the gamma photon scattering data, for three hidden layers with 32, 16, and 8 nodes.

Next, I constructed the neural network with only 1 hidden layer that contained 32 nodes. The results are shown in Figure 4. There appears to be some underfitting occuring in this plot which could also hurt its effectiveness on unseen data.

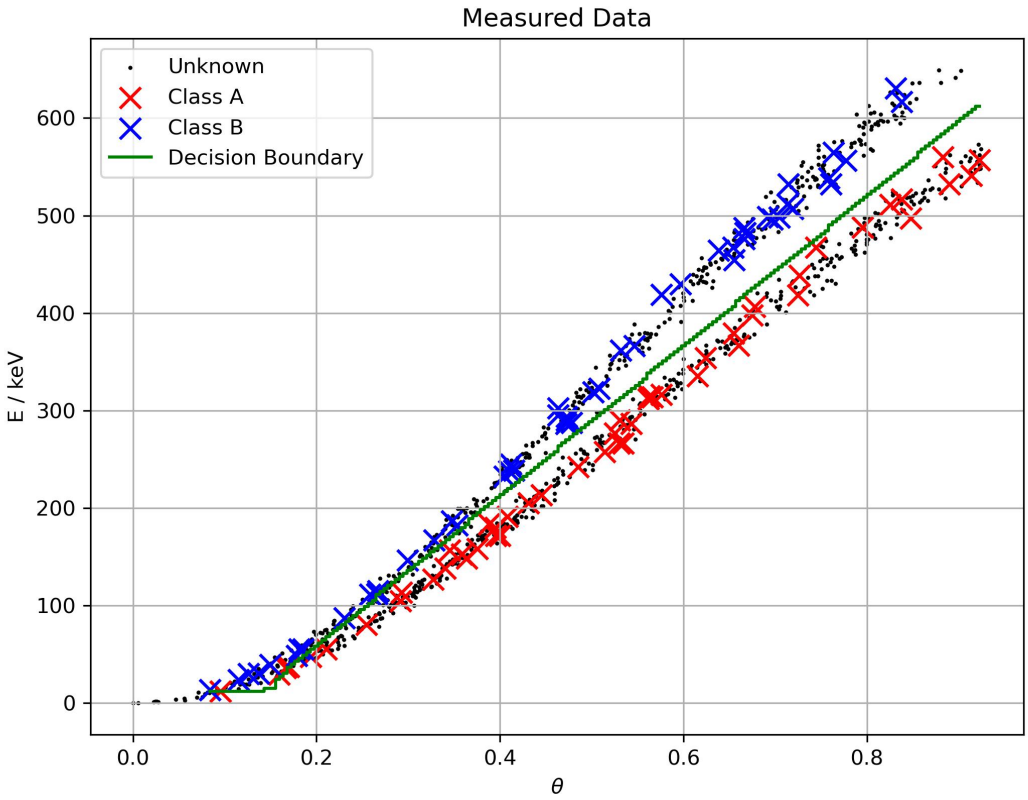

Figure 4: The decision boundary from the neural network plotted overtop of the gamma photon scattering data, for one hidden layer with 32 nodes.

Finally, I constructed the neural network with 1 hidden layer that contained 64 nodes. The results are shown in Figure 5 and the fit is quite smooth, similar to that from the original network with two hidden layers (Figure 2). This maintains a high classicfication accuracy with a simple network (1 hidden layer) and I'd expect it to perform well on unseen data.

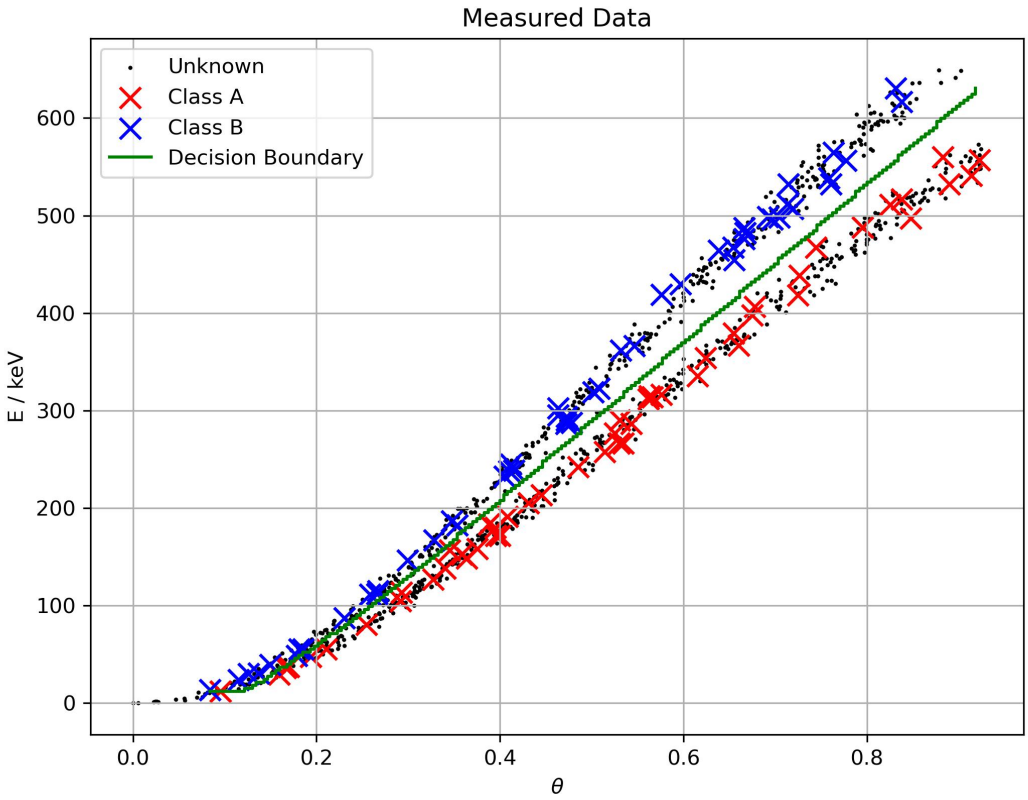

Figure 5: The decision boundary from the neural network plotted overtop of the gamma photon scattering data, for one hidden layer with 64 nodes.

3. Regression for Energy Estimation

Using your classification of the data (either classifying using the model from problem 1 or problem 2) we shall now fit one regression model to each class respectively in order to infer the incident energies E_A, E_B.

(a) (1p) Recall the Compton formula

$$E' = \frac{E_0}{1 + \frac{E_0}{m_e c^2}(1 - \cos\theta)}$$

that describes the energy E' of a photon after having Compton interacted with angle θ, where E_0 was the initial energy. Let $t = 1 - \cos\theta$ and Taylor expand the measured energy $E(t) = E_0 - E'(t)$ in terms of t.

Answer: Letting $t = 1 - cos\theta$ we want to Taylor expand the energy

$$E(t) = E_0 - E'(t) = E_0 \left(1 - \frac{1}{1 + \frac{E_0}{m_e c^2}(t)}\right). \tag{4}$$

Letting $a = \frac{E_0}{m_e c^2}$ we can rewrite Equation 4 as

$$E(t) = E_0 - E'(t) = E_0 \left(1 - \frac{1}{1+at}\right). \tag{5}$$

We know the Taylor expansion of the function $\frac{1}{1+x}$ is given by

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + \dots, \tag{6}$$

and we can now plug Equation 6 into Equation 5 to yield the Taylor expansion of Equation 5 as

$$\begin{aligned} E(t) &= E_0 \left(1 - \frac{1}{1+at}\right) \\ &= E_0 \left(1 - (1 - at + a^2t^2 - a^3t^3 + \dots)\right) \\ &\approx E_0 \left(at - a^2t^2 + a^3t^3)\right) \\ \therefore E(t) &= \frac{E_0^2}{m_e c^2} t - \frac{E_0^3}{(m_e c^2)^2} t^2 + \frac{E_0^4}{(m_e c^2)^3} t^3, \end{aligned} \tag{7}$$

where Equation 7 gives the Taylor expansion up to the qubic term.

(b) (3p) Fit each of the two classes of data to the regression model

$$y = \mathbf{W}\vec{x} + b$$

where now $\vec{x} = (t, t^2, t^3, \dots)^\top$, $y = E$, $b = 0$, using mean square error loss (`nn.MSELoss()`), with learning rate 10^3 for 25 000 epochs. Keep at least the cubic terms. Plot both fitted curves in the same plot together with the data.

Answer: I used the neural network in part 2. (a) to classify the data points into either class A or class B and fit each of the two classes using the mean square error loss function and the parameters given above. Both the fitted curves and the classified data are plotted together in Figure 6.

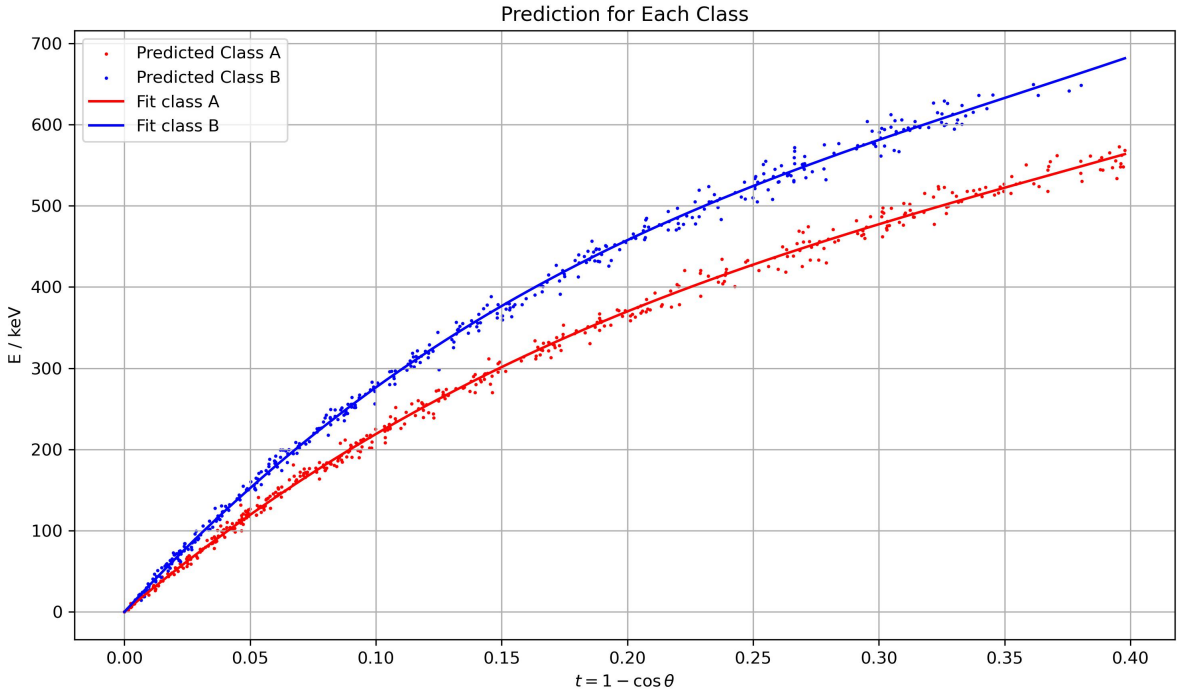

Figure 6: The classified data with fitted curves according to the mean square error loss function.

(c) (1p) Derive a relationship $E_0 = E_0(W_i)$ between the found Taylor coefficients W_i and the initial energy E_0.

Answer: We know that we used the model $y = \mathbf{W}\vec{x} + b$ where $\vec{x} = (t, t^2, t^3, \ldots)^\top$, $y = E$, $b = 0$. We can relate this to Equation 7 to obtain

$$\begin{aligned} \mathbf{W}\vec{x} + b &= \frac{E_0^2}{m_e c^2} t - \frac{E_0^3}{(m_e c^2)^2} t^2 + \frac{E_0^4}{(m_e c^2)^3} t^3 \\ W_i (t, t^2, t^3, \ldots)^\top &= \frac{E_0^2}{m_e c^2} t - \frac{E_0^3}{(m_e c^2)^2} t^2 + \frac{E_0^4}{(m_e c^2)^3} t^3 \\ \therefore W_i &= \frac{E_0^{i+1}}{(m_e c^2)^i}. \end{aligned} \tag{8}$$

(d) (2p) Use the linear term W_0 to deduce and report your estimates for $E_{A,0}$ and $E_{B,0}$. Why does the linear term give a better estimate than the higher order terms here?

Answer: Using the linear term W_0 I extracted energy estimates of $E_{A,0} = 1155.18$ keV and $E_{B,0} = 1311.04$ keV for classes A and B, respectively. Here the linear term gives a better estimate because the higher-order terms are more sensitive to noise.

(e) (3p) Fitting a regression model with mean square error loss implicitly means that we assume the errors (noise) to be identically and independently distributed (iid) normal random variables. In this case, we have assumed the noise to be approximately iid normal, but looking closely at the data we can see that this is not the

case. Describe a way to handle the varying noise to get a more robust fit, and investigate how the result changes when you apply this method.

Answer: In order to handle the varying noise I used the Huber loss function (`torch.nn.HuberLoss()`) from PyTorch. The Huber loss function combines the mean square error (MSE) and mean absolute error (MAE) loss functions. The Huber loss function uses the MSE loss function for small residuals which are less than a user specified parameter, δ, and switches to use the MAE loss function for residuals greater than this δ. The MSE is more sensitive to larger errors so switching to MAE reduces the influence of exceptionally large noise to the overall loss function.

To set the parameter δ for the Huber loss function I calculated the residuals of the data from the MSE loss function, calculating a standard deviation of $\sigma_A \approx 8.6$ and $\sigma_B \approx 9.0$ for classes A and B, respectively. Using the Huber loss function I obtained energy estimates of $E_{A,0} = 1164.42$ keV and $E_{B,0} = 1320.23$ keV. The new fit and the data is shown in Figure 7.

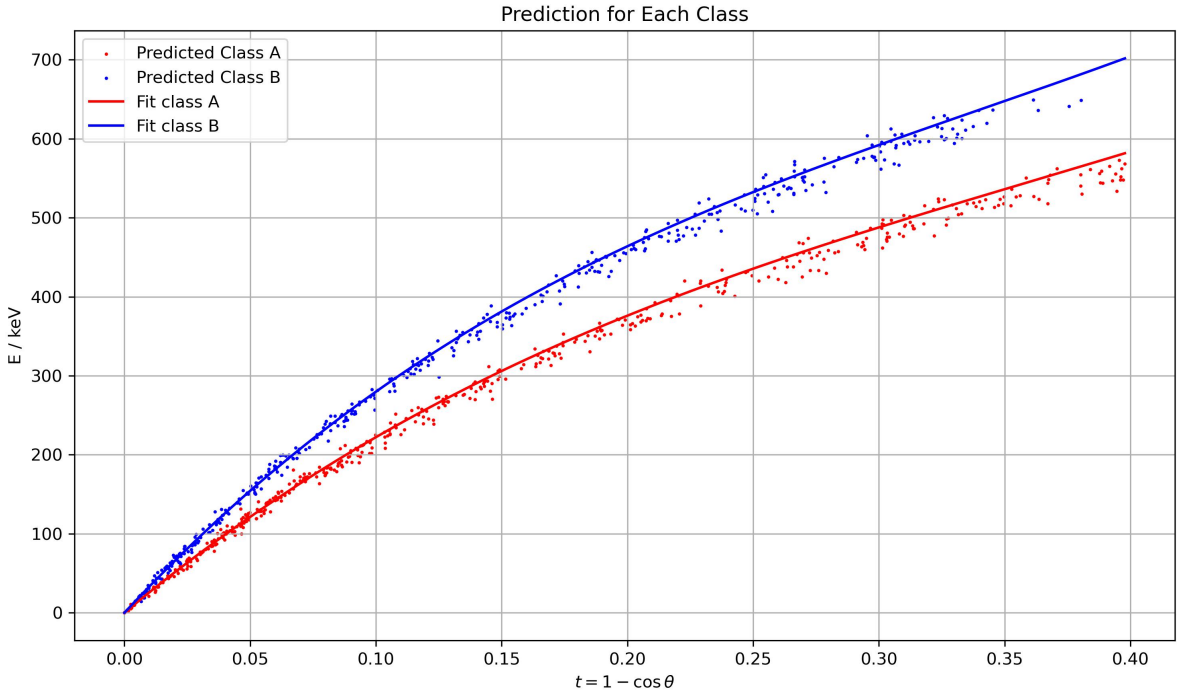

Figure 7: The classified data with fitted curves according to the Huber loss function.

4. Nuclide Identification (1p)

Find a reasonable candidate for the unknown nuclide using your estimated energies. For this question only, you are allowed to ask an AI chatbot for help!

Answer: According to ChatGPT the nuclide is likely Cobalt-60 (Co-60). Co-60 has successive gamma emissions of approximately 1173 keV and 1332 keV. These emissions are within 21 keV of my energies calculated in Part 3 (d) (1155 keV and 1311 keV), and within 12 keV of my energies calculated in Part 3 (e) (1164 keV and 1320 keV).

Final question: Did you use an AI tool (other than the machine learning models you trained in this exercise) for anything else than information searching, when solving this problem set (including problem 4)? If so, please write a brief statement of how you used AI.

Answer: I used an ChatGPT to help me understand why the linear W_0 term gives a better estimate than the higher order terms in Part 3 (d), and I used ChatGPT to help me determine an approach to handle the varying noise in Part 3 (e). I also used ChatGPT to help me calculate the residuals of the data from the MSE fit and to understand why I should use the standard deviation of the residuals as the δ value in the Huber loss function. I also use ChatGPT to help me check my work with the Taylor expansion, which gave me the idea to use the known taylor expansion of $\frac{1}{1+x}$ instead of writing out all the derivatives.

Total number of points: 20

Remember to motivate your answers wherever applicable.

Good luck!