# Homework 6

Please use PyTorch to complete the problems in Homework 6.

## Problem 1

The *RMS Titanic*, a British ocean liner carrying 1,309 passengers, sank on its maiden voyage from England to the United States. Various versions of the ship's passenger list have been created–we will use TitanicSurvival.csv, which includes data on each passenger's name, sex, age, passenger class (`"1st"`, `"2nd"`, `"3rd"`), and survival status. You can load in the data from this URL: https://vincentarelbundock.github.io/Rdatasets/csv/carData/TitanicSurvival.csv. Ultimately, we would like to predict the survival status of each passenger based on their demographic characteristics.

a. Import the dataset, and do whatever data cleaning/wrangling you deem necessary. At minimum, you should center the columns of the predictors so that there is no need for an intercept/bias–we do not recommend centering the outcome, as that will cause problems when training the model.

b. Write your own function that will complete a forward pass for the entire dataset. Generate three random weights, and run a single forward pass resulting in the probability that a 29 year-old female passenger in first class survives. Do not forget that you will need to standardize these inputs (or look up a passenger with that information in the tensor with the pre-processed data).

   *Hint: your function will involve the logistic function. You may be tempted to use functions like* `np.exp()` *or* `sum()`*, but since you are working with PyTorch, you will need to use the PyTorch versions of those functions,* `torch.exp()` *and* `torch.sum()`.

c. Write your own function defining the cross entropy loss function for an entire dataset. Given that the passenger from the previous example survived, what is the value of the cross entropy?

   *Hint: similar to the previous problem, you may be tempted to use functions like* `np.log()`*, but since you are working with PyTorch, you will need to use the PyTorch version,* `torch.log()`.

c. Carry out a backward pass, i.e., calculate the gradient of the loss function.

d. Using PyTorch and your functions from previous questions, train the model on the whole dataset. Use a learning rate of $1 \times 10^{-5}$ and train for 1000 epochs. Keep track of the gradients and loss function values in every epoch. Then, create a plots of the loss function and gradient versus epoch.

e. Repeat part d. for four different learning rates: $1 \times 10^{-2}$, $1 \times 10^{-3}$, $1 \times 10^{-4}$, and $1 \times 10^{-6}$. Make sure to generate new sets of random weights each time you start using a new value for the learning rate. Using your results from part d., create new plots of the loss function for the different learning rates (your plot should have five lines, one for each learning rate). What trends in the relationship do you see?

f. Compare the confusion matrices for the learning rates $1 \times 10^{-2}$, $1 \times 10^{-4}$, and $1 \times 10^{-6}$. What trends in the confusion matrices do you see?

## Problem 2

From HW5, we know that a Kaggle user has created a dataset with over 20,000 recipes from the website Epicurious. Loosely speaking, there are a few groups of variables in the dataset:

- The outcome of interest (`cake`, whether a recipe is tagged as a cake)
- The nutritional variables (`calories`, `protein`, `fat`, `sodium`)
- Ingredient tags (`almond`, `amaretto`, `anchovy`, and so on)
- Place tags (`alabama`, `alaska`, `aspen`, `australia`, and so forth)

- Other tags (`advance.prep.required`, `anthony.bourdain`, etc.)

We will be using the same dataset and creating an AutoEncoder to reduce the number of features. You may want to use the same pre-processed data you used in Question 5. As always, you will have to make choices during pre-processing–these choices are up to you.

a. Load the datasets into your environment, and standardize the predictors.

   *Note that you will also need to convert your DataFrame to a tensor!*

b. Using the functions from PyTorch, create an Autoencoder class that will reduce the number of features in this dataset to 10. You can use whatever number of hidden layers that you like.

c. Create instances of: the model using your AutoEncoder from part b., the mean squared error loss function, and your choice of optimizer (one of the options we have covered in class).

d. Train the model using reasonable choices for the number of epochs, batch size, and learning rates. You may have already defined these values for your work in part c.

   *Hint: This did take some time on my laptop (roughly 20 minutes), so work with a random sample of your data until you feel like you have chosen good hyperparameters, or change the* `dtype` *to .to(dtype = torch.float16).*

e. Create plots of Feature 1 against Feature 2, Feature 3 against Feature 4, Feature 5 against Feature 6, Feature 7 against Feature 8, and Feature 9 against Feature 10. Incorporate some of the cluster labels from Homework 5 (by color, or perhaps by including some of the recipe names on the plot). Are you seeing any of the same kinds of clusters you saw in Homework 5?