Homework 7

1. Embeddings and un-embeddings. In `beer_class.json`, you are given a 1000-vector `v_true`, consisting of categorical raw outputs corresponding to beer types, and a 1000-vector `y_hat` consisting of predictions from some classifier, which have been embedded. Note that the classifier does not necessarily have zero error rate.

   (a) Inspect `v_true`. Assume that all classes are represented in `v_true`. What is \( V \), the label set? What is \( K \), the number of classes/labels? (Order does not matter.)

   (b) Inspect `y_hat`. Propose an embedding \( \psi : V \rightarrow \mathbb{R} \) for the raw outputs that is consistent with both `y_hat` and `v_true`. Furthermore, propose an un-embedding \( \psi^\dagger : \mathbb{R} \rightarrow V \) for the predictions that is consistent with both `y_hat` and `v_true`.

   (c) Using your un-embedding from part (b), un-embed the classifier predictions `y_hat` into a new 1000-vector `v_hat`. Report the confusion matrix. What is the overall error rate of the classifier, assuming that your embedding and un-embedding were actually used?

2. Fitting a multi-class classifier. In `multi_class.json`, you will find a 300 \( \times \) 30 matrix `U_train` and a 300-vector `v_train` consisting of raw training input and output data, and a 300 \( \times \) 30 matrix `U_test` and a 300-vector `v_test` consisting of raw test input and output data, respectively. We will work with \( x = \phi(u) = (1, u) \).

   In `multi_logistic.jl` we have also provided you with a function

   \[
   \text{multi_logistic}(X, Y, \text{reps}).
   \]

   This function takes in input/output data `X` and `Y`, and an array of representations of our embeddings `reps`. That is, the \( k \)th element of `reps` is \( \psi(v_k) \), for \( k = 1, \ldots, K \), and \( K \) is the number of labels.

   The function outputs the parameterized predictor `g` for the multi-class classifier. (Since `g` is a predictor, its argument is simply `x`; to get a prediction `yhat` from a data input `x`, use `yhat = g(x)`.) You must include the Flux and LinearAlgebra Julia packages in your code in order to utilize this function.

   (a) Inspect `v_train` and `v_test`. What is \( V \), the label set? What is \( K \), the number of labels? Report the number of instances of each label in the training set. Do the same for the test set.

   (b) Propose a simple embedding \( \psi : \mathbb{R} \rightarrow \mathbb{R}^K \) for your raw output data `v`, so that \( y = \psi(v) \).

   (c) Apply your embedding from part (b) to both the training and test raw output data. Using `multi_logistic.jl`, fit a multi-class classifier to the training data. Use a nearest neighbor un-embedding to un-embed your data. Report the confusion matrices of your classifier on the training and test set. Also report the overall training error rate and the overall test error rate.
Julia hints. `unique(z)` returns the number of unique elements in an array `z`. For the nearest neighbor unembedding $\psi^\dagger(\hat{y})$, you may use

$$\text{psiinv}(\text{yhat}) = \text{reps}[\text{argmin}([\text{norm}(\text{yhat}-\text{reps}[k]) \text{ for } k=1:\text{length}(\text{reps})])]$$

where `reps` is an array whose $k$th element is $\psi(v_k)$, for $k = 1, \ldots, K$. 