Homework 8

1. Classification of political speeches. You are given the text of 3119 speeches made by presidential candidates between 1996 and 2016. (The collection of speeches is called the corpus.) Each speech was either made by a Democrat (D) or a Republican (R). You will learn a classifier that can identify a candidate’s political affiliation based on their speech.

The file speeches.json contains U, a list of all the speeches; V, the list of the corresponding party affiliations; words, a list of every word used at least once in one of the speeches; and freqs, a list of how many times each word was used in the speeches.

The ith speech in the corpus is \( u^i \), and the corresponding affiliation is \( v^i \in \{D, R\} \). You will need to utilize the LinearAlgebra, Statistics, DataStructures, and Flux packages.

We will be utilizing a particular feature transformation, the term-frequency inverse document frequency (TFIDF) embedding, which is a metric of how important a word is in a corpus. (We will be using this feature transformation along with other standard feature transformations, i.e., appending a constant term and standardization.) We define the term-frequency embedding as

\[
T^{TF}(u)_j = \frac{\text{number of occurrences of word } j \text{ in } u}{\text{the number of words in } u}
\]

This maps a speech \( u \) to a vector in \( \mathbb{R}^{n_{\text{words}}} \), where \( n_{\text{words}} \) is the number of words in the corpus, which we take as the top 2000 most frequent words (so \( n_{\text{words}} = 2000 \)).

From this, we construct the TFIDF embedding. The document frequency of word \( j \) is

\[
f_{\text{doc}}(j) = \frac{\text{the number of documents in which the word occurs}}{n}
\]

and the TFIDF embedding is

\[
T^{TFIDF}(u)_j = T^{TF}(u)_j \log(1 + 1/f_{\text{doc}}(j))
\]

The file utils.jl contains utility functions you may find useful. In particular, the functions TFIDF(U, words, freqs) and standardizemultiasone(U, means, stds) will help with the input embeddings. multi-logistic(X, Y, reps; lambda) will solve the multi-class classification problem; the inputs are training input and output data \( X \) and \( Y \), a list of the representations of the embedded outputs \( \text{reps} \), and an optional regularization hyper-parameter term \( \text{lambda} \), which defaults to 1. The function outputs the parameterized predictor predicty for the multi-class classifier, and the model parameters theta. (Since predicty is a predictor, its argument is simply \( x \); to get an (embedded) prediction \( \hat{y} \) from a data input \( x \), use \( \hat{y} = \text{predicty}(x) \).)
Randomly partition 80% of the data into a training set, and 20% into a test set. Using the embedding

\[ x = \phi(u) = (1, \text{standardize}(T^{\text{TFIDF}}(u))) , \]

i.e., by standardizing the TFIDF-embedded vector and appending a constant feature to it (so \( d = n_{\text{words}} + 1 = 2001 \)).

The embedded output \( y = \psi(v) \) embeds the party affiliations; we will use \( \psi : \{D, R\} \rightarrow \mathbb{R} \) such that \( \psi(D) = 0 \) and \( \psi(R) = 1 \). Embed the raw training and test outputs in this way.

(a) Inspect the multi_logistic function in utils.jl. What is the loss function used? What is the regularization function used (if any)? A simple response, using no math, is sufficient (e.g., “SVM loss with \( \ell_1 \)-regularization”).

(b) For the regularization hyper-parameter \( \lambda \in \{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\} \), train a classifier on the training set, using the multi_logistic function in utils.jl. Report the test accuracy, and highlight the value of \( \lambda \) that gave you the best test accuracy. In addition, report the \( 2 \times 2 \) confusion matrix on the test data.

(c) Using the model that gave you the best test accuracy, analyze the weights and find the five most useful words for classification.

*Hint.* For part (c), use the absolute value of the weights to determine the importance of a word to the classifier.

*Julia hints.*

- For accuracy, you may use the following function in Julia:

\[
\text{accuracy}(a, b) = \text{Statistics.mean}(a .== b),
\]

where \( a \) and \( b \) are arrays.

- The function \( \text{TFIDF}(U, \text{words}, \text{freqs}) \) has two outputs: the TFIDF-transformed feature vector, and a dictionary corresponding to the words used in the corpus. The second output will be useful for part (c).

- For part (b), training may take several minutes.

2. **Multi-class animal classification.** Our task is to create a predictor which identifies the class type of the animal based on 16 traits: **H**air, **F**eathers, **E**ggs, **M**ilk, **A**irborne, **A**quatic, **P**redator, **T**oothed, **B**ackbone, **B**reathes, **V**enomous, **F**ins, **L**egs, **T**ail, **D**omestic, **C**atsize. All traits except **L**egs (Numeric) are Boolean.

In animals.json, you will find an \( 101 \times 16 \) matrix \( U \) of data, with rows \((u_i)^T\), and 101-vector \( v \) with the class type of the animal. The unique animal class types are **M**ammal, **B**ird, **R**eptile, **F**ish, **A**mphibian, **B**ug, **I**nvertebrate.

Randomly partition the data into a training set consisting of 50% of the data, and a test set consisting of the remaining 50% of data.
(a) Propose a feature embedding function $\phi$ with at least three unique feature engineering transformations to apply to your input data, so that $y = \phi(u)$. The feature transformations need not be complicated. Apply your feature embedding to raw input data.

(b) Propose a simple embedding $\psi$ for your raw output data $v$, so that $y = \psi(v)$. In addition, propose a simple un-embedding $\psi^\dagger$, such that $v = \psi^\dagger(y)$. Apply the embedding to your raw output data.

(c) Using multi_logistic from utils.jl, train a multi-class logistic classifier on your training set and evaluate on the test set. Un-embed your output data using your un-embedding found in part (b). Report the overall accuracy for the test set, as well as the confusion matrix.