Homework 1

Deadline: Friday, Jan. 25, at 11:59pm.

Submission: You need to submit two files through MarkUs `https://markus.teach.cs.toronto.edu/csc411-2019-01`:

- Your answers to Questions 1, and outputs requested for Question 2, as a PDF file titled `hw1_writeup.pdf`. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.
- Your code for Question 2, as the Python file `hw1_code.py`. This should contain the functions `load_data`, `select_model`, and `compute_information_gain`.

Neatness Point: One of the 10 points will be given for neatness. You will receive this point as long as we don't have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Weekly homeworks are individual work. See the Course Information handout `http://www.cs.toronto.edu/~mren/teach/csc411_19s/syllabus.pdf` for detailed policies.

1. **[4pts] Nearest Neighbours and the Curse of Dimensionality.** In this question, you will verify the claim from lecture that "most" points in a high-dimensional space are far away from each other, and also approximately the same distance.

   (a) **[2pts]** First, consider two independent univariate random variables X and Y sampled uniformly from the unit interval $[0, 1]$. Determine the expectation and variance of the random variable Z, defined as the squared distance $Z = (X - Y)^2$.

   (b) **[1pts]** Now suppose we sample two points independently from a unit cube in d dimensions. Observe that each coordinate is sampled independently from $[0, 1]$, i.e. we can view this as sampling random variables $X_1, \ldots, X_d, Y_1, \ldots, Y_d$ independently from $[0, 1]$. The squared Euclidean distance can be written as $R = Z_1 + \cdots + Z_d$, where $Z_i = (X_i - Y_i)^2$. Using the properties of expectation and variance, determine $\mathbb{E}[R]$ and $\mathrm{Var}[R]$. You may give your answer in terms of the dimension d, and $\mathbb{E}[Z]$ and $\mathrm{Var}[Z]$ (the answers from part (a)).

   (c) **[1pt]** Based on your answer to part (b), compare the mean and standard deviation of R to the maximum possible squared Euclidean distance (i.e. the distance between opposite corners of the cube). Why does this support the claim that in high dimensions, "most points are far away, and approximately the same distance"?

2. **[5pts] Decision Trees.** *This question is taken from a project by Lisa Zhang and Michael Guerzhoy.*

   In this question, you will use the `scikit-learn` decision tree classifier to classify real vs. fake news headlines. The aim of this question is for you to read the `scikit-learn` API and get comfortable with training/validation splits.

   We will use a dataset of 1298 "fake news" headlines (which mostly include headlines of articles classified as biased, etc.) and 1968 "real" news headlines, where the "fake news" headlines are from `https://www.kaggle.com/mrisdal/fake-news/data` and "real news" headlines are from `https://www.kaggle.com/therohk/million-headlines`.

   Each headline appears as a single line in the data file. Words in the headline are separated by spaces, so just use `str.split()` in Python to split the headlines into words.

   You will build a decision tree to classify real vs. fake news headlines. Instead of coding the decision trees yourself, you will do what we normally do in practice — use an existing implementation. You should use the `DecisionTreeClassifier` included in `sklearn`. Note that figuring out how to use this implementation is a part of the assignment.

   All code should be included in the file `hw1_code.py` which you submit through MarkUs.

   (a) **[1pt]** Write a function `load_data` which loads the data, preprocesses it using a vectorizer (`http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text`), and splits the entire dataset randomly into 70% training, 15% validation, and 15% test examples.

   (b) **[1pt]** Write a function `select_model` which trains the decision tree classifier using at least 5 different values of `max_depth`, as well as two different split criteria (information gain and Gini coefficient), evaluates the performance of each one on the validation set, and prints the resulting accuracies of each model. You should use `DecisionTreeClassifier`, but you should write the validation code yourself. Include the output of this function in your solution PDF (`hw1_writeup.pdf`).

   (c) **[1pt]** Now let's stick with the hyperparameters which achieved the highest validation accuracy. Extract and visualize the first two layers of the tree. Your visualization does not have to be an image: it is perfectly fine to display text. It may also be hand-drawn. Include your visualization in your solution PDF (`hw1_writeup.pdf`).

   (d) **[2pts]** Write a function `compute_information_gain` which computes the information gain of a split on the training data. That is, compute $I(Y, x_i)$, where Y is the random variable signifying whether the headline is real or fake, and x_i is the keyword chosen for the split.

   Report the outputs of this function for the topmost split from the previous part, and for several other keywords.