AMCS 215
Mathematical Foundations of Machine Learning
Assignment 7, Due October 24, 2023

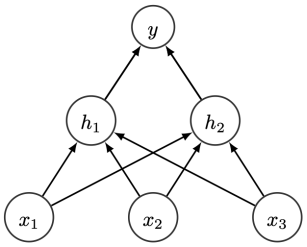
Reading material:

- chapter 5 from `https://d2l.ai` and/or section 13.2-13.3 from Reference 1
- section 15.7 from `https://d2l.ai`

Notes:

- Exam is scheduled on Wed Nov 1. Please bring your laptops to the exam.

1. Backpropagation.
The following graph shows the structure of a simple neural network with a single hidden layer. The input layer consists of three features $x = (x_1, x_2, x_3)$. The hidden layer includes two units $h = (h_1, h_2)$. The output layer includes one unit $\hat{y}$. We ignore bias terms for simplicity.



We use ReLU units $\varphi(z) = \max(0, z)$ as activation functions for the hidden and the output layer.
Let $J = \ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$ be the loss function. Here y is the target value for the output unit.
Denote by W and V weight matrices connecting input and hidden layer, and hidden layer and output respectively. They are initialized as follows:

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \text{ and } V = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

- write out symbolically (in terms of φ, W, and V) the function $x \to \hat{y}$ represented by this network.
- consider a data sample with features $x = (1, 2, 1)$ and target $y = 1$. Compute the output value $\hat{y}$.
- compute the gradient of the loss function with respect to the weights. Specifically,
  - write down symbolically the gradient with respect to V, $\frac{\partial J}{\partial V}$
  - write down symbolically the gradient with respect to W, $\frac{\partial J}{\partial W}$
  - compute the values of this gradient numerically for the parameters above

2. House price prediction.
You will find on Blackboard a starter file for training a model to predict house prices from given house features. There are four features used in this regression: one binary feature (whether or not the house has covered parking), one numerical feature (size, measure in thousands of square meters), and two categorical features (architectural style and location). Each of the categorial features is represented as a one-hot vector of size 3. This gives us feature vectors of size 8 in total for each house. 200 examples are given in the training set, and 40 in the test set.

- build a least squares predictor of house prices with the given features. Evaluate the prediction accuracy of the model obtained.
- build a neural network to predict prices from the given features. Use a network with two hidden layers of 10 units each with ReLU activation. The output unit has no activation. Using mean square loss, train the network with an ADAM optimizer for a number of epochs of your choice (using a batch size of, say, 32) and plot the loss versus the number of training epochs. Evaluate the prediction accuracy of the model obtained. Comment.

3. Analogies with word vectors.
Words can be represented by dense vectors that somewhat capture their meaning. These vectors are called word embeddings or word vectors and are often the starting point for natural language processing tasks. In this exercise, we will use pre-trained word embeddings[1] to represent words. A starter file is posted on Blackboard.

- we will use the cosine similarity measure to compute the similarity between two word vectors u and v. Write an expression for this similarity measure.
- write a function `most_similar` that returns the 10 most similar words to a given word. Test it on an example such as `most_similar('bread')`
- write a function `doesnt_match` that returns the word that doesn't match in a list of words. Test it on an example such as `doesnt_match(['breakfast', 'cereal', 'dinner', 'lunch')`.
- write a function `analogy` that returns the best word that completes an analogy. test it on an example such as `analogy('man', 'king', 'woman')`

In order to enhance the performance on these word analogy tasks, we can perhaps use "better" word vectors. "Better" word vectors are obtained by using a larger and richer text corpus for pre-training or by using an embedding space with a larger dimension to capture more aspects of word meaning. You may want to explore the results produced using one of the larger embeddings listed in the starter file.

4. Sentiment analysis with pre-trained word vectors
Sentiment analysis is a classification task that takes as input a natural language text fragment and labels it with a corresponding feeling it might be associate with. In this exercise, we will consider a binary sentiment analysis task that classifies movie reviews into positive or negative ones using pre-trained word embeddings. We will use a training dataset of 25,000 movies reviews from IMDb (the Internet Movie Database), which have been labeled by positive/negative sentiment. Reviews have been preprocessed, and each review is encoded as a list of word indices stored as integers. You will find a starter file on Blackboard that loads this data, and represents each review as a vector of $T = 256$ integer indices.

Your task is to build a neural network model with one hidden layer of size 32, and train it to minimize a binary cross entropy loss.

- generate the training data for your model by averaging the word embeddings of every review (we may think of this average as the "review embedding"). This is the input to the model: a vector of size D, the dimension of the embedding, or a batch of such vectors.
- build the model and train it to minimize a binary cross entropy loss. Use a `DataLoader` to serve the data to the training loop in batches of size 512. Use Adam with a learning rate of 10^{-3}. Plot loss vs epoch. Evaluate the quality of the model by generating a confusion matrix.
- in the starter file, you will find another way to define this same model, where an embedding layer is introduced as the first layer of the network. The parameters (embedding matrix) of this layers are frozen at their pre-trained values. The forward pass computes the average of the words embeddings of a review. Read the code and answer the following questions: (1) what is the total number of parameters in the model? and (2) what is the number of trainable parameters?

5. Sentiment analysis with trainable embeddings
Instead of using fixed (pre-trained) word embeddings for the sentiment classification task as we did in the previous exercise, we will now make the word embeddings trainable as well. We will use an embedding of dimension $D = 50$.

- the classification network for this problem consists, as above, of an embedding layer, an averaging operation (with no trainable variables), a dense hidden layer of size 32, and a final dense output layer. Write down the mathematical description of the operations involved in evaluating the function defined by this network.
- train and test the model starting from a random initialization of all parameters.
- train and test the model starting from a pre-trained embedding of dimension 50 (and random initialization of the remaining parameters). This is called *finetuning*. Comment on the efficiency of training in this case, relative to starting from random initial values.

[1] In class, we briefly described how these word vectors can be generated. In a future assignment, we will go over the details of the generation process.