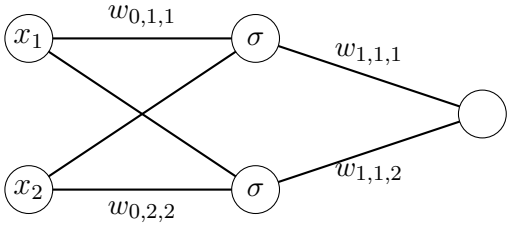
Assigment 3

EECS 4404/5327

The assignment is due on Wednesday, April 3, before the end of the day.
Place your submission in the dropbox Lassonde building, ground floor.

1. **Backpropagation**
Consider a neural network with one hidden layer containing two nodes, input dimension 2 and output dimension 1. That is, the fist layer contains two nodes $v_{0,1}, v_{0,2}$, the hidden layer has two nodes $v_{1,1}, v_{1,2}$, and the output layer one nodes $v_{2,1}$. All nodes between consecutive layers are connected by an edge. The weights between node $v_{t,i}$ and $v_{t+1,j}$ is denoted by $w_{t,j,i}$ as (partially) indicated here: The nodes in the middle layer apply a differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$, which has derivative σ'.



   (a) The network gets as input a 2-dimensional vector $\mathbf{x} = (x_1, x_2)$. Give an expression for the output $\mathrm{N}(\mathbf{x})$ of the network as a function of x_1, x_2 and all the weights.

   (b) Assume we employ the square loss. Give an expression for the loss $\ell(\mathrm{N}(\cdot), (\mathbf{x}, t))$ of the network on an example $(\mathbf{x}, t)$ (again, as a function of x_1, x_2, t and all the weights).

   (c) Consider the above expression of the loss as a function of the set of weights $L(w_{0,1,1}, w_{0,2,1}, w_{0,1,2}, w_{0,2,2}, w_{1,1,1}, w_{1,1,2}) = \ell(\mathrm{N}(\cdot), (\mathbf{x}, t))$. Compute the 6 partial derivatives

$$\frac{\partial L}{\partial w_{1,1,1}}\ ,\ \frac{\partial L}{\partial w_{1,1,2}}\ ,\ \frac{\partial L}{\partial w_{0,1,1}}\ ,\ \frac{\partial L}{\partial w_{0,1,2}}\ ,\ \frac{\partial L}{\partial w_{0,2,1}}\ ,\ \frac{\partial L}{\partial w_{0,2,2}}$$

   (You may use the notation $a_{i,j}$ and $o_{i,j}$ for the input and output of nodes as appropriate.)

(6 + 4 + 10 marks)

2. **k-means clustering**
In this question you will implement and experiment with the k-means algorithm.

(a) Implement the k-means algorithm and include your code in your submission. As always, the data, consisting of n datapoints which are d-dimensional vectors, is given as an $n \times d$-dimensional matrix. The output of your algorithm should be one n-dimensional vector containing values on $\{1, \ldots, k\}$ indicating the cluster assignments of the points. We will work with three ways of initializating the centers. In the first, you will simply set the inital centers by hand. In the second you will choose k datapoints uniformly at random from the dataset as your initial centers. For the third initialization, choose the first center uniformly at random, and then choose each next center to be the datapoint that maximizes the sum of (euclidean) distances from the previous datapoints.

(b) Load the first dataset `twodpoints.txt`. Plot the datapoints. Which number of clusters do you think would be suitable from looking at the points? Show that, for the number k that you chose by visually inspecting the data, the k-means algorithm can converge to to different clusterings depending on the initializations. Show two such instances by plotting the clusterings that k-means produced (use a different color for the points of each cluster) together with the initial centers. In this part, set the initial centers manually.

(c) Can the above phenomenon also happen if the initial centers are chosen uniformly at random? What about the third way of initializing? For each way of initializing, run the algorithm several times and report your findings.

(d) From now on, we will work with the third method for initializing cluster centers. Run the algorithm for $k = 1, \ldots 10$ and plot the k-means cost of the resulting clustering (with respect to the last set of centers, which are the means of the output clusters). What do you observe? How does the cost evolve as a function of the number of clusters? How would you determine a suitable number of clusters from this plot (eg in a situation where the data can not be as obviously visualized).

(e) Repeat the last step (that is, plot the cost as a function of $k = 1, \ldots, 10$) for the next dataset `threedpoints.txt`. What do you think is a suitable number of clusters here? Can you confirm this by a visualization?

(f) Load the UCI "seeds" dataset from the last assignment and repeat the above step. Recall that the task for this dataset is 3-class classification. Remove the last column (which contains the class labels) before the clustering. From how the the k-means cost evolves, what seems like a suitable number of clusters for this dataset? How could you use this for designing a simple 3-class classifier for this dataset? What is the empirical loss of this classifier?

(g) Design a simple (two-dimensional) dataset where the 2-means algorithm with the third initialization method will always fail to find the optimal 2-means clustering. Explain why it will fail on your example or provide plots of the data with initializations and costs that show that 2-means converges to a suboptimal clustering.

Discuss what you think is the issue with this initialization method and how it could be fixed.

(4 + 4 + 4 + 4 + 4 + 5 + 5 marks)

IF ANYTHING IS UNCLEAR, COME TO MY OFFICE HOURS OR WRITE ME EMAILS AND ASK QUESTIONS :)