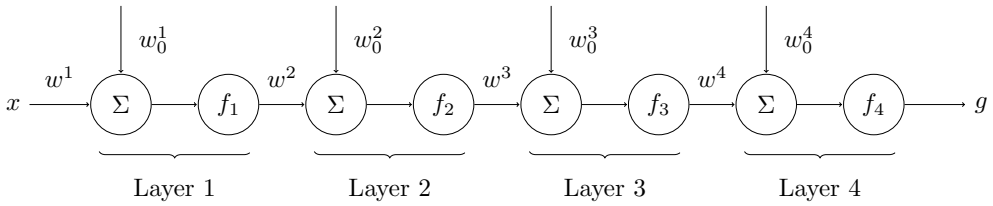
6.390 Introduction to Machine Learning

Recitation Week #8

Issued October 31, 2022

1. Kim constructs a fully connected deep neural network with 4 layers, pictured in the figure below. He uses a squared-error loss and ReLU activation functions for all hidden layers, denoted by f_1, f_2, f_3 in the figure, and an identity activation $f(z) = z$ for the output layer, denoted by f_4. The ReLU activation function is implemented as $\text{ReLU}(z) = \max(0, z)$, with $\partial \text{ReLU}(z)/\partial z = 1$ if $z > 0$, and 0 otherwise. Kim has a data set $\mathcal{D}_n = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$, where each $x^{(i)}$ is a 1-dimensional feature and $y^{(i)}$ is a 1-dimensional label.



Consider the following, which will help us represent how the neural network is operating:

$$a^0 = x, \quad z^l = w^l a^{l-1} + w_0^l, \quad a^l = f_l(z^l), \quad g = a^4.$$

The weights are initialized as follows:

$$w_0^1 = -1,\ w^1 = 3,\ w_0^2 = 1,\ w^2 = 4,\ w_0^3 = -5,\ w^3 = 1,\ w_0^4 = 1,\ w^4 = 1.$$

(a) Before training, Kim is curious about the output of his network as initialized. What will Kim observe at the output of the neural network when he provides the feature, $x^{(1)} = 1$? For each layer l, compute the values of a^l, z^l by means of a *forward pass*.

	$a^0 =$
$z^1 =$	$a^1 =$
$z^2 =$	$a^2 =$
$z^3 =$	$a^3 =$
$z^4 =$	$a^4 =$

(b) Following the above construction, Kim wants to derive the formula for back-propagation. He uses the squared-error loss function, $\mathcal{L}(g^{(i)}, y^{(i)}) = (g^{(i)} - y^{(i)})^2$, where $g^{(i)} = \text{NN}(x^{(i)}; W)$, and W collects all of the weights and offsets across all of the layers. Kim derives the gradient of the loss function with respect to weight w^1 as:

$$\frac{\partial \mathcal{L}(g,y)}{\partial w^1} = \frac{\partial \mathcal{L}(g,y)}{\partial g} \cdot \frac{\partial g}{\partial z^4} \cdot \frac{\partial z^4}{\partial a^3} \cdot \frac{\partial a^3}{\partial z^3} \cdot \frac{\partial z^3}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a^1} \cdot \frac{\partial a^1}{\partial z^1} \cdot \frac{\partial z^1}{\partial w^1}.$$

Provide equations for each of the factors in the equation above:

$\frac{\partial z^1}{\partial w^1} =$	$\frac{\partial a^1}{\partial z^1} =$	$\frac{\partial z^2}{\partial a^1} =$
$\frac{\partial a^2}{\partial z^2} =$	$\frac{\partial z^3}{\partial a^2} =$	$\frac{\partial a^3}{\partial z^3} =$
$\frac{\partial z^4}{\partial a^3} =$	$\frac{\partial g}{\partial z^4} =$	$\frac{\partial \mathcal{L}}{\partial g} =$

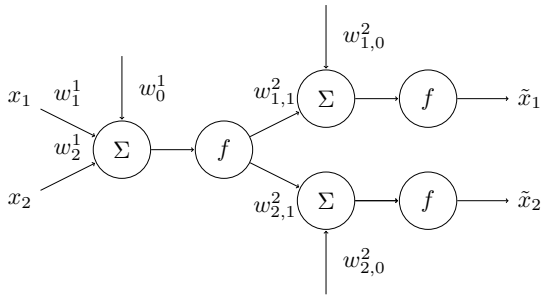
(c) Now, we are well equiped to compute a gradient descent step for updating the weight w^1 through backpropagation. Using the formula,

$$w^1 = w^1 - \eta \frac{\partial \mathcal{L}(g,y)}{\partial w^1},$$

and the components that you found in parts (a) and (b), calculate one gradient descent update for the training data point $(x^{(1)}, y^{(1)}) = (1, 2)$, and a step size $\eta = 0.1$.

(d) Kim next looks to find the gradient with respect to the offset to the first neuron, w_0^1. Write out the equation for $\frac{\partial \mathcal{L}(g,y)}{\partial w_0^1}$, and identify which factors are shared with the equation for $\frac{\partial \mathcal{L}(g,y)}{\partial w^1}$.

2. Otto N. Coder wants to find a way to represent his 2-dimensional data points in 1-dimensional space. Consider the following *autoencoder* with input $x = [x_1, x_2]^\top \in \mathbb{R}^2$ and output $\tilde{x} = [\tilde{x}_1, \tilde{x}_2]^\top \in \mathbb{R}^2$. The autoencoder has one hidden layer with one hidden unit.



The encoder of this autoencoder is described by the output of the first hidden layer,

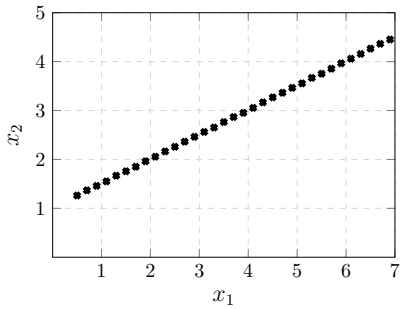
$$a^1 = f(w^1_1 x_1 + w^1_2 x_2 + w^1_0),$$

and the decoder of this autoencoder is described by the outputs of the output layer,

$$\tilde{x}_1 = f(w^2_{1,1} a^1 + w^2_{1,0}),\ \tilde{x}_2 = f(w^2_{2,1} a^1 + w^2_{2,0})$$

The goal is to learn a set of weights such that $x_1 \approx \tilde{x}_1$ and $x_2 \approx \tilde{x}_2$ by means of first representing the input in a lower dimensional space. Autoencoders are generally used for unsupervised learning and *not* for supervised learning problems like regression, but today we're going to play around with shapes like lines and ReLUs just to make sure we understand the basic math first.

(a) For this part, suppose that the activation functions f are the identity $f(z) = z$. Let a dataset $\mathcal{D}_n = \{x^{(i)}\}_{i=1}^n$ be composed of n datapoints which are Cartesian coordinates in 2-dimensional space which are plotted below:



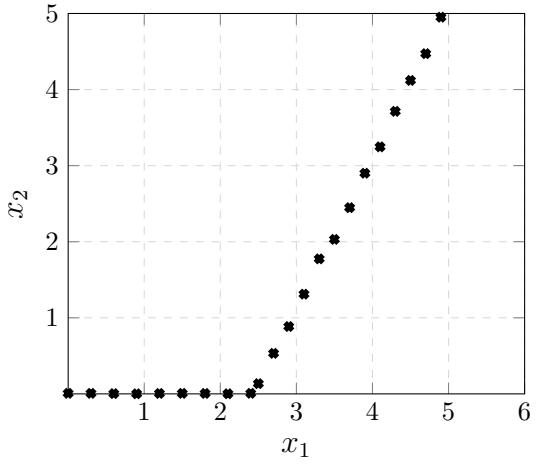
i. Can you write a formula for x_2 as a function of x_1?

ii. Consider the autoencoder structure depicted in the figure at the start of this question. Using your result from Part (a).i., can you find weights and offsets for this autoencoder that just about perfectly recover $\tilde{x}_1 = x_1$ and $\tilde{x}_2 = x_2$, even though there is only one hidden unit?

$w_0^1 =$	$w_1^1 =$	$w_2^1 =$
$w_{1,0}^2 =$	$w_{1,1}^2 =$	
$w_{2,0}^2 =$	$w_{2,1}^2 =$	

iii. Intuitively, what would the weights and offsets of the autoencoder be learning?

(b) Otto modifies his autoencoder to have reLU activation functions so that he can model nonlinear relationships. Suppose that he has a new data set $\mathcal{D}_n = \{x^{(i)}\}_{i=1}^n$ composed of n datapoints, plotted in this figure:



Again consider the autoencoder structure in the figure at the start of this question. Find values for the weights and offsets in the autoencoder that can just about perfectly recover this new data set.

$w_0^1 =$	$w_1^1 =$	$w_2^1 =$
$w_{1,0}^2 =$	$w_{1,1}^2 =$	
$w_{2,0}^2 =$	$w_{2,1}^2 =$	