Purdue University Duc Viet Le
CS 59000SA

Assignment 4

Problem 1. We have a convolutional neural network for images of 5 by 5 pixels. In this network, each hidden unit is connected to a different 4 x 4 region of the input image: The first hidden unit is connected to the upper left 4x4 portion of the input image. The second hidden unit is connected to the upper right 4x4 portion of the input image. The third hidden unit is connected to the lower left 4x4 portion of the input image. The fourth hidden unit is connected to the lower right 4x4 portion of the input image. Because its a convolutional network, the weights (connection strengths) are the same for all hidden units: the only difference between the hidden units is that each of them connects to a different part of the input image. The array of weights, which are the same for each of the four hidden units, are given below.

$$\begin{array}{llll} w_{11}=1 & w_{12}=1 & w_{13}=1 & w_{14}=0 \\ w_{21}=0 & w_{22}=0 & w_{23}=1 & w_{24}=0 \\ w_{31}=1 & w_{32}=1 & w_{33}=1 & w_{34}=0 \\ w_{41}=0 & w_{42}=0 & w_{43}=1 & w_{44}=0 \end{array}$$

Consider the image

$$\begin{array}{ccccc} 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 255 & 0 \\ 0 & 0 & 255 & 255 & 0 \\ 0 & 0 & 0 & 255 & 0 \\ 0 & 255 & 255 & 255 & 0 \end{array}$$

For the training case with that "3" input image, what is the output y_1, y_2, y_3, y_4 of each of the four hidden units?

Ans: In order to compute h_i, we pair-wise product the weight matrix with the area it's assigned to, then adding up the sum:

$$\begin{aligned} y_1 &= \sum_{1\le i,j\le 4} w_{ij}x_{ij} = 765 \\ y_2 &= \sum_{1\le i,j\le 4} w_{ij}x_{i(j+1)} = 1785 \\ y_3 &= \sum_{1\le i,j\le 4} w_{ij}x_{(i+1)j} = 510 \\ y_4 &= \sum_{1\le i,j\le 4} w_{ij}x_{(i+1)(j+1)} = 1020 \end{aligned} \tag{1}$$

Therefore, we have $(y_1, y_2, y_3, y_4) = (765, 1785, 510, 1020)$

Problem 2. Claire had a dataset of 28 x 28 pixel handwritten digits nicely prepared to train a neural network, but Brian has gone and accidentally scrambled the images by re-ordering the pixels in some totally meaningless way, and now they cant get the original dataset back! Luckily, all of the images (in both the training set and the test set) were changed in the same way. For example, if pixels number 1 and number 3 switched places in one image, then they switched places in every other image as well. Because of that,

Claire thinks that perhaps she can still train a neural network to identify handwritten digits, using these scrambled images.

Whether Claire is right or not depends largely on the type of neural network that she has in mind. Which of the following neural networks will be at a disadvantage because of Brians mistake? Choose all that apply. Explain your choices.

1. A feed-forward neural network with no hidden layer and logistic units (and no convolution).
2. A feed-forward neural network with one hidden layer of linear units (and no convolution).
3. A convolutional neural network where the size of each weight filter is 8 x 8.
4. A convolutional neural network where the size of each weight filter is 10 x 10.

Ans: 3 and 4 are correct answers. In general, because in the case 1 or case 2, the order of the input pixels does not matter much to the feed-forward neural network. To the feed-forward neural network, two pixels close to each other are just like two different possibly uncorrelated input features. On the other hand, a filter of a CNN looks at patches of the original image. The order of pixels actually matters to CNN –¿ if we shuffle the pixels, it will hurt CNN. Also, the feed-forward neural network is not suitable for image, because it's vulnerable to position, size, orientation. Convolution neural network deal with the problem of orientation by applying filter and pooling with function such as sum and max such that the ordering of the pixel does not matter

Problem 3. Consider a neural network with only one training case with input $x = (x_1, x_2, \ldots, x_n)^T$ and correct output t. There is only one output neuron, which is logistic, i.e. $y = (wx)$ (notice that there are no biases). The loss function is squared error. The network has no hidden units, so the inputs are directly connected to the output neuron with weights $w = (w_1, w_2, \ldots, w_n)^T$. Were in the process of training the neural network with the backpropagation algorithm. What will the algorithm add to w_i for the next iteration if we use a step size (also known as a learning rate) of ϵ?

Ans: I think there is a typo in this question: "logistic, i.e. $y = (wx)$" $\rightarrow$ "logistic, i.e. $z = (wx)$"

If it's logistic, I believe having no biases should not change anything because:

$$\frac{\partial y}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{dy}{dz} = x_i y(1-y) \tag{2}$$

Therefore, we have:

$$\frac{\partial E}{\partial w_i} = x_i y(1-y)(t-y) \tag{3}$$

So, with ϵ learning rate, we have to update w_i:

$$\Delta w_i = \epsilon x_i y(1-y)(t-y) \tag{4}$$

Problem 4. Suppose that we have a vocabulary of 3 words, a, b, and c, and we want to predict the next word in a sentence given the previous two words. Also suppose that we dont want to use feature vectors for words: we simply use the local encoding, i.e. a word is encoded by a 3-component vector with one entry being 1 and all other two entries being 0.

In the language models we discussed, each of the context words has its own dedicated section of the network, so we would encode this problem with two 3-dimensional inputs. That makes for a total of 6 dimensions; clearly, the more context words we want to include, the more input units our network must have. Heres a method that uses fewer input units:

We could instead encode the counts of each word in the context. So a context of a a would be encoded as input vector [2 0 0] instead of [1 0 0 1 0 0], and b c would be encoded as input vector [0 1 1] instead of [0 1 0 0 0 1]. Now we only need an input vector of the size of our vocabulary (3 in our case), as opposed to the size of our vocabulary times the length of the context (which makes for a total of 6 in our case). Are there any significant problems with this idea? Choose one below. Justify your answer

1. Yes: although we could encode the context in this way, we would then need a smaller bottleneck layer than we did before, thereby lowering the learning capacity of the model.

2. Yes: even though the input has a smaller dimensionality, each entry of the input now requires more bits to encode, because its no longer just 1 or 0. Therefore, there would be no significant advantage.

3. Yes: the network loses the knowledge of the location at which a context word occurs, and that is valuable knowledge.

4. Yes: the neural networks shown in the course so far cannot deal with integer inputs (as opposed total binary inputs).

Ans: Option 3 is the correct answer. This is the correct answer, because as mentioned, the network loses the knowledge of the location at which a context work occurs.