Weicong Feng

Professor Konstantinos Krampis

BIOL79303

09/28/2021

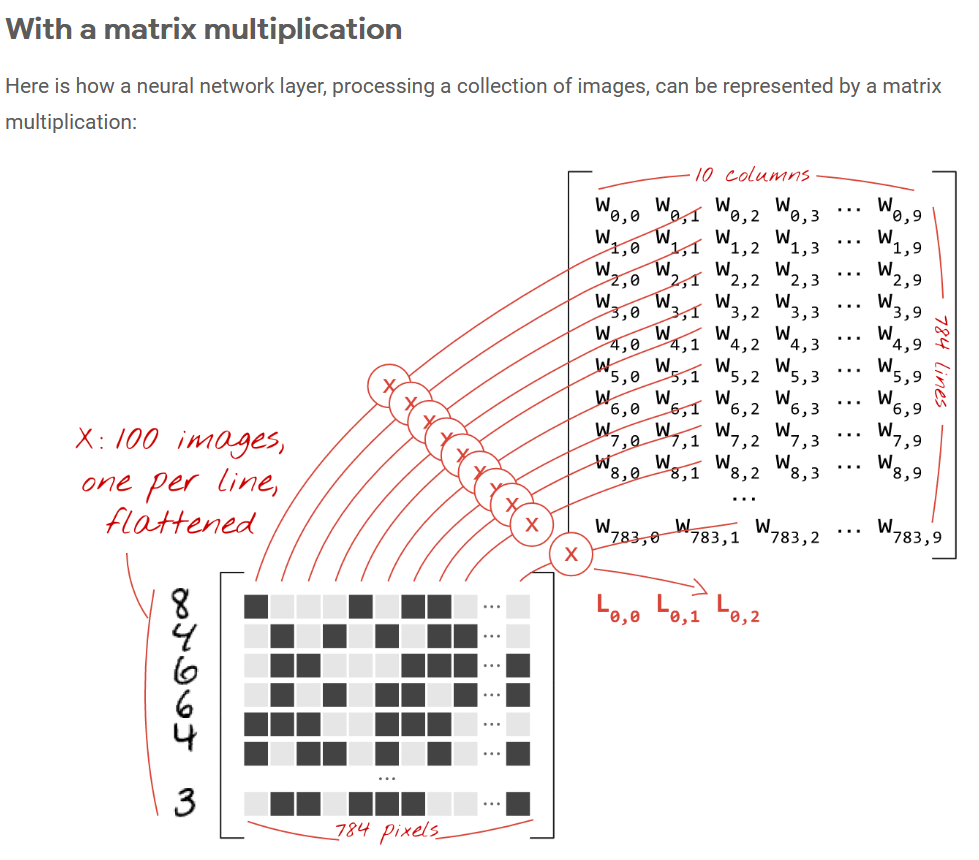
Assignment4

Assignment 4

Due date: Wednesday, 09/29/2021

**Question 1.**

On the slide 4 “Neural Networks 101” shown on the link and image below, <https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist#3>

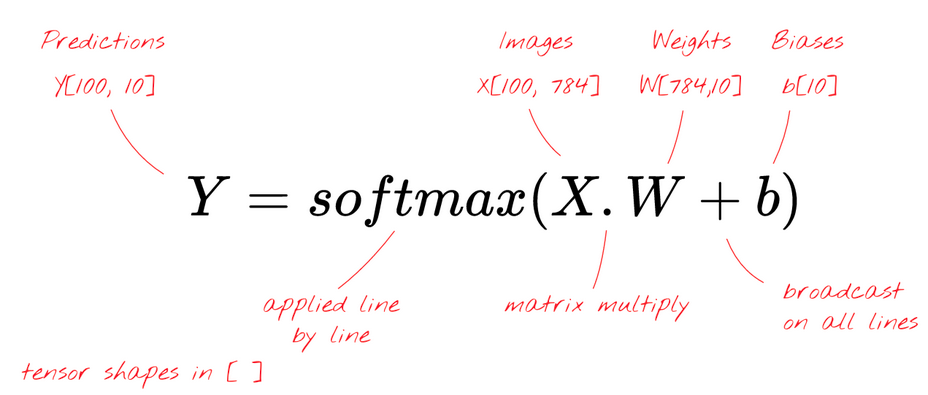


Please clarify why the weights matrix W has dimensions 10x784; explain what each dimension corresponds to and how it is used to represent the summations / multiplications in the logit.

Ans: The weight matrix W has dimension of 10x784 is because the input X has a dimension of 100x784, and the number of neurons in that layer is 10. It means the first dimension corresponds to the number of neurons in specific layer, the second dimension corresponds to the second dimension of input or the number of neurons in previous layer. Notice that the multiplication is XWT instead of WX. So, the multiplication happens on 100x784 and 784x10, as shown above, it ends up with a result of 100x10. After transposing of W, each row of X (instance in input matrix) is multiple with each column of WT, then sum up. Each column in WT corresponds to the weights in different neurons. In the result, the result of each logit function is a scalar, each row of the result corresponds to a instance in input, while each column corresponds to the result from 10 neurons respectively.

**Question 2.**

On the same slide 4 “Neural Networks 101”, you see the matrices from Question 1 in the formula

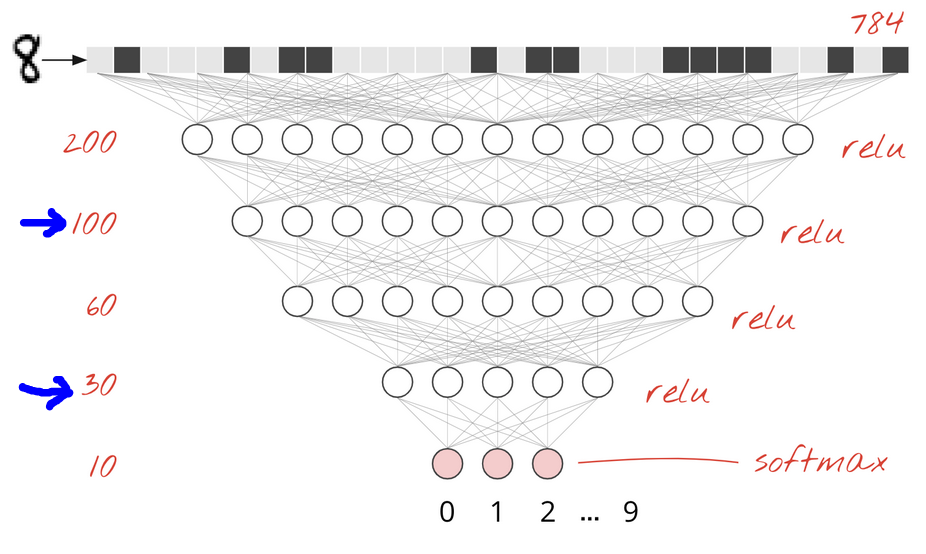


What is the “broadcasting” used for in this matrix multiplication and how does it work? Why are the predictions matrix 100x10?

Ans: Broadcasting refers to how matrix operations behave when their matrices differ in size. Smaller matrix or vector are “broadcast” across the larger one, until they have a compatible shape. As the example in “Neural Networks 101”, the dot product between input of 100x784 and WT of 784x10 ends up with a matrix of 100x10. The b is a vector of 1x10 in which each element corresponds to a neuron. When b is added to the result matrix of dot product between X and WT which is a matrix of 100x10, the b will be broadcasted to 100x10. It means b will be copied 100 times and stacked vertically. So, the sum of X.WT+b is still 100x10. Softmax is an element-wise operation. As a result, the Y is a matrix of 100x10.

**Question 3.**

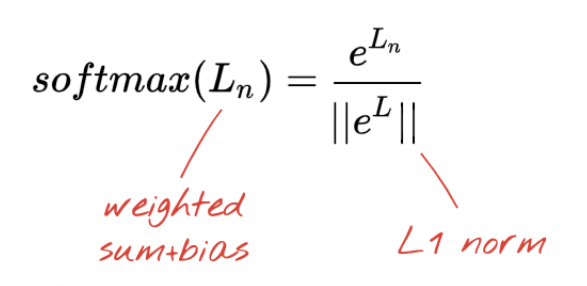
On the same slide 4 “Neural Networks 101”, see the image below. How many connections we have coming in each neuron in the layers I point with blue arrow below? Why do we have 10 neurons in the last layer?



Ans: In the 2nd layer with 100 neurons, there are 200 connections to previous layer and 60 connections to subsequent layer in each neuron; in the 4th layer with 30 neurons, there are 60 connections to previous layer and 10 connections to subsequent layer in each neuron. The reason why there are 10 neurons in the last layer is the model need to output 10 possibilities of 10 classes, where each neuron corresponds to a possibility of a digit.

**Question 4.**

On the same slide 4 “Neural Networks 101”, what is the output of the softmax function when applied to the logit (the L). How does the output of the softmax compares with previous logit functions like the logistic for example, and what does this output represent?



Ans: The softmax function is a generalization of the logistic function to multiple dimensions. It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.



In multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability for the j'th class given a sample vector x and a weighting vector w is:

Text

Description automatically generated

The input of logistic function typically is a scalar, and output a scalar between 0 to 1 to binarily classify; while the input of softmax function typically is a list which include multiple scalar elements, and output a list with same number of elements that correspond the probabilities of each input elements, so it is usually used as the activation function in multiple classifiers.

**Question 5.**

On the same slide 4 “Neural Networks 101”, what is cross-entropy? What do the valleys and mountains in the cross-entropy plot shown below?

Ans: The cross-entropy is a common cost function in machine learning, which is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions. In binary classification, the cross-entropy can be calculated as:

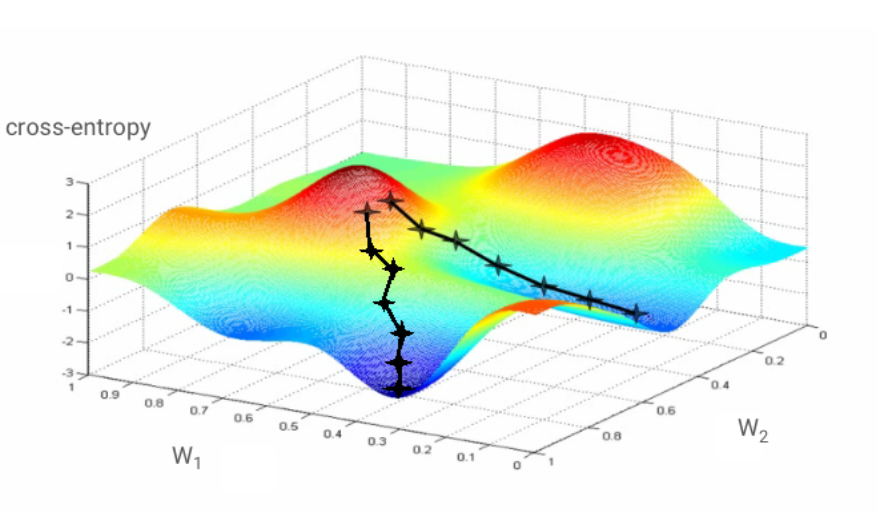
A picture containing text, watch

Description automatically generated

When it is used in multiclass classification, it can be computed as:

Text

Description automatically generated



The peaks on above chart stand for the high cross-entropy, while the valleys are the low points of cross-entropy. During the training, gradient descent is used to minimum the loss function, cross-entropy.

**Question 6.**

How do the filters / kernels designed to capture elements of images in CNNs? Could you give an example of a 10x10 filter that is designed to capture vertical lines towards the left edge of the image and, and one filter that is designed to capture semi-circles (the open side of the semi-circle is towards the right side of the image). For helping you to answer the question, please see the additional links posted regarding the CNNs, and the information therein, which we already touched upon in the class.

Ans:

1. The filter captures vertical line on the left:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |

1. The filter captures semi-circles with open side towards the right:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Question 7.**

Define the terms filter / kernel, stride, activation map (or wrongly “layer”) and padding, in Convolutional Neural Networks (CNNs). Let’s say you have a set of images which have mostly vertical, horizontal, diagonal lines and a few circles. How many filters would you design to capture all these elements in the image?

Ans: The filter, is called kernel, is set of weights that are used to convolute with input data or previous level output activation map to detect spatial patterns such as edges in an image. The stride is the number of pixels by which the window moves after each operation. The padding refers to the number of rows and columns of pixels adding zeroes to each side of the boundaries of the input. Activation map, is called feature map, is the output of each layer in a CNN model, the pattern captured from input.

It need 4 filters to capture these elements in the image of the example.

**Question 8.**

What are the resulting size number of activation maps (wrongly I called “layer”), when we have a starting image of 100x100 pixels, with 30 filters/kernel of size 20x20, stride 2 and padding 0?

Ans: It should be 30x41x41.

**Question 9.**

In the posted paper “Deep Learning: New Computational Modeling Techniques for Genomics” in the class materials, explain how does the DNA is transformed to and image so that can be processed with CNNs? List some of the applications mentioned in the paper of Artificial Neural Networks in general to DNA data.

Ans: In the paper, the DNA sequence is represented by one-hot encode. The paper shows some ANN applications on DNA data, for instance, predicting the binding affinity of the TAL1–GATA1 transcription factor complex, classifying genomic regions as bound versus unbound by a particular transcription factor, detecting an open reading frame in a DNA sequence, modelling polypharmacy side effects, and so on.

**Question 10.**

What are the different types of Artificial Neural Networks described in the posted paper? Describe differences of each with the feed forward neural networks we have seen so far, and what are the ideal applications for each of these new types of Artificial Neural Networks described in the paper.

Ans: The paper describes CNN(Convolution Neural Network), RNN(Recurrent Neural Network), GCN(Graph-convolutional Neural Network).

CNN scans the input sequence by filters, sharing parameters, so that the total number of parameters is drastically reduced. Different filter can capture various features. CNN is widely used in predicting binarized in vitro and in vivo binding affinities of a transcription factor and the in vitro binding affinity of an RNA-binding protein, predicted the presence or absence of 919 chromatin features, predicted 164 binarized DNA accessibility features given a 600 bp sequence.

RNN is developed for time series data. The output of previous layer is combined with input are feed into next layer in RNN. This architecture can decide what information should be kept and what information should be forgot. Hence, CNN carry over information through infinitely long sequences via memory and process sequences of widely varying length, such as mRNA sequences. However, it is much slower than CNNs because it is difficult to parallelized. An RNN could detect an open reading frame in a DNA sequence regardless of the position in the sequence. At the same time, RNN has a lot of applications in miRNA biology.

GCN use the individual features of nodes in a graph and the node connectivity to solve machine learning tasks on graphs. GCNs sequentially apply multiple graph transformations (layers), whereby each graph transformation aggregates features from the neighboring nodes or edges in a nonlinear manner and represents nodes or edges with a new set of features. GCN can be used in node classification80,81, unsupervised node embedding, edge classification and graph classification.