Indium Early Career Program **Meera Bharadwaj**

Mentor: Gayathri Jayaraman, Indium Software Date: 05/22/2024-07/31/2024

What are Machine Learning Algorithms

Machine learning algorithms use past data to predict the future. Consider a simplified algorithm that, say, calculates the price of a house. This would depend on multiple factors, carpet area, location, how old the property is, additional features and so on. Each factor is given a certain level of importance which is what we want to figure out. Consider the training phase of the algorithm. The goal is to figure out the correct set of inputs that would predict the cost. (Here, an example of an input might be 100 for location, 40 for additional features and so on) The machine is fed the correct price-call it y_{true}. The machine picks out a certain set of inputs and calculates the cost for a certain set of weights(an example of a weight would be cost per square meter). This price is y. The difference squared between y and y true is the cost function. We need to minimize the cost function. This can be done using stochastic gradient descent. This is an example of supervised machine learning, where the user feeds the correct output to reduce the cost. Some machine learning algorithms include linear regression, logistic regression, decision trees.

An Example- Naive Bayes Algorithm

Consider the naive Bayes algorithm. This is a classifier algorithm commonly used to classify email as regular or spam. Regular and spam messages have different content and thus different probabilities of certain words occurring in them. The word money might appear in spam emails a lot more than regular ones. Say the probability that a mail is spam given that money appears is 0.15. Say the probability that a mail is normal given that money appears is 0.02. The probability that the phrase 'shall we meet?' appears given that it is a normal mail is 0.05. The probability that the phrase 'shall we meet?' appears given that it is a spam mail is 0.01 and so on. These values are dependent on the training set. Say the probability that a mail is normal is 0.7 and spam is 0.3. Consider a new email that we need to classify. For example, say the email has the words money, money, shall we meet. The probability that it's spam is proportional to the
$p(spam) * p(money|spam) * p(money|spam) * p(spam|shallwemeet)$. Here that value is 6.75E-5
The probability that it's regular is proportional to
$p(regular) * p(money|regular) * p(money|regular) * p(regular|shallwemeet)$.
Here that value is 1.4E-5. Now, the mail is classified as spam mail.

Consider the case where the word buy is classified as appearing in spam but not in regular emails. The new email has the word buy.To avoid the problem of the email having a probability of zero when the email has the words buy, shall we meet, shall we meet, we add constant alpha to the conditional probabilities.

This algorithm is considered naive as it does take into account the order of words or grammatical syntax. All words are considered to occur independently.

Multilayer Perceptron

Consider a multilayer perceptron that recognizes handwritten digits. Each neuron has an activation between 0 and 1.

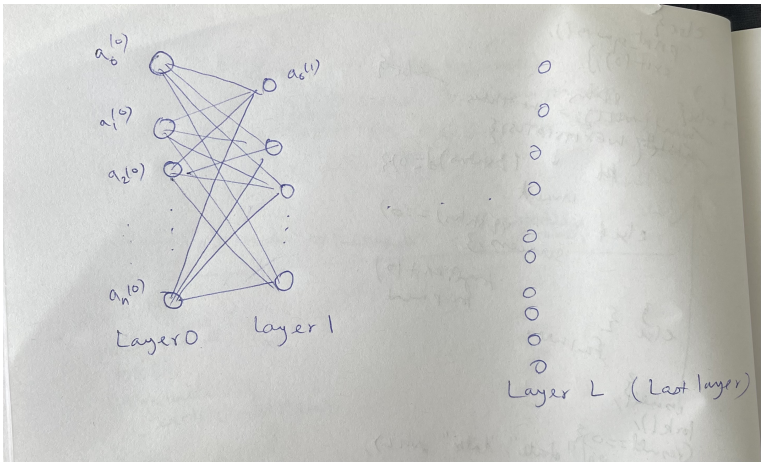

Figure 1: A neural network

Each neuron is connected to every neuron in its adjacent networks and each connection has a certain weight. The activation of a neuron in the first layer depends on the weights and activation of the neurons in the previous(zeroth) layer.
$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + w_{0,2}a_2^{(0)} + ... + w_{0,n}a_n^{(0)} + b_0)$ where the Sigmoid function is $\sigma = 1/(1 + e^{-x})$
The above equation dictates the activation of one neuron. For an entire layer, the activation is

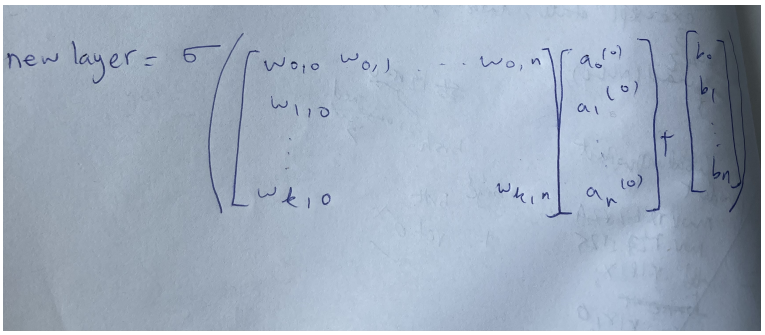

Figure 2: activation of new layer

Consider all the weights and the biases of out network as a column vector that we need to determine. Our network has nine possible outputs, numbers from 0 to 9.
The cost function is the sum of the square of the difference between the true value and the recognized value. To reduce the cost function over all sets of data, we use gradient descent. We adjust the inputs in the direction of the steepest gradient descent. Initially, the network starts with random weights and biases.

Backpropagation

Say we want the network to recognize the input image as 3.

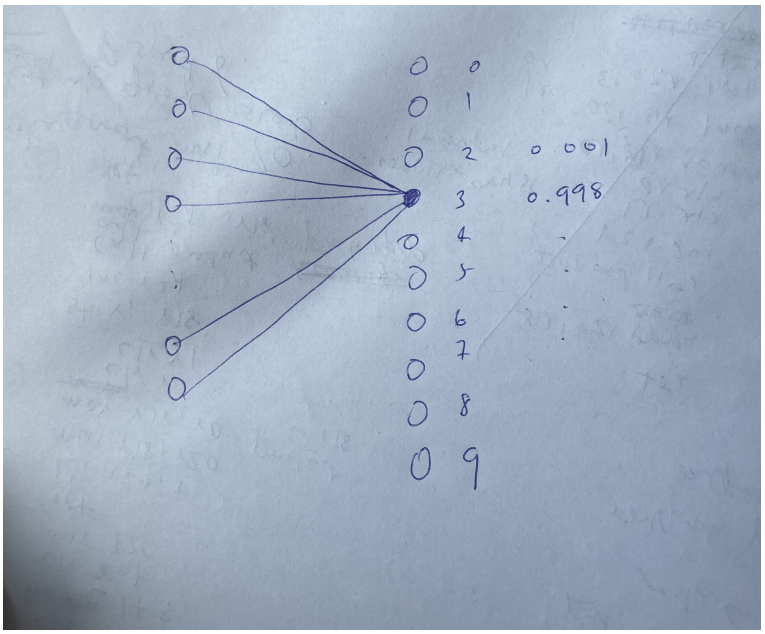

Figure 3: network recognizes image as 3

Now the the activation of 3 is a 0.5. We need it to be 1 or very close to 1. The activation of the other numbers need to be close to 0.
To achieve this, one can increase the weights that connect to three from neurons with higher activation.
One can increase the bias
Or one could increase the activation of the neurons in the previous layer.
Now consider the 2 neuron. Its activation needs to decrease, which would result in a different set of actions. The net result is the sum of the required actions of each such neuron. We apply the same idea to every neuron in the previous layer and so on.
Now consider a simple two neuron network.
Refer fig 4.
Again, average over all the samples in the training data. This is computationally heavy, so random sets are chosen for the costs to be averaged over. This is stochastic gradient descent.

Transfer Learning

Transfer learning is used when we want out model to perform a task similar to one that model was originally trained on. An analogy: someone who has trained regularly or played a sport would find it easier to take on a challenging hike. The mobility and core strength gained over regular training would help them in said hike.
We use transfer learning when there are pre-existing models trained on extensive data similar to the model we want to use or if our task and the model's task have the same

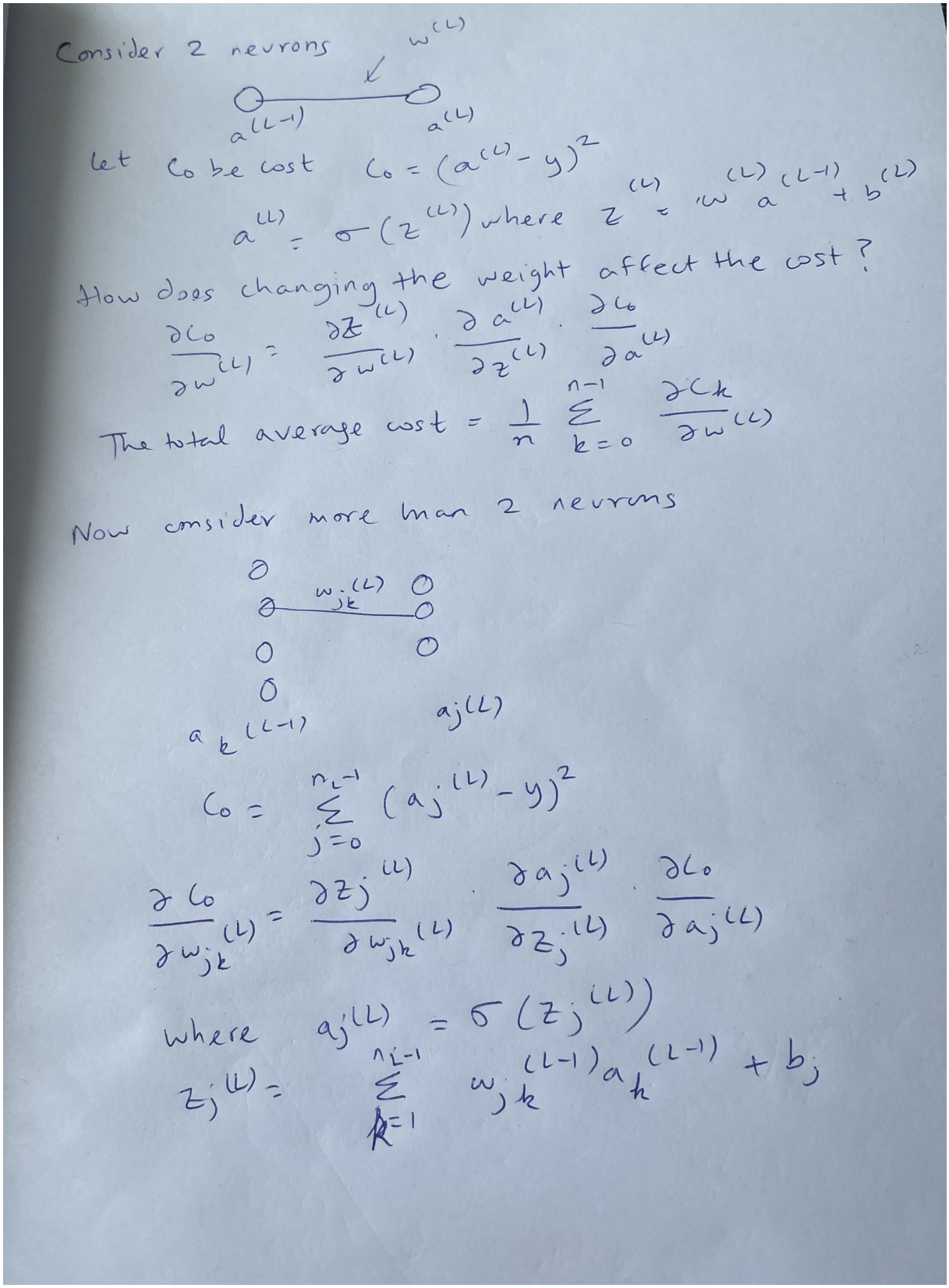

Figure 4: how does changing the weights affect the cost

inputs. If the inputs do not match, add a pre-processing layer to change the input size. The original model would have two kinds of layers: frozen and modifiable. The frozen layers are the layers from the original model that we do not modify. These layers collect and process data that is similar to our input data. The modifiable layers are the layers that we train during fine-tuning. These layers take into account the specific requirements of our task and input data. Transfer learning is used in natural language processing and computer vision as these tasks require huge amounts of computational power.
Used transfer learning with pytorch to classify images based on the scene present- building, forest, mountain, glacier, sea, street using 25k images from a kaggle dataset

Conclusion

Learnt deep learning, neural networks, natural language processing concepts to build and train a classifier with transfer learning.

Sources

https://www.3blue1brown.com/topics/neural-networks
https://www.youtube.com/watch?v=O2L2Uv9pdDA
https://builtin.com/data-science/transfer-learning
https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/