Deep Learning and Neural Networks

Inal Mashukov

University of Massachusetts Boston

Summer 2023

1 Dot Product

An artificial neural network consists of an input layer, hidden or dense layers, and an output layer. Each layer can be represented as a vector of **neurons**. The input layer, $x \in \mathbb{R}^{n \times 1}$ is passed into the network and is sequentially multiplied by the set of weights $W = \{w_i\}_{i=1}^{m}$, where each $w_i \in \mathbb{R}^{n \times 1}$. In other words, we take the dot product of two vectors $x \cdot w_i \in \mathbb{R}$. Each neuron has a certain bias associated with that neuron, and it is added to the result to produce the result:

$$z_{i,j} = \sum_{k=1}^{n} w_{i,k,j} \cdot x_{i,k} + b_{i,j} \tag{1}$$

$$\implies z_i = \forall_{j=1}^{m} \sum_{k=1}^{n} w_{i,k,j} \cdot x_{i,k} + b_{i,j} \tag{2}$$

where $z_i \in \mathbb{R}^m$ is the new vector in the hidden layer.

2 Activation Function

The result of the dot product z_i is then passed through an activation function which determines whether a particular neuron "fires" or becomes activated.

Activation functions include non-linear ones such as the Sigmoid:

$$y_i = \frac{1}{1 + e^{-z_i}} \tag{3}$$

as well as Rectified Linear Unit:

$$y_i = ReLU(z_i) = max(0, z_i) \tag{4}$$

$$\implies y_i = \forall_{j=1}^{m} \max\left(0, \sum_{k=1}^{n} w_{i,k,j} \cdot x_{i,k} + b_{i,j}\right) \tag{5}$$

An activation function can also be linear:

$$y_i = z_i \tag{6}$$

In the case of the output layer, we can utilize the Softmax activation function which normalizes the values in a vector and outputs a probability distribution:

$$\hat{y_i} = softmax(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{n} e^{z_j}} \tag{7}$$

The Softmax activation function also prevents **overflow**, effectively preventing the input values from exploding, normalizing them while retaining their relative scale.

3 Loss Function

To measure the accuracy of a neural network, we apply a loss function which become the error metric for the neural network. Applicable functions include but are not limited to functions like the absolute error function:

$$|x(t) - r(t)| \tag{8}$$

where we take the difference between the predicted and the observed values. Another function that can be used to measure loss is the mean absolute error:

$$\frac{|x(t) - r(t)|}{n} \tag{9}$$

where n is the number of examples.
A networks accuracy can be a binary measure of the error, that is, comparing the results for a class as true or false, but losing information at the same time. The loss function of choice for classification with Softmax being the activation is the categorical cross entropy:

$$L_i = -\sum_j y_{i,j} \log(\hat{y}_{i,j}) \tag{10}$$

where L_i is the sample loss value, $y, \hat{y}$ being the target and predicted values, respectively. This loss function is convenient to use when doing back-propagation and optimization.

4 Training and Loss Optimization

Training a neural network is done to identify how erroneous it is, to measure the loss, and to minimize it. We want to find the network **weights** that achieve the lowest loss:

$$W_i = argmin_W(L_i) \tag{11}$$

The procedure for minimizing a continuous function of loss is called **gradient descent**. The algorithm for gradient descent includes:

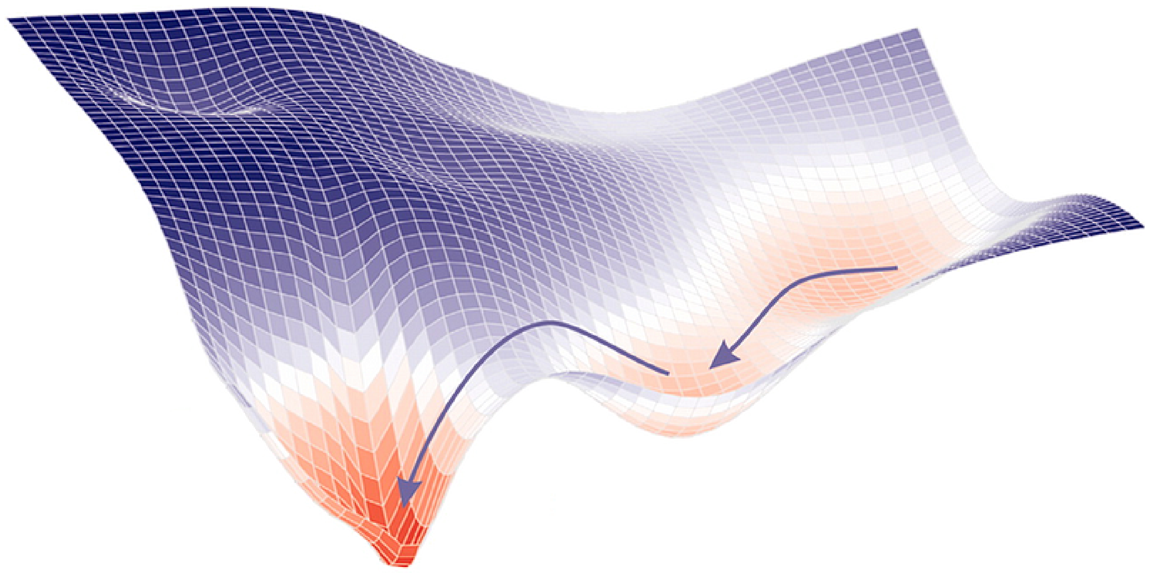
1. Randomly pick initial weights $(w_0, w_1, ...w_i) \sim N(\mu, \sigma^2)$.
2. Compute the loss at the specific location, as well as its derivatives - the gradient:
$$\nabla L(W) = \frac{\partial L(W)}{\partial W} \tag{12}$$
3. The gradient thus tells us the direction of greatest loss - the maxima. $\implies$ Take small steps in the direction opposite to the gradient.
4. Repeat the procedure until convergence
5. Update the weights:
$$W_t = W_{t-1} - \alpha \frac{\partial L(W)}{\partial W} \tag{13}$$
where α is the learning rate.

We then recursively **propagate** the gradients back through the hidden layers.

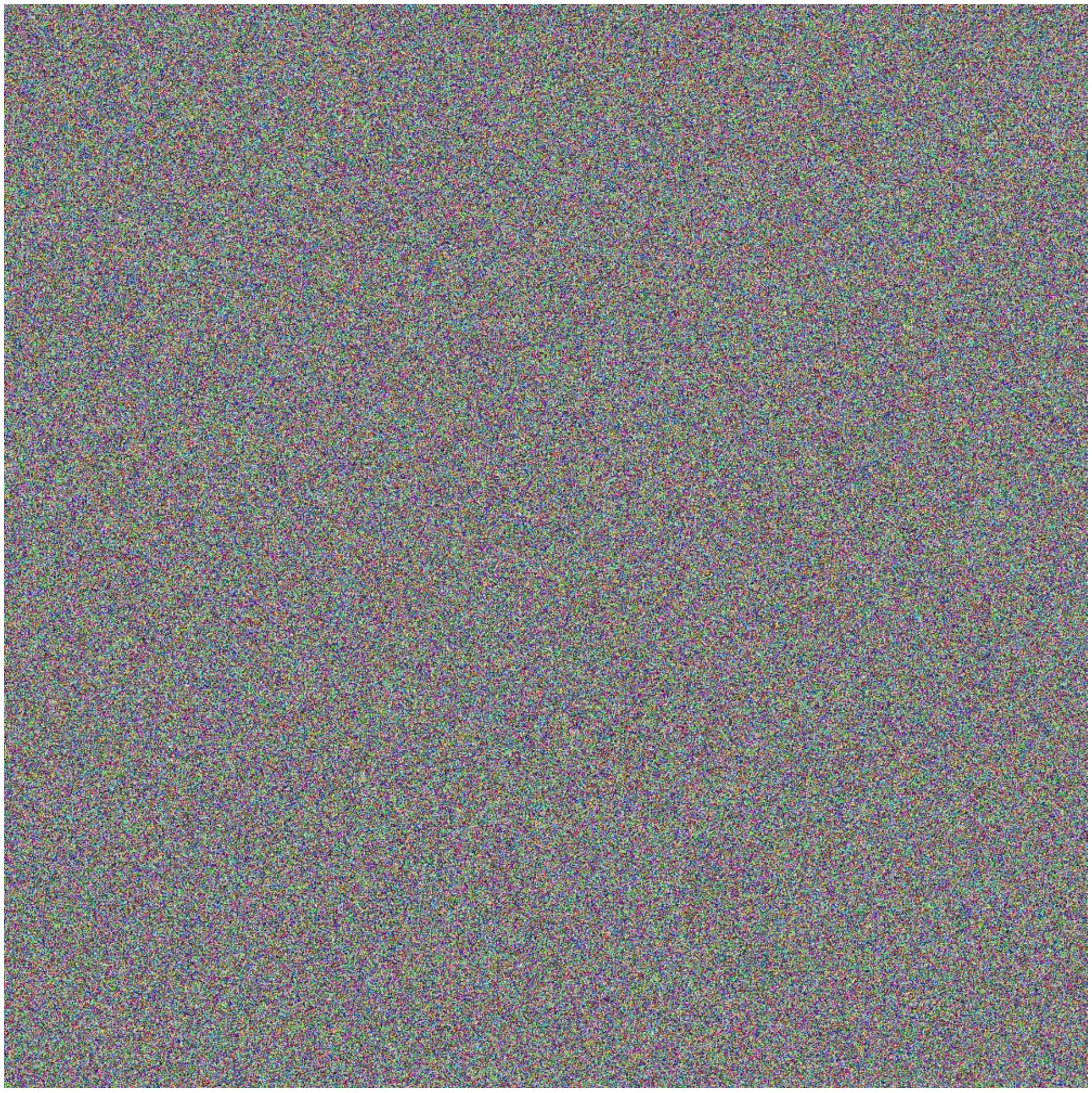
5 Overfitting and Underfitting, Regularization

Underfitting refers to the situation when the resulting model does not have the capacity to fully learn the data. Overfitting refers to the situation when the model is too complex and has extra parameters, it focuses too much on memorizing the data it has already seen, failing to capture the general pattern when provided with new information.
Regularization is a technique that constrains our optimization problem to discharge complex models, improving generalization of our model on unseen data.
Some regularization methods include the dropout regularization - during training, randomly set some activations to 0, as well as early stopping - stop training before we have a chance to overfit.

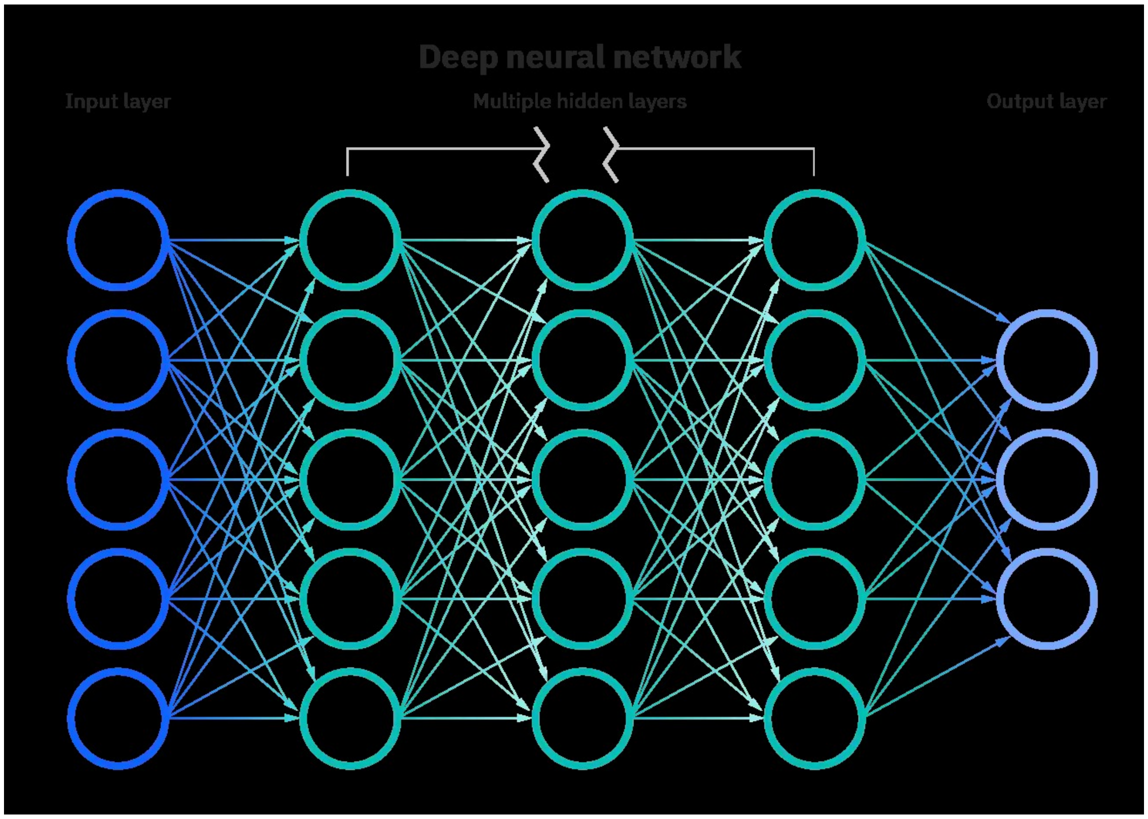
6 An Illustration of Gradient Descent



7 An Image of a Matrix of Random Values in RGB



8 An Example Diagram of a Neural Network



References

[1] Mashukov, I., University of Massachusetts, Boston, MA, USA, 2023.