Motivation of Deep Learning, and Its History and Inspiration

🎙️ *Yann LeCun*

Course plan

- Basics of Supervised Learning, Neural Nets, Deep Learning
- Backpropagation and architectural components
- Convolutional neural network and its applications
- More Deep Learning Architectures
- Regularization Tricks / Optimization Tricks / Understanding how Deep Learning works
- Energy-based models
- Self-supervised learning and beyond

Inspiration of Deep Learning and its history

On a conceptual level, deep learning is inspired by the brain but not all of the brain's details are relevant. For a comparison, aeroplanes were inspired by birds. The principle of flying is the same but the details are extremely different.

The history of deep learning goes back to a field which changed its name now to cybernetics. It started in the 1940s with McCulloch and Pitts. They came up with the idea that neurons are threshold units with on and off states. You could build a Boolean circuit by connecting neurons with each other and conduct logical inference with neurons. The brain is basically a logical inference machine because neurons are binary. Neurons compute a weighted sum of inputs and compare that sum to its threshold. It turns on if it's above the threshold and turns off if it's below, which is a simplified view of how neural networks work.

In 1947, Donald Hebb had the idea that neurons in the brain learn by modifying the strength of the connections between neurons. This is called hyper learning, where if two neurons are fired together, then the connection linked between them increases; if they don't fire together, then the connection decreases.

Later in 1948, cybernetics were proposed by Norbert Wiener, which is the idea that by having systems with sensors and actuators, you have a feedback loop and a self-regulatory system. The rules of the feedback mechanism of a car all come from this work.

In 1957, Frank Rosenblatt proposed the Perceptron, which is a learning algorithm that modifies the weights of very simple neural nets.

Overall, this idea of trying to build intellectual machines by simulating lots of neurons was born in 1940s, took off in 1950s, and completely died in late 1960s. The main reasons for the field dying off in 1960 are:

- The researchers used neurons that were binary. However, the way to get backpropagation to work is to use activation functions that are continuous. At that time, researchers didn't have the idea of using continuous neurons and they didn't think they can train with gradients because binary neurons are not differential.
- With continuous neurons, one would have to multiply the activation of a neuron by a weight to get a contribution to the weighted sum. However, before 1980, the multiplication of two numbers, especially floating-point numbers, were extremely slow. This resulted in another incentive to avoid using continuous neurons.

Deep Learning took off again in 1985 with the emergence of backpropagation. In 1995, the field died again and the machine learning community abandoned the idea of neural nets. In early 2010, people start using neuron nets in speech recognition with huge performance improvement and later it became widely deployed in the commercial field. In 2013, computer vision started to switch to neuron nets. In 2016, the same transition occurred in natural language processing. Soon, similar revolutions will occur in robotics, control, and many other fields.

Supervised Learning

90% of deep learning applications use supervised learning. Supervised learning is a process by which, you collect a bunch of pairs of inputs and outputs, and the inputs are feed into a machine to learn the correct output. When the output is correct, you don't do anything. If the output is wrong, you tweak the parameter of the machine and correct the output toward the one you want. The trick here is how you figure out which direction and how much you tweak the parameter and this goes back to gradient calculation and backpropagation.

Supervised learning stems from Perceptron and Adaline. The Adaline is based on the same architecture with weighted inputs; when it is above the threshold, it turns on and below the threshold, it turns off. The Perceptron is a 2-layer neuron net where the second layer is trainable and the first layer is fixed. Most of the time, the first layer is determined randomly and that's what they call associative layers.

History of Pattern Recognition and introduction to Gradient Descent

The foregoing is the conceptual basis of pattern recognition before deep learning developed. The standard model of pattern recognition consists of feature extractor and trainable classifier. Input goes into the feature extractor, extracting relevant useful characteristics of inputs such as detecting an eye when the purpose is recognizing the face. Then, the vector of features is fed to the trainable classifier for computing weighted sum and comparing it with the threshold. Here, a trainable classifier could be a perceptron or single neural network. The problem is feature extractor should be engineered by hand. Which means, pattern recognition/computer vision focus on feature extractor considering how to design it for a particular problem, not much devoted to a trainable classifier.

After the emergence and development of deep learning, the 2-stage process changed to the sequences of modules. Each module has tunable parameters and nonlinearity. Then, stack them making multiple layers. This is why it is called “deep learning”. The reason why using nonlinearity rather than linearity is that two linear layers could be one linear layer since the composition of two linear is linear.

The simplest multi-layer architecture with tunable parameters and nonlinearity could be: the input is represented as a vector such as an image or audio. This input is multiplied by the weight matrix whose coefficient is a tunable parameter. Then, every component of the result vector is passed through a nonlinear function such as ReLU. Repeating this process, it becomes a basic neural network. The reason why it is called a neural network is that this architecture calculates the weighted sum of components of input by corresponding rows of a matrix.

Back to the point of supervised learning, we are comparing the resulting output with target output then optimize the objective function which is the loss, computing a distance/penalty/divergence between the result and target. Then, average this cost function over the training set. This is the goal we want to minimize. In other words, we want to find the value of the parameters that minimize this average.

The method of how to find it is computing gradient. For example, if we are lost in a smooth mountain at foggy night and want to go to the village in the valley. One way could be turning around and seeing which way the steepest way is to go down then take a small step down. The direction is (negative) gradient. With the assumption that the valley is convex, we could reach the valley.

The more efficient way is called Stochastic Gradient Descent (SGD). Since we want to minimize average loss over the training set, we take one sample or small group of samples and calculate the error, then use gradient descent. Then, we take a new sample and get a new value for the error, then get the gradient which is a different direction normally. Two of the main reasons for using SGD are that it helps a model to converge fast empirically if the training set is very large and it enables better generalization, which means getting similar performance on various sets of data.

Computing gradients by backpropagation

Computing gradients by backpropagation is a practical application of the chain rule. The backpropagation equation for the input gradients is as follows:

$$\frac{\partial C}{\partial \boldsymbol{x}_{i-1}} = \frac{\partial C}{\partial \boldsymbol{x}_i}\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{x}_{i-1}}$$
$$\frac{\partial C}{\partial \boldsymbol{x}_{i-1}} = \frac{\partial C}{\partial \boldsymbol{x}_i}\frac{\partial f_i(\boldsymbol{x}_{i-1}, \boldsymbol{w}_i)}{\partial \boldsymbol{x}_{i-1}}$$

The backpropagation equation for the weight gradients is as follows:

$$\frac{\partial C}{\partial \boldsymbol{w}_i} = \frac{\partial C}{\partial \boldsymbol{x}_i}\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{w}_i}$$
$$\frac{\partial C}{\partial \boldsymbol{w}_i} = \frac{\partial C}{\partial \boldsymbol{x}_i}\frac{\partial f_i(\boldsymbol{x}_{i-1}, \boldsymbol{w}_i)}{\partial \boldsymbol{w}_i}$$

Note that instead of scalar inputs, they will be vector inputs. More generally, multi-dimensional inputs. Backpropagation allows you to compute the derivative of the difference of the output you want and the output you get (which is the value of the objective function) with respect to any value inside the network. Finally, backpropagation is essential as it applies to multiple layers.

It is important to consider how to interpret inputs. For example, an image of 256×256 would require a 200,000 valued matrix. These would be huge matrices that the neural network layers will need to handle. It would be impractical to utilize such matrices. Therefore, it is important to make hypothesis of the structure of the matrix.

Hierarchical representation of the Visual Cortex

Experiments by Fukushima gave us an understanding of how our brain interprets the input to our eyes. In summary, it was discovered that neurons in front of our retina compress the input (known as contrast normalization) and the signal travels from our eyes to our brain. After this, the image gets processed in stages and certain neurons get activated for certain categories. Hence, the visual cortex does pattern recognition in a hierarchical manner.

Experiments in which researchers poked electrodes in specific areas of the visual cortex, specifically the V1 area made researchers realize that certain neurons react to motifs that appear in a very small area in a visual field and similarly with neighbouring neurons and neighbouring areas in the visual field. Additionally, neurons that react to the same visual field, react to different types of edges in an organized manner (*e.g.* vertical or horizontal edges). It is also important to note that there's also the idea that the visual process is essentially a feed forward process. Hence, somehow fast recognition can be done without some recurrent connections.

📝 *Yunya Wang, SunJoo Park, Mark Estudillo, Justin Mae*
27 Jan 2020