Response to Rumelhart et al. (1986)

A Literature Review Based on the Heilmeier Catechism

Matthew Evans

January 27, 2025

What are you trying to do?

In their paper, Rumelhart et al. (1986) introduce a technique to automatically learn and represent "useful new features" to improve the accuracy of neural networks.

How is it done today, and what are the limits of current practice?

The Perceptron Model

The paper mentions the perceptron model which has a single layer of inputs (often) directly connected to a single layer of outputs. Like other neural networks, the edges connecting inputs to outputs have weights which can be learned (in this case, by the "perceptron-convergence procedure"), with each weight indicating the impact of a given input feature node on a given output node. Perceptron models can be enhanced with so-called "feature analyzers" in which domain-specific knowledge is used to manually adjust the impact of a feature on the networks output.

Limitations of the Perceptron Model

The perceptron's weaknesses stem from its classification function $y = \text{sign}(W^T X + b)$ represents a hyperplane (i.e., a line). Its representational power is limited to tasks expressible as a linear combination of its inputs. Hence, if a dataset is not linearly separable, the perceptron will be unable to find a decision boundary in the data, and will fail to converge on a solution. Furthermore, the perceptron is particularly sensitive to noise and outliers (e.g., mislabeled data) polluting what would otherwise be a linearly separable dataset, preventing convergence and yielding poor generalization (high variance).

What is new in your approach and why do you think it will be successful?

Learning the Weights of Hidden Units

Rumelhart et al. (1986) build on the multi-layer perceptron model, providing a solution for learning (rather than manually fixing) the weights on connections between internal ("hidden") layers and the output layer. Specifically, their "back-propagation" approach identifies each hidden unit's contribution to the error in the output layer and facilitates adjusting the weights of hidden units, enabling convergence to a local local minimum in error function.

The paper describes a two-step process whereby outputs at each layer are computed (i.e., the forward-pass), and then the weights of connection are then adjusted according to the adjacent unit's contribution to the overall output error.

Forward Pass

The forward pass computes the output of each node j in all hidden layers and the output layer.

The input the node, x_j, is given by

$$x_j = \sum_i y_i w_{ji}$$

where y_i is output of unit i into unit j, and w_{ji} is the weight of the connection from unit i to unit j.

The output of the node, y_j, is given by

$$y_j = \frac{1}{1 + e^{-x_j}}$$

This output function is, notably, continuously differentiable and non-linear.

Backwards Pass

The backwards pass computes each unit's contribution to the overall error of the network.

The output layer's error given as

$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2$$

where j are the units of the penultimate layer and c are the training examples.

We use the chain rule to compute $\frac{\partial E}{\partial w_{ji}}$, each weight's contribution to the error, as follows. For some fixed data point c, for each unit j, we have

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j} \left(\frac{1}{2} \sum_j (y_j - d_j)^2 \right) \tag{1}$$

$$= y_j - d_j \tag{2}$$

Using the chain rule, we compute $\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial x_j}$

with $\frac{\partial y_j}{\partial x_j}$ given by

$$\frac{\partial y_j}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\frac{1}{1 + e^{-x_j}} \right) \tag{3}$$

$$= -(1 + e^{-x_j})^{-2}(-e^{x_j}) \tag{4}$$

$$= \frac{e^{x_j}}{(1 + e^{-x_j})^2} \tag{5}$$

$$= y_j(1 - y_j) \tag{6}$$

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial x_j} \tag{7}$$

$$= (y_j - d_j) y_j (1 - y_j) \tag{8}$$

Finally, we use the chain rule to compute $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}}$

with $\frac{\partial x_j}{\partial w_{ji}}$ given by

$$\frac{\partial x_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left(\sum_i y_i w_{ji} \right) \tag{9}$$

$$= y_i \tag{10}$$

Finally

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}} \tag{11}$$

$$= (y_j - d_j) y_j (1 - y_j) y_i \tag{12}$$

We now use the following update rule to adjust adjust each weight

$$\Delta W = -\varepsilon \frac{\partial E}{\partial w}$$

where ε is the learning rate, used to scale the update of w on each iterative step.

Who cares and what difference will it make?

The ability to compute the gradient of the error with respect to the weights of each internal edge, combined with the nonlinear sigmoidal output function enables a multi-layer neural network to perform classification tasks on nonlinear datasets without manually filtering (i.e., manipulating) edge weights. This improvement over the perceptron model enables learning complex patterns and relationships in data, which is particularly beneficial for fields such as image and speech recognition, where data is inherently nonlinear and high-dimensional.

The methods described in the paper have enabled advances in research and industry, particularly in the development of deep learning algorithms. These advancements have led to significant improvements in various applications, including image and speech recognition, natural language processing, and autonomous systems. By automating the learning of complex patterns in data, the techniques introduced by Rumelhart et al. (1986) have paved the way for more sophisticated and accurate models, driving innovation and progress in both academic research and practical implementations.

Costs, Benefits, and Impact

Costs

The technique described in the paper enables the construction of highly sophisticated models, yet with added sophistication comes the risk of over-fitting to training data. Additionally, the method described in this paper is inherently more complex to implement and requires additional compute to train.

Benefits

The use of back-propagation and hidden layers enables a network to identify non-obvious "regularities" in data, enabling the creation of learning models which can correctly solve highly complex tasks in non-linearly separable, high-dimensional spaces.

Impact

With the hindsight perspective of several decades since the paper's publication, we can see the enormous beneficial impact this technique has had in the research and development of neural networks, which have since been used to remarkable effect in a plethora of problem domains such as:

- **Image Recognition:** Convolutional neural networks (CNNs) have revolutionized the field of computer vision, enabling significant advancements in image classification, object detection, and image segmentation.
- **Speech Recognition:** Recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) have greatly improved the accuracy and efficiency of speech recognition systems, leading to widespread adoption in virtual assistants and transcription services.
- **Natural Language Processing:** Techniques such as word embeddings and transformers have transformed natural language processing tasks, including machine translation, sentiment analysis, and text generation.
- **Autonomous Systems:** Neural networks have been instrumental in the development of autonomous vehicles, enabling real-time decision-making and navigation in complex environments.
- **Healthcare:** Deep learning models have been applied to medical imaging, drug discovery, and personalized medicine, leading to improved diagnostic accuracy and treatment outcomes.

These advancements highlight the profound impact of the back-propagation algorithm introduced by Rumelhart et al. (1986), which has become a cornerstone of modern deep learning techniques.