16

Artificial intelligence

The human brain contains some 10^{12} cells. A typical neuron gets information from hundreds or thousands of other neurons, and sends information to hundreds or thousands of other neurons. The total number of connections between neurons is of the order of 10^{14}–10^{15}. Although there are many kinds of neurons, the typical neuron has three parts: a globular **cell body** that contains the cell's nucleus and most of its organelles, a tail-like structure called the **axon**, and several tree-like structures called **dendrites**. A neuron sends signals to other neurons along its axon which branches into hundreds or thousands of axon split ends that stop just before they touch the dendrites (or cell bodies) of other neurons. The thin gap between the split end of an axon and the dendrite is a **synapse**. A neuron's signal travels as an electrochemical wave until it reaches a synapse which it crosses as the diffusing molecules of a neurotransmitter.

16.1 A brief history of artificial intelligence

1842 Ada Lovelace: "The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform." The modern view of artificial intelligence is different.

1950 Alan Turing: the Turing test.

1960 Marvin Minsky: Steps toward artificial intelligence.

1961 James Slagle: A heuristic program that solves symbolic integration problems in freshman calculus: symbolic automatic integrator (SAINT)

1964–1966 Joseph Weizenbaum: Eliza a program that simulates a psychiatrist (tex-edit.com)

programs that can beat some intelligence tests

programs that can analyze forms and learn what an arch is

1975 Edward Shortliffe, Bruce Buchanan, and Stanley Cohen: Mycin, an expert system based on 600 rules that could recommend what antibiotic to give a patient. Was right 69% of the time, better than infectious-disease experts.

post 1975 Modern rule-based expert systems perform many useful functions, such as parking airplanes at airports.

1997 IBM's Deep Blue computer beat the reigning World Chess Champion, Garry Kasparov. Deep Blue epitomizes the bulldozer approach to artificial intelligence.

post 1997 Imagination, loops that tie thinking, perception, and action

16.2 Slagle's symbolic automatic integrator

James Slagle's Ph.D. thesis submitted to MIT's math department in 1961 contained a program that was capable of doing almost all of the indefinite integrals that appeared on MIT's final exams in calculus. And the program written in LISP ran on a computer with only 32 kB of memory!

Slagle's program is a superb example of a rule-based expert system. It first does problem reduction by applying all safe transformations:

1. $\int -f(x)\,dx = -\int f(x)\,dx$
2. $\int c\,f(x)\,dx = c\int f(x)\,dx$
3. $\int \sum_i f_i(x)\,dx = \sum_i \int f_i(x)\,dx$, which is an AND node.
4. If in the integral $\int (P(x)/Q(x))\,dx$ the degree of the polynomial $P(x)$ exceeds that of $Q(x)$, then divide.

The program then looks in its small table of 26 integrals to see if it's done.

If it's not done, it tries tricks such as these:

trig substitutions

$$\begin{aligned} f(\sin x, \cos x, \tan x, \csc x, \sec x, \cot x) &= g_1(\sin x, \cos x) \\ &= g_2(\tan x, \csc x) = g_3(\cot x, \sec x) \end{aligned} \tag{16.1}$$

set $y = \tan x$

$$\int f(\tan x)\,dx = \int \frac{f(y)}{1+y^2}\,dy \tag{16.2}$$

set $x = \sin y$

$$\int f(1-x^2)\,dx = \int f(\cos^2 y)\,\cos y\,dy \tag{16.3}$$

set $x = \tan y$

$$\int f(1+x^2)\,dx = \int f(\sec^2 y)\,(1+\tan^2 y)\,dy. \tag{16.4}$$

We apply Slagle's program to the integral

$$\int \frac{-5x^4}{(1-x^2)^{5/2}}\,dx = -\int \frac{5x^4}{(1-x^2)^{5/2}}\,dx = -5\int \frac{x^4}{(1-x^2)^{5/2}}\,dx. \tag{16.5}$$

After making these safe transformations, we set $x = \sin y$ and get

$$\int \frac{x^4}{(1-x^2)^{5/2}}\,dx = \int \frac{\sin^4 y}{\cos^5 y}\,\cos y\,dy = \int \frac{\sin^4 y}{\cos^4 y}\,dy. \tag{16.6}$$

We now transform this to

$$\int \tan^4 y\,dy \quad \text{or} \quad \int \frac{1}{\cot^4 y}\,dy \tag{16.7}$$

which is an OR node. The program is an and/or tree or a goal tree. We set $z = \tan y$ and get

$$\int \tan^4 y\,dy = \int \frac{z^4}{1+z^2}\,dz. \tag{16.8}$$

Because the degree of the numerator exceeds that of the denominator, we follow rule 4 and divide:

$$\int \frac{z^4}{1+z^2}\,dz = \int \left(z^2 - 1 + \frac{1}{z^2+1}\right)\,dz. \tag{16.9}$$

We now use the sum rule (3) and do each integral separately:

$$\begin{aligned} I_1 &= \int z^2\,dz = \frac{z^3}{3} \\ I_2 &= \int -1\,dz = -z \\ I_3 &= \int \frac{dz}{z^2+1} = \arctan z. \end{aligned} \tag{16.10}$$

Finally since $z = \tan y = \tan(\arcsin x)$, we have

$$\int \frac{-5x^4}{(1-x^2)^{5/2}}\,dx = -\frac{5}{3}\tan^3(\arcsin x) + 5\tan(\arcsin x) - 5\arcsin x. \tag{16.11}$$

16.3 Neural networks

As of this writing, early 2018, most things that a human can do in one second can be done by artificial intelligence. The main technique is the training and use of neural networks. The more data one has, the better one can train a neural network. And the more data one has, the more neurons one needs to optimally use the training data. So the trend is toward huge data sets and large neural networks.

A single neuron takes input signals a_i from other neurons $i = 1, \ldots, n$, giving weight w_i to signal a_i. It fires if the sum of the n weighted inputs $w_i a_i$ exceeds its bias b. The signal the neuron sends out is then

$$a = \tfrac{1}{2}\left(w_1 a_1 + \cdots + w_n a_n - b + |w_1 a_1 + \cdots + w_n a_n - b|\right) \tag{16.12}$$

which often is written as

$$a = \mathrm{ReLU}\left(w_1 a_1 + \cdots + w_n a_n - b\right). \tag{16.13}$$

Computer scientists used to use the more complicated sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$, but they have found that the simpler function $\mathrm{ReLU}(x) = (x + |x|)/2$ works just as well.

The neurons of a typical network are organized into layers $\ell = 1, \ldots, m$ of neurons with n_ℓ neurons labeled by the index $j = 1, \ldots, n_\ell$. In this notation, the signal emitted by the jth neuron of layer ℓ is

$$a_j^\ell = \mathrm{ReLU}\left(w_{j1}^\ell a_1^{\ell-1} + \cdots + w_{jn_{\ell-1}}^\ell a_{n_{\ell-1}}^{\ell-1} - b_j^\ell\right). \tag{16.14}$$

If there are m layers of neurons, then the prediction of the network is the n_m nonnegative numbers a_j^m. If the **task** of the network is to classify vectors x into C categories, then the probability the network assigns to vector x's being in the ith category is

$$p(i|x) = \frac{a_i^m}{\sum_{k=1}^C a_k^m}. \tag{16.15}$$

For example, a vector x might represent the darkness of the pixels of an image of the number 2 handwritten on a white sheet of paper. A perfect network would give $p(i|x) = \delta_{i2}$.

If the neural network assigns probability $0 \le p(i|x) \le 1$ to image x's belonging to the ith category while the correct category is $\ell(x)$ then the squared error made by the neural network is

$$E^2(x) = \sum_{i=1}^C |p(i|x) - \delta_{i\ell(x)}|^2 \tag{16.16}$$

summed over the C categories. The squared error made by the neural network on N images $\{x\} = \{x_1, x_2, \ldots, x_N\}$ would be

$$E^2(\{x\}, \{v\}) = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{C} |p(i|x_k) - \delta_{i\ell(x_k)}|^2. \tag{16.17}$$

One **trains** a neural network by adjusting its parameters w_{jk}^{ℓ} and b_j^{ℓ} so as to lower its error $E(\{x\})$. If the network has m layers of n neurons, then the w_{jk}^{ℓ} and the b_j^{ℓ} constitute $M = mn(n+1)$ adjustable parameters, or 3,003,000 if $n = 10^3$ and $m = 3$. We can number the parameters w_{jk}^{ℓ} and b_j^{ℓ} with a single index ℓ, setting $v_1 = w_{11}^1$, $v_2 = w_{12}^1$, $\ldots$, $v_M = b_n^m$. The error of the network will depend upon these parameters, so we should write it as $E(\{x\}, \{v\})$.

One can use the Monte Carlo method (section 15.6) of simulated annealing to find the parameters $\{v\}$ that minimize the error $E(\{x\}, \{v\})$.

Another procedure is to compute the partial derivatives of the squared error $E^2(\{x\}, \{v\})$ with respect to the parameters $\{v\}$ so as to form its gradient

$$\nabla E^2(\{x\}, \{v\}) = \left(\frac{\partial E^2(\{x\}, \{v\})}{\partial v_1}, \ldots, \frac{\partial E^2(\{x\}, \{v\})}{\partial v_M} \right). \tag{16.18}$$

One then changes the parameters $\{v\}$ by a suitably small negative multiple $-\epsilon$ of the gradient $\nabla E^2(\{x\}, \{v\})$

$$v_i' = v_i - \epsilon \, \frac{\partial E^2(\{x\}, \{v\})}{\partial v_i}. \tag{16.19}$$

16.4 A linear unbiased neural network

If we simplify our neural network by replacing the function $\mathrm{ReLU}(x)$ by x and setting all the biases to zero, then the most elaborate neural network reduces to a linear map, $y = A\,x$, in which the real matrix A maps an unknown vector x into a category y.

Suppose X is a matrix that represents the training set of vectors, so that the ith element of its kth column is the ith element of the kth training vector $x^{(k)}$, that is, $X_{ik} = x_i^{(k)}$. Let T be the matrix of correct assignments of the training vectors X. That is, T_{ik} is the correct assignment of the training vector $x^{(k)}$. Ideally, we then should like to have $A\,X = T$. If the training matrix X were a square nonsingular matrix, we could set $A = T\,X^{-1}$, but if we have lots of training vectors, more than there are categories to which we seek to assign them, then X has more columns than rows. The rows of X

typically are long and linearly independent. The matrix $X\,X^{\mathsf{T}}$ then has an inverse, and we may use the form (1.433) of the Moore-Penrose pseudomatrix $X^+ = X^{\mathsf{T}}\,(X X^{\mathsf{T}})^{-1}$. Our best guess for the matrix A then is

$$A = T\,X^+ = T\,X^{\mathsf{T}}\,(X X^{\mathsf{T}})^{-1}. \tag{16.20}$$

This method (16.20) does not work, however, when the matrix $X X^{\mathsf{T}}$ is singular as in the MNIST database of handwritten numbers.

Example 16.1 (Reading handwritten numbers) The MNIST website `http://yann.lecun.com/exdb/mnist/` lists four files that one can use to train and test a neural network. The gzipped file train-images-idx3-ubyte.gz contains 60,000 images $x(i)$ of handwritten numbers. Each image $x(i)$ is a real 28-by-28 matrix, which is equivalent to a real vector in a space of 784 dimensions. The file train-labels-idx1-ubyte.gz contains the 60,000 labels that the 60,000 handwritten numbers of the train-images file represent. The files t10k-images-idx3-ubyte.gz and t10k-labels-idx1-ubyte.gz are similar files of 10,000 different handwritten integers and their labels. Unfortunately, these files are in high-endian format, so people using Intel processors must translate these files into low-endian format.

We seek a matrix A_{ik} with 10 rows, $i = 0, \ldots, 9$, and 784 columns, $k = 1, \ldots, 784$. The singular-value decomposition of this matrix should be of the form

$$A = \sum_{\ell=0}^{9} |\ell\rangle\langle\bar{\ell}| \tag{16.21}$$

in which the ith element of the vector $|\ell\rangle$ is $\delta_{i\ell}$, and the vector $|\bar{\ell}\rangle$ is the normalized sum of all the 60,000 training vectors $x(i,\ell)$ that represent the integer ℓ. We make it in two steps

$$|\tilde{\ell}\rangle = \sum_{i=1}^{60{,}000} |x(i,\ell)\rangle \qquad \text{and} \qquad |\bar{\ell}\rangle = \frac{|\tilde{\ell}\rangle}{\sqrt{\langle\tilde{\ell}|\tilde{\ell}\rangle}}. \tag{16.22}$$

This unbiased linear neural network correctly identifies 82.16% of the handwritten test images of the MNIST website. □

Further reading

- A link (`http://hubel.med.harvard.edu/book/bcontex.htm`) to David Hubel's book *Eye, Brain, and Vision*.
- A link (`https://playground.tensorflow.org/`) good website on neural networks. You can play with them there.
- A link (`https://www.tensorflow.org`) to the TensorFlow website.

- A link (`http://neuralnetworksanddeeplearning.com/index.html`) to Michael Nielsen's book, *Neural Networks and Deep Learning*.
- A link (`http://www.deeplearningbook.org`) to *Deep Learning* by Ian Goodfellow and Yoshua Bengio and Aaron Courville.
- Kevin Murphy's book *Machine Learning: A Probabilistic Perspective* is available at `http://b-ok.org`.
- *The Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani, and Jerome Friedman is available at `http://b-ok.xyz/book/659984/d8d61d`.
- Andrew Ng's course Machine Learning is at `https://www.coursera.org/learn/machine-learning`.
- The excellent lectures of MIT Professor Patrick Winston are available online (`https://ocw.mit.edu/6-034F10`) and were used in sections 16.1 and 16.2.
- The website `https://dspace.mit.edu/handle/1721.1/11997` makes James Slagle's MIT PhD thesis available.