It is Just a Machine that Learns

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

Center for Humanities Computing
Aarhus University, Denmark

April 30, 2019

Outline

Elon Musk
"With Artificial Intelligence, we are summoning the demon"

Andrew Ng
"Fearing a rise of killer robots is like worrying about overpopulation on Mars"

Jeff Hinton
"Whether or not it turns out to be a good thing depends entirely on the social system, and doesn't depend at all on the technology"

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

OpenAI's transformer-based model

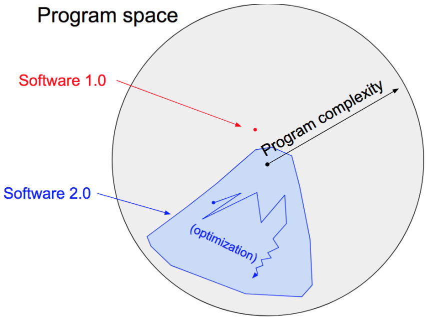
OpenAI on GPT-2

"We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization—all without task-specific training."

"Due to concerns about large language models being used to generate deceptive, biased, or abusive language at scale, we are only releasing a much smaller version of GPT-2 along with sampling code. We are not releasing the dataset, training code, or GPT-2 model weights."

- **PR Focus** - reporters were given early information
- **Gatekeeping** - malicious uses were hypothesized and we have no way of testing
- **Misdirected** - not releasing affects researchers more than malicious actors due to the model price
- **Dual use** - OpenAI did not discuss dual-use technology

AI from the perspective of software development



Just a machine that learns

Machine learning emerged from AI - **build a computer system that automatically improves with experience**

- application is too complex for a manually designed algorithm
- application needs to customize its operational environment after it is fielded

A well-posed learning problem

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E

Historically, ML is "just" part of the **industrial age's efforts towards perfecting task automation**

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

Humanities research meets machine learning

As a consequence of the data surge, we are (also) "jumping the automation bandwagon"

– plus theoretical innovations that rely on ML/DL (e.g., lexical $\rightarrow$ compositional semantics)

Inherent challenges in our data and users

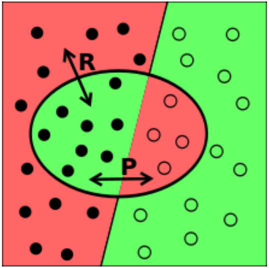
– data are unstructured, heterogeneous, need normalization, low resource varieties

– users lack of computational literacy, ++gab between technology and domain knowledge

Types of problems solved by ML:

– initially ML was the solution to a(-ny) research problem

– increasingly, ML solves auxiliary tasks related to automation



← `relevant objects` (e.g., ham)

→ `irrelevant objects` (e.g., spam)

⬭ objects classified with relevant class label

ERROR

CORRECT

Precision: fraction of retrieved instances that are relevant

$$P = \frac{TP}{TP + FP} \tag{1}$$

Recall: fraction of relevant instances that are retrieved

$$R = \frac{TP}{TP + FN} \tag{2}$$

P and *R* are inversely related. Identify balance through a Precision-Recall curve.

Impossibility results

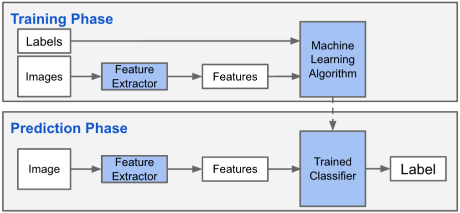
Assume differing base rates, $Pr_a(Y = 1) \neq Pr_b(Y = 1)$, and an imperfect learning algorithm, $C \neq Y$, then you cannot simultaneously achieve:

- **Precision parity**: $Pr_a(Y = 1 \mid C = 1) = Pr_b(Y = 1 \mid C = 1)$
- **True positive parity**: $Pr_a(C = 1 \mid Y = 1) = Pr_b(C = 1 \mid Y = 1)$
- **False positive parity**: $Pr_a(C = 1 \mid Y = 0) = Pr_b(C = 1 \mid Y = 0)$

"Suppose we want to determine the risk that a person is a carrier for a disease Y, and suppose that a higher fraction of women than men are carriers. Then our results imply that in any test designed to estimate the probability that someone is a carrier of Y, at least one of the following undesirable properties must hold: (a) the test's probability estimates are systematically skewed upward or downward for at least one gender; or (b) the test assigns a higher average risk estimate to healthy people (non-carriers) in one gender than the other; or (c) the test assigns a higher average risk estimate to carriers of the disease in one gender than the other. The point is that this trade-off among (a), (b), and (c) is not a fact about medicine; it is simply a fact about risk estimates when the base rates differ between two groups"

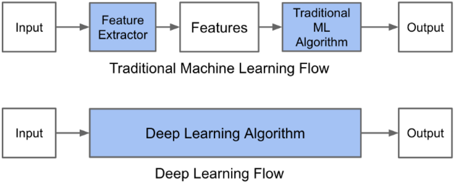
Kleinberg J., S. Mullainathan, & M. Raghavan (2016), Inherent Trade-Offs in the Fair Determination of Risk Scores, arXiv:1609.05807

Basic supervised pipeline



Machine Learning Phases

The emergence of deep learning



Neurons

Basic computational unit of a neural network

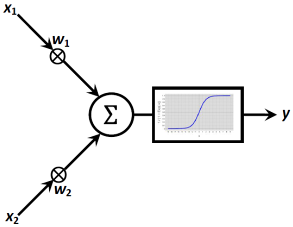


Figure 1: A neuron takes inputs, x_1, x_2, does *some math on them*, and generates an output, y

The input is weighted

$$x_1 \to x_1 \times w_1$$
$$x_2 \to x_2 \times w_2$$

then added with a bias

$$(x_1 \times w_1) + (x_2 \times w_2) + b$$

and finally passed through an activation function

$$y = f(x_1 \times w_1 + x_2 \times w_2 + b)$$

A word on the activation functions

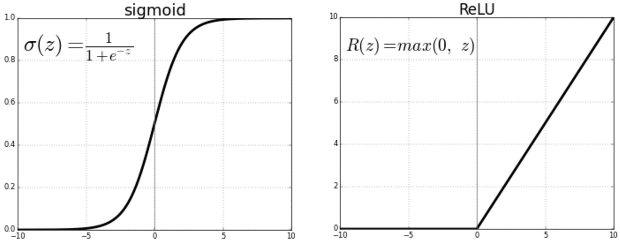


Figure 2: The sigmoid activation function "squashes" an unbounded $(-\infty, +\infty)$ to a bounded $(0, 1)$ set. Computationally simpler activation functions, such as rectifiers, are starting to replace sigmoids.

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

Example

cat/dog classifier where x_1 "has fur" and x_2 "barks" and we are generally more likely to encounter dogs, so when "it has fur and barks", then:

$$
\begin{aligned}
w &= [0, 1] \\
b &= 2
\end{aligned}
$$

$$
\begin{aligned}
(w \cdot x) + b &= ((w_1 \times x_1) + (w_2 \times x_2)) + b \\
&= 1 \times 0 + 1 \times 1 + 2 \\
&= 3
\end{aligned}
$$

$$f(w \cdot x + b) = f(3) = \frac{1}{1 + e^{-}3} = 0.953$$

Neurons in a network

An artificial neural network is just a set of neurons wired together (typically) in a layered structure.

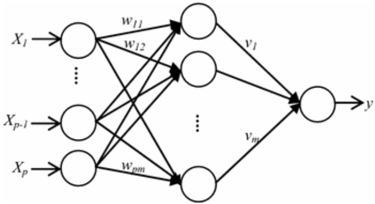


Figure 3: Feedforward neural network with one hidden layer of size *m*. A hidden layer is any layer between the input and output. Hidden layers perform transformations on the input or previous hidden layers. A network can have many hidden layers.

A neural network can have any number of neurons and layers. *Deep* in deep learning just refers to representations learned in multi-layered (deep) structures. The core idea is to propagate input forward through the transformations of the hidden layers in order to get an output.

Example

continue example from before (cat/dog), with one hidden layer and two hidden units, $w = [0, 1]$, $b = 0$, and $x = [0, 1]$:

$$\begin{aligned} h_1 = h_2 &= f(w \cdot x + b) \\ &= f((0 \times 0) + (1 \times 1) + 0) \\ &= f(1) \\ &= 0.731 \end{aligned}$$

$$\begin{aligned} o_1 &= f(w \cdot [h_1, h_2] + b) \\ &= f((0 \times h_1) + (1 \times h_2) + 0) \\ &= f(0.731) \\ &= 0.675 \end{aligned}$$

Training the model

It is impossible to compute the perfect weights for a neural network. Instead learning becomes an optimization problem and algorithms are used to run through the space of possible weights that the model can use to make a good prediction.

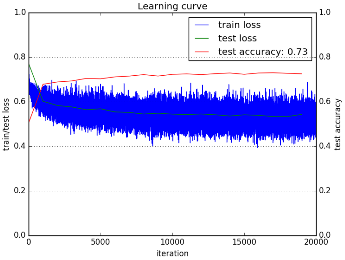


Figure 4: Training is an optimization problem: minimizing loss function & maximize test accuracy

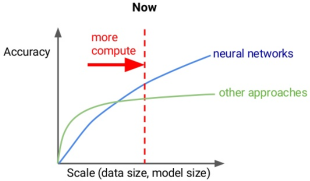


Figure 5: Currently there seems to be no upper limit on performance - except for the perfect classifier

– Training consists of iteratively adjusting the weights in order to minimize a loss function.

– Neural network models are typically trained using the *gradient descent* optimization algorithm and weights are updated using the backpropagation (of error) algorithm

Loss function

Mean squared error loss:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

– a good prediction lowers loss $\rightarrow$ training a network $\sim$ trying to mininmize loss

– iow: a loss function maps the networks output onto the "loss" associated with a prediction $\sim$ evaluated how well the neural network captures the data structure

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

If the goal is to minimize loss of the network, the loss is a function of weights w and biases b. For a fully connected one-layered feedforward network $(2 \times 2 \times 1)$ then:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

Modifying w_1 then, will change L as $\frac{\partial L}{\partial w_1}$. Using the chain rule:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} \times \frac{\partial y_{pred}}{\partial w_1}$$

Assume a simple binary classifier, $True : 1$, $MSE = (1 - y_{pred})^2$, then:

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

For $\frac{\partial y_{pred}}{w_1}$, let h_1, h_2, o_1 be the output of the neurons they represent, then:

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

where f is the sigmoid activation function.

Because w_1 only modulates h_1 and not h_2:

$$\frac{\partial y_{pred}}{w_1} = \frac{\partial y_{pred}}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

and with the chain rule:

$$\frac{\partial y_{pred}}{\partial h_1} = w_5 \times f'(w_5 h_1 + w_6 h_2 + b_3)$$

Repeat procedure for $\frac{\partial h_1}{\partial w_1}$:

$$h_1 = f(w_1 x_1 + w_2 x_2 + b1)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 \times f'(w_1 x_1 + w_2 x_2 + b1)$$

Compute the derivative of the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) \times (1 - f(x))$$

Put it all together and we can compute:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} \times \frac{\partial y_{pred}}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

as:

$$-2(1 - y_{pred}) \times w_5 \times f'(w_5 h_1 + w_6 h_2 + b_3) \times x_1 \times f'(w_1 x_1 + w_2 x_2 + b1)$$

Backprobagation: The system of computing the partial derivatives by working backwards. Backpropagation in this form was derived by Stuart Dreyfus in 1962.

Dreyfus, S (1962). The numerical solution of variational problems. Journal of Mathematical Analysis and Applications. 5(1)

Training with Backprop

The most widely used training algorithm is *Stochastic Gradient Descent*, which is a set of formal steps for modifying weights and biases to minimize loss:

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

where the learning η rate controls the speed of training

- if $\frac{\partial L}{\partial w_1}$ is positive, then w_1 will decrease and L decrease
- if $\frac{\partial L}{\partial w_1}$ is negative, then w_1 will increase and L decrease

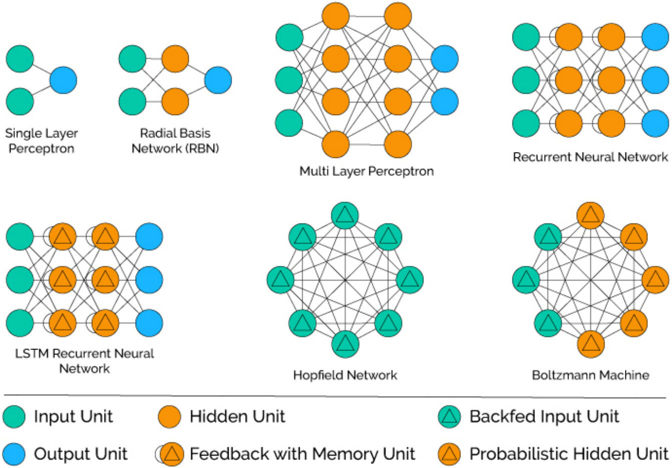
Algorithm 1 Gradient Descent

while $t < maxiter$ **do**
  for *all* i, j **do**
    $w_{ij} = w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$
  end for
end while

Underlying AI is just rather "dumb" system that improves its performance on a pre-specified task over time by **recursively sending the output of its computations backwards to the parent**.

Kristoffer L Nielbo
kln@cas.dk
knielbo.github.io

ANN architectures



TAK

kln@au.dk
knielbo.github.io

slides: http://knielbo.github.io/files/kln_fip19.pdf

& tak til

Mads Rosendahl Thomsen, Comparative Literature, School of Communication and Culture, Aarhus University, DK
Jianbo Gao and Bin Liu, Institute of Complexity Science and Big Data, Guangxi University, CHN
Culture Analytics @ Institute of Pure and Applied Mathematics, UCLA, US