Rudiments of Neural Cryptography

Francesco Moraglio

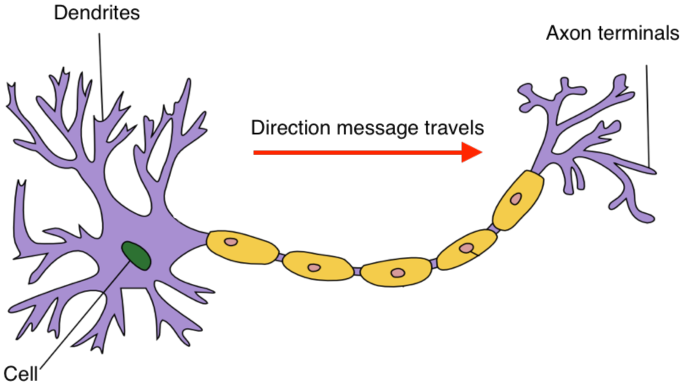
University of Torino

21/07/2020

Overview

Neural Networks

Artificial Neural Networks (ANNs) are models of computation based loosely on the way in which the brain is believed to work. A biological neural network consists of interconnected nerve cells, whose bodies are where neural processing takes place.



Artificial Neural Networks

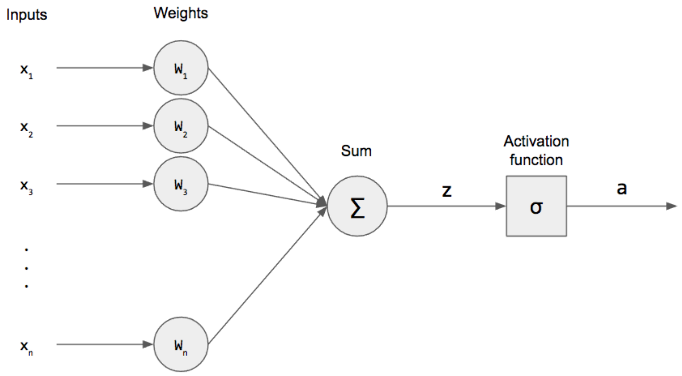
Interconnections between cells are not all equally weighted: this is the key feature modeled by ANNs. Their theoretical principles were firstly formulated in the Forties. Among them, a fundamental, widely applied learning principle is the following.

Hebbian Learning Law

When an axon of cell A is near-enough to excite cell B and when it repeatedly and persistently takes part in firing it, then some growth process or metabolic change takes place in one or both these cells such that the efficiency of cell A is increased.

Perceptron: 1

The first complete neural model is called Perceptron and appeared in the late Fifties. It serves as a building block to most later models.



Perceptron: 2

The input/output relations of the Perceptron are defined to be

$$z = \sum_i w_i x_i \quad \text{(summation output)}$$
$$y = f_N(z) \quad \text{(cell output)},$$

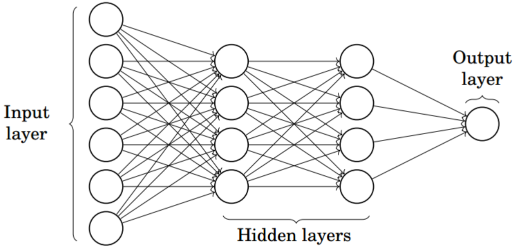
where w_i is the (adjustable) weight at input x_i. Function f_N is nonlinear and it is called *activation*. Typical activation functions used in ANNs include

- sigmoid function;
- hyperbolic tangent;
- Heaviside step function.

Training

The training of an ANN is the procedure of adjusting its weights. This task can be performed with several techniques. For the simplest architectures, we recall

- **Least Mean Square**. Minimizes the (approximated) square expectation of the error over all training sets. It is simple, but it is suitable only for bipolar perceptrons and may lead to inaccurate results.
- **Gradient Descent**. Attempts to provide weight-setting estimates from one training set to the next.



More advanced algorithms were required to set weights of Multi-Layer Perceptrons (MLPs), whose computational capabilities are sensitively higher than those of single-layer NNs.

- **Back Propagation (BP)** was developed with this specific purpose. It is an extension of the Gradient Descent method, that "propagates" from the last layer back to the first one, iterating over a pre-fixed set of training vectors.

Adam

Adam, a recent method for stochastic optimization, revealed itself to be a powerful tool for the training of most modern NN architectures. Let $f(\theta)$ be a noisy objective function; we are interested in minimizing $\mathbb{E}[f(\theta)]$.

- Consider $g_t = \nabla_\theta f_t(\theta)$, the gradient evaluated at time step t.
- Adam works by updating exponential moving averages of the gradient and of the squared gradient.
- Exponential decay rates of such moving averages are controlled by hyperparameters.

The name "Adam" is derived from "adaptive moment estimation".

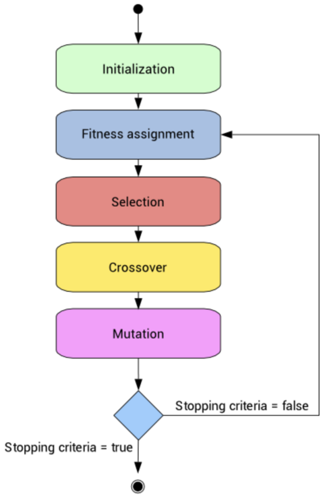
```
**Require:** $\alpha$: Stepsize;
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates;
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$;
**Require:** $\theta_0$: Initial parameter vector;
$m_0 \leftarrow 0$ (Initialize first moment vector);
$v_0 \leftarrow 0$ (Initialize second moment vector);
$t \leftarrow 0$ (Initialize time step);
**while** $\theta_t$ *not converged* **do**
    $t \leftarrow t + 1$;
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients);
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Biased first moment est.);
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Biased second moment est.);
    $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ (Bias-corrected first moment est.);
    $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ (Bias-corrected second moment est.);
    $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ (Update parameters);
**return** $\theta_t$
```

Genetic Algorithms: 1

Genetic Algorithms (GAs) were invented by John Holland in the 1960s and can be considered a key building block of artificial intelligence. These are stochastic search algorithms that can generate "sufficiently good" solutions to an optimization problem. Nowdays GAs are widely employed in applied science and, in neural cryptography, they can be used to

- optimize network architectures involved in communication;
- perform effective cryptanalitic attacks.

Genetic Algorithms: 2



- The GA is a method for evolving from a population of "chromosomes" (elements of a solution space) to a new population by using an imitation of natural selection together with the genetic-inspired operators of crossover and mutation.
- The main theoretical result on the behavior of GAs is known as Holland Theorem. It implies that the best "building blocks" of the solutions propagate exponentially over time in the population.

Genetic Algorithms: 3

- The genome of the evolutionary model is represented by a size N multiset of bit strings:
$$\mathbf{P}(t) = \{\vec{v}_1, \ldots, \vec{v}_N\},$$
where $t \in \mathbb{N}$ denotes the t-th generation.
- Selection is based on a fitness function:
$$f : \mathbf{P}(t) \longrightarrow \mathbb{R}_0^+.$$
- Total fitness is defined to be $F = \sum_{\vec{v} \in \mathbf{P}(t)} f(\vec{v})$; average fitness is $\overline{F} = \frac{F}{N}$.
- For a given chromosome $\vec{v}$, its selection probability is given by
$$P_S(\vec{v}) = \frac{f(v)}{F}.$$

Genetic Algorithms: Holland Theorem

- Let $P_C \in (0,1)$ be the crossover probability and $P_M \in (0,1)$ the mutation probability.
- A schema H is a set of bit strings that can be described as a template made up of symbols 0, 1 and $*$ (free entry). Denote by $o(H)$ the order of H, that is its number of defined bits ($\neq *$) and we let $d(H)$ be the length of the scheme (the distance between its outermost defined bits).

Theorem

Let H be a schema with at least one instance present in $\mathbf{P}(t)$ and let $m(H,t)$ the number of instances of H at time t. Moreover let $\hat{U}(H,t)$ be the observed average fitness. Then $\mathbb{E}\left[m(H,t+1)\right] \geq \frac{\hat{U}(H,t)}{\overline{F}} m(H,t) \left(1 - P_C\left(\frac{d(H)}{n-1}\right)\right)\left[(1-P_M)^{o(H)}\right]$.

Neural Cryptography

- From the Nineties on, researchers have made attempts at combining the features of neural networks with cryptography: this field is commonly known as neurocryptography.
- One of the first attempts can be found in [Volná(2000)]. In this work, the author builds a symmetric cryptosystem by making use of neural networks, whose parameters constitute the key of the cipher.
- GAs are used for the optimization of the designed NN topology. Adaptation of the best found network architecture is then finished with BP.

The NNs of Volná

- The GA of Volná evolves the architecture of feedforward NNs whose weights are initialized as follows. For every existing connection, three digits are generated and weights are computed as

$$\begin{aligned} w_{ij,kl} = \eta[e_2(e_1 2^1 + e_0 2^0)]; \quad & i,k \in \{1,\dots,L\}, \\ & j \in \{1,\dots,n_i\}, \\ & l \in \{1,\dots,n_k\}, \end{aligned}$$

  where $w_{ij,kl} = w(x_{ij}, x_{kl})$ is the weight value between the j-th unit in the i-th layer and the l-th unit in the k-th layer and

$$\begin{aligned} \eta &= \text{learning parameter}; \quad \eta \in (0,1) \\ e_0, e_1 &= \text{random digits} \\ e_2 &= \text{sign bit.} \end{aligned}$$

- L is the total number of layers, while n_i and n_k are the number of units in layers i and k.

- On the basis of the error between desired and real output, the algorithm computes the fitness precursor value $f_i^\star$, for each individual $i = 1, \ldots, N$, that is

$$f_i^\star = k_1(E_i)^2 + k_2(U_i)^2 + k_3(L_i)^2,$$

  where k_j, $j = 1, 2, 3$ are fixed constants and

$$\begin{aligned} E_i &= \text{error for network } i \\ U_i &= \text{number of hidden units} \\ L_i &= \text{number of hidden layers.} \end{aligned}$$

- The general fitness function f is then calculated as follows:

$$f_i = \begin{cases} k - (f_i^\star + k_5) & \text{if} \quad E_i > k_4 \\ k - f_i^\star & \text{otherwise.} \end{cases}$$

- k, k_4 and k_5 also denote constants.

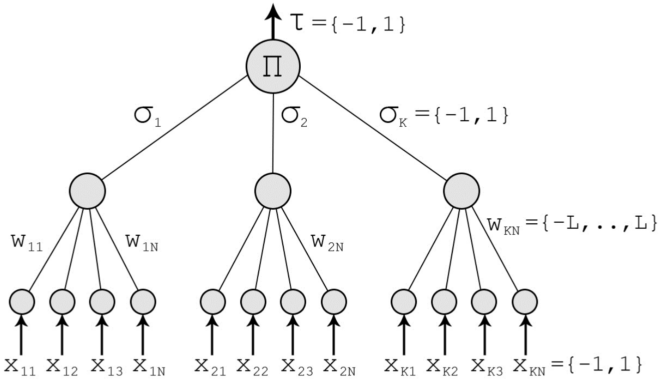
Training set

Plaintext			Cyphertext
Char	*ASCII Code*	*Bit String Representation*	*Bit String Representation*
a	97	00001	000010
b	98	00010	100110
c	99	00011	001011
d	100	00100	011010
e	101	00101	100000
f	102	00110	001110
g	103	00111	100101
h	104	01000	010010
i	105	01001	001000
j	106	01010	011110

- The ciphertext is a randomly generated chain of bits.

KKK Key Exchange Protocol

The first complete cryptosystem based on neural network is known in literature as KKK, from the surnames of its inventors [Kanter, Kinzel and Kanter(2001)].
This protocol is based on the synchronization of the weights of two tree parity machines, that represent the participants.


$\tau = \{-1, 1\}$
Π
σ_1
σ_2
$\sigma_K = \{-1, 1\}$
w_{11}
w_{1N}
w_{2N}
$w_{KN} = \{-L, .., L\}$
x_{11} x_{12} x_{13} x_{1N} x_{21} x_{22} x_{23} x_{2N} x_{K1} x_{K2} x_{K3} $x_{KN} = \{-1, 1\}$

- The two NNs participating in the communication start from private key vectors $E_k(0)$ and $D_k(0)$. Mutual learning from the exchange of public information leads the two nets to develop a common, time dependent key: $E_k(t) = -D_k(t)$. This is then used for both encryption and decryption.
- At each step of the training process (and of encryption/decryption), a common public input vector is needed.
- Sender and recipient send their outputs to each other and in case they do not agree on them, weights are updated according to a Hebbian learning rule.
- As soon as the two NNs are synchronized, so they stay forever.

KKK: Shamir's Insights

In the paper cited before, no mathematical proof of its core principle, synchronization, is given. This result was instead attained in [Klimov, Mityagin and Shamir(2002)]. Such work also contains a section dedicated to the cryptanalysis of KKK. Besides it is robust against attacks based on intercepting the key using the same neural network structure, this protocol can be broken using

- Genetic Attacks;
- Geometric Attacks;
- Probabilistic Attacks.

These results marked the beginning of a long period without any substantial contribution to neural cryptography.

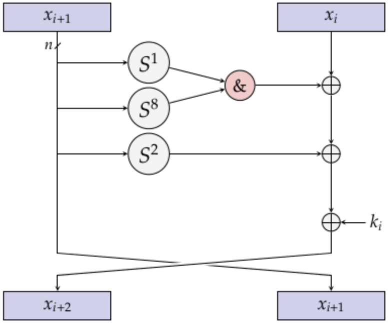
Neural Cryptanalysis

Not only NNs reveal themselves capable of learning how to communicate securely: recent studies show they can be employed to perform efficient cryptanalysis. Consider the family of block ciphers released by NSA in [Beaulieu et al.(2013)], namely

- **Simon**, a cipher efficient in hardware implementations in IoT (Internet of Things) devices;
- **Speck**, the sister algorithm of Simon, optimized for software implementations.

Both algorithms are compositions of the basic functions of modular addition, bitwise rotation and bitwise addition.

Simon



- The attack to Simon cipher consists in recovering the key, given a set of plaintext-ciphertext pairs.
- MLPs are employed; input layer has one neuron per each bit of the plaintext-ciphertext pairs. Output layer has same size of the key; each cell is binary.
- Activation function is chosen to be the Linear Rectifier:

$$y = \sum_j w_j x_j + b.$$

Speck

A more advanced attack was developed against Speck, by making use of convolutional NNs to build a (neural) distinguisher.

- Let $F : \{0,1\}^n \longrightarrow \{0,1\}^m$ be a map (round function). A differential transition for F is a pair

$$(\Delta_{\text{in}}, \Delta_{\text{out}}) \in \{0,1\}^n \times \{0,1\}^m$$

- A differential attack uses nonrandom properties of the output of the cipher when it is being given input data with known difference distribution, i. e. $\mathbb{P}(\Delta_{\text{in}} \to \Delta_{\text{out}})$.
- Nets are trained to distinguish the output of Speck with given input conditions from random data.

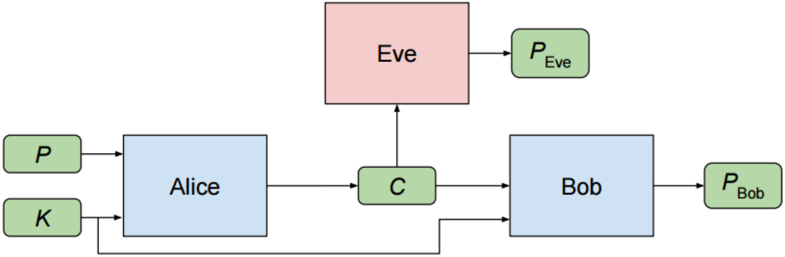
Adversarial Neural Cryptography

Neural networks can learn to protect the secrecy of their data from other neural networks. This is the core principle of Adversarial Neural Cryptography (ANC), a revolutionary approach to cryptography invented by Google researchers in 2016. Such technique is based on

- the problem of the secure communication between two parties, when a third participant wishes to eavesdrop on it;
- the adversarial training of NNs representing the three participants.

This scenario can be summarized as follows.

- Alice ($\mathcal{A}$) wants to sent message P to Bob ($\mathcal{B}$). P and key K are the input of Alice.
- K is also known by Bob and it is used to decipher C, that is the output of Alice.
- Eve ($\mathcal{E}$) wants to eavesdrop on the communication. She knows only C and tries to recover P.



ANC: Method

- NNs have parameters we denote by $\theta_{\mathcal{A}}$, $\theta_{\mathcal{B}}$ and $\theta_{\mathcal{E}}$.
- Denote the outputs of Alice and Bob as $A(\theta_{\mathcal{A}}, P, K)$ and $B(\theta_{\mathcal{B}}, C, K)$, respectively. Write $E(\theta_{\mathcal{E}}, C)$ for the output of Eve.
- The loss function of Eve is the following.

$$L_{\mathcal{E}}\left(\theta_{\mathcal{A}}, \theta_{\mathcal{E}}\right) = \mathbb{E}_{P,K}\left[d\left(P, E(\theta_{\mathcal{E}}, A(\theta_{\mathcal{A}}, P, K))\right)\right]$$

- Similarly, for Bob we have

$$L_{\mathcal{B}}\left(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}\right) = \mathbb{E}_{P,K}\left[d\left(P, B(\theta_{\mathcal{B}}, A(\theta_{\mathcal{A}}, P, K), K)\right)\right].$$

- The joint loss function of Alice and Bob takes into account the loss of $\mathcal{B}$ and the value of loss of optimal Eve $\left(O_{\mathcal{E}}\left(\theta_{\mathcal{A}}\right)\right)$:

$$L_{\mathcal{AB}}(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}) = L_{\mathcal{B}}\left(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}\right) - L_{\mathcal{E}}(\theta_{\mathcal{A}}, O_{\mathcal{E}}\left(\theta_{\mathcal{A}}\right)).$$

Network Architecture

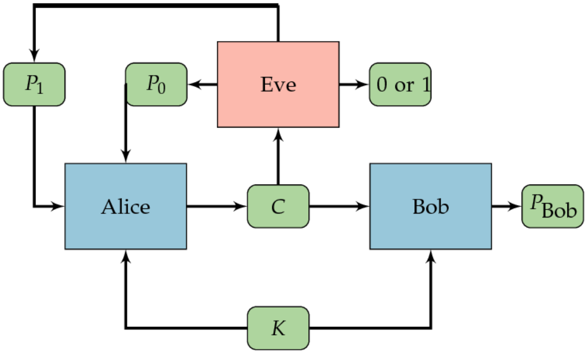
Neural Nets in the ANC protocol share the so-called "Mix & Transform" architecture:

- a first fully-connected (FC) layer, that enables mixing between input vectors;
- a sequence of convolutional layers;
- an output layer of a size suitable for a plaintext or a ciphertext.

Adjustment of the weights is based on Adam optimizer. The training of Eve is alternated with that of Alice and Bob.

Chosen-Plaintext Attack ANC

Main limitation to the security of original ANC model lies in the job of Eve, that appears to be too hard. A solution to this problem is to strengthen Eve by letting it mount a Chosen-Plaintext Attack (CPA).



The resulting cryptosystem is named CPA-ANC and has the following key features.

- Eve choose two plaintexts P_0 and P_1 and sends them to Alice.
- Alice chooses one plaintext randomly, encrypts it to C and send it to Bob and Eve.
- Bob decrypts the message using its neural network.
- Eve outputs 0 if it believes P_0 was encrypted or 1 if it believes P_1 was encrypted.

Training follows a similar procedure to that of the original ANC, but leads to a much stronger communication channel.

References

- Graupe, D. (2007). *Principles of Artificial Neural Networks (2nd Edition)*. Word Scientific Publishing, Singapore.
- McCulloch, W. and Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics 5, 115-133.
- Hebb, D. O. (1949). *The organization of behavior; a neuropsychological theory*. Wiley, New York.
- Rosenblatt, F. (1958). *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65.
- Widrow, B. and Hoff, M. E. (1960). *Adaptive Switching Circuits*. IRE WESCON Convention Record, 96-104.

- MINSKY, M. and PAPERT, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
- RUMELHART, D., HINTON, G. and WILLIAMS, R. (1986). *Learning representations by back-propagating errors*. Nature 323, 533–536.
- SEJNOWSKI, T. J. and ROSENBERG, C. R. (1987). *Parallel Networks that Learn to Pronounce English Text*. Complex Systems, 1.
- KINGMA, D.P. and LEI BA, J. (2015). *Adam: a method for stochastic optimization*. CoRR.
- MITCHELL, M. (1998). *An introduction to Genetic Algorithms*. MIT Press.
- HOLLAND, J. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

- LAURIA, F. E. (1990). *On Neurocryptology.* Proceedings of the Third Italian Workshop on Parallel Architectures and Neural Networks, 337-343.
- VOLNÁ, E. (2000). *Using Neural Network in Cryptography*. University of Ostrava.
- KANTER, I., KINZEL, W. and KANTER, E. (2001). *Secure exchange of information by synchronization of neural networks*. Bar Ilan University.
- KLIMOV, A., MITYAGIN, A. and SHAMIR, A. (2002). *Analysis of Neural Cryptography*. Weizmann Institute.
- ABADI, M. and ANDERSEN, D. G. (2016). *Learning to protect communications with Adversarial Neural Cryptography*. Google Brain

Coutinho, M., Robson de Oliveira Albuquerque, R., Borges, F. , Villalba, L. J. G. and Kim T. H. (2018). *Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography*. University of Brasília.

Jayachandiran, K. (2018). *A Machine Learning Approach for Cryptanalysis*. Rochester Institute of Technology.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L. (2013). *The Simon and Speck Families of Lightweight Block Ciphers*. National Security Agency.

Alani, M. M. (2012). *Neuro-cryptanalysis of DES*. World Congress on Internet Security (WorldCIS-2012)

Gohr, A. (2019). *Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning*. Bundesamt für Sicherheit in der Informationstechnik (BSI).