Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Draft Handout 5: Artificial neural networks

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce the Artificial neural network as a model and procedure in classical and Bayesian framework. Motivation, set-up, description, computation, implementation, tricks. We focus on the Feedforward network.

Reading list & references:

(1) Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
- Ch. 20 Neural Networks

(2) Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.
- Ch. 5 Neural Networks

(3) Neural Networks for Pattern Recognition
- Ch. 4 The multi-layer perceptron

(4) LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (2002). Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-50). Berlin, Heidelberg: Springer Berlin Heidelberg.

1. Intro and motivation [1]

Note 1. Artificial Neural Networks (NN) are statistical models which have mostly been developed from the algorithmic perspective of machine learning. They were originally created as an attempt to model the act of thinking by modeling neurons in a brain. In ML, NN are used as global approximators.

Note 2. The original biological motivation for feed-forward NN stems from McCulloch & Pitts (1943) who published a seminal model of a NN as a binary thresholding device in discrete time, i.e.

$$n_j(t) = 1\left(\sum_{\forall i \to j} w_{j,i} n_i(t-1) > \theta_i\right)$$

where the sum is over neuron i connected to neuron j; $n_j(t)$ is the output of neuron i at time t and $0 < w_{j,i} < 1$ are attenuation weights. Thus the effect is to threshold a weighted sum of the inputs at value θ_i. Perhaps, such a mathematical model involving compositions of interconnected non-linear functions could be able to mimic human's learning mechanism and be implemented in a computing environment (with faster computational abilities) with purpose to discover patters, make predictions, cluster, classify, etc...

[1] In this Section, formulas not needed to be memorized.

Remark 3. Mathematically, NN are rooted in the classical theorem by Kolmogorov stating (informally) that every continuous function $h(\cdot)$ on $[0,1]^d$ can be written as

$$h(x) = \sum_{i=1}^{2d+1} F_i \left(\sum_{j=1}^{d} G_{i,j}(x_j) \right) \tag{1.1}$$

where $\{G_{i,j}\}$ and $\{F_i\}$ are continuous functions whose form depends on f. Perhaps, one may speculate that functions $\{G_{i,j}\}$ and $\{F_i\}$ can be approximated by sigmoids or threshold functions of the form $\sigma\left(w^\top x\right)$ allowing the number of the tunable coefficients w to be high enough such that they can represent any function -hence the property of NN as global approximators.

Example 4. Consider a regression problem with predictive rule $h : \mathbb{R}^d \to \mathbb{R}^q$, suitable for cases where the examples (data) consist of input $x \in \mathbb{R}^d$, and output targets $y \in \mathbb{R}^q$. A 2 layer artificial neural network is

$$h_k(x) = \sigma_{(2)} \left(w_{(2),k,0} + \sum_{\forall j} w_{(2),k,j} \sigma_{(1)} \left(w_{(1),j,0} + \sum_{\forall i} w_{(1),j,i} x_i \right) \right)$$

for $k = 1, ..., q$. One may choose with $\sigma_2(\alpha) = \alpha$, $\sigma_1(\alpha) = 1/(1 + \exp(-\alpha))$. The only left here is to learn unknown parameters $\left\{w_{(\cdot),\cdot,\cdot}\right\}$.

2. Feedforward Neural Network (Mathematical set-up)

Note 5. An (Artificial) Neural Network (NN) can be depicted as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes $\mathcal{V}$ correspond to neurons and edges $\mathcal{E}$ correspond to links between them.

Note 6. A Feed-Forward Neural Network (FFNN) or else multi-layer perceptron is a special case of NN which can be depicted by a directed acyclic graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. [2] (See Figure 2.1).

[2](its vertices can be numbered so that all connections go from one vertex to another with a higher number)

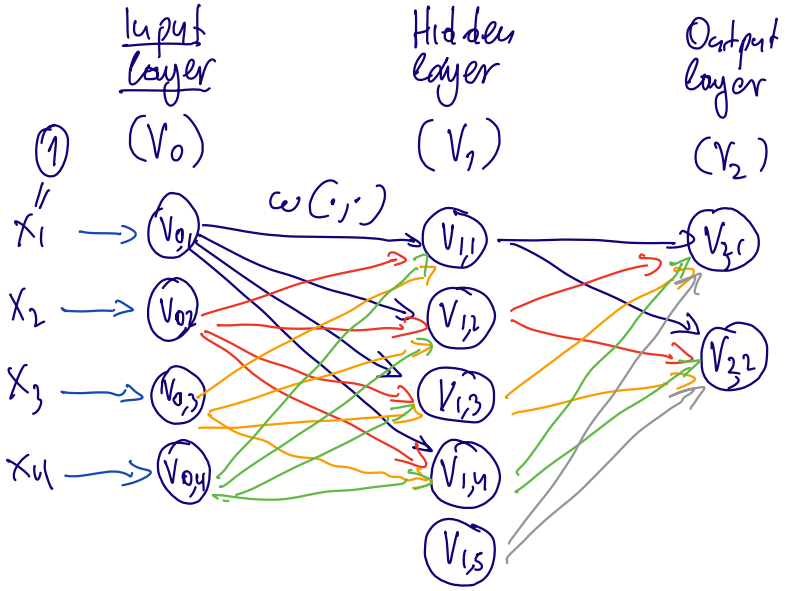

FIGURE 2.1. Feed forward neural network (1 hidden layer)

Note 7. We assume that the network is organized in layers. The set of nodes is decomposed into a union of (nonempty) disjoint subsets

$$V = \cup_{t=0}^{T} V_t$$

such that every edge in $\mathcal{E}$ connects a node from V_t to a node from V_{t+1}, for $t = 1, ..., T$.

Note 8. The first layer V_0 is called **input layer**. If x has d dimensions, then the first layer V_0 contains d nodes.

Note 9. The last layer V_T is called **output layer**. If y has q dimensions, then the last layer V_T contains q nodes.

Note 10. The intermediate layers $\{V_1, ..., V_{T-1}\}$ are called **hidden layers.**

Note 11. In a neural network, the nodes of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ correspond to neurons.

Notation 12. The **i-th neuron of the t-th layer** is denoted as $v_{t,i}$.

Note 13. The output of neuron i in the input layer V_0 is simply x_i that is $o_{0,i}(x) = x_i$for $i = \{1, ..., d\}$.

Note 14. Each edge in the graph $(v_{t,j}, v_{t+1,i})$ links the output of some neuron $v_{t,j}$ to the input of another neuron $v_{t+1,i}$; i.e. $(v_{t,j}, v_{t+1,i}) \in \mathcal{E}$.

Note 15. We define a weight function $w : \mathcal{E} \to \mathbb{R}$ over the edges $\mathcal{E}$.

Note 16. Activation of neuron i at hidden layer 1 is the weighted sum of the outputs $o_{0,i}(x) = x_i$ of the neurons in V_0 which are connected to $v_{1,i}$ where weighting is according to function w, that is

$$\alpha_{1,i}(x) = \sum_{\forall j:(v_{0,j},v_{1,i})\in\mathcal{E}} w((v_{0,j},v_{1,i}))\, x_i \tag{2.1}$$

Note 17. Each single neuron $v_{t,i}$ is modeled as a simple scalar function, $\sigma_t(\cdot) : \mathbb{R} \to \mathbb{R}$, called **activation function** at layer t.

Note 18. We denote by $o_{t,i}(x) := \sigma_{t-1}(\alpha_{t-1,i}(x))$ **the output of neuron** $v_{t,i}$ when the network is fed with the input x.

Remark 19. Activation of neuron i at layer t is the weighted sum of the outputs $o_{t-1,j}(x)$ of the neurons in V_{t-1} which are connected to $v_{t,i}$ where weighting is according to function w, that is

$$\alpha_{t,i}(x) = \sum_{\forall j:(v_{t-1,j},v_{t,i})\in\mathcal{E}} w((v_{t-1,j},v_{t,i}))\, o_{t-1,j}(x) \tag{2.2}$$

Note 20. The input of a neuron is obtained by taking a weighted sum of the outputs of all the neurons connected to it, where the weighting is according to w.

Definition 21. To sum-up: The the feed-forward NN formula in a layer by layer manner is performed (defined) according to the following recursion.

Note 22. At $t = 0$, for $i = 1, ..., |V_0|$

$$o_{0,i}(x) := x_i$$

At $t > 0$, for $i = 1, ..., |V_{t+1}|$

$$\alpha_{t+1,i}(x) = \sum_{\forall j:(v_{t,j},v_{t+1,i})\in\mathcal{E}} w((v_{t,j},v_{t+1,i}))\, o_{t,j}(x)$$
$$o_{t+1,i}(x) = \sigma_{t+1}(\alpha_{t+1,i}(x))$$

Note 23. The input to $v_{t+1,i}$ is activation $\alpha_{t+1,i}(x)$ namely a weighted sum of the outputs $o_{t,j}(x)$ of the neurons in V_t which are connected to $v_{t+1,i}$, where weighting is according to w. The output of $v_{t+1,i}$ is the application of the activation function $\sigma_{t+1}(\cdot)$ on its input $\alpha_{t+1,i}(x)$.

Note 24. Depth of the NN is the number of the layers T.

Note 25. Size of the network is the number $|V|$.

Note 26. Width of the NN is the number $\max_{\forall t}(|V_t|)$.

Note 27. The architecture of the neural network is defined by the triplet $(\mathcal{V}, \mathcal{E}, \sigma_t)$.

Note 28. The neural network can be fully specified by the quadruplet $(\mathcal{V}, \mathcal{E}, \sigma_t, w)$.

Example 29. Figure 2.1 denotes a NN with depth 2, size 11, width 5. The neuro with no incoming edges has$o_{1,5} = \sigma(0)$.

Notation 30. To easy the notation, we denote the weights as $w_{(t),j,i} := w\left(\left(v_{t,j}, v_{t+1,i}\right)\right)$. Using this notation, $w_{(t),j,i} = 0$ is equivalent in (2.2) to $\left(v_{t,j}, v_{t+1,i}\right) \notin \mathcal{E}$ and means that the link $v_{t,j} \to v_{t+1,i}$ is not in the network.

Note 31. Often a **constant neuron** $v_{t,0}$ (at each layer t and $i = 0$) which outputs 1; i.e. $o_{0,0}(x) = 1$ and $o_{t,0}(x) = 1$. The corresponding weight $w_{(t),k,0}$ is called **bias**. This resembles to the constant term in the linear regression.

Example 32. (Cont. Example 4) The 2 layer neural network

$$h_k(x) = \sigma_{(2)}\left(w_{(2),k,0} + \sum_{\forall j} w_{(2),k,j}\sigma_{(1)}\left(w_{(1),j,0} + \sum_{\forall i} w_{(1),j,i}x_i\right)\right) \tag{2.3}$$

can be written according to the recursion in Note **??** as

- Input layer
$$o_{(0),i}(x) = \begin{cases} 1 & i = 0 \\ x_i & i = 1, ..., d \end{cases}$$
- Hidden layer
$$\begin{aligned} \alpha_{(1),j}(x) &= w_{(1),j,0} + \sum_{\forall j} w_{(1),j,i}x_i \\ o_{(1),j}(x) &= \sigma_{(1)}\left(\alpha_{(1),j}(x)\right) \\ &= \sigma_{(1)}\left(w_{(1),j,0} + \sum_{\forall j} w_{(1),j,i}x_i\right) \end{aligned}$$
- Hidden layer
$$\begin{aligned} \alpha_{(2),k}(x) &= w_{(2),k,0} + \sum_{\forall j} w_{(2),k,j}o_{(2),k}(x) \\ o_{(2),k}(x) &= \sigma_{(2)}\left(\alpha_{(2),k}(x)\right) \\ &= \sigma_{(2)}\left(w_{(2),k,0} + \sum_{\forall j} w_{(2),k,j}o_{(2),k}(x)\right) \end{aligned}$$

If $h_k(x) \in \mathbb{R}$, we can choose $\sigma_{(2)}$ as the identity function i.e., the $\sigma_{(2)}(\alpha) = \alpha$. Note that (2.3) is also presented more compact

$$h_k(x) = \sigma_{(2)}\left(\sum_{\forall j} w_{(2),k,j}\sigma_{(1)}\left(\sum_{\forall i} w_{(1),j,i}x_i\right)\right)$$

by considering the first x as 1, i.e.g $x_1 = 1$ similar to the linear regression models.

Remark 33. Activation functions σ_t are non-increasing functions often sigmoids or threshold functions. Their choice is problem-dependent. some examples

- Identity function: $\sigma(\alpha) = \alpha$ cannot be used in hidden layers

- Threshold sigmoid: $\sigma(\alpha) = 1(\alpha > 0)$
- Logistic sigmoid: $\sigma(\alpha) = \frac{1}{1+\exp(-\alpha)}$
- Rectified linear unit: $\text{RELU}(\alpha) = \max(\alpha, 0)$

Example 34. Examples on the choice of the activation function at the output layer T:

- In the univariate regression problem with prediction rule $h(\cdot) \in \mathbb{R}$ and examples $z_i = (x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, we can choose $\sigma_T(\alpha) = \alpha$ to get $h(\alpha) = \sigma_T(\alpha) = \alpha$.
- In the binary logistic regression problem with prediction rule $h(\cdot) \in [0,1]$ and examples $z_i = (x_i, y_i) \in \mathbb{R}^d \times \{0,1\}$, we can choose $\sigma_T(\alpha) = \frac{1}{1+\exp(-\alpha)}$ to get $h(\alpha) = \sigma_T(\alpha) = \frac{1}{1+\exp(-\alpha)} = \frac{\exp(\alpha)}{1+\exp(\alpha)}$.

3. Learning Neural Networks

Note 35. Assume we are interested in a prediction rule $h_{\mathcal{V},\mathcal{E},\sigma,w} : \mathbb{R}^{|V_0|} \to \mathbb{R}^{|V_T|}$ which is modeled as a feed-forward Neural Network with $(\mathcal{V}, \mathcal{E}, \sigma, w)$; that is

$$h_{\mathcal{V},\mathcal{E},\sigma,w}(x) = o_T(x)$$

where $o_T = \left(o_{T,1}, ..., o_{T,|V_T|}\right)^\top$ is according to the Definition 21.

Note 36. Learning $(\mathcal{V}, \mathcal{E}, \sigma)$ is a model selection task, as it involves learning the architecture of the neural network.

Note 37. We assume that the architecture $(\mathcal{V}, \mathcal{E}, \sigma)$ of the neural network is fixed (given), and that there is interest in learning the weight function $w : \mathcal{E} \to \mathbb{R}$ or equivalently in vector form the vector of weights $\left\{w_{(t),j,i}\right\}$ where $w_{(t),j,i} := w((v_{t,j}, v_{t+1,i}))$.

Note 38. The class of hypotheses is

$$\mathcal{H}_{\mathcal{V},\mathcal{E},\sigma} = \{h_{\mathcal{V},\mathcal{E},\sigma,w} : \text{for all } w : \mathcal{E} \to \mathbb{R}\}$$

for given $(\mathcal{V}, \mathcal{E}, \sigma)$. To simplify notation we will use h_w instead of $h_{\mathcal{V},\mathcal{E},\sigma,w}$.

Note 39. Assume that there is available a training set of examples (data-set) $\mathcal{S} = \{z_i = (x_i, y_i); i = 1, ..., n\}$ with $x_i \in \mathcal{X} = \mathbb{R}^{|V_0|}$ and $y_i \in \mathcal{Y}$.

Note 40. We consider a loss function $\ell(w, z)$ at some value of weight vector $w \in \mathbb{R}^{|\mathcal{E}|}$ and at some example $z = (x, y)$.

Example 41. For instance, loss function can be based on

(1) some norm $\ell(w, z) = \frac{1}{2}\|h_w(x) - y\|_2^2$, or
(2) the pdf/pmf of the sampling distribution $f(y|w)$ of the target y given the weight vector w as $\ell(w, z) = -\log(f(y|w))$

Definition 42. Error function is a performance measure that can be defined as

$$\text{EF}(w|z) = -\sum_{i=1}^{n} \ell(w, z_i^*)$$

where $\mathcal{S}^* = \{z_i^* = (x_i^*, y_i^*) ; i = 1, ..., n^*\}$ is a set of examples which does not necessarily need to be the training set $\mathcal{S}$.

4. Classical learning of Neural Network

Problem 43. Find $h_w \in \mathcal{H}_{\mathcal{V},\mathcal{E},\sigma}$ essentially find $w \in \mathbb{R}^{|\mathcal{E}|}$ under loss $\ell(\cdot,\cdot)$, and training data-set $\mathcal{S} = \{z_i = (x_i, y_i) ; i = 1, ..., n\}$. Compute $w^* \in \mathbb{R}^{|\mathcal{E}|}$, by minimizing the risk function $R_g(w)$

$$w^* = \arg\min_{w\in\mathcal{H}} (R_g(w)) = \arg\min_{w\in\mathcal{H}} (\mathrm{E}_{z\sim g}(\ell(w,z))) \tag{4.1}$$

or by minimizing the empirical risk function $\hat{R}_S(w)$

$$w^* = \arg\min_{w\in\mathcal{H}} \left(\hat{R}_S(w)\right) = \arg\min_{w\in\mathcal{H}} \left(\frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)\right) \tag{4.2}$$

Problem 44. Often neural network models are over-parameterized, namely the dimensionality of $w \in \mathbb{R}^{|\mathcal{E}|}$ is too large. One can resort to shrinkage methods. For a given architecture $(\mathcal{V}, \mathcal{E}, \sigma)$, 'node selection' can be performed if $w_{(t),j,i} = 0$ which is equivalent in (2.2) to $(v_{t,j}, v_{t+1,i}) \notin \mathcal{E}$ and means that the link $v_{t,j} \to v_{t+1,i}$ is not active (essentially not in the network).
The problem is to find w^* such that

$$w^* = \arg\min_{w\in\mathcal{H}} (R_g(w) + J(w;\lambda)) \tag{4.3}$$

$$= \arg\min_{w\in\mathcal{H}} (\mathrm{E}_{z\sim g}(\ell(w,z) + J(w;\lambda))) \tag{4.4}$$

where $J(w;\lambda)$ is a shrinkage term such as

- Ridge: $J(w;\lambda) = \lambda \|w\|_1$
- LASSO: $J(w;\lambda) = \lambda \|w\|_2^2$
- Elastic net: $J(w;\lambda = (\lambda_1, \lambda_2)) = \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$

Note 45. Set $w_{t,i,j}^* = 0$ if $w_{t,i,j}^*$ is less than a threshold user specific value $\xi > 0$ i.e. $\left|w_{t,i,j}^*\right| < \xi$.

Note 46. Training a neural network model is usually a high-dimensional problem (essentially w has high dimensionality) and a big-data problem (essentially we need a large number of training examples to learn a large number of weights). For this reason, Stochastic Gradient Descent (and its variations) is a suitable stochastic learning algorithm.

Note 47. To address Problem 43, the recursion of the SGD with batch size m is

$$w^{(t+1)} = w^{(t)} - \eta_t \frac{1}{m}\sum_{j=1}^{m} \partial_w \ell\left(w^{(t)}, z_j^{(t)}\right)$$

To address the Problem 44the recursion of the SGD with batch size m is

$$w^{(t+1)} = w^{(t)} - \eta_t \left[\frac{1}{m}\sum_{j=1}^{m} \partial_w \ell\left(w^{(t)}, z_j^{(t)}\right) + \partial_w J(w;\lambda)\right]$$

for some positive λ which is user specified, or chosen via cross validation.

Appendix A. About graphs

Definition 48. A directed graph is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising

- a set of nodes $\mathcal{V}$ (or vertices), where a nodes are abstract objects, and
- a set of edges $\mathcal{E} = \{(v, u) \,|\, v \in \mathcal{E}, u \in \mathcal{E}, v \neq u\}$ (or directed edges, directed links, arrows) which are ordered pairs of vertices (that is, an edge is associated with two distinct vertices).

Definition 49. An edge-weighted graph or a network is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ equipped with a weight function $w : \mathcal{E} \to \mathbb{R}$ that assigns a number (the weight) to each edge $e \in \mathcal{E}$.

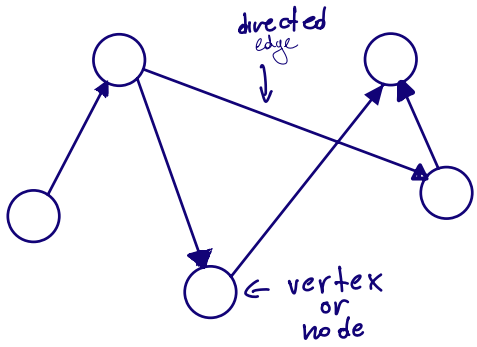

Figure A.1. A directed graph