Gradient-based optimization methods for neural network training

Ph.D. in Industrial Engineering
Università di Firenze, A.A. 2020/2021

Simone Rebegoldi
Dipartimento di Ingegneria Industriale
Università di Firenze

UNIVERSITÀ DEGLI STUDI FIRENZE

Seminar content

- Overview on models arising in machine learning[1]
- Analysis of stochastic gradient methods[2,3]
- Noise-reduction techniques
- Methods with adaptive choice of the learning-rate
- Application to Artificial Neural Networks

[1] Goodfellow, Bengio, Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[2] Bottou, Curtis, Nocedal. *Optimization Methods for Large-Scale Machine Learning*, Siam Review 60 (2), 223–311, 2018.

[3] Bianconcini, Bellavia, Krejic, Morini, *Subsampled first-order optimization methods with applications in imaging*, Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging, 2021.

Machine learning algorithms

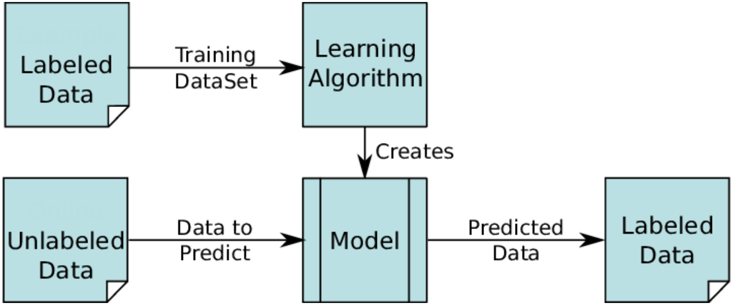
A machine learning algorithm is an algorithm able to learn from data.

Mitchell, Machine Learning, McGraw-Hill, New York, 97, 1997.
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- **Tasks**: **classification** (object recognition, design of feasible industrial components ...), **regression**, **transcription** (character recognition, speech recognition ...), **machine translation** ...
- **Experience: large-scale data sets**
- **Measure: training error, validation error, testing error**

Supervised learning process

- *training set*: $\{(a_i, b_i)\}_{i=1,\dots,N}$
- *testing set*: $\{(a_i^{\text{test}}, b_i^{\text{test}})\}_{i=1,\dots,N_{\text{test}}}$
- a_i is called the feature vector.
- b_i is the true output associated to the input a_i.



Training as solving an optimization problem

Training consists in solving a finite-sum minimization problem.

Finite-sum minimization problems

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{N} \sum_{i=1}^{N} \phi_i(x),$$

- $\phi_i : \mathbb{R}^n \to \mathbb{R}$, $\phi_i \in C^1(\mathbb{R}^n)$, $i = 1, \ldots, N$ and f bounded below.
- Goal: compute ϵ_g-approximate first-order critical points:

$$\|\nabla f(\hat{x})\| \leqslant \epsilon_g.$$

- Challenge: when N is large, the evaluation of f and its derivative information is computationally expensive.

Classification example (1)

The MNIST Dataset[1]

DATA	Training Size N	Test Size	Numb. of Features d
MNIST	60000	10000	784

Each feature vector (row in the feature matrix) consists of 784 pixels – unrolled from the original 28x28 pixels images.

digits classification: hand-written digits 0, 1,... 9.

[1] http://yann.lecun.com/exdb/mnist

Classification example (2)

Mushrooms Dataset[1]

DATA	Training Size N	Test Size	Numb. of Features d
Mushrooms	5000	3124	112

Each feature vector consists of 0 or 1.

Safe to eat or deadly poison?

[1] https://www.kaggle.com/uciml/mushroom-classification

Classification example (3)

Parametric Design of Centrifugal Pumps[1,2]

DATA	Training Size N	Test Size	Numb. of Features d
Pumps	61600	440	15400

- **Features**: parameters describing the pump geometry
- **Task**: classify feasible and unfeasible pumps

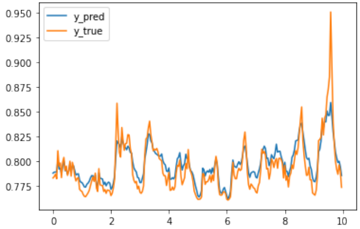
[1] Riccietti, Bellucci, Checcucci, Marconcini, Arnone, Engineering Optimization 50, 1304–1324, 2018.

[2] Checcucci, Schneider, Marconcini, Rubechini, Arnone, De Franco, Coneri, Proc. of 12th International Symposium on Experimental and Computational Aerothermodynamics of Internal Flows, 2015.

Regression example

Benzene estimation[1]

DATA	Training Size N	Test Size	Numb. of Features d
Air	6294	2697	7



Benzene concentration predicted and true (10 days)

- **Features:** concentrations of 7 pollutants measured in the centre of an Italian city characterized by heavy car traffics from March 2004 until April 2005.
- **Task:** predict the value of benzene concentration.

[1] De Vito, Massera, Piga, Martinotto, Di Francia, In Sensors and Actuators B: Chemical 129 (2), 2008.

Fundamentals

- **Goal**
  Determine a prediction function (model) $h : \mathcal{A} \to \mathcal{B}$ such that, given $a \in \mathcal{A}$, the value $h(a)$ offers an accurate prediction about the true output b associated to the input a.

- **General problems**
  - Binary classification: classify instances into $\kappa = 2$ classes.
  - Multi-class classification: classifying instances into one of $\kappa \geqslant 3$ classes.

    The prediction function $h : \mathcal{A} \to [0,1]^\kappa$ is such that, given $a \in \mathcal{A}$, the value $(h(a))_j$ is the prediction of the probability of input a to be classified in class j. The input is then associated to the class corresponding to the highest probability.
  - Regression: in this case, $b \in \mathbb{R}^{d_b}$ and hence $h : \mathcal{A} \to \mathbb{R}^{d_b \times \kappa}$.

Empirical risk

Goal: determine a prediction function (model) $h : \mathcal{A} \to \mathcal{B}$, belonging to a family of prediction functions $\mathcal{H}$.

- Choose a prediction function parametrized by a vector $x \in \mathbb{R}^n$

$$h \in \mathcal{H} = \{h(\cdot; x) : x \in \mathbb{R}^n\}.$$

- Introduce a loss function $\ell : \mathcal{A} \times \mathcal{B} \to \mathbb{R}$ that, given an input-output pair (a, b), yields the loss $\ell(h(a; x), b)$ when b is predicted by $h(a; x)$.
- Given a set of examples $\{(a_i, b_i)\}_{i=1}^N$ (training set), $a_i \in \mathbb{R}^d$ (features), $b_i \in \mathbb{R}^p$ (label), compute x so as to minimize

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(h(a_i; x), b_i)}_{\phi_i(x)} \qquad \text{Empirical Risk}$$

- Testing set to evaluate generalization properties of the model.

Logistic classification model (logit or logistic regression)

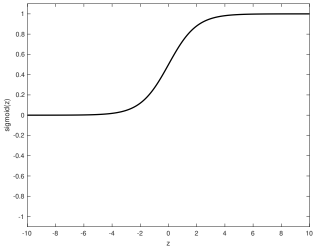
Given $\{(a_i, b_i)\}_{i=1}^N$, $a_i \in \mathbb{R}^d$, $b_i \in \{-1, +1\}$ (binary classification problem).

- Assume the conditional probability $P(b|a)$ of b being the label of a is

$$P(b|a) = \zeta(a, b; x) = \frac{1}{1 + e^{-ba^T x}}, \qquad \zeta(a, b; x) : \mathbb{R}^n \to (0, 1)$$

$$\zeta(a, 1; x) + \zeta(a, -1; x) = \frac{1}{1 + e^{-a^T x}} + \frac{1}{1 + e^{a^T x}} = 1$$

and $x \in \mathbb{R}^d$ is the parameters vector.
The function $\zeta(z) = \frac{1}{1+e^{-z}}$ is called the sigmoid function.



Logistic loss

- We take x that maximizes $P(b_1, b_2, \ldots, b_N | a_1, a_2, \ldots a_N)$:

$$\max_{x \in \mathbb{R}^d} \prod_{i=1}^{N} \zeta(a_i, b_i; x) = \max_{x \in \mathbb{R}^d} \prod_{i=1}^{N} \frac{1}{1 + e^{-b_i a_i^T x}}.$$

- Taking the logarithm and setting $n = d$:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\log(1 + e^{-b_i a_i^T x})}_{\phi_i(x)}.$$

Logistic loss

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\log(1 + e^{-b_i a_i^T x})}_{\phi_i(x)}$$

- Given $\hat{x}$ resulting from the classifier training, we classify the new instance $\hat{a} \in \mathbb{R}^n$ as follows

$$\text{If } P(1|\hat{a}) = \frac{1}{1 + e^{-\hat{a}^T \hat{x}}} \geqslant 0.5, \quad \text{set } \hat{b} = 1$$
$$\text{If } P(1|\hat{a}) = \frac{1}{1 + e^{-\hat{a}^T \hat{x}}} < 0.5, \quad \text{set } \hat{b} = -1.$$

Logistic loss with ℓ_2 regularization

The logistic loss with ℓ_2 regularization is given by

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + e^{-b_i a_i^T x}) + \frac{1}{2N}\|x\|_2^2.$$

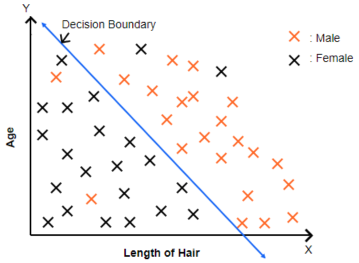
where $\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$.

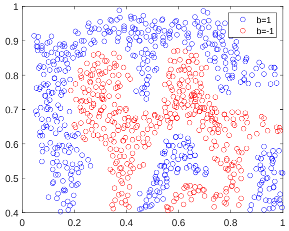
Regularization avoids overfitting (i.e. fitting too closely noisy data) by imposing sparsity on the model parameters (see tomorrow's seminar).

ℓ_2 regularization makes the logistic loss strongly convex
$\Rightarrow$ good convergence properties for machine learning algorithms

Overcoming linear models

The logistic regression model assumes that the data is linearly separable, i.e. can be separated by a line.
This is not the case for many real-life applications.





Non-linearity often needs to be encoded in the prediction function.

Feedforward Neural Networks

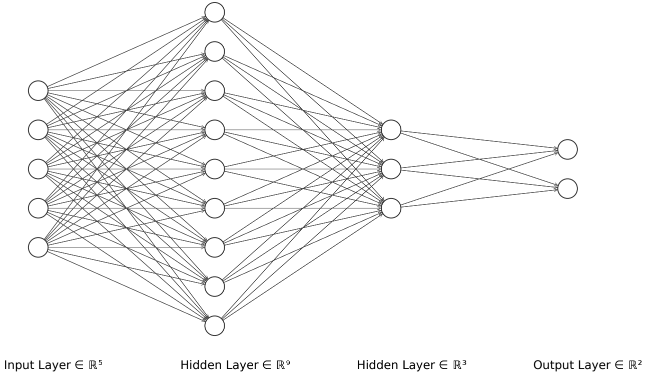
Feedforward Neural Networks or Multilayer perceptrons (MLPs) are non-linear models which aim at approximating an unknown function.

- They are called feedforward because information flows through the function being evaluated from x, through some intermediate computations, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself, as opposed to the so-called recurrent neural networks.
- They are called networks as they are typically obtained by composing together several different functions.
- They are called neural since they are vaguely inspired by neuroscience.

Feedforward Neural Networks

The prediction function h is determined by the network's architecture, the vector of parameters x is given by the network's weights and bias.

- Network's layers: $L_1, \ldots, L_m$ (where $m \geqslant 2$), L_1 *input layer*, L_m *output layer*. Case $m > 2$: $L_2, \ldots, L_{m-1}$ are the so-called hidden layers.
- n_i number of neurons of layer L_i; $n_1 = d$, $n_m = p$.



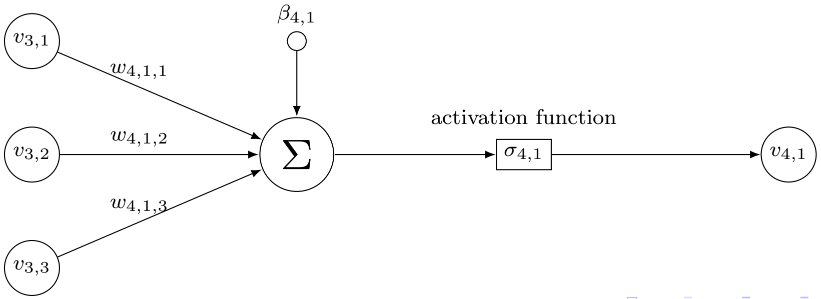
Example: $m = 4$ levels, $n_1 = d = 5$, $n_2 = 9$, $n_3 = 3$, $n_4 = p = 2$.

Feedforward Neural Networks

- Let $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,n_i})^T \in \mathbb{R}^{n_i}$ be the output of layer L_i and $\boldsymbol{\sigma}_i = (\sigma_{i,1}, \ldots, \sigma_{i,n_i})^T \in \mathbb{R}^{n_i}$ contain the *activation* functions $\sigma_{i,j} : \mathbb{R} \to \mathbb{R}$.
- The output of the j-th neuron of the layer L_i, for $i = 2, \ldots, m$ is the scalar

$$v_{i,j} = \sigma_{i,j}\left(\sum_{k=1}^{n_{i-1}} w_{i,j,k} \cdot v_{i-1,k} + \beta_{i,j}\right) \qquad j = 1, \ldots, n_i,$$

  where $\beta_{i,j} \in \mathbb{R}$ is called *bias* and the parameters $w_{i,j,k}$ are called *weights*.
- v_{i1} is given by the input data a.



Output

- Letting $\mathbf{W}_i \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_{i-1}}$ be the matrix with (j,k)-entry given by

$$(\mathbf{W}_i)_{j,k} = w_{i,j,k}, \quad 1 \leqslant j \leqslant n_i, \ 1 \leqslant k \leqslant n_{i-1}$$

and $\beta_i = \left(\beta_{i,1}, \ldots, \beta_{i,n_i}\right)^T \in \mathbb{R}^{n_i}$, the output of the whole layer L_i is

$$\mathbf{v}_i = \boldsymbol{\sigma}_i \left(\mathbf{W}_i \mathbf{v}_{i-1} + \beta_i\right).$$

- In fact, the output of each layer is defined recursively and depends on the output of the previous layer:

$$\begin{cases} \mathbf{v}_1 = a \\ \mathbf{v}_i = \boldsymbol{\sigma}_i \left(\mathbf{W}_i \mathbf{v}_{i-1} + \beta_i\right) \quad i = 2, \ldots, m \end{cases}$$

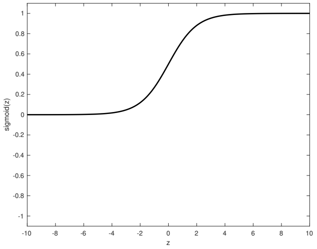
- Case $m = 2$, $p = 1$. Given the input a, the output of the network is:

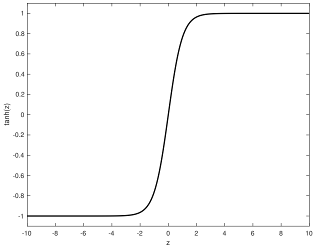
$$v = \sigma\left(\sum_{k=1}^{d} w_k \cdot a_k + \beta\right) = \sigma\left(w^T a + \beta\right)$$

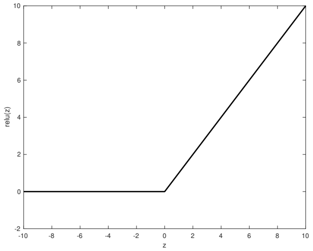
Activation functions

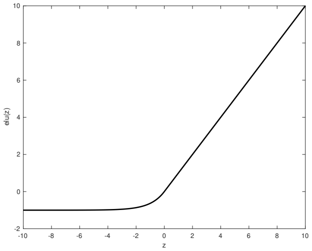
- linear: $\sigma(z) = z$;
- sigmoid or *logistic*: $\sigma(z) = \frac{1}{1+\mathrm{e}^{-z}}$.
  Used in the output level if the label belongs to $[0,1]$.
- softmax : $\boldsymbol{\sigma}\colon \mathbb{R}^k \to \mathbb{R}^k$ defined by $\sigma(\boldsymbol{z})_j := \frac{\mathrm{e}^{z_j}}{\sum_{i=1}^k \mathrm{e}^{z_i}}$ for $j = 1, \dots, k$.
  Produces positive estimates that sum up to 1 and is particularly useful in classification when the output represents the probability of some $\in \mathbb{R}^s$ to belong to different classes.
- hyperbolic function: $\sigma(z) = \tanh(z) = \frac{\mathrm{e}^z - \mathrm{e}^{-z}}{\mathrm{e}^z + \mathrm{e}^{-z}}$; output in $[-1,1]$.
- ReLU (Rectified Linear Unit): $\sigma(z) = \max(0, z)$.
- ELU (Exponential Linear Unit): $\sigma(z) = z \cdot \mathbb{1}_{[z \geqslant 0]} + (\mathrm{e}^z - 1) \cdot \mathbb{1}_{[z<0]}$.

Activation functions









Why using neural networks?

Universal approximation theorem (Hornik et al., 1989; Cybenko, 1989)
A feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (as the sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.

Deep learning

Deep learning

Machine learning algorithms based on artificial neural networks.

Main steps

- Choose the network architecture and the activation functions
- Choose the loss function
- Train the network to compute weights and bias $\Rightarrow$ Model

Neural network training

The procedure for choosing the parameters $\{(\mathbf{W}_i, \beta_i)\}_{i=2,\ldots,m}$ is called training phase.

- Let
$$x = \left(\mathrm{vec}(\boldsymbol{W}_2), \boldsymbol{\beta}_2^{\mathrm{T}}, \ldots, \mathrm{vec}(\boldsymbol{W}_m), \boldsymbol{\beta}_m^{\mathrm{T}}\right)^{\mathrm{T}},$$
where $\mathrm{vec}(A)$ is the vector obtained by stacking A column by column.
- Given the set of known data $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1,\ldots,N}$ (*training set*), the aim is to choose the parameters so that the output $\mathbf{v}_m(x; \mathbf{a}_i)$ of the neural network corresponding to the input $\mathbf{a}_i$ is as close as possible to the true output $\mathbf{b}_i$ for every $i = 1, \ldots, N$.
- We have to minimize the empirical risk:

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(\mathbf{v}_m(x; \mathbf{a}_i), \mathbf{b}_i)}_{\phi_i(x)}.$$

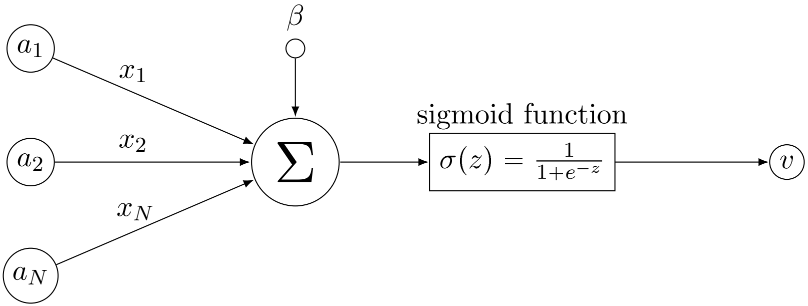
Logistic regression as a neural network

Case $m = 2$ (no hidden layers), $\sigma(z) = 1/(1+e^{-z})$ (sigmoid function)

- Given $\{(a_i, b_i)\}_{i=1}^{N}$, $a_i \in \mathbb{R}^n$, $b_i \in \{0,1\}$ $(p = 1)$, the output is:

$$v = \sigma\left(w^T a + \beta\right) = \frac{1}{1 + e^{-a_i^T w + \beta}}.$$

If $\beta = 0$, this is the probability $P(1|a_i)$ used in the logistic regression!



Logistic regression as a neural network

- If we interpret the output of the network as a predicted label, setting $x = (w_1, \dots, w_d, \beta)^T$, it is reasonable to use a least squares loss[1]:

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} \left(b_i - \frac{1}{1 + e^{-a_i^T x(1:d) + x_{d+1}}} \right)^2 \quad \text{non-convex}$$

- Let $\hat{x} = (\hat{w}_1, \dots, \hat{w}_d, \hat{\beta})^T$ the approximation computed by the network training.
- Binary classification: the classifier is such that

$$\begin{aligned} \frac{1}{1 + e^{-a_i^T \hat{w} + \hat{\beta}}} &\geqslant 0.5 \quad \Rightarrow \quad b_i = 1 \\ \frac{1}{1 + e^{-a_i^T \hat{w} + \hat{\beta}}} &< 0.5 \quad \Rightarrow \quad b_i = 0 \end{aligned}$$

[1] Xu, Roosta, Mahoney, Proc. of the 2020 SIAM International Conference on Data Mining, 2020.