

SUBSTANTIATION OF THE BACKPROPAGATION TECHNIQUE VIA THE HAMILTON-PONTRYAGIN FORMALISM FOR TRAINING NONCONVEX NONSMOOTH NEURAL NETWORKS¹

Vladimir I. Norkin

V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine, Kiev & Faculty of Applied Mathematics of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Email: vladimir.norkin@gmail.com

Abstract. The paper observes the similarity between the stochastic optimal control of discrete dynamical systems and the training multilayer neural networks. It focuses on contemporary deep networks with nonconvex nonsmooth loss and activation functions. In the paper, the machine learning problems are treated as nonconvex nonsmooth stochastic optimization problems. As a model of nonsmooth nonconvex dependences, the so-called generalized differentiable functions are used. A method for calculating stochastic generalized gradients of the learning quality functional for such systems is substantiated basing on Hamilton-Pontryagin formalism. This method extends a well-known “backpropagation” machine learning technique to nonconvex nonsmooth networks. Stochastic generalized gradient learning algorithms are extended for training nonconvex nonsmooth neural networks.

Key words: machine learning, deep learning, multilayer neural networks, nonsmooth nonconvex optimization, stochastic optimization, stochastic generalized gradient.

Introduction. The machine learning problem consists of the identification of parameters of a neural network model, e.g., neural weights, using a set of input-output observations. The training task is formulated as the task of minimizing some smooth loss functional (empirical risk), which measures the average forecast error of the neural network model.

Methods of training (identification) of large neural network models are discussed in many articles and monographs [1, 2]. To train deep (i.e., multilayer) neural networks, the stochastic gradient method, and its modifications are mainly used [2, 3], adopted from the theory of stochastic approximation and stochastic programming since only they are practically applicable for training such networks. The stochastic gradient of the risk functional is a random vector whose mathematical

¹ The work is partially supported by grant CPEA-LT-2016/10003 funded by the Norwegian Agency for International Cooperation and Quality Enhancement in Higher Education (Diku).

expectation approximates the gradient of the target functional, and the stochastic gradient descent method is an iterative method for changing the desired model parameters in the direction of the stochastic (anti-) gradient.

To solve smooth neural network training problems, the BackProp method is widely used [1, 4], i.e. a special method for calculating gradients of the target functional over various parameters. The history of discovery, development, and application of the BackProp method was studied in [5]. Nonsmooth machine learning tasks arise when using nonsmooth (module type) indicators of the quality of training, when applying nonsmooth regularizations, and also when using nonsmooth (for example, piecewise linear, linear rectification ReLU, etc.) activation functions in multilayer neural networks [1, 5]. Such functions give rise to essentially nonconvex nonsmooth functionals of the quality of learning, and the question arises of the convergence of the stochastic generalized gradient descent method in solving such problems. This problem has been relatively recently recognized and is already being considered in the literature [6]. However, it is usually assumed using the Clarke stochastic subgradients [7] of the optimized functional but the problem of their calculation for deep networks is not discussed.

In this paper, we extend the BackProp method to calculating stochastic gradients of nonconvex nonsmooth problems for training multilayer neural networks and formulate the method in terms of stochastic generalized gradients of nonsmooth Hamilton-Pontryagin functions. As a model of nonsmooth nonconvex dependencies, we use the so-called generalized differentiable functions [8, 9]. We also consider an important version of the BackProp method for training the so-called recurrent neural networks, i.e. networks with feedbacks and memory [1, Ch. 10].

Nonconvex nonsmooth learning problems and calculation of stochastic generalized gradients. Let us consider a standard neural network model. Let the network consists of m layers of neurons, each layer $i \in \{1, \dots, m\}$ has n_i neurons with

numbers $j = 1, \dots, n_i$ and each of them has n_{i-1} inputs and one output. In the initial layer there are n_1 neurons, each neuron of this layer has n_0 common inputs and one output. The outputs of the neurons of each layer go to the inputs of the neurons of the next layer. The output layer of the network may consist of one or more neurons.

In the theory of neural networks, the standard mathematical model of neuron (i, j) is some smooth activation function $g_i^j(x_i, w_{ij}, v_{ij})$ (e.g., the logistic sigmoid, the hyperbolic tangent, and etc. [1, Section 6.3.2], which expresses the dependence of the output signal $x_{(i+1)j}$ of the neuron (i, j) on the input signal x_i , for example,

$x_{(i+1)j} = g_i^j(x_i, w_{ij}, v_{ij}) = \left(1 + \exp\left\{-\langle x_i, w_{ij} \rangle - v_{ij}\right\}\right)^{-1}$, where $x_i \in \mathbb{R}^{n_{i-1}}$ is a common input of all neurons in layer i ; $w_{ij} \in \mathbb{R}^{n_{i-1}}$ and $v_{ij} \in (-\infty, +\infty)$ are the individual weight vector and the activation level of neuron $j \in \{1, \dots, n_i\}$ in layer i ; the expression $\langle x_i, w_{ij} \rangle$ denotes the scalar product of vectors x_i and w_{ij} . The weights w_{ij} and thresholds v_{ij} may satisfy constraints $w_{ij} \in W_{ij}$, $v_{ij} \in V_{ij}$. Here notation like \mathbb{R}^n is used for n -dimensional Euclidian vector space.

Note that activation functions themselves can be random, for example, neurons can accidentally fall into the so-called sleep (drop out) state, i.e. produce a zero output signal: $g_i^j(x_i, w_{ij}, v_{ij}, \omega_{ij}) = \omega_{ij} \cdot g_i^j(x_i, w_{ij}, v_{ij})$, where ω_{ij} is an additional random parameter taking values 1 or 0 with probabilities p_{ij} and $1 - p_{ij}$. We assume that the random parameters $\{\omega_{ij}\}$ are independent and combined into a common vector $\omega = \{\omega_{ij}\}$ that takes values from a finite set Ω .

In what follows, we assume that the activation functions $g_i^j(x_i, w_{ij}, v_{ij}, \omega_{ij})$ of neurons $j = 1, \dots, n_i$ in each layer i for any fixed value of ω_{ij} are generalized differentiable over their variables (x_i, w_{ij}, v_{ij}) in the sense of the following definition, which covers all practical examples.

Definition 1 [8 – 10]. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$ is called generalized differentiable at point $\bar{z} \in \mathbb{R}^n$, if in some ε -neighborhood $\{z \in \mathbb{R}^n : \|z - \bar{z}\| < \varepsilon\}$ of the point \bar{z} it is defined an upper semicontinuous at \bar{z} multivalued mapping $\partial f(\cdot)$ with convex compact values $\partial f(z)$ and such that the following expansion holds true:

$$f(z) = f(\bar{z}) + \langle d, z - \bar{z} \rangle + o(\bar{z}, z, d),$$

where $d \in \partial f(z)$, $\langle \cdot, \cdot \rangle$ denotes a scalar product of two vectors, and the remainder term $o(\bar{z}, z, d)$ satisfies the condition: $\lim_{k \rightarrow \infty} o(\bar{z}, z^k, d^k) / \|z^k - \bar{z}\| = 0$ for all sequences $d^k \in \partial f(z^k)$, $z^k \rightarrow \bar{z}$ as $k \rightarrow \infty$. A function f is called generalized differentiable if it is generalized differentiable at each point $\bar{z} \in \mathbb{R}^n$; the mapping $\partial f(\cdot)$ is called the generalized gradient mapping of the function f ; the set $\partial f(z)$ is called a generalized gradient set of the function $f(\cdot)$ at point z ; vectors $d \in \partial f(z)$ are called generalized gradients of the function $f(\cdot)$ at point z .

Properties of generalized differentiable functions were studied in details in [8 – 10]. In particular, it is shown that they are locally Lipschitzian and their Clarke subdifferential $\partial_c f(z)$ can serve as a generalized gradient mapping $\partial f(z)$ of $f(\cdot)$, and it always holds $\partial_c f(z) \subseteq \partial f(z)$. This class of functions contains continuously differentiable, convex and concave functions and is closed with respect to finite maximum, minimum, superposition operations, and also with respect to taking the mathematical expectation.

Suppose there is a (training) set $\{(x_1^s \in \mathbb{R}^{n_0}, y_{m+1}^s \in \mathbb{R}^{n_m}), s = 1, \dots, S\}$ of observations of a network inputs-outputs. The standard training (identification) task for the network with the training quality criterion $\varphi(x_{m+1}^s, y_{m+1}^s)$ and regularization is as follows:

$$J(\{w_{ij}\}, \{v_{ij}\}) = \mathbf{E}_\omega \frac{1}{S} \sum_{s=1}^S \varphi(x_{m+1}^s, y_{m+1}^s) + \lambda \sum_{i=1}^m \sum_{j=1}^{n_i} \left(\|w_{ij}\|^\alpha + |v_{ij}|^\alpha \right) \rightarrow \min_{\{w_{ij} \in W_{ij}\}, \{v_{ij} \in V_{ij}\}}, \quad (1)$$

where $x_{m+1}^s \in \mathbb{R}^{n_k}$ is the vector of outputs of the last network layer for a training example s ; $y_{m+1}^s \in \mathbb{R}^{n_k}$ is a known, generally speaking, multidimensional vector of observations of the network outputs; $\|w_{ij}\|$ denotes a norm of the vector w_{ij} ; $\lambda > 0$, $\alpha \geq 1$; \mathbf{E}_ω is the mathematical expectation operator over ω ; the sequence of layers' outputs $\{x_i^s = (x_{i1}^s, \dots, x_{in_i}^s)^T, i = 2, \dots, m+1\}$ for a given first layer input $x_1^s \in \mathbb{R}^{n_0}$ is given by the relations

$$x_{(i+1)j}^s = g_i^j(x_i^s, w_{ij}, v_{ij}, \omega_{ij}), \quad j = 1, \dots, n_i; \quad i = 1, \dots, m. \quad (2)$$

Moreover, the training examples may contain not only the input and output of a network (for example, features and labels of objects) $\{(x_1^s, y_{m+1}^s), s = 1, \dots, S\}$, but also may include additional intermediate features $y_i^s \in \mathbb{R}^{n_i}$, $i \in I \subset \{2, \dots, m\}$, which can be used to improve the learning of the intermediate layers of the network, i.e. training examples may take the form of sequences $\{(x_1^s, \{y_i^s, i \in I\}, y_{m+1}^s), s = 1, \dots, S\}$. That's why, next we consider the following general network training task:

$$J(u) = \mathbf{E}_\theta \sum_{i=1}^m f_i(x_i(\theta), u_i) + \mathbf{E}_\theta f_{m+1}(x_{m+1}(\theta)) \rightarrow \min_{u \in U} \quad (3)$$

subject to constraints (satisfied for all values of the random parameter $\theta \in \Theta$):

$$x_{i+1}(\theta) = g_i(x_i(\theta), u_i, \theta) = \left\{ g_i^j(x_i(\theta), u_{ij}, \theta) \right\}_{j=1}^{n_i}, \quad i = 1, \dots, m; \quad x_1(\theta) \in \mathbb{R}^{n_0}. \quad (4)$$

Here $u = (u_1, \dots, u_m) \in \mathbb{R}^l$ ($l = \sum_{i=1}^m n_i$) is the vector of all adjusted parameters ($u = (\{w_{ij}\}, \{v_{ij}\})$ for problem (1), (2)); $x_i = (x_{i1}, \dots, x_{in_i})^T$ is the input vector for neurons in layer i ; u_{ij} is the vector of the adjusted parameters of neuron (i, j) ; $u_i = \{u_{ij}\}_{j=1}^{n_i}$ is the vector of the adjusted parameters of all neurons in layer i ; g_i^j is the activation function of neuron j in layer i ; $g_i = \{g_i^j\}_{j=1}^{n_i}$ is the vector activation function of the neurons in layer i ; $x_1(\theta) \in \mathbb{R}^{n_0}$ is a random vector of input signals to the network; θ is a random vector parameter, that defines the distribution of input signals and

influences on propagation of signals through the network ($\theta = (s, \omega)$ for problem (1), (2)); \mathbf{E}_θ denotes the sign of the mathematical expectation over θ .

Assumption. Suppose that, in problem (3), (4), the functions $f_i(x_i, u_i)$, $g_i^j(x_i, u_{ij}, \theta)$ and $f_{m+1}(x_{m+1})$ are generalized-differentiable over the totality of their arguments, respectively, over (x_i, u_i) , (x_i, u_{ij}) , and x_{m+1} (for fixed θ). Here, the activation function $g_i^j(x_i, u_{ij}, \theta)$ can be of a general form, i.e. optionally, the function g_i^j may depend not on all elements of the vector x_i and the dimension of the vector of the adjustable parameters u_{ij} may not coincide with the dimension of the vector of inputs x_i . The random parameter $\theta \in \Theta$ is a random variable defined on some probability space.

Note that in the literature (see, for example, [6]), for the purpose of training neural networks, it is proposed to use (stochastic) Clarke subgradients of the risk functional $J(u)$, but these subgradients are relatively simple to calculate only for subdifferentially regular Lipschitz functions [7, §2.3, §2.7], and for general nonconvex nonsmooth functions, their calculation is a problem.

For arbitrary generalized differentiable (over the totality of variables) vector functions $g_i(x, u) \in \mathbb{R}^{n_i}$ with arguments $x = (x^1, \dots, x^n)^T \in \mathbb{R}^n$, $u = (u^1, \dots, u^l)^T \in \mathbb{R}^l$, we denote the matrices:

$$g_{ix} = \begin{pmatrix} g_{ix^1}^1 & \dots & g_{ix^n}^1 \\ \dots & \dots & \dots \\ g_{ix^1}^{n_i} & \dots & g_{ix^n}^{n_i} \end{pmatrix} = \begin{pmatrix} g_{ix}^1 \\ \dots \\ g_{ix}^{n_i} \end{pmatrix}, \quad g_{iu} = \begin{pmatrix} g_{iu^1}^1 & \dots & g_{iu^l}^1 \\ \dots & \dots & \dots \\ g_{iu^1}^{n_i} & \dots & g_{iu^l}^{n_i} \end{pmatrix} = \begin{pmatrix} g_{iu}^1 \\ \dots \\ g_{iu}^{n_i} \end{pmatrix};$$

and for arbitrary generalized differentiable (over the totality of arguments) scalar functions $f_i(x, u)$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^l$, and $f_{m+1}(x)$, $x \in \mathbb{R}^n$, let us introduce vectors

$$(f_{ix})^T = (f_{ix^1}, \dots, f_{ix^n}), \quad (f_{iu})^T = (f_{iu^1}, \dots, f_{iu^l}), \quad (\varphi_{kx})^T = (\varphi_{kx^1}, \dots, \varphi_{kx^n}),$$

where $(f_{ix}, f_{iu})^T$, $(g_{ix}^j, g_{iu}^j)^T$ are some generalized gradients of the functions $f_i(\cdot, \cdot)$, $g_i^j(\cdot, \cdot, \theta)$; $f_{(m+1)x}(\cdot)$ is some generalized gradient of the function $f_{m+1}(\cdot)$; the expression $(\cdot)^T$ denotes the transposition of matrix (\cdot) .

The next theorem exploits the similarity between optimal control problems for discrete dynamical systems [11, 12] and multilayer neural networks and formalizes a method for calculating stochastic generalized gradients in the problem of training a nonconvex nonsmooth neural network. It extends the well-known method of ‘‘backpropagation of the error’’ (BackProp) [1, 4, 5] to nonconvex nonsmooth learning problems.

Theorem 1. Under the assumptions made, the objective function $J(u)$ of problem (3), (4) is generalized differentiable with respect to variables $u = (u_1 \in \mathbb{R}^{n_1}, \dots, u_m \in \mathbb{R}^{n_m})$, and the vectors

$$h_u(u, \theta) = \left(h_{1u_1}(x_1(\theta), \psi_1(\theta), u_1), h_{2u_2}(x_2(\theta), \psi_2(\theta), u_2), \dots, h_{mu_m}(x_{mm}(\theta), \psi_m(\theta), u_m) \right)^T \quad (5)$$

are stochastic generalized gradients of the function $J(u)$ at a given point u , i.e., $\mathbf{E}_\theta h_u(u, \theta) \in \partial J(u)$, where $h_i(x_i, \psi_i, u_i) = f_i(x_i, u_i) + g_i^T(x_i, u_i) \cdot \psi_i$, $i = 1, \dots, m$, is a discrete (over i) Hamilton-Pontryagin function; the vector

$$\begin{aligned} h_{iu_i}(x_i(\theta), \psi_i(\theta), u_i) &= \left(h_{iu_i^1}(x_i(\theta), \psi_i(\theta), u_i), \dots, h_{iu_i^{n_i}}(x_i(\theta), \psi_i(\theta), u_i) \right)^T = \\ &= \left(f_{iu_i^1}(x_i(\theta), u_i) + g_{iu_i^1}^T(x_i(\theta), u_i) \cdot \psi_i(\theta), \dots, f_{iu_i^{n_i}}(x_i(\theta), u_i) + g_{iu_i^{n_i}}^T(x_i(\theta), u_i) \cdot \psi_i(\theta) \right)^T \in \mathbb{R}^{n_i} \end{aligned} \quad (6)$$

is the u_i -component of a generalized gradient of function $h_i(\cdot, \psi_i, \cdot)$, $i = 1, \dots, m$; $x(\theta) = (x_1(\theta), \dots, x_{m+1}(\theta))$ is a discrete random trajectory of process (4), corresponding to the given sequence of parameters $(u_1, \dots, u_m) = u$ and the random initial data $x_1(\theta) \in \mathbb{R}^{n_0}$. Here, the random sequence of auxiliary (conjugate) vector functions $(\psi_1(\theta), \dots, \psi_m(\theta)) = \psi(\theta)$ is determined through the backpropagation equations (adopted from the Pontryagin maximum principle): $\psi_m(\theta) = f_{(m+1)x_{m+1}}(x_{m+1}(\theta))$,

$$\psi_{i-1}(\theta) = h_{ix_i}(x_i(\theta), \psi_i(\theta), u_i) = f_{ix_i}(x_i(\theta), u_i) + g_{ix_i}^T(x_i(\theta), u_i) \cdot \psi_i(\theta), \quad i = m, m-1, \dots, 2; \quad (7)$$

$(f_{ix}(x_i(\theta), u_i), f_{iu}(x_i(\theta), u_i))^T$, $(g_{ix}^j(x_i(\theta), u_i), g_{iu}^j(x_i(\theta), u_i))^T$ are some generalized gradients of the functions $f_i(\cdot, \cdot)$, $g_i^j(\cdot, \cdot, \theta)$ at the point $(x_i(\theta), u_i)$ and $f_{(m+1)x}(x_{m+1}(\theta))$ is some generalized gradient of the function f_{m+1} at the point $x_{m+1}(\theta)$, which are used in (6), (7).

Proof. Note that process (4) can be formally treated as a stochastic dynamic system [12, 13] in the discrete time $i = 1, \dots, m+1$ with states x_i , control parameters u_i , a given initial state $x_1(\theta)$, and optimality criterion (3). The stochasticity of system (4) is generated by the random input $x_1(\theta)$, a random mechanism of dropping out of neurons, and possibly other factors. Using relations (4), the vectors x_i , $i = 2, \dots, m+1$, can be sequentially excluded from the formulation of the optimization problem (3). Then under the sign of summation in (3) there remains some complex composite function $f(u, \theta) = \sum_{i=1}^m f_i(\tilde{x}_i(u_1, \dots, u_{i-1}, \theta), u_i) + f_{m+1}(\tilde{x}_{m+1}(u_1, \dots, u_m, \theta))$, which depends on optimization variables u and where $\tilde{x}_i(u_1, \dots, u_{i-1}, \theta)$, $i = 2, \dots, m+1$, are complex compound functions of their arguments. And since the class of generalized differentiable functions is closed with respect to compositions, then under the made assumptions this function $f(u, \theta)$ becomes generalized differentiable with respect to u for each $\theta \in \Theta$. The expectation \mathbf{E}_θ (in this case, summation) does not move out from the class of generalized differentiable functions, therefore the function $J(u) = \mathbf{E}_\theta f(u, \theta)$ is also generalized differentiable with respect to u . Now, similar to the proofs of Theorems 6 - 8 from [13], applying the rules of differentiation of the sum, the chain rule of differentiation of complex generalized differentiable functions [8, 9] (which are analogues to the rules of differentiation of smooth and convex functions), and introducing auxiliary variables (7), after some algebraic

manipulations [13], we obtain formula (6) for stochastic generalized gradients $h_u(u, \theta)$ of the function $J(u)$.

Neural networks may have a complex and heterogeneous structure. However, introducing additional dummy neurons, such networks can be reduced to a canonical multilayer form and for them one can apply the formulas of Theorem 1.

Let us consider an important special case of networks in which the adjusted parameters are the same for each layer. For example, a network can consist of identical neurons, or some identical neurons are added to each layer in an already-trained network, or the network consists of duplicates of the same layer. Then, similarly to (3), (4), the training task consists in solving the following problem:

$$J(u) = \mathbf{E}_\theta \sum_{i=1}^m f_i(x_i(\theta), u) + \mathbf{E}_\theta f_{m+1}(x_{m+1}(\theta)) \rightarrow \min_{u=(u_1, \dots, u_l) \in U \subseteq \mathbb{R}^l}, \quad (8)$$

$$x_{i+1}(\theta) = g_i(x_i(\theta), u, \theta) = \left\{ g_i^j(x_i(\theta), u, \theta) \right\}_{j=1}^{n_i}, \quad i = 1, \dots, m; \quad x_1(\theta) \in \mathbb{R}^{n_0}. \quad (9)$$

Theorem 2. Under the assumptions made, the objective function $J(u)$ of problem (8), (9) is generalized differentiable with respect to variables $u = (u_1, \dots, u_l)$, and the vector

$$\left(\sum_{i=1}^m h_i(x_i(\theta), \psi_i(\theta), u) \right)_u = \left(\sum_{i=1}^m h_{iu_i}(x_i(\theta), \psi_i(\theta), u), \dots, \sum_{i=1}^m h_{iu_i}(x_i(\theta), \psi_i(\theta), u) \right)^T$$

is a stochastic generalized gradient of the function $J(u)$ at point u , i.e.

$\mathbf{E}_\theta \left(\sum_{i=1}^m h_i(x_i(\theta), \psi_i(\theta), u) \right)_u \in \partial J(u)$, where $h_i(x_i, \psi_i, u) = f_i(x_i, u) + g_i^T(x_i, u) \cdot \psi_i$, $i = 1, \dots, m$, is a discrete (over i) Hamilton-Pontryagin function; $x(\theta) = (x_1(\theta), \dots, x_{m+1}(\theta))$ is a discrete random trajectory of process (9), corresponding to the vector parameter u and the random initial data $x_1(\theta) \in \mathbb{R}^{n_0}$. Here, the random sequence of auxiliary (conjugate) vector functions $(\psi_1(\theta), \dots, \psi_m(\theta)) = \psi(\theta)$ is determined through the backpropagation equations: $\psi_m(\theta) = f_{(m+1)x_{m+1}}(x_{m+1}(\theta))$,

$$\psi_{i-1}(\theta) = h_{ix_i}(x_i(\theta), \psi_i(\theta), u) = f_{ix_i}(x_i(\theta), u) + g_{ix_i}^T(x_i(\theta), u) \cdot \psi_i(\theta), \quad i = m, m-1, \dots, 2.$$

The method of stochastic generalized gradient descent and its variants.

The stochastic gradient descent method [3] is the main method for training deep neural networks, firstly, because of enormous dimensions of such networks and, secondly, due to the regularizing properties of the method. The properties of the stochastic gradient method and its modifications were studied in details in cases of smooth and convex optimized functions [2, 3, 12]. In [9, 14, 15], this method and its modifications were substantiated for solving nonconvex nonsmooth stochastic programming problems (with generalized differentiable functions), and thus are applicable to solving nonconvex nonsmooth machine learning problems.

References

1. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. Cambridge, Massachusetts: The MIT Press, <http://www.deeplearningbook.org>
2. Bottou, L., Curtis, F.E., Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. SIAM Rev., 60, No. 2, pp. 223–311. Doi: <https://doi.org/10.1137/16M1080173>
3. Newton, D., Yousefian, F., Pasupathy, R. (2018). Stochastic Gradient Descent: Recent Trends. INFORMS TutORials in Operations Research, Published online: 19 Oct 2018, pp. 193-220. Doi: <https://doi.org/10.1287/educ.2018.0191>
4. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning Representations by Back-Propagating Errors. Nature, 323, pp. 533-536. Doi: <https://doi.org/10.1038/323533a0>
5. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, pp. 85–117. Doi: <http://dx.doi.org/10.1016/j.neunet.2014.09.003>

6. Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J.D. (2019). Stochastic subgradient method converges on tame functions. *Found. Comput. Math.*, pp. 1-36. Doi: <https://doi.org/10.1007/s10208-018-09409-5>
7. Clarke, F.H. (1990). *Optimization and nonsmooth analysis*, Vol. 5 of *Classics in Applied Mathematics*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2nd ed.
8. Norkin, V.I. (1980). Generalized differentiable functions. *Cybernetics*, 16, No. 1, pp. 10-12. Doi: <https://doi.org/10.1007/BF01099354>
9. Mikhalevich, V.S., Gupal, A.M., Norkin, V.I. (1987). *Methods of Nonconvex Optimization*, Moscow: Nauka. (In Russian).
10. Norkin, V.I. (1986). Stochastic generalized-differentiable functions in the problem of nonconvex nonsmooth stochastic optimization. *Cybernetics*, 22, No.6, pp. 804-809. Doi: <https://doi.org/10.1007/BF01068698>
11. Bryson, A.E., Ho, Y-C. (1969). *Applied Optimal Control. Optimization, Estimation, and Control*. Waltham, Massachusetts: Blaisdell Publishing Company, A Division of Ginn and Company.
12. Ermoliev, Y.M. (1976). *Methods of stochastic programming*. Moscow: Nauka.
13. Norkin V.I. (2019). Generalized gradients in problems of dynamic optimization, optimal control, and machine learning. Preprint. Kyiv: V.M.Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine. Access (2019.09.19): http://www.optimization-online.org/DB_HTML/2019/09/7374.html
14. Ermoliev, Y.M., Norkin, V.I. (1998). Stochastic generalized gradient method for solving nonconvex nonsmooth stochastic optimization problems, *Cybern. Syst. Anal.*, 34, No. 2, pp. 196-215. Doi: <https://doi.org/10.1007/BF02742069>

15. Ermoliev, Y.M., Norkin, V.I. (2003). Solution of nonconvex nonsmooth stochastic optimization problems. *Cybern. Syst. Anal.*, 39, No. 5, pp. 701-715. Doi: <https://doi.org/10.1023/B:CASA.0000012091.84864.65>