QUASI BLACK HOLE EFFECT OF GRADIENT DESCENT IN LARGE DIMENSION: CONSEQUENCE ON NEURAL NETWORK LEARNING

Anne Bouillard, Philippe Jacquet

Nokia Bell Labs

ABSTRACT

The gradient descent to a local minimum is the key ingredient of deep neural networks learning techniques. We consider a function $L_m(.)$ in dimension n with a random set of m absolute minima. When $\log m = o(n)$, we show that a gradient descent from an initial random point quasi always ends on a unique local minimum approximately at the centroid of the absolute minima. This fake minimum acts like an absorbing node, but its value by function $L_m(.)$ can be far above the values obtained by $L_m(.)$ on the absolute minima and sometimes gives very bad coefficients for the neural network. Fortunately in most cases the fake minimum leads to a neural network with not so bad prediction, with an error rate of order $n^{-1/4}$. The only way to escape the fake minimum is to start a new gradient descent from a new random point and we show that finding a good initial point takes in average time which is at least proportional to e^{bn}/mn^2 for some $b > 0$.

Index Terms— gradient descent, large dimension, stochastic geometry

1. INTRODUCTION

Deep Neural Networks (DNN) have created a revolution in artificial intelligence [5, 6]. The list of their successful applications is so impressive that there is no need to stress the importance of this revolution. Clearly one can pretend that DNNs have successfully passed the Turing test [2] in its perfection in imitating and sometimes superceding human intelligence. Beyond the euphoria around this incontestable breakthrough one can start thinking about its limits.

Surprisingly the limit may lay in the intrication of machine learning with the machine. In short can a DNN mimic a machine as well it can mimic a human intelligence? To be clearer, can a DNN be trained to solve problem that a simple algorithm would solve in an optimal way? We know that a DNN is a Turing complete machine [12, 13] and any algorithm can be implemented by suitably tuning the weights in the neurons. But the question is not in programming a DNN but in training and if so can we do it in a reasonable time and complexity [9]?

Consider the problem of finding the maximum of N numbers. A DNN with $O(\log N)$ layers involving $2N$ ReLUs (Rectified Linear Units) can solve the problem by extrapolating the DNN shown in Figure 1. There are $\Theta(N^2)$ coefficients. An optimal DNN will use a divide-and-conquer strategy and partition the N numbers into $N/2$ pairs, then $N/4$ pairs, etc. There are $\frac{N!}{2^N}$ possibilities thus $m = \exp(N(\log N - \log 2 - 1) + o(N))$ absolute minima. The gradient descent of the deep learning operates in a space of dimension $n = \Theta(N^2)$. Experiments show that the neural network does not converge, or converges to a wrong minimum. In fact insisting on the gradient descent, the convergence to an absolute minimum takes an $O(m)$ time which is more than exponential in N. The aim of the paper is to attempt an explanation of this phenomenon via a simplified model.

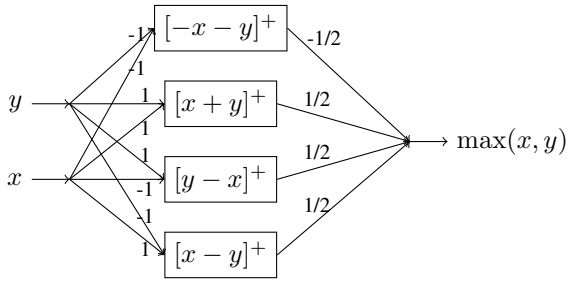

Fig. 1. NN for computing the maximum of two numbers.

1.1. DNN coefficients, gradient and invariant

A DNN made of k layers of respective size $N, N_1, \ldots N_k$ and can be represented by a sequence of rectangular matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$. Matrix $\mathbf{M}_i$ has dimension $N_{i-1} \times N_i$ with $N_0 = N$ and $N + k = 1$. An input vector $\mathbf{t}$ of dimension N produces a label $f(\mathbf{t})$ defined by the following operations:

$$f(\mathbf{t}) = \mathbf{t}^\top \mathbf{M}_1 \mathbf{I}^+(\mathbf{a}_1)\mathbf{M}_2\mathbf{I}^+(\mathbf{a}_2)\cdots\mathbf{I}^+(\mathbf{a}_{k-1})\mathbf{M}_k \qquad (1)$$

where the $\mathbf{a}_i$ are bias vectors of dimension N_i and with $\mathbf{I}^+(\mathbf{a})$ the operator which change each component x_i of a vector $\mathbf{x} \in \mathbb{R}^n$ of a vector into $[x_i + \alpha_i]^+ = \max(x_i + \alpha_i, 0)$ with $\mathbf{a} = (\alpha_1, \ldots, \alpha_n)$. To simplify our presentation we will hereafter that all bias vectors are null as in figure 1.

We can write the $N_0 \times N_1 + \cdots + N_{k-1} \times N_k + N_k$ coefficients of the matrix by a vector $\mathbf{z} \in \mathbb{R}^n$ with $n = N_0 \times N_1 + \cdots + N_{k-1} \times N_k + N_k$. We define $f(\mathbf{t}, \mathbf{z})$ the function $f(\mathbf{t})$ applied to the test vector $\mathbf{t}$ when the coefficients of the matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$ are defined by the vector $\mathbf{z}$ which summarizes all the coefficients of the sequence of the matrices in $(\mathbf{M}_1, \cdots, \mathbf{M}_k)$.

The principle of the DNN is at each vector test $\mathbf{t}$ we compare the prediction $f(\mathbf{t}, \mathbf{z})$ with the ground truth label $\bar{f}(\mathbf{t})$, the one we would get from a human expert as the principle of supervised machine learning [10, 11]. The loss function can be expressed as $L(\mathbf{t}, \mathbf{z}) = (f(\mathbf{t}, \mathbf{z}) - \bar{f}(\mathbf{t}))^2$ or any increasing function of it. Computing the gradient $\nabla L(\mathbf{t}, \mathbf{z})$ with respect to the coefficients vector $\mathbf{z}$ we update the later via the following "normalized" gradient descent step:

$$\mathbf{z} \leftarrow \mathbf{z} - r \frac{1}{\|\nabla L(\mathbf{t}, \mathbf{z})\|} \nabla L(\mathbf{t}, \mathbf{z}), \tag{2}$$

where r is the learning rate. There are many variant of the gradient descent, for more details see [4, 8, 3].

1.1.1. Homothetic invariants

If we define the DNN by $\mathbf{M}_1 \mathbf{I}^+_{N_1} \mathbf{M}_2 \mathbf{I}^+_{N_2} \cdots \mathbf{I}^+_{N_{k-1}} \mathbf{M}_k$, clearly the DNN defined by $\mathbf{M}'_1 \mathbf{I}^+_{N_1} \mathbf{M}'_2 \mathbf{I}^+_{N_2} \cdots \mathbf{I}^+_{N_{k-1}} \mathbf{M}'_k$ gives the same label function when $\mathbf{M}'_i = \tau_i \mathbf{M}_i$ where $(\tau_1, \ldots, \tau_k)$ is a tuple of strictly positive real numbers such that $\tau_1 \cdots \tau_k = 1$. Similarly we still have the same label function when $\mathbf{M}'_i = \Delta_{i-1} \mathbf{M}_i \Delta_i^{-1}$ where the Δ_i are diagonal matrices with strictly positive coefficients on the diagonal (assuming $\Delta_0 = \mathbf{I}_N$ and $\Delta_k = 1$.

Going back to the vector notation of DNN, the set of vector $\mathbf{z}$ which provide an homothetic equivalent of a vector $\mathbf{z}$ is called the homothetic class of $\mathbf{z}$. An homothetic class is a topologically connected subset of $\mathbb{R}^n$. There is a closed formula (not given here) which expresses a function $g(\mathbf{z})$ which gives an unique element of the class of $\mathbf{z}$.

Let $\mathbf{K}_n = g(\mathbb{R}^n)$ be a Riemann variety of dimension $n - \sum_i N_i - k + 1$ where DNNs coefficients are represented without homothetic invariant.

1.1.2. Permutation invariant

Another invariant is in the permutation of the lines and columns in the matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$. Indeed let $(\mathbf{J}_1, \ldots, \mathbf{J}_{k-1})$ be a sequence of permutation matrices of respective size $N_1 \times N_1, \ldots, N_{k-1} \times N_{k-1}$. We still have the same label function if for all integers i $\mathbf{M}'_i = \mathbf{J}_{i-1} \mathbf{M}_i \mathbf{J}_i^{-1}$ (by default $\mathbf{J}_0$ and $\mathbf{J}_k$ are identity matrices). This defines the permutation class of a vector $\mathbf{z}$ which corresponds to certain permutations of its coefficients.

It is important to notice that the permutation class of a vector $\mathbf{z}$ is *not* a topologically connected set, even combined with the homothetic classes. There are $m = N_1! \cdots N_{k-1}!$ possible permutations, and they map each vector $\mathbf{z}$ into several mirror images $\mathbf{z}'$. The multiplicity of the mirror images will be the cause of the troubles we will notice in the gradient descents. We notice that although m is large, it is not exponential in n. Indeed $\log(m) = o(n)$ when n tends to infinity.

2. THE MODEL AND RESULT

In our model we will consider that the DNN is of no risk type, that is, the label is a deterministic function of the test vector $\mathbf{t}$ and that the function which produces this label is a DNN of the same size. In other words, there exists a vector $\mathbf{b}$ which produces the exact label, *i.e.* for every test vector $\mathbf{t}$:

$$\bar{f}(\mathbf{t}) = f(\mathbf{t}, \mathbf{b})$$

We notice that for simple algorithmic problems such as finding the maximum, the DNN is implementable by a neural network as long it has enough layers ($\log_2 N$ extrapolated from Figure 1).

In the following, we will take the following simplifying assumptions on the space and the distribution of the global minima, and on the loss function:

1) We ignore the Riemannian aspect of $\mathbf{K}_n = g(\mathbb{R}^n)$ which we simply let equal to $\mathbb{R}^n$.

2) The global minima $\mathbf{b}_1, \ldots \mathbf{b}_m$ are taken uniformly at random in the hypercube : we forget that they are m mirror images obtained by permutations of some coefficients, which has in fact marginal importance. In fact one could take a more general distribution for the $\mathbf{b}'_i s$ in $\mathbb{R}^n$, such as Gaussian [7], *etc*.

3) We modify the loss function. Clearly the $\mathbf{b}_i$ are systematic roots of the loss function $L(\mathbf{t}, \mathbf{z}) = (f(\mathbf{t}, \mathbf{z}) - f(\mathbf{t}, \mathbf{b}))^2$. To simplify we skip the test vectors $\mathbf{b}$ and define $L_m(\mathbf{z}) = \prod_{i=1}^m \|\mathbf{z} - \mathbf{b}_i\|^2$. The points $\mathbf{b}_i$ are the global minima of the function $L_m(\mathbf{z})$. The result we will show also holds for any increasing function of $L_m(\mathbf{z})$. However, the result may not hold for every function of the form $f(\mathbf{z} - \mathbf{b}_1, \ldots, \mathbf{z} - \mathbf{b}_m)$, *e.g.* $\min_i \{\|\mathbf{z} - \mathbf{b}_i\|\}$.

That way the problem is indeed within the stochastic geometry domain in large dimension.

Our problem is now generalized (although simplified) to the problem of gradient descent in large dimension when global minima are a Poisson point process. We will show the following theorem:

Theorem 1 *A local minimum exists at $\mathbf{b}^*$ which is close to the centroid of the points $\mathbf{b}_1, \ldots, \mathbf{b}_m$ and with high probability the gradient descent converges to it, assuming that n and m tend both to infinity with $\log m = o(n)$.*

In the following section we will provide a strategy of proof. In the next section we will show some experiments and discuss the impact on DNNs. Unfortunately due to the lack of time the experiments have not yet produced on real DNNs.

3. STRATEGY OF PROOF

There are two main steps in the proof: first we show that the gradient descent is nearly directed toward the origin with high probability, and then that there is a local minimum in a neighborhood of the origin. These two steps respectively involve the computation of the gradient and the second derivative of L_m:

$$\begin{aligned}\frac{1}{L_m(\mathbf{z})}\nabla L_m(\mathbf{z}) &= \sum_{i=1}^m \frac{2}{\|\mathbf{z}-\mathbf{b}_i\|^2}(\mathbf{z}-\mathbf{b}_i),\\ \frac{1}{L_m(\mathbf{z})}\nabla^2 L_m(\mathbf{z}) &= \sum_{i=1}^m \frac{2}{\|\mathbf{z}-\mathbf{b}_i\|^2}\\ &\quad\left(\mathbf{I}_n - \frac{2}{\|\mathbf{z}-\mathbf{b}_i\|^2}(\mathbf{z}-\mathbf{b}_i)\otimes(\mathbf{z}-\mathbf{b}_i)\right)\\ &\quad + \frac{1}{L_m(\mathbf{z})}\nabla L_m(\mathbf{z})\otimes\nabla L_m(\mathbf{z}),\end{aligned}$$

where $\otimes$ is the tensor product.

3.1. The gradient analysis

We denote $D_m(\mathbf{z}) = \sum_{i=1}^m \frac{1}{\|\mathbf{z}-\mathbf{b}_i\|^2}$, and fist show that the gradient is directed to the origin with high probability.

Lemma 1 *For all $c_1 > 0$ and for all $\mathbf{z} \in \mathbf{K}_n$ there exists $b_1 > 0$ such that for all n and m with $\log(m) = o(n)$,*

$$P\left(\|\frac{1}{L_m(\mathbf{z})}\nabla L_m(\mathbf{z}) - D_m(\mathbf{z})\mathbf{z}\| > \frac{c_1 m}{\sqrt{n}}\right) < me^{-b_1 n}.$$

The proof, as most as most of the further proofs, is based on the Chernoff bounds [1].

Now we consider a normalized gradient descent defined by a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ in $\mathbf{K}_n$ such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{r}{\|\nabla L_m(\mathbf{x}_k)\|}\nabla L_m(\mathbf{x}_k),$$

where $r > 0$, the step size, is fixed. For $d > 0$, let $\mathcal{B}_n(d)$ denote the ball of radius d centered on origin.

Theorem 2 *From any given starting point $\mathbf{x}_0 \in \mathbf{K}_n$, and for any $d > 0$ and for a step size $r > 0$ fixed, with probability larger than $1 - O(\frac{\sqrt{n}}{r}me^{-bn})$ the normalized gradient descent ends in $\mathcal{B}_n(d\sqrt{n})$.*

Proof Let $\mathbf{x}_0 \notin \mathcal{B}_m(d\sqrt{n})$. Since for all $\mathbf{z} \notin \mathcal{B}_m(d\sqrt{n})$, $\|\mathbf{z}\| \geq d\sqrt{n}$, from Lemma 1 the gradient $\nabla L_m(\mathbf{z})$ is colinear to $\mathbf{z} + \mathbf{q}$ with $\|\mathbf{q}\| \leq c_1\frac{m}{\sqrt{n}D_m(\mathbf{z})} = O(\frac{c_1}{d}\|\mathbf{z}\|)$. Since c_1 can be as small as possible, $\nabla L_m(\mathbf{z})$ is practically oriented toward $\mathbf{z}$ with a negligible angle. Thus with probability greater than $1 - me^{-bn}$, we have $\mathbf{x}_1$ close to $\mathbf{x}_0 - r\frac{\mathbf{x}_0}{\|\mathbf{x}_0\|}$. In fact we have $\|\mathbf{x}_1\| \leq \|\mathbf{x}_0\| - r(1-c_1)$. Consequently we need at most $\frac{\|\mathbf{x}_0\| - d\sqrt{n}}{r(1-c_1)} = O(\sqrt{n}/r)$ steps to get to the black hole $\mathcal{B}_m(d\sqrt{n})$. Of course this happens if the conditions of Lemma 1 apply at each step, which occurs with probability higher than $1 - O(\frac{\sqrt{n}}{r}me^{-bn})$.

Corollary 1 *For all sequences $\epsilon_n \to 0$, for all $d > 0$ and $r > 0$ the gradient descent would not leave $\mathcal{B}_n(d\sqrt{n} + r)$ before $\epsilon_n\frac{e^{bn}}{m}$ steps with high probability.*

3.2. Second derivative analysis

The previous results are resilient to generalization, for example if $\nabla L_m(\mathbf{z})$ is proportional to $\sum_{i=1}^m \frac{\mathbf{z}-\mathbf{b}_i}{\|\mathbf{z}-\mathbf{b}_i\|_k^\alpha}$ for some k and $\alpha > 0$.

The next results are probably more dependent of the expression of $L_m(\mathbf{z})$ via the Euclidean norm $\|.\|$.

Next, Lemma 2 shows that with high probability the loss function is convex on a neighborhood of the origin, and Lemma 3 that there exists a local minimum in the interior of this neighborhood.

We define the inequality between $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$: $A \preceq B$ if for all $\mathbf{x} \in \mathbb{R}^n$: $\langle \mathbf{x}, \mathbf{A}\mathbf{x}\rangle \leq \langle \mathbf{x}, \mathbf{B}\mathbf{x}\rangle$.

Lemma 2 *There exists $d_1 > 0$, $c_2 > 0$ and $b > 0$ such that for all $\mathbf{z}$ such that $\|\mathbf{z}\|^2 < d_1 n$:*

$$P(\frac{1}{L_m(\mathbf{z})}\nabla^2 L_m(\mathbf{z}) \succeq c_2\frac{m}{n}\mathbf{I}_n) \geq 1 - me^{-bn}.$$

Lemma 3 *For all $d > 0$ small enough there exists $b > 0$ such with probability greater than $1 - mne^{-bn}$ we have $\forall \mathbf{z} \in \mathbf{K}_n$:*

$$\|\mathbf{z}\| = d\sqrt{n} \Rightarrow \langle \nabla L_m(\mathbf{z})\mathbf{z}\rangle > 0.$$

One can deduce the following theorem from these lemmas. Note that the local minimum is unique since since the $L_m(.)$ is convex on $\mathcal{B}_n(d\sqrt{n})$ as soon as $d < d_2$.

Theorem 3 *With probability greater than $1 - me^{-bn}$, the function $L_m(\mathbf{z})$ is strictly convex on the $\mathcal{B}_m(d_1\sqrt{n})$ and for all $0 < d_2 < d_1$ there exists a local minimum $\mathbf{b}^*$ in $\mathcal{B}_n(d_2\sqrt{n})$.*

The consequence of this theorem is that the normalized gradient descent stays in $\mathcal{B}_n(d\sqrt{n})$ forever with high probability.

This is not necessarily the case for the supervised gradient descent which may experience unbounded jumps when $\mathbf{x}_k$ comes close to $\mathbf{b}^*$.

Of course all these results have a trivial generalization when

- the distribution of the coefficients are not uniform;
- $\mathbf{E}[\mathbf{b}_i] \neq 0$ (*i.e.* the $\mathbf{b}^*$ does not converge to zero);
- the coefficients in each $\mathbf{b}_i$ have mild dependencies;
- the vectors $\mathbf{b}_i$'s have mild dependencies.

And also a less trivial generalization when

- the norm is the general $\|\cdot\|_\ell$ for $\ell \geq 2$;
- the norm is $\|\cdot\|_\infty$ we have the component-wise convergence.

4. EXPERIMENTATION

Figure 2 show the normalized gradient descent (see Equation (2)) in dimension 10 with $m = 4$ (top), $m = 10$ (bottom) We have set $r = 0.01$. The number of global minima is too small and the gradient descents converge to them. For $m = 10$ the gradient descents end all on the quasi centroid $\mathbf{z}^*$. For the purpose of these figures we set $\mathbf{K}_n = [0, 1]^n$.

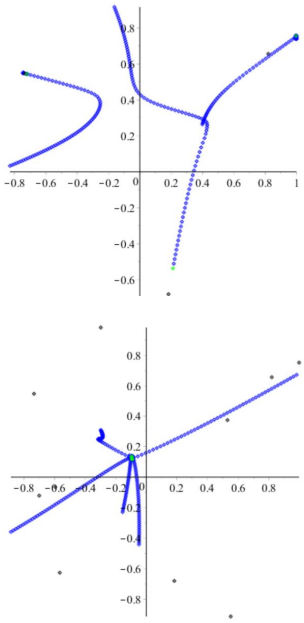

Fig. 2. Normalized gradient descents starting from three random points for $n = 10$ and $m = 4$ (top) and $m = 10$ (bottom), projected on the two first dimensions. Global minima are in black, ending point are green.

5. CONSEQUENCE ON DNN PERFORMANCE

5.1. Tuning of the model to mirror image effect

To take into account the mirror image phenomenon we can assume that a vector $\mathbf{b}$ is selected at random and that the m mirror images $\mathbf{b}_i$ are obtained by a subgroup of coordinate permutations: $\mathbf{b}_i = \sigma_i(\mathbf{b})$. The entropy of global minima set then drops from $O(nm)$ to $O(n)$. Theorem 2 is still valid since the lemmas which support it are all based on a single probabilistic instance of $\mathbf{b}_i$:

$$\mathbf{P}(\left|\|\mathbf{z} - \mathbf{b}_i\|^2 - \|\mathbf{z}\|^2 - n/3\right| > cn) < e^{-bn},$$

and then it is extended to

$$\mathbf{P}(\forall i : \ \left|\|\mathbf{z} - \mathbf{b}_i\|^2 - \|\mathbf{z}\|^2 - n/3\right| > cn) < me^{-bn}$$

indifferently of the correlation between the $\mathbf{b}_i$'s.

5.2. Non zero average and zero average coefficients

The following part is extrapolated from the previous part but is only conjectured since the transition from simplified function loss gradient descent to actual random DNNs is not yet proved. A DNN made of k layers of respective size $N, N_1, \ldots N_k$ and can be represented by a sequence of rectangular matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$ as shown in Equation (1). The DNN accepts input vectors of dimension N.

Let us make again the simplified assumption that each coefficient of the matrices is produced at random but with a non zero average $\mathbf{E}[b] \neq 0$ and variance $\mathbf{v}(b)$.

According to our main results on gradient descent in large dimension the learning should converge to a DNN whose matrices are the average of the $\mathbf{M}_i$, *i.e.* corresponding to the label function

$$f^*(\mathbf{t}) = \mathbf{t}^\top \mathbf{E}[\mathbf{M}_1]\mathbf{I}^+_{N_1}\mathbf{E}[\mathbf{M}_2]\mathbf{I}^+_{N_2}\cdots\mathbf{I}^+_{N_{k-1}}\mathbf{E}[\mathbf{M}_k].$$

We have $\mathbf{E}[\mathbf{M}_i] = \mathbf{E}[b]\mathbf{1}_{N_{i-1}} \otimes \mathbf{1}_{N_i}$ where $\mathbf{1}_N$ is the vector of dimension N made of 1's. With the exception that $\mathbf{E}[\mathbf{M}_1]$ is of the form $\mathbf{u} \otimes \mathbf{1}_{N_1}$ where $\mathbf{u}$ is a vector of dimension N because the permutations of the rows in the first matrix are not permitted in the permutation class of $\mathbf{b}$.

Using the Gaussian limit of the sum of independent variable we arrive to the estimate that

$$f^*(\mathbf{t}) = f(\mathbf{t})\left(1 + \frac{1}{\mathbf{E}[b]}O(\sum_{i=0}^{k-1}\sqrt{\mathbf{v}(b)/N_i})\right).$$

In other words such a random DNN has an error rate of order $\frac{k}{n^{1/4}}$. However, this is obtained via the hypothesis that the coefficients of the matrices are "typical" of a random selection.

In the case where $\mathbf{E}[b] = 0$ or when some $\mathbf{E}[\mathbf{M}_\ell] = 0$ then there will be an unbounded error rate. This is unfortunately the case with many algorithmic problem such as the maximum extraction. It is easy to see that $\mathbf{E}[\mathbf{M}_1] = 0$. To get a satisfactory convergence to global minima, one has to wait for an exponential of iterations.

Unfortunately due to lack of time we did not perform enough simulations to validate those conjectures on large scale DNN's with large training sets.

6. CONCLUSION

We have shown that a gradient descent in large dimension with a large number of global minima always ends to a fake minimum located close to the centroid of the global minima. This may explain why the DNNs have difficulties to reproduce the output of simple algorithms.

7. REFERENCES

[1] Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. The Annals of Mathematical Statistics, 23(4), 493-507.

[2] Turing, A. M. (2009). Computing machinery and intelligence. In Parsing the Turing Test (pp. 23-65). Springer, Dordrecht.

[3] Mason, L., Baxter, J., Bartlett, P. L., & Frean, M. R. (2000). Boosting algorithms as gradient descent. In Advances in neural information processing systems (pp. 512-518).

[4] Bertsekas, D. (1976). On the Goldstein-Levitin-Polyak gradient projection method. IEEE Transactions on automatic control, 21(2), 174-184.

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[6] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

[7] Rasmussen, C. E. (2004). Gaussian processes in machine learning. In Advanced lectures on machine learning (pp. 63-71). Springer, Berlin, Heidelberg.

[8] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

[9] Kearns, M. J. (1990). The computational complexity of machine learning. MIT press.

[10] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160, 3-24.

[11] Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2012). Deep learning via semi-supervised embedding. In Neural Networks: Tricks of the Trade (pp. 639-655). Springer, Berlin, Heidelberg.

[12] Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. arXiv preprint arXiv:1410.5401.

[13] Zaremba, W., & Sutskever, I. (2015). Reinforcement learning neural turing machines-revised. arXiv preprint arXiv:1505.00521.