TECHNICAL SCIENCES

EFFICIENCY ANALYSIS OF FIRST-ORDER STOCHASTIC OPTIMIZATION ALGORITHMS FOR IMAGE REGISTRATION

Voronov S.,
PhD, Associate Professor, Radio Engineering department, Ulyanovsk State Technical University
Amir M.,
Senior Research Engineer, Checklens GmbH
Kozlov A.,
Senior Research Engineer, Checklens GmbH
Zinollayev A.,
Senior Research Engineer, Checklens GmbH
Aimoldin A.
Senior AI Engineer, Harrison-AI Pty Ltd

Abstract
This work presents a comparative experimental analysis of different first-order stochastic optimization algorithms for image registration in spatial domain: stochastic gradient descent, Momentum, Nesterov momentum, Adagrad, RMSprop, Adam. Correlation coefficient is considered as the objective function. Experiments are performed on synthetic data generated via wave model with different noise-to-signal ratio and real-world images.

Keywords: image registration, stochastic optimixstion, stochastic gradient descent, Adagrad, RMSprop, Adam.

1. Introduction

Image registration is referred to as a process by which the most accurate match is determined between two images, which may have been taken at the same or different times, by the same or different sensors, from the same or different viewpoints. The goal of the registration process is to determine the optimal transformation, which will align the two images. This has applications in many fields as diverse as medical image analysis, pattern matching, and computer vision for robotics, as well as remotely sensed data processing. In all of these domains, image registration can be used to find changes in images taken at different times, or for object recognition and tracking.

Spatial domain methods operate directly on pixels, and the problem of the estimation of registration parameters $\overline{\alpha}$ becomes the problem of searching for the extreme point of a multi-dimensional objective function $J(\mathbf{Z}, \overline{\alpha})$. The objective function measures the similarity between two images $\mathbf{Z}^{(1)} = \left\{z_{\bar{j}}^{(1)}\right\}$ and $\mathbf{Z}^{(2)} = \left\{z_{\bar{j}}^{(2)}\right\}$, where $\bar{j} \in \Omega$ are nodes of grid mesh Ω on which the images are defined. There is a wide variety of similarity measures that can be used as objective functions [1]. The decision of which objective function to choose is usually based on the specifics of images, deformation properties and conditions. Recently, objective functions from the theory of information are becoming more popular. Among these functions the most interesting is mutual information. It has been found to be especially robust for multimodal image registration and registration of images with great non-linear intensity distortion [2]. However, mutual information has some drawbacks. One of them is relatively high computational complexity.

The choice of optimization search technique depends on the type of problem under consideration. Traditional nonlinear programming methods, such as the constrained conjugate gradient, or the standard back propagation in neural network applications, are well suited to deterministic optimization problems with exact knowledge of the gradient of the objective function. Optimization algorithms have been developed for a stochastic setting where randomness is introduced either in the noisy measurements of the objective function and its gradient, or in the computation of the gradient approximation. Stochastic gradient ascend (descend) is one of the most powerful technique of this class [3]. It is an iterative algorithm, where registration parameters can be found as follows [4]:

$$\hat{\overline{\alpha}}_t = \hat{\overline{\alpha}}_{t-1} - \mathbf{\Lambda}_t \overline{\beta}_t\left(J(Z_t, \overline{\alpha}_{t-1})\right),$$

where $\overline{\beta}$ – gradient estimation vector of the objective function J obtained using not each pixel in the images but a sample Z_t taken randomly on each iteration, $\mathbf{\Lambda}_t$– positive-definite gain (learning rate) matrix: $\mathbf{\Lambda}_t = \|\lambda_{it}\|, \lambda_{it} > 0, i = \overline{1,m}$; m – the number of registration parameters.

The main disadvantage of this optimization algorithm is presence of a large number of local extreme points of the objective function due to the use of small samples and relatively short working range in terms of registration parameters to be estimated. To overcome these problems the number of sample elements can be increased. However, this leads to significant increase in computational efforts. Another significant problem is choosing the hyperparameter $\mathbf{\Lambda}_t$ as it largely affects not only the convergence rate but also the estimation accuracy. Thus, the problem of optimization of stochastic gradient algorithm for image registration is an important, especially for real-time processing systems. To overcome the mentioned problems some modifications of the "classical" stochastic gradient descent have been proposed. This paper is devoted to comparative experimental analysis of these modifications: Momentum, Nesterov momentum, Adagrad, RMSprop, Adam.

These algorithms are very effective, especially in training artificial neural networks [5].

2. Stochastic optimization algorithms

Let us consider the most popular modifications of stochastic gradient descent optimization algorithm which can be used for solving image registration problem.

2.1. Momentum

The idea behind Momentum optimization is quite simple [6]: if one imagine a bowling ball rolling down a gentle slope on a smooth surface: it will start out slowly, but it will quickly pick up momentum until it eventually reaches terminal velocity (if there is some friction or air resistance). In contrast, regular gradient descent will simply take small regular steps down the slope, so it will take much more time to reach the bottom. Recall that gradient descent simply updates the parameter estimates $\hat{\bar{\alpha}}$ by directly subtracting the gradient of the cost function with regards to the parameters $J(\mathbf{Z}, \bar{\alpha})$ multiplied by the learning rate $\lambda > 0$. It does not care about what the earlier gradients were. If the local gradient is tiny, it goes very slowly. Momentum optimization cares a great deal about what previous gradients were: at each iteration, it adds the local gradient to the momentum vector $\bar{m}$ multiplied by the learning rate λ, and it updates the weights by simply subtracting this momentum vector. In other words, the gradient is used as an acceleration, not as a speed. To simulate some sort of friction mechanism and prevent the momentum from growing too large, the algorithm introduces a new hyperparameter h, simply called the momentum, which must be set between 0 (high friction) and 1 (no friction). A typical momentum value is 0.9. Thus, the equation for parameter estimate updates can be written as follows:

$$\hat{\bar{\alpha}}_t = \hat{\bar{\alpha}}_{t-1} - \bar{m}_t,$$

where $\bar{m}_t = h\bar{m}_{t-1} + \mathbf{\Lambda}_t \bar{\beta}_t\big(J(Z_t, \bar{\alpha}_{t-1})\big)$.

One can easily verify that if the gradient remains constant, the terminal velocity (i.e. the maximum size of the weight updates) is equal to that gradient multiplied by the learning rate λ multiplied by $\frac{1}{1-h}$. For example, if $h = 0.9$, then the terminal velocity is equal to 10 times the gradient times the learning rate, so Momentum optimization ends up going 10 times faster than "classical" stochastic gradient descent. This allows Momentum optimization to escape from plateaus much faster.

2.2. Nesterov momentum

In [7] the author proposes one small variant to Momentum optimization which is almost always faster than vanilla Momentum optimization. The idea behind Nesterov momentum optimization consists in measuring the gradient of the cost function not at the local position but slightly ahead in the direction of the momentum. Hence, the only difference from vanilla Momentum optimization is that the gradient is measured on t-th iteration at the point $\hat{\bar{\alpha}}_{t-1} + h\bar{m}_t$ rather than at $\hat{\bar{\alpha}}_{t-1}$:

$$\hat{\bar{\alpha}}_t = \hat{\bar{\alpha}}_{t-1} - \Big(h\bar{m}_{t-1} + \mathbf{\Lambda}_t \bar{\beta}_t\big(J(Z_t, \hat{\bar{\alpha}}_{t-1} + h\bar{m}_t)\big)\Big).$$

This small tweak works because in general the momentum vector will be pointing in the right direction (i.e., toward the optimum), thus it will be slightly more accurate to use the gradient measured a bit farther in that direction rather than using the gradient at the original position. Figure 1 shows this effect. Here $\lambda\beta_1$ represents the gradient of the cost function measured at the starting point $\hat{\bar{\alpha}}_{t-1}$, and $\lambda\beta_2$ represents the gradient at the point located at $\hat{\bar{\alpha}}_{t-1} + h\bar{m}_t$. As one can see, the Nesterov update ends up faster optimizers slightly closer to the optimum. After a while, these small improvements add up and the procedure ends up being significantly faster than regular Momentum optimization. Moreover, we should note that when the momentum pushes the weights across a valley, $\lambda\beta_1$ continues to push further across the valley, while $\lambda\beta_2$ pushes back toward the bottom of the valley. This helps reduce oscillations and thus converges faster.

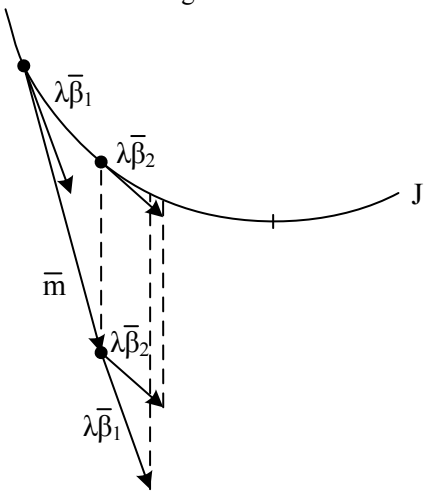


Figure 1. *Difference between Momentum and Nesterov momentum optimization.*

2.3. Adagrad

If we consider the elongated bowl problem again: gradient descent starts by quickly going down the steepest slope, then slowly goes down the bottom of the valley. However, it would be better if the algorithm could detect this early on and correct its direction to point a bit more toward the global optimum.

The Adagrad algorithm [8] achieves this by scaling down the gradient vector along the steepest dimensions:

$$\hat{\bar{\alpha}}_t = \hat{\bar{\alpha}}_{t-1} - \mathbf{\Lambda}_t \bar{\beta}_t\big(J(Z_t, \hat{\bar{\alpha}}_{t-1})\big) // (\bar{c}_t + \varepsilon)^{1/2},$$

where $\bar{c}_t = \bar{c}_{t-1} + \bar{\beta}_t\big(J(Z_t, \hat{\bar{\alpha}}_{t-1})\big)^2$.

The first step of this algorithm on each iteration is accumulating the square of the gradients into the vector $\bar{c}_t$. If the cost function is steep along the i-th dimension, then c_{it} will get larger and larger at each iteration. The second step is almost identical to "classical" stochastic gradient descent, but with one big difference: the gradient vector is scaled down by a factor of $(\bar{c}_t + \varepsilon)^{1/2}$ (the // symbol represents the element-wise division, and ε is a smoothing term to avoid division by zero, typically set to 10^{-8}). In short, this algorithm decays the learning rate, but it does so faster for steep dimensions than for dimensions with gentler slopes. This is called an adaptive learning rate. It helps point the resulting updates more directly toward the global optimum. One additional benefit is that it requires much less tuning of the learning rate hyperparameter.

Adagrad often performs well for simple quadratic problems, but unfortunately it often stops too early when training neural networks. The learning rate gets scaled down so much that the algorithm ends up stopping entirely before reaching the global optimum.

2.4. RMSprop

Although Adagrad slows down a bit too fast and ends up never converging to the global optimum, the RMSProp algorithm [9] fixes this by accumulating only the gradients from the most recent iterations (as opposed to all the gradients since the beginning of training). It does so by using exponential decay in the first step:

$$\bar{c}_t^r = d\bar{c}_{t-1}^r + (1-d)\bar{\beta}_t\left(\mathrm{J}(Z_t, \hat{\bar{\alpha}}_{t-1})\right)^2,$$

where d is the decay rate and it is typically set to 0.9.

Except on very simple problems, this optimizer almost always performs much better than Adagrad. It also generally performs better than Momentum optimization and Nesterov momentum. In fact, it was the preferred optimization algorithm of many researchers until Adam optimization came around.

2.5. Adam

Adam [10] which stands for adaptive moment estimation, combines the ideas of Momentum optimization and RMSProp: just like Momentum optimization it keeps track of an exponentially decaying average of past gradients, and just like RMSProp it keeps track of an exponentially decaying average of past squared gradients:

$$\hat{\bar{\alpha}}_t = \hat{\bar{\alpha}}_{t-1} - \boldsymbol{\Lambda}_t \bar{m}_t^A // \left(\bar{c}_t^A + \varepsilon\right)^{1/2},$$

$$\bar{m}_t^A = \frac{d_1 \bar{m}_{t-1}^A + (1-d_1)\bar{\beta}_t\left(\mathrm{J}(Z_t, \hat{\bar{\alpha}}_{t-1})\right)}{1-d_1},$$

$$\bar{c}_t^A = \frac{d_2 \bar{c}_{t-1}^A + (1-d_2)\bar{\beta}_t\left(\mathrm{J}(Z_t, \hat{\bar{\alpha}}_{t-1})\right)^2}{1-d_2}.$$

One can notice the similarity of Adam update rule to both Momentum optimization and RMSProp. The only difference is that it computes an exponentially decaying average rather than an exponentially decaying sum for $\bar{m}_t^A$ and $\bar{c}_t^A$, but these are actually equivalent except for a constant factor as the decaying average is just $(1-d_1)$ and $1-d_2$ times the decaying sum respectively. The momentum decay hyperparameter d_1 is typically initialized to 0.9, while the scaling decay hyperparameter d_2 is often initialized to 0.999. As earlier, the smoothing term ε is usually initialized to a tiny number such as 10^{-8}. In fact, since Adam is an adaptive learning rate algorithm like both Adagrad and RMSProp, it requires less tuning of the learning rate hyperparameter $\boldsymbol{\Lambda}_t$.

3. Experiments and analysis

3.1. Synthetic data

For efficiency analysis of differentit optimization algorithms it is reasonable to use simulated images whose intensity probability distribution function and correlation function can be priori defined during their synthesis. In conducted experiments simulated images based on wave model [11] with intensity probability distribution function and correlation function close to Gaussian and with different correlation radius were used. In addition, an unbiased Gaussian noise was used in simulations. Figure 2 shows an example of such synthesized image.

In order to measure the performance of the optimization algorithms we tested them on images with different noise-to-signal ratio and with different μ – the number of points in the sample using for estimation of the gradient of the chosen objective function. Correlation coefficient [1] is chosen as an objective function to be optimized. Similarity model is considered as the deformation model to be estimated. For all of the below results the deformation parameters are the following: horizontal shift – 20 pixels to the right, vertical shift – 15 pixels upwards, clockwise rotation – 17 degrees, scale factor – 0.9. In each experiment, optimal hyperparameters were chosen experimentally as different algorithms better perform with different hyperparameters and their choice is out of the scope of this article.

The number of iterations before convergence of mismatch Euclidean distance [4] E which is an integral measure of registration parameters' convergence is used as the performance criterion. Moreover, all the results are averaged by 50 realizations to make them more consistent and reproducible. In addition, result stability is analyzed using final error distributions.

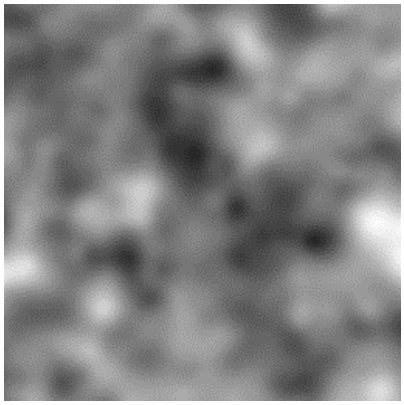
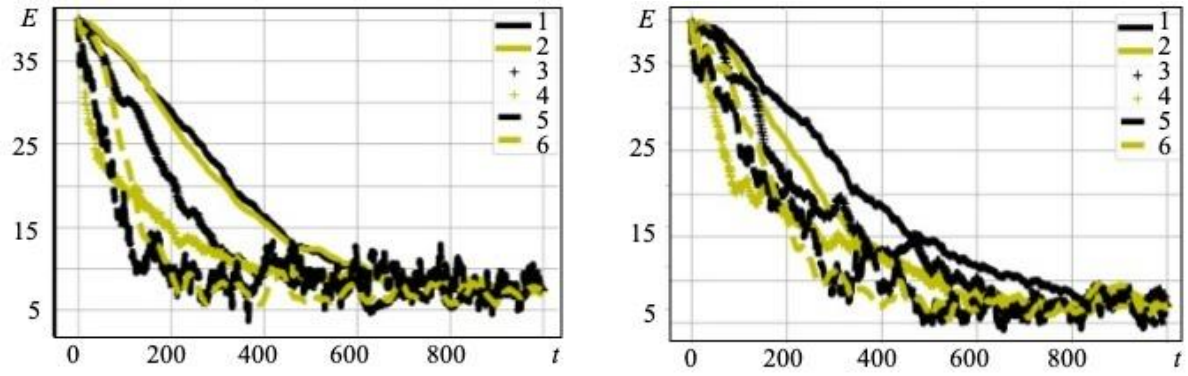


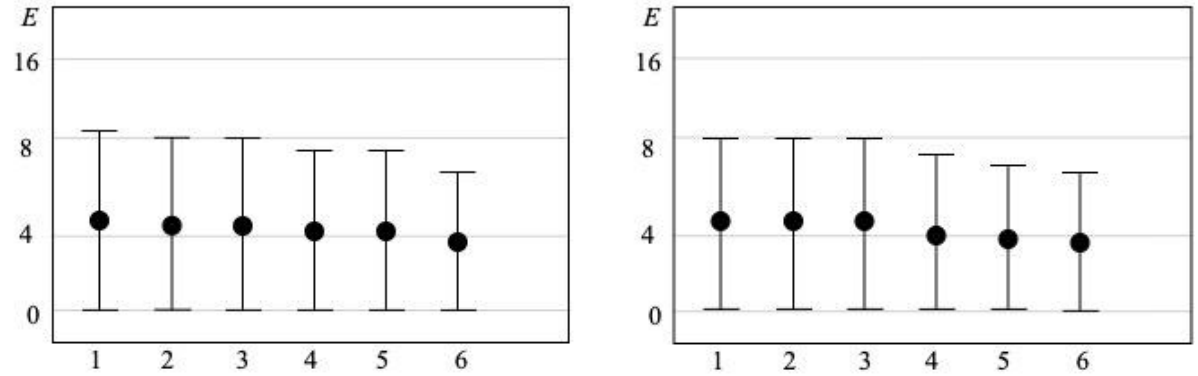
Figure 2. The image synthesized using wave model.

3.1.1. Different sample size. Figure 3 shows the convergence of mismatch Euclidean distance for the algorithms with different sample size μ. Hereafter, curve 1 corresponds to "classical" stochastic gradient descent, 2 – stochastic gradient descent with Momentum, 3 – Nesterov momentum (plus markers), 4 – Adagrad (plus markers), 5 – RMSprop (dashed line), 6 – Adam (dashed line).



a) $\mu = 5$ *b)* $\mu = 20$

Figure 3. *Mismatch Euclidean distance on the number of iterations for different sample size.*



a) $\mu = 5$ *b)* $\mu = 20$

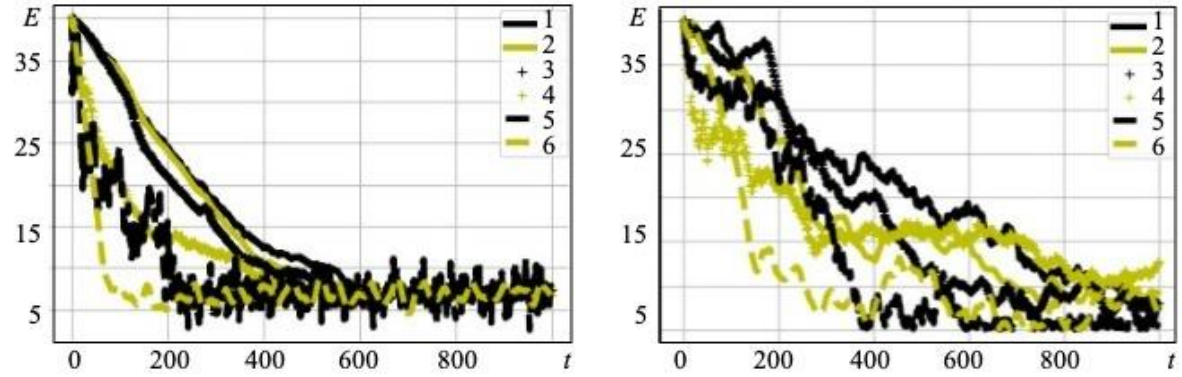
Figure 4. *Mean and standard deviation of final mismatch Euclidean distance.*

One can see that in both cases "classical" stochastic gradient descent shows the worst result as it starts to converge only after 800 iterations for $\mu = 5$ and after 700 iterations for $\mu = 20$. Momentum optimization algorithm performs almost identically but slightly better for $\mu = 20$. The best results in both cases are provided by Adam and RMSprop optimizations as they start to converge after 300 iterations for $\mu = 5$ and after 200 iterations for $\mu = 20$. Moreover, it is obvious that Adam algorithm in both situations has less variance then RMSprop, thus we can conclude that it is more stable and hence preferable. Adagrad and Nesterov momentum algorithms show close results in terms of number of iterations before convergence (500 iterations for $\mu = 5$ and after 450 iterations for $\mu = 20$), but in the beginning Adagrad has much faster convergence rate and with some optimization (e.g. increasing or dropping learning rates after a number of iterations) it possibly can outperform Nesterov momentum.

Figure 4 shows mean and standard deviation of final mismatch Euclidean distance. One can see that the behaviour of the final error for every algorithm corresponds to the behaviour of their convergence curves.

Also, we can conclude that all of the algorithms have better convergence rate with bigger sample size. It is reasonable from theoretical point of view because the objective function gradient estimates become less noisy.

3.1.2. Images with different signal-to-noise ratio. Let us test the algorithms in case of noisy images with different signal-to-noise ratio q. In this experiment we were using the sample size $\mu = 5$ for each algorithm and q. Figure 5 shows the convergence of algorithms for $q = 50$ and $q = 2$.



a) $q = 50$ *b)* $q = 2$

Figure 5. *Mismatch Euclidean distance on the number of iterations for different signal-to-noise ratio.*

As in the previous experiment, "classical" stochastic gradient descent shows the slowest convergence rate in both set-ups. Adam and RMSprop algorithms have the best result. For $q = 50$ their curves are almost identical but when the noise increases Adam has much faster convergence rate in the beginning, hence it can potentially show better result. In addition, we can notice that with very intense noise ($q = 2$) Adagrad converges with bigger error in comparison with other algorithms and it performs almost the same as "classical" stochastic gradient descent. In order to reduce the error, we can choose smaller learning rates. However, with smaller rate sometimes it was not able to converge at all.

Additionally, it is clear that noise affects not only the convergence rate but also the variance of estimates for all of the algorithms as the curves become less smooth.

3.2. Real data

Satellite images were used for the comparative analysis on real data. Figure 6 shows an example of the images taken in different weather conditions.

Figure 7 shows an example of mismatch Euclidean distance convergence of the algorithms for images shown in figure 6 and figure 8 demonstrates mean and standard deviation of final mismatch Euclidean distance. For this experiment μ was set to 25 and the results were averaged by 100 realizations as they became noisier in comparison with synthesized images, especially for the algorithms with adaptive learning rates.

One can easily notice that the results on real data are almost identical to the results on synthesized images. Again, Adam and RMSprop algorithms are the fastest in terms of convergence rate. However here we can see the algorithms with adaptive learning rates have much noisier curves that the others. It can be explained by the fact that when dealing with real images we have noisier gradient estimation, thus in these algorithms the learning rate estimation becomes less stable.

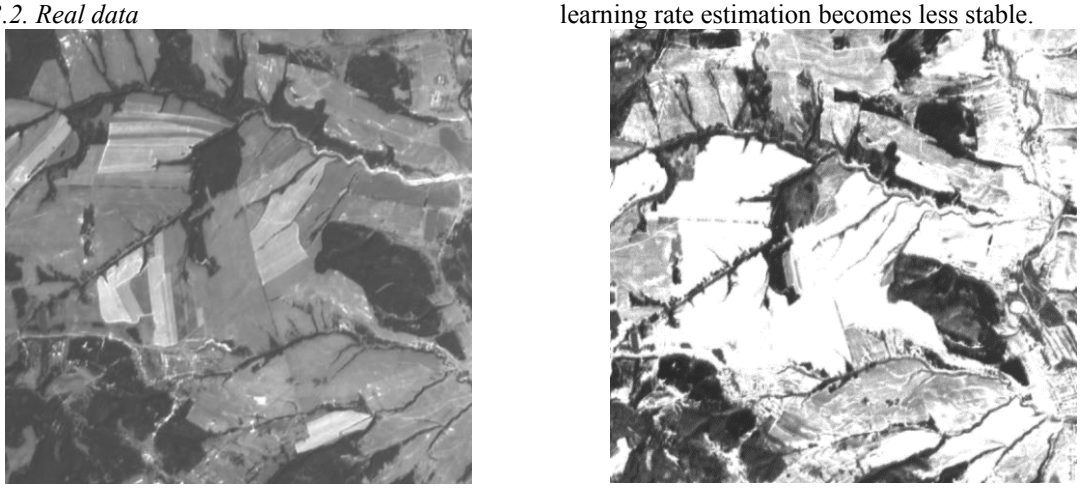


Figure 6. *An example of real satellite images which were used for the comparative analysis.*

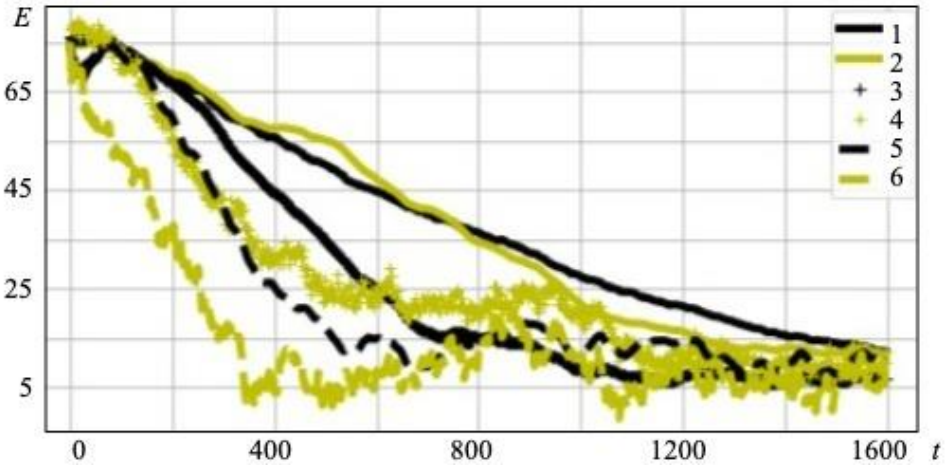


Figure 7. *Mismatch Euclidean distance on the number of iterations for real images.*

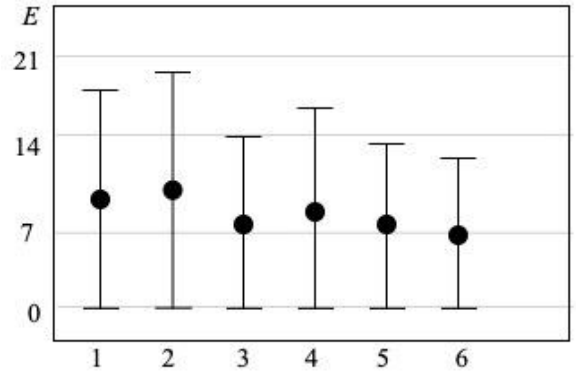


Figure 8. *Mean and standard deviation of final mismatch Euclidean distance in case of real images.*

4. Conclusion

The comparative analysis of different optimization algorithms for solving image registration problem in spatial domain shows that in each case "classical" stochastic gradient descent shows the worst result in terms of the convergence rate of registration parameters' estimates. Momentum optimization algorithm just slightly outperforms it. Adagrad and Nesterov momentum algorithms show close results in terms of number of iterations before convergence but except for the situation with intense noise Adagrad in the beginning has much faster convergence rate and with some optimization (e.g. increasing or dropping learning rates after a number of iterations) it possibly can outperform Nesterov momentum The best results are provided by Adam and RMSprop optimizations Moreover, it is obvious that Adam algorithm is almost always preferable as it has less variance then RMSprop.

Furthermore, we can conclude that all of the algorithms have better convergence rate with bigger sample size. It is reasonable from theoretical point of view because the objective function gradient estimates become less noisy. Additionally, it is clear that noise affects not only the convergence rate but also the variance of estimates for all of the algorithms as the curves become less smooth.

Experiments on real satellite images show mostly identical results.

Acknowledgments

This work was supported by the Russian Foundation for Basic Research, projects no. 18-41-730006 r_a.

REFERENCES:

1. Goshtasby, A.A. Image registration. Principles, tools and methods / A.A. Goshtasby. – Austria: Advances in Computer Vision and Pattern Recognition. – Springer, 2012. – 441 p.

2. Tashlinskii, A.G., Voronov, S.V. Specifics of objective functions for recurrent estimation of interframe geometric deformations // The 11th International conference "Pattern Recognition and Image Analysis: New Information Technologies" Conference proceedings. – 2013. – V.1. – P. 326-329.

3. Papamakarios, G. Comparison of Modern Stochastic Optimization Algorithms. Technical report / G. Papamakarios. – University of Edinburgh, 2014. – 13 p.

4. Tashlinskii, A.G., Safina, G.L., Voronov, S.V. Pseudogradient optimization of objective function in estimation of geometric interframe image deformations // Pattern recognition and image analysis. – 2012. – V. 22, №. 2. – P. 386-392.

5. Goodfellow I, Bengio Y., Courville A. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016. – 800 p.

6. Polyak, B. T. Some methods of speeding up the convergence of iteration methods // Computational Mathematics and Mathematical Physics. – 1964. – V.4, № 5. – P. 1–17.

7. Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/k2) // Soviet Mathematics Doklady. – 1983. – V. 27. – P. 372–376.

8. Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization // Journal of Machine Learning Research. – 2011. – V. 12. – P. 2121-2159.

9. University of Toronto csc321 course lecture 6 slides [Electronic resource]. — Access mode: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (18.11.2017).

10. Kingma, D., Ba, J. Adam: A method for stochastic optimization // Proceedings of the 3rd International Conference on Learning Representations (ICLR). – 2015

11. Krasheninnikov, V.R. Foundations of Image Processing Theory. –Ulyanovsk: UlSTU, 2003. – 150p.