Sapienza Università di Roma

Fundamentals of Data Science

Final Project
A faster rate in the gradient descent framework

Students

Andrei Caraman
1664744
Simone Piperno
1792917
Leonardo Skerl
1802968
Gianmarco Ursini
1635956

Professor

Prof. Fabio Galasso

Accademic Year 2021/2022

Introduction

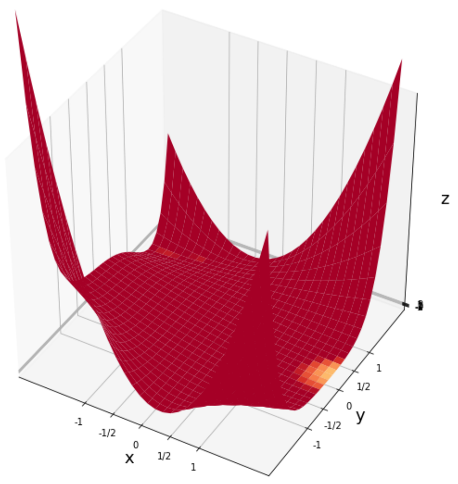
Aim of the project is to obtain (through the leverage of Runge-Kutta integrators) algorithms with a faster convergency rate in the framework of gradient descent w.r.t. the simplest gradient descent algorithm (the **Vanilla GD** with a convergency rate of the order $O(N^{-1})$ where N denotes the number of learning steps) and the **Nesterov Accelerated Gradient** (with a convergency rate of the order $O(N^{-2})$) algorithm. In the first part, the technical development of those new algorithms is done in a protected environment (i.e. descent over the Beale's function), while in the second part those algorithms are applied over a real world classification task.

1) Technical development

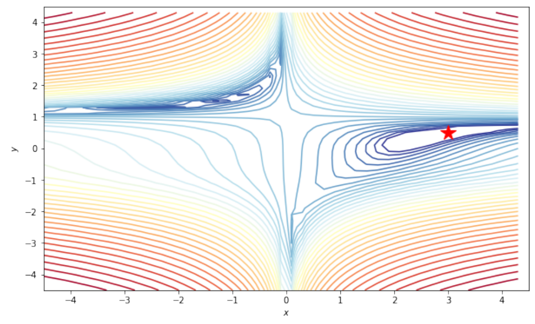
1.1) The Beale's function

The Beale's function is a typical test function for optimization task. It contains two local minima, a global minima and a saddle point, so the performances of descending algorithms can be put on the test and challenged also due to the presence of those tricky points. Beale's function is defined as follows:

$$z = f(x,y) = (1.5 - x + x\cdot y)^2 + (2.25 - x + x\cdot y^2)^2 + (2.625 - x + x\cdot y^3)^2 \qquad \underset{(x,y)}{\operatorname{argmax}}\, z = (\hat{x},\hat{y}) = (3, 0.5)$$



(a) 3D view of Beale's function.



(b) Contour plot of Beale's function.

1.2) The Runge-Kutta algorithms

The Runge-Kutta methods are a family of iterative methods used to find approximate solutions to ODEs using temporal discretization. In our project, two different RK descent algorithms will be implemented:

1. In the first approach, we will not leverage the order of the RK integrator to get a faster rate but the power of using an **adaptive learning rate**: indeed, due to the fact that the optimal learning rate computation will require information about the 2^{nd} derivative of the loss, a RK integrator of order 2 will be sufficient (at the same learning step it allows to compute differences between loss's gradients, i.e. to get informations over the Hessian structure of the loss);

2. In the second approach, a faster rate of convergence will be obtained through the leverage of the order of RK integrator (i.e. the rate will be a function of the RK integrator's order);

1.3) RK-2 Adaptive learning rate

It can be theoretically proven that if the loss function has bounded 1^{st} and 2^{nd} derivatives, the greatest learning rate that preserve convergency can be picked [AT21]. Denoting with $\theta^{\text{old/new}}$ the learning parameters that respectively are inputted/outputted in a learning step, with $f(\cdot)$ the loss function, with lr the initial learning rate and with β the hyperparameter that balance the usare of the optimal learning rate, the updating policy proceeds as follows:

$$\begin{cases} g = \nabla f(\theta^{\text{old}}) \\ \hat{\theta} = \theta^{\text{old}} - lr \cdot g \\ \tilde{g} = \nabla f(\hat{\theta}) \\ lr^{\text{opt}} = \begin{cases} \frac{\langle g-\tilde{g}, g\rangle}{||g-\tilde{g}||^2}, & \text{if } \langle g-\tilde{g}, g\rangle > 0 \Rightarrow lr^* = \beta \cdot lr + (1-\beta) \cdot lr^{\text{opt}} \\ lr, & \text{if } \langle g-\tilde{g}, g\rangle \leq 0 \Rightarrow lr^* = (1-\beta) \cdot lr^{\text{opt}} \end{cases} \\ \theta^{\text{new}} = \theta^{\text{old}} - lr^* \cdot g \end{cases}$$

Since the rate of convergence was not provided in [AT21], a numerical analysis on top of the during-learning loss values will be performed to estimate the **RK-2 A.L.R.** convergency rate.

1.4) Leveraging the order of the integrator with RK-4

It can be theoretically proven [Zha+18] that in the limit $lr \to 0$, the N.A.G. updating policy at a particular learning step t is equivalent to the following ODE:

$$\ddot{\theta}(t) + \frac{3}{t}\dot{\theta}(t) + \nabla f(\theta(t)) = 0$$

The loss value $f(\theta(t))$ decreases at a rate of the order $\mathcal{O}(t^{-2})$ along the temporal trajectory of the ODE. This convergence rate can be accelerated to an arbitrary rate in continuous time via **time dilation** as in [WWJ16]. In particular, one can show [Zha+18] that by manipulating the previous equation it is possible to achieve a convergence rate of $O(N^{-p})$, with p being an arbitrary value[1].

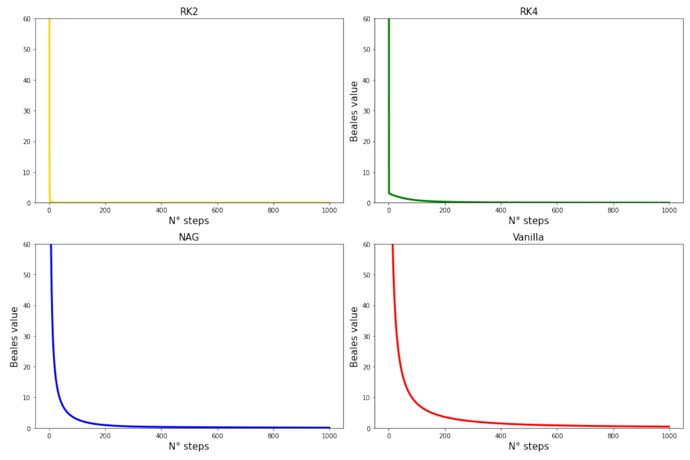
By denoting with $\theta^{old/new}$ the learning parameters that respectively are inputted/outputted in a learning step and by $f(\cdot)$ the loss function the problem is solved in the previously mentioned time by the following algorithm:

$$\begin{cases} k1 = lr\nabla f(\theta^{old}) \quad lr = CN^{\frac{1}{5}} \\ k2 = lr\nabla f(\theta_1) \quad \theta_1 = \theta^{old} + \frac{lr}{2}k1 \\ k3 = lr\nabla f(\theta_2) \quad \theta_2 = \theta_1 + \frac{lr}{2}k2 \\ k4 = lr\nabla f(\theta_3) \quad \theta_3 = \theta_2 + \frac{lr}{2}k3 \\ \theta^{new} = \theta^{old} - (\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}) \end{cases}$$

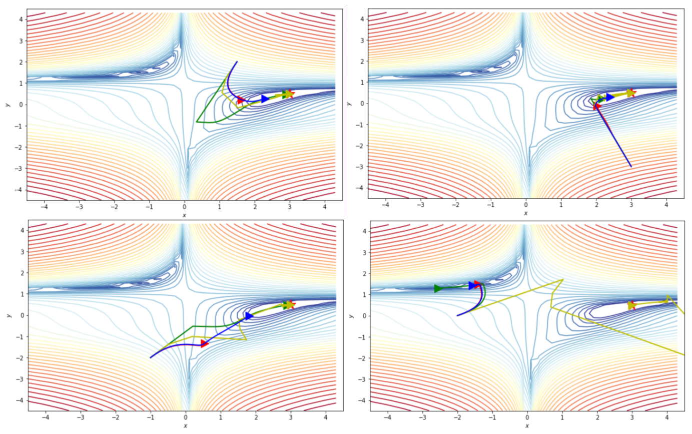
where C is a hyperparameter that depends on p, θ.[2]

[1] it has obviously some limits and cannot be set as high as you want: you may incur in stability problems.
[2] for more details check [Zha+18]

1.5) Results



(a) Loss trends during learning

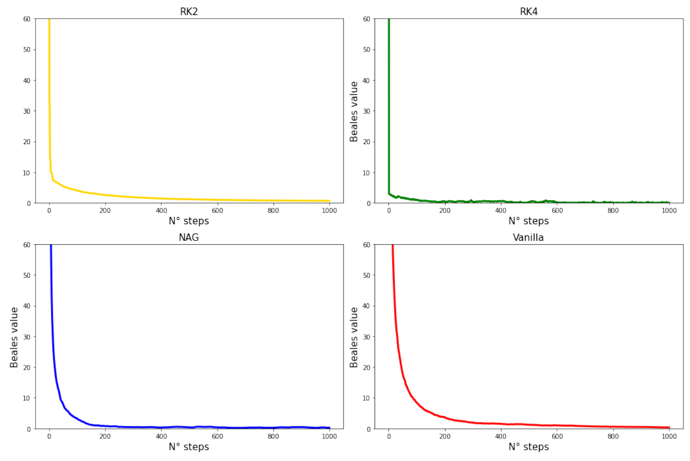


(b) Learning trajectory on Beale's function.

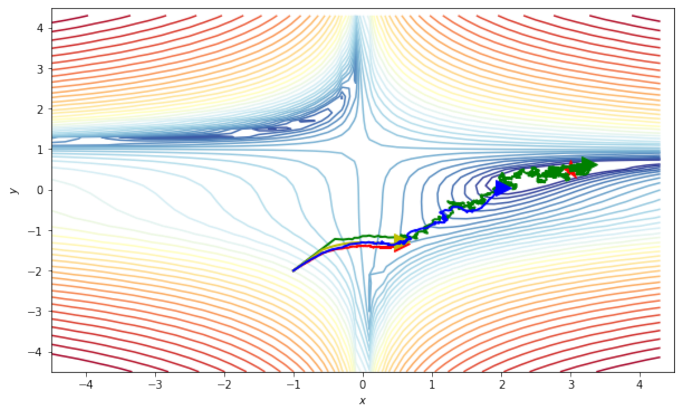
As clear from those picture, **RK-2 A.L.R.** is outperforming all the other descent methods in most of the scenarios, even when tricky initializations are used. To be highlighted is the cristal clear boost obtained in the convergency rate through the adoption of the adaptive learning rate.

1.6) Noise injection

Performances of the different gradient descent methods can be studied even under a gaussian noise injection: Indeed, all the times a gradient $\nabla f(\theta) \in \mathbb{R}^p$ is computed over the loss, a vector of gaussian noise $N =$`noise-strength`$\cdot[\mathcal{N}(0,1), ..., \mathcal{N}(0,1)]^T \in \mathbb{R}^p$ is added to the true gradient (where `noise-strength` is a scalar variable that allows to modulate the amount of noise injected). The effects of noise in the training phase are shown below:



(a) Noisy loss trends during learning



(b) Noisy learning trajectory on Beale's function.

As we can see, the performance of **RK-2 A.L.R.** abruptly degrades under the injection of noise, while **RK-4** is the only able to keep its "noisy" walk up to the global minima.

1.6) Numerical estimates for rates of convergence

Since both reference papers [AT21] and [Zha+18] didn't mention exact convergency rates, a numerical analysis has been performed to extract those values. We claim that (given a loss function $f(\cdot)$ and denoting with $\theta(N)$ the learning parameters computed at the learning step N) a gradient descent/ascent algorithm converge at a rate d to the optimal solution θ^* if up to constant value C:

$$|f(\theta(N)) - f(\theta^*)| \propto \frac{C}{N^d}$$

Since if the Beale's function is used as a loss $f(\theta^*) = 0$, we can simply use the loss values computed during learning to extract the rate d. Indeed, a log-rescaling of the y-axes can be performed and the rates d can be numerically estimated by the data through the usage of a linear model:

$$\log(|f(\theta(N))|) = \log(C) - d \cdot \log(N)$$

Several simulations with random parameter inizializations are performed. During each simulation, 4 different linear models are fitted to extract d from the 4 different algorithms. Being $d_{\text{vanilla}} = 1$, $d_{\text{RK-2}}$, $d_{\text{RK-4}}$ and d_{NAG} values will be normalized over the known d_{vanilla}. The final values of the rates are then estimated as the average of the rate values for each algorithm across all the simulation performed. Numerical analysis results are reported in the following table: Even if it's clear that a numerical analysis can't capture the exact and

	RK-2 A.L.R.	**RK-4**	**NAG**	**Vanilla**
d	$20.3 \simeq 20$	$2.7 \simeq 3$	$1.7 \simeq 2$	1

Table 1: Approximated convergency rates for the implemented algorithms.

formal rate of convergency of the given algorithms, we appreciate the fact that the estimated d_{NAG} is under approximation equal to the theoretical rate. It is clear from this analysis that (on average) **RK-4** algorithm is converging at a faster rate w.r.t. the **NAG** one, while the usage of **RK-2 A.L.R.** is resulting into an impressive convergency rate boost.

2) The Titanic dataset

2.1) The dataset

We want now to apply our methods over a real dataset. For this purpose we selected the "Titanic" dataset from Titanic Dataset, a list of all passenger in the famous Titanic. The dataset is splitted in two files:

- train.csv, a dataset with 891 samples with 12 columns
- test.csv, a dataset with 418 samples with 11 columns

Both of dataset have the same features except for "Survived" feature, a binary feature that assume "1" if the passenger survived and "0" otherwise. Our purpose is apply the methods previously described and try to predict, training our models over the "train.csv" dataset, which of the passenger in "test.csv" dataset is survived or not. We will use different methods to find which one is the best and faster method to reach the best predictor.

2.2) The preprocessing

For first we have preprocessed the original dataset in the following ways:

- "Sex" feature, a qualitative binary variable, is converted in a discrete quantitative feature with two levels: "Male" = 0 and "Female" = 1
- "Embarked" feature, a qualitative variable with three levels, that states from which gate a passenger was embarked is converted in a discrete quantitative variable with three levels: "C" = -1, "Q" = 0, "S" = 1
- We have dropped some features that we thought were irrelevant or with too many different qualitative levels or with too many missing values:
  - "Ticket" feature, that states the number of ticket purchased by the passenger
  - "Cabin" feature, that states the number of the cabin where the passenger has stayed during the trip
  - "Name" feature, that states the name of the passenger
- We have normalized some features in order to avoid different scales or variability problem like: "Age" and "Fare" both quantitative features that state respectively the age of each passenger and the cost of the ticket
- We have cleaned the dataset from the missing values that we have found in "Age" feature and "Embarked" feature
- In the end we have added a column of ones in the corresponding matrix we have created in order to create the typical column for x_0 feature

2.3) Our classifier

Our model is made up of a logistic regressor for binary classification. The gradient ascent is executed through the usage of the 4 above implemented gradient descent methods:

- Vanilla
- RK-2 Adaptive Learning Rate
- RK-4
- NAG

2.4) The training

All the 4 different gradient ascent algorithms are then used during the training phase. Reported below the log-likelihood values.

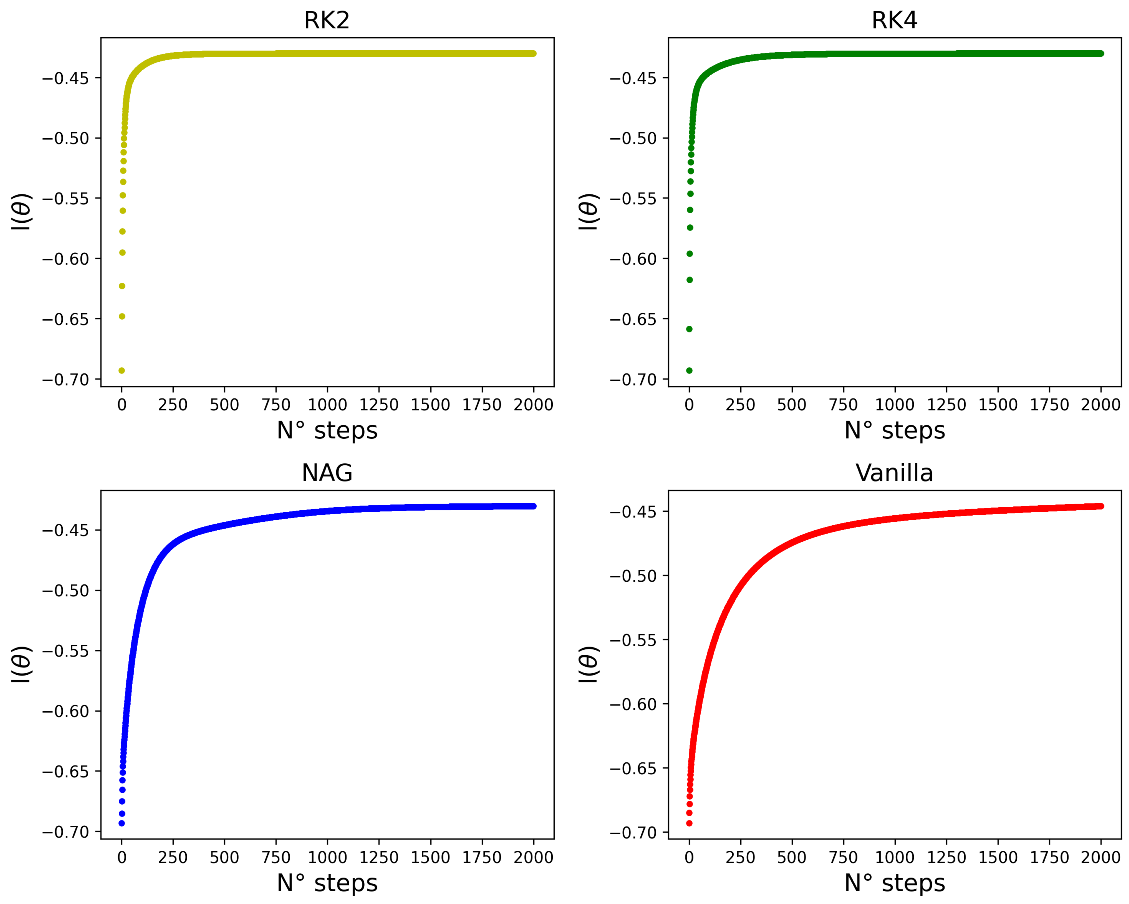


Figure 4: Log-likelihoods values during training.

2.5) The results

Due to the fact that the Kaggle competition didn't provide an explicit list of test labels to be used during the test accuracy computation, it is mandatory to submit a prediction **.csv** on the competition page to assess the test accuracy. Attached below results of the retrieved test accuracies:

	RK-2 A.L.R.	**RK-4**	**NAG**	**Vanilla**
Accuracy	0.76	0.763	0.761	0.77

Table 2: Accuracies for the 4 different ascent algorithms.

References

[WWJ16] Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. *A Variational Perspective on Accelerated Methods in Optimization*. 2016. arXiv: 1603.04245 [math.OC].

[Zha+18] Jingzhao Zhang et al. *Direct Runge-Kutta Discretization Achieves Acceleration*. 2018. arXiv: 1805.00521 [math.OC].

[AT21] Imen Ayadi and Gabriel Turinici. "Stochastic Runge-Kutta methods and adaptive SGD-G2 stochastic gradient descent". In: *25th International Conference on Pattern Recognition (ICPR 2020)*. Milano, Italy, Jan. 2021. URL: https://hal.archives-ouvertes.fr/hal-02483988.