Optimization for Total-Variation Image Denoising

Bingcheng HU
Department of Electric and Computer Engneering
Shanghai Jiao Tong University
Student ID: 516021910219
`bingcheng@sjtu.edu.cn`

Abstract

Nowadays, smartphones are generally equipped with high-definition cameras, and the effect of taking pictures at night is always unsatisfactory, because when the light is weak, the internal and external noises become relatively large. TV regularization is an algorithm that reduces data noise by processing data in different dimensions. In this paper, the optimization of the TV denoising algorithm for monochrome pictures is discussed.

Keywords: total variation, regularized learning, sparsity, convex optimization

1 Introduction

Total variation has a long history in the field of image processing, and it was originally a good solution for signal denoising problems [Condat, 2013, Durand and Froment, 2003, Easley et al., 2008, Selesnick et al., 2014]. Condat [2013] propose a fast denoising algorithm for filtering discrete signals using total variation regularization [Condat, 2013]. The optimization problem of the one-dimensional (1D) discrete signals denoising is shown as below.

$$\min_{x\in\mathbb{R}^N} \frac{1}{2}\sum_{k=1}^{N} |y[k]-x[k]|^2 + \lambda \sum_{k=1}^{N-1} |x[k+1]-x[k]|$$

where y is the noised signal and by solving this optimization problem, we can get the denoised signal x. Condat [2013] presents an efficient way to implement this convex optimization problem, and one example of the usage of the algorithm in signal processing is shown in figure 1 [Condat, 2013].

Figure 1: One example of the noised signal (in red) , the unknown ground truth (in green), and the TV-denoised signal x (in blue) [Condat, 2013].

Submitted as 34th Optimization in Machine Learning Course Assignment (VE485 2020). Do not distribute.

Apart from the signal denoising, the total variance method was considered a well-established way to solve image processing problems [Chen et al., 2010, Blomgren et al., 1997, Chen et al., 2015, Thanh and Dvoenko, 2015]. I real life, people need to take photos at night and their phones always catch a noised picture because of the lack of light. Nowadays, many smartphone companies introduced lots of algorithms to improve the quality of the photo (Chen et al., 2010, Blomgren et al., 1997, Chen et al., 2015). In those algorithms, the total variant is still a very important part of denoising.

In the photograph field, Total variant denoising can help the photographer get a better picture. But for non-professionals, it is difficult to adjust the regularization parameter. Chen et al. [2010] proposed a majorization-minimization approach for adaptative total variation image denoising. Using their methods, regularization parameter doesn't need to be specified by authors. This layman-friendly method can easily make total variant image denoising expand into daily life. Figure 2 shows the practical application of this algorithm. The picture shows a picture of a zebra. The picture on the left is the original picture, and the picture with noize added in the middle. On the right is the image after noise reduction using the total variant denoising. By carefully comparing Figure 1.c and Figure 1.a, you can find that the denoised image has an excellent performance in detail retention.

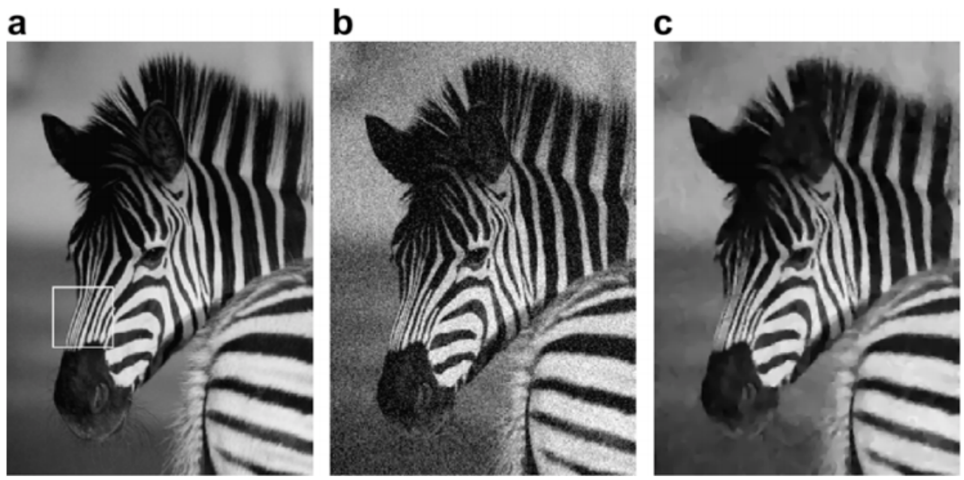

Figure 2: One example of the original image (a) , the noised image (b), and the TV-denoised image (c) [Chen et al., 2010].

Total variant image denoising is also a significant part of medical imaging. Thanh and Dvoenko [2015] proposes a TV denoising algorithm for biomedical images. This algorithm achieves noise reduction in line with medical standards by adjusting parameters. This noise reduction method will retain important details such as blood vessels, so that the details will not be erased by the noise reduction algorithm. However, this algorithm will cause color distortion, so it can not be used to denoise everyday photos. Figure 3 shows the MRI pictures of the human brain before and after noise reduction. Observing the pictures shows that the details of the images are well preserved.

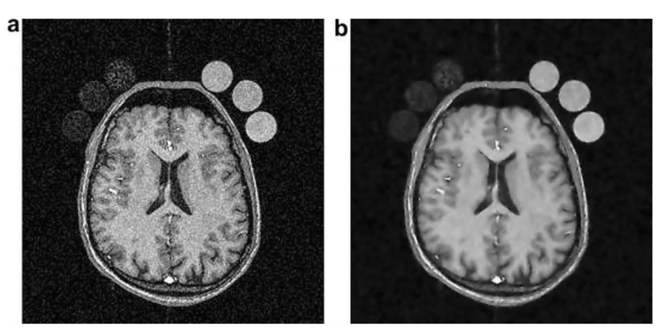

Figure 3: One example of the original noised image of the human brain scanned image (a) , and the TV-denoised image (b) [Thanh and Dvoenko, 2015].

In the field of total variant denoising, lots of new technologies were introduced these years. Many of them can achieve high quality, while still some algorithms were too slow for use in daily life []. Blomgren et al. [1997] studies one of the most useful images restoring algorithm and the optimization illustration of its problem is shown as below [Blomgren et al., 1997].

$$\min_u \alpha TV(u) + \frac{1}{2}\|\mathbb{K}u - z\|_{\mathcal{L}^2}^2$$

The Total Variation norm chosen in this equation is $TV(u) = \int_\Omega |\nabla u| dxdy$. Because the Total Variation norm does not penalize disruption in u , it ensures a better restoration for edges[Blomgren et al., 1997].

Beck and Teboulle [2009] introduced gradient-based strategies for image denoising and deblurring problems, and the model of the algorithm is based on TV minimization model with constraints [Beck and Teboulle, 2009]. The innovative invention of this paper is the usage of a fast iterative shrinkage/thresholding algorithm (FISTA) with its "novel monotone version" [Beck and Teboulle, 2009]. The convex non-smooth minimization problem is represented as below.

$$\min_{\mathbf{x}} |\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2 + 2\lambda \mathrm{TV}(\mathbf{x}), \quad (\lambda > 0)$$

Where $TV(\Delta)$ stands for the l1-based, anisotropic TV, which is defined by

$$\begin{aligned} \mathbf{x} \in \mathbb{R}^{m \times n}, \quad \mathrm{TV}_{l_1}(\mathbf{x}) = & \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \{|x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|\} \\ & + \sum_{i=1}^{m-1} |x_{i,n} - x_{i+1,n}| + \sum_{j=1}^{n-1} |x_{m,j} - x_{m,j+1}| \end{aligned}$$

which is more complex than we have learned in class.

Chen et al. [2015] invented the fractional-order TV denoising model. The given image is denoted as the function $f : \Omega \to \mathbb{R}$, where Ω is the subset of R^2 in the image domain and the fractional-order TV denoising model is described by

$$\min_u \int_\Omega \frac{1}{2}(u - f)^2 + \mu |\nabla u| \mathrm{d}\Omega,$$

and after minimize this function, u will be the clean image which is wanted.

But because of the non-differentiability of the fractional-order TV regularization term, this paper used a proximity algorithm to solve it, which is a discrete model formulation

$$\min_p \frac{1}{2}\|p - g\|_2^2 + \mu \left\|A^\alpha p\right\|_1$$

where matrix $A^\alpha = [A_1^\alpha, A_2^\alpha, \ldots, A_N^\alpha]^T \in \mathbb{R}^{2N \times N}$.

Whatever the methods used in those papers, the regularization term is always the key point in controlling the model complexity. In this case, the least absolute shrinkage and selection operator(LASSO) needs to be introduced as figure 4 shows.

$$\min_x \gamma \sum_{j=1}^n |x_j| + \frac{1}{2} \sum_{i=1}^m \left(x^T a_i - b_i\right)^2$$

In this formula, both the l1 norm and the l2 norm can be used. But using the l1 norm can keep more sparsity. First, let's check the l2 norm

$$\min_x \gamma |x| + 1/2(x-1)^2$$

where we can get $f'(x) = \gamma\, \mathrm{sgn}(x) + (x - 1)$ such that $x^* = 0$ if $\gamma > 1$.

And for l1 norm

$$\min_x 1/2\gamma x^2 + 1/2(x-1)^2$$

where we can get $f'(x) = \gamma x + (x - 1)$ such that $x^* \neq 0$.

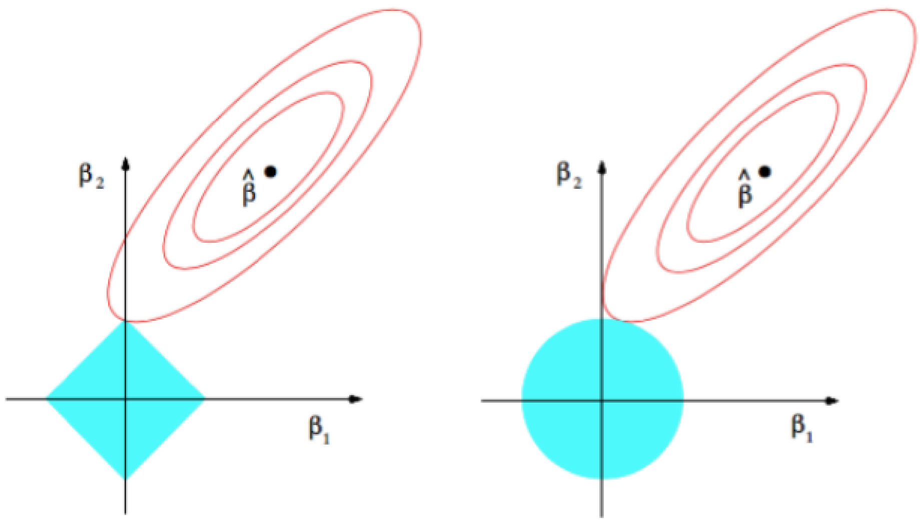

Figure 4: Estimation picture of the lasso (left) and ridge regression (right) [Friedman et al., 2001].

Both L1 norm and L2 norm regularization can help reduce the risk of overfitting, but the former also brings an additional benefit: it is easier to obtain "sparse" (sparse) solutions than the latter, that is, what it finds ω will have fewer non-zero components. This characteristic is marked as the following picture:

The previous section introduced some literature that used different Total Variance, and also introduced some basic knowledge about convex optimization. The following section will introduce the problems we need to solve.

2 Problem Statement

The goal of this article is to implement a real TV denoising algorithm from the python code step by step. For any picture, it can be shown as $X \in R_{n\times n}$. However, sparsity is not an inherent property of pictures, so we need a way to express the sparsity of pictures. Obviously, the difference between a pixel in the picture and the surrounding pixels can indicate whether this pixel is conspicuous in the picture. However, the noise of the picture will often produce many conspicuous points, which will affect the clarity of the picture.

Therefore, in order to eliminate noise, we need to propose a method to represent the number and intensity of the noise point in the image. This method is using the total variation (TV). For picture X, its TV is written as below.

$$\|X\|_{TV} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\sqrt{(X_{i,j}-X_{i+1,j})^2+(X_{i,j}-X_{i,j+1})^2} or \; \|X\|_{TV} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}|x_{l,j}-X_{i+1,j}|+|X_{i,j}-X_{i,j+1}|$$

Which of these two formulas is better? The two adjacent stores in the first formula are coupled together, which means that these two items cannot be handled separately. This leads to optimization difficulties. However, the two terms in the second formula are separate, which means that the two terms can be handled separately. From this perspective, the second method is more convenient. However, in fact, the adjacent points in the picture are coupled together, so using the first method will get better results. So we decided to use the left formula as part of our TV denoising algorithm.

Could it be possible that reducing the total variation help reduce noise? Through the above analysis, we know that the answer is yes. But if you do not set a limit, and directly reduce the total variation, it will cause other hard problems, such as the required details are eliminated and the sharp edges of the original become blurred. So in our optimization process, we will need an item so that the difference between denoised image and Noisy image is not too large. So a new variable is introduced to this problem.

$$dissimilarity = \|F - X\|_2^2$$

By minimize this term, we can make the denoised image and Noisy image similar to each other.

Finally, our optimization problem becomes the following form:

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2$$

In the following part of this article, the main task is to achieve an optimized solution to this problem through Python programming.

3 Total Variance in Image denoising

In this section, the complete derivation process of the image denoising algorithm will be introduced. To make the start simpler to understand, we choose the general gradient descent (GD) algorithm as the first step of denoising.

As stated in the previous section, our aim is to

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2.$$

According to the gradient descent algorithm, the problem can be solved with the following procedures:

Algorithm 1 Gradient Descent Algorithm

---

procedure GRADIENT DESCENT ALGORITHM WITH EXACT SEARCH
given a starting point $x \in \operatorname{dom} f$
repeat:
   1. $\Delta x := -\nabla f(x)$
   2. Line search. Choose step size t via exact or backtracking line search.
   3. Update. $x := x + t\Delta x$
until stopping criterion is satisfied.

So it's needed to compute $\Delta x := -\nabla f(x)$ every step, then we need to do exact line search in the direction of the negative side of the gradient.

3.1 Calculate the gradient of the aim function

In this section, the procedure of calculating the partial differentiation. Both of the following two items will be partially differentiated separately.

$$\|X\|_{TV} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2}$$

$$dissimilarity = \|F - X\|_2^2$$

In the following subsections, the full procedures will be implemented.

3.1.1 Calculate the partial differentiation of $\|X\|_{TV}$

The condition of the related blocks is shown in figure 5. According to the differential equation of the equation of TV, for every $x_{i,j}$, there will be three part of equations connected with it, and they are

$$\sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2},$$
$$\sqrt{(X_{i-1,j} - X_{i,j})^2 + (X_{i-1,j} - X_{i-1,j+1})^2},$$
$$\sqrt{(X_{i,j-1} - X_{i+1,j-1})^2 + (X_{i,j-1} - X_{i,j})^2},$$

and as you can see, there will be three inverse “L” shape block hit the center block for every $x_{i,j}$, and each of them is colored with blue, green, and red.

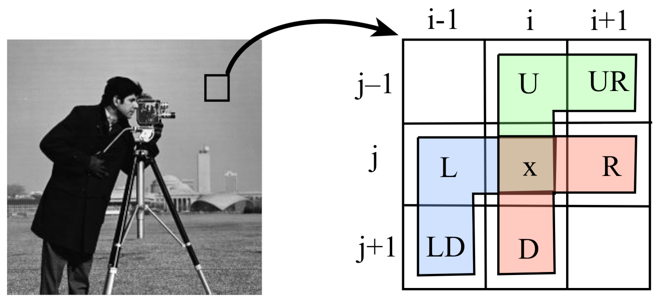

Figure 5: the representation of one center pixel surrounded by eight pixels.

By looking at the three equations connedted with $x_{i,j}$, it can be found that they can be seperated to to part, $\sqrt{(x-a)^2+(x-b)^2}$ and $\sqrt{(a-x)^2+(a-b)^2}$. To Calculate $\frac{\partial}{\partial x}\|X\|_{TV}$, the partial diffrenciation of these two equation should be calculated first.

$$\frac{\partial}{\partial x}(\sqrt{(x-a)^2+(x-b)^2}) = \frac{-a-b+2x}{\sqrt{a^2-2ax+b^2-2bx+2x^2}}$$

$$\frac{\partial}{\partial x}(\sqrt{(a-x)^2+(a-b)^2}) = -\frac{a-x}{\sqrt{(a-b)^2+(a-x)^2}}$$

The Python implecation of these equations are shown below:

```
def delTV(L, R, U, D, LD, UR, x):
    def delTV1(a,b,x):
        divider = a*a - 2*a*x + b*b - 2*b*x + 2*x*x
        return (-a-b+2*x)/math.sqrt(divider) if divider>0 else 0
    def delTV2(a,b,x):
        divider = math.sqrt((a-b)*(a-b)+(a-x)*(a-x))
        return (-a+x)/divider if divider>0 else 0
    return delTV1(D, R, x) + delTV2(L,LD,x) + delTV2(U,UR,x)
```

3.1.2 Calculate the partial differentiation of $\|F-X\|_2^2$

For this part, it can be solved easily.

$$\frac{\partial}{\partial x}\left((a-x)^2\right) = 2x-2a$$

3.2 Deal with the boundary condition

After implementing all the cells inside the image, there comes a problem that when meeting with the boundary, all these equations will lose some information as shown in figure 6. For the shadow part, they are the missing message.

It’s not easy to deal with this problem, and for the current state, ignoring the boundary pixels is one choice. This problem will be solved in future updates.

3.3 Implement the Gradient Descent for Total Variation Denoising

In the folder of code, *GD-general.ipynb* and *GD-general.py* can be found (the content of them is the same). In the inner folder img, some original images can be seen, and they are downloaded from *https://www.math.ust.hk/ masyleung/Teaching/CAS/MATLAB/image/target2.html*. After running *GD-general.ipynb*, the output files are saved in the inner folder *gen-img*, and the example of the

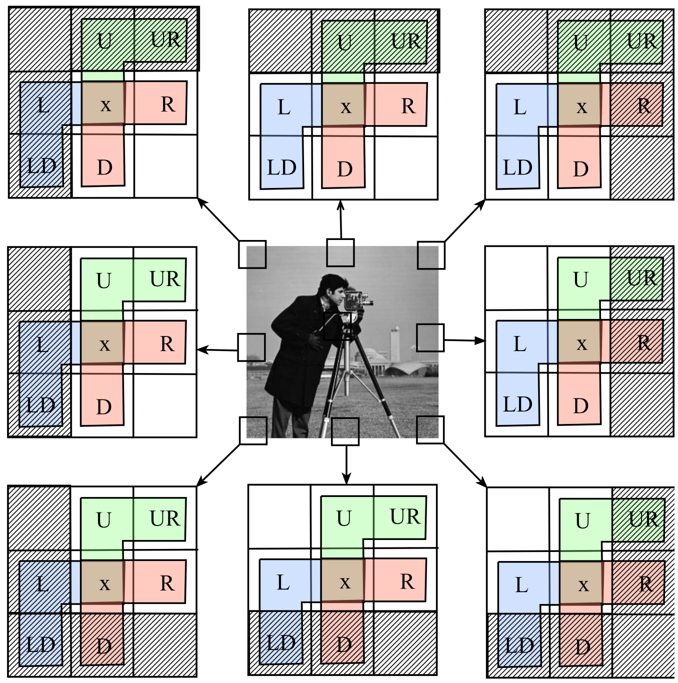

Figure 6: The boundary condition for the sample image which is hard to solve.

denoising processing are shown in figure 7. The Noised images are generated with Gauss Noise (GN). The algorithm of Gauss Noise is in img.py.

```
    def gaussNoise(im_array, sigma):
        im_array_flat = im_array.flatten()
        for i in range(im_array.shape[0]*im_array.shape[1]):
            pointInFlat = int(im_array_flat[i])+ random.gauss(0,sigma)
            if pointInFlat < 0:
                pointInFlat = 0
            if pointInFlat > 255:
                pointInFlat = 255
            im_array_flat[i] = pointInFlat
        im_array = im_array_flat.reshape([im_array.shape[0],im_array.
                                          shape[1]])
        return im_array
```

As shown in figure 7, the leftmost one is the original image directly downloaded from the website, and the next one is the noised image. Then the following three images are the denoised image with iteration 1, 10, and 100. It can be found that after 10 iterations, the noised image is already much better than the first iteration.

Original image Noised image Iteration 1 Iteration 10 Iteration 100

Figure 7: The original image, noised image, and three images are the denoised image with iteration 1, 10, and 100 with gradient descent.

To show the speed of convergence, $f(x^k)$ vs. time and $f(x^k)$ vs. iteration k are shown in figure 8 and figure 9. It can be found that at first iterations, the speed of convergence is fast. However, after some iterations, the speed is much slower.

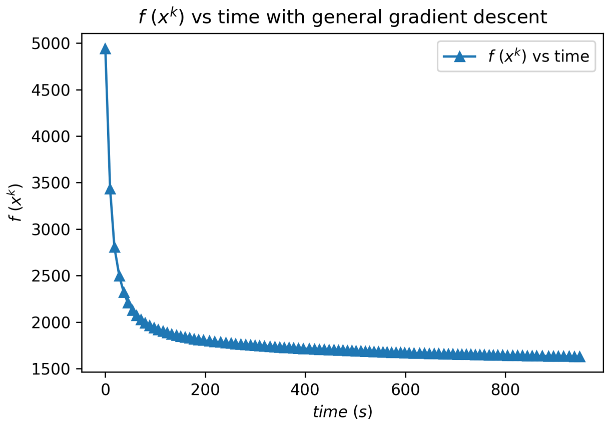

Figure 8: $f(x^k)$ vs. time for general gradient discent.

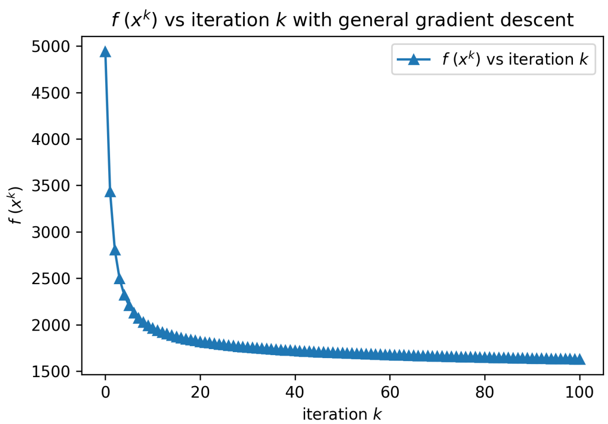

Figure 9: $f(x^k)$ vs. iterration k for general gradient discent.

3.4 Implement the Nesterov Acceleration to Accelerate Gradient Descent

To accelerate the speed of convergence, Nesterov acceleration is chosen as the speed-up algorithm. And the thinking of methods is shown below.

$$\begin{aligned} z_{k+1} &= x_k - t_k \nabla f(x_k) \\ x_{k+1} &= z_{k+1} + \delta_k (z_{k+1} - z_k) \quad \delta_k \in [0, 1) \end{aligned}$$

And the algorithm is shown below [Ioannis, 2018].

Ioannis [2018] also Compared Polyak's method with Nesterov's algorithm. Ioannis says that Polyak's method evaluates the gradient before adding momentum, while Nesterov's algorithm evaluates the gradient after applying momentum as shown in figure 10 [Ioannis, 2018].

Just applying this method, the output is very similar to the pure exact gradient search (figure 11). But the difference is not easy to be seen by human eyes. So the $f(x^k)$ vs. time and $f(x^k)$ vs. iteration k are presented to show the speed of convergence, in figure 12 and figure 13. It can be found that at first iterations, the speed of convergence is fast. However, after some iterations, the speed is much slower.

Algorithm 2 Gradient Descent Algorithm

procedure GRADIENT DESCENT ALGORITHM WITH NESTEROV ACCELERATION
given a starting point $x \in \operatorname{dom} f$, learning rate δ_k
repeat:
1. $z_{k+1} = x_k - t_k \nabla f\left(x_k\right)$
2. $x_{k+1} = z_{k+1} + \delta_k \left(z_{k+1} - z_k\right) \quad \delta_k \in [0, 1)$
3. Line search. Choose step size t via exact or backtracking line search.
4. Update. $x := x_{k+1}(t_k)$

until stopping criterion is satisfied.

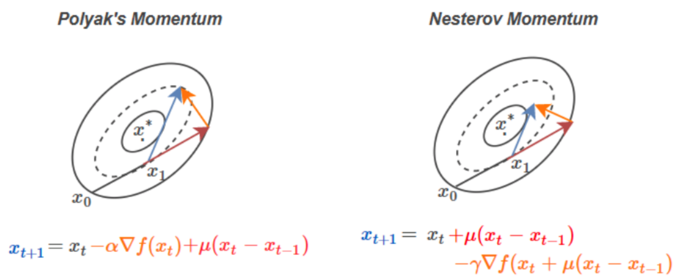

Figure 10: Comparison of Polyak's method and Nesterov's algorithm.

3.5 Comparison of gradient descent and gradient descent accelerated by Nesterov method

Still, comparing those graphs in one figure is a better way to show the difference. Figure 14 shows the comparison of $f(x^k)$ vs. time for gradient descent and gradient descent accelerated by Nesterov method.

When accelerated by the Nesterov method, it only cost 82 seconds to reach the value that general gradient descent needs to cost 918 seconds to reach, which is 10 times faster!

Comparing $f(x^k)$ vs. iteration for gradient descent and gradient descent accelerated by Nesterov method as figure 15 present the same output: Nesterov method only cost 9 iterations to reach the value that general gradient descent need to cost 100 iterations to reach, which is also about 10 times faster!

4 Optimization of the TV algorithm

5 Conclusion

References

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434, 2009.

Peter Blomgren, Tony F Chan, Pep Mulet, and Chak-Kuen Wong. Total variation image restoration: numerical methods and extensions. In *Proceedings of International Conference on Image Processing*, volume 3, pages 384–387. IEEE, 1997.

Dali Chen, YangQuan Chen, and Dingyu Xue. Fractional-order total variation image denoising based on proximity algorithm. *Applied Mathematics and Computation*, 257:537–545, 2015.

Qiang Chen, Philippe Montesinos, Quan Sen Sun, and Peng Ann Heng. Adaptive total variation denoising based on difference curvature. *Image and vision computing*, 28(3):298–306, 2010.

Laurent Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.

Original image Noised image Iteration 1 Iteration 10 Iteration 100

Figure 11: The original image, noised image, and three images are the denoised image with iteration 1, 10, and 100 with gradient descent accelerated by the Nesterov method.

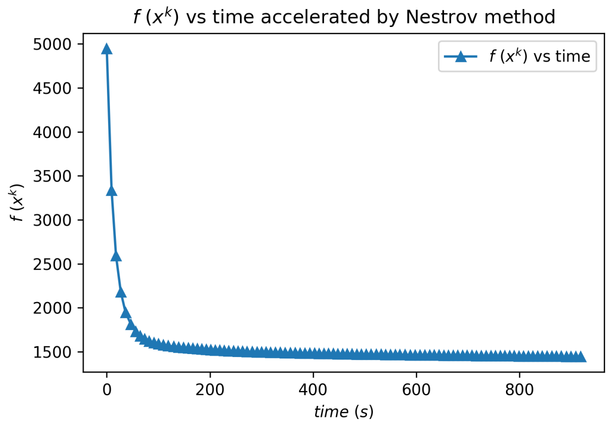

Figure 12: $f(x^k)$ vs. time for gradient discent accelerated by Nestrov method.

Sylvain Durand and Jacques Froment. Reconstruction of wavelet coefficients using total variation minimization. *SIAM Journal on Scientific computing*, 24(5):1754–1767, 2003.

Glenn R Easley, Demetrio Labate, and Flavia Colonna. Shearlet-based total variation diffusion for denoising. *IEEE Transactions on Image processing*, 18(2):260–268, 2008.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Mitliagkas Ioannis. Nesterov's accelerated gradient, stochastic gradient descent. *IFT 6085 - Lecture 6*, 2018.

Ivan W Selesnick, Ankit Parekh, and Ilker Bayram. Convex 1-d total variation denoising with non-convex regularization. *IEEE Signal Processing Letters*, 22(2):141–144, 2014.

DNH Thanh and SD Dvoenko. A denoising of biomedical images. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):73, 2015.

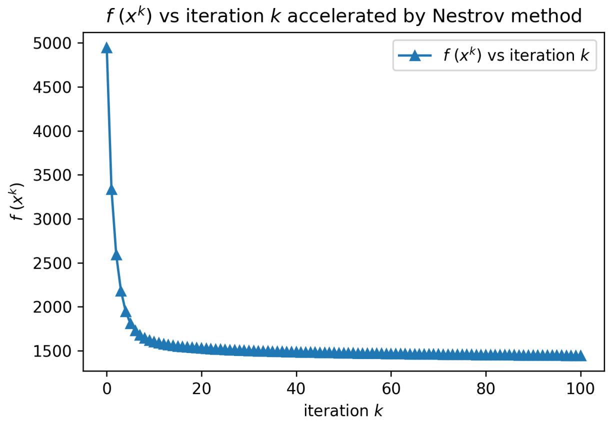

Figure 13: $f(x^k)$ vs. iteration k for gradient discent accelerated by Nestrov method.

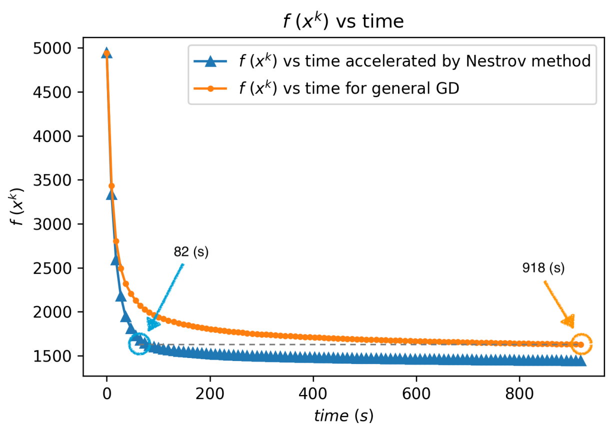

Figure 14: Comparision of $f(x^k)$ vs. time for gradient descent and gradient descent accelerated by Nesterov method.

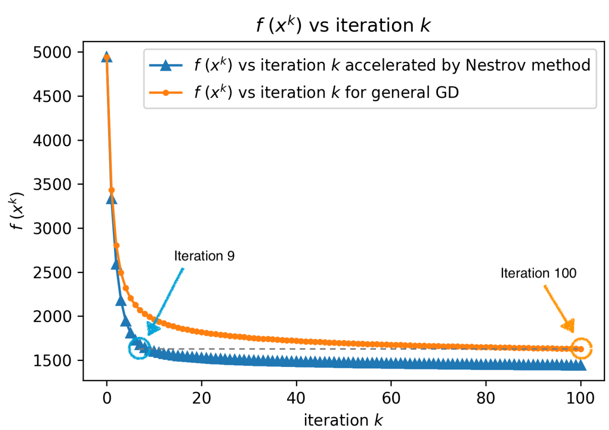

Figure 15: Comparision of $f(x^k)$ vs. iteration k for gradient descent and gradient descent accelerated by Nesterov method.