Convex Optimization Course Project Report

Names: Yuhao Bao, Xu Duan, Jun Gu, Di Luan, Su Xing

(in alphabetic order)_

Content

1 . Introduction

1.1 Dictionary learning model

Given an image in patched form X, to learn dictionary from it, we need to solve the following optimization problem:

$$\underset{D,A}{min}\ \frac{1}{2}\|X - DA\|_F^2 + \lambda\|A\|_0$$

where

- $\|\cdot\|_F^2$ is the sum of the squares of all the elements;
- $\|\cdot\|_0$ returns the number of non-zeros elements;
- $\lambda > 0$ is a balancing weight.

1.1.1 ℓ1-norm with soft-thresholding shrinkage

Apparently, the above problem is non-convex due to $\|\cdot\|_0$. To circumvent this difficulty, we replace it with the ℓ1-norm, leading to

$$\underset{D,A}{min}\ \frac{1}{2}\|X - DA\|_F^2 + \lambda\|A\|_1$$

According to the sample code, we use soft-thresholding shrinkage as a convex approximation of the ℓ1-norm to make the problem optimizable. In the process of dictionary learning, ℓ1-norm regularization is applied to the coefficient matrix A, aiming to induce sparsity.

We modified part of the code to adapt it for color images, and then changed the initial value of matrix C so that it is not equivalent to matrix A. We further adjusted the parameters p、gap、λ、μ following the sample code and obtained the Peak Signal-to-Noise Ratio (PSNR) values for 18 McM images.

1.1.2 Using genetic algorithms adapted to non-convex optimization

In the example algorithms provided by the teacher, the descent methods adopted in the optimization process of D(Dictionary) and A (Coefficient matrix) are both gradient descent methods, which have a high probability of falling into the local optimal solution. Among them, the choice of D plays a decisive role; It is considered that if D can be randomly selected all the time in the optimization process, the probability that the optimization result is a local optimal solution rather than a global optimal solution

can be greatly reduced. Therefore, it is considered to replace the descending method of D with a genetic algorithm that can adapt to non-convex optimization.

$$\min_{D,A} \frac{1}{2}\|X - DA\|_F^2 + \lambda\|A\|_0$$

The detailed introduction of this method is in Section 1.2.7.

1.2 optimization algorithm

Problem:

$$\min_{D} \quad \frac{1}{2}\|X - DA\|^2 + \lambda\|A\|_1$$
$$\text{s.t.} \quad \|d_i\|_2 = 1$$

gradient:

$$\nabla f_0(D) = (-X + DA)A^T$$

Hessian Matrix:

$$\nabla^2 f_0(D) = AA^T$$

1.2.1 Gradient Descent

Gradient descent minimizes the objective function through iterative optimization. In each iteration, the algorithm computes the gradient of the objective function with respect to the parameters, and then updates the parameters along the negative gradient direction to gradually reduce the value of the objective function.

Update:

$$D \coloneqq D - \gamma(L_D)\nabla f_0(D)$$

Advantages:

1) Simple and intuitive: The core idea of gradient descent is intuitive and straightforward, making it easy to understand and implement.
2) Fewer computations per iteration: It only requires the computation of Lipschitz continuous parameters and gradients.

Disadvantages:

1) Sensitivity to step size: The choice of step size significantly impacts the algorithm's performance. Too small of a step size may lead to slow convergence, while too large of a step size may result in oscillations or failure to converge.
2) Poor performance on "valley-shaped" functions: Due to the poor flexibility in selecting the descent direction, gradient descent often oscillates between two sides of a narrow "valley-shaped" objective function, leading to a significantly increased number of iterations.

1.2.2 Conjugate Gradient

The basic idea of conjugate gradient method is to search through a series of conjugate directions in order to converge to the optimal solution more quickly. At each iteration, the conjugate gradient method selects a direction that is conjugate to the previous search direction to reduce the search space.

Update:

$$D \coloneqq \begin{cases} D - \gamma(L_D)\nabla f_0(D), & k = 1 \\ D - \gamma(L_D)\left(\nabla f_0(D)\right)^{\star}, & o.w. \end{cases}$$

Advantages:

1) Fast convergence rate: Compared with gradient descent method, conjugate gradient method usually converges to the optimal solution faster.
2) Symmetric positive definite matrices: The conjugate gradient method is particularly suitable for solving linear systems with symmetric positive definite matrices, which is common in some optimization problems.

Disadvantages:

1) It is difficult to apply to non-quadratic problems: conjugate gradient method is not effective in solving non-quadratic problems, or it cannot converge.
2) Additional information needs to be stored: In order to calculate conjugate directions, the previous gradient and search direction need to be stored, adding some computation and memory overhead.
3) Prone to matrix singularity: When taking the conjugate of the gradient, there may be a case of matrix singularity, resulting in the calculation can not be carried out.

1.2.3 Newton's Method

The basic idea of Newton's method is to use the second-order information of the objective function (Hessian matrix) to guide the search direction, so as to approximate the optimal solution faster. At each iteration, Newton's method uses the second derivative of the objective function (Hessian matrix) to update the current solution vector.

Update:

$$D \coloneqq D - \nabla^2 f_0(D)^{-1}\nabla f_0(D)$$

Advantages:

1) Fast convergence: Compared with first-order methods, Newton's method usually converges to the optimal solution faster.
2) Second-order information utilization: The second-order information of the objective function is used to make the search direction more accurate.
3) One time convergence of quadratic form: For quadratic form, the first step can be convergent to a minimum.

Disadvantages:

1) Hessian matrix calculation, storage and inversion: the cost of computing and storing Hessian matrices can be very high, especially in high-dimensional problems, the cost of inverse operations is extremely high, often leading to a significant increase in the cost of each step of the calculation. Although Newton's method may have fewer iterations, the calculation speed of each step is often much greater than other methods, especially for the case of high dimension in this project.
2) Not necessarily convergent: In some cases, Newton's method may be unstable or not convergent because of the non-positive nature of the Hessian matrix.
3) Initial point selection: The selection of initial point may affect the performance of the algorithm.

1.2.4 Broyden-Fletcher-Goldfarb-Shanno

BFGS method belongs to a kind of quasi-Newtonian method. Its main idea is to dynamically adjust the search direction and step size to approximate the optimal solution by estimating the inverse matrix of the second derivative of the objective function (Hessian matrix).

Update:

$$g^{(k)} = D^{(k+1)} - D^{(k)}$$

$$y^{(k)} = \nabla f_0\left(D^{(k+1)}\right) - \nabla f_0\left(D^{(k)}\right)$$

$$\rho^{(k)} = \frac{1}{(y^{(k)})^T g^{(k)}}$$

$$H^{(k+1)} = \begin{cases} I, & k = 1 \\ \left(I - \rho^{(k)} y^{(k)} \left(g^{(k)}\right)^T\right)^T H^{(k)} \left(I - \rho^{(k)} y^{(k)} \left(g^{(k)}\right)^T\right) + \rho^{(k)} y^{(k)} \left(g^{(k)}\right)^T, & o.w. \end{cases}$$

$$D^{(k+1)} := D^{(k)} - H^{(k)} \nabla f_0\left(D^{(k)}\right)$$

Advantages:

1) Avoiding the calculation of the inverse matrix of Hessian matrix: BFGS method avoids a large number of inverse operations through the first-order estimation of the inverse Hessian matrix, especially for high-dimensional problems, greatly reducing the computational overhead of Newton method. However, compared with first-order methods such as gradient descent, the calculation speed is still slow.
2) Global convergence: BFGS method has global convergence and can be applied to general nonlinear optimization problems.
3) Avoiding the storage of the complete Hessian matrix: BFGS avoids the direct storage and calculation of the Hessian matrix by maintaining the estimation of the inverse Hessian matrix.

Disadvantages:

1) Computational complexity: Although BFGS method avoids inverse operation, it still involves a large number of matrix multiplication, and the computational complexity is relatively high, especially for large-scale problems.
2) Selection of initial matrix: For the estimation of the initial inverse Hessian matrix, the initial selection may have an impact on the performance of the algorithm.

1.2.5 Adaptive Gradient

The adaptive gradient method adaptively adjusts the step size based on the historical gradient information of each parameter. Specifically, the step size of each parameter gradually decreases over time, such that the parameter with a larger gradient during training has a smaller step size, while the parameter with a smaller gradient has a larger step size.

Update:

$$s \coloneqq s + \nabla f_0(D) \cdot \nabla f_0(D)$$
$$\gamma \coloneqq \gamma - \frac{\gamma_0}{\sqrt{s + \varepsilon}}$$
$$D \coloneqq D - \gamma \nabla f_0(D)$$

Where

- s is the cumulative gradient sum of squares,
- γ_0 is the global initial step,
- ε is a small positive real number, which is used to prevent cases where the denominator is 0.

Advantages:

1) Each parameter has its own rhythm: parameters with larger gradients have smaller steps, while parameters with smaller gradients have larger steps.
2) Adaptive step size: Adjust the step size adaptively according to the historical gradient information, so it can better adapt to the change of different parameters. With the number of iterations, the step size keeps shrinking, which also conforms to the intuitive idea that the step size is smaller when approaching the minimum value in the later stage.
3) Sparse data adaptability: Due to the accumulation of historical gradient squares, adaptive gradient method can provide a larger step size for sparse gradient parameters, which is conducive to sparse data training.
4) The cost of single-step calculation is small: the calculation of Lipschitz continuous parameters is avoided, the calculation is simplified, and many calculations are low-dimensional, and the calculation speed is fast.

Disadvantages:

1) Step size decays too quickly: As training progresses, the accumulated gradient square may become very large, causing the gradient to decay too quickly or

even close to zero. In the early stage of iteration, the step size decays too fast, and the convergence speed is insufficient in the late stage, which even causes the algorithm to stop convergence in advance.

2) No consideration of the stability of the gradient: The adaptive gradient method treats all historical gradient squares equally, without considering the stability of the gradient. In some cases, this can lead to over-scaling of the learning rate.
3) Sensitive to initial global step: A given initial global step that is too small will result in convergence that is too slow, while a given initial global step that is too large will result in non-convergence.

1.2.6 Improved Adaptive Gradient

In order to avoid the shortcoming of the adaptive gradient method, the hyperparameter β can be added to evolve into the Root Mean Square Propagation method, that is, using β to carry the weighted moving average of the gradient square. The Adaptive Delta method can be developed by replacing the given β with Δγ, which is a first-order method to simulate the second-order Newton method. The Momentum method can also be integrated into Adaptive Momentum Estimation to further suppress oscillations. However, due to the high dimension of matrix A and D in this project, these methods often introduce a large amount of extra computation, but the improved effect often can not compensate for these extra computation, so the effect is often poor.
Update:

$$\tilde{L}_D = \|AA^T\|_2 + 0.1$$
$$\gamma_s = \frac{1.9}{\tilde{L}_D}$$
$$s \coloneqq s + \nabla f_0(D) \cdot \nabla f_0(D)$$
$$\gamma \coloneqq \gamma - \frac{\gamma_s}{\sqrt{s + \varepsilon}}$$
$$D \coloneqq D - \gamma \nabla f_0(D)$$

Advantages:

1) Inherits the advantages of adaptive gradient method: parameters with larger gradient have smaller step sizes, while parameters with smaller gradient have larger step sizes. And the step size can be adjusted adaptively according to the historical gradient information, so it can better adapt to the change of different parameters. With the number of iterations, the step size keeps shrinking, which also conforms to the intuitive idea that the step size is smaller when approaching the minimum value in the later stage. The calculation is low dimensional and the calculation speed is fast.
2) Longer step length in the early stage: It avoids the artificial setting of the initial global step size, enhances the versatility of the method, and makes the step length in the early stage longer.
3) Avoid excessive decay of the late step size: Because the function of Lipschitz parameter is used to constrain and estimate the step size, the decay effect of the late iteration is not obvious. At the same time, the stability of gradient is

considered and excessive scaling of step size is avoided.

Disadvantages:

Poor effect on low dimensional problems: Because it still requires a lot of calculation, it is often not as good as Newton's method for low dimensional problems.

In this project, the idea of adaptive incremental method is absorbed and the adaptive gradient method is improved. In the adaptive increment method, the change of step size $\Delta\gamma$ is used to replace the given β for moving weighted average, but the oscillation is still large. In this project, a function of the Lipschitz parameter of each step is used instead of β, avoiding the disadvantage of the step decay too fast when all the global initial step size γ_0 is used, and the oscillation problem caused by the step size variation $\Delta\gamma$ is avoided. Compared with adaptive gradient method and adaptive increment method, the calculation amount is increased, but it can greatly reduce the number of iteration steps, so the performance is greatly improved.

1.2.7 Genetic Algorithm

The basic idea of genetic algorithms (GAs) is to mimic the process of natural selection and evolution to optimize solutions to complex problems. Genetic algorithms operate on a population of potential solutions, using evolutionary operators such as selection, crossover, and mutation to iteratively improve the quality of solutions over generations.

1. Selection Operator:

   Select individuals from the current population based on their fitness, with higher-fitness individuals having a higher probability of being chosen.

Update:

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Where P_i is the probability of selecting individual i, and f_i is the fitness of individual i.

2. Crossover Operator:

   Combine genetic information of two parents to create new individuals, simulating the crossover of genetic material in natural reproduction.

Update:

$$\text{Offspring}_i = \text{Crossover}\left(\text{Parent}_{\text{random1}},\ \text{Parent}_{\text{random2}}\right)$$

Where Crossover is a function that combines genetic information from two parents to create an offspring.

3. Mutation Operator:

   Introduce small random changes to individual solutions, simulating genetic mutations.

Advantages:

1) Parallel Search: Genetic algorithms explore multiple potential solutions simultaneously, allowing for parallel search in the solution space.
2) Global Search: GAs are well-suited for exploring a large and complex solution

space, making them effective for global optimization problems.

3) Adaptability: GAs can adapt to different types of problems without requiring a deep understanding of the problem structure.

Disadvantages:

1) Convergence Speed: Genetic algorithms might take more iterations to converge compared to some gradient-based methods, especially in problems where smoothness and continuity are crucial.
2) Parameter Sensitivity: Performance can be sensitive to the choice of parameters such as mutation rate, crossover rate, and population size.
3) Noisy or Discontinuous Fitness Functions: GAs may struggle with noisy or discontinuous fitness functions, as they rely on the evaluation of fitness to guide the search.

In this project, first of all, the patch size is set as 8×8 pixels, and the patch interval of each row/column is 4 pixels. The size of a Dictionary is 100 atoms.

Considering the genetic algorithm for D, 10 "genes" of the first generation D (parent generation) will be generated randomly, and each gene will contain 100 non-repeating patches learned from the original image.

After the K-generation "Dictionary" is generated by genetic algorithm, the appropriate A (Coefficient matrix) is found by gradient descent method for each "gene". The number of iterations is 12000 times, and the step size selection method is the same as the example code.

After obtaining the A (Coefficient matrix) corresponding to the K-generation "Dictionary" and each "gene", the error is calculated for each Dictionary:

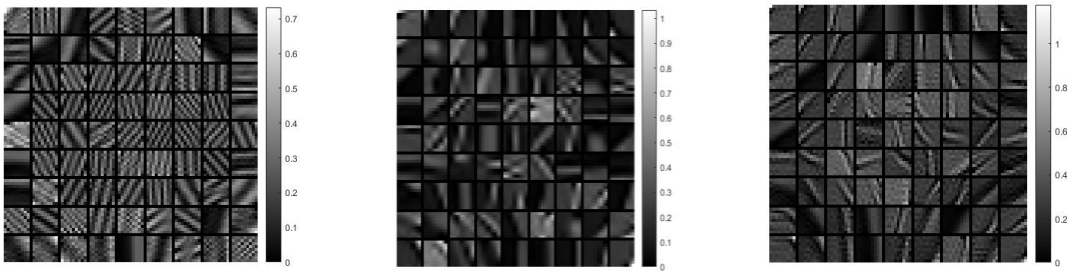
$$E = \frac{1}{2}\|X - DA\|_F^{\,2} + \lambda\|A\|_0, \lambda = 0.5$$

The two groups with the smallest E in the "gene" (denoted as D_1 and D_2) are directly inherited. Copy D_1 and D_2 three times each. The Atom of even-numbered positions in D_1 of the three groups of copied patches was randomly replaced with the first or last patch adjacent to each other in patches of the original map, and the Atom of even-numbered positions in D_2 of the three groups of copied patches was randomly replaced with any of the front and back 40 patches of patches of the original map for mutation. Then each Atom of D_1 and D_2 in the three groups after mutation is rotated forward to avoid the mutation of Atoms in fixed positions all the time. Further, two new "genes" are randomly generated using the same method as the initialization method. At this point, the obtained 2+2×3+2=10 "genes" is the k+1 generation "genes".

For grayscale images, there is only one color channel and only one genetic algorithm cycle. For color images, the three RGB color channels will be iterated once by genetic algorithm, and finally the processed image will be pieced together. The symbol of the end of the iteration of genetic algorithm is that the two "genes" (D_1 and D_2) with the smallest E in the three successive generations of "genes" are the same.

1.3 Basic summary of your obtained results.

First, Figures 1.1 and 1.2 are the dictionaries learned using ℓ1-norm with soft-thresholding shrinkage method with gradient descent.



(a) Barbara (b) Cameraman (c) Lena

Figure 1.1 : Dictionary learned from three grayscale images.

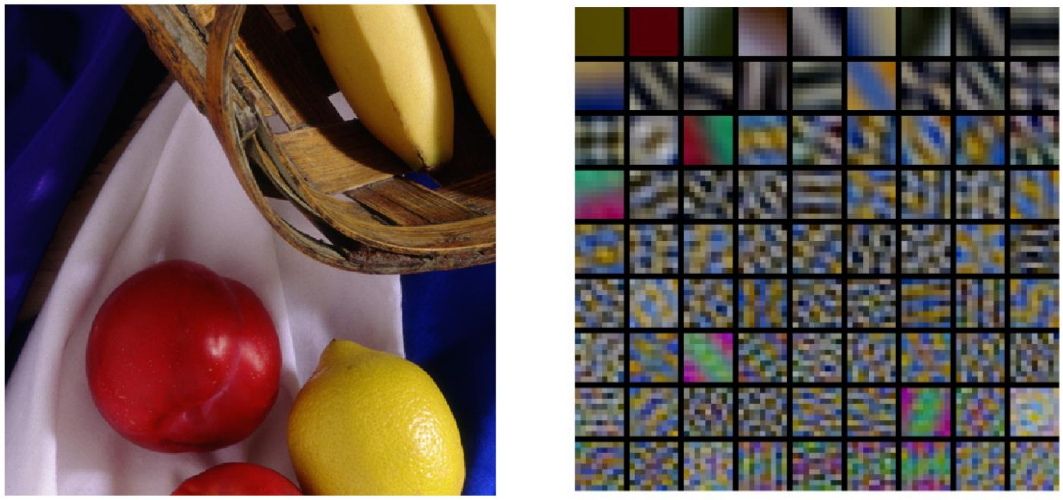


Figure 1.2 : the learned dictionary of McM14

In addition, As shown in Figures 1.3 and 1.4, we use the genetic algorithm introduced in 1.2.7 to obtain the dictionary learning results for grayscale and color images.

(a) Barbara (b) Cameraman (c) Lena

Figure 1.3 : Dictionary learned from three grayscale images.
(Genetic Algorithm)

Figure 1.4 : Dictionary learned from McM01 and McM04.
(Genetic Algorithm)

1.3.1 Evaluate the selected parameters

The size of the patch, denoted by p, determines the dimensionality of the target representation vector. Increasing p significantly increases the number of rows in the matrix X.

The step size gap during each iteration determines the number of patches. Reducing gap significantly increases the number of rows in matrix X, i.e., the number of vectors to be represented.

The ratio of p to gap determines the number of times each pixel is learned. For example, if p = 9 and gap = 4, each pixel is, on average, learned $9/4$ times, and the final result is the average of these $9/4$ outcomes. Generally, when p is larger and gap is smaller, each pixel is learned more times, leading to more accurate results. However, at the same time, computational complexity significantly decreases with the increase of p and the decrease of gap.

Taking the example of the color image McM01 (see table McM01.xlsx), if gap is fixed at 4 and p increases from 7 to 15, the average PSNR increases from 26.77 to 27.91. However, the computation time increases from about 10 minutes to about 1 hour. It's worth noting that PSNR does not monotonically increase with p; for example, when p increases to 17, PSNR shows a decreasing trend. If p is fixed at 15 and gap decreases from 6 to 3, the average PSNR increases from 27.45 to 28.02. The computation time increases from about 35 minutes to about 70 minutes.

Table 1.1 The processing results color picture McM01

size	gap	lambda	mu		R	G	B	
9	4	1.5	50		24.86	24.69	24.67	24.74
9	4	1.5	90		27.88	27.5	27.22	27.53333
9	4	1.5	95		27.93	27.53	27.23	27.56333
9	4	1.5	100		27.92	27.5	27.18	27.53333
9	4	1.5	110		27.78	27.33	26.99	27.36667
9	4	1.5	200		25.42	27.76	24.63	25.93667
9	4	2	95		27.83	27.39	27.06	27.42667
9	4	1.2	95		26.97	26.67	25.51	26.38333
7	4	1.5	95		27.08	26.73	26.52	26.77667
11	4	1.5	95		28.12	27.71	27.36	27.73
13	4	1.5	95		28.25	27.88	27.47	27.86667
15	4	1.5	95		28.27	27.94	27.51	27.90667
17	4	1.5	95		28.25	27.93	27.45	27.87667
15	6	1.5	95		27.82	27.48	27.07	27.45667
15	3	1.5	95		28.41	28.07	27.6	28.02667

λ is the regularization parameter during the learning process. The selected parameter is the ratio by which it is reduced every 500 iterations, without choosing an initial value. Empirically, the optimal range for this parameter is very small, around 1.5.

μ is the regularization parameter during the reconstruction process. Its range is relatively large, and it has a significant impact on the results. Since adjusting μ does not change the computation time significantly, and adjusting μ only requires rerunning the reconstruction process, it is relatively fast. Therefore, μ is the key parameter to adjust. Empirically, adjusting μ follows the following rule: if the intensities of the RGB channels tend to be the same (e.g., RGB vector is μ, representing black (0, 0, 0), white (255, 255, 255), or middle gray), in this case, a larger μ leads to better results. For example, if McM04 is mainly composed of white, increasing μ from 100 to 450 increases the average PSNR from 24.11 to 31.75. On the contrary, if the intensities of the RGB channels vary, such as green (255, 0, 0), red (0, 255, 0), blue (0, 0, 255), or combinations of two channels, a smaller μ leads to better results. If McM13 is mainly composed of yellow (255, 255, 0), the optimal value for μ is around 105.

This phenomenon can be explained as follows: if the intensities of the RGB channels tend to be the same, random noise generally disrupts this pattern of identical channels. Since the dictionary D is learned from the original image, the channels of each atom in D are naturally the same. Therefore, D cannot capture different RGB patterns, and a lighter weight needs to be assigned to D, hence the need to increase the regularization parameter μ.

By adjusting these parameters, we evaluated the impact on the pre-processing results, as shown in Table 1.2.

Table 1.2 Evaluation of the effect of adjusting parameters

	p	gap	λ	μ		R	G	B	mean
McM01	15	3	1.5	95		28.41	28.07	27.6	28.026667
McM02	9	4	1.5	150		30.1	30.69	29.9	30.23
McM03	13	4	1.5	260		29.96	29.73	28.78	29.49
McM04	9	4	1.5	450		32.18	32.48	30.63	31.763333
McM05	9	4	1.5	200		30.54	30.25	29.3	30.03
McM06	9	4	1.5	180		31.57	30.89	30.73	31.063333
McM07	13	4	1.5	200		30.95	30.9	30.34	30.73
McM08	9	4	1.5	350		32.23	32.87	32.44	32.513333
McM09	9	4	1.5	180		30.5	31.15	31.17	30.94
McM10	15	4	1.5	120		31.67	31.91	31.64	31.74
McM11	15	4	1.5	110		32.07	31.57	32.81	32.15
McM12	17	4	1.5	125		32.96	32.12	32.24	32.44
McM13	15	4	1.5	105		34.95	35.52	33.89	34.786667
McM14	15	4	1.5	125		33.23	33.75	32.25	33.076667
McM15	15	4	1.5	110		32.16	33.41	33.22	32.93
McM16	15	4	1.5	120		29.37	28.01	30.2	29.193333
McM17	17	4	1.5	110		29.16	29.08	29.43	29.223333
McM18	17	4	1.5	195		29.39	29.27	31.29	29.983333

1.3.2 Evaluate the descent method

Firstly, in the case of grayscale images simple dictionary, we choose these descent methods: gradient descent method, Newton method, BFGS method, adaptive gradient

method and improved adaptive gradient method. The single-step descent speed is shown in Figure 1.5.

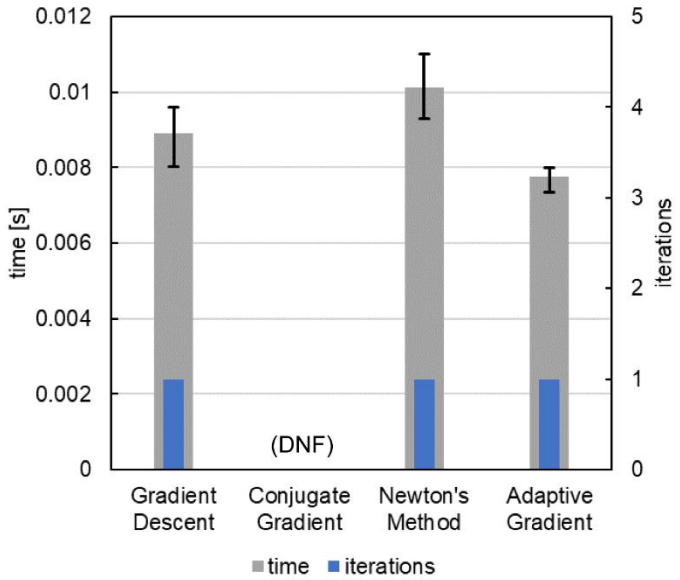


Figure 1.5 : Time of single step descent for different methods

For the simple dictionary of grayscale images, except for the singular situation of conjugate gradient method which cannot be completed, all the other methods can meet the residual requirement by dropping all the time. Among them, in the conjugate gradient method, due to the characteristics of the initial value, the matrix singularity can not be completed due to the rounding of the computer's internal calculation, which also reflects one of the shortcomings of the conjugate gradient method. For Newton's method, because it needs to calculate the inverse matrix of Hessian matrix, it consumes a lot of time, resulting in a slow single step speed, which is also reflected in the learning of the improved dictionary of color graphs. In the adaptive gradient method, the global initial step size $\gamma_0=1$ is set, although there are more operations in each step, many of them are low-dimensional or even scalar calculations, which is faster than the gradient descent method.

In the case of improving the dictionary of color graph, we use several descent methods: gradient descent, Newton method, BFGS method, adaptive gradient method, improved adaptive gradient method. The decline speed of these five methods is compared, as shown in Figure 1.6.

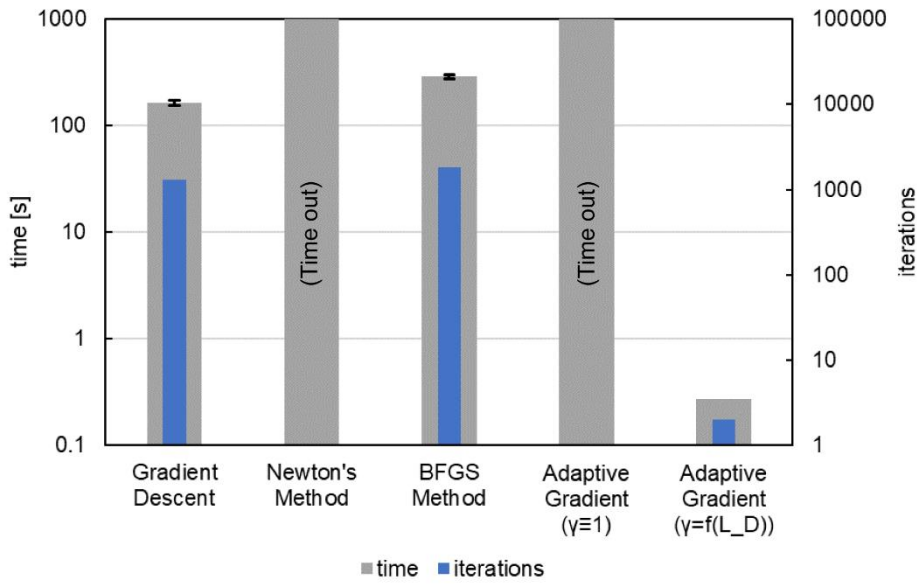


Figure 1.6 : Single-step descent times and iterative steps for different methods (both on a logarithmic scale, Where, Adaptive Gradient (γ≡1) is the Adaptive Gradient method, and Adaptive Gradient (γ=f(L_D)) is the improved one)

the number of iteration of gradient descent method is about n=1314, and it takes about t=163s. Newton's method times out due to the need for a large number of high-dimensional matrix inversion operations; The number of iterations of BFGS method is about n=1837, and it takes about t=288s. Although it avoids a large number of high-dimensional matrix inversion operations of Newton's method, it still requires a large number of matrix multiplication operations, so its calculation amount is still greater than that of gradient descent method, and due to the rounding error of computer calculation, etc. The error of estimating the inverse matrix of Hessian matrix is still large, and the number of iterations is also higher than that of gradient descent method, so the effect is not good. Although the residuals of adaptive gradient method were once smaller than gradient descent method and BFGS method in the early stage, because the step size of adaptive gradient method declined too fast in the late stage, when the number of iteration steps exceeded 800, the residuals declined slowly and time out. The iterative times of the improved adaptive gradient method are about n=2, and it takes about t=0.3s. Because it combines the advantages of low computation of the gradient descent method and the adaptive gradient method, and the advantages of the Newton method for selecting the direction of descent, it can fall within the tolerance in two steps, and its calculation speed is significantly improved compared with the Newton method.

In addition, we use genetic algorithm to process grayscale images. Corresponding PSNR values, corresponding processed images and patches are obtained.

2. Numerical results

2.1 Task 3: color images denoising

First, on the basis of the teacher's example code that uses LASSO to achieve images denoising, we modified part of the code to adapt to the color picture, and then adjust the parameters to get the optimal value.

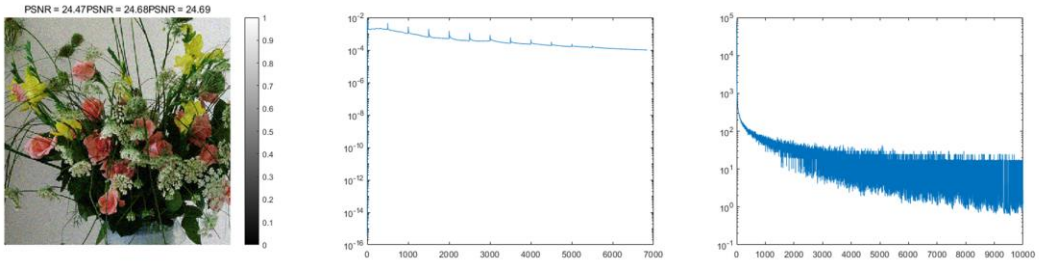
Table 2.1: PSNR values of 18 McM images.

	Red Channel	Green Channel	Blue Channel	Average of three
McM01	33.74234077	33.43900631	33.01178331	33.39771013
McM02	38.95979933	39.68324243	38.32986877	38.99097018
McM03	39.68729215	38.88349616	38.2285969	38.9331284
McM04	46.73540852	45.67196584	43.87278566	45.42672
McM05	43.21394135	43.77063341	42.24097149	43.07518209
McM06	40.71895106	40.428633	39.09874774	40.0821106
McM07	38.8511658	39.14353436	37.86531091	38.62000369
McM08	36.45161692	36.8425202	37.07822198	36.79078637
McM09	38.47959657	38.50774611	37.47669712	38.15467993
McM10	36.45836024	36.93582863	36.27720842	36.55713243
McM11	31.42377874	30.54075926	32.29899495	31.42117765
McM12	36.20240338	35.67748538	35.12188765	35.6672588
McM13	33.81113767	35.44479562	33.0388777	34.09827033
McM14	34.31555043	34.67013018	33.0649681	34.01688291
McM15	31.54718912	33.22585276	32.50959965	32.42754718
McM16	32.94289906	30.87191946	32.58082134	32.13187995
McM17	34.6886241	34.28128043	34.46157522	34.47715992
McM18	34.86225225	34.21650492	36.62239577	35.23371764

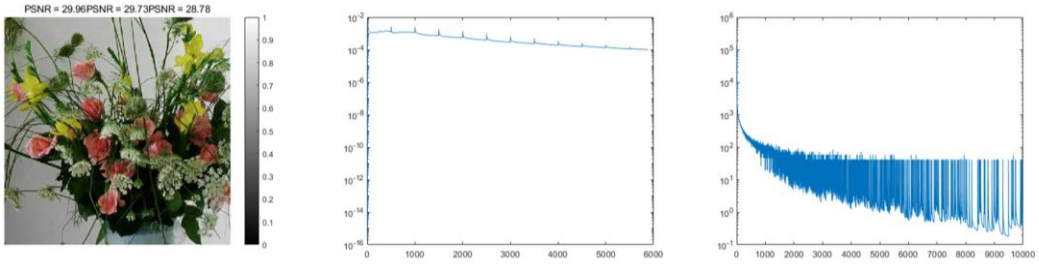
Table 2.2: PSNR values of 18 noised McM images.

	Red Channel	Green Channel	Blue Channel	Average of three
McM01	22.10115807	22.08483094	22.12430702	22.10343201
McM02	22.08913362	22.12198734	22.11767621	22.10959906
McM03	22.11304449	22.12283435	22.10692433	22.11426773
McM04	22.10189385	22.11924021	22.13804614	22.11972673
McM05	22.11176981	22.09841174	22.1162559	22.10881248
McM06	22.09628357	22.10553808	22.11824368	22.10668844
McM07	22.10062112	22.11876386	22.12767328	22.11568609
McM08	22.10190861	22.11833707	22.10584304	22.10869624
McM09	22.1104843	22.10795686	22.09495662	22.10446593
McM10	22.09660069	22.12707287	22.12256991	22.11541449
McM11	22.11724931	22.10700275	22.1116296	22.11196055
McM12	22.12802877	22.10973469	22.09827413	22.11201253
McM13	22.10274646	22.09362642	22.1197805	22.10538446
McM14	22.10554975	22.09762833	22.10014095	22.10110634
McM15	22.11614555	22.09800552	22.12780957	22.11398688
McM16	22.09431187	22.09097603	22.10823505	22.09784098
McM17	22.10813582	22.1133089	22.10318871	22.10821115
McM18	22.12412012	22.12447466	22.13156204	22.12671894

We select different parameters for dictionary learning and noise reduction of noisy images, and the change of parameters directly affects the learning of results. For example, in Figure 2.1, is the denoising effect in the McM03 image.



(a) Bad (b)Bad learn (c)Bad Reconstruct



(a) Good (b)Good learn (c)Good Reconstruct

Figure 2.1 The effect of two different parameters
(bad: p=9, gap=4, lambda=1.5, mu=100; good: size=13, gap=4, lambda=1.5, mu=260)

In addition, we use genetic algorithm to process color images denoising. The corresponding PSNR values are obtained.

Table 2.3 PSNR values of McM01~McM05 images.

Image name	PSNR		
	R	G	B
McM01	35.97	36.57	36.51
McM02	34.07	37.56	36.45
McM03	36.64	35.47	36.02
McM04	34.73	34.80	34.11
McM05	35.78	35.38	36.42

2.2 Task 4: unknown ground truth

First, on the basis of the teacher's example code, we modified part of the code to

adapt to the color picture, and then changed the initial value of C, so that it is not A, the following is the result of processing all the color pictures

Table 2.4: PSNR values of 18 McM images

	Red Channel	Green Channel	Blue Channel	Average of three
McM01	27.92595343	27.51100511	27.18742932	27.54146262
McM02	27.91237562	28.20492661	27.94900716	28.02210313
McM03	22.9981419	23.34759287	22.90844308	23.08472595
McM04	23.15043474	23.06504338	23.05131553	23.08893122
McM05	25.70224221	25.69717377	25.69527792	25.6982313
McM06	27.81858896	27.60596751	27.65312206	27.69255951
McM07	27.07551246	27.07689752	26.927769	27.02672633
McM08	23.06431209	24.25402453	23.3882391	23.56885857
McM09	27.53240714	27.85671516	27.80861606	27.73257945
McM10	29.59086358	29.70017729	29.58519638	29.62541242
McM11	31.01329844	30.59382695	31.68776103	31.09829547
McM12	27.74099255	27.50450398	27.5595938	27.60169677
McM13	33.86120767	34.15122198	33.1288201	33.71374992
McM14	30.28492738	30.40622795	29.8700681	30.18707448
McM15	31.26021798	32.13185334	32.0249755	31.80568227
McM16	26.63149411	26.12766162	27.09579965	26.61831846
McM17	27.76367884	27.68189599	27.959469	27.80168128
McM18	25.4275817	25.42080227	25.82037007	25.55625134

In addition, we use genetic algorithm to process noisy color images. The corresponding PSNR values are obtained.

Table 2.5 : PSNR values of McM01~McM05 images

Image name	**PSNR**		
	R	**G**	**B**
McM06	28.04	28.02	27.97
McM07	26.70	26.60	26.35
McM08	27.05	27.16	27.04
McM09	27.52	27.62	27.64
McM10	28.01	28.14	28.07

3 Conclusion

In this project, we delved into the realm of dictionary learning models and optimization algorithms to address the challenges of grayscale and color image processing. Our primary focus was on two key aspects: dictionary learning and descent methods. The following summarizes our findings and achievements:

We applied the ℓ1-norm with soft-thresholding shrinkage to grayscale images, demonstrating its efficacy in image denoising. The approach successfully reduced noise while preserving important features. And we also employ genetic algorithms tailored for non-convex optimization, we extended our work to color images. The model showcased its ability to adapt and optimize, showcasing promising results even in scenarios with unknown ground truth.

We explored a variety of optimization algorithms to enhance the efficiency of our dictionary learning model. Each algorithm presented unique advantages, catering to different aspects of the problem:

1) Gradient Descent: A fundamental optimization method, gradient descent, demonstrated its reliability in finding the minimum of the objective function.

2) Conjugate Gradient: Offering improvements over gradient descent, the conjugate gradient method displayed faster convergence in certain cases.
3) Newton's Method: By utilizing second-order information, Newton's method exhibited enhanced convergence, particularly beneficial for complex, non-linear problems.
4) Broyden-Fletcher-Goldfarb-Shanno (BFGS): This quasi-Newton method further refined our optimization efforts, showing competitive performance in finding optimal solutions.
5) Adaptive Gradient: We introduced the adaptive gradient algorithm, adapting learning rates to different parameters, enhancing the model's ability to converge efficiently.
6) Improved Adaptive Gradient:** Building upon the adaptive gradient, our improved version fine-tuned the learning rates dynamically, aiming for more precise convergence.
7) Genetic Algorithm: Incorporating genetic algorithms into the optimization process demonstrated versatility, especially in non-convex scenarios where traditional methods might struggle.

Thorough evaluation of selected parameters led to optimized settings for our dictionary learning model, ensuring its effectiveness in grayscale image denoising. In addition, our investigation into descent methods revealed nuances in their performance, with certain methods excelling in specific scenarios. This information is valuable for selecting the most appropriate approach based on the characteristics of the problem at hand.

In summary, we have conducted a comprehensive exploration of dictionary learning models and optimization algorithms through this project. These methods have been successfully applied to de-noising grayscale and color images, as well as valuable insights gained from parameter evaluation and descent method analysis