Yue Pan
yuepan@student.ethz.ch

Assignment 3

Image Interpretation
October 23, 2019

Exercise 1

Topic: Linear Regression for polynomial fitting. For this exercise, please refer to *ii_fit_poly.m* and *ii_apply_poly.m*.

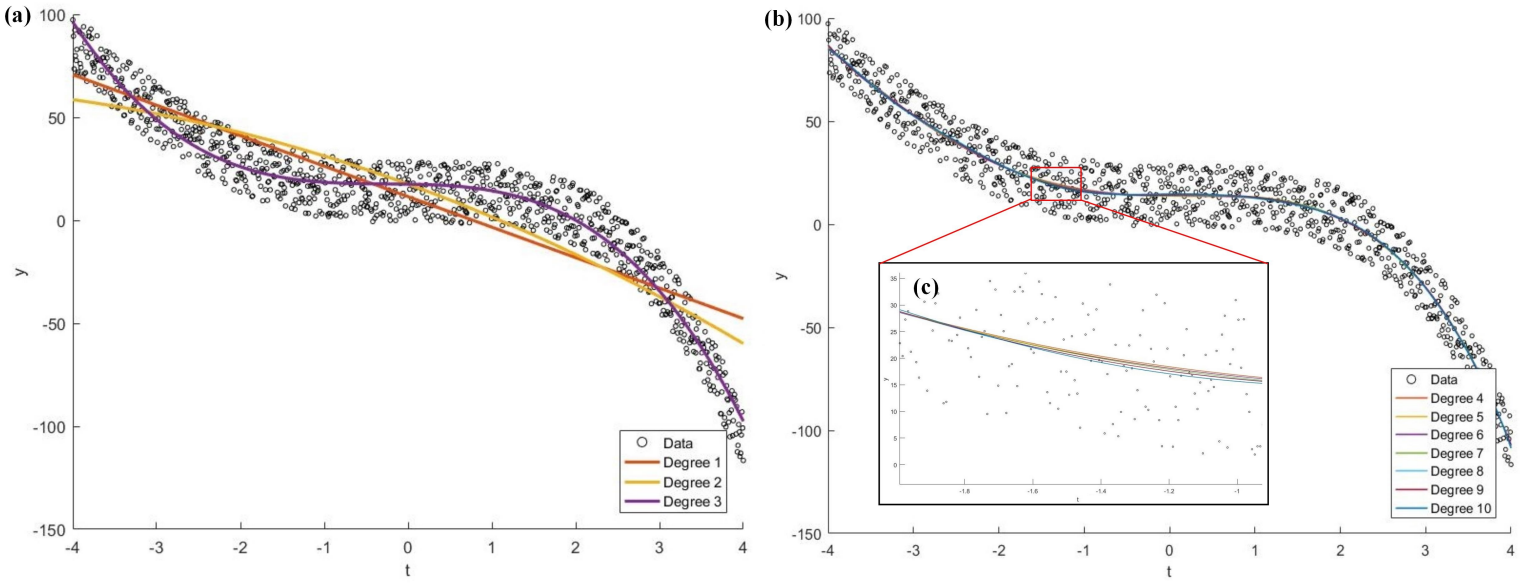

Figure 1: Regression result of different degree of fitting polynomial.

The fitted polynomial should obey the function as shown in Eq.1, in which k is the degree. Here, I would discuss which degree works the best from 1 to 10 on this task.

$$y = \theta_k t^k + \theta_{k-1} t^{k-1} + \cdots + \theta_1 t + \theta_0 \tag{1}$$

According to Fig.1(a), when degree is less than 4, the polynomial cannot fit the data well. As shown in Fig.1(b) and (c), polynomials whose degree ranging from 4 to 10 are all visually fitted well and almost overlapped with each other. For polynomial with high degree, the parameters θ for high degree terms (> 7) are very small (< 0.001), which only contributes a bit to the polynomial function. But for medium degree (4 6), estimated coefficient θ is about 0.1, which could not be ignored. Generally speaking, models with more parameters would fit the training dataset better, but may be poorer at prediction (overfitting).

Since the data for fitting is not noisy enough, the phenomenon of overfitting is not obvious. So I manually add gaussian noise (standard deviation $\sigma = 30$) to the data, and apply the same regression procedure to it. As shown in Fig.3(a) and (b), the 10 degree polynomial twists too much along with the data points, which indicates overfitting. In conclusion, I think degree **6** is the best among all the choices since the polynomial would neither underfit nor overfit.

Exercise 2

Topic: Ridge regression for polynomial fitting. For this exercise, please refer to *ii_fit_poly_ridge.m* and *ii_apply_poly.m*.

As for ridge regression, a regularization term is added to the cost function so that the smaller parameters with low fitting residual would be preferred, thus getting rid of the overfitting problem. The key is to balance the fitting and regularization terms, which means the setting of λ should be deeply studied. In order to amplify

the phenomenon of overfitting and to study ridge regression's effect on overfitting with regard to different λ, the noisy dataset introduced in Exercise 1 is used again.

It is found that when λ is set too small, the model tends to be overfitting ($\lambda = 10^{-6}$ and $\lambda = 10^{-3}$ in Fig.3(c) and (d)) and when λ is set too large, the model tends to be underfitting ($\lambda = 10^{6}$ and $\lambda = 10^{3}$ in Fig.2). In conclusionthe parameters of my best model is (degree=**6**, λ=**1**).

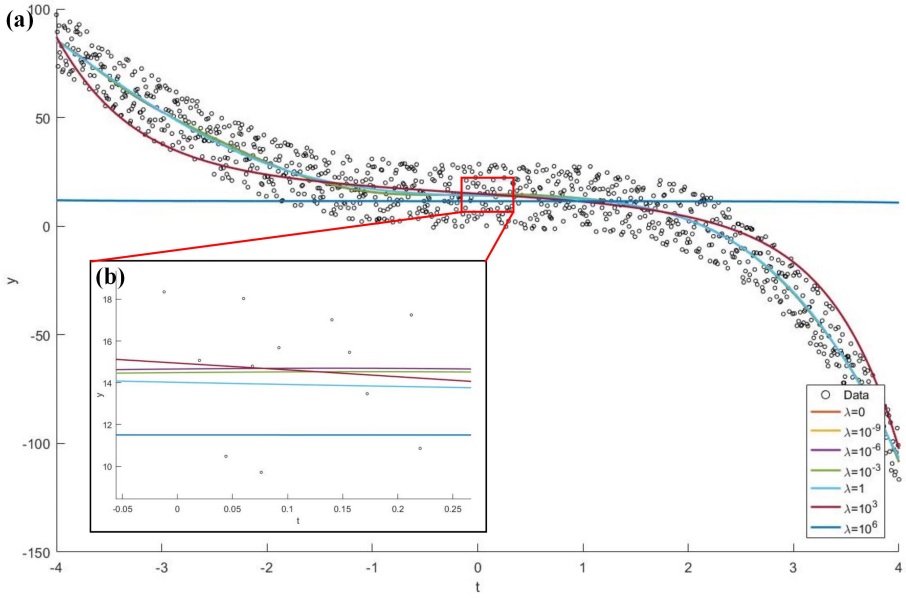

Figure 2: Regression result of different degree of fitting polynomial (degree=6).

Exercise 3

Topic: Right face prediction using linear regression. For this exercise, please refer to *ii_predict_face.m* and *ii_predict_face_g_n.m.* .

This problem can be simply modeled as a fully connected neural network without any hidden layer. The input layer consists of all the pixels on the left half of the face and one constant neuron while the output layer consists of all the pixels on the right half of the face. Then we'd like to learn the linear coefficients θ from the training data and apply it to predict right face of unseen images. Denote the number of training images, pixel number in the left half face and right half of face respectively as K,L,R, Eq.2 can be drawn. Then the unknown coefficients θ is calculated as Eq.3. Though the coefficients' number is way more than sample number K and the matrix X is rank deficient, we can still estimate θ from limited data. The prediction result using estimated θ is shown in Fig.4. Although the prediction results are weird and even a bit creepy, they do like faces.

The alternative method is to set a initial guess of θ using the face symmetry principle and adopt an iterative least square fitting (regression) to update θ. In this case, the Gaussian-Newton method is applied to minimize the cost function. Unfortunately, this method ends up with poor results as shown in the fourth column of Fig.6 (prediction gray value is out of bound).

$$\underset{K\times(L+1)}{X}\quad\underset{(L+1)\times R}{\theta} = \underset{K\times R}{Y} \tag{2}$$

$$\hat{\theta} = \left(X^T X\right)^{-1} X^T Y \tag{3}$$

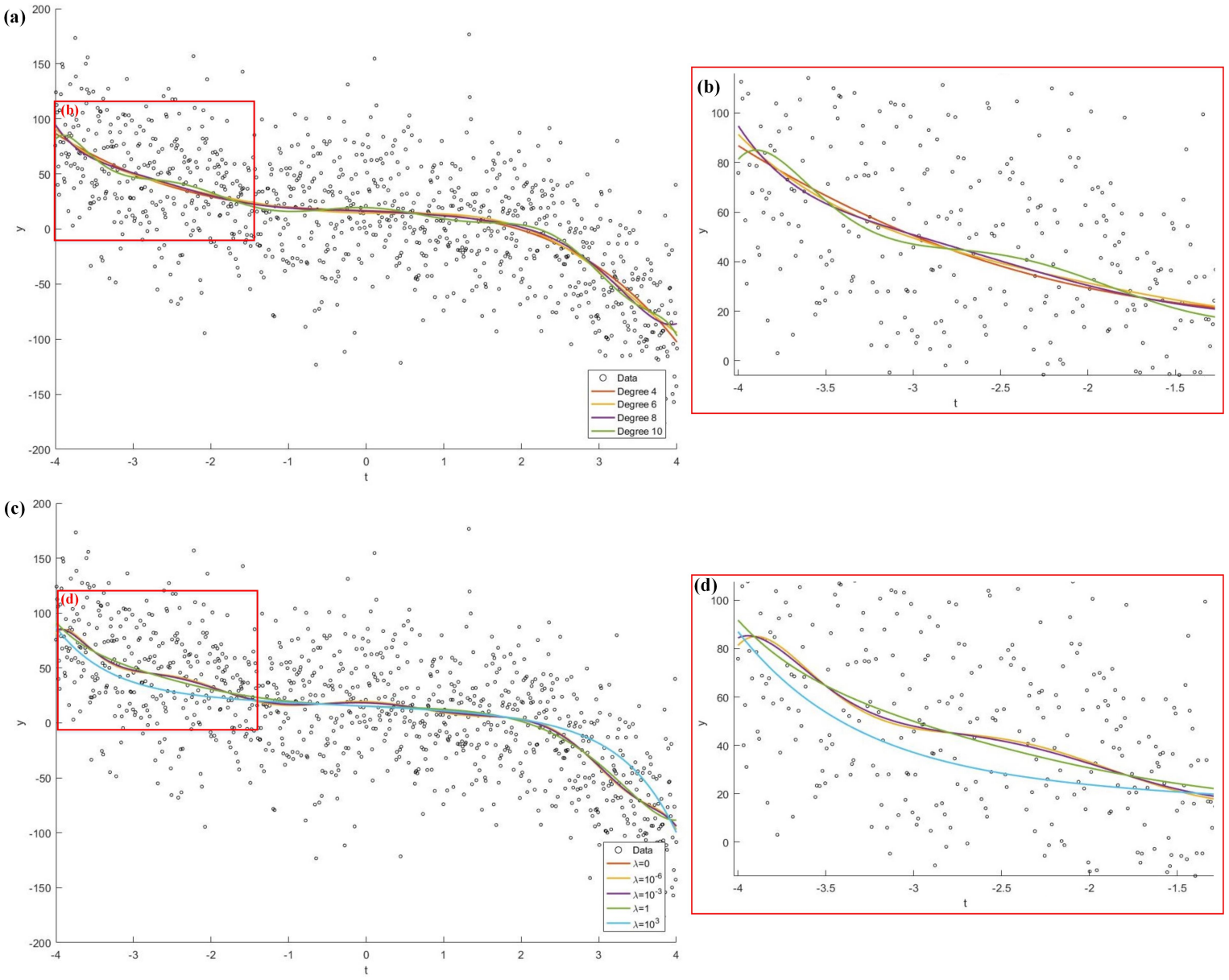

Figure 3: Regression result on a noisy dataset: (a) linear regression of different degree of fitting polynomial, (b) detailed zoom-in part of (a), (c)ridge regression of polynomial with 10 degrees under different λ values, (d) detailed zoom-in part of (c).

Exercise 4

Topic: Right face prediction using ridge regression. For this exercise, please refer to *ii_predict_face_reg.m* and *ii_predict_face_l_m.m*.

Ridge regression can also be used for this face prediction application. Add a regularization term to the cost function and θ can be estimated as Eq. 4. For different λ (ranging from 10^{-7} to 10^5), ridge regression's results are shown in Fig.5. It's clear that results achieved by using ridge regression are way more better than those in Exercise 3. According to Exercise 2, λ should neither be too small nor too large in order to get a promising result. In this case, results for $\lambda = 10^5$ show a kind of blurring due to the underfitting problem. However, according to

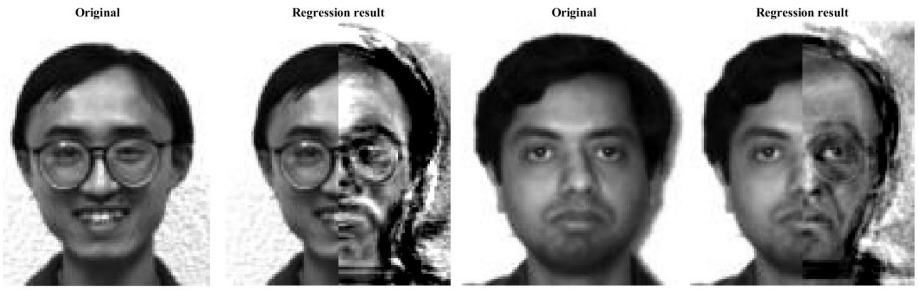

Figure 4: Right face prediction result on small-size image dataset using linear regression

the experiment, the regression results are almost the same for $10^{-7} \leqslant \lambda \leqslant 100$. When $\lambda < 10^{-8}$, it's found that the result become black and white mosaic and when $\lambda = 0$, ridge regression degrades to classic linear regression (the same result as Exercise 3).

$$\hat{\theta} = \left(X^T X + \lambda I\right)^{-1} X^T Y \tag{4}$$

$$\Delta\hat{\theta} = \left(J^T J + \lambda I\right)^{-1} J^T R \tag{5}$$

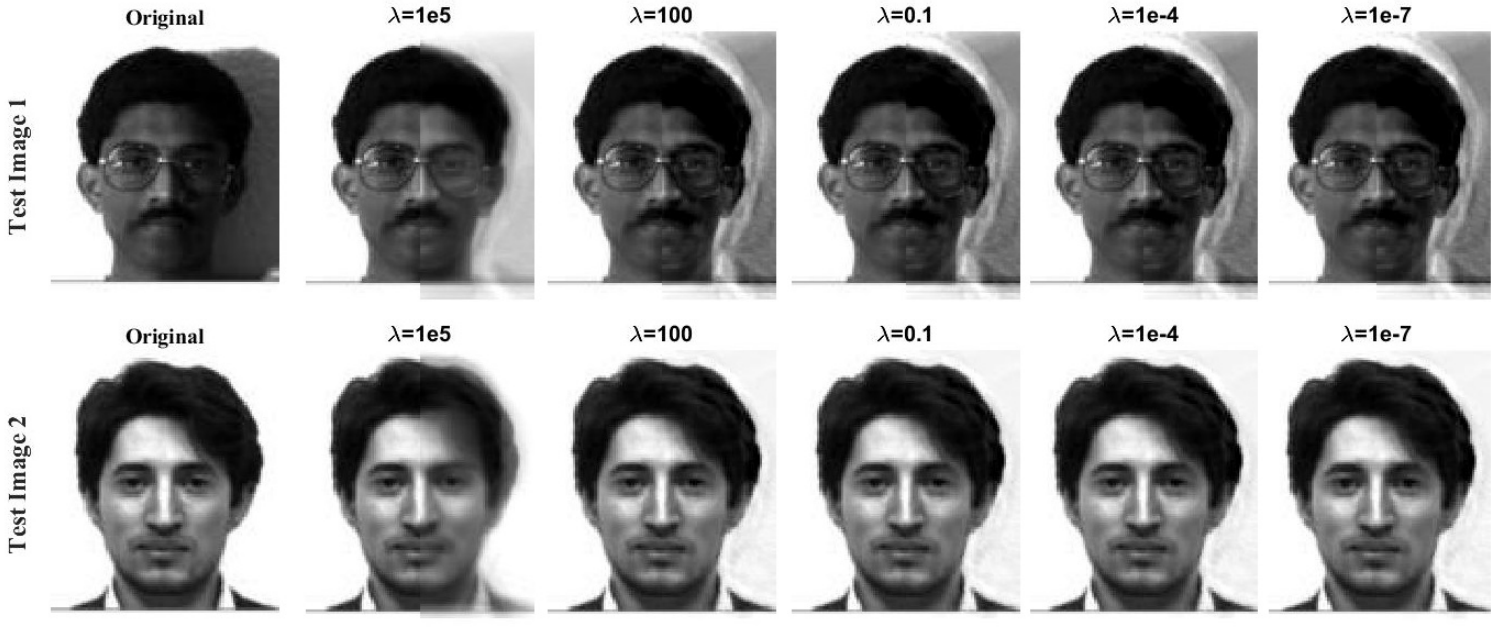

Figure 5: Right face prediction result on small-size image dataset using ridge regression with different λ

There's also an alternative method here. The Levenberg-Marquardt method is used to optimize the cost function with ridge regularization. The initial guess of θ is still set according to the symmetric assumption of left and right face. Then Eq. 5 is calculated and θ is updated iteratively. The regression result is shown in the fifth column of Fig.6, which is even better than single ridge regression's result thanks to good initial guess.

An overall comparison of four regression methods are shown in Fig.6. Among them, the iterative ridge regression and single ridge regression are better solutions. The consuming time is also listed in Table 1. It shown that classic linear regression is the fast among these methods. Though iterative ridge regression may have better result, its consuming time is much longer than single ridge regression.

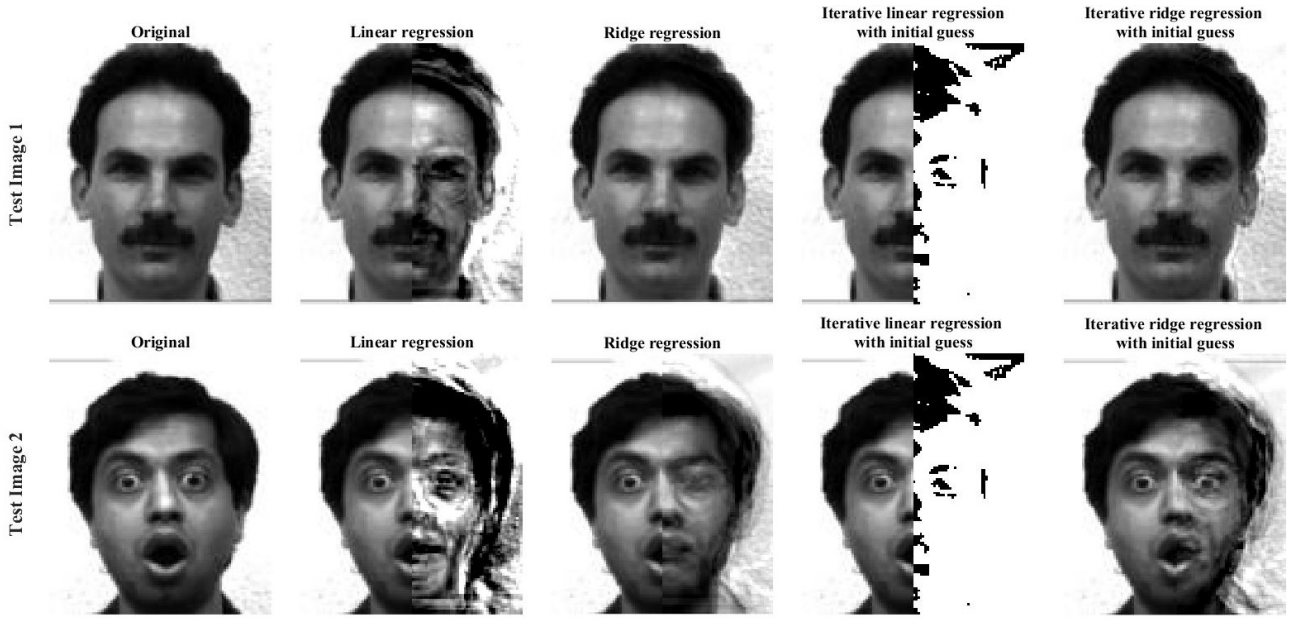

Figure 6: Comparison of different regression methods for right face prediction

Table 1: Training consuming time on small-size image dataset for different regression methods

Method	Linear Regression	Ridge Regression	Iterative Linear Regression (Gauss-Newton)	Iterative Ridge Regression (Levenberg-Marquardt)
Running time (s)	0.857	6.550	60.537 (10 iterations)	47.544 (10 iterations)

Exercise 5

Topic: Right face prediction using gradient descent. For this exercise, please refer to *ii_predict_face_reg_big.m*.

For this task, a dataset with larger images would be processed, which means we may meet some problem on memory and consuming time. The specification of two different datasets are listed in Table 2, from which we can find that the large image is about 4 times larger than the small image. The normal matrix's size can be calculated as Eq.6, which means we need to inverse a matrix whose elements' number is about 5×10^8.

$$\mathrm{X^T X \text{ size}} = (231 \times 98 + 1) \times (231 \times 98 + 1) = 5.12524 \times 10^8 \tag{6}$$

Firstly, the experiment platform is a PC with a 16 GB RAM and an Intel Core i7-7700HQ 2.80GHz CPU.

Since the inversion of matrix has a time complexity of $O(n^3)$, and the matrix size of large dataset is about 4 times of small dataset's, theoretically, the consuming time on large dataset should be about $4^3 = 64$ times of small dataset's. So we can expect that the consuming time should be $64 \times 6.5 = 416$ seconds, in which 6.5 is the ridge regression time consumed on small dataset reported in Table 1. Actually, this is not unacceptable.

According to the final experiment result, the consuming time should be even shorter, as shown in Table 3. As shown in Fig.7, the ridge regression result on large dataset is quite well, which proves that the experiment platform (My PC) is powerful enough for this task.

However, to avoid the inversion of extremely large matrix, a numerical optimization using gradient descent is also adopted. Unfortunately, the learning rate is very hard to determine in this case and I finally use a adaptive learning rate that would guarantee the amount of change in one iteration would not exceed a tiny value (for

Table 2: Specification and matrix size for small and large datasets

Dataset	#Train(Test) Sample	Image size	X size	*theta* size
Small Image	132 (33)	115×98	$132 \times (115 \times 49 + 1)$	$(115 \times 49 + 1) \times (115 \times 49)$
Large Image	132 (33)	231×195	$132 \times (231 \times 98 + 1)$	$(231 \times 98 + 1) \times (231 \times 97)$

example 1^{-6}. To do so, a good enough initial guess should be provided. Similar to tasks before, I used the initial guess of θ based on face symmetric assumption. After 20 iterations, the model's prediction result is shown as Fig.8, which is not quite well compared to ridge regression's. What's more, this method take too much time for training. For each iteration, all the training samples are taken to calculate the gradient, thus being very time-consuming. For the detailed implement of gradient descent, please refer to the code *ii_predict_face_reg_big.m* and its comments.

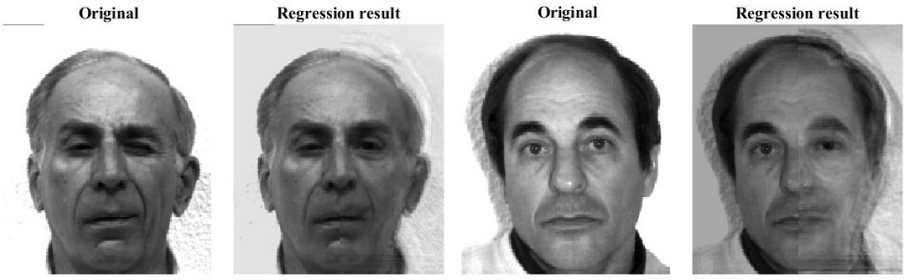

Figure 7: Right face prediction result on large-size image dataset using ridge regression

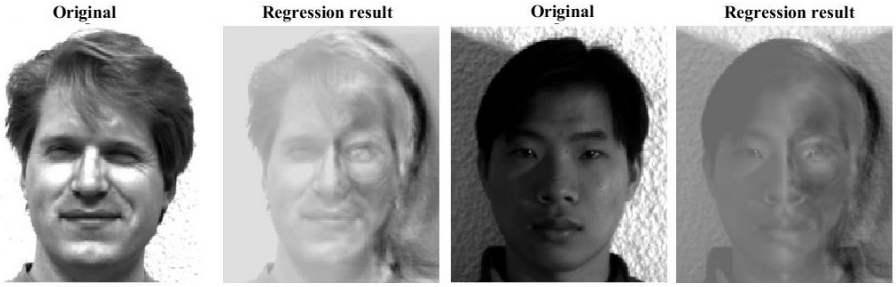

Figure 8: Right face prediction result on large-size image dataset using gradient descent with symmetric initial guess (20 iterations)

Table 3: Training consuming time on large-size image dataset for different methods

Method	Ridge Regression	Gradient Descent
Running time (s)	282.613	874.262 (20 iterations)