Report

1. Optimization Strategy

Gradient descent is used for convergence of objective function to local minimum by following negative gradient direction of the objective function at current point. A parameter α_k is needed to determine the step size taken along the direction at every iteration k to reach a (local) minimum. In other words, the aim of gradient descent is to find minimum point by following gradient directions over the surface created by the objective function. The algorithm description is defined as

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{1}$$

The step size $\alpha_k > 0$ is an important parameter to control convergence rate. On one hand, too small values of α_k will cause the algorithm to converge very slowly. On the other hand, too large values of α_k will cause the algorithm to overshoot the minima and diverge. Therefore, the step size should be chosen appropriately in order to get decreased objective function value for every iteration.

In this project, the loss function that is least square form has been minimized according to given training data. The custom activation function called sigmoid function has been calculated according to given condition. In order to minimize our loss function, the partial derivatives of loss function is calculated both with respect to b and w.

db = -(2/N) * np.sum((y -A)) (2)

dw = -(2/N) * np.dot(x,(y -A)) (3)

where A denotes activation function and y represents training labels.

In each iteration, activation function is updated and loss function minimization performance depends on activation function classification. Cost function continues to decrease until a tolerance value is reached. The optimization results of training data are shown in Figure 1, 2, 3 and Table 1, 2, 3 respectively.

Table 1 Selected learning rate and number of iterations for linear data by using Gradient Descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.0005	0.0001	2541	96.0

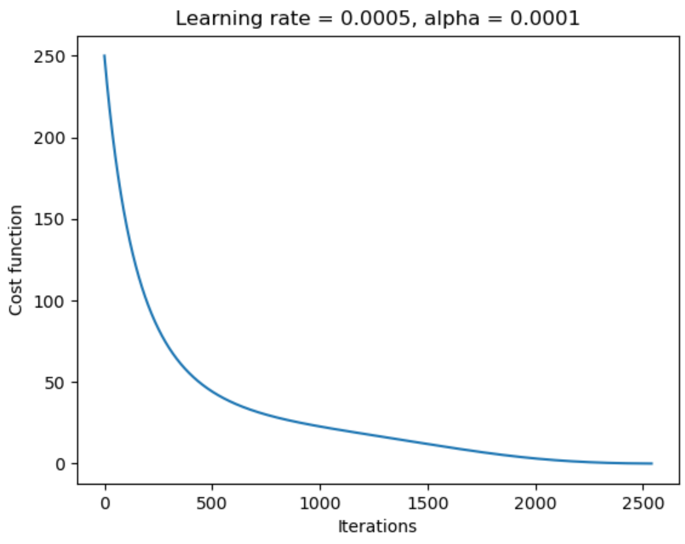
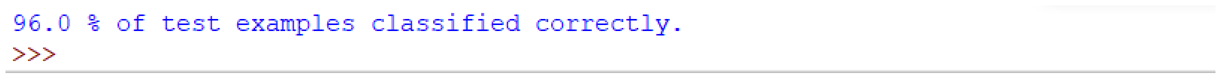


Figure 1 Cost function vs. iteration count for linear training data by using Gradient Descent method



```
96.0 % of test examples classified correctly.
>>>
```

Table 2 Selected learning rate and number of iterations for mnist_01 data by using Gradient descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.005	0.001	7674	99.57

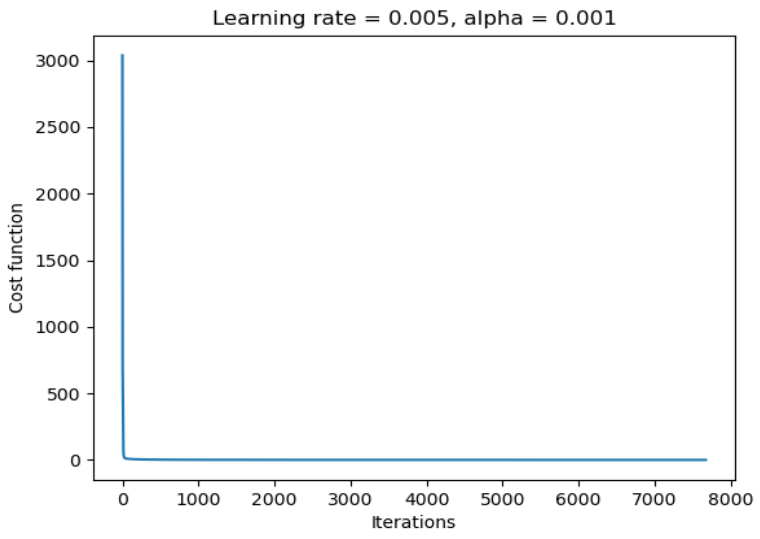
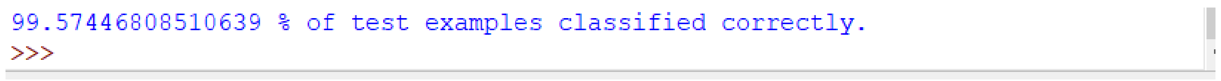


Figure 2 Cost function vs. iteration count for mnist_01 training data by using Gradient Descent method



```
99.57446808510639 % of test examples classified correctly.
>>>
```

Table 3 Selected learning rate and number of iterations for moon data by using Gradient Descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.005	0.001	1645	84.2

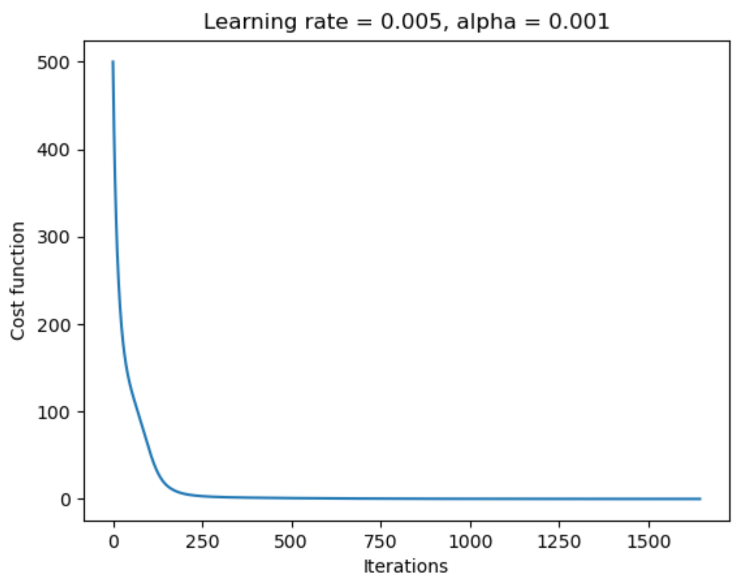


Figure 3 Cost function vs. iteration count for moon training data by using Gradient Descent method

```
84.2 % of test examples classified correctly.
>>>
```

2. Improvements

Optimizing of training data can be time consuming by using gradient descent. Therefore, Nesterov`s Accelerated Gradient descent achieves a faster convergence rate under the same assumption as gradient descent. The update of each Nesterov iteration mainly consists of the following two steps:

$$y_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{4}$$

$$x_{k+1} = (1-\gamma_k) y_{k+1} + \gamma_k y_k. \tag{5}$$

It firstly makes a step in the direction of previously accumulated gradient and then 'slides' a little bit for correction. Note that, Nesterov's accelerated gradient saves the previous gradient which is controlled by momentum term γ_k. Thus, the direction is determined by the current gradient and the previous gradient together. Similar to momentum, how many previous gradients is reserved is controlled by momentum term γ_k, This additional Nesterov's momentum ensures acceleration of Nesterov's method with respect to gradient descent. The Nesterov optimization results of traing data are shown in Figure 4, 5, 6 and Table 4, 5, 6 respectively.

Table 4 Selected learning rate and number of iterations for linear data by using Nesterov Accelerated Gradient Descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.0005	0.0001	120	97.5

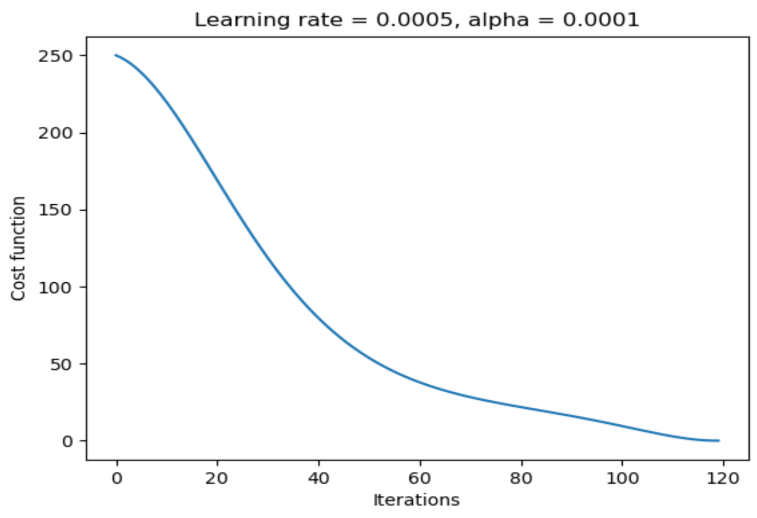


Figure 4 Cost function vs. iteration count for linear training data by using Nesterov Accelerated Gradient Descent method

```
97.5 % of test examples classified correctly.
>>>
```

Table 5 Selected learning rate and number of iterations for mnist_01 data by using Nesterov Accelerated Gradient descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.005	0.001	332	99.669

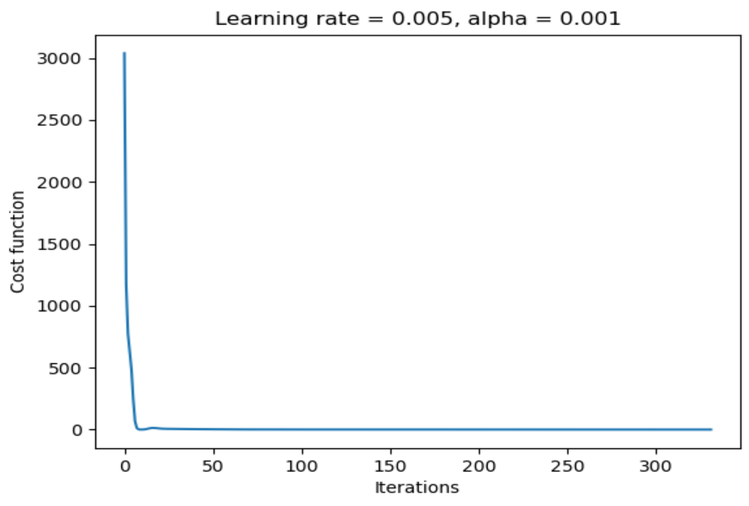


Figure 5 Cost function vs. iteration count for mnist_01 training data by using Nesterov Accelerated Gradient Descent method

```
99.66903073286052 % of test examples classified correctly.
>>>
```

Table 6 Selected learning rate and number of iterations for moon data by using Nesterov Accelerated Gradient Descent

Learning Rate	Alpha value	Iteration number	Test Accuracy (%)
0.005	0.001	108	83.6

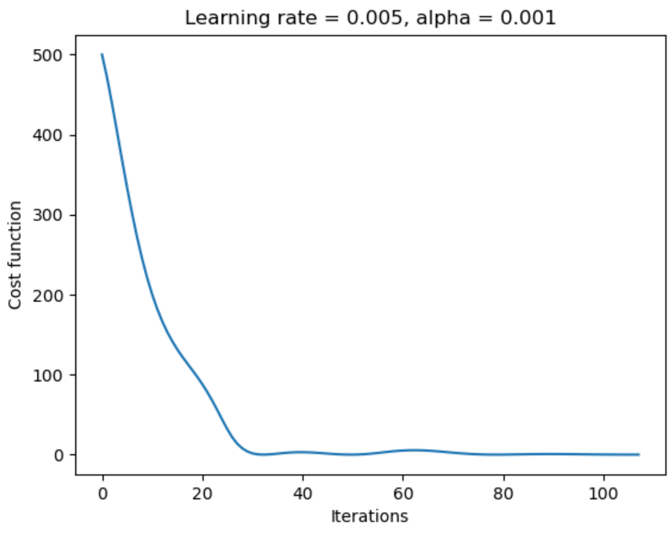


Figure 6 Cost function vs. iteration count for moon training data by using Nesterov Accelerated Gradient Descent method

```
83.6 % of test examples classified correctly.
>>>
```

3. Result and Discussion

As a general rule, the use of appropriate step size has a major impact in convergence rate. Results show that Nesterov accelerated gradient descent is better method than gradient descent method for the three different training data regarding lower iteration numbers. In order to increase optimization performance, Line search methods (backtracking, strong Wolfe) that identify a proper step size that is not too large but at the same time achieves sufficient reduction of the objective function can be applied. Moreover, the activation function can be modified by adding more conditions. It would be interesting to optimize with Levenberg-Marquardt algorithm.