Problem 1

(a).

Logistic regression is used to classification base on MLE and sigmoid function.
For binary classification task, we have training data $S = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$
with sigmoid function, we can represent a postererior probility for (x_i, y_i):

$$f_\theta(x) = Pr[y_i \mid x_i, \theta] = \frac{1}{1+\exp(-y_i \cdot \theta^T x_i)}$$, which is the likelihood of y_i given x_i.

Because of the i.i.d of S, the log-likelihood of all y_i:

$$\sum_{i=1}^{n} \log(Pr[y_i \mid x_i, \theta]) = -\sum_{i=1}^{n} \log(1+\exp(-y_i \cdot \theta^T x_i))$$

MLE leads to logistic regression:

$$\hat{\theta} = \underset{\theta \in R^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \log(1+\exp(-y_i \cdot \theta^T \cdot x_i))$$

(b) Because the local minimum of convex problem is the global opitimal solution, while finding a a stationary point for non-convex optimization problem is NP-hard.

(c). No. Logistic regreesion is actually used for classification instead of regression. The output of Logistic regreesion is discrete.

(d). Because Nesterov's Acceleration adds a Nesterov's momentum: $w_k = \theta_k + \frac{k-1}{k+1}(\theta_k - \theta_{k-1})$, which leads to a larger step towards next θ.

(e) The generalized learning problem is:

$$\theta_{k+1} = \underset{\theta \in R^d}{\arg\min} \{ g_k(\theta) + \frac{1}{2}\lambda \|\theta - \theta_k\|_2^2 \qquad (1)$$

For proximal gradient descent method for lasso:

$$\theta_{k+1} = \underset{\theta \in R^d}{\arg\min} \{ g_k(\theta) + \nabla g_k(\theta_k)^T(\theta - \theta_k) + \lambda\|\theta\|_1 + \frac{1}{2}\lambda\|\theta - \theta_k\|_2^2 \} \qquad (2)$$

rewrite (2):

$$\theta_{k+1} = \underset{\theta \in R^d}{\arg\min} \{ \frac{1}{2u_k} \|\theta - \alpha\|_2^2 + \lambda\|\theta\|_1 \} \qquad (3)$$

where $\alpha = (\theta_k - u_k \nabla g(\theta_k))^T$

$\xRightarrow{\text{Decomposable}}$ $\theta_{k+1}[i] = \underset{\theta_i \in \mathbb{R}}{\arg\min}\ f(\theta_i) = \frac{1}{2u_k}(\theta_i - a_i)^2 + \lambda|\theta_i|$.

$$\frac{\partial f(\theta_i)}{\theta_i} = \frac{\theta_i}{u_k} + \lambda = 0, \quad \theta_i = -\lambda u_k, \text{ if } \theta_i > 0$$

$$\theta_i = \lambda u_k, \text{ if } \theta_i < 0$$

so
$$\hat{\theta}_{k+1} = \begin{cases} a_i - \lambda u_k, & \text{for } a_i \geq \lambda u_i \\ 0, & \text{for } a_i \in (-\lambda u_k, \lambda u_k) \\ a_i + \lambda u_k, & \text{for } a_i \leq -\lambda u_k \end{cases}$$

(f) SVM will select two parallel hyperplanes that linearly sperate the two classes and the distance between them is as large as possible. So we should $\underset{\theta \in \mathbb{R}^d}{\text{maximize}} \frac{2}{\|\theta\|_2}$, which is to $\underset{\theta \in \mathbb{R}^d, b \in \mathbb{R}}{\text{minimize}} \|\theta\|_2^2$. Finally, the learning problem is:

$$\underset{\theta \in \mathbb{R}^d,\, b \in \mathbb{R}}{\arg\min} \|\theta\|_2^2$$
$$\text{s.t } y_i(\theta_i^T x + b) \geq 1, \ \forall i \in \mathbb{R}$$

(g) Kernel Method is used to aviod the previous computational issue like calculating nolinear transform $\Phi(x)$ and $\Phi(x')$ for lifting data to higher dimension. We only need to find a function k satisfies $k(x, x') = \Phi(x)^T \Phi(x')$, $\forall x, x'$.

Problem 2.

1a). Firstly We difine proximal mapping as a function of r and t as follows:

$$\text{prox}_{t,r}(x) = \underset{\theta \in \mathbb{R}^{d \times 1}}{\text{algmax}} \frac{1}{2t}\|\theta - x\|_2^2 + r(\theta)$$

- For Proximal Gradient Descent can be defined as follows:
with initinalization X_0 then repeat:

$$\theta_{k+1} = \text{prox}_{t_k}(\theta_k - t_k \nabla g(\theta_k)), \ k = 1, 2, 3 \ldots$$

This can be further expressed as:

$$\theta_{k+1} = \theta_k - t_k G_{t_k}(\theta_k)$$

where G_t is a generalized gradient of f.

$$G_t(x) = \frac{x - \text{prox}_t(x - t\nabla g(x))}{t}$$

- For Proximal Stochastic Gradient Descent with Momentum =

As before, the problem is : $\min g(\theta) + \gamma(\theta)$ with g convex.

Firstly choose initial point $\theta_0 \in R^n$ and repeat $(K = 1, 2, 3 \ldots)$ =

$$V = \theta_K + \frac{K-2}{K+1}(\theta_K - \theta_{K-1})$$

$$\theta_K = \mathrm{prox}_{t, K}(V - t_K \nabla g(\theta_i)) \quad , \text{ for } \forall i \text{ from } [1, n].$$

- For Proximal Accelerated Gradient Descent with Momentum =

As before, the problem is : $\min g(\theta) + \gamma(\theta)$ with g convex.

Firstly choose initial point $\theta_0 \in R^n$ and repeat $(K = 1, 2, 3 \ldots)$ =

$$V = \theta_K + \frac{K-2}{K+1}(\theta_K - \theta_{K-1})$$

$$\theta_K = \mathrm{prox}_{t, K}(V - t_K \nabla g(V))$$

1b).

B） Training loss, test loss and test accuracy of 4 methods of optimization can be seen from the figures below:

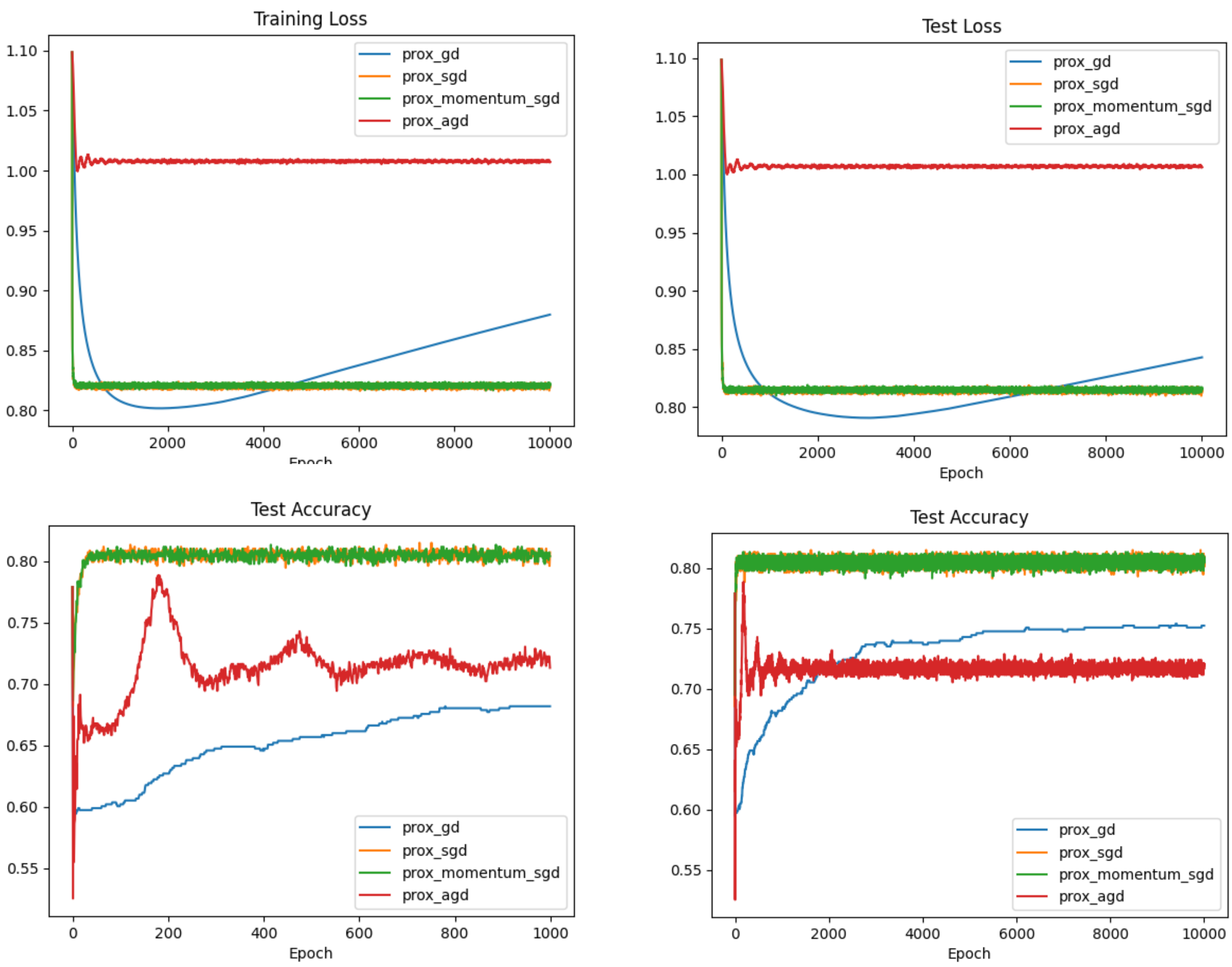


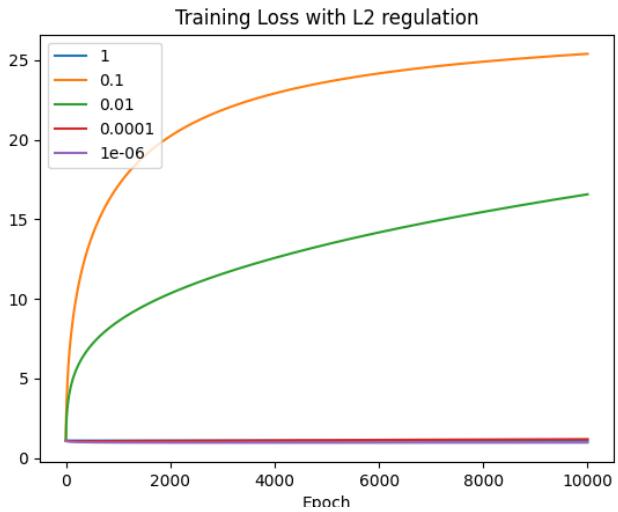
Figure 1

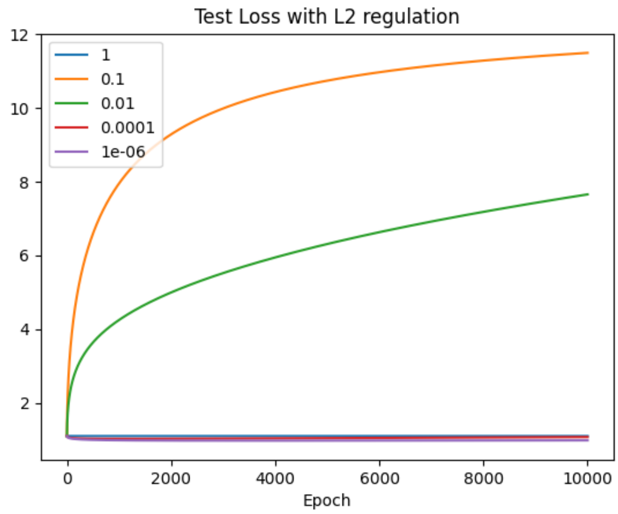
For these 4 algorithms, we can see that PSGD and PSGD with momentum have highest accuracy and fastest convergence speed, and random shuffle is used here to speed up the convergence and get superior performance.

PGD has the worst performance with low test accuracy and slow convergence speed, while AGD performs in the middle.

Besides, the slight bouncing is observed in PSGD , PSGD with momentum, and AGD especially in the early stage(the bottom left figure).

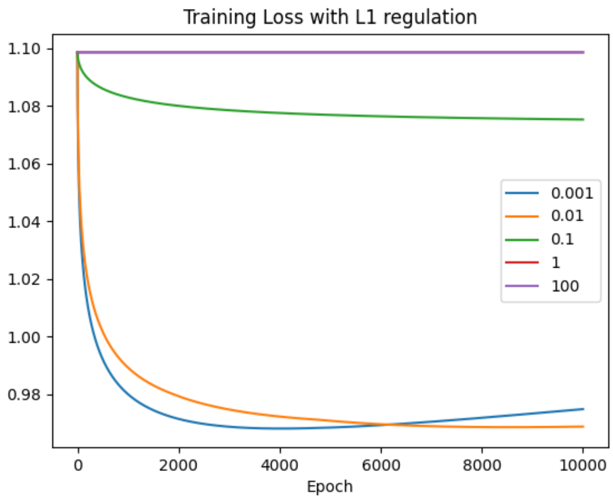
For proximal gradient descent with L2-regularization, the training loss, test loss, and test accuracy of different lambda are as shown below:

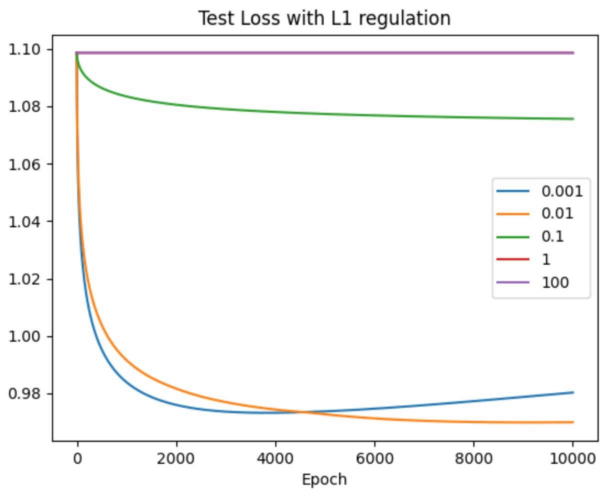


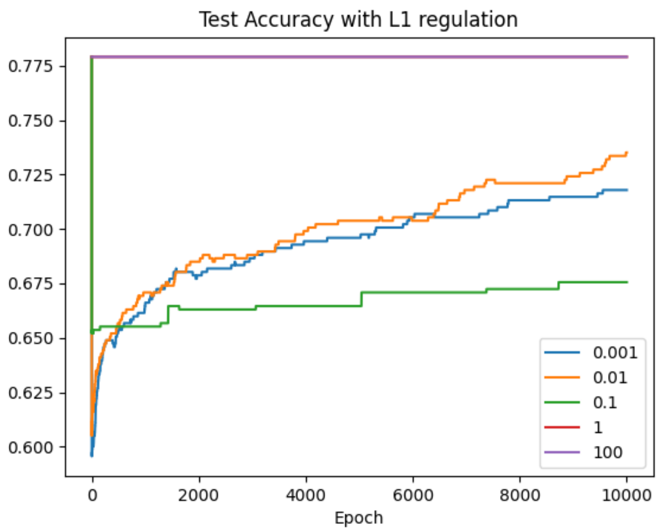


For PGD with l2-norm with different λ, it has been shown the larger the λ, the greater the performance. However, when the λ is too large, the performance will be worse because of under-fitting. PGD with L2- norm is very sensitive to penalization parameter, so we needs a moderate λ .

For proximal gradient descent with L1-regularization, the training loss, test loss, and test accuracy of different lambda are as shown below:



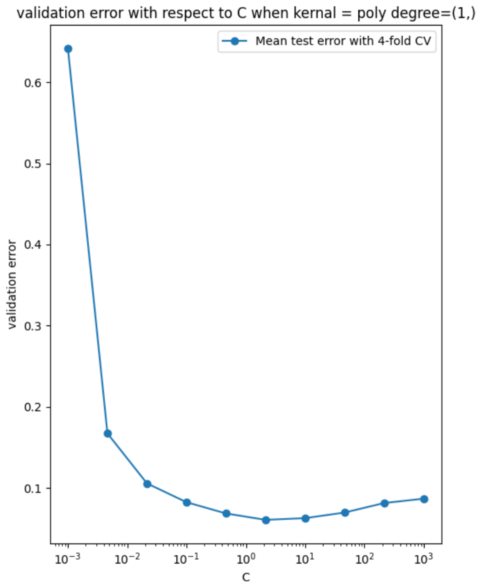


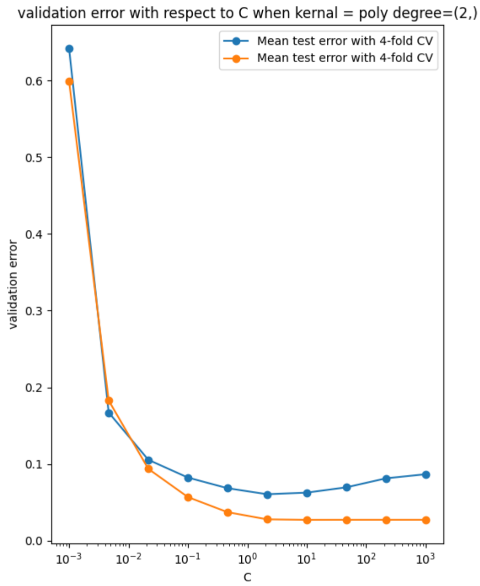


For PGD with l1-norm with different λ, the outcome is different. When lambda becomes too large like 100, the loss function won't get decedent. The best lambda is between 0.001 to 0.01.

Problem 3

For model with degree of 1 (blue line in both figures below), best C is 2.15443469 and the error with best C is :0.05, while For model with degree of 2 (orange line in right figure below), best C is 10. and the error with best C is :0.00.





The CV curve of SVM using rbf kernel is shown as follows. Best C = *46.42* , best gamma = *0.0000* , and the error with best C and gamma is :*0.00*