Machine Learning Homework 2 Dai Xunlian 225040015

Dai Xunlian@link.cuhk.edu.cn

October 2025

1 Q1

1.1 Q1a

Answer: d+1

1.2 Q1b

Test error is the empirical error measured on a specific test set, while out-of-sample error is the true expected error over all possible unseen data.

1.3 Q1c

False
For in sample error, if we make it as small as possible, we correspondingly need higher complexity of hypothesis space. However it would make the differential between in sample error and out sample error larger.

1.4 Q1d

1) True

2) False. Because the log likelihood of LR is not quadratic in the parameters.

3) True

4) False. It could be applied to multi-classification

2 Q2

2.1 Q2.1

For a single sample, the objective function is:

$$f(\Theta, b) = -y^\top \log \sigma(\Theta x + b) \tag{1}$$

where $\sigma(z) = \frac{\exp(z)}{1^\top \exp(z)}$ is the softmax function, $x \in \mathbb{R}^d$ is the input sample, $y \in \mathbb{R}^K$ is the one-hot encoded label vector, $\Theta \in \mathbb{R}^{K \times d}$ is the weight matrix, and $b \in \mathbb{R}^K$ is the bias vector.

Let $\mathbf{1}$ denote the all-one vector. We have:

$$f = -y^\top \left(\log \exp(\Theta x + b) - \mathbf{1} \log(1^\top \exp(\Theta x + b)) \right) \tag{2}$$

$$= -y^\top (\Theta x + b) + \log(1^\top \exp(\Theta x + b)) \tag{3}$$

Let $z = \Theta x + b$. Differentiating f with respect to Θ:

$$df = -y^\top (d\Theta) x + \frac{\exp(z)^\top (d\Theta) x}{1^\top \exp(z)} \tag{4}$$

Using the property $1^\top (u \odot v) = u^\top v$:

$$df = -y^\top (d\Theta) x + \frac{\exp(z)^\top (d\Theta) x}{1^\top \exp(z)} \tag{5}$$

Taking the trace and rearranging using the exchange property:

$$\mathrm{Tr}(df) = \mathrm{Tr}\left[\left(-y + \frac{\exp(z)}{1^\top \exp(z)} \right) x^\top d\Theta \right] \tag{6}$$

Hence, the gradient is:

$$\boxed{\frac{\partial f}{\partial \Theta} = \left(-y + \sigma(\Theta x + b) \right) x^\top} \tag{7}$$

Differentiating f with respect to b:

$$df = -y^\top db + \frac{\exp(z)^\top db}{1^\top \exp(z)} \tag{8}$$

This directly gives:

$$\boxed{\frac{\partial f}{\partial b} = -y + \sigma(\Theta x + b)} \tag{9}$$

For n samples, the loss function is:

$$L(\Theta, b) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\ell=1}^{K} \mathbf{1}\{y_i = \ell\} \log \sigma_\ell(\Theta x_i + b) \tag{10}$$

The gradients are:

$$\boxed{\frac{\partial L}{\partial \Theta} = \frac{1}{n} \sum_{i=1}^{n} \left(\sigma(\Theta x_i + b) - y_i\right) x_i^\top} \tag{11}$$

$$\boxed{\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} \left(\sigma(\Theta x_i + b) - y_i\right)} \tag{12}$$

2.2 Q2.2

2.2.1 Results

The loss curve and accuracy of COFFEE and WEATHER datasets are shown in Figure 1 and 2

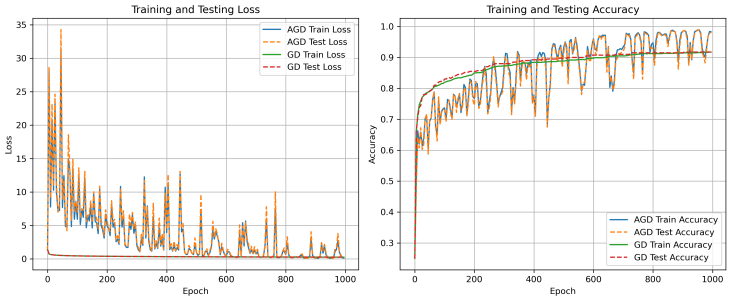

Figure 1: Training and testing loss/accuracy comparison between AGD and GD of Coffee dataset

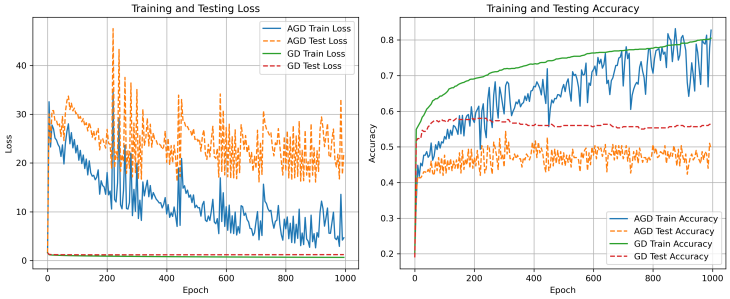

Figure 2: Training and testing loss/accuracy comparison between AGD and GD of Weather dataset

2.2.2 Analysis

Based on the training results from both datasets, here are my observations:

Coffee Dataset **Convergence Speed:** GD demonstrates significantly faster and more stable convergence compared to AGD. GD's loss curves quickly drop to near-zero and remain stable, while AGD exhibits severe oscillations throughout training with loss values fluctuating dramatically between 0 and over 30.

Accuracy: GD achieves superior and stable performance with both training and testing accuracy reaching approximately 90-95% and maintaining consistency. AGD shows highly unstable accuracy with extreme fluctuations, ranging from 70% to nearly 100

Overfitting: GD shows minimal overfitting with training and testing accuracies closely aligned. AGD's erratic behavior makes overfitting assessment difficult, though the similar ranges of train/test accuracy suggest the primary issue is instability rather than overfitting.

Weather Dataset **Convergence Speed:** GD converges smoothly and steadily, with training loss decreasing monotonically. AGD again shows unstable convergence with persistent oscillations in loss values, though less severe than in the coffee dataset.

Accuracy: GD achieves stable training accuracy around 75-80%, while testing accuracy plateaus at approximately 55-56%. AGD shows highly volatile training accuracy (ranging 40-80%) with testing accuracy remaining flat around 45-50%, demonstrating poor generalization.

Overfitting: GD exhibits moderate overfitting with a noticeable gap (20%) between training and testing accuracy. AGD shows severe overfitting - despite volatile training accuracy improvements, testing accuracy remains consistently low and stagnant, indicating poor generalization capability.

3 Q3

The experiment result of Subgraindient method convergence is shown in Figure 3

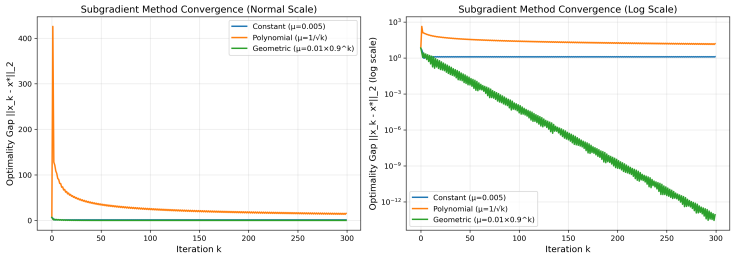

Figure 3: Subgraindient method convergence result

Based on our experimental settings, the three learning rate strategies exhibit distinct convergence behaviors. The geometric diminishing rate ($\mu = 0.01 \times 0.9^k$) achieves the best performance, converging monotonically to an optimality gap

of approximately 10^{-14} by iteration 300, effectively balancing aggressive initial progress with sufficient diminishing for convergence. The constant learning rate ($\mu = 0.005$) converges rapidly initially but plateaus around gap of 1 due to fixed step sizes preventing fine-grained convergence. The polynomial rate ($\mu = 1/\sqrt{k}$) exhibits the slowest convergence, reaching only gap of 200, as the learning rate diminishes too quickly for substantial later progress.

Compared to gradient descent on smooth functions, subgradient methods demonstrate fundamentally different characteristics. While gradient descent achieves linear convergence, subgradient methods are limited to sublinear convergence rates even optimally. The observed oscillatory behavior, especially with constant learning rates, is more pronounced due to variability in subgradient selection at non-differentiable points. This heightened sensitivity to learning rate schedules confirms that carefully designed diminishing strategies are essential for practical convergence in non-smooth optimization.

4 Q4

4.1 Q4.1

For a model of degree K, construct the design matrix $X \in \mathbb{R}^{n\times(K+1)}$ where $X_{ij} = L_j(x_i)$ for $i = 1, \ldots, n$ and $j = 0, \ldots, K$. Let $y = [y_1, \ldots, y_n]^\top$ be the label vector and $w = [w_0, \ldots, w_K]^\top$ be the parameter vector. The least squares solution is:

$$\hat{w} = (X^\top X)^{-1} X^\top y \tag{13}$$

For f_2, set $K = 2$; for f_{10}, set $K = 10$.

4.2 Q4.2

By definition,

$$\mathrm{Er}_{\mathrm{out}}(f_K) = \mathbb{E}_x[(f_K(x) - g(x))^2] = \int_{-1}^{1} \left(\sum_{k=0}^{K} w_k L_k(x) - \frac{1}{C_Q} \sum_{q=0}^{Q_g} a_q L_q(x) \right)^2 dx \tag{14}$$

Expanding the square and utilizing the orthogonality property $\int_{-1}^{1} L_i(x)L_j(x)dx = \frac{2}{2i+1}\delta_{ij}$:

$$\mathrm{Er}_{\mathrm{out}}(f_K) = \sum_{k=0}^{K} w_k^2 \cdot \frac{2}{2k+1} - 2 \sum_{k=0}^{\min(K,Q_g)} \frac{w_k a_k}{C_Q} \cdot \frac{2}{2k+1} + 1 \tag{15}$$

This can be simplified as:

$$\boxed{\mathrm{Er}_{\mathrm{out}}(f_K) = \sum_{k=0}^{K} \frac{2w_k^2}{2k+1} - \frac{4}{C_Q} \sum_{k=0}^{\min(K,Q_g)} \frac{w_k a_k}{2k+1} + 1} \tag{16}$$

4.3 Q4.3

The result of Q_g versus n and σ versus n are shown in Figure 4 and 5

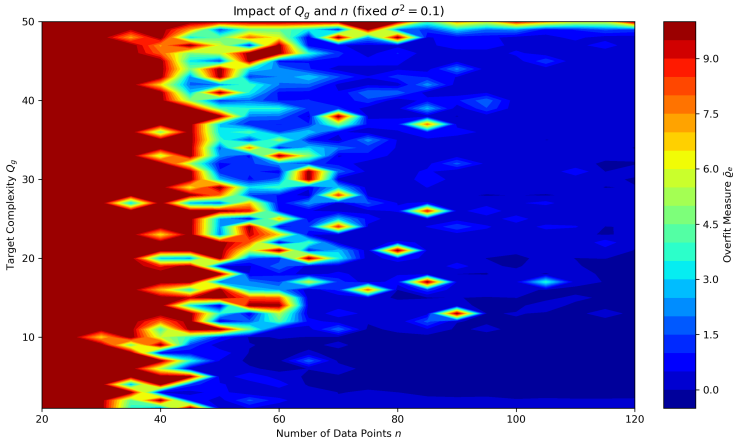

Figure 4: Subgraindient method convergence result

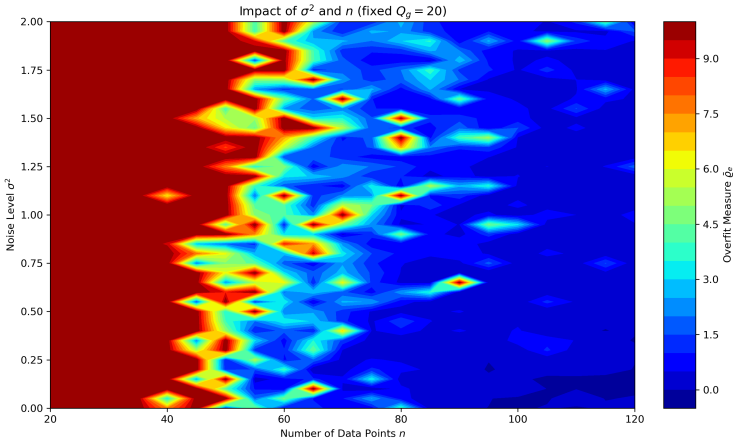

Figure 5: Subgraindient method convergence result

Experiment 1: Impact of Q_g and n (fixed $\sigma^2 = 0.1$)

Overfitting is most severe (red regions) when $Q_g < 15$ and $n < 60$, as f_{10} becomes excessively complex relative to the simple target. As n increases beyond 60, overfitting diminishes across all Q_g values. When $Q_g > 10$, the measure becomes negative (blue), indicating f_{10} underperforms f_2 as neither model captures the high-complexity target adequately.

Experiment 2: Impact of σ^2 and n (fixed $Q_g = 20$)

Higher noise ($\sigma^2 > 1.0$) with limited data ($n < 50$) produces severe overfitting, as f_{10} fits noise rather than signal. Increasing n substantially reduces overfitting across all noise levels. Even at low noise, small n causes moderate overfitting due to model-target complexity mismatch.

Summary: Causing Factors of Overfitting

Three factors drive overfitting: (1) Insufficient data—small n fails to constrain complex models; (2) Model-target mismatch—f_{10} overfits when Q_g is low, underfits when $Q_g > 10$; (3) High noise—large σ^2 amplifies spurious pattern fitting. Overfitting peaks when all three align: low n, low Q_g, high σ^2.