In the name of GOD.



Sharif University of Technology

Machine Learning

Spring 2025

Hamid R. Rabiee, Zahra Dehghanian

Homework 2 — Deadline: 1403/12/26

1. [20] **Consider the following two distributions and answer the question:**

   Distribution 1:

   A scientist is studying the decay of a radioactive material. The measured radiation levels $Y_1, Y_2, \ldots, Y_N$ over time are assumed to be independent and normally distributed:

   $$Y_i \sim N(\mu_i, \sigma^2)$$

   where the mean follows an exponential decay model:

   $$\mu_i = \theta e^{-t_i}$$

   Here, t_i represents time, and the decay process suggests that as time increases, the expected radiation level decreases exponentially. The values of Y_i are assumed to be independent given the parameter θ.

   Here, θ is the unknown parameter, and σ^2 is a known constant variance.

   Distribution 2:

   Suppose we want to model a 1-dimensional dataset of N real-valued features ($x^{(i)}$) and target values ($y^{(i)}$) using the following exponential transformation model:

   $$y^{(i)} \sim N(\exp(wx^{(i)}), 1)$$

   where w is the unknown scalar parameter.

   The data points $y^{(i)}$ are assumed to be independent given the parameter w.

   Can the maximum conditional negative log-likelihood estimator of the unknown parameter (θ in Distribution 1, w in Distribution 2) be solved analytically?

   - If so, find the expression for the MLE of the parameter.
   - If not, explain why and provide the gradient update rule for the parameter using gradient descent.

   Solution:

Distribution 1:

Negative Log-Likelihood (NLL). Given N observations $\{Y_i\}_{i=1}^N$, each distributed as

$$Y_i \sim N(\theta e^{-t_i},\, \sigma^2),$$

the full negative log-likelihood is

$$\mathrm{NLL}(\theta) \;=\; \sum_{i=1}^{N}\Big[\tfrac{1}{2}\,\log\big(2\pi\sigma^2\big) \;+\; \tfrac{\big(Y_i-\theta e^{-t_i}\big)^2}{2\,\sigma^2}\Big] \;=\; \tfrac{N}{2}\,\log\big(2\pi\sigma^2\big) \;+\; \frac{1}{2\,\sigma^2}\sum_{i=1}^{N}\big(Y_i-\theta e^{-t_i}\big)^2.$$

The constant term $\frac{N}{2}\log(2\pi\sigma^2)$ does not depend on θ, so it does not affect the estimate.

MLE derivation. To find the MLE, we set the derivative (w.r.t. θ) of the second term to zero:

$$\frac{\partial}{\partial\theta}\Big(\frac{1}{2\,\sigma^2}\sum_{i=1}^{N}\big(Y_i-\theta e^{-t_i}\big)^2\Big) \;=\; 0.$$

This simplifies to

$$\sum_{i=1}^{N}\big(Y_i-\theta e^{-t_i}\big)\big(-e^{-t_i}\big) \;=\; 0 \quad\Longrightarrow\quad \sum_{i=1}^{N} Y_i\,e^{-t_i} \;=\; \theta\sum_{i=1}^{N} e^{-2t_i}.$$

Hence, the closed-form MLE is

$$\boxed{\hat\theta \;=\; \frac{\sum_{i=1}^{N}\big(Y_i\,e^{-t_i}\big)}{\sum_{i=1}^{N} e^{-2t_i}}}.$$

Distribution 2:

Negative Log-Likelihood (NLL). For observations $\{(x^{(i)},y^{(i)})\}_{i=1}^N$, each

$$y^{(i)} \sim N\big(\exp(w\,x^{(i)}),\,1\big),$$

the full negative log-likelihood is

$$\mathrm{NLL}(w) \;=\; \sum_{i=1}^{N}\Big[\tfrac{1}{2}\,\log\big(2\pi\big) \;+\; \tfrac{\big(y^{(i)}-e^{wx^{(i)}}\big)^2}{2}\Big] \;=\; \frac{N}{2}\,\log(2\pi) \;+\; \sum_{i=1}^{N}\frac{\big(y^{(i)}-e^{wx^{(i)}}\big)^2}{2}.$$

Again, the constant $\frac{N}{2}\log(2\pi)$ does not depend on w.

No closed-form solution. If we differentiate the second term w.r.t. w and set to zero,

$$\sum_{i=1}^{N}\Big(y^{(i)}-e^{wx^{(i)}}\Big)\Big(-x^{(i)}e^{wx^{(i)}}\Big) \;=\; 0,$$

we obtain a transcendental equation in w with no simple closed-form root.

Gradient descent update. We thus resort to numerical methods such as gradient descent. The gradient (w.r.t. w) of the non-constant part of NLL(w) is

$$\frac{\partial}{\partial w} \sum_{i=1}^{N} \frac{\left(y^{(i)} - e^{wx^{(i)}}\right)^2}{2} \;=\; \sum_{i=1}^{N} \Big(y^{(i)} - e^{wx^{(i)}}\Big) \Big(-x^{(i)} e^{wx^{(i)}}\Big).$$

A gradient descent update step with learning rate η is:

$$w \;\leftarrow\; w - \eta \sum_{i=1}^{N} \Big(y^{(i)} - e^{wx^{(i)}}\Big) \left(-x^{(i)} e^{wx^{(i)}}\right) \;=\; w + \eta \sum_{i=1}^{N} \Big(y^{(i)} - e^{wx^{(i)}}\Big) \left(x^{(i)} e^{wx^{(i)}}\right).$$

2. [15] Consider the linear regression model:

$$Y_i = \alpha + \beta X_i + \epsilon_i.$$

(a) Suppose the errors ϵ_i are independent but **not identically distributed** normal variables where the variance σ_i^2 depends on X_i as follows:

$$\epsilon_i \sim N(0, \sigma_i^2), \quad \sigma_i^2 = \sigma^2 X_i^2,$$

where σ^2 is a constant.

i. Derive the estimators $\hat{\alpha}$ and $\hat{\beta}$ for α and β, respectively, that minimize the weighted sum of squared residuals:

$$S(\alpha, \beta) = \sum_{i=1}^{n} \frac{(Y_i - \alpha - \beta X_i)^2}{\sigma_i^2}.$$

ii. Discuss the differences between the derived estimators and those obtained using ordinary least squares (OLS).

(b) Suppose the errors ϵ_i are i.i.d. exponential random variables with a mean of 1. Given a dataset of 1,000 points, where the X_i values are uniformly distributed over the interval $[0, 100]$, determine the distribution of the least squares estimators $\hat{\alpha}$ and $\hat{\beta}$ for α and β fitted to Y on X. Provide a detailed explanation and compute the parameters of the distributions as precisely as possible.

Solution:

(a) i. **Weighted Least Squares Estimators**
Given the model:

$$Y_i = \alpha + \beta X_i + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma_i^2)$ and $\sigma_i^2 = \sigma^2 X_i^2$, we aim to find estimators for α and β that minimize the weighted sum of squared residuals:

$$S(\alpha, \beta) = \sum_{i=1}^{n} \frac{(Y_i - \alpha - \beta X_i)^2}{\sigma_i^2}.$$

Since $\sigma_i^2 = \sigma^2 X_i^2$, the weight for each observation is given by $w_i = \frac{1}{X_i^2}$. Thus, we minimize:

$$S(\alpha, \beta) = \sum_{i=1}^{n} \frac{(Y_i - \alpha - \beta X_i)^2}{\sigma^2 X_i^2}.$$

To find the estimators, we take partial derivatives:

$$\frac{\partial S}{\partial \alpha} = -\frac{2}{\sigma^2} \sum_{i=1}^{n} \frac{Y_i - \alpha - \beta X_i}{X_i^2} = 0,$$

$$\frac{\partial S}{\partial \beta} = -\frac{2}{\sigma^2} \sum_{i=1}^{n} \frac{X_i(Y_i - \alpha - \beta X_i)}{X_i^2} = 0.$$

Rearranging, we obtain the normal equations:

$$\sum_{i=1}^{n} \frac{Y_i}{X_i^2} = \alpha \sum_{i=1}^{n} \frac{1}{X_i^2} + \beta \sum_{i=1}^{n} \frac{X_i}{X_i^2},$$

$$\sum_{i=1}^{n} \frac{X_i Y_i}{X_i^2} = \alpha \sum_{i=1}^{n} \frac{X_i}{X_i^2} + \beta \sum_{i=1}^{n} \frac{X_i^2}{X_i^2}.$$

Solving for $\hat{\alpha}$ and $\hat{\beta}$, we obtain the weighted least squares (WLS) estimators:

$$\hat{\beta} = \frac{\sum_{i=1}^{n} \frac{Y_i X_i}{X_i^2} - \frac{(\sum_{i=1}^{n} \frac{Y_i}{X_i^2})(\sum_{i=1}^{n} \frac{X_i}{X_i^2})}{\sum_{i=1}^{n} \frac{1}{X_i^2}}}{\sum_{i=1}^{n} \frac{X_i^2}{X_i^2} - \frac{(\sum_{i=1}^{n} \frac{X_i}{X_i^2})^2}{\sum_{i=1}^{n} \frac{1}{X_i^2}}},$$

$$\hat{\alpha} = \frac{\sum_{i=1}^{n} \frac{Y_i}{X_i^2} - \hat{\beta} \sum_{i=1}^{n} \frac{X_i}{X_i^2}}{\sum_{i=1}^{n} \frac{1}{X_i^2}}.$$

Since WLS is a form of generalized least squares, these estimators are unbiased under the given assumptions.

ii. **Comparison with OLS:** OLS assumes homoscedasticity, whereas WLS accounts for heteroscedasticity by assigning weights. WLS generally produces more efficient estimators when errors have non-constant variance.

(b) **Distributionof OLS Estimators with Exponential Errors**

Given that $\epsilon_i \sim \text{Exp}(1)$ (mean 1, variance 1), the least squares estimators $\hat{\alpha}$ and $\hat{\beta}$ follow an approximate normal distribution due to the central limit theorem (CLT), provided n is large.

Since $E(\epsilon_i) = 1$, the expectation of Y_i is:

$$E(Y_i) = \alpha + \beta X_i + 1.$$

We have from OLS estimators:

$$\hat{\beta} = \frac{S_{xy}}{S_{xx}} = \sum_{i=1}^{n} \frac{(X_i - \bar{X})Y_i}{S_{xx}},$$

$$\hat{\alpha} = \bar{Y} - \hat{\beta}\bar{X} = \sum_{i=1}^{n} \Big(\frac{1}{n} - \frac{\bar{X}(X_i - \bar{X})}{S_{xx}}\Big) Y_i$$

It implies that $\hat{\alpha}$ is biased by 1,

$$E(\hat{\alpha}) = \sum_{i=1}^{n} \Big(\frac{1}{n} - \frac{\bar{X}(X_i - \bar{X})}{S_{xx}}\Big) E(Y_i) = \alpha + 1.$$

However, $\hat{\beta}$ remains an unbiased estimator of β, as it depends on the centered values of X_i and the error term has a mean that cancels out in the derivation.

The variances of the estimators are:

$$\mathrm{Var}(\hat{\beta}) = \frac{\sigma^2}{S_{xx}}, \quad \mathrm{Var}(\hat{\alpha}) = \sigma^2 \left(\frac{1}{n} + \frac{\bar{X}^2}{S_{xx}} \right).$$

Since $\sigma^2 = 1$ given the variance of the exponential errors, we get:

$$\mathrm{Var}(\hat{\beta}) = \frac{1}{S_{xx}}, \quad \mathrm{Var}(\hat{\alpha}) = \left(\frac{1}{n} + \frac{\bar{X}^2}{S_{xx}} \right).$$

By the CLT, for large n, the sums of independent random variables tend to be normally distributed. Thus, the least squares estimators $\hat{\alpha}$ and $\hat{\beta}$ are approximately normal:

$$\hat{\alpha} \sim N(\alpha + 1, \frac{1}{n} + \frac{\bar{X}^2}{S_{xx}}), \quad \hat{\beta} \sim N(\beta, \frac{1}{S_{xx}}).$$

3. [15] (Linear Classification) Suppose you are given the following dataset:

Example Number	X_1	X_2	Y
1	-1	2	-1
2	-2	-2	$+1$
3	1	-1	$+1$
4	-3	1	-1

Please perform the Batch Perceptron algorithm on this data. Assume you start with initial weights $\theta^T = [0, 0]$, bias $b = 0$, and that we pass all of our examples through in order of their example number.

(a) What would be the updated weight vector θ be after we pass example 1 through the Perceptron algorithm?

(b) What would be the updated bias b be after we pass example 1 through the Perceptron algorithm?

(c) What would be the updated weight vector θ be after we pass example 2 through the Perceptron algorithm?

(d) What would be the updated bias b be after we pass example 2 through the Perceptron algorithm?

(e) What would be the updated weight vector θ be after we pass example 3 through the Perceptron algorithm?

(f) What would be the updated bias b be after we pass example 3 through the Perceptron algorithm?

Solution:
ith initial weight vector:

$$\theta^T = [0, 0], \quad b = 0$$

The perceptron update rule is:

$$\theta \leftarrow \theta + yx$$

$$b \leftarrow b + y$$

where $x = (X_1, X_2)$ and y is the label.

Processing Example 1: $(-1, 2), y = -1$

$$\theta = [0, 0] + (-1) \times [-1, 2] = [1, -2]$$

$$b = 0 + (-1) = -1$$

Processing Example 2: $(-2, -2), y = +1$

We compute:

$$\hat{y} = \text{sign}(\theta \cdot x + b) = \text{sign}([1, -2] \cdot [-2, -2] + (-1))$$

$$= \text{sign}((1 \times -2) + (-2 \times -2) + (-1))$$

$$= \text{sign}(-2 + 4 - 1) = \text{sign}(1) = +1$$

Since $\hat{y} = y = +1$, no update is needed.

Processing Example 3: $(1, -1), y = +1$

$$\hat{y} = \text{sign}([1, -2] \cdot [1, -1] + (-1))$$

$$= \text{sign}((1 \times 1) + (-2 \times -1) + (-1))$$

$$= \text{sign}(1 + 2 - 1) = \text{sign}(2) = +1$$

Since $\hat{y} = y = +1$, no update is needed.

Processing Example 4: $(-3, 1), y = -1$

$$\hat{y} = \text{sign}([1, -2] \cdot [-3, 1] + (-1))$$

$$= \text{sign}((1 \times -3) + (-2 \times 1) + (-1))$$

$$= \text{sign}(-3 - 2 - 1) = \text{sign}(-6) = -1$$

Since $\hat{y} = y = -1$, no update is needed.

Final Values

After processing all examples, the final weight vector and bias remain:

$$\theta = [1, -2], \quad b = -1$$

4. [20] (Multi class Classification) Consider the following multi-class classification problem:

   - We have a total of m classes and a total of l training examples.
   - The l training examples are uniformly split across the m classes, such that each class has $\frac{l}{m}$ examples.
   - All training examples are of dimensionality d.

   We will assess the efficiency of the two training paradigms of all-vs-all and one-vs-all under different learning scenarios.

   - **All-vs-All:** For every pair of labels $\langle i, j \rangle$, a classifier is learned over the following dataset: the examples labeled with one class $i \in \{1, \ldots, m\}$ are considered "positive", and those labeled with the other class $j \in \{1, \ldots, m\}, j \neq i$ are considered "negative".
   - **One-vs-All:** For every label $i \in \{1, \ldots, m\}$, a classifier is learned over the following dataset: the examples labeled with the label i are considered "positive", and examples labeled with any other class $j \in \{1, \ldots, m\}, j \neq i$ are considered "negative".

   ### (a) Training Complexity with $O(dn^2)$ Learning Algorithm

   We are given a magical black-box binary classification algorithm (we don't know how it works, but it just does!) which has a learning time complexity of $O(dn^2)$, where n is the total number of training examples supplied (positive+negative) and d is the dimensionality of each example.

   What are the overall training time complexities of the all-vs-all and the one-vs-all paradigms, respectively, and which training paradigm is most efficient?

   ### (b) Training Complexity with $O(d^2n)$ Learning Algorithm

   We are now given another magical black-box binary classification algorithm which has a learning time complexity of $O(d^2n)$, where n is the total number of training examples supplied (positive+negative) and d is the dimensionality of each example.

   What are the overall training time complexities of the all-vs-all and the one-vs-all paradigms, respectively, and which training paradigm is most efficient, when using this new classifier?

   ### (c) Evaluation Complexity

   Suppose we have learned an all-vs-all multi-class classifier and now want to proceed to predicting labels on unseen examples.

   We have learned a simple linear classifier with a weight vector of dimensionality d for each of the $\frac{m(m-1)}{2}$ classifiers ($w_i^T x = 0$ is the simple linear classifier hyperplane for each $i = 1, \ldots, \frac{m(m-1)}{2}$).

   We have two evaluation strategies to choose from. For each example, we can:

   - **Counting:** Do all predictions, then perform a majority vote to decide the class label.
   - **Knockout:** Compare two classes at a time; if one loses, never consider it again. Repeat until only one class remains.

What are the overall evaluation time complexities per example for Counting and Knockout, respectively?

Solution:
Given:

- m classes and l training examples.
- The examples are uniformly distributed across the m classes, so each class has $\frac{l}{m}$ examples.
- Each example has dimensionality d.

(a) Learning Complexity: $O(dn^2)$

All-vs-All (AVA)

Each classifier is trained on a subset of the data where only two classes are considered. The number of unique classifier pairs is:

$$\frac{m(m-1)}{2} \tag{1}$$

Each classifier is trained on a dataset containing only the examples of two classes, so each classifier has:

$$n = 2 \times \frac{l}{m} = \frac{2l}{m} \tag{2}$$

The training time complexity per classifier is:

$$O(dn^2) = O\left(d\left(\frac{2l}{m}\right)^2\right) = O\left(d\frac{4l^2}{m^2}\right) \tag{3}$$

Since there are $\frac{m(m-1)}{2}$ classifiers, the total training complexity is:

$$O\left(\frac{m(m-1)}{2} \times d\frac{4l^2}{m^2}\right) = O\left(d\frac{2(m-1)l^2}{m}\right) = O\left(2dl^2\right) = O\left(dl^2\right) \tag{4}$$

One-vs-All (OVA)

Each classifier is trained on a dataset where one class is positive and the rest are negative. Each classifier has:

$$n = l \tag{5}$$

Since there are m classifiers, the total training time is:

$$O(m \times dl^2) = O(dml^2) \tag{6}$$

Comparison

Comparing the complexities:

$$AVA : O\left(dl^2\right)$$
$$OVA : O(dml^2)$$

Since $O(dml^2) > O\left(dl^2\right)$ for $m > 1$, the All-vs-All approach is more efficient.

(b) Learning Complexity: $O(d^2n)$

All-vs-All (AVA)

For each of the $\frac{m(m-1)}{2}$ classifiers:

$$O(d^2n) = O\left(d^2\frac{2l}{m}\right) = O\left(d^2\frac{2l}{m}\right) \tag{7}$$

Total complexity:

$$O\left(\frac{m(m-1)}{2} \times d^2\frac{2l}{m}\right) = O\left(d^2\frac{m-1}{m}lm\right) = O(d^2l(m-1)) \tag{8}$$

One-vs-All (OVA)

For each of the m classifiers:

$$O(d^2n) = O(d^2l) \tag{9}$$

Total complexity:

$$O(md^2l) \tag{10}$$

Comparison

Comparing the complexities:

$$AVA : O(d^2l(m-1))$$
$$OVA : O(d^2lm)$$

For large m, the complexities are approximately equal, but All-vs-All is slightly more efficient since its leading factor is $(m-1)$ instead of m.

(c) Evaluation Time Complexity

Each classifier has a weight vector of dimensionality d, and we need to compare the number of required evaluations.

Counting

Each of the $\frac{m(m-1)}{2}$ classifiers needs to compute a dot product:

$$O\left(\frac{m(m-1)}{2} \times d\right) = O(m^2d) \tag{11}$$

Knockout

The knockout method follows a tournament-like structure, performing $m-1$ pairwise comparisons:

$$O((m-1)d) = O(md) \tag{12}$$

Comparison

$$\text{Counting} : O(m^2 d)$$
$$\text{Knockout} : O(md)$$

Since $O(md) < O(m^2 d)$, the knockout strategy is more efficient.

5. [15 + 15] Complete the attached notebooks. You are permitted to use chatbots or other resources for assistance, but you must ensure that you fully understand the code and implementations. During the online sessions, you may be asked to explain specific functions, code lines, and your overall approach. Be prepared to demonstrate your understanding in detail.