Machine Learning and Deep Learning I Homework 1

Instructor: Joonseok Lee

Deadline: 2024/4/11 Thr, 18:00

Instruction

- No unapproved extension of deadline is allowed. Late submission will result in 0 credit.
- Optimize your code as much as you can. We do not guarantee to run unreasonably inefficient codes for grading. Remember, vectorization is important for efficient computation!
- You will be given skeleton files for doing your assignment. Detailed instructions are given in the comments below each method to complete. Please read them carefully before jumping into implementation!
- Most of experiment/visualization sections are already given to you. Just plot the results, and use them to verify your implementation unless other instructions are provided.
- Each assignment is built and tested under Google Colaboratory. If you work on a local machine, you need to handle version issue on your own.
- Explicitly mention your collaborators or reference (*e.g.*, website) if any. If we detect a copied code without reference, it will be treated as a serious violation of student code of conduct.

What to Submit

Please upload a single zip file named with your student ID (e.g., `2024-00000.zip`) on eTL, containing

- Your report containing answers to the questions (For the format, pdf, doc, hwp and pdf scan files of handwriting are allowed.)
- Your complete `hw1_gd.ipynb`, `hw1_nb.ipynb` files

Please keep in mind...

- The graders reserve the right to score zero for unreadable poor handwriting. We strongly encourage the students to submit with typed answers.
- Please erase any unnecessary print codes or comments that you have wrote before submission.

1 Maximum Likelihood [10 pts]

Suppose we have n i.i.d (independent and identically distributed) data samples from the following probability distribution. This problem asks you to build a log-likelihood function, and find the maximum likelihood estimator of the parameter(s).

(a) **Poisson distribution.** The Poisson distribution is defined as

$$P(x_i = k) = \frac{\lambda^k e^{-\lambda}}{k!} (k = 0, 1, 2, ...).$$

What is the maximum likelihood estimator of λ? [5 pts]

(b) **Exponential distribution.** The probability density function of Exponential distribution is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

What is the maximum likelihood estimator of λ? [5 pts]

2 Linear Regression [15 pts]

In class, we derived a closed form solution (normal equation) for linear regression problem: $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$, where $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{Y} \in \mathbb{R}^n$. A probabilistic interpretation of linear regression tells us that we are relying on an assumption that each data point is actually sampled from a linear hyperplane, with some white noise. The noise follows a zero-mean Gaussian distribution with constant variance. Mathematically,

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. In other words, we are assuming that each data point is independent to each other (no covariance) and that each data point has same variance.

Now, suppose we keep the independence (no covariance) assumption but remove the same variance assumption. Then, data points would be still sampled independently, but now they may have different variance σ_i. Thus, the covariance matrix would be still diagonal, but with different values:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}. \tag{2}$$

Derive the equation of maximum likelihood estimation for this problem using matrix-vector notations with Σ. [15 pts]

3 Gradient Descent - Parameter Estimation [35 pts]

Follow the instructions below to complete `hw1_gd.ipynb` provided on the ETL.
In this exercise, we are going to implement gradient descent to estimate the parameters of the Beta distribution. The Beta distribution is parameterized in terms of two positive parameters α, β:

$$p_\theta(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

where $\theta = (\alpha, \beta)$ and B (beta function) is a normalization constant to ensure that the total probability is 1.

(a) Implement `compute_loss(self, X)` by deriving the log-likelihood $\ell(\theta)$ of N i.i.d (independent and identically distributed) samples from Beta distribution. [10 pts]
(Recall that the likelihood function $L(\theta)$ of i.i.d samples is given as a product of the probabilities of all occurrences. Log-likelihood $\ell(\theta)$ is a log of $L(\theta)$.)

(b) Implement `backward(self, X)` by deriving the partial derivative of $\ell(\theta)$ with regard to α and β, respectively. [10 pts]
(You may use that $\frac{\partial B(\alpha,\beta)}{\partial \alpha} = B(\alpha, \beta)(\psi(\alpha) - \psi(\alpha + \beta))$. You do not need to expand $\psi()$.)
(For the implementation of $\psi()$ function, you may use `scipy.special.polygamma`.)

(c) Implement `train_step(self, X, lr)` to complete a gradient descent procedure for maximizing the likelihood. [5 pts]

(d) Generate 500 samples from a Beta distribution with parameters $\alpha = 3, \beta = 5$, and estimate α and β using gradient descent over your model, and plot the training history. [5 pts]

(e) Repeat experiment in (d) 100 times and plot histograms of the resulting estimates for the two parameters. [5 pts]

4 Naive Bayes Classifier [40 pts]

In this section, we are going to implement multinomial naive Bayes classifier.

Consider K number of classes: $\{C_1, ..., C_K\}$. An instance to be classified, $\mathbf{x}$, is represented by a histogram vector $\mathbf{x} = (x_1, ..., x_d) \in \mathbb{N}_0^d$, where $\mathbb{N}_0$ denotes non-negative integer. Suppose $\mathbf{x}$ follows a multinomial distribution, then the probability of observing $\mathbf{x}$ given the k-th class is

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_{j=1}^{d} x_j)!}{\prod_{j=1}^{d} x_j!} \prod_{j=1}^{d} {p_{kj}}^{x_j}, \tag{3}$$

where p_{kj} is probability of j-th event in k-th class. With the class's prior probability $p(C_k)$, we get the following joint distribution:

$$p(\mathbf{x}, C_k) = p(C_k)p(\mathbf{x} \mid C_k). \tag{4}$$

With this generative model, the naive Bayes classifier is defined as

$$\begin{aligned} \hat{y} &= \underset{k \in \{1,...,K\}}{\operatorname{argmax}} \log p(C_k \mid \mathbf{x}) \\ &= \underset{k \in \{1,...,K\}}{\operatorname{argmax}} \left(\log p(C_k) + \sum_{j=1}^{d} x_j \cdot \log p_{kj} \right). \end{aligned} \tag{5}$$

Here are **all the equations that you need to implement** the above classifier:

$$\hat{p}(C_k) = \frac{\alpha + \sum_{i=1}^{n} \mathbf{1}(y^{(i)} = k)}{K\alpha + n}, \tag{6}$$

$$\hat{p}_{kj} = \frac{\alpha + \sum_{i=1}^{n} \mathbf{1}(y^{(i)} = k) x_j^{(i)}}{d\alpha + \sum_{i=1}^{n} \mathbf{1}(y^{(i)} = k) \sum_{j=1}^{d} x_j^{(i)}}, \tag{7}$$

where $\mathbf{1}$ is the indicator function, n is the number of samples, and α is a smoothing factor to overcome zero-frequency problem, which is a hyperparameter. Now, follow the instructions below to complete `hw1_nb.ipynb` provided on the ETL.

(a) Implement `fit(self, X, y)` to get `self.priors` and `self.probs`. [10 pts]
(`self.priors` and `self.probs` correspond to (6) and (7), respectively.)

(b) Implement `predict(self, X)` which predicts the class of input instances. [10 pts]
(This method corresponds to (5))

Now, we are going to apply the model to a real-world task. The goal is to predict the security risk of a system by analyzing its log data generated from factory automation control systems. The security risk level ranges from 0 to 3 (4 classes), which represents severity of the security threat (3 indicates the highest risk). The log is a whole text describing the status of systems. We preprocess the logs into BoW (Bag of Words) composed of 1,000 different words, so that each log can be represented by a word histogram vector.

(c) Fit multinomial Bayes classifier with `train_X` and `train_y`. Then, measure the accuracy by comparing predicted class of `test_X` to true class `test_y`. [10 pts]

(d) By analyzing the logs, we found out that the word '`ossec`' does not appear in the logs of security level 2 at all. After setting α equal to 1, check the estimated probability of the word '`ossec`' given security level 2. Then explain why it is zero or non-zero, depending on your observation. [10 pts]
(You may try different smoothing factor α (e.g., $\alpha = 0$) and observe differences in estimated probabilities of the word '`ossec`'.)