Lectures of September 4th, 2019
Scribe: Lingfeng Zhang
Student Number: 300134245
Email: lzhan278@uottawa.ca

1 Course Information (Lecture 00)

- Professor: Yongyi Mao
- Office: SITE 5039
- Email: ymao@uottawa.ca
- Office Hour: send email to schedule an appointment(most Monday Morning)
- Course Prerequisites:
  - Mathematics
    - Linear Algebra
    - Calculus
    - Probability Theory and Statics
  - Programming Language
    - Python
  - Framework
    - Pytorch (Recommended)
    - TensorFlow
  - Writing Format
    - $\LaTeX$
  - Running Platform
    - Local
    - Google Colab (free GPU, running code on the cloud)
- Textbook and References
  1. No Textbook
  2. Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT press, 2016, Electronic version available at `http://www.deeplearningbook.org`
  3. Richard S. Sutton (Author), Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998; the second edition is to appear soon.
- Grading Scheme

Project-Augmented Exam-Gated Homeworks

$$Grade = HomeworkGrade \times WrittenExamGrade + ProjectGrade + ScribeBonus \quad (1)$$

$$Grade \leqslant 110\% \quad (2)$$

1. HomeworkGrade worth 60% and the maximum members of each homework group is 2.
2. WrittenExamGrade is the any number between 0 and 1,like the weight of HomeworkGrade. The content of this exam is related to the homework, making sure you have grasped the idea of your finished homework totally.
3. ProjectGrade worth 40% and the maximum members of each project group is 6. To choose the topic of project, it follows first-come-first-pick basis. There is extra 5% bonus about project presentation on the research plan(scheduled at some out-of-class slots) .
4. ScribeBonus worth up to 5%. Students can submit well-formatted PDF version lecture scripts by using LaTeX,each of which is awarded up to 1%. Maximum number of scripts for each lecture is 3(first come first grab in UO brightspace's Discussion module).

2 Machine Learning Basis (Lecture 01)

Deep Learning $\neq$ Machine Learning $\neq$ Artificial Intelligence

Definition by Tom M. Mitchell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

task := input/output specificatoin

Application in ML

- Predicting stock prices
- Loan application
- Discovering lung cancers
- Playing chess

- ect...

Main Classes of ML Problems

1. Supervised Learning
   (a) training set, testing set
   (b) with target
   (c) Examples
      i. Regression
      ii. Classification
2. Unsupervised Learning
   (a) find structure in dataset
   (b) Examples
      i. Clustering
      ii. Density Estimation, like DBSCAN Algorithm
3. Reinforcement Learning
   (a) Gaining more rewards by changing "state"
   (b) Example: Chess-Playing program (Alpha Go)

Model: $\mathcal{H}$ of hypothesis

1. Parametric models: with fixed number of parameters(initialized manually sometimes), independent of sample size
2. Non-Parametric models

Loss Functions: $\mathcal{L}(\theta)$

- θ represents model parameters in space Θ
- one-to-one correspondence: $\Theta \Longleftrightarrow \mathcal{H}$
- measures *error*
- learning is to reduce *error*

$$\hat{\theta} := arg \min_{\theta \in \Theta} \mathcal{L}(\theta) \tag{3}$$

In this equation, arg represents θ values when $\mathcal{L}(\theta)$ reaches the minimum

Example 1 ***Regression***

X takes value in $\mathbb{R}^K \Rightarrow K$ kinds of features

$$Y \approx f(X;\theta) \tag{4}$$

Linear Regression Model

$$Y \approx \theta^T X + b \tag{5}$$

$$Y \approx \begin{bmatrix} \theta \\ b \end{bmatrix}^T \begin{bmatrix} X \\ 1 \end{bmatrix} \tag{6}$$

Mean Square Error (MSE)

$$\mathcal{L}(\theta) := \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta^T x_i)^2 \tag{7}$$

In the equation (7), N represents the sample size

Maximum Likelihood Principle

Under the maximum likelihood (ML) principle, the best parameter of a given probability model is declared as the one that maximizes the data likelihood under the model.

principle is a kind of belief. We believe the principle is true but there is no prove of the principle.

i.i.d means independent and identically distribution

In probability theory and statistics, a collection of random variables is independent and identically distributed if each random variable has the same probability distribution as the others and all are mutually independent.[**?**]

likelihood is the probability of the data given the parameters of the model

Theorem:

Minimizing the MSE loss is equivalent to maximizing the data likelihood under a Gaussian probability model.

Proof: ML means Maximizing likelihood

$$\begin{aligned}
\hat{\theta}_{ML} &:= arg\max_{\theta} \log p(\mathcal{D} \mid \theta) \\
&= arg\max_{\theta} \log \prod_{i=1}^{N} p_{Y|X}(y_i \mid x_i; \theta) \\
&= arg\max_{\theta} \sum_{i=1}^{N} \log p_{Y|X}(y_i \mid x_i; \theta) \\
&= arg\max_{\theta} \sum_{i=1}^{N} \log N(y_i; f(x_i;\theta), \sigma^2) \\
&= arg\max_{\theta} \sum_{i=1}^{N} \log(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - f(x_i;\theta))^2}{2\sigma^2}}) \\
&= arg\max_{\theta} \sum_{i=1}^{N} -\frac{(y_i - f(x_i;\theta))^2}{2\sigma^2} \\
&= arg\min_{\theta} \sum_{i=1}^{N} (y_i - f(x_i;\theta))^2 \\
&= arg\min_{\theta} \mathcal{L}(\theta) \\
&= \hat{\theta}
\end{aligned}$$

Question 1: Why add log?

Aim at replacing exponential function with exponent, to reduce the complexity of function.

The value of probability is between 0 and 1, and the multiplication of these probability is also between $[0, 1]$. The log function is increasing in the interval [0,1], so it does not change the arg when the whole function reaches its maximum.

Question 2: How to illustrate the process from step 3 to step 4?

The variable Y follow the Gaussian Distribution with μ is equal to the function $f(X; \theta)$

For more details about this proof, to see this website:

`https://www.jessicayung.com/mse-as-maximum-likelihood/`

Minimize $\mathcal{L}(\theta)$ in closed form

$$\mathbf{X}^T := \begin{bmatrix} x_1 & x_2 & ... & x_N \end{bmatrix} \tag{8}$$

$$\mathbf{X} := \begin{bmatrix} x_1^T \\ x_2^T \\ ... \\ x_N^T \end{bmatrix} \tag{9}$$

$$\theta := \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ ... \\ \theta_m \end{bmatrix} \tag{10}$$

Minimize $\mathcal{L}(\theta)$ is minimizing $\parallel y - \mathbf{X}\theta \parallel^2$

$$ColumnSpace(\mathbf{X}\theta) = SPAN(\mathbf{X}) \tag{11}$$

$$\mathbf{X}(\hat{\theta}) := \hat{y} \tag{12}$$

$\hat{y}$ lives in space $SPAN(\mathbf{X})$ and y is out of the space. To minimize $\mathcal{L}(\theta)$, $y - \hat{y}$ is orthogonal to $SPAN(\mathbf{X})$.

$$\mathbf{X}^T(y - \hat{y}) = 0 \tag{13}$$

From equation (12) and (13):

$$\mathbf{X}^T y - \mathbf{X}^T\mathbf{X}\hat{\theta} = 0 \tag{14}$$

$$\mathbf{X}^T y = \mathbf{X}^T\mathbf{X}\hat{\theta} \tag{15}$$

$$(\mathbf{X}^T)^{-1}\mathbf{X}^T y = (\mathbf{X}^T)^{-1}\mathbf{X}^T\mathbf{X}\hat{\theta} \tag{16}$$

$$(\mathbf{X}^T)^{-1}\mathbf{X}^T y = \mathbf{X}\hat{\theta} \tag{17}$$

$$\mathbf{X}^{-1}(\mathbf{X}^T)^{-1}\mathbf{X}^T y = \mathbf{X}^{-1}\mathbf{X}\hat{\theta} \tag{18}$$

$$\mathbf{X}^{-1}(\mathbf{X}^T)^{-1}\mathbf{X}^T y = \hat{\theta} \tag{19}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{20}$$

Gradient Descent (GD)

feed whole training dataset to update θ

maybe reach the local minimum

time-consuming

$$\theta := \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ ... \\ \theta_n \\ b \end{bmatrix} \tag{21}$$

$$x := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \\ 1 \end{bmatrix} \tag{22}$$

$\lambda:$ learning rate

$\mathcal{L}(\theta)$ in equation (7)

start from random θ

$$\begin{aligned} \theta^{new} &:= \theta^{old} - \lambda \frac{d\mathcal{L}}{d\theta}(\theta^{old}) \qquad (23) \\ &= \theta^{old} + \lambda \frac{1}{N} \sum_{i=1}^{N} 2(y_i - {\theta^{old}}^T x_i)x_i \qquad (24) \end{aligned}$$

Stochastic Gradient Descent (SGD)

feed single training data point to update θ

maybe pass global minimum

start from random θ

$$\theta^{new} := \theta^{old} + \lambda 2(y - {\theta^{old}}^T x)x \tag{25}$$

Mini-Batched Stochastic Gradient Descent (Mini-Batched SGD)

feed batch-sized training dataset to update θ

to trade-off GD and SGD

start from random θ

$$\theta^{new} := \theta^{old} + \lambda \frac{1}{\mid \mathcal{B} \mid} \sum_{(x,y)\in\mathcal{B}} 2(y_i - {\theta^{old}}^T x_i)x_i \tag{26}$$

Smaller learning rate

slower convergence, i.e., longer training time

the final loss value closer to the optimum (or to a local optimum in case of complex loss surface)

higher chance of getting stuck at local optima (in case of complex loss surface)

Larger learning rate

harder convergence

Adaptive learning rate

reduce learning rate gradually (following certain criterion) beneficial

Larger mini-batch size

less noisy gradient, i.e, smoother drop of training loss

better utilizing parallel-computing resources (computing within a batch can be paralleled)

Some argue that noisy gradients(most occur in SGD) allow the optimization to escape from local optima (in case of complex loss surface)

Modelling

Higher-order polynomial model sometimes occur overfitting when the training dataset is not large enough. Most time higher-order polynomial model perform well than simple model.

References

[1] Clauset, Aaron, *A brief primer on probability distributions*, Santa Fe Institute, 2011.