CSE 475: Statistical Methods in AI **Monsoon 2019**

SMAI-M-2019 1: Mathematical Foundations of Machine Leaning - I

Lecturer: C. V. Jawahar *Date: 29 July 2019*

1.1 About the Course: SMAI

Machine learning has taken the central stage of modern artificial intelligence (AI). Data driven intelligence has found deep inroads into various walks of our life. Such techniques have found great success in diverse problems in perception (such as natural language processing, computer vision, speech understanding), control (robotics, automation) to computer systems (networks, operating systems, architecture design) and interdisciplinary domains (like biology, environmental science etc.). This makes this sub-discipline of modern AI, i.e., statistical methods in AI (SMAI), critical to the research in many areas of wider interest to IIIT-H, and the society around.

1.1.1 Course Plan and Execution

This course will focus on foundations of machine learning. (Though applications are hinted throughout the course, applications are not the focus.) This is not an advanced course. This builds on top of the basic math knowledge a good undergraduate student might have obtained in the initial years of study. Though this course may expect the students to do some implementations, this course is not a programming/implementation/system building oriented course.

1.1.1.1 Major Topics/Modules

- Mathematical Foundations of Machine Learning
- Linear Methods in Machine Learning
- Neural Networks and Introduction to Deep Learning
- Selected topics Eg. (i) SVMs (ii) Decision Trees and Forests (iii) Ensembling (iv) Clustering and Unsupervised Learning

1.1.1.2 Delivery

Two lectures in a week. (> 20 Lectures) each of 1.5 Hrs. Come to the class 5 mins before time. Come prepared. In addition, a Tutorial (not all weeks) and Office Hours (announced on moodle), every week. It is expected that student go beyond the lectures and read/think and some extra work.

1.1.1.3 Evaluation

Course expects students to regularly work on the course, understand the basics and apply the concepts to solve problems. Final grade will be based on:

- Two short quizzes in the scheduled time for 15 % (7+8)
- A mid semester exam in the scheduled slot for 15 %
- A final exam in the scheduled slot for 25 %
- Regular home works (say around 3 problems (could vary between 2-4)) loosely associated with each lecture. These problems need to be solved regularly. The expected solutions are mostly handwritten, though there could also be some simple programming/scripting/visualizing based question. Relative weight: 25 to 30%. To take care of personal emergencies and busy schedules, students are given option to submit a lesser number of questions or select the best sub-set of answers. Typically this will be 60 out of 75 questions.
- Three assignments or mini projects worth 15 to 20%. These are well defined larger problems, that demand more work and go to multiple weeks.
- Minor changes in the course evaluation (5%) based on the practical changes/adjustments need to make.

1.1.2 Pre-requisites

This course assumes some background in mathematics, specially (i) Linear Algebra, (ii) Probability and Statistics, and (iii) Differential calculus. Background from undergraduate mathematics courses is good enough. Also some familiarity with programming, specially programming using python is expected.

1.1.3 How to get best out of this course?

It is assumed that the students taking this course are genuinely interested in the subject. To get best out of this course, the following are recommended:

- **Be Regular:** Be regular to the class/lecture. Reach classroom 5 mins before the start. Remain fully attentive. Engage in the discussions. Don't shy away from asking questions. Solve/Attempt home works immediately after the class, without delaying to the last minute. Use office hours and interactions with TAs for clearing doubts and getting personalized supports.
- **Practice Makes Perfect:** Many problems (home works or that you find in text books) may look to be simple. Good to still tryout and improve by practicing. If you notice that you are taking more time to solve the problems, your problem solving skills can be sharpened by practicing.
- **Scripting and Rapid Prototyping:** There could be many simple programming style problems. Do not treat them as an elaborate "software development process". In most cases, the objective is to use a computer to solve a sufficiently large problem than teaching you programming or program design. Also expectation is to teach you how to look and interpret data/learning. Python has emerged as the default programming language in this space, given its advantages in rapid prototyping, wide support packages/libraries, and compatibility with wide varity of hardware/software infrastructure.
- **Improve your visualization and imagination skills** You may want to visualize what happens when a mathematical operation takes place. This helps in understanding math in an intuitive manner, specially for machine learning tasks. Geometry is key to this.
- **Be comfortable in thinking beyond** What is taught in the lecture needs to be interpreted and expanded in your mind. Do not look at lecture notes as fact-list for your examination preparation. Ask basic questions like why and why not with out fear. May be in the entire process, you will find a novel algorithm in your name.!!
- **Read and Explore:** You are expected to read and explore beyond attending the lectures. However, this area is very rich in content on Internet. A student may find it impossible to understand everything related available on Internet. Don't worry. No hurry. Suggested mode: (i) Attend lectures, take simple notes (ii) Read lecture notes (if available!!) on the same day of the lectures itself (iii) Read/View text books/writeups/videos. (iv) Solve home works (v) go wider and wilder reading.
- **Learn from your classmates** You learn a lot from the classmates, and from discussions. However, maintain strict academic integrity while solving home works.

1.2 Problem Space

Let us start with our basic problem. We are given a number of examples in the form $\{\mathbf{x_i}, y_i)\}$ for $i = 1, \ldots, N$. For the simplicity, we assume that $\mathbf{x_i}$ is a real d-dimensional vector. And y_i is a scalar (say real number or an integer). Our interest is in finding a function $f()$ such thar $f(\mathbf{x_i})$ is same (or very similar) as y_i.

(At this moment, we will deal with the requirement of the "very similar" to be as close as possible for all the N training samples that we have. Later we should also make sure that the same function will work for all the samples that we come across in the future also (i.e., unseen samples).)

For example, $\mathbf{x_i}$ could be an email that is represented with a real vector. and y_i could be 0 or 1 corresponding to "spam" (1) or not (0). In this case it is a classification problem. One may also formulate the problem as classification of the email into "spam" (0), "personal" (1) and "professional" (2). In this case, this is a multiclass classification problem. You may also predict "how important/urgent" is an email by looking at the content. In this case, then y_i is a real number (say in the raneg $[0-1]$ where 0 means least urgent and 1 means extremely urgent.

1.2.1 Classification

In classification problems y_i is an integer. What value we assign is of not much significance. For example, some people use 0 and 1, while some where else we use -1 and $+1$ as the class IDs. The choice is often based on some conveniences (simpler form of equations!!). In both these cases, it is a "binary" classification. In many cases we also call these classes as ω_1 and ω_2.

Multi class classification where the number of classes is more than 2 is a popular case. Though multiclass classification is very popular and important in practice, many discussions in the linear classifiers assume that the number of classes is only 2. That makes the life simple.

1.2.2 Regression

In the case of regression, y_i is real quantity. It could be a real vector or a simple real number. Let us assume it as a real number at this stage.

1.3 Vectors and Matrices - I

We started with discussions on classifying emails. (or we could separate apples from oranges). How do we represent these physical entities? There are two popular ways in which we represent physical entities. Vectors in d dimensions. Sequences of Vectors/Observations. This makes the vectors fundamental to this area. These observations get transformed into knowledge with a set of transformations. Matrices (or sometimes Tensors) play a critical role in this. We are often interested in entities like vectors and matrices in linear algebra. They play a critical role in this course.

Vector A vector $\mathbf{x} \in R^n$ or $[x_1, x_2, \ldots, x_n]^T$. We think vectors as points in a space, with each element giving coordinate along an axis.

For example a fruit is represented in 2 dimension with colour and shape as dimension. Often these two measurements are real numbers.

Matrix A matrix $\mathbf{A} = [A_{ij}]$ of order $M \times N$ has MN elements. When $M = N$, it is a square matrix.

Types of Matrices You may also know a number of special matrices like (i) Diagonal matrix (ii) Triangular Matrix (iii) Singular Matrix (iv) More (??).

Linear Transformations Tools and techniques from Linear algebra provides a way of compactly and conveniently representing and manipulating sets of linear equations. For example,

$$a_1x_1 + a_2x_2 + \ldots + a_Nx_N = b$$

is a linear equation and can be written compactly as

$$\mathbf{a}^T\mathbf{x} = b.$$

In this case, $\mathbf{x}$ is transformed into a scalar b with the help of $\mathbf{a}$.

We can also linearly transform $\mathbf{x}$ by a matrix $\mathbf{A}$ as

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

A popular problem that we are interested (for many later lectures) is in finding *what is the best transformation* for our problem.

1.3.1 Operations on Matrices and Vectors

- **Dot product, inner product or scar product** of two vectors, $\mathbf{x}$ and $\mathbf{y}$ is often represented as $\mathbf{x} \cdot \mathbf{y}$ or $\mathbf{x}^T\mathbf{y}$ (more generally also as $< \mathbf{x}, \mathbf{y} >$. The product results in a scalar.
- $\mathbf{B} = \mathbf{x}\mathbf{y}^T$ is a matrix of size $d \times d$, if both $\mathbf{x}$ and $\mathbf{y}$ are $d \times 1$. *In this case,* $\mathbf{B}$ *is of rank 1. It has only one non-zero eigen value. Why?*
- $\mathbf{z} = \mathbf{A}\mathbf{x}$ is a vector. $\mathbf{A}$ has d columns. If $\mathbf{A}$ has $m(< d)$ number of rows, Then $\mathbf{A}$ is doing a *dimensionality reduction.*
- $\mathbf{A} + \mathbf{B}$ is feasible if both have the same size, and the result is a matrix.
- The multiplication of two matrices, $\mathbf{A}$ $\mathbf{B}$ is feasible when the number of columns of $\mathbf{A}$ is same as the number of rows of $\mathbf{B}$.
- Given a scalar α, we can compute the vectors αx and $\alpha\mathbf{A}$, by multiplying each element by α.

1.3.2 Norms

Norm A norm of a vector is informally the length of the vector, represented as $||\mathbf{x}||$. A popular way to visualize is as the Euclidean L2 norm as

$$\sqrt{\sum_{i=1}^{n} x_i^2}$$

A popular class of norms of our interest is Lp norm defined as:

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

When we $p = 2$, it is L2 norm, which we already saw. Another popular norm is $L1$ norm. $||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$

There are two special (pseudo) norms of interest

$$L_\infty = \max_i |x_i| \text{ and } L_0 = \text{ number of no-zero } x_i$$

There are also matrix norms. The most popular one is Frobenius norm defines as

$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} A_{ij}^2} = \sqrt{Tr(A^TA)}$$

1.3.3 Notes

Some of the key words that you need to know include: (i) Scalars, Vectors, Matrices and Tensors (ii) Multiplying Matrices and Vectors (iii) Identity and Inverse Matrices (iv) Linear Dependence and Span (v) Norms (vi) Special kinds of matrices and vectors (viii) Eigen decomposition and Singular value decomposition (ix) The determinant This notes may not discuss all these in detail. You may want to read any popular text book for refreshing the details.

1.4 Geometric Interpretations

Consider a set of points $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$, each $\mathbf{x_i} \in R^2$. We can visualize this as a set of points on a 2D plane as in Figure **??**(a). The respective dimensions are the first and second dimensions of the point/sample $\mathbf{x}$. If $\mathbf{x}$ is an fruit, the first and second dimension could be the "colour" and "smell" of it. (Indeed, colour and smell will have to be captured with an appropriate sensor.) In practice, these sensor measurements are also scaled or normalized to certain ranges like $[-1, +1], [0, 1]$ or $[0, 100]$.

Why do we have to normalize the data/vectors/samples?

Lines, Planes and Hyperplanes Consider the 2D plane example, all equations of the form

$$w_1 x_1 + w_2 x_2 = \mathbf{w}^T \mathbf{x} = 0$$

are lines in this plane.

When the data is in 3D, these are planes and when the data is more than 3 dimension, these are hyperplanes. You should be comfortable in imagining such geometric entities in high dimensionm. Most of the figures, we use, will be in 2D. With some effort, we can also draw 3D.

The equation of line we discussed above always pass through the origin. We will see how to relax this in the next lecture (Hint: *bias*)?

Sides of Planes/Lines When a sample $\mathbf{x} = [x_1, x_2]^T$ is on this line, then $\mathbf{w}^T\mathbf{x}$ is zero. All the points where this product is Positive is one side of the line and all the points where this dot product is negative is on the other side of the line.

In general the hyperplane divides the space into two half spaces $H1$ and $H2$ as $H1 = \{\mathbf{x}|\mathbf{w}^T\mathbf{x} > 0\}$ and $H2 = \{\mathbf{x}|\mathbf{w}^T\mathbf{x} < 0\}$. Don't worry about the equality case too much.

Angle between vectors We know that $\mathbf{x}^T\mathbf{y} = ||\mathbf{x}|| \cdot ||\mathbf{y}|| \cdot \cos\theta$. Where θ is the angle between the vectors.

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||}$$

In many cases, both the vectors of interest could be "normalized" or unit norm, and the cosine angle between these vectors is just the dot product of the two vectors (since the denominator is one).

The cosine of the angle between two vectors is often used as a measure of similarity between two vectors. This quantity is in the range [-1,+1]. When the similarity is maximum, it is +1.

Figure 1.1: TBD

1.5 Nearest Neighbour Algorithm

Let us now look at a simple, yet practical, algorithm. We start by looking at a 2D visualization of emails in Figure **??**. The red dots are spam emails and green dots are non-spam emails.

Problem: We are given now, three new emails $\mathbf{x}_2$, $\mathbf{x}_2$ and $\mathbf{x}_3$. We have been asked, whether these are spam emails or not?.

It may be obvious for us that $\mathbf{x}_1$ is a spam email and $\mathbf{x}_2$ is a non-spam email. Why? We have a simple argument. All the emails similar to $\mathbf{x}_1$ are spams and we also treat $\mathbf{x}_1$ as spam. Similarly all emails similar to $\mathbf{x}_2$ are non-spam and we treat $\mathbf{x}_2$ as non-spam. Why is it? Two very similar emails will map to very similar points in the feature space. We can also use the distance between the points as similarity between the points. The smaller the distance, the larger the similarity.

How do we convert this into a computational algorithm? We can do it in two steps. (i) Given a test sample, find the nearest neighbour in the training data (ii) Assign the label of the neatest neighbour to the test sample.

This nearest neighbour classification algorithm works well in practice. However, to make it more robust to noisy data, we use K nearest neighbours instead of 1 nearest neighbour. As a result, label of the test sample is the majority of the labels in the neighbouring examples.

This leaves an open question. What is the value of K? Often it is a small odd number such as 3, 5 or 7.

- Why should K be odd?
- Can there be any time ties in the majority labels? How do we handle the tie breaks?
- What are the practical issues in implementing this algorithm for spam filters on gmail?

1.5.1 Distance Functions

In the previous section, we assumed that the distances are Euclidean. It need not be. There are many other possible distance functions. Nearest neighbour algorithm can use any distance function.

Metric Distance A distance function $d(\cdot,\cdot)$ is a metric if it obeys triangular inequality.

For points $\mathbf{x}$ $\mathbf{y}$ and $\mathbf{z}$, triangular inequality is defined as

$$d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z})$$

1.6 A note on notations

You may want to develop the skill of interpreting the notations without being explicitly written in all cases. In general small bold letters are vectors and capital bold letters are Matrices. (You will also see some Greek letters later.)

Suffixes like A_{ij} is the (i,j) th element of the matrix. $\mathbf{A_i}$ is the i th matrix and $\mathbf{x}_i$ is the ith vector. Transpose is represented as $*^T$. $\mathbf{x}^T$ is the row vector if $\mathbf{x}$ is a column vector. $\mathbf{A}^T$ is $N \times M$, if $\mathbf{A}$ is $M \times N$.

In machine learning, we are often interested in learning a set of coefficients. Frequently, they are represented as $\mathbf{w}$ or matrix $\mathbf{W}$. (However, not all learnable parameters are w.)

There are also many hyperparameters in the solution.

1.7 Home Works

1. We have a binary classification problem with 10 samples, 5 each from class A and B. All samples are in 2D. The data is given below with first 5 from class A and the next 5 from class B.

   $\mathcal{D} = \{[0.5, 0.5]^T, [0.75, 0.25]^T, [0.25, 0.75]^T, [1, 1]^T, [2, 2]^T,$

   $[-2, -2]^T, [-0.25, -0.25]^T, [-1.5, -2.5]^T, [-3, -2]^T, [-2, -3]^T\}$

   Given an unknown sample $\mathbf{q} = [0, 0]^T$, find its label using a simple K nearest neighbour algorithm with K = 1 and K = 3.

   A hand drawn sketch with approximate space is also expected.

2. Create a data set (say an xl sheet !) of the following data for all the players in India's WC Cricket squad 2019 (15 players).

   - Name
   - Age
   - Height of the player
   - Role (batsman=1, Bowler = 2, Wicket Keeper = 3, Al-rounder = 4)
   - Batting average (ODI)
   - Bowling Average (ODI)
   - Number of matches played (ODI).

   (a) Using a nearest neighbour scheme find who is the most similar player to Kumar Sangakara in the Indian team? Is it MS Dhoni? (Note thar Sangakara has also to be represented in the same feature space.)

   (b) Who is most similar to David Warner in Indian team? Is it Rohit Sharma?

   (c) Test one more players of your choice.

   May be that the results of Nearest Neighbour method is not meaningful here. What can we do to improve the similarity? Can you define a weighted distance function that solves this immediate worry?

   Submit data, results and analysis in handwritten form.

3. We know that the vector/sample $\mathbf{x}$ is on left of the hyperplane if $\mathbf{w}^T\mathbf{x} < 0$ and right (other side) if $\mathbf{w}^T\mathbf{x} > 0$.

   - Given $\mathbf{x_1} = [1, 2, 3]^T, \mathbf{x_2} = [0, 3, 4]^T, \mathbf{x_3} = [2, 4, 4]^T$, find 'a' hyper-plane $\mathbf{w}$ such that $\mathbf{w}^T\mathbf{x}_1 < 0$ and $\mathbf{w}^T\mathbf{x}_2 > 0$ $\mathbf{w}^T\mathbf{x}_3 > 0$
   - Find four points $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$ and $\mathbf{x_4}$ in 4 dimension (d=4) such that that there exists no $\mathbf{w}$ that can keep $\mathbf{x}_1$ and $\mathbf{x}_2$ on one side and $\mathbf{x}_3$ and $\mathbf{x}_4$ on the other side. (similar to ExOR in 2D)