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

SMAI-M-2019 3: Mathematical Foundations of ML - III

Lecturer: C. V. Jawahar *Date: 5 Aug 2019*

3.12 Problem Space - III

3.12.1 How do we formulate the Training Problem?

Let us come back to our problem. We are given training examples $\{(\mathbf{x_i}, y_i)\}$ $i = 1, \ldots, N$. What do we want to do? We want to find the most appropriate $\mathbf{w}$ such that we can mimic y_i as $f(\mathbf{w}, \mathbf{x}_i)$.

Indeed we may not find a single $\mathbf{w}$ that can make $y_i = \mathbf{w}^T\mathbf{x_i}$ for all i. This could be due to various reasons including errors, noise or uncertainty in the data. Therefore we want to model the problem as find the "most appropriate" $\mathbf{w}$. This naturally lead to an optimization problem. Most appropriate in what sense? We need to define an appropriate sense or objective that can be computed. This is the objective function. In machine learning, we also use the term loss function or error function frequently. All these are used with very similar meanings.

Our problem is then to find $\mathbf{w}$ that minimizes

$$Total{-}Loss = \sum_{i=1}^{N} Loss{-}Per{-}Sample = \sum_{i=1}^{N} L(\mathbf{w}, \mathbf{x_i}, y_i)$$

Why do we have to sum over i? Why not products? That is also possible. We do summation so that differentiation is easier later. (do you remember how to differential $u + v$ and uv?). There may be many different ways in which you can define loss functions.

Q: Do you see any other advantage or disadvatage of products over sum? Which will be more sensitive to outliers? (samples which may have very larger error). Or samples where the loss vanishes (becomes zero)?

3.12.2 Loss Function

Consider the regression problem in 1D. You have (x_i, y_i). By augmenting 1, x_i becomes a 2 dimensional vector $\mathbf{x}_i$. Our problem is to find the vector $\mathbf{w} = [w_1, w_0]^T$ such that the model is $y = w_1x_1 + w_0$. Look at this as a line fitting.

Error or loss in this case is the difference between the model prediction and actual.

$$L(\mathbf{w}, \mathbf{x_i}, y_i) == (y_i - \mathbf{w}^T\mathbf{x_i})^2$$

We square the error such that no loss is negative. This is required since we will now be adding the loss from different examples to get the total loss.

And the total loss or objective is then sum over all the examples

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x_i})^2 \tag{3.3}$$

Note that we have only one $\mathbf{w}$ for all the samples.

Similarly for a classification problem, loss can be 1 if the classification is wrong and 0 if the classification is correct.

Let us assume that the classifier is

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \text{else} \end{cases} \tag{3.4}$$

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(1 - y_i \cdot f(\mathbf{w}, \mathbf{x_i})) \tag{3.5}$$

- Q: If we had used a $0-1$ convention for the classes, how the equation could have been written?
- Q: The problem with this loss is that this is not differentiable. Why?

3.12.3 Optimization

The optimization problem we need to solve is

$$minimze \;\; J(\mathbf{w}) \tag{3.6}$$

Though we know what problem to solve, very often we can not find the "best $\mathbf{w}$" in practice. There are two prominent classes of optimization problems:

- Convex optimization. A well behaved class of problem. You can find the optima. And often efficiently.
- Non-Convex optimization problem. This is a class of nasty optimization problems. Unfortunately, we will encounter them very frequently. In this case, very often, we have to be happy with "a" minima/solution and not the best solution.

More details of these classes of problems is beyond the scope of this course.

3.13 Vectors and Matrices -III

1. When is $\mathbf{Ax} = \mathbf{b}$ uniquely solvable, when $\mathbf{A}$ is a square matrix?
2. When $\mathbf{A}$ has more rows than column or when $\mathbf{A}$ has more columns than rows, how do we solve, and what type of a solution we obtain?

3.13.1 Tensors

3.14 Eigen Values and Eigen Vectors

Eigen values and eigen vectors of a square matrix A are defined by the characteristic equation

$$\mathbf{Ax} = \lambda \mathbf{x}$$

Here $\mathbf{x}$ is the eigen vector and λ is the eigen value.

- Q: How do we compute eigen values and eigen vectors with hand for small matrices?
- Q: How many non-zero eigen values could be there for $\mathbf{A}$ of $n \times n$?
- Q: How are λ_i related to determinant and trace?
- Q: Can eigen values be imaginary? When?
- Q: What can you say about the PD and PSD from eigen values?

3.14.1 Eigen Decomposition of a matrix

Let $\lambda_1, \ldots, \lambda_n$ be the eigen values and $\mathbf{v_1}, \ldots, \mathbf{v_n}$ be the eigen vectors of the $n \times n$ matrix $\mathbf{A}$

We know that

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \forall i$$

Let $\mathbf{V}$ be a $n \times n$ matrix with columns as $\mathbf{v_1}, \ldots, \mathbf{v_n}$.

$$\mathbf{AV} = \mathbf{\Lambda V} = \mathbf{V\Lambda}$$

where Λ is the diagonal matrix of eigen values.

$$A = \mathbf{V\Lambda V^T}$$

since eigen vectors are orthogonal. We can also write it as

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$$

We will see how the eigen decomposition of a square matrix A is related to the SVD of $\mathbf{A}$. And also in the case of non square matrices, how the SVD of AA^T and $A^T A$ are related to the eigen values and eigen vectors of A.

Eigen Spectrum

It is good to observe the eigen values in a sorted manner as a spectrum to observe t

3.15 Singular Value Decomposition (SVD)

Singular value decomposition is a very popular factorization scheme with many applications in machine learning. The singular value decomposition (SVD) is a factorization of a real or complex matrix. It has many useful applications in signal processing, statistics, machine learning and optimization. In general, the matrices that we deal with are real.

Formally, the singular value decomposition (SVD) of an $m \times n$ matrix $\mathbf{A}$ is a factorization of the form

$$\mathbf{A} = \mathbf{UDV^T}$$

3.15.1 Minor Variation in Notations

You see minor variations in the notation of SVD across resources. This comes from whether your original matrix $\mathbf{A}$ has more rows or columns. If $\mathbf{A}$ is of $m \times n$, the rank of $\mathbf{A}$ can be $\min(m, n)$. In many applications of machine learning, we see that the rank of $\mathbf{A}$ is often much smaller than $\min(m, n)$. This is because data has a structure. And ML techniques thrive in situations where there is some structure for the data.

- U is $m \times m$, $U^TU = I_m$, D is diagonal $m \times n$ and V is $n \times n$ as $V^TV = VV^T = I_n$.

  One may look at U as an $m \times m$ real or complex unitary matrix, D is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V^T (the conjugate transpose of V, or simply the transpose of V if V is real) is a $n \times n$ unitary matrix.

- Alternatively, U is $m \times n$, $U^TU = I_n$, D is diagonal $n \times n$ and V is $V^TV = VV^T = I_n$.

In both casesm $\mathbf{UD}$ is finally the same. Is it so?
Q: Convince that these two different ways will not lead to different interpretations.

Notes

- The diagonal entries D_{ii} of D are known as the singular values of M. If D is interpreted as $m \times n$ with $m < n$, then the last $m - n$ rows are just zeros
- The m columns of $\mathbf{U}$ and the n columns of $\mathbf{V}$ are called the left-singular vectors and right-singular vectors of $\mathbf{A}$, respectively.
- Note that $U^TU = I$ and $V^TV = VV^T = I$ The singular value decomposition and the eigen decomposition are closely related. We know that the eigen decomposition leads to

$$\mathbf{A} = \sum_{i=1}^{N} \lambda \mathbf{x}\mathbf{x}^T$$

Do you see the connection between this and the SVD?

3.15.2 Interpretations

- The left-singular vectors of M are eigenvectors of MM^T.
- The right-singular vectors of M are eigenvectors of M^TM.
- The non-zero singular values of M (found on the diagonal entries of D) are the square roots of the non-zero eigenvalues of both M^TM and MM^T.

These are quite simple to see. For example if we look $\mathbf{A}^T\mathbf{A}$ as

$$\mathbf{A}^T\mathbf{A} = (UDV^T)^T(UDV^T) = VDU^TUDV^T = VD^2V^T$$

3.15.3 Examples of Effective Uses

Applications that employ the SVD include computing the pseudoinverse, least squares fitting of data, matrix approximation, and determining the rank, range and null space of a matrix. Let us assume $A = UDV^T$ Or

$$\mathbf{A} = \sum_i D_i u_i v_i^T$$

$$A^{-1} = (UDV^T)^{-1} = VD^{-1}U^T$$

D^{-1} is easy to calculate since it is diagonal in n flops. If A is singular, one can find the approximation by discarding the zero elements. $D_i^{-1} = \frac{1}{D_i}$ is $D_i > t$ and zero otherwise. One can solve $Ax = b$ with the help of this inverse.

3.15.4 Other Factorization Methods

SVD is not the only matrix decomposition technqiues that we use. There are many more. Urge you to be also familiar with other such as

- Cholesy
- QR, LU etc.

3.16 Interpretations

Example 1 Consider a vector $\mathbf{x} \in R^d$. Let us now compute a $d \times d$ square matrix $\mathbf{A} = \mathbf{x}\mathbf{x}^T$.

We know that the matrix $\mathbf{A}$ is $d \times d$. However, what is the rank of $\mathbf{A}$? (It can be at max r. But that is a useless answer for this problem.) It is 1. Why? Each row of $\mathbf{A}$ is only a linear multiple of the other row. first row is $x_1\mathbf{x}$. Second row is $x_2\mathbf{x}$ and so on. Or we can express a row as a multiple of another one. And there fore there is only one linearly independent row.

Q: What about columns? How many linearly independent columns be there? Can you argue?

Q: If we take SVD of $\mathbf{A}$, what do we obtain U, D and V? If we compute eigen values and eigen vectors of $\mathbf{A}$, what do we obtain? Validate your answers analytically and then also verify it with a numerical library.

Example 2: Data matrix Consider a phenomenon/process that gives us data on a straight line. i.e., $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$. We now create a matrix $\mathbf{D}$ by arranging these vectors as columns. i.e., $\mathbf{D}$ is a $2 \times N$ matrix. What is the rank of $\mathbf{D}$?

If all columns of $\mathbf{D}$ are samples from on a line we can easily see that any column can be obtained from another column with a scalar multiplication. Thus $\mathbf{D}$ is rank 1.

Note: In general **Data matrix D** is obtained by keeping samples as columns. Such matrices are of $d \times N$ in size. Which one is higher in practice, d or N?

Q: What about samples that lie on a plane in 3D? What should be the rank of this? What about hyperplanes in d dimension?

Q: Can you guess (and later empirically validate) what should be the SVD of $\mathbf{D}$?

Example 3: Noisy Data Matrix In real life, the situation is not that ideal, even if the process is a simple linear model, you may not get samples that can lie exactly on a line/plane/hyperplane. This is what often we see in data sets.

In such cases, the rank or non-zero eigen/singular values do not really provide insight. For example, an eigen value of very small magnitude may be practically as good as zero. This lead to “discarding” eigen/singular values that are very small (compared to the major ones).

For example, consider our $\mathbf{D}$ of $2 \times N$. If we have one sample not on the line, (while all other $N-1$ are on). What do you see in SVD? You get a major eigen value/singular value and then the second one that is non-zero is very small. A popular question is can we then find a ”rank 1” approximation of $\mathbf{D}$ and remove the influence of this “odd” sample. Yes, indeed. We can take SVD of $\mathbf{D}$ as UDV^T. Then make the seond singular value zero. Then compute back UDV^T.

(Note: Don’t get confused with the data matrix D and SVD’s D).

Q: Can you visualize what happens to this on data with a small python code?

3.17 Subspaces

In general, it is observed that though the data is d dimensional (when you observe, curate), it lies in a lower dimension for all practical purposes. This is a sub-space of the original space.

- Lines in 2D
- Planes in 3D
- Lines in 3D
- Planes in 4D.
- d' dimensional hyperplanes in d dimensional space.

A line is 3D can be understood as all points that are in certain direction from a fixed point. Or

$$\{\mathbf{x} | \mathbf{x} = \mathbf{x}_0 + u\}.$$

We can define a subspace in this manner.

$$\{\mathbf{x} | \mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^{d'} \mathbf{u}_i\}$$

There are only d' linearly independent vectors in this sub-space. (You will see more formal details in Chapter 2 (2.6?) of the book).

Let us come back and ask the question of why do data lie in a sub-space? Most physical processes generate data based on some simple model. However, when we are capturing the data, we have no cue about what is the inherent dimension of the data.

Think of the problem of email classification as spam or non-spam. As a designer of the algorithm, we land up defining and extracting a feature set based on the Vocabulary of English. (Q: What is the rough size of this?). However, in practice the data/problem is in a much low-dimension. Unfortunately, we have no systematic way to capture the data in the lower dimension. (Note: in the case, of emails, it is NOT 2. We may be showing examples in 2D in the lecture, does not mean that the real life problems are often in 2D!!).

3.18 Dimensionality Reduction

When the data lies in a lower-dimensional subspace, it is natural to ask the question, why are you solving in the original space and why not move to the lower dimensional space. It is a valid question, and popularly this is done by **dimensionality reduction**.

In general, during dimensionality reduction, we transform the data from high dimension (say d) to low dimension (say d'). In linear dimensionality reduction schemes, this is achieved by a matrix multiplication.

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

This is a linear dimensionality reduction.

We can also do nonlinear dimensionality reduction with nonlinear functions (such as Neural Networks) as

$$\mathbf{z} = f(\mathbf{w}, \mathbf{x}).$$

In either case, our objective is to find an effective lower dimensional space where data lie. Or find a space where the problem can be solved better.

3.18.1 Curse of Dimensionality

There are also many curse for the dimensionality. If we have a feature/data representation that is very large, it creates many practical problems for us. For example, if we are estimating some data statistics (like covariance matrices), the number of parameters to estimate increase quadratically with the number of dimensions. And indeed, this demand more and more samples to estimate these parameters.

There are many other practical issues like:

- Combinatorics
- Sampling
- Optimization
- Distance Function
- etc.

Urge to read articles such as Wiki on curse of dimensionality to appreciate this behavior.

3.18.2 Low-Dimensional Projections

It is common to create low-dimensional projections of large dimensional data for effective and efficient machine learning algorithms

3.19 Home works (Submit by 12 Aug 5pm)

1. With hand compute the eigen values and eigen vectors of the following matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

What is the trace, determinant and rank of this matrix? How is this related to the eigen values/vectors. Demonstrate.

2. Consider a linear transformation (such as a dimensionality reduction) that maps a q-dimensional vector $\mathbf{x}$ to a p dimensional vector $\mathbf{y}$ (where $p \leq q$; BTW, for dimensionality reduction, in practice, p can be much smaller than q.) as

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

   (a) What are the dimensions of the matrix $\mathbf{A}$?

   (b) Can there exists a matrix $\mathbf{A}$ such that the Euclidean distance between $\mathbf{y}_1$ and $\mathbf{y}_2$ is same as $\mathbf{x}_1$ and $\mathbf{x}_2$. Under what conditions on $\mathbf{A}$ and $\mathbf{x}$, is this feasible?

   (c) Create a specific example and demonstrate $\mathbf{A}$ when (a)$q = 2$ and $p = 2$, (b) $q = 2$ and $p = 1$ and (c) $q = 4$ and $p = 2$. If no examples like these are feasible, explain/prove why.

3. Consider a line in 2D (say line l as $w_1x_1 + w_2x_2 + w_3 = 0$)

   (a) Assume we have a set of N points on this line $\mathcal{D}_1 = \{\mathbf{x}_i = [x_1^i, x_2^i]^T\}$. Let the mean of these points be $\mu = [\mu_1, \mu_2]^T$. How many non-zero eigen values will be there for the following matrices?

$$\mathbf{A}' = [x_1^1 - \mu_1, x_2^1 - \mu_2][x_1^1 - \mu_1, x_2^1 - \mu_2]^T$$

$$\mathbf{A} = \frac{1}{N}\sum_{i=1}^{N}[x_1^i - \mu_1, x_2^i - \mu_2][x_1^1 - \mu_1, x_2^i - \mu_2]^T$$

   or compactly

$$\mathbf{A} = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i - \mu][\mathbf{x}_i - \mu]^T$$

   (b) Assume we have a set of points on a line perpendicular to the line l and passing through μ. (What is the equation of this line?) Let $\mathcal{D}_2 = \{\mathbf{x}_j = [x_1^j, x_2^j]^T\}$ be a set of N points on this perpendicular line. Let the mean of these points be $\mu = [\mu_1, \mu_2]^T$. How many non-zero eigen values will be there for the following matrices?

$$B' = [x_1^1 - \mu_1, x_2^1 - \mu_2][x_1^1 - \mu_1, x_2^1 - \mu_2]^T$$

$$B = \frac{1}{N}\sum_{i=1}^{N}[x_1^i - \mu_1, x_2^i - \mu_2][x_1^i - \mu_1, x_2^i - \mu_2]^T$$

or compactly

$$\mathbf{B} = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i - \mu][\mathbf{x}_i - \mu]^T$$

(c) Now consider the main question. Consider a set of points "around" the line with mean as μ. Let us compute the covariance matrix as:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i - \mu][\mathbf{x}_i - \mu]^T$$

What will be the eigen value and eigen vector of the Σ. Discuss.

(d) Write a python program to create 1000 points around the line (not just on the line). Compute covariance matrix, eigen values and eigen vectors. (i) Plot the data (ii) Plot the eigen vectors (first with red and second with green starting from mean. (iii) Overlay these two plots and create a single plot.

Note: this question is long. But the answer is not that. Conceptual understanding of this question could be the key for appreciating PCA in one of the next lectures.