CS 541: Artificial Intelligence

Instructor: Jie Shen
Scribe: Lauren Brandenstein and Shaurya Chandhoke

Lecture 1
August 31, 2021

Course Overview

Our course instructor is Jie Shen (jie.shen@stevens.edu) and our TA is Tianhao Zhu (tzhu12@stevens.edu). Office hours for this course are 1-3 pm EST Tuesday. Any questions on homework or grading, we can email the TA and CC the instructor.

There is no required textbook for the course and all exam and quiz material will be based off of the lecture notes. There are a couple recommended textbooks, but they are not required.

For grading, Scribe Notes are worth 40% of the final grade. Scribe Notes will be completed in groups of 2 for each lecture. The Midterm Exam is scheduled for October 18th and it is worth 30% of the final grade. There will be occasional quizzes throughout the semester which will count for 30% of the final grade. These quizzes help to refresh what we learned in the previous lecture.

Scribe Notes

The group of two must send a draft of scribe notes within one week of the lecture and send to the TA and instructor. The group will receive feedback on their scribe notes and they will have 3 additional days for final lecture notes. The point of scribe notes is that they are comprehensive enough that anyone in the class can understand what was covered if they were not able to attend the lecture.

Policy

Any additional questions on grading/regrading should be presented to the TA, the instructor will not address grading.

Additional tips for the course are to utilize Google, Wikipedia, and talk with other classmates. However, everything must be in your own words or have an included acknowledgment.

About the Course

This is not an introductory course and it is made to challenge students and think out of the box. This course is research oriented and it will emphasize theoretical aspects and mathematical models of AI applications. If you feel that you are not comfortable with calculus, linear algebra, and probability, this may not be the course for you. CS 556 is another course that has a mathematical foundation for machine learning, which may be more of a beginner course.

Syllabus

In this course we will be covering,

- Review of calculus, probability, linear algebra
- Random projection - An efficient method of reducing the dimensions of data
- Singular value decomposition, principal component analysis
- K-means clustering, subspace clustering
- Dictionary learning and sparse coding
- Low-rank matrix estimation with applications to recommender systems such as your recommended videos on YouTube
- Computational social science
- Robust mean estimation, robust classification
- Knowledge distillation

Note: The midterm will cover the first 5 bullet points, up to dictionary learning and sparse coding.

Linear Algebra Overview

For this course, we do not need extensive knowledge of linear algebra, but it important to have some basic knowledge on vector and matrix operations, norms, etc. To begin, we should understand what a vector is. A d-dimensional column vector $\mathbf{x}$ is a set of d numbers represented below. It is important to understand what a vector is because most data in artificial intelligence will be represented as a vector.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Vector Operations

Transpose: The transpose of a column vector will be a row vector. On the other hand, the transpose of a row vector is a column vector.

$$\mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_d \end{bmatrix}$$

Example:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$
$$\mathbf{x}^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Vector multiplied by a scalar: Assuming that a $\in \mathbb{R}$,

$$a\mathbf{x} = \begin{bmatrix} ax_1 & ax_2 & \cdots & ax_d \end{bmatrix}$$

Example:

$$a = 2 \qquad \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad a\mathbf{x} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}^T$$

Vector addition: Add each component of the individual vectors to a new vector,

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 + y_1 & x_2 + y_2 & \cdots & x_d + y_d \end{bmatrix}$$

Note: After learning how to multiply a vector by a scalar, and vector addition, a$\mathbf{x}$ + b$\mathbf{y}$ will also be known.

Vector multiplication: Before multiplying vectors or matrices, it is important to check the dimensions of each vector or matrix. The rule of thumb for vector multiplication is that the number of columns of the first matrix must be equal to the number of rows in the second matrix. For example, take the following matrices M and N. While this example uses matrices, the same rule applies to a vector since they can be considered as a one dimensional matrix.

$$M_{m \times p} \qquad N_{p \times n}$$

Matrix M has p columns and Matrix N has p rows, so these matrices can be multiplied. The matrix they produce following multiplication will have dimensions (m $\times$ n). In other words, the dimensions of a matrix produced by vector and matrix multiplication will have the same number of rows as the first matrix and the same number of columns from the second.
Following the rules outlined above, vector multiplication is simply computing the inner products or individual components of each vector and is represented by:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{d} x_i y_i \in \mathbb{R}$$

Example:

$$Let \ \mathbf{x} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \ Calculate \ \mathbf{x}\mathbf{y}^T.$$

$$\mathbf{x}\mathbf{y}^T = 1 \times 1 + 2 \times 1 + 3 \times 1 = 6$$

Note: In this example, the first vector $\mathbf{x}$ has 3 columns and the second vector $\mathbf{y}$ once transposed, has 3 rows. This means they can be multiplied!

Vector Norms

Two-Norm: The two-norm is one of the more widely used norms in artificial intelligence and machine learning. It is found by taking the sum of each component of the vector squared and is represented by the following formula:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}$$

One very important use of two-norms is to measure the difference of two objects by creating 2 vectors and calculating the difference between the two vectors using, $\|x - y\|_2$.

Example:

$$\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \qquad \|x\|_2 = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$$

One-Norm: Defined as the sum of absolute values of vector. Can be calculated with the following formula:

$$\|x\|_1 = \sum_{i=1}^{d} |x_i|$$

Infinity-Norm: Defined as the maximum value of the components in $\mathbf{x}$. Can be calculated with the following formula:

$$\|x\|_\infty = max_{1 \leq i \leq d} |x_i|$$

Matrices

The difference between vectors and matrices is that a vector is a set of numbers, and a matrix is a set of vectors. Many of the operations mentioned earlier such as multiplication by a scalar or addition, are very similar to matrix multiplication by a scalar or matrix addition. Some examples to showcase this are below.

Example (Matrix multiplication by scalar):

$$a = 3 \qquad \mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad a\mathbf{X} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

Note: This is very similar to vector multiplication by a scalar. Every element of the matrix is multiplied by a.

Matrix addition is also very simple where one would sum the corresponding components of each matrix, but the matrices you are adding must have the same dimensions. The new matrix created from the matrix addition will also have the same dimensions as the original two matrices.

Matrix multiplication can be more tricky since you can only multiply matrices when the number of columns of the first matrix is equivalent to the number of rows of the second matrix. The new matrix created from the matrix multiplication will have the same number of rows as the first matrix and the same number of columns as the second matrix. These details were described in length under vector multiplication. For clarity, below is an example of vector multiplication.

Example: Let $\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. Calculate $\mathbf{x}^T\ \mathbf{y}$.

$$\mathbf{x}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Here, the first vector $\boldsymbol{x}$ *has dimensions (3 × 1) and vector* $\boldsymbol{y}$ *has dimensions (1 × 3). Since the number of columns of* $\boldsymbol{x}$ *is equivalent to the number of rows in* $\boldsymbol{y}$*, these two vectors can be multiplied. The resulting matrix will have dimensions of (3 × 3)*

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} x^T \times 1 & x^T \times 2 & x^T \times 1 \end{bmatrix}$$

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} 1 \times 1 & 1 \times 2 & 1 \times 1 \\ 2 \times 1 & 2 \times 2 & 2 \times 1 \\ 3 \times 1 & 3 \times 2 & 3 \times 1 \end{bmatrix}$$

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 3 & 6 & 3 \end{bmatrix}$$

Other matrix definitions to know:

Symmetric Matrix: If $\mathbf{X}^T = \mathbf{X}$, then $\mathbf{X}$ is a symmetric matrix. In other words, the individual components above the diagonal of a matrix are the same as their counterparts below the diagonal.

Following the idea of a symmetric matrix, a **diagonal matrix** is one where the all elements that are not located along the diagonal are 0. For example:

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Lastly, if you try to compute the inverse of a matrix, it must be a square matrix. In other words, the number of rows must be equal to the number of columns.

Probability Overview

We know probability to be the measure of likelihood that an event will occur from 0 to 1. The most basic example from any statistics course is flipping an unbiased coin where the outcome is either heads or tails, each with the same probability of .5.
In this example we can denote flipping a coin a random variable X. The two outcomes or events of this random variable are heads (0) or tails (1). Events = 0,1.

Before going further into probability, it is important to define different random variables, discrete and continuous.

Discrete Random Variables

If the random variable X is discrete, it will take on a given value in a countable set, meaning there are a countable number of outcomes for X. The probability of a discrete random variable will be defined by the probability mass function shown below. This function determines the probability that X takes on a specific value.

$$p(x) = \mathbb{P}(X = x)$$

Continuous Random Variables

On the other hand, if random variable X is continuous it can take on any value in a given range of numbers. To find the probability that a continuous random variable X is less than or equal to a given x, one would use the probability density function as shown below.

$$\mathbb{P}(x) = \int_{-\infty}^{x} p(z)dz$$

For random variables, there are two popular distributions, uniform and normal/Gaussian. In simple terms, a uniform distribution refers to one where any outcome is equally likely to occur as another event. In a normal or Gaussian distribution looks like a bell shaped curve centered around the mean. In a normal distribution, it can be denoted as $\mathbb{N}(\mu, \sigma^2)$ where μ represents the mean and σ^2 represents the variance. The probability density function or PDf of a normal distribution is shown below.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Example of Uniform Distribution: In a uniform distribution with range of [1,3], the PDF of random variable Z is P(Z) = $\frac{1}{3-1} = \frac{1}{2}$
With this distribution,

$$\mathbb{P}(Z \leq a) = \begin{cases} 0 & a < 1 \\ 1 & a > 3 \\ \dfrac{a-1}{3-1} & a \in [1,3] \end{cases}$$

Expected Value

There are different formulas for expected value for discrete and continuous random variables. For discrete random variables, its expectation is found by taking the summation of the outcome multiplied by its probability. On the other hand, to find the expected value of a continuous random variable,the expected value will be the integral of the outcome of the random variable multiplied by its probability. The formulas are shown below.

$$Discrete : \mathbb{E}[X] = \sum xp(x)$$
$$Continuous : \mathbb{E}[X] = \int xp(x)(dx)$$

Fun Practice

Lets take an example where you are playing a game for money. Each time you play the game you will either win $1 or lose $1. The probability that you win a game is .6. This example shown in terms of probability is shown below.

$$\mathbb{P}(X = 1) = .6 \quad \mathbb{P}(X = -1) = .4$$

If you want to win $100, how many times will you need to play this game?

In terms of the expectation of this game, $1 \times .6 \ + \ -1 \times .4 = .2$. We found this by multiplying each outcome by its probability, the formula for expectations of discrete random variables described above. Therefore, since x_i are independently and identically distributed, we have $\sum_{i=1}^{n} \mathbb{E}[x_i] = .2n$ Since we want to win $100,

$$\mathbb{E}[S] = \mathbb{E}[\sum x_i]$$
$$\mathbb{E}[S] = \sum_{i=1}^{n} \mathbb{E}[x_i]$$
$$\mathbb{E}[S] = .2n$$
$$100 = .2n$$
$$n = 500$$

This means that the average number of times you would play the game in order to win $100 is 500 times. However, the expected value is the average outcome of multiple trials. You could play the game 500 times and not win $100. There is no certainty with expected value, it is just the average outcome.

A different take on expectation

Many shouldn't be surprised to realized that in the real world, flipping a coin 500 times in fact may *not* yield $100. In fact, in the real world, it may take less than 500 tries or more than 500 tries as the expected value simply yields the average number of tries expected. It turns out that the expected value of the sample random variable problem is too generalized for certain real world scenarios; gamblers or weather forecasters would most likely not rely solely on this statistical measure for their predictions. As such, a more robust alternative to the expected value will be observed.

Markov's Inequality

Often given the probability, the expected value is determined; we shall observe what relationships exist when the given information is switched. The first unique relationship is determined by the following theorem:

Theorem : Markov's Inequality *Let X be a positive continuous random variable ($X > 0$), then*

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}[X]}{t} \quad \forall t > 0$$

where t is some target or evaluation value.

This inequality states that given a positive random variable, the upper bounded probability that the random variable will be *at least* t is *at most* the proportion between the expected value of the random variable and t. By solving for the upper bound with low probability, the lower bound can be found with high probability. Figure 1 shows a graphical representation of this relationship. **[1]**

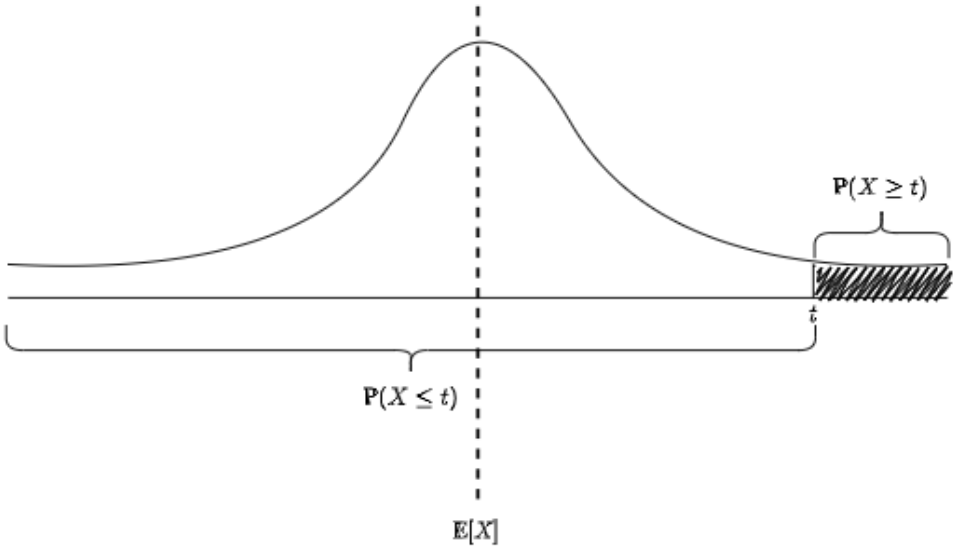


Figure 1: Intuition behind the Markov Inequality. By solving for $\mathbb{P}(X \geq t)$ with low probability, the lower bound $\mathbb{P}(X \leq t)$ can be found with high probability.

Proof

This relationship can be established intuitively by observing the expectation of a positive continuous random variable **[2]**

$$\begin{aligned}
\mathbb{E}[X] &= \int_0^\infty xp(x)\,d(x) \\
&\geq \int_t^\infty xp(x)\,d(x) \\
&\geq t\int_t^\infty p(x)\,d(x) \\
&\geq t\mathbb{P}(X \geq t)
\end{aligned}$$

$$\begin{aligned}
\therefore \quad & \mathbb{E}[X] \geq t\mathbb{P}(X \geq t) \\
& \frac{\mathbb{E}[X]}{t} \geq \mathbb{P}(X \geq t)
\end{aligned}$$

$$q.e.d$$

Example

Using the same data provided in the previous example, the following is now known

$$\begin{aligned}
\mathbb{P}(X = 1) &= 0.6 \\
\mathbb{P}(X = -1) &= 0.4 \\
\mathbb{E}[X] &= 0.2n
\end{aligned}$$

Using the following information, an alternative question can be asked – what is the upper bound of the proportion of flips that will yield at least $100?

Solution

The solution can be found by leveraging the Markov Inequality [**4**]

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}[X]}{t}$$

As a statistician, it is up to you to determine the confidence level of the bounds. Normally, the assumption is made that the right hand side

$$\frac{\mathbb{E}[X]}{t} = 0.01$$

By setting the upper bound to be a low probability value, we are establishing that we can get $X \leq t$ with high probability instead. Plugging in what is known yields

$$\begin{aligned}
\mathbb{P}(X \geq t) &\leq \frac{0.2n}{t} \\
\mathbb{P}(X \geq 100) &\leq \frac{0.2n}{t} \\
\mathbb{P}(X \geq 100) &\leq 0.002n
\end{aligned}$$

This establishes the upper bound probability inequality for the problem. To obtain the lower bound probability, the following rule can be applied

$$\begin{aligned}
\mathbb{P}(X \leq t) &= 1 - \mathbb{P}(X \geq t) \\
\mathbb{P}(X \leq 100) &= 1 - \mathbb{P}(X \geq 100)
\end{aligned}$$

The lower bound inequality thus follows these properties

$$\begin{aligned}
1 - \frac{0.2n}{t} &= 0.99 \\
X &\leq t \\
-X &\geq -t
\end{aligned}$$

This results in the following system of equations

$$\begin{cases} 1 - \frac{0.2n}{t} = 0.99 \\ t = 100 \end{cases}$$

*(Recap) A brief definition of the variance

Expanding upon the knowledge of the expected value, it is also possible to derive another statistical measure called the **variance**, denoted $Var(X)$. It is defined as a measure of the spread between data within a set and is defined as follows

Definition : Variance

$$Var(X) := \mathbb{E}[X - \mathbb{E}[X]]^2 = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

If $X_1, X_2, \cdots, X_n$ are independent, then $Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i)$

Chebyshev's Inequality

Markov's Inequality, while effective, is too loose of a measure. There exists another inequality that can refine the results more efficiently that expands upon the underlying foundation.

Theorem : Chebyshev's Inequality

$$\begin{aligned} &\textit{If} \quad X > 0 \\ &\textit{Then} \quad \mathbb{P}(|X - \mathbb{E}[X]| \geq t) \leq \frac{Var(X)}{t^2} \quad \forall t > 0 \end{aligned}$$

Proof

The inequality is derived using the following intuition:

Let Y be another positive continuous random variable defined as follows **[4]**

$$Y = |X - \mathbb{E}[X]|^2$$

Then the Markov inequality yields

$$\begin{aligned} \mathbb{P}(Y \geq t) &\leq \frac{\mathbb{E}[Y]}{t} \\ \mathbb{P}(|X - \mathbb{E}[X]|^2 \geq t^2) &\leq \frac{\mathbb{E}[|X - \mathbb{E}[X]|^2]}{t} \\ \mathbb{P}(|X - \mathbb{E}[X]| \geq t) &\leq \frac{Var(X)}{t^2} \\ & \qquad\qquad\qquad\qquad q.e.d \end{aligned}$$

Which is the Chebyshev Inequality. In short, it is an extension of the Markov Inequality with tighter constraints.

Example

Using the same information as in the previous examples, the following is known

$$\begin{aligned} \mathbb{P}(X = 1) &= 0.6 \\ \mathbb{P}(X = -1) &= 0.4 \\ \mathbb{E}[X] &= 0.2n \\ Var(X) = \sum_{i=1}^{n} Var(X_i) &= \sum_{i=1}^{n} \left(\mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2\right) = \sum_{i=1}^{n} (1 - 0.04) = 0.96n \end{aligned}$$

Similar to the Markov inequality process, the same question is asked – what is the upper bound of the proportion of flips that will yield at least \$100?

Solution

The solution can be found by leveraging the Chebyshev Inequality. Similar to the Markov Inequality example, the following assumption will be made

$$\frac{Var(X)}{t^2} = 0.01$$

Which when used in the inequality can be used to find the below system of equations **[4]**

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq t) \leq \frac{Var(X)}{t^2}$$
$$\mathbb{P}(|X - 0.2n| \geq 100) \leq \frac{0.96n}{t^2}$$

Assuming that the confidence level is 1%, then with probability 99%

$$\mathbb{P}(|X - 0.2n| \leq 100) \geq 1 - \frac{0.96n}{t^2}$$

Rewriting the left hand side without the absolute value signs and focusing on the lower bound yields

$$-t \leq X - 0.2n \leq t$$
$$X \geq 0.2n - t$$

This implies that with probability 99%

$$\begin{cases} 1 - \frac{0.96n}{t^2} = 0.99 \\ 0.2n - t = 100 \end{cases}$$

Solving this system of equations yields $n \approx 3325$ with $t \approx 0.2(3325) - 100 = 565$

Tradeoffs

With Markov's inequality, it is only necessary to know the expectation to obtain what is considered the sub-optimal solution. With Chebyshev's inequality, an extra piece of information is required (the variance), but while it's still considered sub-optimal, it is still much better than Markov's.

References

[1] Inamdar, Tanmay. Randomized Algorithms.
https://homepage.cs.uiowa.edu/šriram/5360/fall18/notes/9.10/week4Notes.pdf.

[2] Behavior of Independent Random Variables.
https://www.stat.cmu.edu/ larry/=stat705/Lecture2.pdf.

[3] Inequalities of Markov And Chebyshev.
https://math.dartmouth.edu/ m20x18/markov.

[4] Shen, Jie. *Artificial Intelligence.* Aug 31 2021.

Note: All theorems and definitions not explicitly cited are taken from lecture 1 of *Artificial Intelligence* by Jie Shen.

CS 541: Artificial Intelligence

Instructor: Jie Shen
Scribe: Ian Gomez and Nick Guo

Lecture 2
September 7, 2021

---

1 Hoeffding's Inequality

Assume that you have a random variable X. Where X comes from a symmetric Bernoulli distribution where

$$P(X = 1) = 1/2$$

and

$$P(X = -1) = 1/2$$

If you gather samples from this distribution N times to get $X_1, X_2, ..., X_n$ as $i.i.d$ random variables. you are able to get a tighter lower bound than the bounds gathered from Chebyshev's and Markov's inequality. Which is given by the formula

$$P(\sum_{i=1}^{n} a_i X_i \geq t) \leq \exp(\frac{-t^2}{2||\mathbf{a}||_2^2})$$

where $\mathbf{a}$ is a vector such that $a_i \in R^n$. By this definition we can define the vector $\mathbf{a}$ as a vector which contains the value 1 for all $a_1, ..., a_n$. We define $\mathbf{a}$ this way to simplify the equation. The equation gets simplified because the squared L2 norm of this $\mathbf{a}$ would be the number of values in $\mathbf{a}$. As show by $\sum_{i=1}^{n} 1^2 = n$. Therefore we get the equation

$$P(\sum_{i=1}^{n} X_i \geq t) \leq \exp(\frac{-t^2}{2n})$$

We can then change this equation to get a two sided bound (lower and upper bound) by taking the absolute value of the summation. Which then transforms the equation to

$$P(|\sum_{i=1}^{n} X_i| \geq t) \leq 2\exp(\frac{-t^2}{2n})$$

We can then generalize this to unsymmetrical Bernoulli distributions by utilizing the formula.

$$P(|\sum_{i=1}^{n} X_i - E[X]| \geq t) \leq 2\exp(\frac{-t^2}{2n})$$

Where the definition of non-symmetric Bernoulli distributions are any Bernoulli distribution such that $P(X = 1) = p$ and $P(X = -1) = 1 - p$

2 Using Hoeffding's Inequality

From last lecture we posed the question of how many times we had to play a coin toss game to ensure that we win 100 dollars given that $P(X = 1) = 0.6$ and $P(X = -1) = 0.4$

Last week we utilized Markov's inequality to get an upper bound, and then we utilized Chebyshev's inequality to get a lower bound. This week we will use Hoeffding's inequality to get an even tighter lower bound than Chebyshev's.

To use the more generalized form of Hoeffding's inequality we must first compute the expectation of the random variable.

$$P(|\sum_{i=1}^{n} X_i - E[X]| \geq t) \leq 2\exp(\frac{-t^2}{2n})$$

Where in this case it is

$$E[\sum_{i=1}^{n} X_i] = \sum E[X_i] = \sum P(X = 1) - P(X = -1) = \sum 0.6 - 0.4 = 0.2n$$

we then plug this into the generalized form of Hoeffding's to get

$$P(|X - 0.2n| \geq t) \leq 2\exp(\frac{-t^2}{2n})$$

As an aside the reason that we do not have a summation here is because in this scenario X represents the summation of $X_1...X_n$.

Given that we know that $|X - 0.2n| \geq t$ we know that $-t \leq X - 0.2n \leq t$. We can solve for -t to get the lower bound.

$$-t \leq X - 0.2n$$

$$X \geq 0.2n - t$$

Then after plugging in 100 for the value of the summation of $X_1, ..., X_n$

$$100 = 0.2n - t$$

After doing this we now have two equations in which we can solve for the value n

$$\begin{cases} 100 = 0.2n - t \\ 0.01 = 2\exp(\frac{-t^2}{2n}) \end{cases}$$

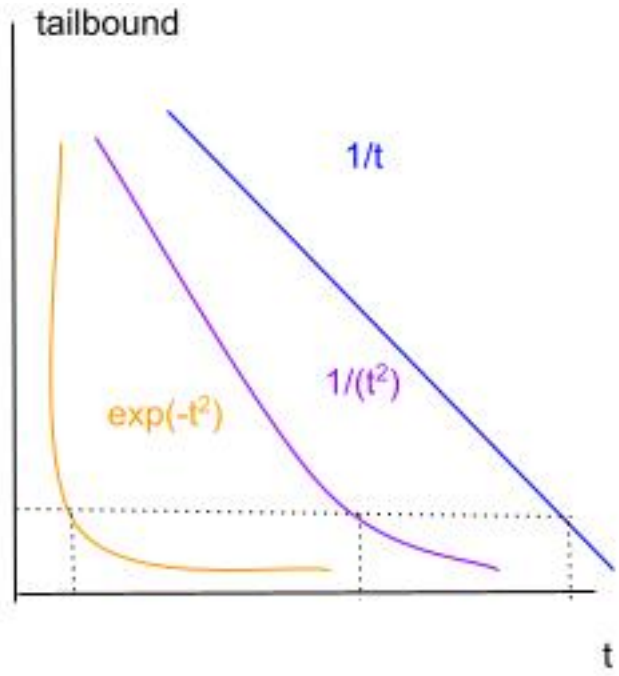
By solving this system of equations to get the value of n we get a value of approximately 700. Which is a much smaller amount than the value we got last week from solving Chebyshev's inequality, which was approximately 3000. The reason that we have a much better approximation is because we are using a stronger inequality. Where this inequality utilizes the fact that we know more information about our distribution to get an estimate which is more accurate.

3 Comparison between Markov, Chebyshev, and Hoeffding inequalities

One important thing to notice, is that the Markov, Chebyshev, and Hoeffding inequalities have different scales of accuracy in respect to the tightness of the bound. The reason being that some inequalities are stronger than the other inequalities. We can analyze this by looking at the tail bound (t) of the function. Given the three equations.

$$\begin{cases} Markov : P(X \geq t) \leq \frac{E[X]}{t} \\ Chebyshev : P(|X - E[X] \geq t) \leq \frac{Var(X)}{t^2} \\ Hoeffding : P(|\sum_{i=1}^{n} X_i - E[X]| \geq t) \leq 2\exp(\frac{-t^2}{2n}) \end{cases}$$

We can see that as t gets larger Markov's inequality decreases at a linear rate. Chebyshev's inequality decreases at a quadratic rate. Finally, Hoeffding's inequality decreases at a exponential rate. Therefore when for a fixed tail bound. What we will notice is that by for solving each inequality you will observe that Hoeffding will have a lower value of t then Chebyshev. Furthermore Chebyshev will have a smaller value than Markov. This property can easily be shown in this graph



Where you can easily see that when fixed at a certain tail bound. The corresponding t value is smaller with the same relation that was previously described.

4 Markov's Inequality Proof of Correctness

Given the definition of Markov's Inequality

$$P(X \geq t) \leq \frac{E[x]}{t}$$

we need to prove that the value of $P(X \geq t)$ is greater than the RHS of the inequality. For this we first consider that

$$E[X] = \int_0^\infty xP(x)dx$$

this can be decmoposed into two integrals where the first integral is the integral from 0 to our tailbound t. While the other integral is from t to infinity.

$$E[X] = \int_0^t xP(x)dx + \int_t^\infty xP(x)dx$$

Now we know that the value of $\int_0^t xP(x)dx + \int_t^\infty xP(x)dx$ is going to be greater than the value of $\int_t^\infty tP(x)dx$. This is because t is the smallest number in the second integral, and in the first integral we know that the value of the integral is guarenteed to be positive. Therefore we can say that

$$E[X] \geq \int_t^\infty tP(x)dx$$

$$E[X] \geq t\int_t^\infty P(x)dx$$

$$E[X] \geq tP(X \geq t)$$

$$\frac{E[X]}{t} \geq P(X \geq t)$$

$$P(X \geq t) \leq \frac{E[X]}{t}$$

5 Markov's Inequality Proof of Tightness

The definition of tightness on an inequality is that there exists a value where an inequality becomes an equality. For example $x^2 \geq 0$ is a tight bound since at $x^2 = 0$ when $x = 0$ $x^2 + 1 \geq 0$ is not a tight bound since $x^2 + 1 \neq 0$ for all values of x

To prove this for Markov's inequality we consider the distribution

x	Pr
0	$1 - \frac{1}{t}$
1	1/t

therefore $E[X] = 1/t$. Therefore plugging the values into Markov's inequality

$$P(X \geq t * E[x]) \leq 1/t$$

$$P(X \geq t * \frac{1}{t}) \leq 1/t$$

$$P(X \geq 1) \leq 1/t$$

Since the distribution only has values 0 and 1 $P(X \geq 1)$ is the same as $P(X = 1)$

$$P(X = 1) \leq 1/t$$

$$P(X = 1) = 1/t$$

$$1/t = 1/t$$

Therefore since we have found a value where the inequality becomes an equality, we have proved that Markov's inequality is a tight inequality. Although we have proved that Markov's inequality cannot be improved we have also said that Chebyshev's and Hoeffding's provide a tighter bound than Markov's inequality. The reason that these are improvements is that for Chebyshev's and Hoeffding's inequality we have extra information which we did not have in Markov's. For example in Chebyshev's we have the extra information of the variance of the distribution, and in Hoeffding's we know that the distribution is a Bernoulli distribution.

6 Moment Generating Functions

In statistics we call the expectation the first order moment of a distribution. While variance is a second order moment of a distribution. The variance is a second order moment because the order of the random variable is 2 by the definition $E[X^2] - (E[X])^2$.

Now for a moment generating function we need the exponent i such that $\forall i \geq 1$. We know the value of $E[X^i]$. This means that knowing the moment generating function of a distribution $\iff$ we know everything about the distribution.

When trying to understand how to represent a distribution mathematically describe a distribution, we usually utilize the PDF of the function. For example when using a Gaussian distribution we describe the distribution using the PDF parameterized by σ^2 and μ.But if giving a random diagram without knowing its PDF, we cannot describe the distribution.

In order to describe a specific distribution, we have to use the moment generating function. It is defined in the following way:

Find a random variable X, then

$$MGF(X) \triangleq E[e^{\lambda X}]$$

In a more perfect way, we utilize the definition that

$$MGF_X(\lambda) \triangleq E_{X\sim D}[e^{\lambda X}]$$

By utilizing the Taylor expansion of $e^{\lambda x}$. We can then calculate the value of all the moments of the function, since the Taylor expansion evaluate e^x as

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

So for $e^{\lambda X}$ we have

$$e^{\lambda X} = \sum_{n=0}^{\infty} \frac{\lambda^n X^n}{n!}$$

which gives you the different orders of the random variable X. Therefore it is equivalent to knowing all the moments $E[X^i]$ for all values of i. Which proves that we by using this function we know everything about the distribution.

For example to show this we can look at the derivative of the MGF. When plugging in the Taylor expansion of e^x into the MGF we see that we get this formula.

$$MGF(X) \triangleq E[\sum_{n=0}^{\infty} \frac{\lambda^n X^n}{n!}]$$

Using the property of expectation we know that the expectation of a sum is equivalent to the sum of the expectation.

$$MGF(X) = \sum_{n=0}^{\infty} E[\frac{\lambda^n X^n}{n!}]$$

After this we can differentiate the MGF k times to get the value

$$\frac{\partial}{\partial \lambda^k} MGF(X) = \sum_{n=k}^{\infty} E[\frac{\lambda^{n-k} X^n}{(n-k)!}] = \sum_{n=k}^{\infty} \frac{\lambda^{n-k}}{(n-k)!} E[X^n]$$

Finally if we plug in $\lambda = 0$ we get.

$$MGF(X) = E[X^k]$$

This is because when $n = k$ the value of

$$\lambda^{n-k} = \lambda^{k-k} = \lambda^0 = 1$$

and when $\lambda = 0$ the value of every other component of the summation will evaluate to 0.

7 Hoeffding's Inequality Proof of Correctness

To prove Hoeffding's Inequality we need to prove the upper bound provided by the formula.

$$P(\sum_{i=1}^{n} X_i \geq t) \leq \exp(\frac{-t^2}{2n})$$

The first concern of this proof is that we cannot apply Markov's inequality due to the assumption that all values are positive in Markov's inequality. So first lets look at the the probability

$$P(\sum_{i=1}^{n} X_i \geq t)$$

$$= P(\sum_{i=1}^{n} \lambda X_i \geq \lambda t)$$

Where we assume that λ is a positive value.

$$= P(e^{\sum_{i=1}^{n} \lambda X_i} \geq e^{\lambda t})$$

Which is true because the e^x is a monotonically increasing function. Now since the values of the exponential function are always positive. We can now apply Markov's inequality. Therefore we get the formula.

$$P(e^{\sum_{i=1}^{n} \lambda X_i} \geq e^{\lambda t}) \leq \frac{E[e^{\sum_{i=1}^{n} \lambda X_i}]}{e^{\lambda t}}$$

To further simplify this we need to first compute the expectation. Using the property of exponents we modify the equation to

$$E[e^{\sum_{i=1}^{n} \lambda X_i}] = E[\prod_{i=1}^{n} e^{\lambda X_i}]$$

Since we have assumed that the random variable $X_1, ..., X_n$ are *i.i.d.* We know that the $E[\prod X_i] = \prod E[X_i]$. Therefore we get

$$E[\prod_{i=1}^{n} e^{\lambda X_i}] = \prod_{i=1}^{n} E[e^{\lambda X_i}]$$

Now we need to compute the value of $E[e^{\lambda X_i}]$ which is given by the value

$$E[e^{\lambda X_i}] = \frac{1}{2}e^{\lambda} + \frac{1}{2}e^{-\lambda} = \frac{1}{2}(e^{\lambda} + e^{-\lambda})$$

Back in lecture 1 we learned proved in Quiz 0 that $\frac{1}{2}(e^{\lambda} + e^{-\lambda})$ is upper bounded by

$$\frac{1}{2}(e^{\lambda} + e^{-\lambda}) \leq e^{\frac{\lambda^2}{2}}$$

Now since we are trying to find an upper bound for Hoeffding's inequality we are allowed to substitute it in since it will still be an upper bound to the original value. Therefore we upper bound the original expectation with

$$E[\prod_{i=1}^{n} e^{\lambda X_i}] \leq \prod_{i=1}^{n} e^{\frac{\lambda^2}{2}} = e^{\frac{n\lambda^2}{2}}$$

Plugging that back into the inequality we are solving for we get

$$P(e^{\sum_{i=1}^{n} \lambda X_i} \geq e^{\lambda t}) \leq \frac{e^{\frac{n\lambda^2}{2}}}{e^{\lambda t}}$$

$$P(e^{\sum_{i=1}^{n} \lambda X_i} \geq e^{\lambda t}) \leq e^{\frac{n\lambda^2}{2} - \lambda t}$$

Now we need to find a value of lambda such that it minimizes the value of the right hand side of the function. We choose the term which minimizes the value since we want the tightest bound. To find that value we solve the quadratic equation and get the value of $\lambda = \frac{t}{n}$. Then plugging that back in to the RHS of the equation we get

$$e^{\frac{-t^2}{2n}}$$

Therefore the overall formula is

$$P(\sum_{i=1}^{n} X_i \geq t) \leq e^{\frac{-t^2}{2n}}$$

Which is the expected final result.

CS541 Artificial Intelligence

Instructor: Jie Shen
Scribe: Elena Korshakova and Kaustubh Lad

Lecture 3
September 14, 2021

Nearest Neighbor Search

Given. Query $q \in \mathbb{R}^d$, and database $x_1, x_2, ..., x_n$.

Goal. Find the closest x_i to q under the ℓ_2-metric $\sim$ find $i = \text{argmin}_{1 \le i \le n} \|x_i - q\|_2$

The most simple **solution**: brute force, with computational cost $O(nd)$. How can we improve the computational cost of this algorithm?
Two solutions: reduce n (by hashing) OR reduce d (by dimension reduction).
For the dimension reduction, there are two following approaches:

- PCA, feature selection: data-dependent, not efficient; in fact, by applying this technique to this type of problems we have a computational cost $O\left(nd^2\right)$ if $d < n$, or $O\left(n^2 d\right)$ if $d > n$, so an algorithm become even less efficient;
- Random projection: independent of data, very fast.

Lets consider the random projection approach in detail:

1. Determine the new dimension of projection some $k < d$ (example: $d = 100$, $k = 5$). (Suppose k-value is already mentioned beforehand to be reduced to)
2. Generate a random matrix of projection $A \in \mathbb{R}^{k \times d}$, so that each $a_{ij} \overset{i.i.d.}{\sim} N(0,1)$ *i.e.* it gives a **Standard Gaussian Matrix**. Notice that A is data-independent and has dimension of $k \times d$.
3. Calculate projections:

$$\forall i \in \{1, 2, ..., n\}: \quad x_i \in \mathbb{R}^d \longmapsto \widetilde{x_i} = \frac{1}{\sqrt{k}} A x_i \in \mathbb{R}^k,$$

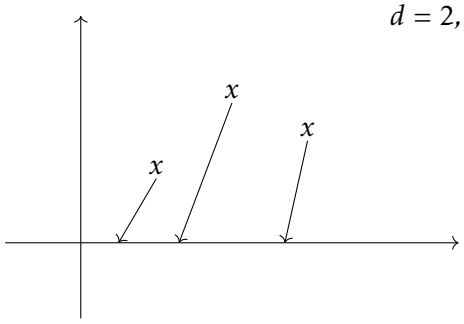
$$q \in \mathbb{R}^d \longmapsto \widetilde{q} = \frac{1}{\sqrt{k}} A q \in \mathbb{R}^n.$$

4. And we hope that $\|\widetilde{x_i} - \widetilde{q}\|_2 \approx \|x_i - q\|_2$.

Our major concern is to keep track of the correct nearest neighbors otherwise if we project n items to zero (i.e origin in a space), then it provides no useful information. Meaning lets suppose we do a projection transformation and as a result construct a zero-matrix and if all the projected points direct to zero then the information has no gain. Inference here is that for the points in the original space which are close must be close after the projection is made and if far then the distance shall be maintained.
Thus, mathematically speaking the projection must preserve the distance, i.e.,

$$\forall x_i, x_j : \left\|x_i - x_j\right\| \approx \left\|\widetilde{x_i} - \widetilde{x_j}\right\|$$

$$\begin{array}{cccc} x_1 & \cdots & x_n & \mathbb{R}^d \\ & \downarrow A & & \downarrow A \\ \widetilde{x_1} & \cdots & \widetilde{x_n} & \mathbb{R}^k \end{array}$$



Focusing on the k factor in the projection in step-3, consider now the following:

$$\widetilde{x_i} = A \cdot x_i \Longrightarrow \forall i, j \in \{1, 2, ..., n\} : \left\|\widetilde{x_i} - \widetilde{x_j}\right\|_2^2 = \left\|Ax_i - Ax_j\right\|_2^2 = \left\|A\left(x_i - x_j\right)\right\|_2^2.$$

Hope/goal:

$$\left\|A\left(x_i - x_j\right)\right\|_2^2 \approx \left\|x_i - x_j\right\|_2^2 \Longrightarrow \forall x \in \mathbb{R}^d : \|Ax\|_2^2 \approx \|x\|_2^2 \text{(with probability 0.99)}.$$

The main idea of the Chebyshev's and Hoeffding's inequalities is that once we get some estimate of the expectation then we can bound the actual outcome with a high probability *i.e.*, the actual outcome should be around the expectation (there is some sort of concentration around the mean in the distribution).
In this way, our goal is

$$\mathrm{E}\left[\|Ax\|_2^2\right] = \|x\|_2^2.$$

Some useful properties:

1. Norm is euclidean, so that
$$\|v\|^2 = v^T v. \tag{1}$$

2.
$$\mathrm{E}\left[x^T A^T A x\right] = \mathrm{E}\left[\sum_{i,j} f(A)\, x_i x_j\right] = \sum_{i,j} \mathrm{E}\left[f(A)\, x_i x_j\right] = \sum_{i,j} \mathrm{E}\left[f(A)\right] x_i x_j = x^T \mathrm{E}\left[A^T A\right] x \tag{2}$$

Then

$$\mathrm{E}\left[\|Ax\|_2^2\right] \overset{(1)}{=} \mathrm{E}\left[(Ax)^T Ax\right] = \mathrm{E}\left[x^T A^T A x\right] \overset{(2)}{=} x^T \mathrm{E}\left[A^T A\right] x.$$

Lecture continuation

Consider now the term $\mathrm{E}\left[A^T A\right]$. Our goal is to show, that

$$\mathrm{E}\left[A^T A\right] = k \cdot I = \begin{bmatrix} k & 0 & \cdots & 0 \\ 0 & k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k \end{bmatrix}$$

Proof.

$$A \in \mathbb{R}^{k \times d}, A^T \in \mathbb{R}^{d \times k} \Longrightarrow A^T A \in \mathbb{R}^{d \times d},$$

$$A = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \overrightarrow{a_1} & \overrightarrow{a_2} & \cdots & \overrightarrow{a_d} \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}, A^T = \begin{bmatrix} \leftarrow & \overrightarrow{a_1}^T & \rightarrow \\ \leftarrow & \overrightarrow{a_2}^T & \rightarrow \\ & \vdots & \\ \leftarrow & \overrightarrow{a_d}^T & \rightarrow \end{bmatrix}, A^T A = \begin{bmatrix} a_1^T \\ \vdots \\ a_d^T \end{bmatrix} \cdot [a_1, ..., a_d]$$

$$\Longrightarrow A^T A = \begin{pmatrix} a_1^T \cdot a_1 & \cdots & a_1^T \cdot a_d \\ \vdots & \ddots & \vdots \\ a_d^T \cdot a_1 & \cdots & a_d^T \cdot a_d \end{pmatrix} \in \mathbb{R}^{d \times d}, \text{ where } \forall i, j \in \{1, 2, ..., d\} : a_i^T \cdot a_j \in \mathbb{R}$$

$$\Longrightarrow \mathrm{E}\left[A^T A\right] = \begin{pmatrix} \mathrm{E}\left[a_1^T \cdot a_1\right] & \cdots & \mathrm{E}\left[a_1^T \cdot a_d\right] \\ \vdots & \ddots & \vdots \\ \mathrm{E}\left[a_d^T \cdot a_1\right] & \cdots & \mathrm{E}\left[a_d^T \cdot a_d\right] \end{pmatrix}.$$

Consider now an expectation $\mathrm{E}\left[a_i^T \cdot a_j\right]$ of an arbitrary element $a_i^T \cdot a_j$ of the last matrix:

Case 1. $i \neq j$:

$$\because \text{ values taken from Gaussian dist. } a_i \text{ and } a_j \text{ are independent}$$

$$\Longrightarrow \mathrm{E}\left[a_i^T \cdot a_j\right] = \mathrm{E}\left[a_i^T\right] \cdot \mathrm{E}\left[a_j\right] = \overrightarrow{0}^T \cdot \overrightarrow{0} = 0.$$

Case 2. $i = j$:

Notice first that

$$\forall i \in \{1, 2, ..., d\}, s \in \{1, 2, ..., k\} : \mathrm{E}\left[a_{i,s}\right] = 0, \mathrm{Var}\left[a_{i,s}\right] = \mathrm{E}\left[a_{i,s}^2\right] - \left(\mathrm{E}\left[a_{i,s}\right]\right)^2 = 1 \Longrightarrow \mathrm{E}\left[a_{i,s}^2\right] = 1. \quad (3)$$

$$\Longrightarrow \mathrm{E}\left[a_i^T \cdot a_i\right] = \mathrm{E}\left[\|a_i\|^2\right] = \mathrm{E}\left[\sum_{s=1}^{k} a_{i,s}^2\right] = \sum_{s=1}^{k} \mathrm{E}\left[a_{i,s}^2\right] \overset{(3)}{=} \sum_{s=1}^{k} 1 = k.$$

Hence, we have

$$\forall i, j \in \{1, 2, ..., d\} : \mathrm{E}\left[a_i^T \cdot a_j\right] = \begin{cases} 0, & i \neq j \\ k, & i = j \end{cases}$$

$$\Longrightarrow \mathrm{E}\left[A^T A\right] = \begin{bmatrix} k & 0 & \cdots & 0 \\ 0 & k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k \end{bmatrix} = k \cdot I \quad \square$$

Therefore,

$$\mathrm{E}\left[\|Ax\|^2\right] = x^T \mathrm{E}\left[A^T A\right] x = x^T \cdot (kI) \cdot x = k x^T x = k \|x\|^2,$$

hence,

$$A^{'} = \frac{1}{\sqrt{k}} A \Longrightarrow \mathrm{E}\left[\left\|A^{'} x\right\|^2\right] = \|x\|^2,$$

and then

$$\begin{aligned}
\widetilde{x_i} = \frac{1}{\sqrt{k}} A x_i \Longrightarrow \forall i, j \in \{1, 2, ..., n\} : \mathrm{E}\left[\left\|\widetilde{x_i} - \widetilde{x_j}\right\|_2^2\right] &= \mathrm{E}\left[\left\|\frac{1}{\sqrt{k}} A x_i - \frac{1}{\sqrt{k}} A x_j\right\|_2^2\right] \\
&= \mathrm{E}\left[\left\|\frac{1}{\sqrt{k}} A\left(x_i - x_j\right)\right\|_2^2\right] \\
&= \left\|x_i - x_j\right\|_2^2
\end{aligned}$$

Remark: By the above proof, we can get that the expectation of l_2 norm of projected vector is the same as l_2 norm of the original vector. Thus following formula should be = rather than $\approx$. While our target is to make sure two l_2 norms are closed $\left(\left\|A\left(x_i - x_j\right)\right\|_2^2 \approx \left\|x_i - x_j\right\|_2^2\right)$. Then we can apply the Markov Inequality to get the JL lemma.
Hence, as a conclusion, for the expectation for any point in the space, the projection matrix will preserve the norm

$$\mathrm{E}\left[\left\|A^{'}\left(x_i - x_j\right)\right\|_2^2\right] \approx \left\|x_i - x_j\right\|_2^2$$

Johnson-Lindenstrauss Lemma. For any $\epsilon \in (0, 1)$ and any integer $d > 0$ let

$$k \geq \frac{24}{3\epsilon^2 - 2\epsilon^3} \log n.$$

Then for any $x_i, x_j \in \{x_1, x_2, ..., x_n\}$ with high probability

$$\begin{aligned}
(1 - \epsilon)\left\|x_i - x_j\right\|_2^2 &\leq \left\|f\left(x_i\right) - f\left(x_j\right)\right\|_2^2 \\
&\leq (1 + \epsilon)\left\|x_i - x_j\right\|_2^2
\end{aligned}$$

where the new data representation is givenby

$$f(x) = \frac{1}{\sqrt{k}} A x, A \in \mathbb{R}^{k \times d}$$

and each entry of A is i.i.d. $N(0, 1)$. Also usually $\epsilon \approx 10^{-6}$.
This lemma simply states that how a set of points in a high-dimensional space can be embedded into a space of much much lower dimension in order to preserve the distances.

Union bound

It is applied when you need to show that the probability of union of some events is less than some value. It can be extended to obtain the upper and the lower bounds on the probability of union events. Let $E_1, E_2, ..., E_n$ be the events. Suppose that the following condition is satisfied:

$$\begin{cases}
\mathrm{P}\left(E_1\right) &= 1 - \delta_1 \\
\mathrm{P}\left(E_2\right) &= 1 - \delta_2 \\
&\vdots \\
\mathrm{P}\left(E_n\right) &= 1 - \delta_n
\end{cases}$$

Then

$$\mathrm{P}(E_1 \cap E_2 \cap ... \cap E_n) \geq 1 - \sum_{i=1}^{n} \delta_i$$

↑

Union bound

Calculus review

All functions in the course are continuous. The subset of the continuous functions is a set of functions that have a 1st order derivative (i.e. gradient).

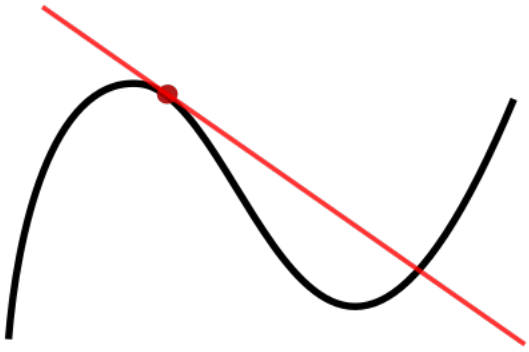


Figure 1: Tangent to a curve that can be found using a derivative

- Common functions: x^t, e^x, $\log x$, etc.
- Combined functions: $f+g$, $f \cdot g$, f/g, $f(g)$, etc.

$$\Longrightarrow (f+g)' = f' + g', (f \cdot g)' = f'g + fg', \left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}, (f(g))' = f'(g) \cdot g'$$

Property 1. $f(x)$ is increasing on $(a,b) \Longleftrightarrow f'(x) \geq 0$ on (a,b). Proof: mean-value theorem.
Property 2. x is local minimum/maximum $\Longrightarrow f'(x) = 0$. Converse is false! Example: stationary point $x = 0$ for the function $f(x) = x^3$, for which $f'(x) = 0$, but $x = 0$ is neither minimum nor maximum (Figure 2).
Definition. Let $x = (x_1, x_2, ..., x_d)$, and $f = f(x) \in \mathbb{R}$ ($f : \mathbb{R}^d \to \mathbb{R}$). Then the gradient $\nabla f \in \mathbb{R}^d$ of f by x is

$$\nabla f(x) \triangleq \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_d}\right).$$

Example. Let $x \in R^2$ and $f(x) = x_1^2 + x_1 x_2$. $\nabla f - ?$

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}\left[x_1^2 + x_1 x_2\right] = \frac{\partial}{\partial x_1}\left[x_1^2\right] + \frac{\partial}{\partial x_1}\left[x_1 x_2\right] = 2x_1 + x_2,$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}\left[x_1^2 + x_1 x_2\right] = \frac{\partial}{\partial x_2}\left[x_1^2\right] + \frac{\partial}{\partial x_2}\left[x_1 x_2\right] = x_1.$$

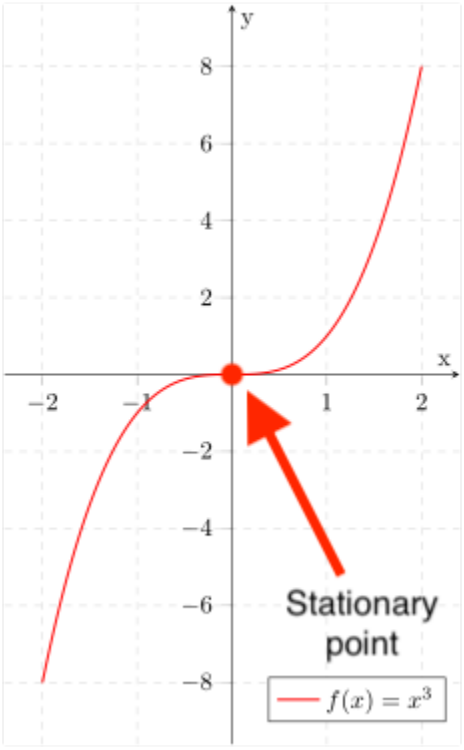


Figure 2: Example of stationary point

Hence,

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right) = (2x_1 + x_2, x_1).$$

We note below some useful gradient calculation

- $\|v\|_2^2 = v^T v \Longrightarrow \nabla_v \|v\|_2^2 = 2v$,
- $\nabla_v Mv = M^T$,
- $\nabla_v v^T Mv = (M + M^T)v$.

ERM

Example. Let $x \in R^d$ and $f(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_d^2}$. ∇f – ?

$$\forall i \in \{1, 2, ..., d\} : f(x) = \sqrt{x_i^2 + c}, \text{ where } c = \sum_{\substack{j=1 \\ j \neq i}}^{d} x_j^2$$

$$\Longrightarrow \frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\sqrt{x_i^2 + c}\right] = \frac{\frac{\partial}{\partial x_i}\left[x_i^2 + c\right]}{2\sqrt{x_i^2 + c}} = \frac{2x_i}{2\sqrt{x_i^2 + c}} = \frac{x_i}{\sqrt{x_i^2 + c}} = \frac{x_i}{f(x)} = \frac{x_i}{\|x\|}.$$

Hence,

$$f(x) = \|x\| \Longrightarrow \nabla f = \left[\frac{x_1}{\|x\|}, \frac{x_2}{\|x\|}, ..., \frac{x_n}{\|x\|}\right] = \frac{x}{\|x\|}$$

GD, SGD

CS 541: [Artificial Intelligence]

Instructor: Jie Shen
Scribe: Hanao Li and David Li

Lecture 4
September 21, 2021

1 Learning Theory

Theorem: Considering classification problems, given pairs of data $\{x_1, y_1\}, ..., \{x_n, y_n\}$, if they meet the following two conditions:

1. n = $\frac{d}{\epsilon^2}$ where d is the dimension of x and some error term ϵ.
2. all y's are correct, then $\exists$ an efficient algorithm, i.e, Support Vector Machine (SVM), that can learn the true model h^*.

Definition: For ground truth model h^*, given pairs of data, for any $i \in [1, n],\ y_i = h^*(x_i)$. If the data comes from a distribution D, then for any point $(x, y) \sim D,\ y = h^*(x)$.

Example: Given ground truth model h^*, our data x can be transformed to sign(w^*x). Any points that are above the w^* will be considered as positive samples and any points that are below the w^* will be considered as negative samples. This is called as half-space. Our goal is to find $\hat{w}$ that satisfies the following equation: $||\hat{w} - w^*|| \leq \epsilon$. More reference about this PAC learning framework can be found from Valiant (1984).

Example: Given patient data x_i, we have different features such as blood pressure, age, etc. We would like to predict y whether they are healthy or not. Assuming we have a model: $y_i = wx_i$ where sample size $n = 100$, our goal becomes to obtain $\hat{w}$ such that $\hat{w}$ can minimize the squared loss $(y_i - \hat{w}x_i)^2$.

We evaluate whether the estimated parameter w is good or not by calculating the testing error. Since our loss function could lead to a small training error, we would hope the model could achieve a small testing error as well. This is only possible when the training and testing data comes from the same distribution.

We claim: $err_{test} \leq err_{train} + \sqrt{\frac{d}{n}}$. Given this function, we can control the training error and gather more training samples to lower the testing error.

2 Learning with Sparsity

2.1 Linear Regression

Example: Given data x with different features such as blood pressure, age, height, color of shoes and whether use macbook, we would like to predict y, this person's weight with sample size $n = 50$. We will use squared error as our loss function $(y_i - wx_i)^2$.

We might get a dense weight such as $w = (0.1, -1, 1, 0.5, 10)$. We expect weights w to have accurate prediction and we hope model can explain data. Imaging having $w = (0.1, 1, 0, 0, 0)$ with $x_1 = (90, 20, 5, 2, 1), x_2 = (90, 20, 20, 4, 0)$, we obtain the same prediction $\hat{y} = 0.1 * 90 + 1 * 20 + 0 * 5 + 0 * 2 + 0 = 29$. And we can say only blood pressure and age affects the weight.

2.2 Prediction Functions

Example: Given x and we could do prediction by inner prodcut using w and x into $f(w, x)$. If w is sparse, then it can explain the data.

Linear Regression: $f(w, x) = wx$

Logistic Regression: $f(w, x) = \frac{1}{1+e^{-wx}}$

Classification: $f(w, x) = \text{sign}(wx)$

2.3 K-sparse

Definition: Given $w \in R^d$, it is k-sparse if number of non-zero elements in $w \leq k$.

Example: Given $w \in R^d$ where $d = 5$, we may call $(1, 0, 0, 0, 0)$ as 1-sparse. It can also be called as 2-sparse, 3-sparse, ..., 5-sparse, but 1-sparse is preferred since it is an upper bound and it is more accurate.

Example: Given $w \in R^d$ where $d = 5$, we may call $(-1, 1, 2, 0, 1)$ as 4-sparse (preferred), or 5-sparse.

Formulation of K-sparse constraint:

$$\begin{aligned} &\min_{w \in R^d} F(w) = \sum_{i=1}^{n} (y_i - wx_i)^2 \\ &\text{s.t.} \quad ||w||_0 \leq k \end{aligned} \tag{1}$$

where 0 in $||w||_0$ means the number of non-zero elements.

In general, this is a NP-hard problem with complexity $O(e^d)$ because we need to enumerate all subset having k entries.

3 Convex Relaxation

3.1 Non-Convex Problem

Example: Given $d = 2$, $k = 1$ and $||w||_0 \leq k$. This means it corresponds to the coordinates of the two x-axis: $\{w : ||w||_0 \leq 1\}$. We want to relax this non-convex set to a convex problem. We first review what is a convex set.

3.2 Convex Set

Definition: Set C is a convex set if the line segment between any two points in C lies in C, i.e, if for any $u, v \in C$ and any $\lambda \in [0, 1]$, we have $(1 - \lambda)u + \lambda v \in C$. When $\lambda = 0$, it is essentially u and when $\lambda = 1$, we have v. See Figure 1 and Figure 2 as an illustration.

3.3 Epigraph

Definition: For any function $F(w)$, epigraph is the region above the curve.

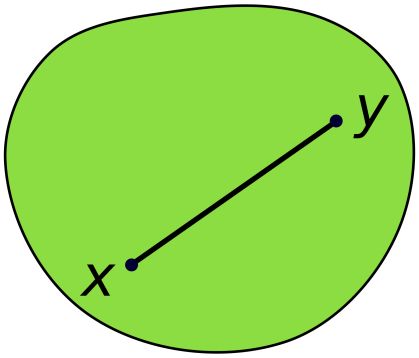


Figure 1: Image from Wikipedia, this is a convex set.

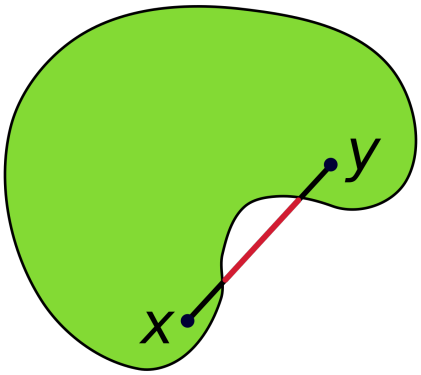


Figure 2: Image from Wikipedia, this is not a convex set.

Theorem: Function $F(w)$ is a convex function if its epigraph is a convex set.

Example: Given $F(w)$, we want to minimize it such that $w \in C$. If $F(w)$ is convex and C is convex as well. It will be easy to solve and there are infinite solvers to find global optimum. For example: Cutting plane, Interior Point, Gradient Descent, etc.

3.4 Relaxation

Formulation: Given a non-convex set N, we hope to get a convex set C. We pick any two points $u, v \in N$ and do convex combination for all points in N: $(1-\lambda)u + \lambda v \rightarrow^{add} C$ for $\lambda \in [0, 1]$. Then we call C as convex hull of N (minimum convex set of N).

Theorem: If C' is convex $\supset N$, then we must have $C \subseteq C'$.

Example: Given R^2, 1-sparse non-convex set $N : \{w : ||w||_0 \leq 1, ||w|| \leq 1\}$ and convex set $C : \{w : ||w||_1 \leq 1\}$, we know C is forms a diamond shape. That is bounded by $y = -x + 1, y = x + 1, y = -x - 1, y = x - 1$. We relax non-convex set into convex set and achieve sparsity.

Example: Given $F(w) = (5 - w_1)^2 + (4 - w_2)^2$, we want to minimize it such that $||w||_{\leq}1$.

This constraint implies $|w_1| + |w_2| \leq 1$. Where we can obtain $y_1 = 5, x_1 = (1, 0)$ or $y_2 = 4, x_2 = (0, 1)$.

Normally, the optimal solution can be achieved by taking the gradient, we have $\frac{\partial F}{\partial w_1} = 2(w_1 - 5) = 0 \rightarrow w_1 = 5$ and $\frac{\partial F}{\partial w_2} = 2(w_2 - 4) = 0 \rightarrow w_2 = 4$. But it does not satisfy the constraint. See left side of Figure 3 where we gradually increase the radius around the center unitl it hits the constraint. In this case, our w is sparse and it is 1-sparse.

Example: Given the same $F(w)$, we want to minimize it such that $||w|| \leq 1$. This time we have a L2 constraint and our weights will be dense. See right side of Figure 3.

Figure 3 shows that the radius of the graph is gradually being increased in order to solve the minimization equation. On the left side of Figure 3, the place where the set $||w||_0$ and function F(w) meet, the intersection occurs on the y-axis maxima. Because of this, we can say that F(w) is 1-sparse. The F(w) on the right side, on the other hand, is dense.

When trying to minimize a function F(w) where $||w||_0$ is a convex set, extending the radius of F(w) in various increments could lead to multiple solutions.

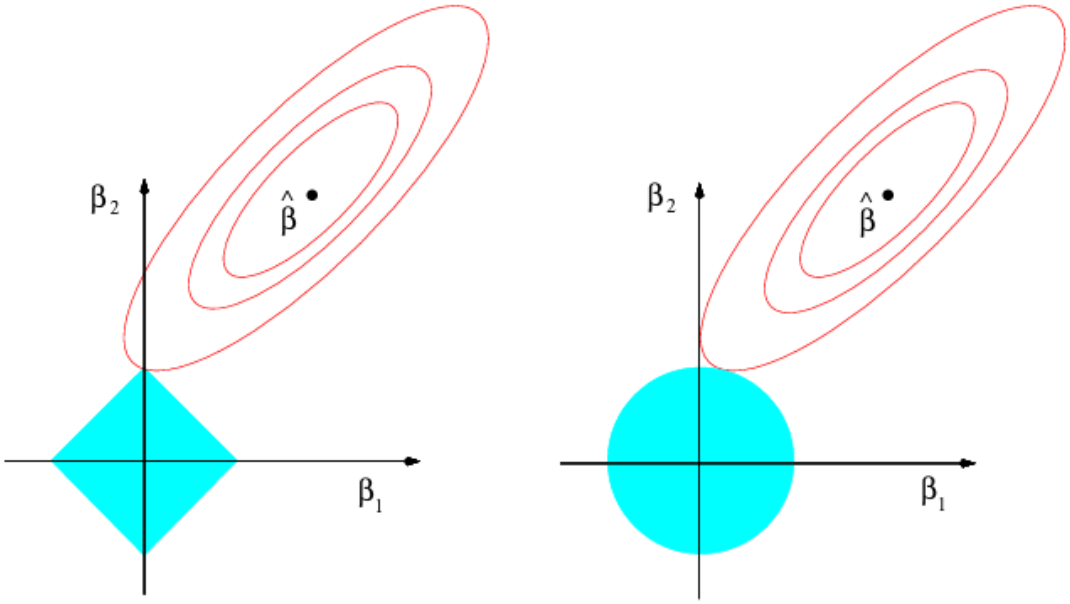


Figure 3: Figure from Elements of Statistical Learning book, page 90. Left figure shows $||w||_1 \leq 1$ where we achieve sparsity for our weight w and right figure shows $||w||_2 \leq 1$ where we achieve dense representation of our weight w. This is an example of gradually increasing radius to find minimums with added constraint.

4 Solving for the True Model

4.1 Natural Method

Assume that we know the true model w^* given by $y_n = w^* x_n$. In a convex set given by the relation $y = wx$, where $w \in R^d$, dataset x and dataset y can lead to solving for w^*.

Natural Method of w*: We can recover the true model using $y_1 = wx_1, ..., y_n = wx_n$:

$$w = (x^T x)^{-1} x^T y \tag{2}$$

Furthermore, the cost of computing is $O(d)$. Additionally, it also requires that in w, $n \geq d$.

4.2 Brute Force

Formulation:

$$\begin{aligned} &\min_{w \in R^d} F(w) \\ &\text{s.t. } ||w||_0 \leq k \end{aligned} \tag{3}$$

Definition: For each index set, the cardinality is $\leq k$: the min F(w) is found by looking into every possible F(w) and its corresponding k-sparsity. For example: $||w||_0 = 1 \rightarrow (\frac{1}{d}), ..., ||w||_0 = k \rightarrow (\frac{k}{d})$.

Example: Given $F(w) = (w_1 - 1)^2 + w_2^2 + (w_3 - 5)^2$ such that $||w||_0 \leq 1$. We start with trying index 1: $w_2 = 0, w_3 = 0$, then we have $\min F(w) = F(w_1) = (w_1 - 1)^2 \rightarrow w_1 = 1, F(w_1 = 1) = 0$. Then we try index 2: $w_1 = 0, w_3 = 0$ to obtain next set of results and so on. After all the tries, we then change our constraint to $||w||_0 \leq 2$ and doing the same procedures again to enumerate all possible subsets.

The cost of the algorithm is $\Theta(d)$ originally. This algorithm's cost will be $\Theta(k)$, if we know the position of non-zero elements. But in practice, we do not know the position of non-zero elements, and the cost will be $\Theta(k \log d)$. Given $d = 1$ million and $k = 100$, our n becomes $100 * 6 = 600$.

CS 541: Artificial Intelligence

Instructor: Jie Shen
Scribe: Joshua Ronai and Yi Rong

Lecture 5
September 28, 2021

Contents

1 Sparsity

1.1 Review

The idea of sparsity is limiting the amount of nonzero features in a model by setting $||w||_0 \leq k$. We can equivalently call this $||w||_1 \leq \sqrt{k}$.
Our natural approach is to optimize a model by minimizing its loss through some function $F(w)$ under the constraint that we maintain some level of sparsity. The following optimization would then follow:

$$\begin{cases} \min F(w) \\ \text{s.t. } ||w||_0 \leq k \end{cases}.$$

To make the problem less computationally intensive, we want to make sure that our constraint set is *convex*. We can ensure that our set of parameters is convex by constructing its *convex hull*. This is done by taking any two points in our nonconvex set $u, v \in N$ and putting all of their convex combinations $\lambda u + (1 - \lambda)v \ \forall \lambda \in [0, 1]$ in the convex set C.

1.1.1 An Example of Convexity

Consider two arbitrary points u, v on the x_1x_2-plane. Let these points be our initial set. Immediately, we can recognize that the set is not convex because any convex combination between these two of the points where $\lambda \neq 0, 1$ will create a new point. To make the set convex, we can make all the convex combinations $\lambda u + (1 - \lambda)v \ \forall \lambda \in [0, 1]$ between the two points. This creates a line between the two points, and is therefore a convex set:

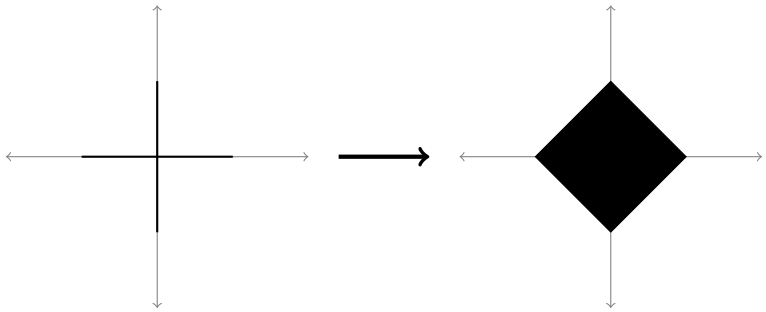
Initial Constraint Convex Hull

What if now instead of starting with two points, we start with three points u, v, w? Well, creating all the convex combinations between each of the points, we will get a triangle that is completely filled in, which will be convex.

Similarly, for four points, the convex hull would be some 4-sided polygon.

1.2 Applying Convexity for Sparsity in 2-Dimensions

Suppose we are given $\mathbb{R}^2$ and our constraint is that the features should be 1-sparse. Then, our non-convex set is just the x_1 or x_2 axes, or $\{w : \ ||w||_0 \leq 1\}$. We can limit this to just the points where the magnitude of the feature vector is normalized, giving us $\{w : \ ||w||_0 \leq 1, ||w||_2 \leq 1\}$. We can then create the convex hull as a diamond. We can visualize this transformation below:



Initial Constraint Convex Hull

1.3 Applying Convexity for Sparsity in d-Dimensions

Suppose we are now in $\mathbb{R}^d$. We still limit $||w||_0 \leq k$ and $||w||_2 \leq 1$. We then also limit $||w||_1 \leq \sqrt{||w||_0||w||_2} \leq \sqrt{k}$.

We can prove this inequality. To start, we already know from the constraint that $||w||_1 \leq \sqrt{k}$. Additionally, given the above inequalities for $||w||_0, ||w||_2$, we can easily see that $||w||_0||w||_2 \leq k$, or equivalently $\sqrt{||w||_0||w||_2} \leq \sqrt{k}$.

We can think of $||w||_1$ as an inner product. We do this by expanding its definition to

$$||w||_1 = |w^1| + |w^2| + \cdots + |w^d| = \langle w, \text{sign}(w) \rangle.$$

This value is always bounded by $||w||_2||\text{sign}(w)||_2$. We define $||\text{sign}(w)||_2^2$ as the number of nonzero features in w, which in this case is $||w||_0$, or k. This, along with $\max(||w||_2) = 1$, means that the upper-bound for $||w||_1$ is

$$||w||_2||\text{sign}(w)||_2 = 1 * \sqrt{(k)} = \sqrt{k}.$$

2 Taking Advantage of Sparsity and Convexity

There are many ways to solve an optimization problem once we know its constraints are convex.

2.1 Hard Threshold

One very important algorithm is called **Hard Threshold**.

Given some $w \in \mathbb{R}^d$, we define $H_k(w)$ as k-sparse. We do this by

1. Finding the index of the k-largest magnitudes of elements in w and putting those indices i in a set S.
2. For any $i \notin S$, we set w^i=0.

This ensures that $|S| = k \rightarrow k$-sparse vector.

2.1.1 Some Examples

Say we have an initial set as $(5, -10, 0)$ and we want to make it 1-sparse. Then $H_k = (0, 10, 0)$ because $|5| < |-10|$.

Say now that we have $(1, 2, 10, -5)$ and we want it to be 2-sparse. The two largest magntiudes come from 10 and -5. So, we get $H_k = (0, 0, 10, -5)$.

2.1.2 Properties

$H_k(w)$ is guaranteed to be k-sparse.

In addition, the k-sparse vector closest to w will be $H_k(w)$. This means that for all vectors v that are k-sparse,

$$||v - w|| \geq ||H_k(w) - w||,$$

so $H_k(w)$ is our best approximation.

2.2 Gradient Descent

But how can we implement Hard Threshold in practice? The answer is by combining it with gradient descent to boost our efficiency.

We begin by setting $w_0 = \vec{0}$. Then, for each $t = 1, \ldots, T$, we update the feature vector with

$$b_t = w_{t-1} - \nu \nabla F(w_{t-1}),$$

where ν is the learning rate. We can then use hard threshold to get

$$w_t = H_k(b_t).$$

We know that $F(b_t) \leq F(w_{t-1})$, which is guaranteed by gradient descent due to the learning rate along with hard threshold making some features zero.

For training purposes, this is feasible if $F(w_t) \approx F(b_t)$. This approximation is only sound because of the Hard Threshold, since $H_k(w)$ is the best k-sparse approximation.

In addition, if our data is Gaussian we can find the global optimum with gradient descent.

2.2.1 Applying Other Constraints

We can also use Gradient Descent when the constraint is $||w||_1 \leq \sqrt{k}$, or projecting onto the 1-norm ball. Then we get that $w_t = \text{proj}_{<1}(b_t)$, where b_t is the same value gained from gradient descent, meaning we need to minimize $||w - b_t||$ with that constraint. Though this runs in *polynomial time*, it is not as efficient as hard threshold.

We could also use $||w||_2 \leq r$, and update $w_t = \frac{b_t}{||b_t||} r$. This is also computable in linear time. Geometrically, this is a circle with radius r, and given some vector b_t, we are trying to find the point on the circle closest to b_t.

3 Principal Component Analysis

We now focus on other methods to reduce computation and dimensionality. Instead of breaking our data down after collecting our feature values, we can break a data matrix itself down with a technique called **Principal Component Analysis**, or PCA.

3.1 Introduction

Consider a data matrix

$$M = \begin{bmatrix} 5 & 4 & 3 & 0 & 1 \\ 5 & 4 & 3 & 0 & 1 \end{bmatrix}.$$

We let each column be a point, meaning we have 5 points. So we are in $\mathbb{R}^2$ where $n = 5$. However, all of these points are on the line $y = x$. This means that we can project them onto the x-axis, making our new matrix

$$M' = \begin{bmatrix} 5 & 4 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

which reduces our dimension from 2 to 1.

Now, we make our a new data matrix

$$M = \begin{bmatrix} 5 + 10^{-3} & 4 + 10^{-6} & \dots \\ 5 - 10^{-2} & 4 - 10^{-5} & \dots \end{bmatrix}.$$

Now, the points are no longer on a straight line. So maybe projecting straight onto the x-axis or y-axis is not the best way to get rid of the random noise.

If we want to be able to reduce the dimensionality of this new data matrix, we need to consider what we expect from our data. When we collect data, we typically assume that there must be some underlying structure to it. Especially, if we have a lot of data points, we would expect some similarity amongst various subgroups of the data. Maybe taking advantage of this underlying structure can help us reduce our computation.

But how do we define mathematically that one point is similar to another? We can define two points X_1 and X_2 being similar as being *linearly dependent* to each other. To understand this, we first review some Linear Algebra.

3.2 Linear Algebra Review

3.2.1 Subspaces

One of the most important notions in Linear Algebra is the notion of a **subspace**. For some subspace $V \in \mathbb{R}^d$:

1. $0 \in V$
2. If $u, v \in V$, then $au + bv \in V$, where $a, b \in \mathbb{R}$.

Let's try building a subspace. Let's start with $V = \{0\}$. If we then add 1 into it to get $V = \{0, 1\}$, then we would also need to include every linear combination of 0 and 1 in V. But since one of the elements of the subspace is 0, this means that the new subspace is $\{a\}$, where $a \in \mathbb{R}$.

We can also define $e_1 = (1, 0, 0 \dots)^T$, $e_2 = (0, 1, 0, \dots)^T$, and so on. Given that, we can create a subspace $V_1 = \{ae_1 + be_2\}$. Moreover, we can generalize this in $\mathbb{R}^d$ as $V_2 = (\sum a_i e_i)$. We can then say the *rank* of the V_1 is 2 because it spans 2 dimensions, and we can similarly say that the rank of V_2 is d. However, we do not always need to have the rank and number of dimensions match. There can be many dimensions but the rank can be very small. These values become different once there is linear dependence.

3.2.2 Linear Independence and Dependence

We say that vectors $v_1, \dots, v_n$ are **linearly independent** if and only if

$$a_1 v_1 + \dots + a_d v_d = 0$$

implies that the only solution is $a_1 = \dots = a_n = 0$.
If there is any other solution, we say that the vectors are **linearly dependent**.
For example, $v_1 = (1, 0)^T$ and $v_2 = (0, 1)^T$ are linearly independent, because

$$a_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$$

can only be solved when $a_1 = a_2 = 0$.

For another example, $v_1 = (1, 1)^T$ and $v_2 = (0, 1)^T$ are linearly independent, because

$$a_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$$

can only be solved when $a_1 = a_2 = 0$.

However, $v_1 = (1, 1)^T$ and $v_2 = (2, 2)^T$ are linearly dependent, because

$$a_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 0 \rightarrow a_1 = -2, a_2 = 1.$$

3.3 Clean Data

Given the previous notions, our underlying assumption about the structure of data is that a *clean* data matrix is a *low rank* data matrix.

A data matrix

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

has a rank 2, but a data matrix

$$M = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

has rank 1, despite both having 2 dimensions. So we can then see that linear dependence among points is directly related to a matrix having a lower rank.

This means that if we want to get rid of noise in a data matrix, we should aim to lower its rank.

We now return back to the data matrix

$$M = \begin{bmatrix} 5 + 10^{-3} & 4 + 10^{-6} & \dots \\ 5 - 10^{-2} & 4 - 10^{-5} & \dots \end{bmatrix}$$

Our hope would be that after the cleaning we get our matrix to

$$X = \begin{bmatrix} 5 & 4 & \dots \\ 5 & 4 & \dots \end{bmatrix}.$$

In general, this means that we aim to convert out data matrix M to a clean matrix X such that $M \approx X$. We can set this up as an optimization problem:

$$\begin{cases} \min_X ||X - M||_F \\ \text{s.t. } \text{rank}(X) \leq k \end{cases},$$

where

$$M_F = \sqrt{\sum_{i,j} m_{ij}^2} \approx ||v||_2.$$

This is the natural formulation.

3.3.1 Solving a Similar Optimization Problem

To solve that optimization problem, we can try solving a similar problem in a new way. Let's assume that $M, X \in \mathbb{R}^d$. We can restate this problem as the optimization problem from sparsity:

$$\begin{cases} \min ||X - M||_2 \\ \text{s.t. } ||X||_0 \leq k. \end{cases}$$

To solve this, we can use *Hard Threshold*. We would then conclude that $X = H_k(M)$ which would run in $O(k \log(d))$. M would be a vector of elements $(m_1, \dots, m_d)$. This means that we can rewrite this as $M = \sum_{i=1}^{d} m_i e_i$, where e_i is the basis vector for the i^{th} dimension.

We can decompose any matrix into this form as long as the vectors e_i form a basis. We can then select the k largest coefficients among m_i, and turn the rest into 0. This is the case for vectors.

3.3.2 Solving the Matrix Case

Armed with what we just learned, we now return to our more complex case.

Our first step would be to decompose

$$M = \sum_{i=1}^{T} \alpha_i C_i,$$

where each C_i is a matrix orthogonal to the other C_j matrices. The α_i are called **singular values.** The C_i are basis matrices with the same dimensions $d \times n$ as M.

We can say that there is some function that turns a matrix M into its orthogonal matrices and their coefficients. This function is *singular value decomposition*, or SVD.

From here, we can find the k largest $\alpha_i \in \mathbb{R}^+$, and turn the rest into 0. We know they will be positive because it is guaranteed from SVD.

3.4 Singular Value Decomposition

As previously stated, the function used to decompose M is **singular value decomposition**. We break down M into $M = U\Sigma V$, where $U \in \mathbb{R}^{d\times d}$, $\Sigma \in \mathbb{R}^{d\times n}$, and $V \in \mathbb{R}^{n\times n}$, n is the number of data points, and d is the number of features.

3.4.1 Properties of the Decomposition

The first thing is that $U^TU = UU^T = I$ and $V^TV = VV^T = I$, meaning that U, V are both *orthogonal matrices.*

The second thing is that the diagonal values of Σ are the α_i, with every other element in Σ being 0.

It's run-time is $\min\{nd^2, n^2d\}$. This decomposition is unique.

What if now we split the matrix product as $M = U(\Sigma V)$, and call $C = \Sigma V$? We can then think of U as a set of column vectors $u_1, \ldots, u_d$. We then figure that for each column we get

$$m_1 = \sum_{i=1}^{d} a_i u_i,\ m_2 = \sum_{i=1}^{d} a_i' u_i, \ldots$$

We can also think of V as a set of row vectors. This means that we can write our data matrix as

$$M = \sum_{i=1}^{d} \alpha_i u_i v_i,$$

which will result in a $d \times n$ matrix, which is what we want. We can also see for $i \neq j$ that $\langle u_i v_i, u_j v_j \rangle = 0$, because U, V are symmetric matrices. When $i = j$, the inner product is equal to the identity matrix I.

Task 1

Hoeffding's Inequalities [5]. Let $X_1, X_2, ..., X_n$ be independent bounded random variables such that $\forall i \in \{1, 2, ..., n\} : X_i \in [a_i, b_i]$ with probability 1. Let $S_n = \sum_{i=1}^{n} X_i$. Then for any $t > 0$, we have

$$\mathrm{P}(S_n - \mathrm{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right), \tag{1}$$

$$\mathrm{P}(\mathrm{E}[S_n] - S_n \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right), \tag{2}$$

$$\mathrm{P}(|S_n - \mathrm{E}[S_n]| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right). \tag{3}$$

1. Given X - one single image, let $n \in \mathbb{N} \setminus \{0\}$ be a number of workers that we hire to label this image. Let s_i be a random variable that represents a success of worker i ($i \in \{1, 2, ..., n\}$) on this task of the labelization. I.e. $s_i = 0$ if a label is not correct, and $s_i = 1$ if a label is correct.

According to the task conditions, $s_i = 1$ with probability $p \in [0.6, 1]$, and $s_i = 0$ with probability $(1 - p) \in [0, 0.4]$. Hence, s_i is the random variable with **Bernoulli distribution with success probability p** for any $i \in \{1, 2, ..., n\}$. According to the task conditions, we can assume that random variables $s_1, s_2, ..., s_n$ are independent and identically distributed (i.i.d). Thus the random variable $S_n = \sum_{i=1}^{n} s_i$ is distributed according to a binomial distribution with parameters n and p [4]: $S_n \sim \mathrm{Bi}(n, p)$.

At the same time, S_n is a total number of correct labels. Therefore, the task goal is to find a minimal n such that:

$$\mathrm{P}\left(S_n > \frac{n}{2}\right) \geq 0.99.$$

A condition $S_n > \frac{n}{2}$ means that the majority of labels is correct. Since $S_n \sim \mathrm{Bi}(n, p)$, $\mathrm{E}[S_n] = np$ according to the properties of the binomial distribution [1].

Hence, according to the Hoeffing's Inequality (2), since $s_1, s_2, ..., s_n$ are independent random variables such that $\forall i \in \{1, 2, ..., n\} : 0 \leq s_i \leq 1$ with probability 1, then

$$\forall t > 0 : \mathrm{P}(np - S_n \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(1 - 0)^2}\right) = \exp\left(-\frac{2t^2}{n}\right).$$

Thus, in particular when $t = n\left(p - \frac{1}{2}\right) > 0$ ($p \geq 0.6 > 0.5 \implies p - \frac{1}{2} > 0$, $n > 0$ by definition, then $n\left(p - \frac{1}{2}\right) > 0$) we have

$$\mathrm{P}\left(\cancel{np} - S_n \geq \cancel{np} - \frac{n}{2}\right) \leq \exp\left(-\frac{2\cancel{n}^{\cancel{2}}\left(p - \frac{1}{2}\right)^2}{\cancel{n}}\right) = e^{-2n\left(p - \frac{1}{2}\right)^2}.$$

At the same time

$$\mathrm{P}\left(\cancel{np} - S_n \geq \cancel{np} - \frac{n}{2}\right) = \mathrm{P}\left(-S_n \geq -\frac{n}{2}\right) = \mathrm{P}\left(S_n \leq \frac{n}{2}\right) = 1 - \mathrm{P}\left(S_n > \frac{n}{2}\right).$$

Hence

$$1 - \mathrm{P}\left(S_n > \frac{n}{2}\right) \leq e^{-2n\left(p - \frac{1}{2}\right)^2} \implies \mathrm{P}\left(S_n > \frac{n}{2}\right) \geq 1 - e^{-2n\left(p - \frac{1}{2}\right)^2}.$$

Therefore by finding the minimum n so that

$$1 - e^{-2n\left(p - \frac{1}{2}\right)^2} \geq 0.99,$$

we will find the minimum n so that

$$\mathrm{P}\left(S_n > \frac{n}{2}\right) \geq 0.99.$$

Consider the previous inequality in more details:

$$1 - e^{-2n\left(p-\frac{1}{2}\right)^2} \geq 0.99 \Longleftrightarrow e^{-2n\left(p-\frac{1}{2}\right)^2} \leq 0.01 \Longleftrightarrow e^{-2n\left(p-\frac{1}{2}\right)^2} \leq e^{\ln 0.01} \Longleftrightarrow -2n\left(p-\frac{1}{2}\right)^2 \leq \ln 0.01$$

$$\Longleftrightarrow -\cancel{2}n\frac{(2p-1)^2}{\cancel{4}2} = -\frac{1}{2}n(2p-1)^2 \leq \ln 0.01 \Longleftrightarrow n \geq -\frac{2\ln 0.01}{(2p-1)^2}$$

Thus, we have

$$\mathrm{P}\left(S_n > \frac{n}{2}\right) \geq 1 - e^{-2n\left(p-\frac{1}{2}\right)^2} \geq 0.99 \Longleftrightarrow n \geq -\frac{2\ln 0.01}{(2p-1)^2}$$

That means that the minimum integer n that satisfy this inequality can be found as

$$n = \left\lceil -\frac{2\ln 0.01}{(2p-1)^2} \right\rceil.$$

Examples for some values of p:

$$p = 0.6 \Longrightarrow n = \left\lceil -\frac{2\ln 0.01}{0.04} \right\rceil = 231,$$

$$p = 0.75 \Longrightarrow n = \left\lceil -\frac{2\ln 0.01}{0.25} \right\rceil = 37,$$

$$p = 0.9 \Longrightarrow n = \left\lceil -\frac{2\ln 0.01}{0.64} \right\rceil = 15.$$

Conclusion. To ensure that the majority of labels is correct with probability 0.99, we should choose the number of workers n according to the following rule:

$$n = \left\lceil -\frac{2\ln 0.01}{(2p-1)^2} \right\rceil.$$

2. Let now $X_1, X_2, ..., X_n$ - multiple images to label. Let s_i ($i \in \{1, 2, ..., n\}$) is a number of correct labels made by the worker i. Hence, $\forall i \in \{1, 2, ..., n\} : s_i \sim \mathrm{Bi}(m, p)$ where $p \in [0.6, 1]$. A total number of correct labels is thus $S_n = \sum_{i=1}^{n} s_i$. Therefore,

$$S_n \sim \mathrm{Bi}\left(\sum_{i=1}^{n} m, p\right) \sim \mathrm{Bi}(mn, p),$$

according to the properties of the binomial distribution [3]. Moreover, $\mathrm{E}[S_n] = mnp$ [1].

A total number of labels for m images made by n workers is equal to $m \cdot n$ (n labels for X_1, n labels for X_2, ..., n labels for X_m). Hence, *the majority of labels is correct* $\Longleftrightarrow S_n > \frac{mn}{2}$.

For this task we can also assume that $s_1, s_2, ..., s_n$ are independent and identically distributed. Furthermore, $\forall i \in \{1, 2, ..., n\} : 0 \leq s_i \leq m$. Then, according to the Hoeffding's inequality 2:

$$\forall t > 0 : \mathrm{P}(mnp - S_n \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(m-0)^2}\right) = \exp\left(-\frac{2t^2}{nm^2}\right).$$

In particular, when $t = mn\left(p - \frac{1}{2}\right) > 0$ ($p \geq 0.6 > 0.5 \Longrightarrow p - \frac{1}{2} > 0$, $m > 0, n > 0$ by definition, then $mn\left(p - \frac{1}{2}\right) > 0$) we have

$$\mathrm{P}\left(\cancel{mnp} - S_n \geq \cancel{mnp} - \frac{mn}{2}\right) \leq \exp\left(-\frac{2\cancel{m^2}n^{\cancel{2}}\left(p-\frac{1}{2}\right)^2}{\cancel{m^2}\,n}\right) = e^{-2n\left(p-\frac{1}{2}\right)^2}.$$

At the same time

$$\mathrm{P}\left(\cancel{mnp} - S_n \geq \cancel{mnp} - \frac{mn}{2}\right) = \mathrm{P}\left(-S_n \geq -\frac{mn}{2}\right) = \mathrm{P}\left(S_n \leq \frac{mn}{2}\right) = 1 - \mathrm{P}\left(S_n > \frac{mn}{2}\right).$$

Hence

$$1 - \mathrm{P}\left(S_n > \frac{mn}{2}\right) \leq e^{-2n\left(p-\frac{1}{2}\right)^2} \Longrightarrow \mathrm{P}\left(S_n > \frac{mn}{2}\right) \geq 1 - e^{-2n\left(p-\frac{1}{2}\right)^2}.$$

From the previous section, the minimal integer n, that satisfy the following inequality:

$$\mathrm{P}\left(S_n > \frac{mn}{2}\right) \geq 1 - e^{-2n\left(p-\frac{1}{2}\right)^2} \geq 0.99,$$

is the n chosen according to the rule:

$$n = \left\lceil -\frac{2\ln 0.01}{(2p-1)^2} \right\rceil.$$

Conclusion. To ensure that the majority of labels is correct with the probability 0.99, we should choose n with the following rule:

$$n = \left\lceil -\frac{2\ln 0.01}{(2p-1)^2} \right\rceil.$$

Task 2

First condition. For the random variable M, $M \in \mathbb{R}$ (real value), for deterministic real value $a \in \mathbb{R}$, the expectation

$$\mathrm{E}[aM] = a\mathrm{E}[M]. \tag{4}$$

Second condition. For random matrix M of size $d \times d$ the expectation $\mathrm{E}[M]$ is the expectation of each element:

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1d} \\ m_{21} & m_{22} & \cdots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & \cdots & m_{dd} \end{pmatrix} \Longrightarrow \mathrm{E}[M] = \begin{pmatrix} \mathrm{E}[m_{11}] & \mathrm{E}[m_{12}] & \cdots & \mathrm{E}[m_{1d}] \\ \mathrm{E}[m_{21}] & \mathrm{E}[m_{22}] & \cdots & \mathrm{E}[m_{2d}] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[m_{d1}] & \mathrm{E}[m_{d2}] & \cdots & \mathrm{E}[m_{dd}] \end{pmatrix}.$$

Another property (extension of the first condition) [2]. For any random variables $X, Y \in \mathbb{R}$, for any deterministic constants $a, b \in \mathbb{R}$, we have

$$\mathrm{E}[aX + bY] = a\mathrm{E}[X] + b\mathrm{E}[Y]. \tag{5}$$

Consider now the term $\mathrm{E}\left[x^T M x\right]$.

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \Longrightarrow x^T = (x_1, x_2, ..., x_d) \Longrightarrow M \cdot x = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1d} \\ m_{21} & m_{22} & \cdots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & \cdots & m_{dd} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{d} x_j m_{1j} \\ \sum_{j=1}^{d} x_j m_{2j} \\ \vdots \\ \sum_{j=1}^{d} x_j m_{dj} \end{pmatrix}$$

$$\Longrightarrow x^T \cdot Mx = (x_1, x_2, ..., x_d) \cdot \begin{pmatrix} \sum_{j=1}^{d} x_j m_{1j} \\ \sum_{j=1}^{d} x_j m_{2j} \\ \vdots \\ \sum_{j=1}^{d} x_j m_{dj} \end{pmatrix} = \sum_{i=1}^{d} x_i \sum_{j=1}^{d} x_j m_{ij} = \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j m_{ij}$$

$$\implies \mathrm{E}\left[x^T M x\right] = \mathrm{E}\left[\sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j m_{ij}\right] \overset{(5)}{=} \sum_{i=1}^{d}\sum_{j=1}^{d} \mathrm{E}\left[x_i x_j m_{ij}\right] \overset{(4)}{=} \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j \mathrm{E}\left[m_{ij}\right].$$

Thus, we have

$$\mathrm{E}\left[x^T M x\right] = \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j \mathrm{E}\left[m_{ij}\right]. \tag{6}$$

Now consider the term $x^T \mathrm{E}[M]\, x$.

$$\mathrm{E}[M] \cdot x = \begin{pmatrix} \mathrm{E}[m_{11}] & \mathrm{E}[m_{12}] & \cdots & \mathrm{E}[m_{1d}] \\ \mathrm{E}[m_{21}] & \mathrm{E}[m_{22}] & \cdots & \mathrm{E}[m_{2d}] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[m_{d1}] & \mathrm{E}[m_{d2}] & \cdots & \mathrm{E}[m_{dd}] \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{1j}\right] \\ \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{2j}\right] \\ \vdots \\ \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{dj}\right] \end{pmatrix}$$

$$\implies x^T \cdot \mathrm{E}[M]\, x = (x_1, x_2, ..., x_d) \cdot \begin{pmatrix} \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{1j}\right] \\ \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{2j}\right] \\ \vdots \\ \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{dj}\right] \end{pmatrix} = \sum_{i=1}^{d} x_i \sum_{j=1}^{d} x_j \mathrm{E}\left[m_{ij}\right] = \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j \mathrm{E}\left[m_{ij}\right].$$

Hence, we have

$$x^T \mathrm{E}[M]\, x = \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j \mathrm{E}\left[m_{ij}\right]. \tag{7}$$

Combining (6) and (7), we have

$$\mathrm{E}\left[x^T M x\right] = \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j \mathrm{E}\left[m_{ij}\right] = x^T \mathrm{E}[M]\, x,$$

and therefore, we finally have

$$\mathrm{E}\left[x^T M x\right] = x^T \mathrm{E}[M]\, x \quad \square$$

References

[1] Expectation of Binomial Distribution - ProofWiki https://proofwiki.org/wiki/Expectation_of_Binomial_Distribution.

[2] Linearity of Expectation Function - ProofWiki https://proofwiki.org/wiki/Linearity_of_Expectation_Function.

[3] Dekking, F. M., Kraaikamp, C., Lopuhaä, H. P., and Meester, L. E. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer Texts in Statistics. Springer-Verlag, London, 2005.

[4] Grimmett, G., Welsh, D. J. A., and Welsh, D. *Probability: An Introduction*. Clarendon Press, 1986. Google-Books-ID: DyifaCLXxkIC.

[5] Nowak, R. Lecture 7: PAC bounds and Concentration of Measure. 6.

CS 541 Artificial Intelligence: Mid-Term Exam

Instructor: Jie Shen

10/22/2019, 18:30 – 21:00 EST

Instructions:

- Closed-book exam, rely entirely on *your* memory (not other's);
- Always give your answer and explain it (guaranteed 1 point for nonempty answer);
- 20 points per problem, totally 110 points (20 * 5 + 10).

0. Write down your name. (10 pts)

1. Give a real-world AI problem which, from your point of view, cannot be solved in five years. If you are going to work on it, what is your first step to approach the main challenges?

2. We learned from the class that if a function is convex and L-smooth, then the best learning rate for gradient descent (GD) is $\frac{1}{L}$. If a function is α-strongly convex and L-smooth, then the best learning rate for GD is $\frac{1}{\alpha+L}$.

- Consider the following program:

$$\min_{w} \; F(w) = w^2, \; \text{s.t.} \; w \in [-1, +1]. \tag{1}$$

  Show that $F(w)$ is strongly convex with $\alpha = 2$ and is smooth with $L = 2$. Plot a possible curve of "$\left|w^t - 0\right|$ v.s. t" where $\{w^t\}_{t\geq 1}$ is the sequence of iterates produced by GD with the best learning rate, and $w^0 = 1$.

- Consider another program with the *same* optimal solution:

$$\min_{w} \; F(w) = w^{100}, \; \text{s.t.} \; w \in [-1, +1]. \tag{2}$$

  Show that $F(w)$ is smooth with $L = 9900$. Plot a possible curve of "$\left|w^t - 0\right|$" v.s. t where $\{w^t\}_{t\geq 1}$ is the sequence of iterates produced by GD with the best learning rate, and $w^0 = 1$.

- State the difference of the convergence behavior of the above to problems.

- Now consider linear regression with observation data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and response $\boldsymbol{y} \in \mathbb{R}^n$. State why we typically minimize the least square loss $F(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$ rather than $F(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^{100}$.

3. Stochastic gradient descent (SGD) bears the benefits of reducing overall computational complexity which makes it a prominent choice in large-scale machine learning. In order to apply SGD to obtain a convergent sequence, a key step is to **(a)** rewrite the objective function in the form $F(\boldsymbol{w}) = \frac{1}{m}\sum_i^m f_i(\boldsymbol{w})$ for some m and some function f_i, and **(b)** ensure that for a uniformly chosen index $i \in \{1, 2, \ldots, m\}$, $\mathbb{E}_i[\nabla f_i(\boldsymbol{w})] = \nabla F(\boldsymbol{w})$. This way SGD perform as GD in expectation which is a good news since we know GD reduces the objective value.

Consider solving the following convex program with SGD:

$$\min_{\boldsymbol{w}} \; F(\boldsymbol{w}) = \left(\frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{x}_i \cdot \boldsymbol{w}|\right) + \|\boldsymbol{w}\|_1 . \tag{3}$$

- Suppose we pick $m = n+1$ with $f_1 = |y_1 - \boldsymbol{x}_1 \cdot \boldsymbol{w}|, f_2 = |y_2 - \boldsymbol{x}_2 \cdot \boldsymbol{w}|, \ldots, f_n = |y_n - \boldsymbol{x}_n \cdot \boldsymbol{w}|, f_{n+1} = |y_{n+1} - \boldsymbol{x}_{n+1} \cdot \boldsymbol{w}|$. Calculate $\mathbb{E}_i[\nabla f_i(\boldsymbol{w})]$.
- If we choose $m = n$ with $f_i = |y_i - \boldsymbol{x}_i \cdot \boldsymbol{w}| + \|\boldsymbol{w}\|_1$ for $i = 1, \ldots n$, what is $\mathbb{E}_i[\nabla f_i(\boldsymbol{w})]$?

4. Both principal component analysis (PCA) and random projection (RP) are widely used tools for dimension reduction. From a unified perspective, the only difference between them lies on the projection matrix. Suppose the data matrix $\boldsymbol{Y} \in \mathbb{R}^{d \times n}$. PCA projects $\boldsymbol{Y}$ onto $\boldsymbol{U}_{1:r}^{\top}\boldsymbol{Y}$ where $\boldsymbol{U}_{1:r}$ consists of the first r columns of $\boldsymbol{U}$ which is produced by singular value decomposition. In contrast, RP transforms the data into $\boldsymbol{AY}$ where $\boldsymbol{A} \in \mathbb{R}^{r \times d}$ is a random matrix (typically Gaussian).

- Give an example where PCA might be a better choice than RP;
- Give another example where RP is preferred;
- What are the main drawbacks of both methods, and what is your solution?

5. Suppose that you have a network of many computers, each of which has limited storage space. State a strategy that allows you to learn a good classification model from a massive amount of data ("massive" means the whole data set cannot be stored in any single machine).

CS 541-B Artificial Intelligence: Final Exam

Instructor: Jie Shen

12/14/2020, 6:30 pm - 9:00 pm EST

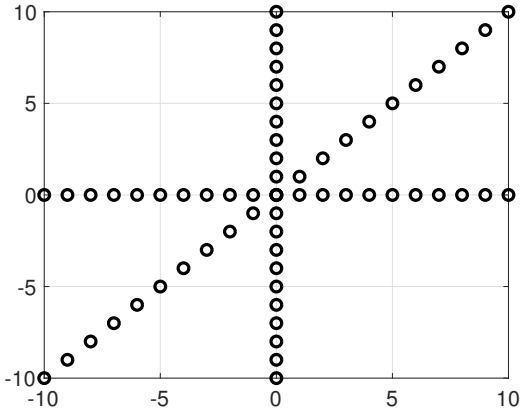
Instructions:

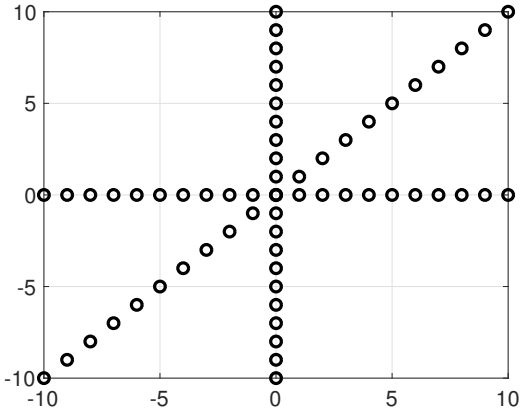
- Open book exam, feel free to use any lecture notes;
- Discussion is not permitted;
- Always give your answer and explain it (guaranteed 5 point for nonempty answer);
- 20 points per problem, totally 110 points (20 * 5 + 10).

0. Write down your name. (10 pts)

1. Consider the following data points (represented by circles) in 2-dimensional space.

- Illustrate the group structure discovered by sparse subspace clustering in the left panel;
- Illustrate the clustering result of single iteration of k-means with $k = 3$ and initial centers $(-10, 0)$, $(0, -10)$, $(10, 10)$ in the right panel.





2. An outlier is a data point that behaves abnormally, that its magnitude is out of control. For example, consider the age of a patient in his transcript: it is an outlier if it shows age = 2000, or if age = 25.5. Even if we observe a "normal" value of 85 is may also be an outlier provided that his actual age is 8.

Outlier may incur for a variety of reasons, e.g. human mistakes. In this case, we have to take such dirty data into algorithmic design. Consider the simple regression model in high-dimensional regime:

$$y_i = \langle \boldsymbol{a}_i, \boldsymbol{w}_{\text{true}} \rangle + e_i, \quad \|\boldsymbol{w}_{\text{true}}\|_0 \leq k, \quad i = 1, \ldots, n. \tag{0.1}$$

In the above expression, $\boldsymbol{a}_i \in \mathbb{R}^d$ is the feature vector, $\boldsymbol{w}_{\text{true}}$ is the groundtruth model we aim to estimate, $e_i \in \mathbb{R}$ is the possible outlier for the ith sample and $y_i \in \mathbb{R}$ is the corrupted response.

- Given $\{\boldsymbol{a}_i, y_i\}_{i=1}^n$ for sufficiently large n, say $n \to \infty$, is it possible to learn the true model without any prior knowledge/condition on $\{e_i\}_{i=1}^n$?
- Now suppose that only n_1 samples are corrupted by outliers where $n_1 \ll n$. In other words, among $\{e_i\}_{i=1}^n$, $n - n_1$ of them are zeros (but we do not know which of them have zero values). Give a proper formulation under which it is possible to simultaneously recover $\boldsymbol{w}_{\text{true}}$ and detect the outliers.

3. Suppose we have a set of data from n patients as in Table 1, where for each column and each row there are some missing entries (represented by the symbol "?"). State when and how we can estimate all the missing values.

Table 1: Patient data.

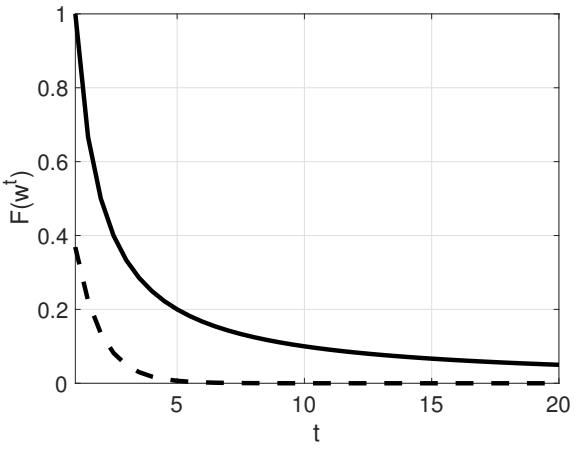
	Age	Weight	Height	Gender	Blood Pressure	$\cdots$	Sharp Pain
Patient 1	?	z_{12}	z_{13}	z_{14}	?	$\cdots$	z_{1m}
Patient 2	?	z_{22}	?	z_{24}	z_{25}	$\cdots$	?
Patient 3	z_{31}	?	z_{33}	?	?	$\cdots$	z_{3m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Patient n	?	z_{n2}	z_{n3}	?	z_{n5}	$\cdots$	?

4. In real-world applications the positive and negative classes are often imbalanced. For example, there are a massive amount of data from airplanes that operate normally, while the number of aviation accidents is relatively few. Therefore, in order for a machine learning system to detect abnormal status, the algorithm must appreciate those negative examples (i.e. the data from flight recorder). Given a set of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{+1, -1\}$, state how you will design a learning algorithm for such abnormality detection system.

5. Consider two functions $F_1(\boldsymbol{w})$ and $F_2(\boldsymbol{w})$: both of them are strongly convex, but F_1 is smooth and F_2 is non-smooth. Suppose we apply GD to optimize these two functions. The following figure shows two convergence curves: a solid line and a dashed line. One is for $F_1(\boldsymbol{w}^t)$ and another for $F_2(\boldsymbol{w}^t)$.

- Explain which curve may correspond to F_2;

- Plot another possible convergence curve of applying GD to optimize F_2 with a different initial iterate or learning rate.



CS 541-A Artificial Intelligence: Final Exam

Instructor: Jie Shen

12/15/2020, 6:30 pm - 9:00 pm EST

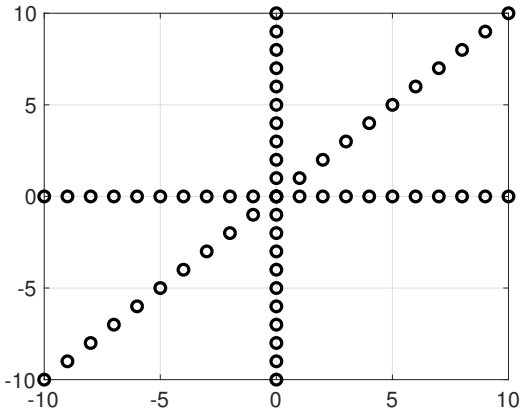
Instructions:

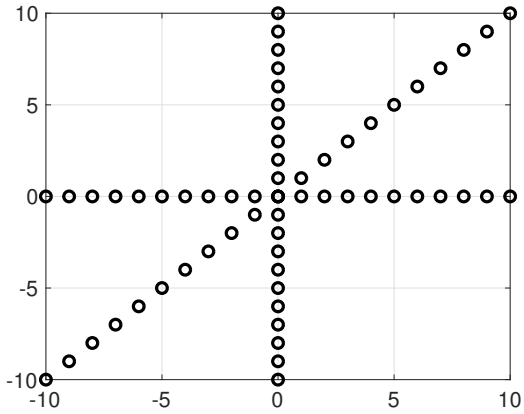
- Open book exam, feel free to use any lecture notes;
- Discussion is not permitted;
- Always give your answer and explain it (guaranteed 5 point for nonempty answer);
- 20 points per problem, totally 110 points (20 * 5 + 10).

0. Write down your name. (10 pts)

1. Consider the following data points (represented by circles) in 2-dimensional space.

- Illustrate the group structure discovered by sparse subspace clustering in the left panel;
- Illustrate the clustering result of single iteration of k-means with $k = 3$ and initial centers $(-10, 0)$, $(0, -10)$, $(10, 10)$ in the right panel.





2. Suppose we have gathered data from n patients as in Table 1, where some entries in the column "Blood Pressure" are missing (represented by the symbol "?"), and other columns are fully observed. Our goal is to estimate these missing values based on the current data matrix. Formulate it as a machine learning problem and state how we can make prediction.

Table 1: Patient data.

	Age	Weight	Height	Gender	Blood Pressure	$\cdots$	Sharp Pain
Patient 1	z_{11}	z_{12}	z_{13}	z_{14}	**?**	$\cdots$	z_{1m}
Patient 2	z_{21}	z_{22}	z_{23}	z_{24}	z_{25}	$\cdots$	z_{2m}
Patient 3	z_{31}	z_{32}	z_{33}	z_{34}	**?**	$\cdots$	z_{3m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Patient n	z_{n1}	z_{n2}	z_{n3}	z_{n4}	z_{n5}	$\cdots$	z_{nm}

3. Now suppose we have another set of data from n patients as in Table 2, where for each column and each row there are some missing entries (represented by the symbol "?"). State when and how we can estimate all the missing values.

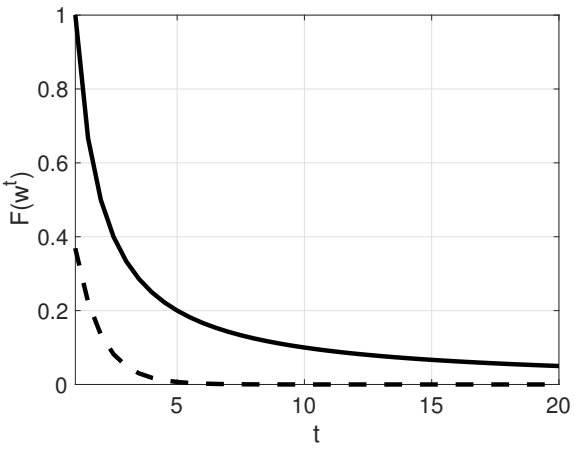
Table 2: Patient data.

	Age	Weight	Height	Gender	Blood Pressure	$\cdots$	Sharp Pain
Patient 1	**?**	z_{12}	z_{13}	z_{14}	**?**	$\cdots$	z_{1m}
Patient 2	**?**	z_{22}	**?**	z_{24}	z_{25}	$\cdots$	**?**
Patient 3	z_{31}	**?**	z_{33}	**?**	**?**	$\cdots$	z_{3m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Patient n	**?**	z_{n2}	z_{n3}	**?**	z_{n5}	$\cdots$	**?**

4. The error type of false positive is defined as follows: an algorithm outputs positive for a sample but in reality its label is negative. In some real-world applications, a learning algorithm should never incur the error of false positive. For example, the face recognition system of a laptop is designed such that it always denies unauthorized access to confidential data. Likewise, a self-driving car shall never recognize a red traffic light as green (but it is fine to classify green as red). Given a set of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{+1, -1\}$, state a proper formulation which is capable of preventing false positive.

5. Consider two functions $F_1(\boldsymbol{w})$ and $F_2(\boldsymbol{w})$: both of them are strongly convex, but F_1 is smooth and F_2 is non-smooth. Suppose we apply GD to optimize these two functions. The following figure shows two convergence curves: a solid line and a dashed line. One is for $F_1(\boldsymbol{w}^t)$ and another for $F_2(\boldsymbol{w}^t)$.

- Explain which curve may correspond to F_1;
- Plot another possible convergence curve of applying GD to optimize F_2 with a different initial iterate or learning rate.


1
0.8
0.6
0.4
0.2
0
F(w^t)
5
10
15
20
t