Diego Medrano Jiménez
Rua Gama Cerqueira
466, apto 920
Cambuci, 01539-010
São Paulo, Brasil

diego.medranojimenez@gmail.com
+55(11)95109-9379

TEST - Cientista de Dados

1. *Given the following function*

$$f(t,x) = \mathrm{e}^{-\omega t^2} \cos\left(\frac{n\pi x}{L}\right) ,$$

find the following partial derivatives:

$$\frac{\partial f}{\partial t} , \qquad \frac{\partial f}{\partial x} .$$

Assuming any of the parameters have an implicit space-time dependence, i.e. ω, L do not depend on x or t, we have

$$\frac{\partial f}{\partial t} = -2\mathrm{e}^{-\omega t^2} \omega t \cos\left(\frac{n\pi x}{L}\right) , \qquad \frac{\partial f}{\partial x} = -\mathrm{e}^{-\omega t^2} \frac{n\pi}{L} \sin\left(\frac{n\pi x}{L}\right) , \quad \text{where} \begin{cases} \omega \neq \omega(t,x) , \\ L \neq L(t,x) , \\ n \in \mathbb{N} . \end{cases}$$

2. *Give an example of a linear vector space (with basis).*

A textbook prototype example of a linear vector space V could be our *3-dimensional space* in which vectors $\vec{v}$ represent *velocities of a particle.* In such a case, the particle velocity is characterized by its velocity components —e.g. v_x, v_y, v_z—, and can be written in terms of a basis —e.g. $\hat{u}_x$, $\hat{u}_y$, $\hat{u}_z$— in this way

$$\vec{v} \equiv (v_x, v_y, v_z) = v_x \hat{u}_x + v_y \hat{u}_y + v_z \hat{u}_z .$$

The particular choice of a basis is just a convention, but it is important to make sure that all its elements are *linearly independent* and *span the whole vector space.* In the expression above, the elements of the basis are defined as

$$\hat{u}_x = (1,0,0) , \qquad \hat{u}_y = (0,1,0) , \qquad \hat{u}_z = (0,0,1) ,$$

which clearly fulfill the two conditions to form a basis. Additionally, this basis is said to be *orthonormal* because it also fulfills the condition

$$\hat{u}_i \cdot \hat{u}_j = \begin{cases} 0 & i \neq j \quad : \textit{orthogonal vectors} \\ 1 & i = j \quad : \textit{normal vectors} \end{cases} \qquad \forall i,j = x,y,z\,,$$

with "$\cdot$" being the scalar product of the vector space. However, the orthonormality condition is not necessary to define the basis. A different valid non-orthonormal basis could be instead

$$u_1 = (1,1,1)\,, \qquad u_2 = (0,1,1)\,, \qquad u_3 = (0,0,1)\,.$$

There are many other examples of n-dimensional linear vector spaces —e.g. space of *linear functions*, *polynomials*, *solutions of homogeneous linear equations*. . . —, but at the end of the day they are all proven to be isomorphic to the respective n-dimensional Euclidean space, which is the one presented above for $n = 3$.

3. *Find the inverse for the matrix* $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

The inverse of a regular $n \times n$ square matrix A is defined as the matrix A^{-1} such that $AA^{-1} = A^{-1}A = \mathbb{I}$. Written in components, it can be proven that

$$A^{-1} = \frac{1}{\det(A)}\,\mathrm{adj}\left(A^T\right) = \frac{n!}{(n-1)!}\,\frac{\varepsilon_{i_1\ldots i_{n-1}i}\varepsilon_{j_1\ldots j_{n-1}j}a_{i_1j_1}\ldots a_{i_{n-1}j_{n-1}}}{\varepsilon_{k_1\ldots k_n}\varepsilon_{l_1\ldots l_n}a_{k_1l_1}\ldots a_{k_nl_n}}\,,$$

where $\det(A)$ is just the determinant of the matrix A, $\mathrm{adj}(A^T)$ is the adjoint of its transverse matrix, and $\varepsilon_{i_1\ldots i_n}$ is the Levi-Civita symbol in dimension n. This explicit expression looks a bit intricated —and it is indeed—, but in the 2-dimensional case, it simply is

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \rightarrow \quad \left\{ \begin{array}{c} \det(A) = ad - bc \\ \mathrm{adj}(A) = \begin{pmatrix} d & -c \\ -b & a \end{pmatrix} \end{array} \right\}$$

$$\Rightarrow \quad A^{-1} = \frac{1}{\det(A)}\,\mathrm{adj}\left(A^T\right) = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

There exist much more optimal and practical ways to compute the inverse of a matrix, specially when the dimension n is big. I will review the *Gauss' method* later on.

4. *Give a definition and an example of a supervised learning algorithm.*

A *supervised learning algorithm* is mainly characterized by the presence of some given input variables X, often called *features*, that affect directly the value of one or more output variables Y, the *targets*. With this in mind:

> *A* ***supervised*** *learning algorithm is a set of rules defining a process in which a machine tries to reproduce the patterns of the mapping $X \mapsto Y$ in order to make predictions over Y when a new input X is given.*

In other words, the algorithm learns from existing data —i.e. training data— how to estimate the target variable, and obtains the best fit to make later target predictions from new features.

We have two kinds of *supervised* learning problems:

- Regression problems: where we need to predict numerical data e.g. *linear regression.*
- Classification problems: where we need to predict categorical data e.g. *support vector machines (SVM).*

5. *Give a definition and an example of an unsupervised learning algorithm.*

On the contrary, an *unsupervised learning algorithm* is characterized by the absence of any output. It just works with the input variables. In this case:

> *An* ***unsupervised*** *learning algorithm is a set of rules defining a process in which a machine tries to unveil the underlying structure or distribution of the existing data, without any additional information.*

In other words, the algorithm tries again to recognize some patterns, but uses them to organize the data and not to make any prediction.

We have two kinds of *unsupervised* learning problems:

- Clustering problems: where we need to group data with similar features e.g. *k-means clustering algorithms.*
- Association rule learning problems: where we need to discover rules that describe large portions of the data e.g. *Apriori algorithms.*

6. *Name 3 types of metrics, and their formulas, used in k-means clustering algorithm.*

A clustering algorithm is used to group all the entries of a dataset with similar features, providing an automatic way of organization even without any previous historical data. Considering all the numerical variables in the dataset as different dimensions of a vector space, a cluster is then defined as a neighborhood of closeby points in that vector space. The concept of *metric* and distance acquires here a lot of importance to make an optimal clusterization.

Given two points in a n-dimensional vector space —i.e. $\vec{x} = (x_1, x_2, \ldots, x_n)$, $\vec{y} = (y_1, y_2, \ldots, y_n)$— we can consider the *Euclidean metric* given by the formula

$$d(\vec{x}, \vec{y}\,|\,\text{Euclidean}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,,$$

which represents the usual shortest distance between two points following a straight line. However, depending on the nature of the problem, it is also interesting to take into account some other options

- $d(\vec{x}, \vec{y}\,|\,\text{Manhattan}) = \sum_{i=1}^{n} |x_i - y_i|\,,$
- $d(\vec{x}, \vec{y}\,|\,\text{Minkowski}) = \Big(\sum_{i=1}^{n} |x_i - y_i|^p\Big)^{1/p}\,,$
- $d(\vec{x}, \vec{y}\,|\,\text{Chebychev}) = \max_{i=1}^{n} |x_i - y_i|\,.$

Notice[1] that the Euclidean metric corresponds to a Minkowski metric with $p = 2$ whereas the Manhattan metric corresponds to $p = 1$.

Apart from these distances, many others can be found inside the `scipy.spatial.distance` Python module such as *Bray-Curtis*, *cosine*, *Jensen-Shannon*, *Mahalanobis*...

7. *Name and describe any algorithm to inverse a matrix.*

As I said before, I am going to describe the *Gauss' method* to obtain the inverse of a matrix A. It turns out to be a much simpler process than the explicit definition, meaning that the computation is faster and computationally less expensive. The idea is pretty simple and uses the implicit definition of the inverse matrix, i.e. $AA^{-1} = A^{-1}A = \mathbb{I}$.

If A is a regular $n \times n$ matrix[2], it is easy to see that it can be transformed into the identity $\mathbb{I}$ by performing a finite amount of *elementary row operations*:

- Swap rows,
- Multiply or divide each element in a row by a constant,
- Replace a row by adding or subtracting a multiple of another row to it.

Then, applying the same set of elemetary row operations to the identity $\mathbb{I}$, the method says that the resulting matrix is precisely the inverse matrix A^{-1}.

Schematically, concatenating both matrices into a single one and performing the same set of elementary row operations on both sides, we would have the following result

$$[A|\mathbb{I}] \quad \xrightarrow{\textit{row ops.}} \quad [\mathbb{I}|A^{-1}]\ .$$

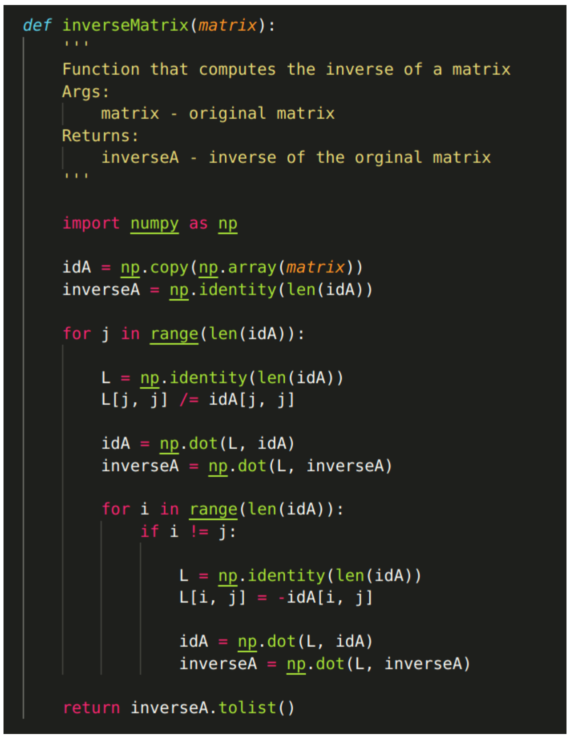
[1] It is possible as well to stablish a hierarchy among the variables of the dataset for any distance by considering a pondered sum, e.g. $d'(\vec{x}, \vec{y}\,|\,\text{Manhattan}) = \sum_{i=1}^{n} w_i |x_i - y_i|$, with $\vec{w} = (w_1, \ldots, w_n)$ being a vector of weights.

[2] This is indeed, according to the definition of the inverse matrix in 'Question 3', a necessary condition for the existence of the inverse, i.e. $\det(A) \neq 0$.

Analogously, writing every row operation as a left matrix operator L_i, it is easy to see from the implicit definition that

$$L_n L_{n-1} \dots L_1 A = \mathbb{I} \qquad \Rightarrow \qquad L_n L_{n-1} \dots L_1 \mathbb{I} = A^{-1} \,. \tag{1}$$

Here is a simple Python function that reproduces the *Gauss' method* in (1):



```python
def inverseMatrix(matrix):
    '''
    Function that computes the inverse of a matrix
    Args:
        matrix - original matrix
    Returns:
        inverseA - inverse of the orginal matrix
    '''

    import numpy as np

    idA = np.copy(np.array(matrix))
    inverseA = np.identity(len(idA))

    for j in range(len(idA)):

        L = np.identity(len(idA))
        L[j, j] /= idA[j, j]

        idA = np.dot(L, idA)
        inverseA = np.dot(L, inverseA)

        for i in range(len(idA)):
            if i != j:

                L = np.identity(len(idA))
                L[i, j] = -idA[i, j]

                idA = np.dot(L, idA)
                inverseA = np.dot(L, inverseA)

    return inverseA.tolist()
```

Fig. 1. See attached notebook `InverseMatrix.ipynb` for details.

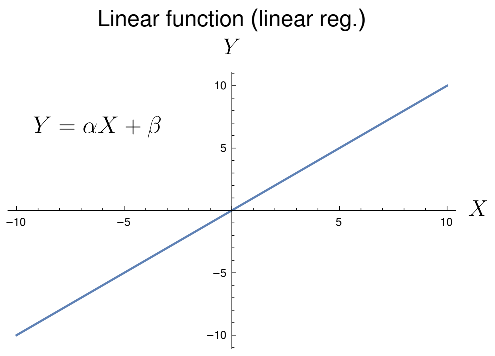
8. *Explain what is a logistic regression and give an example of application (describe it as best as you can).*

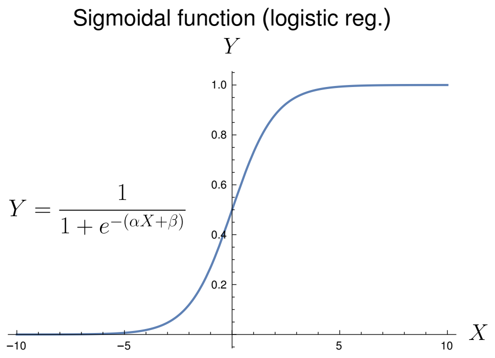
A logistic regression is a *supervised learning algorithm* that allows, based on a prior knowledge of some of the data, to make predictions on a categorical variable in order to solve a *classification problem*. The input features X can be either numerical or categorical[3], whereas the predicted target Y is typically a binary variable: 0 / 1, 'yes' / 'no' …

The mathematical idea behind the algorithm is pretty similar to that of the linear regression, in which data is fitted into a linear function, but in the logistic regression a *sigmoidal* function[4] is used.

[3]Categorical variables can be transformed into numerical or dummy variables if needed.

[4]The argument of the exponential in the sigmoidal function is normally chosen to be lineal —i.e. $-(\alpha X + \beta)$—, but higher degree arguments can be used according to the nature and complexity of the problem. The correct regression function to be used is ultimately derived from the *Maximum Likelihood Method.*





As a consequence, the regression is able to classify the data into two groups —e.g. 'success'/'failure'— depending on how close the target fits into 0 or 1[5].

Let us better explain an example of application...

- ***Hit Songs*** in Music History

The production of *hit songs* is definitely a big concern for music industry nowadays. It has been decades of research on how to produce a good song and obtain its perfect recipe. Certainly the problem is pretty complicated because hit songs strongly depend on musical genres, trends, culture... and such a guideline would be very subjective. Imagine how amazing would be for a record company to have the 'secret formula' to produce hits sistematically. Well... that formula does not really exist, but... what if we could at least uncover some features and common patterns of 'success' looking at different songs throughout music history? The logistic regression can help us on approaching this problem!!

Here is an example of potential variables X that could affect the level of success and that are easily readable from a song:

- *Song title*: this is just the ID of the song.
- *Year*: important variable keeping track of the musical trends.
- *Artist name*: famous artists are already more probable to produce a hit song.
- *Duration*: I guess there is an equilibrium on how short or long a popular song is.
- *Lyrics*: the topic of the song is clearly something to take into account. We could include here *key words* form the lyrics and ultimately define a *single topic.*
- *Time signature*: the rhythm of the song is a really strong feature in some musical genres such as reggaeton.
- *Loudness*: measure of the volume range of the song. Pop music is known for example to be super-compressed.
- *Tempo*: how slow or fast the song is.
- *Key*: including both major and minor keys.
- *Chords:* presence of repetitive chord progessions. Remeber the famous chord progression in pop music I-V-vi-IV.
- *Presence of vocals*: songs with lead vocals are usually more popular.
- *Pitch range*: short pitch range could sound a bit monotonous whereas a long range could sound caotic.
- ...

[5]These are the limits of the sigmoidal function.

Of course, these considerations just reflect my personal opinion about which variables could be significant, but many others may be considered.

For the output target variable Y we could then wander if a song will reach the top10 lists as a measure of success:

$$\text{Success:} \qquad Y = \begin{cases} 1 & : \text{song reaches top10 lists of best songs}\,, \\ 0 & : \text{otherwise}\,. \end{cases}$$

Thus, after performing the logistic regression over these variables on historical data, we would be able to make a prediction (or at least to get important insights) about how a new song will behave when released into the music market.

9. *Describe in steps how would you perform preliminary analysis on a dataset.*

There are always some clear steps that I follow on any preliminar data analysis:

- **Loading the data:** First I always make sure that all the data is correctly loaded in a dataframe with the correct number of entries and that all the existing variables have perfectly been separated into different columns.
- **Checking data types:** I start getting familiar with the content of each variable and try to understand what they actually represent. Knowing the approximate numerical intervals or the categories of each variable is crucial for a posterior analysis.
- **Handling missing values:** Many prediction algorithms struggle a lot with *Null* values when trying to fit the data. Depending on the nature of the problem I am working on, I simply discard those values or replace them by appropiate ones.
- **Visualizing data:** It is important to start doing some plots in order to absorb the data quickly and to actually understand it. This can be done both for numerical and categorical values through *scatterplots* or *histograms*. Computing the mean values, standard deviations. . . might also be important at this stage to get an idea of the data distribution.
- **Identifying and removing outliers:** The data visualization step is superuseful as well to identify possible outliers —i.e. data out of the trend—. It is important to process or to remove them in order to create a model with good predictions.
- **Looking for correlations:** There may exist correlations between the different variables that have to be taken into account in the modelling stage. The correlations allow you as well to identify the most relevant variables of the problem and to get important insights about data behavior.
- **Transforming categorical into numerical data:** Most of the time, prediction models need numerical values as input. The use of dummy variables allows you to do this.
- **Feature scaling:** Finally, whenever needed and depending on the variable, I apply *data normalization* or *data standarization* mainly to prevent the features with larger magnitudes from dominating the behavior of the model.

And thus, the dataset is ready for a deeper analysis to start constructing your model or to start doing your predictions.

10. *Given the two tables A and B, give the result for the query below.*

ID	Name	Born_in
9991	John	Liverpool
9992	Paul	Liverpool
9993	Ringo	Liverpool
9994	George	Liverpool

Table A

ID	Instrument	Football_Club
9991	Vocal	Liverpool
9992	Base	Liverpool
9993	Drums	Liverpool
9995	Guitar	Arsenal
9992	Piano	Liverpool

Table B

```
SELECT A.ID, A.Name, A.Born_in, B.Instrument, B.Football_Club
   FROM A LEFT JOIN B
   ON A.ID = B.ID;
```

- This query is going to result in a single table containing five different columns, corresponding to the information in *ID*, *Name*, *Born_in*, *Instrument* and *Football_Club*.
- The correspondence between the original tables is done through identification of their *IDs*.
- Since it is a *left join*, the resulting table will contain *all* the entries of *Table A*, completed whenever possible with the appropiate information from the *Table B* columns. *Null* values will be introduced to fill the blanks.
- In the presence of *non-unique IDs*, new *Table A* entries will be generated to include the corresponding information from *Table B*.

As a result:

ID	Name	Born_in	Instrument	Football_Club
9991	John	Liverpool	Vocal	Liverpool
9992	Paul	Liverpool	Piano	Liverpool
9992	Paul	Liverpool	Base	Liverpool
9993	Ringo	Liverpool	Drums	Liverpool
9994	George	Liverpool	*Null*	*Null*

Table 1. Result of the SQL query above.

Check the attached file `SQLquery.ipynb` for details on this query.

Social media:

- GitHub: `https://github.com/d-medrano/`
- LinkedIn: `https://www.linkedin.com/in/diego-medrano-jiménez-b1b3a5b2/`
- Instagram: `https://www.instagram.com/d.medrano.j/`
- YouTube: `https://www.youtube.com/channel/UCWIOOsOFRLaXEQfONvwRAhA/`