EE2211 : Introduction to Machine Learning (Fall 2020)
Different Types of Data

Vincent Y. F. Tan

In this document, we will summarize the different types of data you will encounter. We will also discuss normalizing data of different scales (or units).

1 Numerical Data

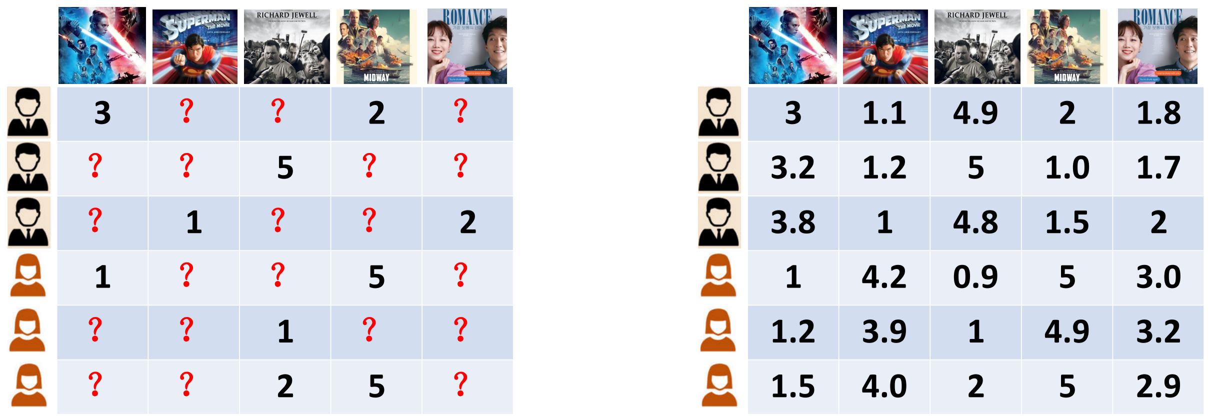
Numerical data, as the name suggests, is data that represents *numbers*. However, simple as it is, there are different types of numerical data we should be aware of.

- Discrete Data: These are data that are distinct and separate and can only take on certain values (usually finitely many values). This type of data can be counted. For example, the "number of heads in 100 coin flips" is discrete numerical data because they can only take on values in the set of 101 values $\{0, 1, 2, \ldots, 99, 100\}$. The "outcomes of a digital thermometer" is also discrete data because the real temperature is necessarily quantized, say to the range $\{35.0, 35.1, 35.2, \ldots, 40.4, 40.5\}$ centigrade. To ascertain whether you're dealing with discrete data, you can ask yourself the following questions: "Can you count it?" and "Can it be divided up in to smaller and smaller parts?". If the first answer is "yes" and the second is "no", you know you're dealing with discrete numerical data. If the discrete data only takes on the two values (e.g., in $\{0, 1\}$), then it is called *binary* data.

- Continuous Data: These are data that can't be counted but they can be measured. The height of a person is a real number. Note that no matter how fine the markings are of a ruler, you can't nail down the height of a person to infinite precision (because your measurements will be "off by a bit"). Similarly, the temperature is a real number; no matter how accurate your thermometer, the reading quantized and so what you get is only an *approximation* (to a certain number of significant digits/decimal points) to the true temperature. Think of it this way, how many decimal points are needed to describe the number $\pi \approx 3.14159\ldots$ or $e \approx 2.71828\ldots$? Infinitely many! This is analogous to the temperature. If the outcome variable is continuous, it makes sense to build a regression model.

Let me show you a real life example in which discrete data is present and we would like to predict discrete data given discrete data. Consider the Netflix problem[1] in which we have n users and m movies. The rating of each user $i \in [n] := \{1, 2, \ldots, n\}$ to each movie $j \in [m]$ is a number in the set $\{1, 2, 3, 4, 5\}$ in which 5 means user i very much likes movie j and 1 means s/he does not like it. Thus, the ratings constitute *discrete* data. In the Netflix problem, n is of the order of 10^6 and m is of the order 10^4. Because each user only rates a small subset of movies (s/he does not have enough time to rate all m movies), what we observe is a small subset of the rating matrix $M \in \{1, 2, 3, 4, 5\}^{n\times m}$. The indices of the entries in M that are observed are captured by the set $\Omega = \{(i, j) \in [n] \times [m] : \text{user } i \text{ has rated movie } j\}$. Thus, we observe M_Ω, defined as the entries of matrix we observe. There is also reason to believe that the full rating matrix M, which seems large, is low rank because the number of types of movies (column rank) is small (romance, action, documentary) and the number of types of users (row rank) is also small. The fact that M is low rank constitutes side-information for us to exploit to learn $M_{\Omega^c} = \{M_{i,j} : (i, j) \in \Omega^c\}$, the entries of the

[1] Read more about it here: `https://en.wikipedia.org/wiki/Netflix_Prize`


SUPERMAN
RICHARD JEWELL
MIDWAY
ROMANCE

3	?	?	2	?
?	?	5	?	?
?	1	?	?	2
1	?	?	5	?
?	?	1	?	?
?	?	2	5	?

3	1.1	4.9	2	1.8
3.2	1.2	5	1.0	1.7
3.8	1	4.8	1.5	2
1	4.2	0.9	5	3.0
1.2	3.9	1	4.9	3.2
1.5	4.0	2	5	2.9

Figure 1: We observe the partially filled matrix M_Ω on the left. We would like to use an algorithm to learn the filled (approximately low-rank) matrix on the right. What do you expect the singular value decomposition of the matrix on the right to look like?

Person	Race	Gender
Ravi	Indian	Male
Alice	Eurasian	Female
Li Ting	Chinese	Female
Ali	Malay	Male
Aisha	Malay	Female
Teck Seng	Chinese	Male
Giselle	Eurasian	Female
Jerry	Eurasian	Male

Table 1: A small dataset of 8 Singaporeans. "Race" and "gender" are the two features here.

matrix we do not observe. See Fig. 1. One simple formulation that exploit the low rank nature of M is the following convex optimization problem:

$$\hat{M} \in \arg\min \left\{ \|\widetilde{M}\|_* : \widetilde{M}_{i,j} = M_{i,j}, \forall\, (i,j) \in \Omega \right\} \tag{1}$$

where $\|\cdot\|_*$ is the nuclear norm (or sum of singular values), which is a proxy of the rank (the nuclear norm is the convex envelope of the rank function). Thus, we are seeking a real-valued matrix $\hat{M} \in \mathbb{R}^{n\times m}$ that minimizes the nuclear norm and is consistent with the given entries in M_Ω. There are many algorithms and provable guarantees for solving the problem in (1). I am personally interested in this problem.

2 Categorical Data

Categorical data represent *characteristics*. Again there are several types but we focus on the two below.

- Nominal data: These data represent discrete units and are used to represent variables that have no natural quantitative value. They are nothing but "labels". For example, consider the dataset in Table 1.

  We see that there are four categories for race $\{\text{Chinese}, \text{Indian}, \text{Malay}, \text{Eurasian}\}$ and two categories for gender $\{\text{Male}, \text{Female}\}$. So the features "race" and "gender" are categorical nominal data. To deal

Person	isChinese	isMalay	isIndian	isEurasian	isMale	isFemale
Ravi	0	0	1	0	1	0
Alice	0	0	0	1	0	1
Li Ting	1	0	0	0	0	1
Ali	0	1	0	0	1	0
Aisha	0	1	0	0	0	1
Teck Seng	1	0	0	0	1	0
Giselle	0	0	0	1	0	1
Jerry	0	0	0	1	1	0

Table 2: A small dataset of 8 Singaporeans after one-hot encoding

with categorical variables within a machine learning framework (which is more amenable to numerical data), we may use *one-hot encoding*, converting the above table/dataset to Table 2. Is the last column of Table 2 necessary for subsequent data analysis?

In Python, you can do one-hot encoding by using LabelEncoder and OneHotEncoding. Please play with the following code and use it to do one-hot encoding of the two nominal variables in Table 1.

```
from numpy import array
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

# define example
data = ['cold', 'cold', 'warm', 'cold', 'hot', 'hot', 'warm', 'cold', 'warm', 'hot']
values = array(data)
print(values)

# integer encode
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded)

# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)

# invert first example
inverted = label_encoder.inverse_transform([argmax(onehot_encoded[0, :])])
print(inverted)
```

- Ordinal data: These data represent discrete and *ordered* units. It is therefore nearly the same as nominal data, except that its ordering matters. For example, the set of all possible educational backgrounds in a survey form could be

  $$\{\text{primary}, \text{secondary}, \text{pre-university}, \text{university}, \text{postgraduate}\}.$$

  One can see that this values of this feature are *ordered* – hence the term *ordinal*. Indeed, the "difference" between primary and secondary is "smaller than" primary and postgraduate. This is the main limitation of ordinal data – the differences between the values are not really known. Because of that, ordinal scales (e.g., integers) are usually used to measure non-numeric features like happiness, customer satisfaction and so on. Thus when you rate your Grab driver, you rate her/him on a scale of 1 to 5 (likened to discrete numerical data), which is more quantitative, and amenable to machine learning, than

$$\{\text{very unsatisfactory}, \text{unsatisfactory}, \text{satisfactory}, \text{good}, \text{excellent}\}.$$

3 Normalizing Data

Often we have feature vectors in which features are on different scales. We let n be the number of training samples and d be the number of features as usual. Say the feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} x_{11} \\ x_{12} \\ \cdots \\ x_{1d} \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} x_{21} \\ x_{22} \\ \cdots \\ x_{2d} \end{bmatrix}, \quad \ldots \quad \mathbf{x}_n = \begin{bmatrix} x_{n1} \\ x_{n2} \\ \cdots \\ x_{nd} \end{bmatrix}.$$

The first feature x_{i1} for $1 \le i \le n$ could be the height of students measured in centimeters so the dynamic range is $[x_{\min,1}, x_{\max,1}] = [140, 190]$. The second feature x_{i2} for $1 \le i \le n$ could measure the (American) shoe size of the students so it ranges from $[x_{\min,2}, x_{\max,2}] = [6, 13]$. So even if both features are deemed equally "important", unfortunately, any machine learning method would place more importance on the first feature because of its larger values, which is not ideal. Thus, we have to scale or normalize the features so that their dynamic ranges are roughly the same. We can use (at least) two methods to do so:

- Z-score scaling: First we calculate the empirical mean and empirical standard deviation of each feature, say the i^{th}. These are

$$\hat{x}_1 = \frac{1}{n}\sum_{i=1}^{n} x_{i1}, \quad \text{and} \quad \hat{\sigma}_1 = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{i1} - \hat{x}_1)^2}. \tag{2}$$

  Then we create the normalized 1st features associated to each training sample as

$$\bar{x}_{i1} := \frac{x_{i1} - \hat{x}_1}{\hat{\sigma}_1}. \tag{3}$$

  We can do this for all features so that, in some sense, they are all "normalized". What is the empirical and empirical standard of the set of numbers $\{\bar{x}_{i1} : 1 \le i \le n\}$ now? Why do we divide by $n-1$ in the second equation in (2)? This is not so easy to answer actually. Can we divide by n? Read up on unbiased estimator of (population) variance or look at the notes for Lecture 3. Is the estimator for standard deviation $\hat{\sigma}_1$ or the estimator for variance $\hat{\sigma}_1^2$ unbiased? Can you prove it and what assumptions do we need to make? What happens as $n \to \infty$? The reason why this method of normalizing is called "Z-score" is because the standard normal random variable with density $x \in \mathbb{R} \mapsto \frac{1}{\sqrt{2\pi}} \exp\big(-x^2/2\big)$ is usually denoted as $Z \sim \mathcal{N}(0, 1)$.

  You can try this code for Question 1 in Tutorial 2 as follows.

```
averageExpenditureList = statistics.mean(expenditureList)
stdevExpenditureList = statistics.stdev(expenditureList)

subtraction = list(np.array(expenditureList) - np.array(averageExpenditureList))
normalizedExpenditureList = list(np.array(subtraction)/np.array(stdevExpenditureList))
print(normalizedExpenditureList)
```

- Min-Max scaling: Define the minimum and maximum values of feature 1 to be

$$x_{\min,1} := \min_{1\le i\le n} x_{i1}, \quad \text{and} \quad x_{\max,1} := \max_{1\le i\le n} x_{i1}. \tag{4}$$

  Then we create the normalized 1st features associated to each training sample as

$$\bar{x}_{i1} := \frac{x_{i1} - x_{\min,1}}{x_{\max,1} - x_{\min,1}}. \tag{5}$$

  Now what are the minimum and maximum of $\{\bar{x}_{i1} : 1 \le i \le n\}$? Create your own code to verify this.