Categorical variables

A categorical variable

- Has a value drawn from a discrete set called *Categories* or *Classes*
    - hence the term "Classification" when the target is categorical
- There is **no** ordering relationship between category/class values
    - { "Red", "Green", "Blue" } (set notation)
    - versus *ordinal* values ["Small", "Medium", "Large"] (sequence notation)
- There is no *magnitude*
    - even if I could order the colors: how much greater is "Blue" than "Red" ?

We will use C to denote the set of possible values in a category/class.

Since the values in C are unordered, C is mathematically a set of values

$$C = \{c_1, c_2, \ldots, \}$$

Since values in a category/class aren't ordered, they are often non-numeric.

Most algorithms deal with non-numeric data by encoding them as numeric.

In our Titanic example for Binary Classification, there were two obvious categorical variables

- `Survived` (the target)
- `Sex`

It might have gone un-noticed that the target was categorical

- Because the data was presented to us as having been encoded with number 1 (Survived) and 0 (didn't Survive)

We certainly noticed that `Sex` was non-numeric

- Hence, we created a transformation to encode Male/Female as 0/1

What if a categorical variable has more than 2 possible values for category/class ?

An obvious choice for a variable with $||C|| > 2$ is to encode the values with distinct integers

This is usually a **bad** idea !

Consider the `Pclass` feature from our Titanic example.

It was presented to us as a numeric feature (integers in $\{1, 2, 3\}$) so we may have taken for granted that `Pclass` was **not** categorical.

But it could have just as easily been present as non-numeric { "First", "Second", "Third" }.

- If not for coincidence, perhaps `Pclass` should be considered a *categorical* rather than *ordinal* feature ?
- Why is encoding the strings as $\{1, 2, 3\}$ any more correct than encoding them as $\{1, 2, 4\}$?

We will give a fuller answer in the module on Model Interpretation. For now:

- In a linear model
$$\mathbf{y} = \Theta^T \cdot \mathbf{x}$$
  - Thus, the contribution of the j^{th} feature $\mathbf{x}_j$ to prediction $\mathbf{y}$ is $\Theta_j * \mathbf{x}_j$
- Consider the encoding of $\mathbf{x}_j$ (`Pclass`) as $\{1, 2, 3\}$
  - The difference in contribution betwen "First", "Second" and "Third" are all equal
- Consider the encoding of $\mathbf{x}_j$ (`Pclass`) as $\{1, 2, 4\}$
  - The difference in contribution betwen "Second" and "Third" is twice that of "First" and "Second"

The arbitrary choice of encoding may have an impact on the prediction.

Bottom line

- Consider whether a feature should be treated as categorical *regardless* of the encoding presented
- Arbitary mapping of a categorical value to an integer has consequences
    - Avoid it !

We will describe the proper way to encode categorical variables

- And revisit the Titanic example, changing `Pclass` to categorical

One hot encoding (OHE)

Categorical variables can have more than 2 classes.

The way to represent a categorical variable v with $||C||$ classes is with a vector $\mathbf{v}$ of length $||C||$.

$\mathbf{v}$ will have the property of being all zero *except* at a single index j where $\mathbf{v}_j = 1$.

Hence the name *one hot* encoding.

We need to create a mapping m from class $c \in C$ to integers $\in [1, ||C||]$.

This will enable us to associate an integer with a class label.

For example i, suppose v has class label c.

Then $\mathbf{v^{(i)}}$ is defined as

$$\begin{aligned}
\mathbf{v}_j^{\mathbf{(i)}} &= 1 \quad \text{if } j = m(c) \\
\mathbf{v}_j^{\mathbf{(i)}} &= 0 \quad \text{if } j \neq m(c)
\end{aligned}$$

n.b., we may slip into writing $\mathbf{v}_{\boldsymbol{c}}^{\mathbf{(i)}}$ rather than $\mathbf{v}_{\boldsymbol{m(c)}}^{\mathbf{(i)}}$

- since c is a category (non-numeric) and $m(c)$ is an integer this is unambgious

The categorical variable v was replaced with $||C||$ binary variables $\mathbf{v}_1, \ldots, \mathbf{v}_{||C||}$.

- for each example i: there is exactly 1 index j such that $\mathbf{v}_j = 1$
- if feature $\mathbf{v}$ of example i was from the j^{th} class in C, then $\mathbf{v}_j^{(\mathbf{i})} = 1$

Each of the new *binary* features is called an *indicator* or *dummy* variable.

This is called **one hot encoding (OHE)** of a variable.

We can use OHE on variables, whether they are targets or features.

Let's see what $\mathbf{v}$ looks like on a number of examples, one for each possible class $c \in C$:

$$\begin{array}{rccccc} & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \cdots & \mathbf{v}_{||C||} \\ m(c) = 1 & 1 & 0 & 0 & & 0 \\ m(c) = 2 & 0 & 1 & 0 & & 0 \\ m(c) = 3 & 0 & 0 & 1 & & 0 \\ \cdots & & & & & \\ m(c) = ||C|| & 0 & 0 & 0 & & 1 \end{array}$$

To be concrete let's consider a data set with

- one numeric variable x_0
- categorical variable x_1 ("Gender") from categories $C_{(1)} = \{" \ Male \ "," \ Female \ "\}$
- categorical variable x_2 ("Location")from categories $C_{(2)} = \{" \ Urban \ "," \ Suburban \ ", " \ Exurban \ "\}$

And a few rows from our data set

$$\mathbf{X}' = \begin{pmatrix} \textbf{const} & \textbf{Gender} & \textbf{Location} \\ 1 & Female & Urban \\ 1 & Female & Exurban \\ 1 & Male & Exurban \\ 1 & Male & Suburban \\ \vdots & & \end{pmatrix}$$

After our One Hot Encoding we get

$$\mathbf{X}'' = \begin{pmatrix} \textbf{const} & \textbf{IsFemale} & \textbf{IsMale} & \textbf{IsUrban} & \textbf{IsSuburban} & \textbf{IsExurban} \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ \vdots & & & & & \end{pmatrix}$$

Notice that the number of features has increased.

Specifically: a single categorical variable x_j from a category $C = \{c_1, c_2, \ldots\}$ of size $||C||$

- has been replaced by $||C||$ new varibles
    - $\text{Is } c_1$
    - $\text{Is } c_2$
    - $\vdots$
    - $\text{Is } c_{||C||}$

Categorical features versus categorical targets

OHE can be applied to a variable, regardless of whether the variable is a feature ($\mathbf{x}$) or a targer ($\mathbf{y}$).

There are some subtle differences in practice

Categorical targets

Although we should use OHE to encode the targets, *in practice* you might see targets encoded as integers

- Binary targets as 0/1
- Other targets as integers
    - `sklearn` method `LabelEncoder` does exactly this

If it's such a bad idea: why does this happen ?

The answer

- *Mathematically* it is a bad idea
- It **may** not matter *from a coding perspective*
    - Often, the code need only be able to *distinguish between* target values
        - e.g., restrict the examples to those with a particular value of the target
    - So the encoding of values is not important
    - In fact: `sklearn` is perfectly happy with non-numeric targets for just this reason !
    - It will matter in the Deep Learning part of the course

Bottom line

- You will see distinct integer (and string) encodings of targets in `sklearn`
- When discussing the mathematics of Loss functions we will represent discrete $\mathbf{y^{(i)}}$ by a *specific encoding*
    - for Binary Classification:
        - by either $0/1$ or $-1/+1$
    - for Multinomial Classification (number of classes $||C|| > 2$)
        - by a *vector* of 0's and 1's of length $||C||$
        - *One Hot Encoding* (OHE)

So please be aware that when we encode categorical targets

- it is for a mathematical formula
- and not *necessarily* a pre-processing step you must perform on the examples

Categorical features

Much as we would like to have a Machine Learning universe in which everything was uniform

- there is one model in which OHE may cause a problem
    - Linear Regression, with an intercept
    - There is a simple fix (i.e., an argument to pass to implementations of OHE)

The issue is called the *Dummy variable* trap and will be explained in a Deep Dive.

Text: another categorial variables

How do you include text variables ? One-hot encoding of the vocabulary !

That's only approximately true, as vocabularies can be quite large and thus, the vectors are very long.

Dealing with Text will be the subject of another lecture

Example: Spam filtering (classification task: Is Spam/Is Not Spam)

- each word is an indicator
- Dealing with large vocabulary
    - sparse matrices
    - word vectors
- feature engineering: an ALLCAP feature

Recap

- We introduced methods to deal with non-numeric variables
- Unfortunately, there are some nuances

Let's

In [4]:

```
print("Done")
```

```
Done
```