Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Handling Multiple Classes

Now, we have developed a **binary** classifier, using logistic regression. But, many (almost all) problems have more than two classes!

Example: Different animals, genres of movies, sub-types of disease, etc.

0.0.1 Approaches to multi-class classification

So, we need to a way to do **multi-classing**. Consider two main approaches:

- **Train** many binary classifiers on different **classes** and **combine** them into a single model.
  - There are several ways to **combine** these **classifiers**. We won't go over them here, but some **names**: OVO (one-versus-one), OVA (one-versus-all).
- **M**ake **one** classifier that handles the multi-class problem by itself.
  - This model will be A **modified** version of logistic regression, using a variant of NLL.

The **latter** approach is what we will use in this **next** section.

Extending our Approach: One-Hot Encoding

Rather than being **restricted** to classes 0 and 1, we'll have k **distinct** classes. Our **hypothesis** will be

$$h : \mathbb{R}^d \to \{C_1, C_2, C_3, \dots C_k\}$$

Where C_i is the i^{th} class. Meaning, we want to **output** one of those k **classes**.

Because we'll be using our computer to do **math** to get the **answer**, we need to represent this with **numbers**. Before, we would simply **label** with 0 or 1.

We could return $\{1, 2, 3, 4, 5...k\}$ for each **label**. But this is **not** a good idea: it implies that there's a natural **order** to the classes, which isn't necessarily true.

If we don't **actually** think C_1 is closer to C_2 than to C_5, we probably shouldn't represent them with numbers that are **closer** to each other.

Instead, each class needs to be a **separate** variable. We can store them in a vector:

$$\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_k \end{bmatrix} \tag{1}$$

So, our **label** will be

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \tag{2}$$

In binary classification, we used 0 or 1 to indicate whether we fit into one **class**. So, that's how we'll do each class: 0 if our data point is **not** in this class, 1 if it **is**.

This approach is called **one-hot encoding**.

Definition 1

One-hot encoding is a way to represent **discrete** information about a data point.

Our k classes are stored in a length-k column **vector**. For **each** variable in the vector,

- The value is **0** if our data point is **not in that class**.
- The value is **1** if our data point is **in that class**.

In one-hot encoding, items are **never** labelled as being in **two** classes at the **same time**.

Example: Suppose that we want to classify **furniture** as table, bed, couch, or chair.

$$\begin{bmatrix} \text{table} \\ \text{bed} \\ \text{couch} \\ \text{chair} \end{bmatrix} \tag{3}$$

For each class:

$$y_{\text{chair}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad y_{\text{table}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad y_{\text{couch}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad y_{\text{bed}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$