Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

NLLM

One loose end left to tie up: our **loss function**. We need to evaluate our hypothesis, and be able to improve it.

For **binary classification**, we did **NLL**:

$$\mathcal{L}_{\text{nll}}(g, y) = -\Big(y \log g + (1-y) \log(1-g) \Big)$$

How do we make this work in **general**? Well, we want to make our two terms have a **similar** form, so we can generalize to more classes.

- g and $1-g$ are both probabilities: we can think of them as g_1 and g_2, respectively.
- If $g = g_1$, then we would expect $y = y_1$. And indeed: it gives a 1 if we're in the first class (+1).
  - Similarly, $1 - y = y_2$.

$$\mathcal{L}_{\text{nll}}(g, y) = -\Big(y_1 \log g_1 + (y_2) \log(g_2) \Big)$$

They have the **same** format now! Much tidier. And it tracks: when one **label** is correct, the other term is $y_j = 0$, and **vanishes**.

Does this **generalize** well? It turns out it does: with **one-hot encoding**, the correct label is **always** $y_j = 1$, and the incorrect labels are **all** $y_j = 0$.

So, we'll write it out:

Key Equation 1

The **loss** function for **multi-class** classification, **Negative Log Likelihood Multiclass (NLLM)**, is written as:

$$\mathcal{L}_{\text{NLLM}}(g, y) = -\sum_{j=1}^{k} y_j \log(g_j)$$

Because of **one-hot encoding**, all terms except one have $y_j = 0$, and thus **vanish**.

Using all of these functions, we can finally do gradient descent on our multi-class classifier. However, we won't go through that work in these notes.