Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Probabilities in multi-class

So, we now know our **problem**: we're taking in a data point $x \in \mathbb{R}^d$, and **outputting** one of the classes as a **one-hot vector**.

So, now that we know what sorts of data we're **expecting**, we need to decide on the formats of our **answer**.

We'll be returning a vector of length-k: **one** for each **class**. When we were doing **binary** classification, we estimated the **probability** of the positive class.

So, it should make sense to do the same **here**: for each class, we'll return the estimated **probability** of our data point being in that class.

$$g = \begin{bmatrix} \mathbf{P}\{x \text{ in } C_1\} \\ \mathbf{P}\{x \text{ in } C_2\} \\ \vdots \\ \mathbf{P}\{x \text{ in } C_k\} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix} \tag{1}$$

We can add an **additional** rule: the probabilities need to add up to **one**: we should assume our point ends up in some class or **another**.

$$g_1 + g_2 + ... + g_k = 1 \tag{2}$$

Concept 1

The different terms of our **multi-class** guess g_i represent the **probability** of our data point being in class C_i.

Because we should assume our data point is in **some** class, all of these probabilities have to **add** to 1.

Scaling values to add up to 1 is called **normalization**. How do we do that?

Well, let's say each class gets a **value** of r_i, before being **normalized**: we have done some other math we won't worry about.

To make the total 1, we'll **scale** our terms by a factor C:

$$C(r_1 + r_2 + ...r_k) = C\left(\sum_{i=1}^{k} r_i\right) = 1 \tag{3}$$

We can get our factor C just by dividing:

$$C = \frac{1}{\sum r_i} \tag{4}$$

We've got g_i now!

$$g = \begin{bmatrix} r_1/\sum r_i \\ r_2/\sum r_i \\ \vdots \\ r_k/\sum r_i \end{bmatrix} \tag{5}$$

Turning sigmoid multi-class

Now, we just need to compute r_i terms to plug in. To do that, we'll see how we did it using sigmoid:

$$g = \sigma(u) = \frac{1}{1+e^{-u}} \tag{6}$$

This function is 0 to 1, which is good for being a probability.

Just for our convenience, we'll switch to positive exponents: all we have to do is multiply by e^u/e^u.

Negative numbers are easy to mess up in algebra.

$$g = \frac{e^u}{1+e^u} \tag{7}$$

We'll think of **binary** classification as a special case of **multi-class** classification. The above probability could be thought of as g_1: the chance of our first class.

Concept 2

Binary classification is a **special** case of **multi-class** classification with only **two** classes.

So, we can use it to figure out the **general** case.

So, what was our **second** probability, $1 - g$? This will be our second class, g_2.

$$g_2 = 1 - g = \frac{1}{1+e^u} \tag{8}$$

This follows an $r_i/(\sum r_i)$ format: the numerators (1 and e^u) add to **equal** the denominator $(1+e^u)$.

$$g = \begin{bmatrix} 1/(1+e^u) \\ e^u/(1+e^u) \end{bmatrix} \tag{9}$$

How do we **extend** this to **more** classes? Well, 1 and e^u are **different** functions: this a problem. We want to be able to **generalize** to many r_i.

How do they make them **equivalent**? We could say $1 = e^0$. So, we could treat both terms as **exponentials**!

$$g_1 = \frac{e^u}{e^0 + e^u} \tag{10}$$

We can do this for an **arbitrary** number of terms. We'll treat them as **exponentials**, just like for e^u and e^0

$$g_i = \frac{r_i}{\sum r_j} = \frac{e^{u_i}}{\sum e^{u_j}} \tag{11}$$

Our Linear Classifiers

What are each of those u_i terms? When we were doing **binary**, it was a **linear regression** function:

$$u(x) = \theta^T x + \theta_0 \tag{12}$$

We can do this for multiple different u_i by just creating a different linear classifier (θ, θ_0) for each one. We'll represent each vector as $\theta_{(i)}$.

$$\theta_{(1)} = \begin{bmatrix} \theta_{1(1)} \\ \theta_{2(1)} \\ \vdots \\ \theta_{d(1)} \end{bmatrix} \qquad \theta_{(2)} = \begin{bmatrix} \theta_{1(2)} \\ \theta_{2(2)} \\ \vdots \\ \theta_{d(2)} \end{bmatrix} \qquad \theta_{(k)} = \begin{bmatrix} \theta_{1(k)} \\ \theta_{2(k)} \\ \vdots \\ \theta_{d(k)} \end{bmatrix} \tag{13}$$

And each can be used on our input:

$$u_1(x) = \theta_{(1)}^T x + \theta_{0(1)} \qquad u_2(x) = \theta_{(2)}^T x + \theta_{0(2)} \dots \tag{14}$$

But this is tedious. Instead, we can combine them all into a $(d \times k)$ **matrix**.

k classes, so we need k classifiers. We'll stack them side-by-side like how we stacked multiple data points to create X.

$$\theta = \begin{bmatrix} \theta_{(1)} & \theta_{(2)} & \cdots & \theta_{(k)} \end{bmatrix} = \begin{bmatrix} \theta_{1(1)} & \theta_{1(2)} & \cdots & \theta_{1(k)} \\ \theta_{2(1)} & \theta_{2(2)} & \cdots & \theta_{2(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{d(1)} & \theta_{d(2)} & \cdots & \theta_{d(k)} \end{bmatrix} \tag{15}$$

And our constants, θ_0, in a $(k \times 1)$ matrix:

$$\theta_0 = \begin{bmatrix} \theta_{0(1)} \\ \theta_{0(2)} \\ \vdots \\ \theta_{0(k)} \end{bmatrix} \tag{16}$$

Concept 3

We can combine **multiple classifiers** $\Theta_{(i)} = \left(\theta_{(i)}, \theta_{0(i)}\right)$ into large **matrices** θ and θ_0 to compute **multiple** outputs u_i at the **same** time.

This will put all of our terms into a $(1 \times k)$ vector u.

$$u(x) = \theta^T x + \theta_0 = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix} \tag{17}$$

Softmax

We now have all the pieces we need. Our **linear regression** for each class:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix} = \theta^T x + \theta_0 \tag{18}$$

The **exponential** terms, to get **logistic** behavior:

$$r_i = e^{u_i} \tag{19}$$

The **averaging** to get probability = 1:

$$g = \begin{bmatrix} r_1 / \sum r_i \\ r_2 / \sum r_i \\ \vdots \\ r_k / \sum r_i \end{bmatrix} \tag{20}$$

And so, our multiclass function is...

Definition 4

The **softmax function** allows us to calculate the probability of a point being in each class:

$$g = \begin{bmatrix} e^{u_1} / \sum e^{u_i} \\ e^{u_2} / \sum e^{u_i} \\ \vdots \\ e^{u_k} / \sum e^{u_i} \end{bmatrix}$$

Where

$$u_i(x) = \theta_{(i)}^T x + \theta_{0(i)} \tag{21}$$

If we are forced to make a **choice**, we choose the class with the **highest probability**: we return a **one-hot encoding**.