Why predict in probabilities rather than discrete labels?

- Due to smoothness
- We can't change discrete labels "a tiny bit" — it's all or nothing
- But we can change probabilities "a tiny bit"

So we have probabilities. Instead of learning discrete labels, we try to predict the probability distribution.

Methods that learn discrete labels directly → Discriminative models because they just want to discriminate between different classes.

→ Generative methods; they can generate samples. You can get discrete labels because of conditional probability.

So think we would have values like 2.5, -1.7. The sum of these won't be equal to 1, which violates the properties of a probability distribution.

That's why we need softmax → function that takes these inputs and outputs probabilities that are positive and sum to 1.

So how do you make a number positive?

z^2 (z raised to power 2)

1. $|z|$
2. max(0,z)
3. exp(z) - It's one-to-one (input to output is unique) and onto (maps to entire range)exp maps the entire number line to the positive part of real numbers, i.e., negative to positive

Now how to make them sum to one?

Answer: normalization, i.e., $z_1/(z_1+z_2)$

So let's define softmax functions:

The softmax function takes all the inputs, exponentiates them, and divides by their sum.

$$\mathrm{softmax}(f_{\mathrm{moe}}(x), f_{\mathrm{eat}}(x)) = \frac{\exp(f_{\mathrm{moe}}(x))}{\exp(f_{\mathrm{moe}}(x)) + \exp(f_{\mathrm{eat}}(x))}$$

Numerator makes it positive and denominator makes it sum to 1.

In vector form:

$$p(y = i|x) = \mathrm{softmax}(f_\theta(x))_i = \frac{\exp(f_{\theta_i}(x))}{\sum_{j=1}^{n} \exp(f_{\theta_j}(x))}$$

Why named softmax?

It behaves like a max when numbers going into it are very large, and for smaller values it has steeper slope.