**Linear Classification**

**Regression vs classification:** Regression gives continuous values (number) whereas classification gives discrete values (labels).

**Linear Regression:**

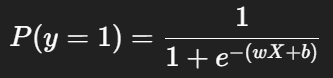
* **Type**: Regression
* **Output**: Continuous number (e.g., 45.7, 100.3)
* **Equation**:

y=wX+b

* **Use case**: Predict house prices, salaries, temperatures.

**Logistic Regression:**

* **Type**: Classification
* **Output**: Probability (between 0 and 1), then mapped to a class (0/1).
* **Equation**:



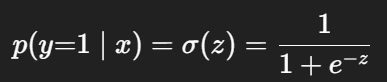
* **Use case**: Spam vs not spam, disease vs no disease, buy insurance vs not buy.

**Logistic Regression Theory**

**Logistic Regression:**

* Logistic Regression is a **linear classifier** that predicts the **probability** of class 1 (e.g., spam) given features x.
* It computes a linear score *z=w^t\*x + b* and then **squashes** that score into *[0,1][0,1]*[0,1] with the **sigmoid**.
* Decision: choose a threshold (usually 0.5) to convert probability → class label.

**Formula:**



**Why Logistic Regression:**

* Linear Regression isn’t suitable for classification (can predict values <0 or >1).
* Logistic regression fixes this using the sigmoid function to bound output in [0,1].

**SoftMax for Multiclass**

**Why SoftMax:**

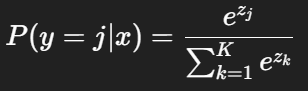
* Logistic regression works well for binary classification (0/1).
* But for multi-class classification (e.g., classifying an image into cat, dog, panda), we need something more powerful.
* That’s where SoftMax Regression (also called Multinomial Logistic Regression) comes in.

**Softmax Function:**

* Given raw scores (**logits**) for each class:



* The **softmax function** converts these scores into probabilities:



* e^zj makes all scores positive.
* Division by the sum makes sure all probabilities add up to **1**.
* Each P(y=j∣x) lies in [0,1].

Example:  
 Scores = [2.0, 1.0, 0.1]  
 Softmax = [0.65, 0.24, 0.11]  
 (So class 1 has the highest probability)

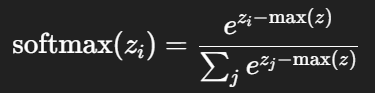
**Loss Function:**

* Softmax gives probabilities for each class.  
   Example (3 classes): [0.1, 0.7, 0.2]
* **Categorical Cross-Entropy** Loss looks only at the probability of the true class.  
   Formula for one sample:



**Numerical Stability:**

* In practice, we compute softmax as:



* Subtracting max(z) prevents overflow when logits are large.

**Connection to Training:**

* During training: we use **softmax + cross-entropy** as the loss.
* During testing: we only use **softmax** (no loss) to make predictions.