**Linear Classification**

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

**Linear Regression:**

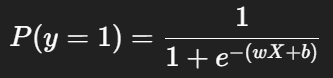
* **Type**: Regression
* **Output**: Continuous number (e.g., 45.7, 100.3)
* **Equation**:

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**:



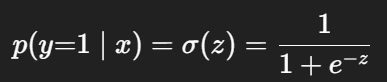
* **Use case**: Spam vs not spam, disease vs no disease, buy insurance vs not buy.

**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:**

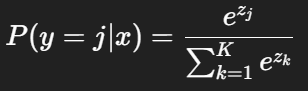
* Logistic regression works well for binary classification (0/1).
* But for multi-class classification (e.g., classifying an image into cat, dog, panda), we need something more powerful.
* That’s where SoftMax Regression (also called Multinomial Logistic Regression) comes in.

**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)

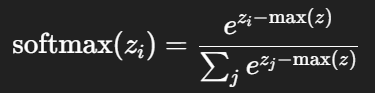
**Loss Function:**

* 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.

**Support Vector Machine (SVM)**

**What is SVM:**

* SVM is a supervised machine learning algorithm used for classification (mainly) and sometimes regression.
* It tries to find the best line (in 2D), plane (in 3D), or hyperplane (in higher dimensions) that separates different classes of data.
* It draws a hyperplane in n dimensions such that it maximizes the margin between different classification groups.

**Support Vectors:**

* Support vectors are the data points that lie closest to the decision boundary (the separating hyperplane).
* They are the most “difficult” points to classify, because they are near the margin.
* These points are critical, because if you remove them, the boundary could shift.

**What is Gamma:**

Gamma in SVM tells the model how much influence each data point has.

* **High Gamma:** Each point has small, close influence → decision boundary becomes very tight/complex around the data, might overfit.
* **Low Gamma:** Each point has large, far influence → decision boundary becomes smoother/simple.

**What is Kernal:**

* The kernel defines the *type of decision boundary (shape of separation)* the SVM will try to create.
* Think of it as the formula or function used to map data into a higher-dimensional space.
* So, kernel = what kind of curve/line to draw.

**Key Terms:**

* **Hyperplane**: The decision boundary (line in 2D, plane in 3D).
* **Margin**: Distance from the hyperplane to the nearest data points of each class.
* **Support Vectors**: The data points that are closest to the boundary (they “support” the decision boundary).
* **Kernel**: A trick to let SVM classify data that is not linearly separable by mapping it into higher dimensions. (e.g., RBF kernel).

**SVM Kernels and Intuitions**

**What are Kernels:**

* Sometimes data cannot be separated by a straight line.
* A kernel is a function that transforms data into a higher-dimensional space where it *can* be separated by a hyperplane (line/plane).
* Instead of manually creating new features, the kernel trick lets SVM do it mathematically behind the scenes.

**Linear Kernels:**

* No transformation, just the dot product.
* Works when data is linearly separable.
* Decision boundary = straight line/plane.

**Formula:**

***K(x,x’) = x.x’***

**Polynomial Kernels:**

* Creates polynomial combinations of features.
* Allows curved boundaries.

**Formula:**

***K(x,x’) = (x.x’ +*** ***c)d***

***c=*** *constant to keep the values positive*

***d=*** *degree of polynomial*

**Example:** With degree=2, it can separate circular patterns.

**RBF Kernels:**

* Most popular.
* Maps data into infinite-dimensional space.
* Creates complex, flexible boundaries.

**Formula:**

***K(x,x’) =exp (-* γ || *x-x’ ||2)***

***exp:*** *shrink the distance to keep the similarity score between 0 and 1 instead of Euclidean distance.*

* Controlled by **gamma**

**Sigmoid Kernels:**

* Similar to neural networks’ activation.

**Formula:**

***K(x,x’) =tanh* (α (*x.x’) + c)***

* Rarely used in practice compared to RBF.

**Intuition Recap:**

* **Linear** → Straight line separation.
* **Polynomial** → Curved boundaries.
* **RBF** → Flexible, handles almost any shape.
* **Sigmoid** → Neural network–like boundaries.