Mathematical Foundations of Linear Classification

Tan N. Nguyen

The University of Information Technology, VNU-HCMC

November 9, 2025

Outline

Definition and Examples

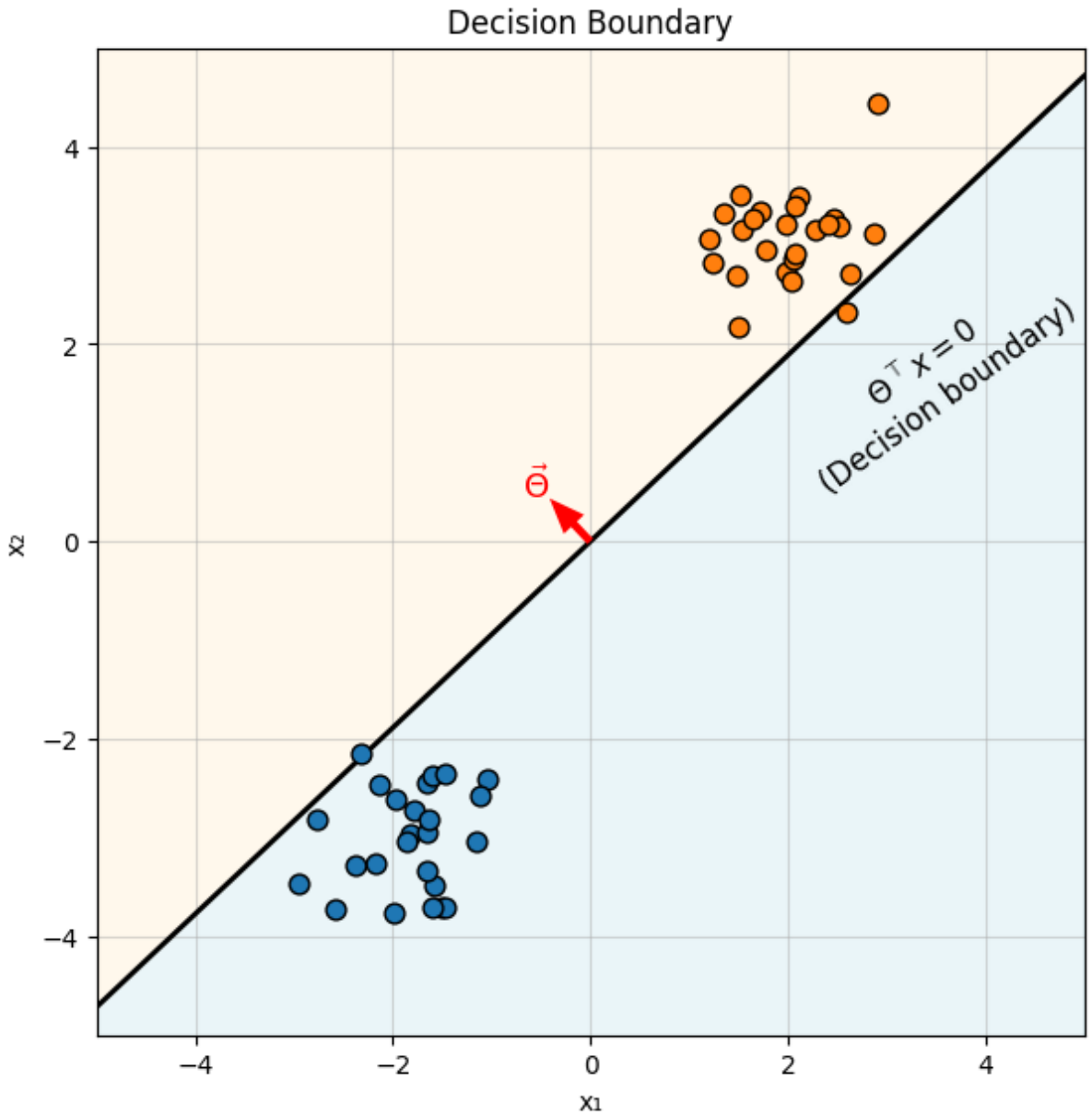
- **Definition:** Binary classification assigns samples to one of two classes.
- **Examples:**
  - Spam email detection (Spam / Not Spam)
  - Sentiment analysis (Positive / Negative)

Hyperplane and Decision Boundary

- **Hyperplane:** Linear boundary separating two classes.
- Mathematically: $w^T x + b = 0$
- Classification:

$$\begin{cases} w^T x + b > 0, & \text{Class 1} \\ w^T x + b < 0, & \text{Class 2} \end{cases}$$

Visualization of Hyperplane



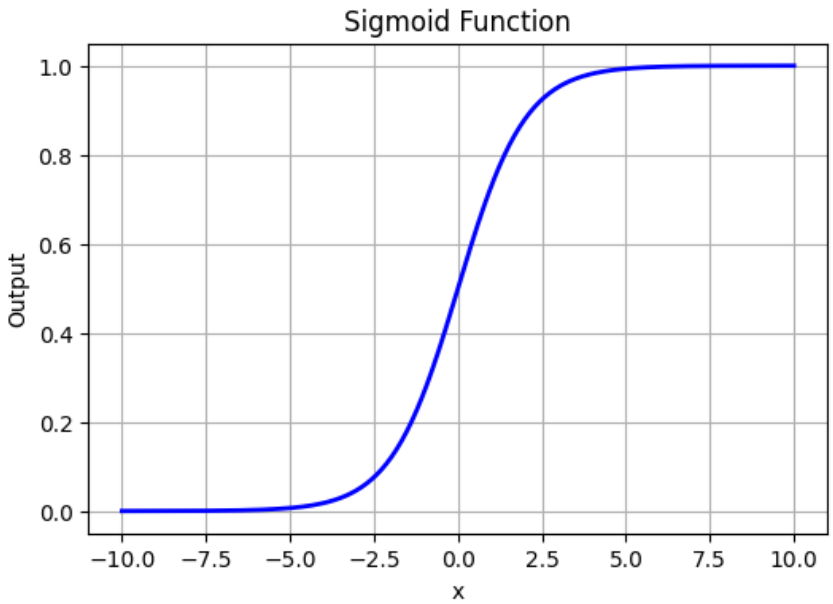
Weights, Hypothesis Function, and Evaluation

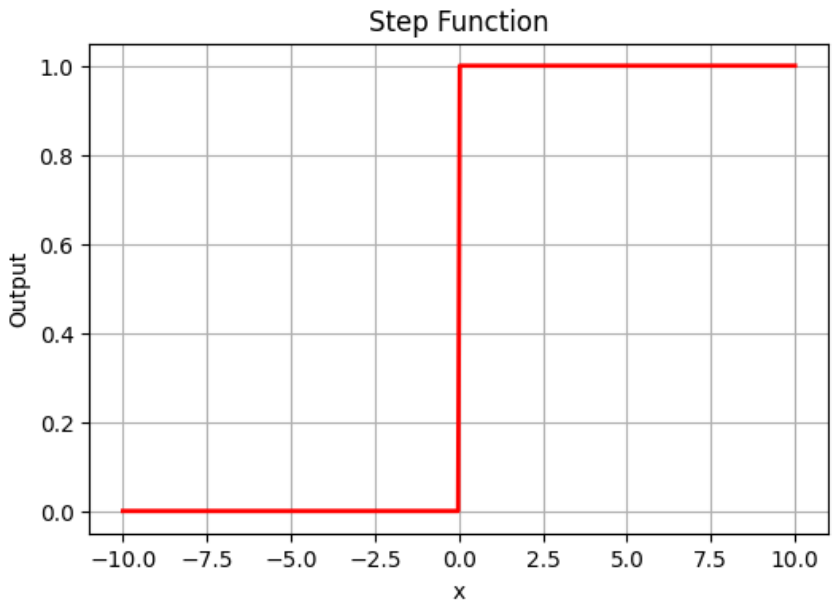
- **Weights (w):** Define direction of separating hyperplane.
- **Hypothesis Function:** $h(x) = \text{sign}(w^T x + b)$
- **Evaluation:** Compare $h(x)$ with true labels y to update weights.

Introduction to Logistic Regression and PLA

- Logistic Regression maps outputs to probabilities.
- Perceptron Learning Algorithm (PLA) uses step function instead of sigmoid.

Activation Functions Visualization





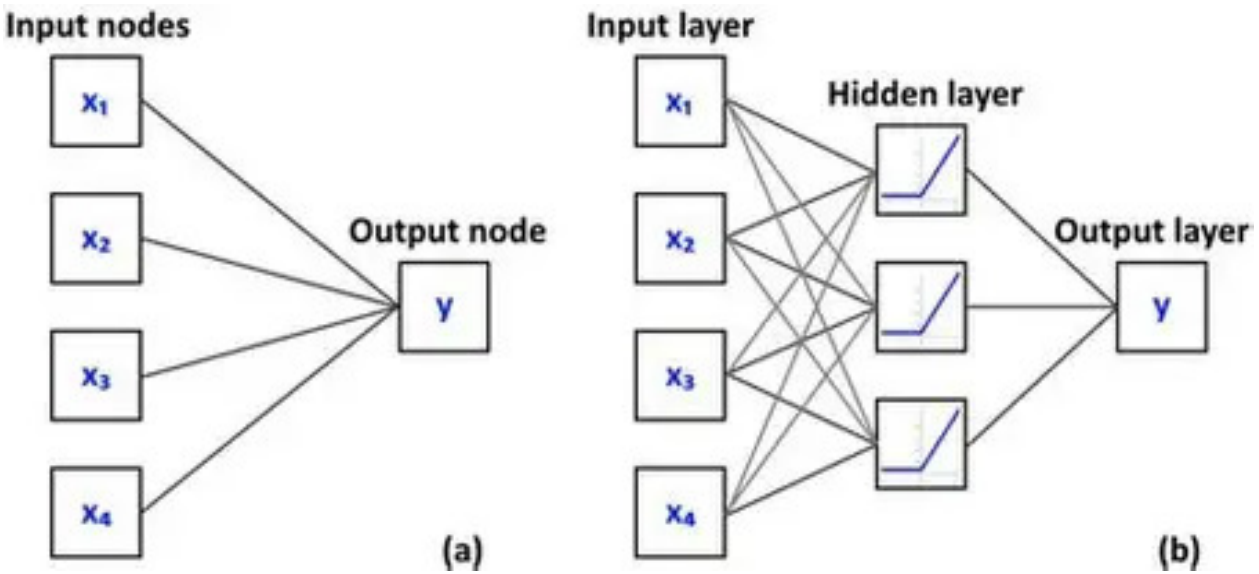
Definition and Types

Perceptron

A perceptron is a simple linear model that classifies data by learning the optimal separating hyperplane.

- Single-layer perceptron: One neuron; linearly separable data.
- Multi-layer perceptron (MLP): Multiple layers; can learn non-linear boundaries.

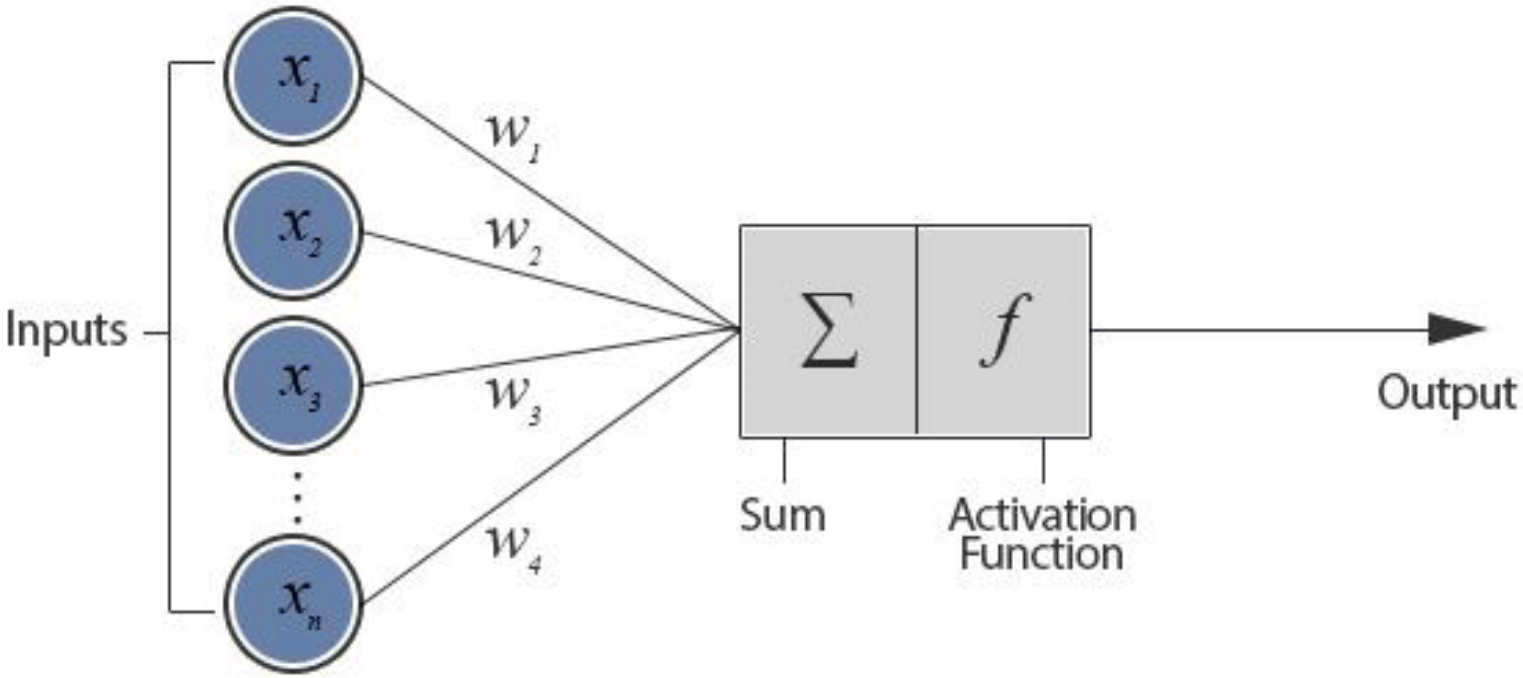
Perceptron Illustration



Basic Components

1. Inputs $(x_1, \ldots, x_n)$
2. Weights $(w_1, \ldots, w_n)$
3. Summation: $z = \sum_i w_i x_i + b$
4. Activation: $y = \text{sign}(z)$

Perceptron Network Structure



Perceptron Loss Function

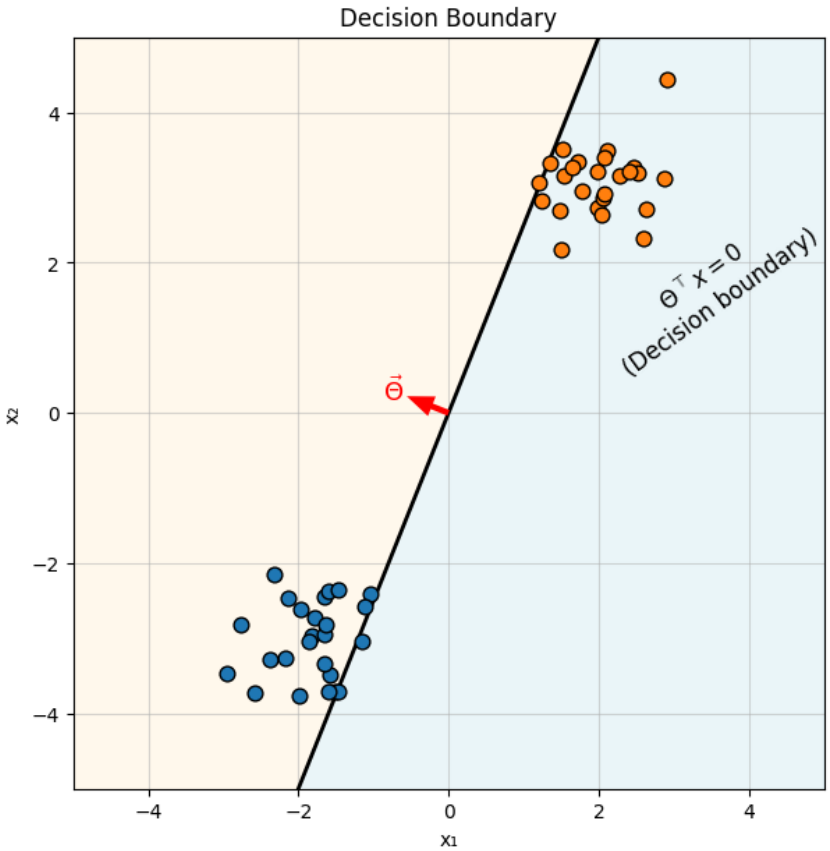
$$L(w) = -\sum_{i \in \mathcal{M}} y_i (w^\top x_i), \quad \mathcal{M} = \{i : y_i (w^\top x_i) \leq 0\}$$

- Loss decreases as $y_i (w^\top x_i)$ increases.

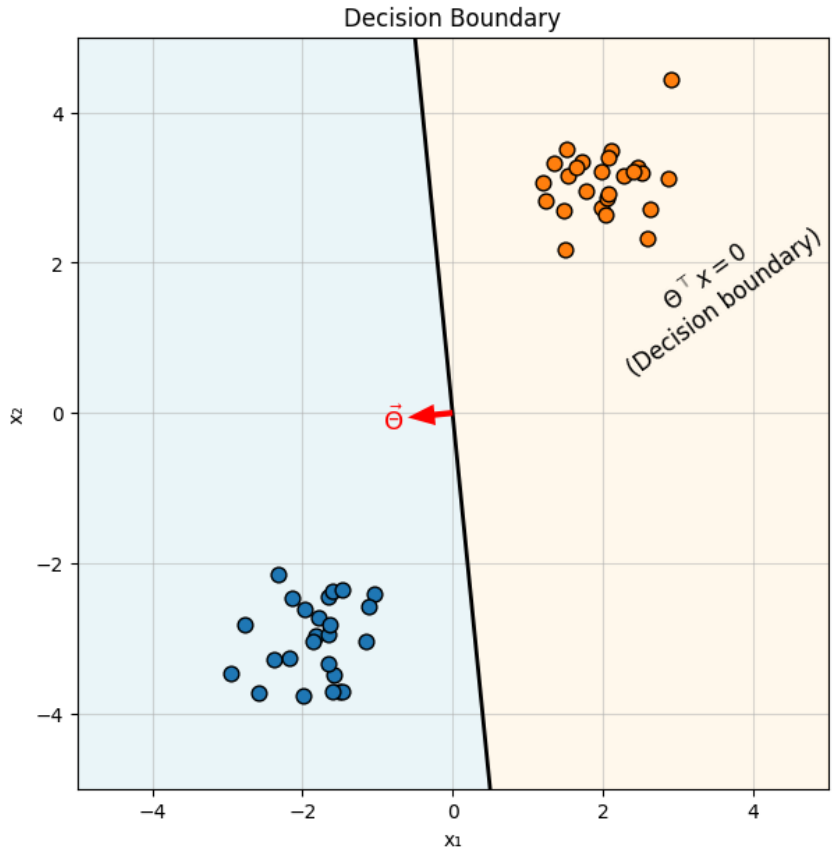
Perceptron Update Rule

- Misclassified sample (x_i, y_i): $y_i(w_t^\top x_i) \leq 0$
- Update: $w_{t+1} = w_t + \eta y_i x_i$
- Margin after update: $y_i(w_{t+1}^\top x_i) = y_i(w_t^\top x_i) + \eta \|x_i\|^2$
- Conclusion: Weight moves toward misclassified sample, increasing margin.

Visualization of Update



Before update



After update

Limitations of the Perceptron

- Only works for linearly separable datasets.
- Struggles with noisy or overlapping data.
- Produces hard decisions (no probabilities).

Conclusion

- Perceptron laid the groundwork for neural networks.
- Introduced iterative weight adjustment.
- Inspired models like Logistic Regression and SVM.

Exponential Family

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

- η — natural parameter
- $T(y)$ — sufficient statistic
- $a(\eta)$ — log-partition function
- $b(y)$ — base measure

$$E[Y;\eta] = a'(\eta), \quad Var(Y;\eta) = a''(\eta)$$

Bernoulli Distribution in GLM Form

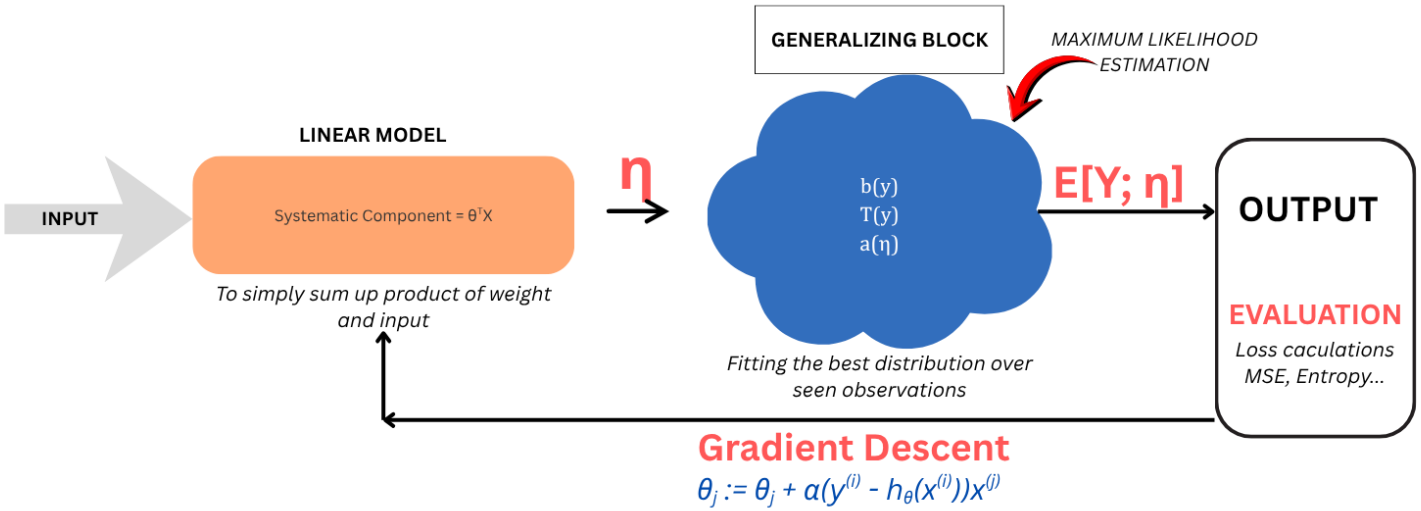
$$p(y;\phi) = \phi^y(1-\phi)^{1-y} = \exp(y \log \frac{\phi}{1-\phi} + \log(1-\phi))$$

- $T(y) = y$
- $\eta = \log \frac{\phi}{1-\phi}$
- $a(\eta) = \log(1+e^\eta)$
- $b(y) = 1$

GLM Components

1. Random component: $Y|X \sim ExponentialFamily(\eta)$
2. Systematic component: $\eta = \theta^T x$
3. Link function: $g(E[Y]) = \eta$
   - Canonical response: $\mu = E[Y;\eta] = g^{-1}(\eta) = a'(\eta)$
   - Canonical link: $\eta = g(\mu)$

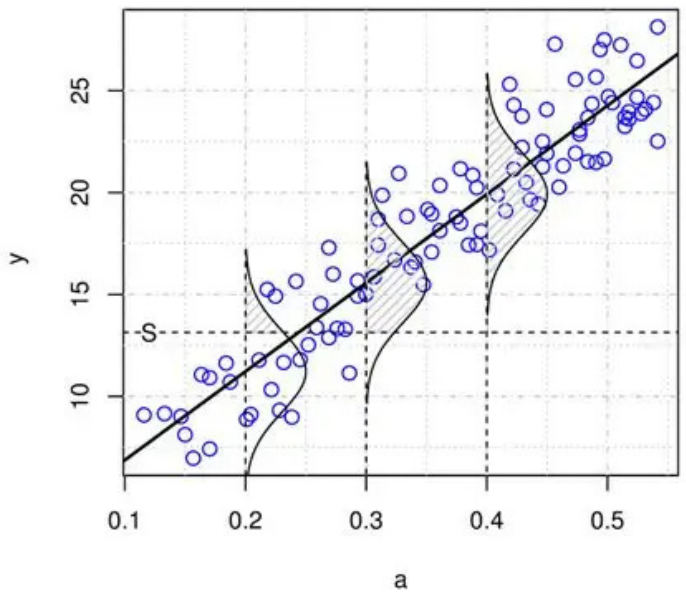
GLM Visualization

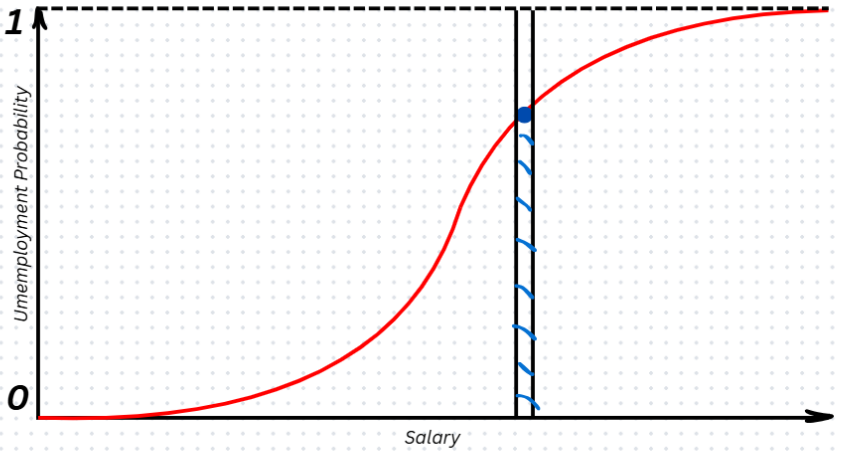


Logistic Regression as a GLM

- $Y \in \{0, 1\}$, $Y \sim Bernoulli(\phi)$
- $\eta = w_0 x_0 + \cdots + w_n x_n$
- Link: $g(\phi) = \log \frac{\phi}{1-\phi} = \eta$
- Inverse link: $\phi = g^{-1}(\eta) = \frac{1}{1+e^{-\eta}}$

Examples of Distributions





Common GLMs: Link and Prediction

Distribution	**Link Function** $g(\mu)$	**Prediction** $\mu = g^{-1}(\eta)$
Normal	Identity	η
Bernoulli	Logit	$\frac{1}{1+e^{-\eta}}$
Poisson	Log	e^{η}
Gamma	Inverse	$1/\eta$

Choosing Probability Distributions

Type of Outcome	**Suggested Distribution**
Real numbers	Gaussian
Binary	Bernoulli
Count	Poisson
Real positive	Gamma / Exponential
Proportions	Beta
Probability vectors	Dirichlet
Bayesian priors	Various

What is Softmax Regression?

- Softmax Regression generalizes Logistic Regression for **multi-class classification** ($K > 2$).
- Each class $k \in \{1, 2, \ldots, K\}$ has a linear score:

$$\eta_k = \mathbf{w}_k^\top \mathbf{x}.$$

- Goal: convert scores η_k to probabilities ϕ_k such that $\sum_k \phi_k = 1$.

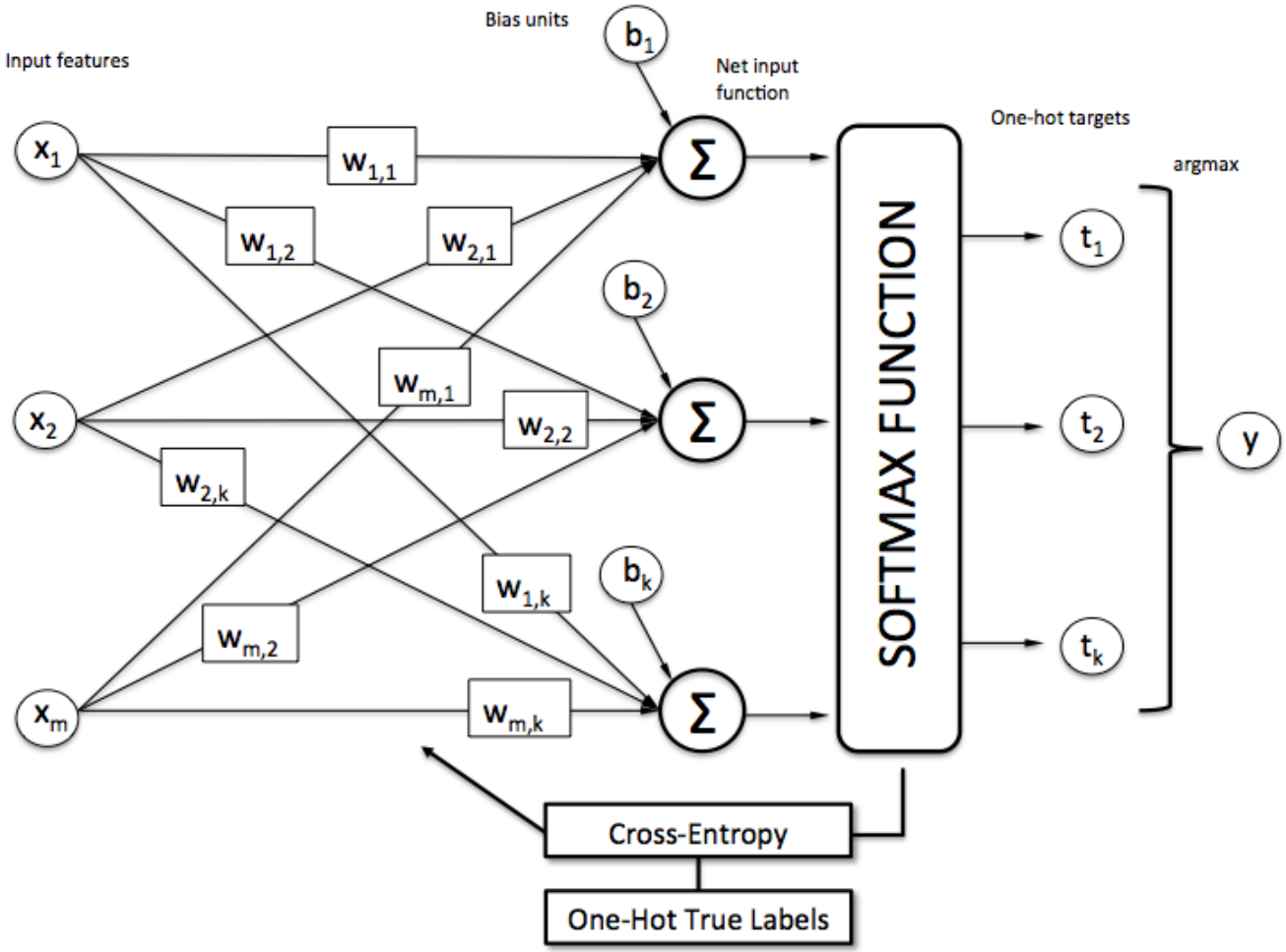
Softmax Function and Probabilities

- Probability for class k:

$$\phi_k = P(Y = k|\mathbf{x}) = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}}$$

- Properties:
  1. $\phi_k > 0$ and $\sum_{k=1}^{K} \phi_k = 1$
  2. Increasing η_k increases ϕ_k
- Inverse link: from η_k to ϕ_k (Softmax is canonical link for Multinomial GLM)

Softmax Network Structure



- Input features $\mathbf{x} \rightarrow$ linear layer producing $\eta_1, \ldots, \eta_K \rightarrow$ softmax layer $\rightarrow$ probabilities $\phi_1, \ldots, \phi_K$
- Output neuron k corresponds to class k

Loss Function (Cross-Entropy)

- Encode true label with one-hot vector $\mathbf{y}_i \in \{0, 1\}^K$
- Log-likelihood for N samples:

$$\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \phi_k(\mathbf{x}_i)$$

- Equivalently, minimize cross-entropy loss:

$$J(\mathbf{W}) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \phi_k(\mathbf{x}_i)$$

Gradient and Update Rule

- Gradient w.r.t $\mathbf{w}_k$:

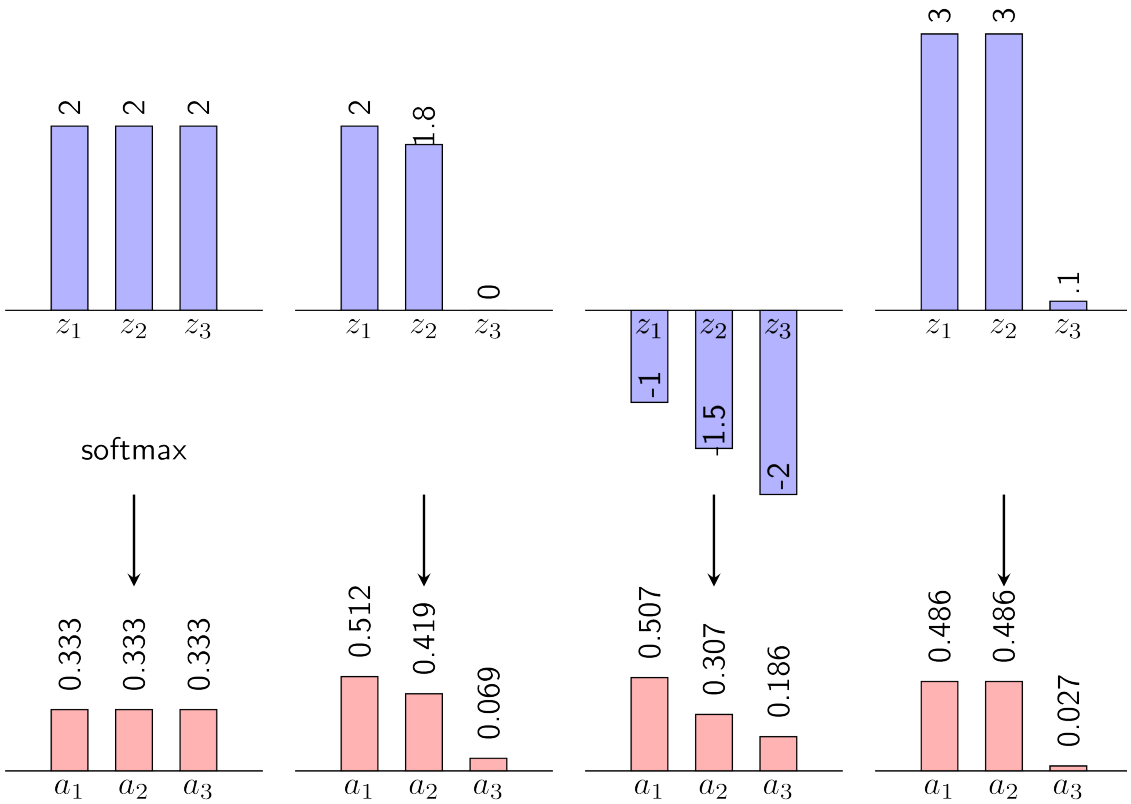
$$\frac{\partial J}{\partial \mathbf{w}_k} = \sum_{i=1}^{N} (\phi_k(\mathbf{x}_i) - y_{ik})\mathbf{x}_i$$

- Update (Gradient Descent):

$$\mathbf{w}_k \leftarrow \mathbf{w}_k - \eta \frac{\partial J}{\partial \mathbf{w}_k}$$

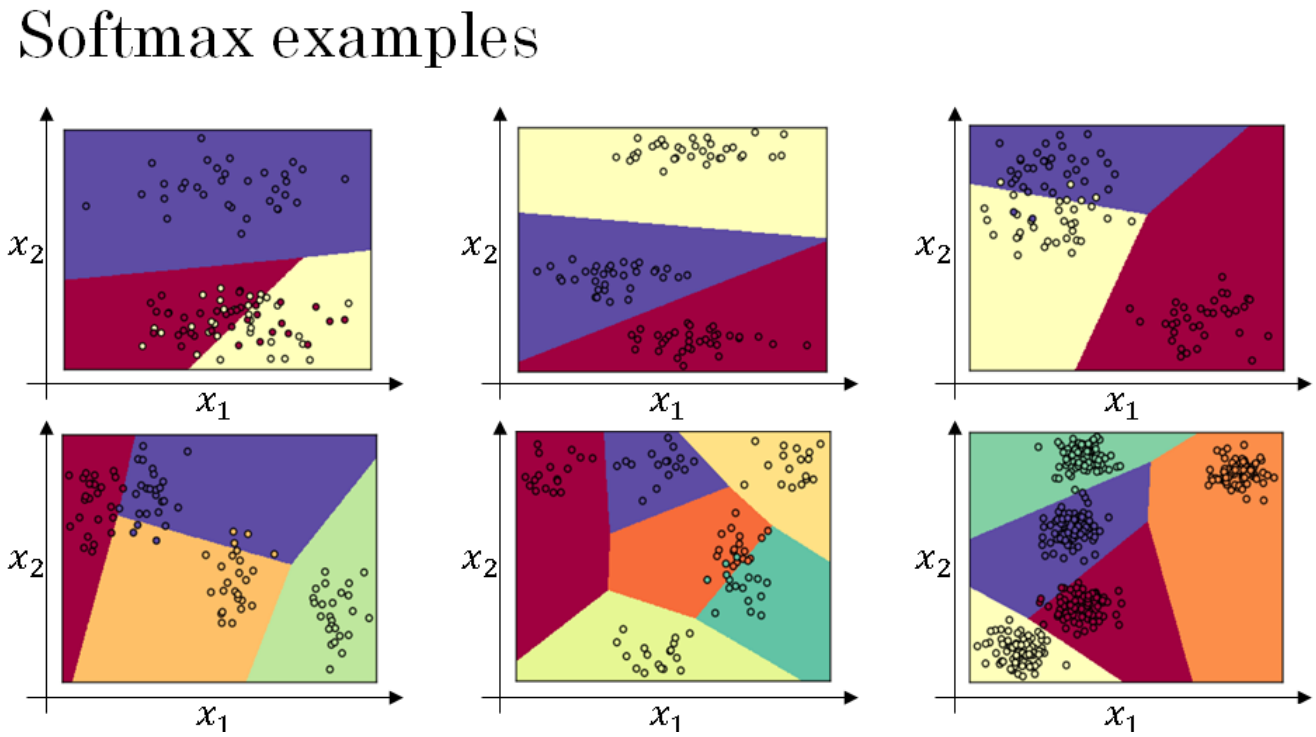
- Interpretation: adjust weights to reduce difference between predicted probability and true label

Output Probabilities Visualization



- Each input produces a probability vector $\phi_1, \ldots, \phi_K$
- Predicted class $= \text{argmax}_k \phi_k$

Decision Boundaries



- A point $\mathbf{x}$ is classified as class j if $\phi_j(\mathbf{x}) \geq \phi_k(\mathbf{x})$ for all $k \neq j$
- Equivalently: $(\mathbf{w}_j - \mathbf{w}_k)^\top \mathbf{x} \geq 0 \rightarrow$ boundaries are linear hyperplanes

Decision Boundaries

- The model predicts class j if $\phi_j \geq \phi_k$ for all $k \neq j$.
- This is equivalent to:

$$\frac{e^{\mathbf{w}_j^\top \mathbf{x}+b_j}}{\sum e^{\eta_l}} \geq \frac{e^{\mathbf{w}_k^\top \mathbf{x}+b_k}}{\sum e^{\eta_l}}$$

- Eliminating the denominator:

$$e^{\mathbf{w}_j^\top \mathbf{x}+b_j} \geq e^{\mathbf{w}_k^\top \mathbf{x}+b_k}$$

- Taking the natural logarithm of both sides:

$$\mathbf{w}_j^\top \mathbf{x} + b_j \geq \mathbf{w}_k^\top \mathbf{x} + b_k$$

- Rearranging the terms:

$$(\mathbf{w}_j - \mathbf{w}_k)^\top \mathbf{x} + (b_j - b_k) \geq 0$$

- This equation defines a **Linear Hyperplane**. Thus, Softmax Regression produces **linear** decision boundaries between any pair of classes.

Summary of Softmax Regression

- Random component: $Y \sim Multinomial(\phi_1, \ldots, \phi_K)$
- Systematic component: linear scores $\eta_k = \mathbf{w}_k^\top \mathbf{x}$
- Link function: canonical link = softmax
- Loss: cross-entropy, optimized by gradient descent
- Linear decision boundaries separate classes in input space

Thank You!

Tan N. Nguyen