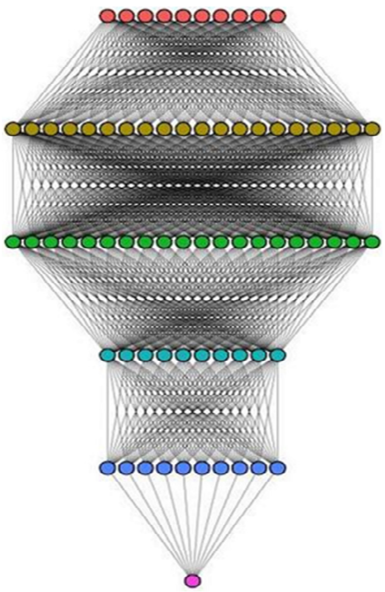
Perceptron

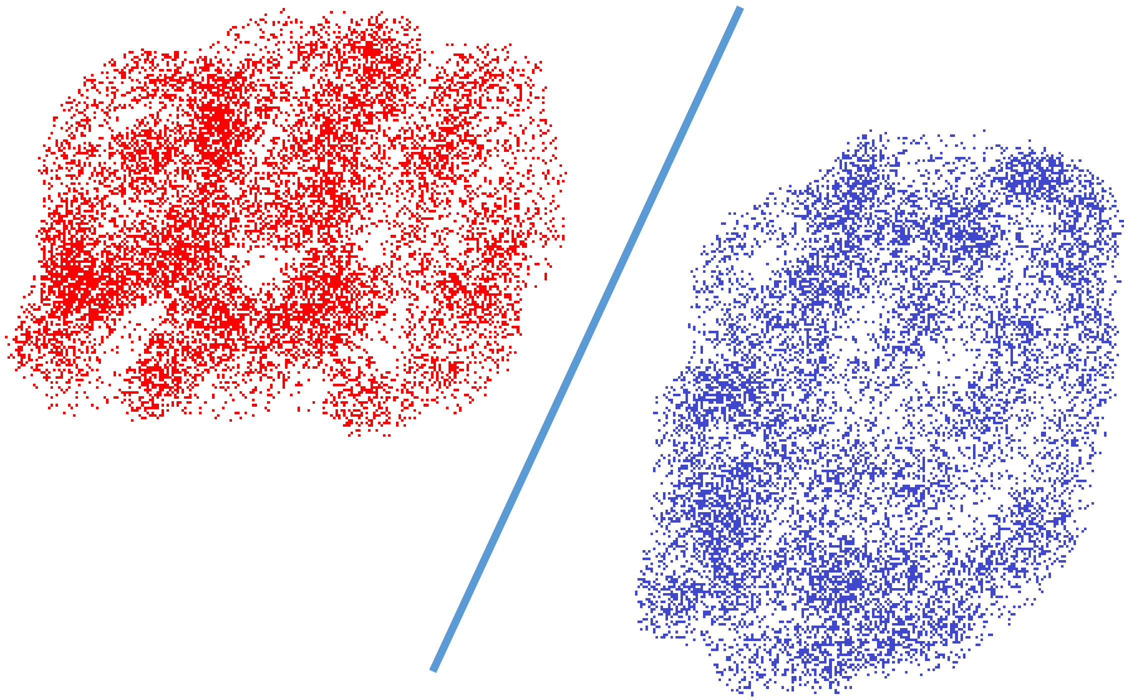
Prithwijit Guha
Dept. of EEE, IIT Guwahati

Supervised Learning



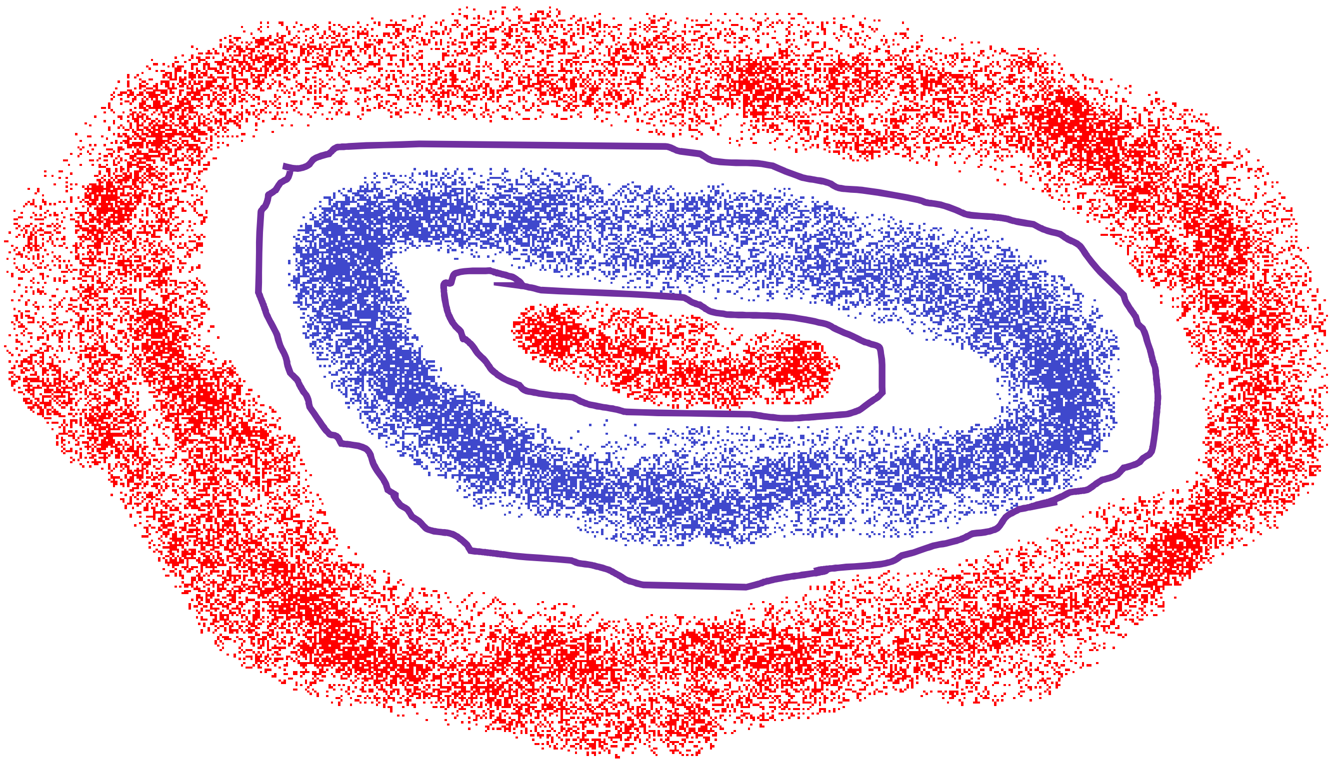
- Bayesian Classification, MAP, Chebyshev Inequality
- Performance Measures, Confusion Matrix, ROC Curves
- Logistic Regression
- Perceptron
- Multi-Layer Perceptron (MLP), ELM
- MLP Architectures, Learning, Interpretations
- Non-parametric Methods and K-NN
- Radial Basis Function Neural Networks
- Data Balancing; SMOTE & Weighted Loss Functions
- Classification & Regression Trees
- Support Vector Machines & Multiple Kernel Learning
- Ensemble Methods, Bagging and Boosting

Linearly Separable



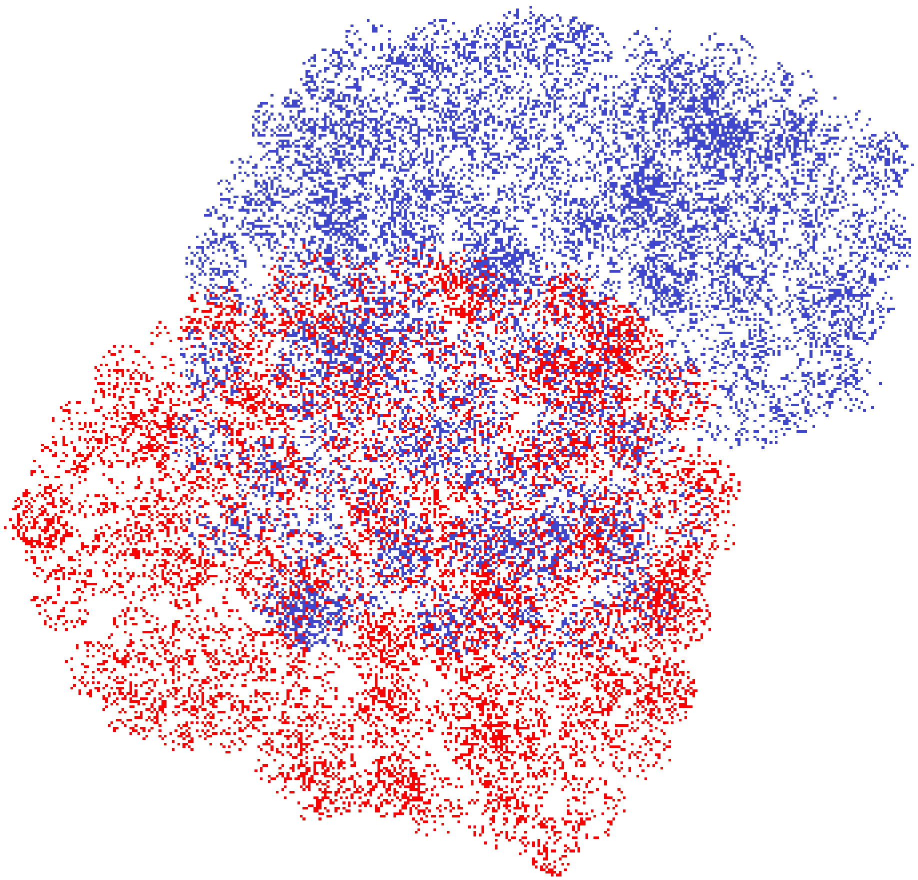
Hyperplane(s) can Separate Features Extracted from Two (Or More) Classes

Not Linearly Separable



Nonlinear Hypersurfaces Separate Features Extracted from Two (Or More) Classes

Not Separable



Require Further Transformations for Enhancing Separability

Binary Classification: Input-Output Specification

$$\boldsymbol{S_b} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$$

$$\boldsymbol{x} \in \mathbb{R}^d \qquad\qquad y \in \{0,1\}$$

$$\boldsymbol{x} \longrightarrow \boxed{\text{Classifier } \boldsymbol{C_b}} \longrightarrow \hat{y} = \boldsymbol{C_b}(\boldsymbol{x})$$

Multi-Category Classification: Input-Output Specification

$$\boldsymbol{S_m} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$$

$$\boldsymbol{x} \in \mathbb{R}^d$$

$$y \in \{1, \dots m\}$$

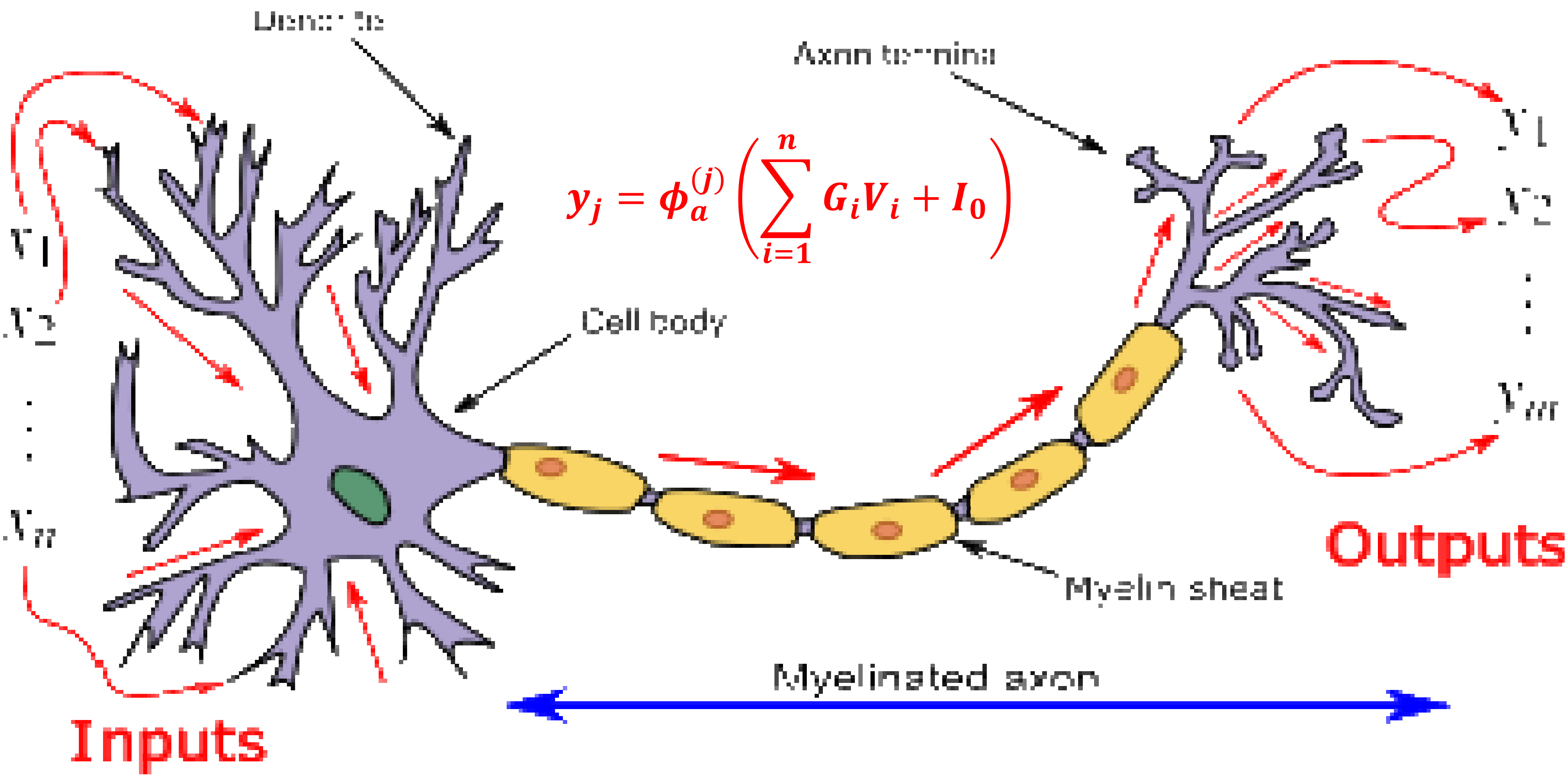
Classifier $\boldsymbol{C_m}$

$$\hat{\boldsymbol{y}} = \boldsymbol{C_m}(\boldsymbol{x})$$

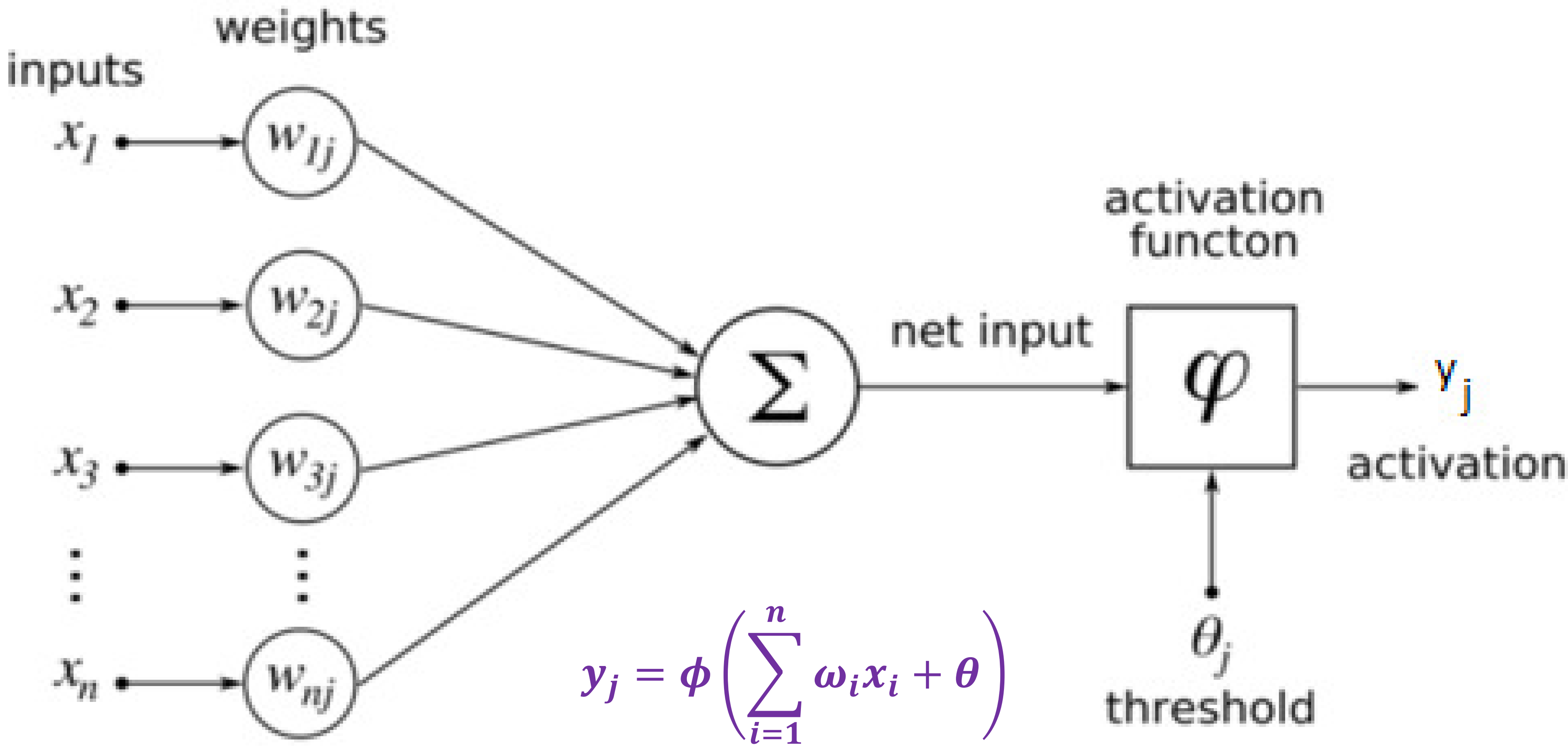
One Hot Output Encoding

$$\boldsymbol{y}_i[j] = \begin{cases} 1, & y_i = j \\ 0, & y_i \neq j \end{cases}$$

Perceptron: Biological Motivation



Perceptron: Mathematical Formulation



$$y_j = \phi\left(\sum_{i=1}^{n} \omega_i x_i + \theta\right)$$

Perceptron: Classifier & Feature Extractor

$$\boldsymbol{S_b} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$$

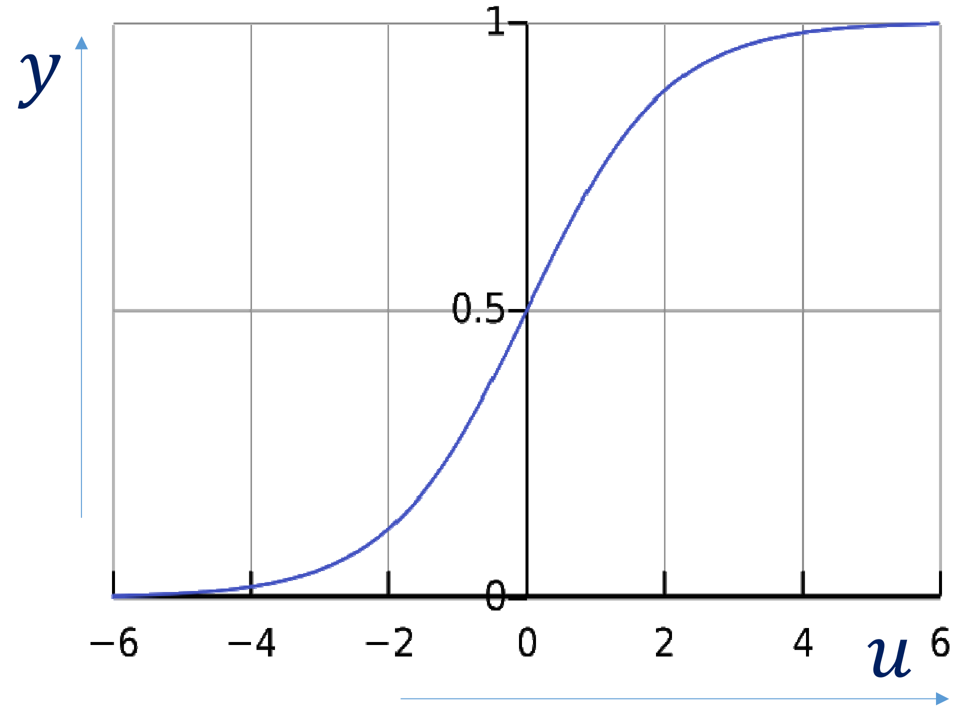
$$\hat{y} = P(\boldsymbol{x}; \boldsymbol{\omega}, b)$$

$$P : \mathbb{R}^d \rightarrow (0,1)$$

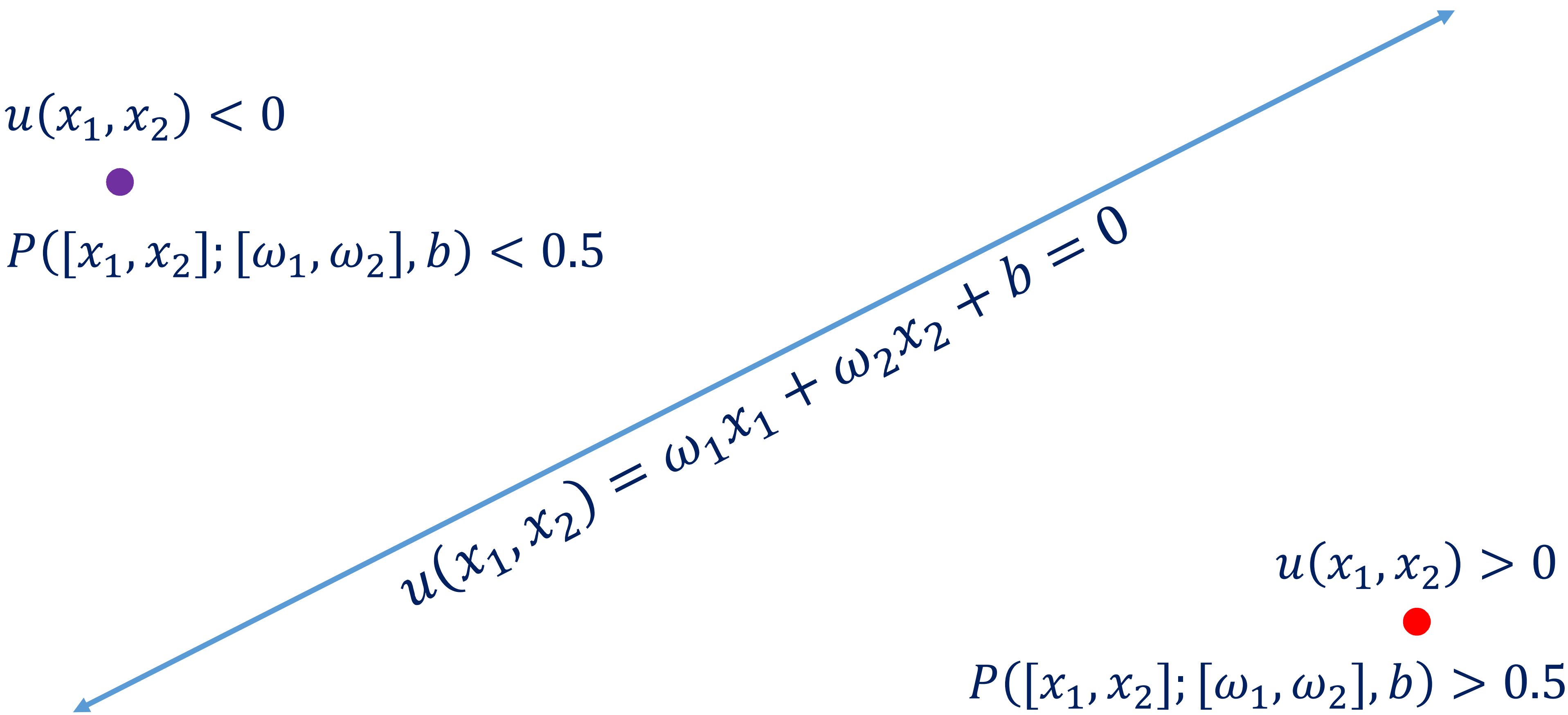
$$u_i = \boldsymbol{\omega}^T \boldsymbol{x}_i + b$$

$$u_i = \sum_{j=1}^{d} \boldsymbol{\omega}_j \boldsymbol{x}_{ij} + b$$

$$\hat{y}_i = \frac{1}{1 + e^{-u_i}}$$



Perceptron: Linear Algebra Interpretation



Perceptron: Learning by Gradient Descent

$$\boldsymbol{S_b} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$$

$$e_i = y_i - \hat{y}_i = y_i - P(\boldsymbol{x}_i; \omega, b) \Rightarrow E = \frac{1}{n}\sum_{i=1}^{n} e_i^2$$

$$\omega_j^{(k+1)} = \omega_j^{(k)} - \eta_{jk}\frac{\partial E}{\partial \boldsymbol{\omega}_j}$$

$$b^{(k+1)} = b^{(k)} - \eta_{jk}\frac{\partial E}{\partial b}$$

Perceptron: Learning by Gradient Descent

$$E = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \quad \Longrightarrow \quad \frac{\partial E}{\partial \boldsymbol{\omega}_j} = \frac{1}{n}\sum_{i=1}^{n} 2e_i \frac{\partial e_i}{\partial \boldsymbol{\omega}_j}$$

$$e_i = y_i - \hat{y}_i \quad \Longrightarrow \quad \frac{\partial e_i}{\partial \boldsymbol{\omega}_j} = -\frac{\partial \hat{y}_i}{\partial \boldsymbol{\omega}_j}$$

$$\frac{\partial \hat{y}_i}{\partial \boldsymbol{\omega}_j} = \frac{\partial \hat{y}_i}{\partial u_i} \times \frac{\partial u_i}{\partial \boldsymbol{\omega}_j}$$

Perceptron: Learning by Gradient Descent

$$\hat{y}_i = \frac{1}{1 + e^{-u_i}} \quad \Rightarrow \quad \frac{\partial \hat{y}_i}{\partial u_i} = \hat{y}_i(1 - \hat{y}_i)$$

$$u_i = \sum_{r=1}^{d} \boldsymbol{\omega}_r \boldsymbol{x}_{ir} + b \quad \Rightarrow \quad \frac{\partial u_i}{\partial \boldsymbol{\omega}_j} = \boldsymbol{x}_{ij}$$

$$\frac{\partial E}{\partial \boldsymbol{\omega}_j} = \frac{1}{n} \sum_{i=1}^{n} 2e_i \frac{\partial e_i}{\partial \hat{y}_i} \times \frac{\partial \hat{y}_i}{\partial u_i} \times \frac{\partial u_i}{\partial \boldsymbol{\omega}_j}$$

Perceptron: Learning by Gradient Descent

$$\frac{\partial E}{\partial \boldsymbol{\omega}_j} = -\frac{2}{n}\sum_{i=1}^{n} e_i \hat{y}_i (1 - \hat{y}_i) \boldsymbol{x}_{ij}$$

$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} e_i \hat{y}_i (1 - \hat{y}_i)$$

Perceptron: Learning by Pseudo-Inverse

$$\boldsymbol{S_b} = \{(\boldsymbol{x}_i, y_i); i = 1, \ldots n\}$$

$$y_i = 0 \Rightarrow \tilde{y}_i^{(0)} = 0 + \epsilon$$

$$y_i = 1 \Rightarrow \tilde{y}_i^{(1)} = 1 - \epsilon$$

Small Number

Example: $\epsilon = 0.001$

$$\tilde{y}_i^{(0)} = 0.001$$

$$\tilde{y}_i^{(1)} = 0.999$$

Perceptron: Learning by Pseudo-Inverse

$$\tilde{y}_i = \frac{1}{1+e^{-u_i}} \Rightarrow e^{-u_i} = \frac{1}{\tilde{y}_i} - 1 \Rightarrow e^{u_i} = \frac{\tilde{y}_i}{1-\tilde{y}_i}$$

$$u_i = \log_e\left(\frac{\tilde{y}_i}{1-\tilde{y}_i}\right)$$

$$u_i = \boldsymbol{\omega}^T \boldsymbol{x}_i + b$$

$$\boldsymbol{\omega}^T \boldsymbol{x}_i + b = \log_e\left(\frac{\tilde{y}_i}{1-\tilde{y}_i}\right)$$

Perceptron: Learning by Pseudo-Inverse

$$A = \begin{bmatrix} \boldsymbol{x}_1^T & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_i^T & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_n^T & 1 \end{bmatrix}_{n\times(d+1)}$$

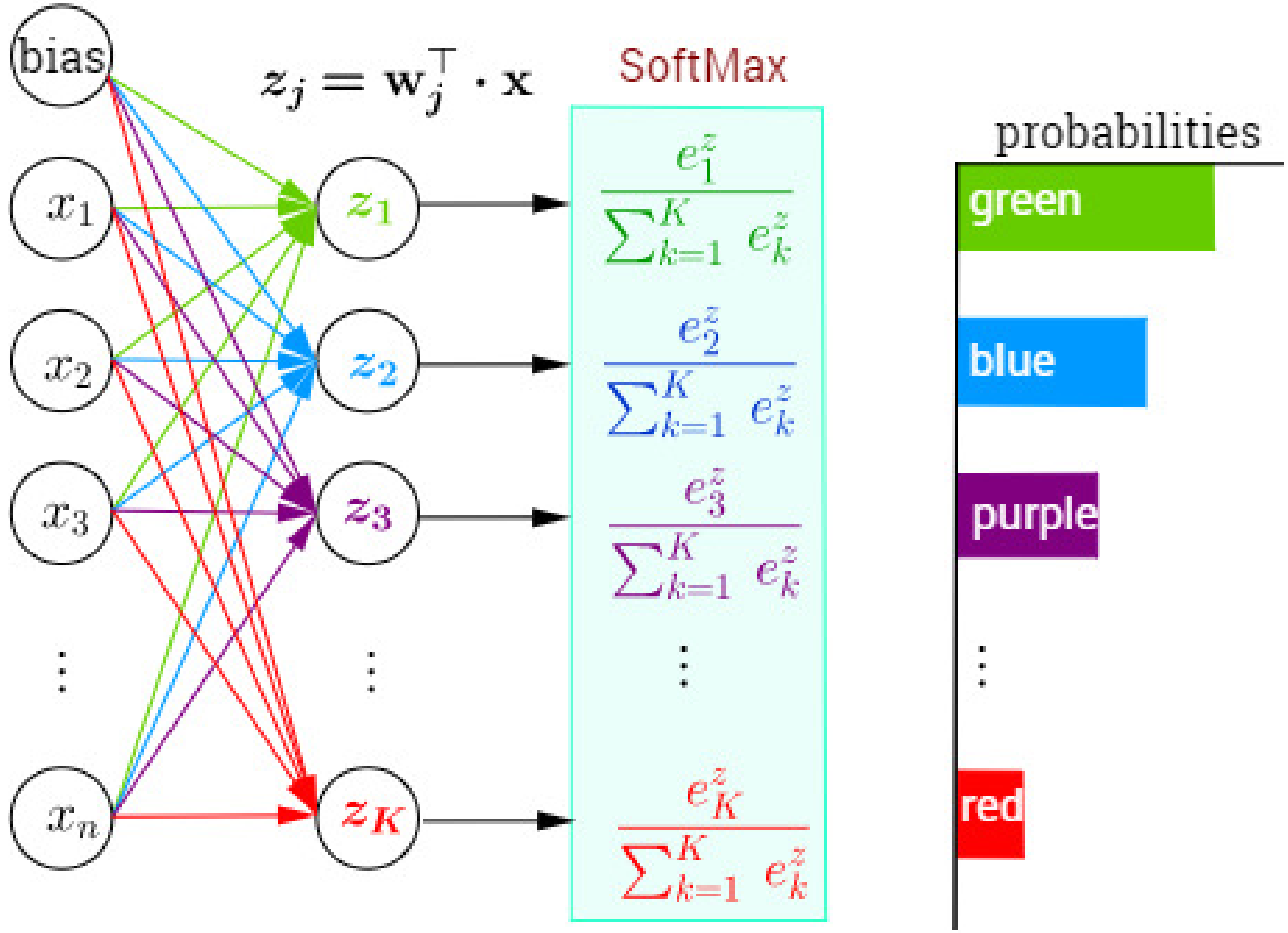
$$P = \begin{bmatrix} \boldsymbol{\omega} \\ b \end{bmatrix}_{(d+1)\times 1}$$

$$B = \begin{bmatrix} \log_e\left(\frac{\tilde{y}_1}{1-\tilde{y}_1}\right) \\ \vdots \\ \log_e\left(\frac{\tilde{y}_i}{1-\tilde{y}_i}\right) \\ \vdots \\ \log_e\left(\frac{\tilde{y}_n}{1-\tilde{y}_n}\right) \end{bmatrix}_{n\times 1}$$

$$P = (A^T A)^{-1} A^T B = A^{\#} B$$

Perceptron: Multi-Category Classification

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Perceptron: Multi-Category Classification

$$\boldsymbol{S_m} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$$

$$\boldsymbol{x} \in \mathbb{R}^d \qquad y \in \{1, \dots m\}$$

$$u_{ij} = \boldsymbol{\omega}_j^T \boldsymbol{x}_i + b_j \quad \Longrightarrow \quad \boldsymbol{u}_i = W\boldsymbol{x}_i + \boldsymbol{b}$$

$$\hat{y}_{ij} = \frac{e^{u_{ij}}}{\sum_{r=1}^{m} e^{u_{ir}}}$$

Soft-Max Activation Function

$$W_{m\times d} = \begin{bmatrix} \boldsymbol{\omega}_1^T \\ \vdots \\ \boldsymbol{\omega}_m^T \end{bmatrix} \qquad \boldsymbol{b}_{m\times 1} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Soft-Max Activation Function

$$\hat{y}_{ij} = \frac{e^{u_{ij}}}{\sum_{r=1}^{m} e^{u_{ir}}}$$

Soft-Max

$$\hat{y}_{ij} = \frac{1}{1 + \sum_{\substack{r=1 \\ r \neq j}}^{m} e^{(u_{ir} - u_{ij})}}$$

$$y = \frac{e^{u}}{1 + e^{u}}$$

Sigmoid

$$y = \frac{1}{1 + e^{-u}}$$

Multi-Class Perceptron: Learning by Gradient Descent

$$\boldsymbol{S_m} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i); i = 1, \dots n\}$$

$$\boldsymbol{x} \in \mathbb{R}^d \qquad \boldsymbol{y} \in \{0,1\}^m$$

$$u_{iq} = \sum_{r=1}^{d} \boldsymbol{\omega}_{qr} \boldsymbol{x}_{ir} + b_q \Longleftarrow q = 1, \dots m \Longrightarrow \widehat{\boldsymbol{y}}_{iq} = \phi_q(u_{iq})$$

$$\boldsymbol{e}_i = \boldsymbol{y}_i - \widehat{\boldsymbol{y}}_i \Longrightarrow \boldsymbol{e}_{iq} = \boldsymbol{y}_{iq} - \widehat{\boldsymbol{y}}_{iq}$$

Multi-Class Perceptron: Learning by Gradient Descent

$$e_{iq} = \boldsymbol{y}_{iq} - \widehat{\boldsymbol{y}}_{iq}$$

$$E = \frac{1}{m \times n} \sum_{i=1}^{n} \sum_{q=1}^{m} \boldsymbol{e}_{iq}^{2}$$

$$\omega_{jk}^{(t+1)} = \omega_{jk}^{(t)} - \eta_{jk}^{(t)} \frac{\partial E}{\partial \boldsymbol{\omega}_{jk}}$$

$$b_{j}^{(t+1)} = b_{j}^{(t)} - \eta_{j}^{(t)} \frac{\partial E}{\partial b_{j}}$$

Multi-Class Perceptron: Learning by Gradient Descent

$$E = \frac{1}{mn}\sum_{i=1}^{n}\sum_{q=1}^{m}\boldsymbol{e}_{iq}^{2} \quad \Rightarrow \quad \frac{\partial E}{\partial \boldsymbol{\omega}_{jk}} = \frac{1}{mn}\sum_{i=1}^{n} 2\boldsymbol{e}_{ij}\frac{\partial \boldsymbol{e}_{ij}}{\partial \boldsymbol{\omega}_{jk}}$$

$$\boldsymbol{e}_{ij} = \boldsymbol{y}_{ij} - \widehat{\boldsymbol{y}}_{ij} \quad \Rightarrow \quad \frac{\partial \boldsymbol{e}_{ij}}{\partial \boldsymbol{\omega}_{jk}} = -\frac{\partial \widehat{\boldsymbol{y}}_{ij}}{\partial \boldsymbol{\omega}_{jk}}$$

$$\frac{\partial \widehat{\boldsymbol{y}}_{ij}}{\partial \boldsymbol{\omega}_{jk}} = \frac{\partial \widehat{\boldsymbol{y}}_{ij}}{\partial u_{ij}} \times \frac{\partial u_{ij}}{\partial \boldsymbol{\omega}_{jk}}$$

Multi-Class Perceptron: Learning by Gradient Descent

$$\hat{\boldsymbol{y}}_{ij} = \phi_j(u_{ij}) \quad \Rightarrow \quad \frac{\partial \hat{\boldsymbol{y}}_{ij}}{\partial u_{ij}} = \phi_j^{(1)}(u_{ij})$$

$$u_{ij} = \sum_{r=1}^{d} \boldsymbol{\omega}_{jr} \boldsymbol{x}_{ir} + b \quad \Rightarrow \quad \frac{\partial u_{ij}}{\partial \boldsymbol{\omega}_{jk}} = \boldsymbol{x}_{ik}$$

$$\frac{\partial E}{\partial \boldsymbol{\omega}_{jk}} = \frac{1}{mn} \sum_{i=1}^{n} 2\boldsymbol{e}_{ij} \frac{\partial \boldsymbol{e}_{ij}}{\partial \hat{\boldsymbol{y}}_{ij}} \times \frac{\partial \hat{\boldsymbol{y}}_{ij}}{\partial u_{ij}} \times \frac{\partial u_{ij}}{\partial \boldsymbol{\omega}_{jk}}$$

Multi-Class Perceptron: Learning by Gradient Descent

$$\frac{\partial E}{\partial \boldsymbol{\omega}_{jk}} = -\frac{2}{mn}\sum_{i=1}^{n} \boldsymbol{e}_{ij}\phi_j^{(1)}(u_{ij})\boldsymbol{x}_{ik}$$

$$\frac{\partial E}{\partial b_j} = -\frac{2}{mn}\sum_{i=1}^{n} \boldsymbol{e}_{ij}\phi_j^{(1)}(u_{ij})$$

Multi-Class Perceptron: Learning by Pseudo-Inverse

$\boldsymbol{S_m} = \{(\boldsymbol{x}_i, y_i); i = 1, \dots n\}$ $\boldsymbol{x} \in \mathbb{R}^d$ $y \in \{1, \dots m\}$

$$\tilde{\boldsymbol{y}}_i[j] = \begin{cases} 1 - \epsilon, & y_i = j \\ \dfrac{\epsilon}{m-1}, & y_i \neq j \end{cases}$$

ϵ : Small Number

Example: $m = 3; y_i = 2; \epsilon = 0.001$ $\Longrightarrow$ $\tilde{\boldsymbol{y}}_i = \begin{bmatrix} 0.0005 \\ 0.999 \\ 0.0005 \end{bmatrix}$

Multi-Class Perceptron: Learning by Pseudo-Inverse

$$\tilde{y}_{ij} = \frac{1}{1 + e^{-u_{ij}}} \Rightarrow e^{-u_{ij}} = \frac{1}{\tilde{y}_{ij}} - 1 \Rightarrow e^{u_{ij}} = \frac{\tilde{y}_{ij}}{1 - \tilde{y}_{ij}}$$

$$u_{ij} = \log_e\left(\frac{\tilde{y}_{ij}}{1 - \tilde{y}_{ij}}\right)$$

$$W\boldsymbol{x}_i + \boldsymbol{b} = \boldsymbol{u}_i$$

$$u_{ij} = \boldsymbol{\omega}_j^T \boldsymbol{x}_i + b_j$$

Multi-Class Perceptron: Learning by Pseudo-Inverse

$P = [W \quad \boldsymbol{b}]_{m\times(d+1)}$ $\qquad$ $B = [\boldsymbol{u}_1 \quad \cdots \quad \boldsymbol{u}_i \quad \cdots \quad \boldsymbol{u}_n]_{m\times n}$

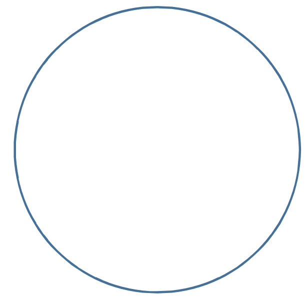
$$A = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_i & \cdots & \boldsymbol{x}_n \\ 1 & \cdots & 1 & \cdots & 1 \end{bmatrix}_{(d+1)\times n}$$

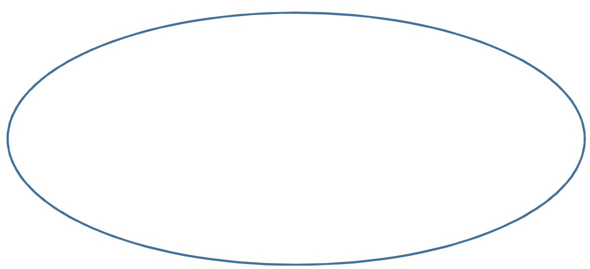
$$P = BA^T(AA^T)^{-1}$$

Moore-Penrose Generalized Pseudo-Inverse

Perceptron: Input Normalization

$$(x - c_x)^2 + \left(y - c_y\right)^2 = r^2$$





$$\left(\frac{x - c_x}{s_x}\right)^2 + \left(\frac{y - c_y}{s_y}\right)^2 = 1$$

Perceptron: Input Normalization

$$\tilde{\boldsymbol{x}}_i = Q^T(\boldsymbol{x}_i - \boldsymbol{\mu})$$

Matrix of Eigen Vectors of Covariance Matrix

Mean Vector

Perceptron: Input Normalization

Mean Value Along
Dimension ***r***

$$\tilde{\boldsymbol{x}}_i[r] = \frac{\boldsymbol{x}_i[r] - \boldsymbol{\mu}[r]}{\boldsymbol{\sigma}[r]}$$

Standard Deviation Along
Dimension ***r***

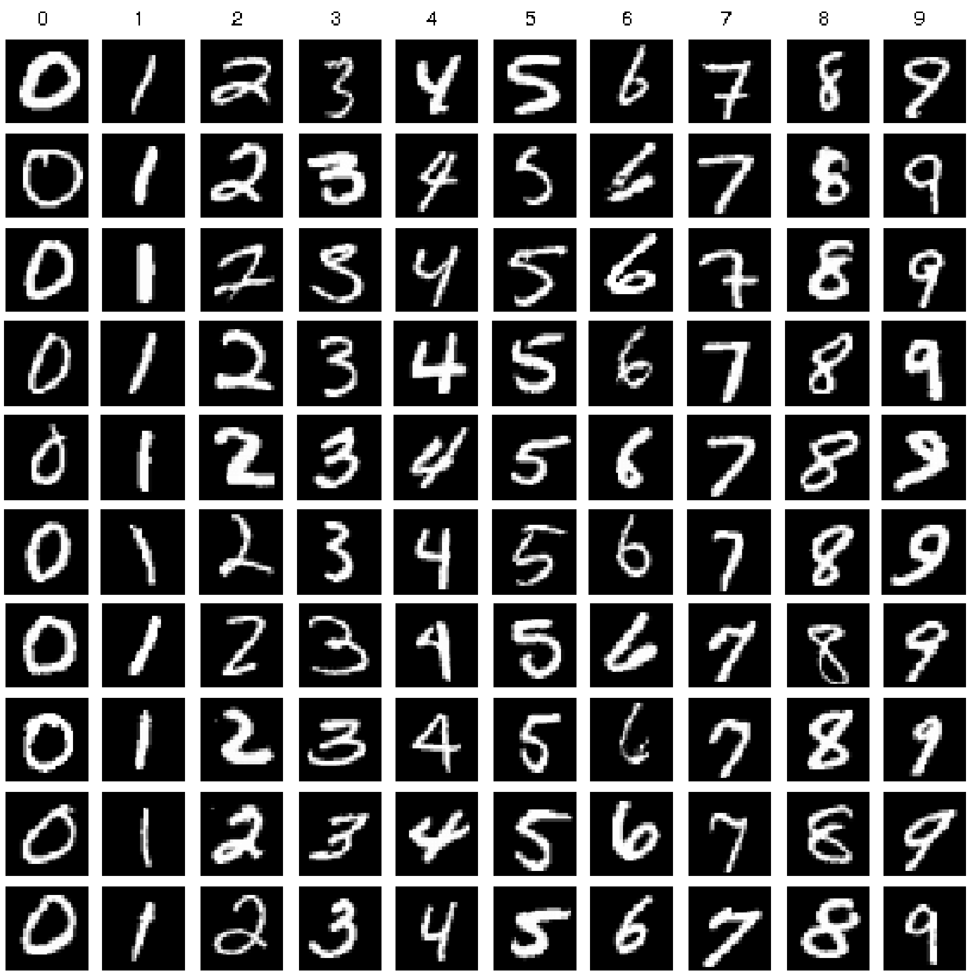
Perceptron: Input Normalization

$$\widetilde{\boldsymbol{x}}_i[r] = \frac{\boldsymbol{x}_i[r] - \boldsymbol{x}_{min}[r]}{\boldsymbol{x}_{max}[r] - \boldsymbol{x}_{min}[r]}$$

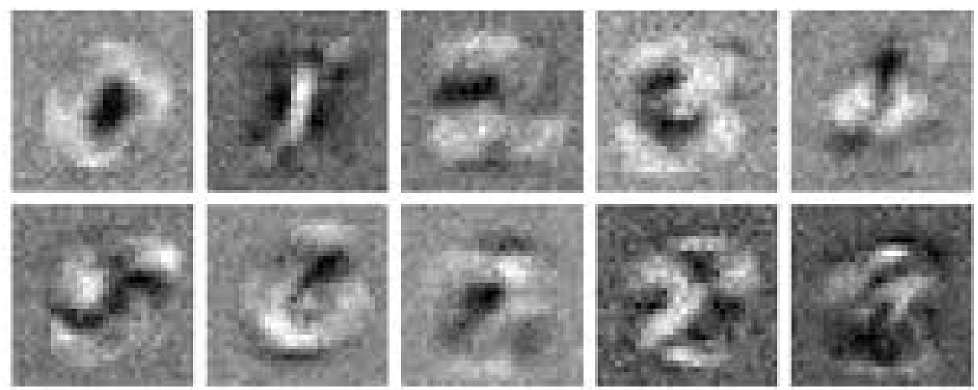
Minimum Value Along Dimension ***r***

Maximum Value Along Dimension ***r***

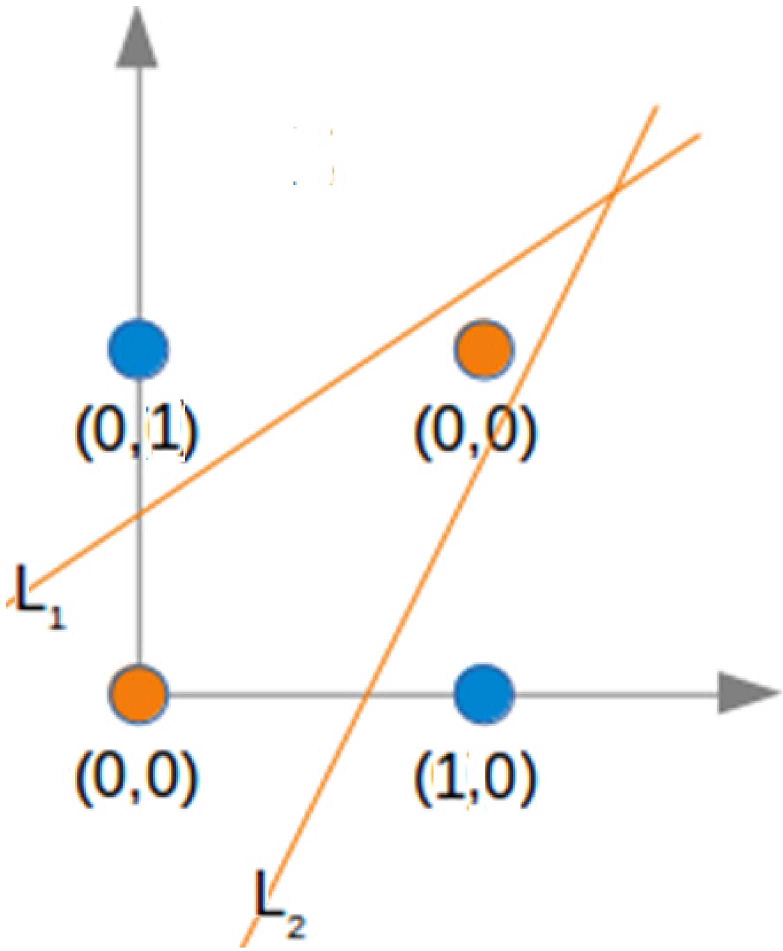
Application to Handwritten Digit Classification



The MNIST Dataset



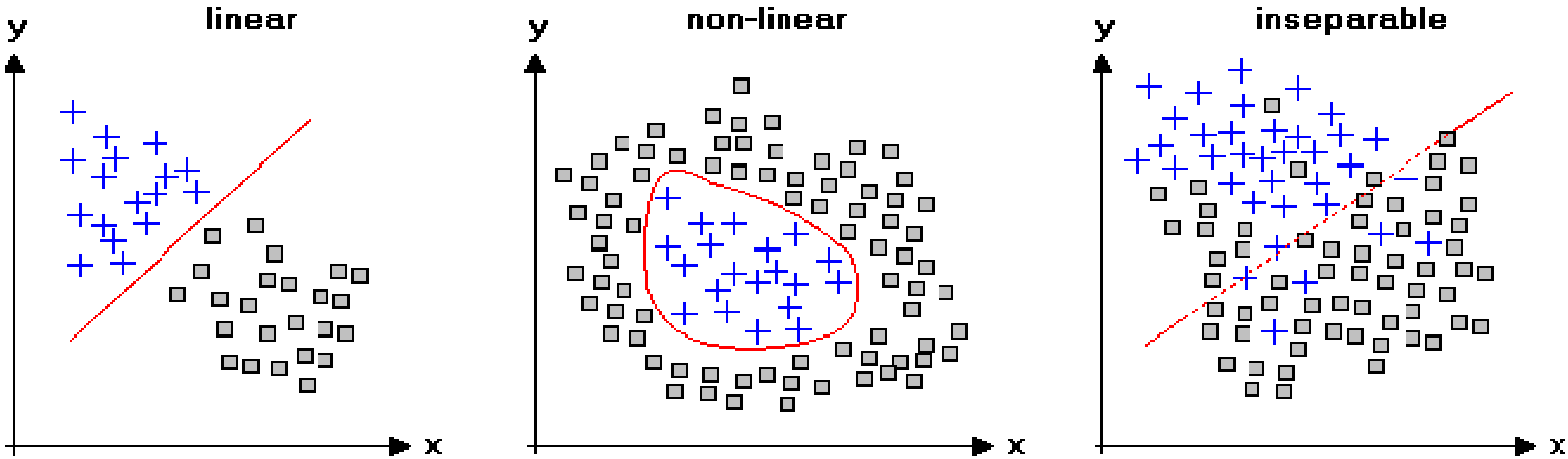
Perceptron: Failure Cases



The XOR Problem

$$y = x_1(1 - x_2) + x_2(1 - x_1)$$

Perceptron: Failure Cases



Covers Theorem of Separability

Nonlinear Transformation to a Higher Dimensional Space Increases the Probability of Linear Separability

Summary

- Separability of Data in Feature Space
- Input-Output Description for Classification Problems
- Perceptron: Motivation & Formulation
- Perceptron: Interpretations
- Perceptron: Learning from Data
- Perceptron: Multi-category Classification
- Perceptron: Applications
- Failure Cases: Towards Multi-Layered Perceptron

Thank You