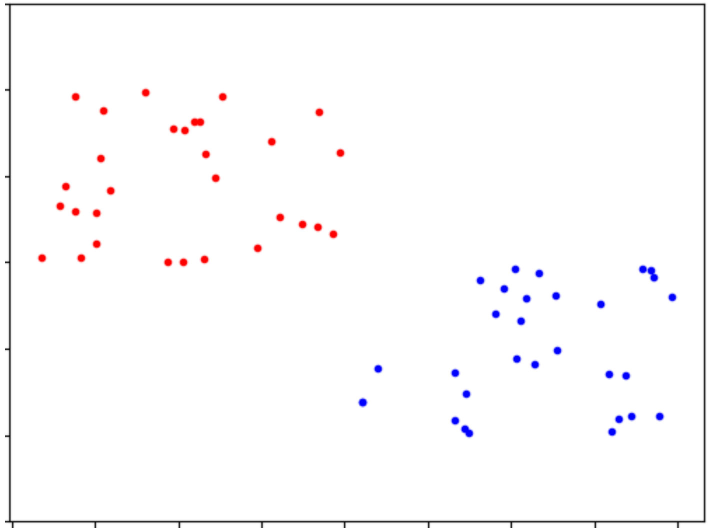
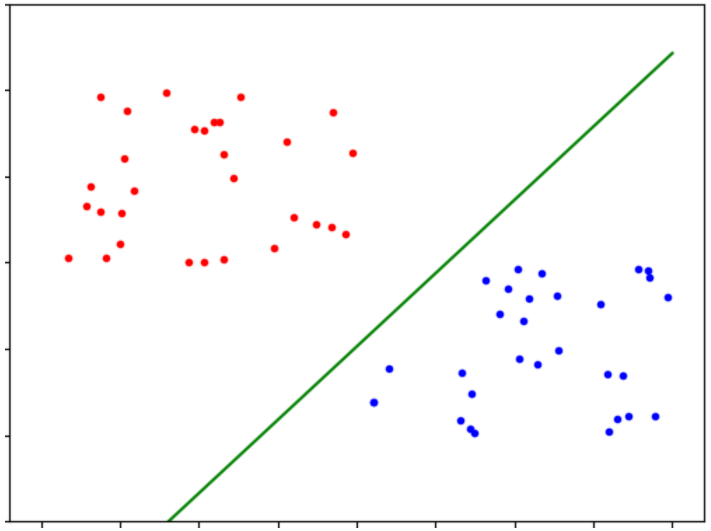
Chapter 1
Feed-Forward Neural Nets

1.1 Perceptrons

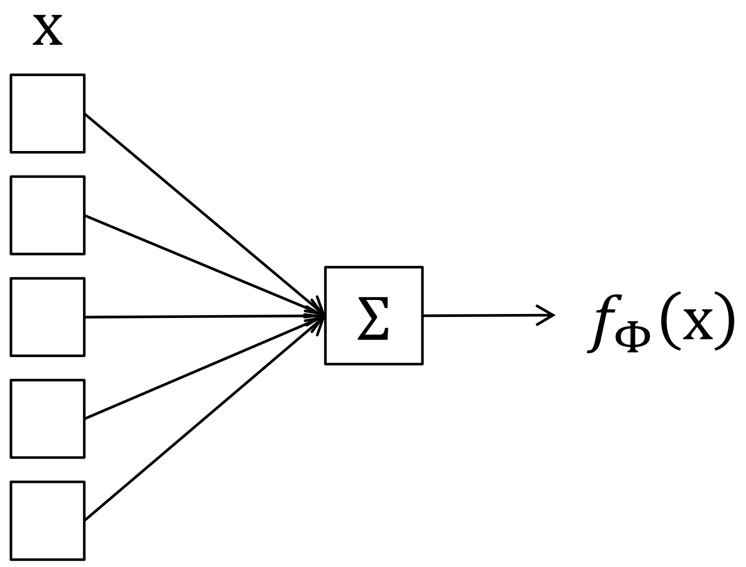
- Find a line that separates red and blue

x_1	x_2	a
0.36	0.24	1
0.82	0.09	1
-0.48	1.00	0
⋮	⋮	⋮



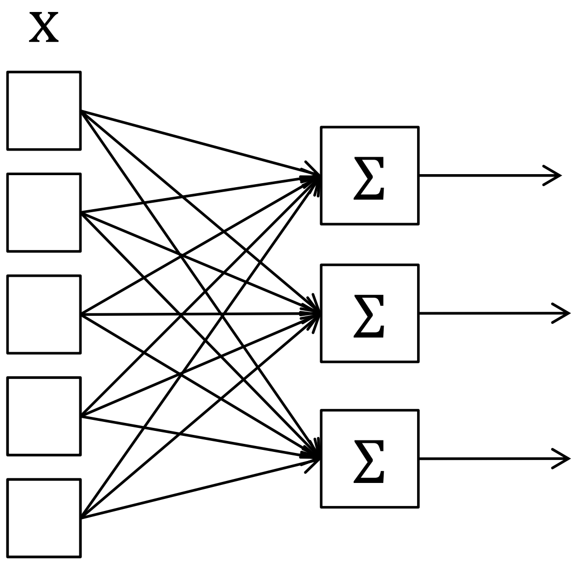


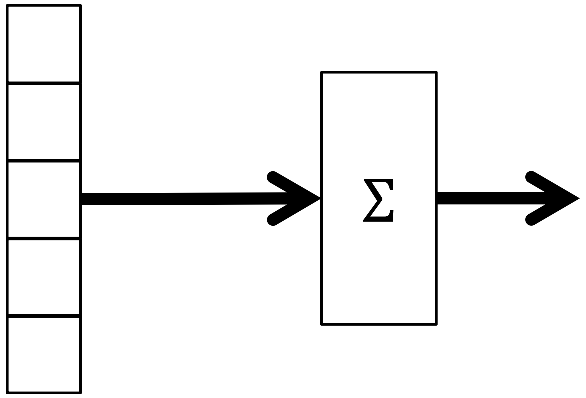
- $\Phi = \{\mathrm{w} \cup b\}$
  where $\mathrm{w} = [w_1, \ldots, w_m]$, $b \in \mathbb{R}$
- $f_\Phi(\mathbf{x}) = \begin{cases} 1 & \text{if } b + w \cdot \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$



- Algorithm
  1. set $b = 0$, $w_i = 0$
  2. until the weights do not change
     (a) for each training example $\mathbf{x}^k$ with answer a^k
        i. if $a^k - f\left(\mathbf{x}^k\right) = 0$, then continue
        ii. else $w_i = w_i + \left(a^k - f\left(\mathbf{x}^k\right)\right) x_i$

Multiclass Decision Problem





1.2 Cross-entropy Loss Functions for Nerual Nets

Softmax

- Definition:

$$\sigma(\mathbf{x})_j = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

$$\sigma(x_1, \dots, x_m) = \left(\frac{e^{x_1}}{\sum_i e^{x_i}}, \dots, \frac{e^{x_m}}{\sum_i e^{x_i}}\right)$$

Cross-entropy

- Definition:

$$H(p, q) = \sum_i p_i \log q_i$$

Cross-entropy loss

- Definition:

$$X(\Phi, \mathbf{x}) = -\ln p_{\Phi}(a_{\mathbf{x}})$$

where $p_{\Phi}(a_x)$ is the probability assigned to $\mathbf{x}$'s label

1.3 Derivatives and Stochastic Gradient Descent

- Equations:

$$X(\Phi, \mathbf{x}) = -\ln p(a)$$

$$p(a) = \sigma_a(\mathbf{l}) = \frac{e^{l_a}}{\sum_i e^{l_i}}$$

$$l_j = b_j + \mathbf{x} \cdot \mathbf{w_j}$$

where $\mathbf{l} = (l_1, \dots, l_m)$

- Gradients:

$$\frac{\partial X(\Phi)}{\partial b_j} = \frac{\partial l_j}{\partial b_j}\frac{\partial X(\Phi)}{\partial l_j} = \begin{cases} -\left(1 - p_j\right) & \text{if } a = j \\ p_j & \text{otherwise} \end{cases}$$

$$\frac{\partial X(\Phi)}{\partial w_{i,j}} = \frac{\partial l_j}{\partial w_{i,j}}\frac{\partial X(\Phi)}{\partial l_j} = \begin{cases} -\left(1 - p_j\right)x_i & \text{if } a = j \\ p_j x_i & \text{otherwise} \end{cases}$$

- Update: For learning rate $\mathcal{L}$

$$b_j = b_j - \mathcal{L}\frac{\partial X(\Phi)}{\partial b_j}$$

$$w_{i,j} = w_{i,j} - \mathcal{L}\frac{\partial X(\Phi)}{\partial w_{i,j}}$$

1.4 Writing Our Program

- Data Normalization
  - $-1 \leq x_i \leq 1$
- Learning Rate
  - 0.0001 in MNIST
- Weights and Bias
  - [-0.1, 0.1]

1.5 Matrix Representation of Neural Nets

- Forward Propagation:

$$\mathbf{L} = \mathbf{XW} + \mathbf{B}$$

where $\boldsymbol{X}$ is the input matrix

- Loss:

$$\Pr\big(A(\mathbf{x})\big) = \sigma(\mathbf{xW} + \mathbf{b})$$
$$L(\mathbf{x}) = -\log(\Pr(A(\mathbf{x}) = a))$$

- Update:

$$\Delta\mathbf{W} = -\mathcal{L}\,\mathbf{X}^T\,\nabla_{\mathbf{l}}X(\Phi)$$

1.6 Data Independence

- iid assumption