IAI, TCG CREST

Avisek Gupta (avisek003@gmail.com)

Machine Learning

14 – Multi-Layered Perceptrons

November 01, 2022

Recall: Multi-class Classification by Softmax Regression

Given a dataset of n instances $X = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(n)}]^T$, $\mathbf{x}^{(i)} \in \mathbb{R}^d$, $X \in \mathbb{R}^{n \times d}$, and ground-truth class labels corresponding to each instance $\mathbf{y} = [y^{(1)}, ..., y^{(n)}]$, $y^{(i)} \in \{0, 1, ..., c-1\}$, $\mathbf{y} \in \{0, 1, ...c-1\}^n$, we wish to estimate a function $\hat{f} : \mathbb{R}^d \rightarrow \{0, 1, ..., c-1\}$ that accurately classifies the data to one of c possible classes.

Softmax Regression models a simultaneous system of c linear discriminating functions to estimate the posterior probabilities of all classes:

$$P(\hat{y}^{(i)} = j|\mathbf{x}^{(i)}) = \frac{\exp(\mathbf{w}_j^T \mathbf{x}^{(i)} + b_j)}{\sum_{j'=1}^{c} \exp(\mathbf{w}_{j'}^T \mathbf{x}^{(i)} + b_{j'})}.$$

Recall: Multi-class Classification by Softmax Regression

Given a dataset of n instances $X = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(n)}]^T$, $\mathbf{x}^{(i)} \in \mathbb{R}^d$, $X \in \mathbb{R}^{n \times d}$, and ground-truth class labels corresponding to each instance $\mathbf{y} = [y^{(1)}, ..., y^{(n)}]$, $y^{(i)} \in \{0, 1, ..., c-1\}$, $\mathbf{y} \in \{0, 1, ...c-1\}^n$, we wish to estimate a function $\hat{f} : \mathbb{R}^d \to \{0, 1, ..., c-1\}$ that accurately classifies the data to one of c possible classes.

Softmax Regression models a simultaneous system of c linear discriminating functions to estimate the posterior probabilities of all classes:

$$P(\hat{y}^{(i)} = j|\mathbf{x}^{(i)}) = \frac{\exp(\mathbf{w}_j^T \mathbf{x}^{(i)} + b_j)}{\sum_{j'=1}^{c} \exp(\mathbf{w}_{j'}^T \mathbf{x}^{(i)} + b_{j'})}.$$

The Softmax Regression model can also be expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

where we wish to estimate $W \in \mathbb{R}^{d \times c}$ and $\mathbf{b} \in \mathbb{R}^c$, to obtain $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^c$.

Recall: Multi-class Classification by Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T\mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

where $\sum_{j=1}^{c} \hat{y}_j^{(i)} = 1$.

Recall: Multi-class Classification by Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

where $\sum_{j=1}^{c} \hat{y}_j^{(i)} = 1$.

Let $\mathbf{t}^{(i)} \in \{0,1\}^c$, $\sum_{j=1}^{c} t_j^{(i)} = 1$ be the **one-hot representation** of the corresponding ground-truth label $y^{(i)} \in \{0,1,...,c-1\}^c$.

Example 1: For $c = 5$, if $y^{(i)} = 2$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}$.

Example 2: For $c = 10$, if $y^{(i)} = 7$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$.

Example 3: For $c = 3$, if $y^{(i)} = 0$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$.

Recall: Multi-class Classification by Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \mathrm{Softmax}(\mathbf{a}^{(i)}),$$

Example: For $c = 3$, if $y^{(i)} = 0$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$.

If $y^{(i)} = 1$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$.

If $y^{(i)} = 2$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$.

Recall: Multi-class Classification by Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

Example: For $c = 3$, if $y^{(i)} = 0$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$.

If $y^{(i)} = 1$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$.

If $y^{(i)} = 2$, then $\mathbf{t}^{(i)} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$.

Using a loss function, we wish to estimate the Logistic Regression parameters so that $\hat{\mathbf{y}}^{(i)} \approx \mathbf{t}^{(i)}$, where, $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^c$, $\sum_{j=1}^{c} \hat{y}_j^{(i)} = 1$.

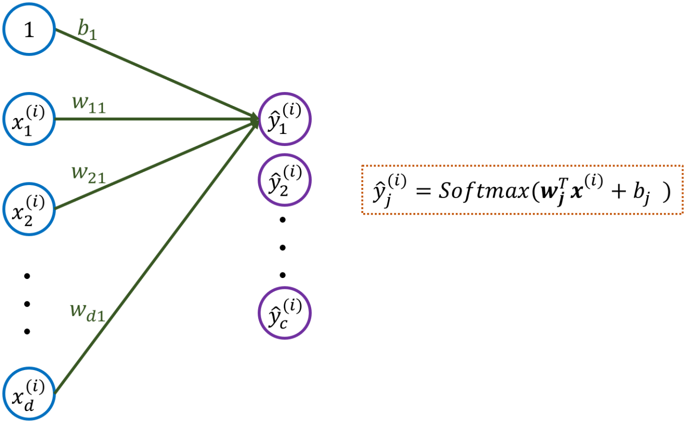
Example: MSE loss $= \sum_{i=1}^{n} ||\mathbf{t}^{(i)} - \hat{\mathbf{y}}^{(i)}||^2$.

Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \mathrm{Softmax}(\mathbf{a}^{(i)}),$$

The Softmax Regression model can be visually expressed as:

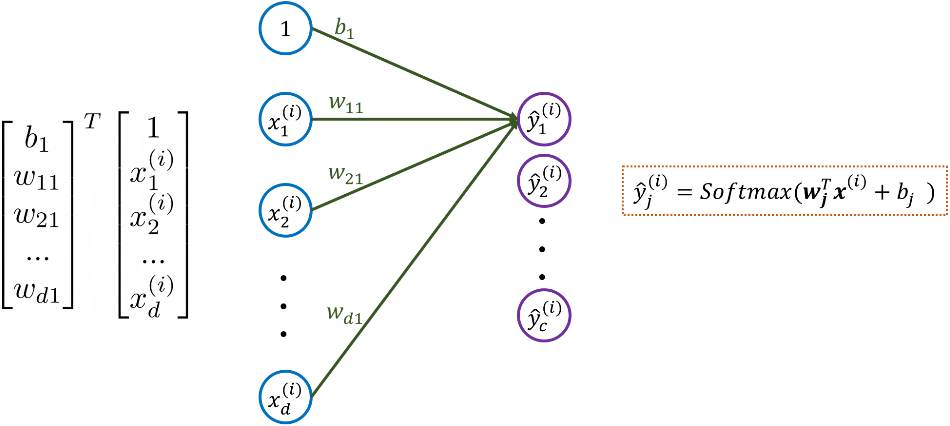


Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

The Softmax Regression model can be visually expressed as:



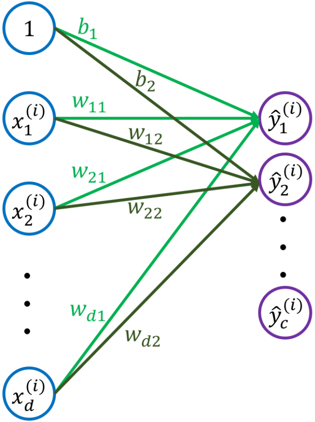
Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T\mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

The Softmax Regression model can be visually expressed as:

$$\begin{bmatrix} b_1 & b_2 \\ w_{11} & w_{12} \\ w_{21} & w_{22} \\ \dots & \dots \\ w_{d1} & w_{d2} \end{bmatrix}^T \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \dots \\ x_d^{(i)} \end{bmatrix}$$



$$\hat{y}_j^{(i)} = Softmax(\boldsymbol{w}_j^T\boldsymbol{x}^{(i)} + b_j)$$

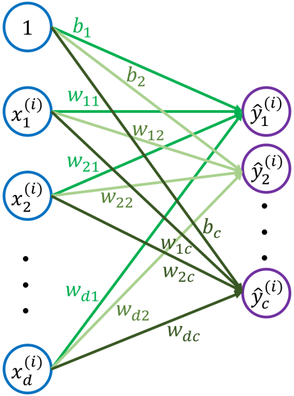
Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

The Softmax Regression model can be visually expressed as:

$$\begin{bmatrix} b_1 & b_2 & \dots & b_c \\ w_{11} & w_{12} & \dots & w_{1c} \\ w_{21} & w_{22} & \dots & w_{2c} \\ \dots & \dots & \dots & \dots \\ w_{d1} & w_{d2} & \dots & w_{dc} \end{bmatrix}^T \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \dots \\ x_d^{(i)} \end{bmatrix}$$



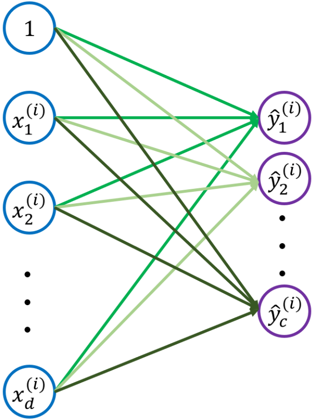
$$\hat{y}_j^{(i)} = Softmax(\boldsymbol{w}_j^T \boldsymbol{x}^{(i)} + b_j)$$

Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

The Softmax Regression model can be visually expressed as:



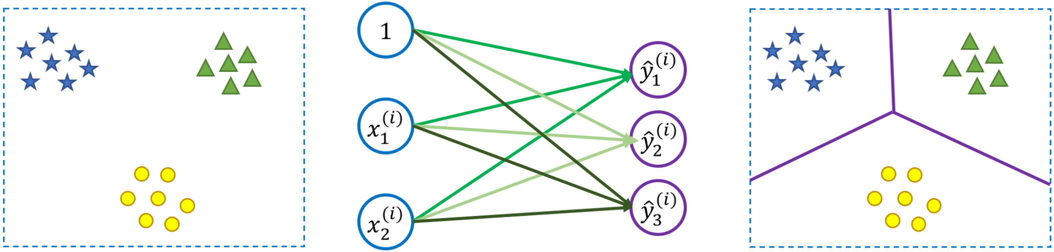
$$\hat{\boldsymbol{y}}^{(i)} = Softmax(W^T \boldsymbol{x}^{(i)} + \boldsymbol{b})$$

Softmax Regression

The Softmax Regression model is expressed as:

$$\mathbf{a}^{(i)} = W^T \mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

We expect Softmax Regression to learn the proper decision boundaries between classes of data.



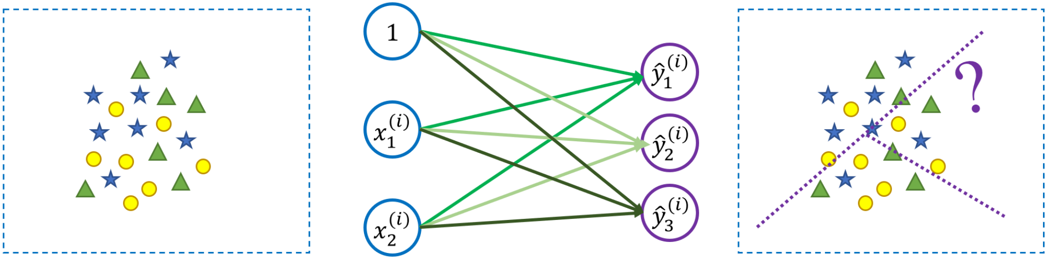
Limitations of this approach...?

Softmax Regression

The Softmax Regression model is expressed as:

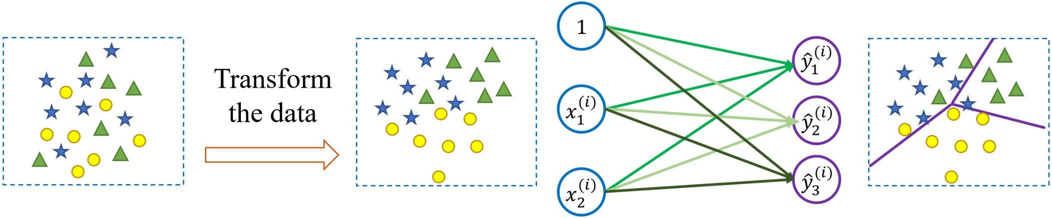
$$\mathbf{a}^{(i)} = W^T\mathbf{x}^{(i)} + \mathbf{b}, \quad \hat{\mathbf{y}}^{(i)} = \text{Softmax}(\mathbf{a}^{(i)}),$$

If the data is not separable by linear decision boundaries, Softmax Regression will not find a high accuracy classifier.



Softmax Regression

Possible Solution - Transform the data to a space where Softmax Regression will have higher chances of success.

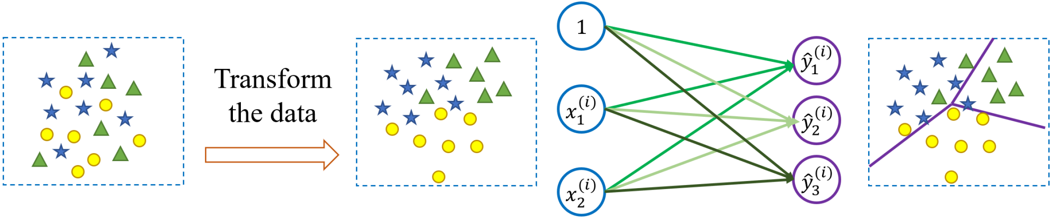


Two types of data transformations:

1. Feature Extraction: Consider a linear or a non-linear transformation of the data (e.g., PCA, Isomap, tSNE,...)
2. Feature Selection: Select a subset of features (e.g., lasso)

Softmax Regression

Possible Solution - Transform the data to a space where Softmax Regression will have higher chances of success.



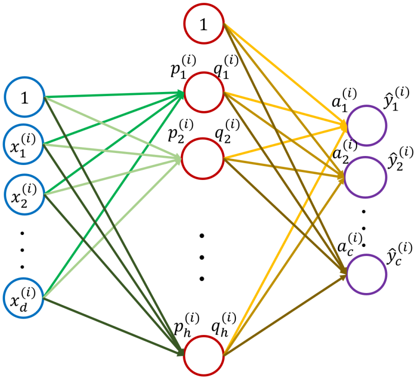
Two types of data transformations:

1. Feature Extraction: Consider a linear or a non-linear transformation of the data (e.g., PCA, Isomap, tSNE,...)
2. Feature Selection: Select a subset of features (e.g., lasso)

Performance of Softmax Regression is limited by the performance of the previous feature extraction / selection approach.

Multi-Layered Perceptron

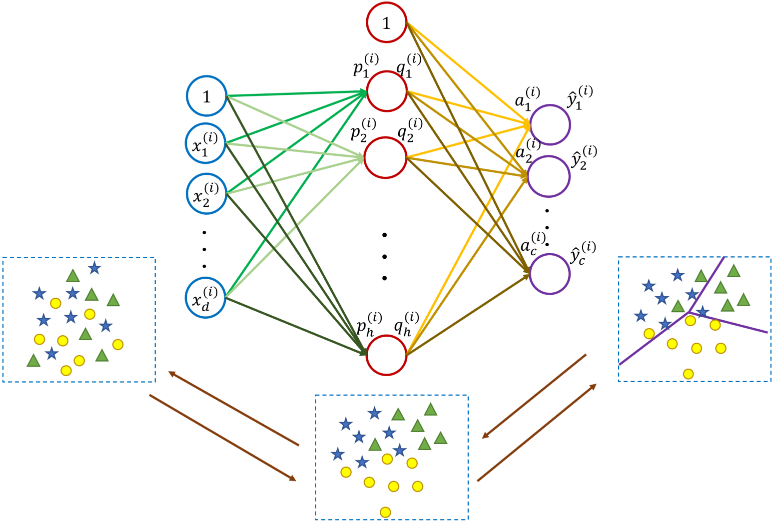
Idea - A single network learns a data transformation (linear / non-linear) followed by a c-class classifier.



A 2-layer multi-layered perceptron first maps the **input layer** to the **hidden layer**, followed by mapping the hidden layer to the **output layer**.

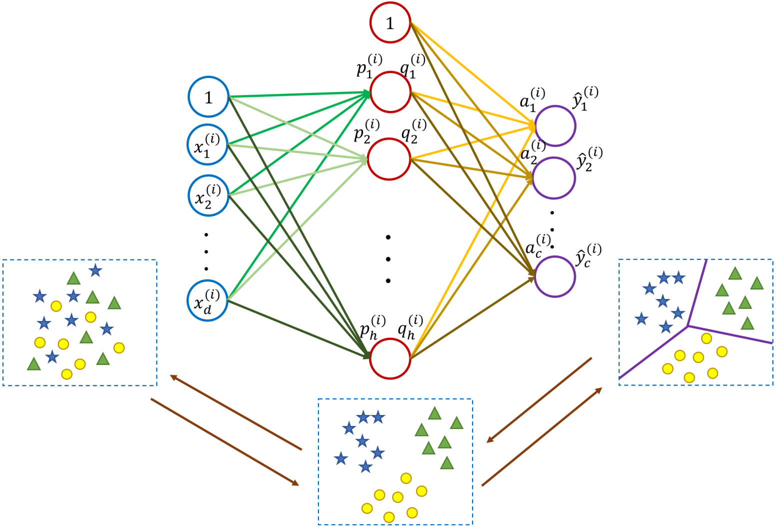
Multi-Layered Perceptron

Idea - A single network learns a data transformation (linear / non-linear) followed by a c-class classifier.

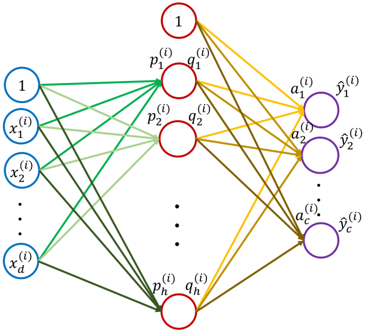


Multi-Layered Perceptron

Idea - A single network learns a data transformation (linear / non-linear) followed by a c-class classifier.



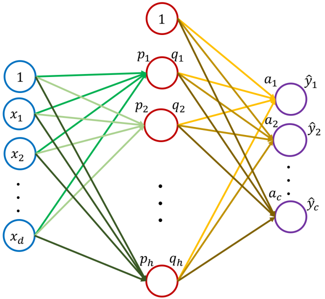
Multi-Layered Perceptron



Feedforward operation in a 2-layer Multi-Layered Perceptron: Given input $\mathbf{x}^{(i)}$, we obtain $\hat{\mathbf{y}}^{(i)}$ by the following sequence of operations -

$$\mathbf{p}^{(i)} = (W^1)^T \mathbf{x}^{(i)} + \mathbf{b}^1,\ \mathbf{q}^{(i)} = \sigma(\mathbf{p}^{(i)})$$
$$\mathbf{a}^{(i)} = (W^2)^T \mathbf{q}^{(i)} + \mathbf{b}^2,\ \hat{\mathbf{y}}^{(i)} = \sigma(\mathbf{a}^{(i)})$$

Multi-Layered Perceptron



Feedforward operation in a 2-layer Multi-Layered Perceptron: Given input $\mathbf{x}$, we obtain $\hat{\mathbf{y}}$ by the following sequence of operations -

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \; q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \; \hat{y}_k = \sigma(a_k)$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \ \hat{y}_k = \sigma(a_k)$$

Measure the loss:

$$J = \frac{1}{2} \sum_{k=1}^{c} (t_k - \hat{y}_k)^2$$

Estimate the model parameters by computing the following gradients:

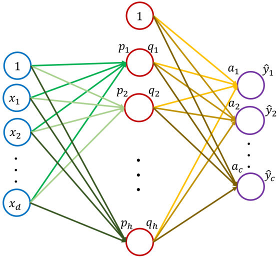
$$\frac{\partial}{\partial W_{lj}^{(1)}} J, \ \frac{\partial}{\partial b_j^{(1)}} J, \ \frac{\partial}{\partial W_{l'k}^{(2)}} J, \ \frac{\partial}{\partial b_k^{(2)}} J$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation: Estimate the model parameters by computing the following gradients:

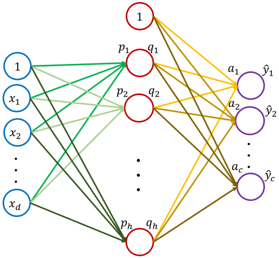
$$\frac{\partial}{\partial W_{lj}^{(1)}} J,\ \frac{\partial}{\partial b_j^{(1)}} J,\ \frac{\partial}{\partial W_{l'k}^{(2)}} J,\ \frac{\partial}{\partial b_k^{(2)}} J$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

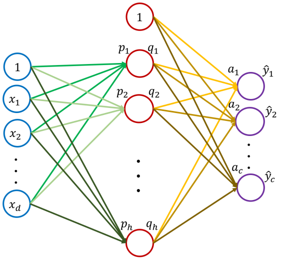
$$\frac{\partial}{\partial W_{l'k}^{(2)}} J =$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial W_{l'k}^{(2)}}$$

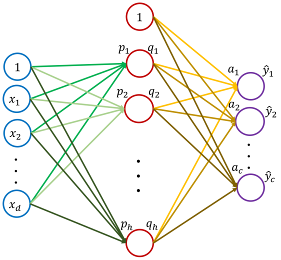
$$\frac{\partial J}{\partial \hat{y}_k} =$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial W_{l'k}^{(2)}}$$

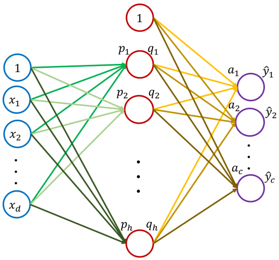
$$\frac{\partial J}{\partial \hat{y}_k} = -(t_k - \hat{y}_k), \quad \frac{\partial \hat{y}_k}{\partial a_k} =$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} . \frac{\partial \hat{y}_k}{\partial a_k} . \frac{\partial a_k}{\partial W_{l'k}^{(2)}}$$

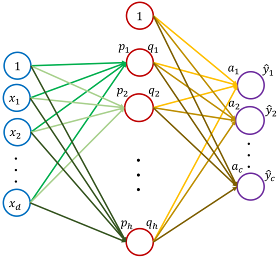
$$\frac{\partial J}{\partial \hat{y}_k} = -(t_k - \hat{y}_k),\quad \frac{\partial \hat{y}_k}{\partial a_k} = \hat{y}_k(1 - \hat{y}_k),\quad \frac{\partial a_k}{\partial W_{l'k}^{(2)}} =$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial W_{l'k}^{(2)}}$$

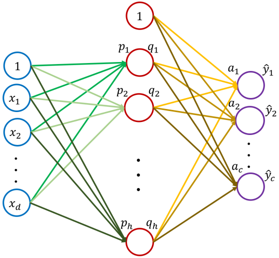
$$\frac{\partial J}{\partial \hat{y}_k} = -(t_k - \hat{y}_k), \quad \frac{\partial \hat{y}_k}{\partial a_k} = \hat{y}_k(1 - \hat{y}_k), \quad \frac{\partial a_k}{\partial W_{l'k}^{(2)}} = q_{l'}$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \; q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \; \hat{y}_k = \sigma(a_k), \; J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial W_{l'k}^{(2)}}$$

$$\frac{\partial J}{\partial \hat{y}_k} = -(t_k - \hat{y}_k), \quad \frac{\partial \hat{y}_k}{\partial a_k} = \hat{y}_k(1 - \hat{y}_k), \quad \frac{\partial a_k}{\partial W_{l'k}^{(2)}} = q_{l'}$$

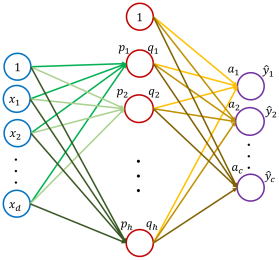
$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1 - \hat{y}_k) q_{l'}$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)},\ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)},\ \hat{y}_k = \sigma(a_k),\ J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1 - \hat{y}_k)q_{l'}, \quad \frac{\partial}{\partial b_k^{(2)}} J = ?$$

$$\frac{\partial}{\partial b_k^{(2)}} J = \frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial b_k^{(2)}}$$

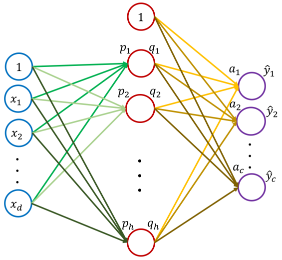
$$\frac{\partial J}{\partial \hat{y}_k} = -(t_k - \hat{y}_k), \quad \frac{\partial \hat{y}_k}{\partial a_k} = \hat{y}_k(1 - \hat{y}_k), \quad \frac{\partial a_k}{\partial b_k^{(2)}} = 1$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \; q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \; \hat{y}_k = \sigma(a_k), \; J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1 - \hat{y}_k)q_{l'}, \quad \frac{\partial}{\partial b_k^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1 - \hat{y}_k)$$

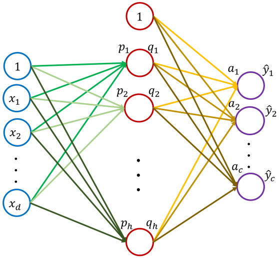
$$\frac{\partial}{\partial W_{lj}^{(1)}} J = ?$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \ q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \ \hat{y}_k = \sigma(a_k), \ J = \frac{1}{2} \sum_{k=1}^{c} (t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k)q_{l'}, \quad \frac{\partial}{\partial b_k^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k)$$

$$\frac{\partial}{\partial W_{lj}^{(1)}} J = \sum_{k=1}^{c} \left[\frac{\partial J}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial q_j} \right] \cdot \frac{\partial q_j}{\partial p_j} \cdot \frac{\partial p_j}{\partial W_{lj}^{(1)}}$$

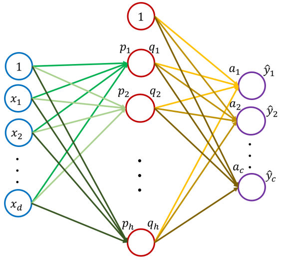
$$\frac{\partial}{\partial W_{lj}^{(1)}} J = \sum_{k=1}^{c} \left[-(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k) W_{jk}^{(2)} \right] q_j(1-q_j) x_l$$

Multi-Layered Perceptron

Feedforward:

$$p_j = \sum_{l=1}^{d} W_{lj}^{(1)} x_l + b_j^{(1)}, \; q_j = \sigma(p_j)$$

$$a_k = \sum_{l'=1}^{h} W_{l'k}^{(2)} q_{l'} + b_k^{(2)}, \; \hat{y}_k = \sigma(a_k), \; J = \frac{1}{2}\sum_{k=1}^{c}(t_k - \hat{y}_k)^2$$



Backpropagation:

$$\frac{\partial}{\partial W_{l'k}^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k) q_{l'}, \quad \frac{\partial}{\partial b_k^{(2)}} J = -(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k),$$

$$\frac{\partial}{\partial W_{lj}^{(1)}} J = \sum_{k=1}^{c}\left[-(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k) W_{jk}^{(2)}\right] q_j(1-q_j) x_l,$$

$$\frac{\partial}{\partial b_j^{(1)}} J = \sum_{k=1}^{c}\left[-(t_k - \hat{y}_k)\hat{y}_k(1-\hat{y}_k) W_{jk}^{(2)}\right] q_j(1-q_j).$$

Gradient Descent on batches of data

1. If the size of the dataset n is small, we can feedforward the entire data at once.

$$P = (W^{(1)})^T X^T + \mathbf{b}^{(1)},\ Q = \sigma(P)$$
$$A = (W^{(2)})^T Q + \mathbf{b}^{(2)},\ \hat{Y} = \sigma(A)$$

Similarly, expressions for backpropagation can be found.

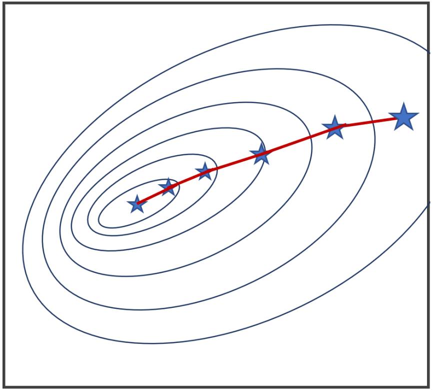
2. Batch Learning: If n is large, we can divide the data into **batches** and feedforward one batch of data at a time. If a batch of data $X_i \in \mathbb{R}^{b\times d}$, then,

$$P_i = (W^{(1)})^T X_i^T + \mathbf{b}^{(1)},\ Q_i = \sigma(P_i)$$
$$A_i = (W^{(2)})^T Q_i + \mathbf{b}^{(2)},\ \hat{Y}_i = \sigma(A_i)$$

3. Stochastic Gradient Descent: If $b = 1$, one data instance is fed to the network at a time.

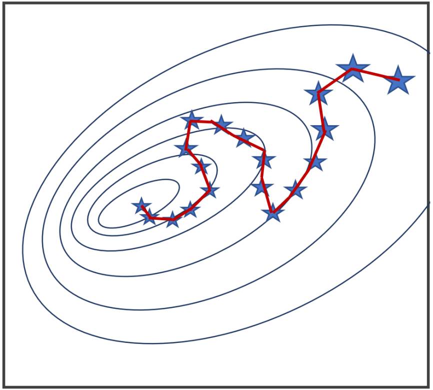
Gradient Descent on batches of data

Gradient Descent on larger batch sizes



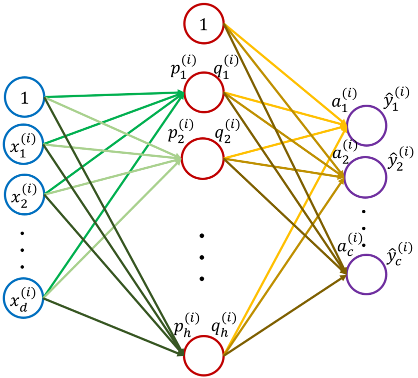
Faster convergence to a local minima,
low exploration of the parameter space

Gradient Descent on smaller batch sizes



Slower convergence to a local minima,
more exploration of the parameter space

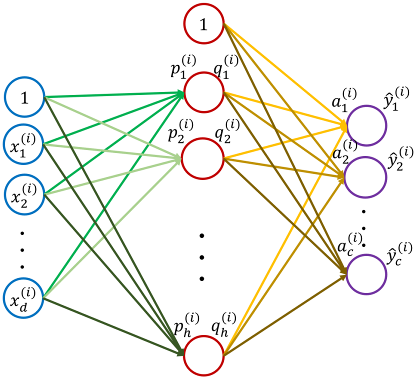
Multi-layered Perceptrons are Universal Function Approximators



Multi-layered Perceptrons with only one hidden layer have been proved to have the capability of approximating any function. [Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". Neural Networks. 4 (2): 251-257]

Caveat - The number of hidden neurons may go to infinity.

Motivation: Deep Neural Networks



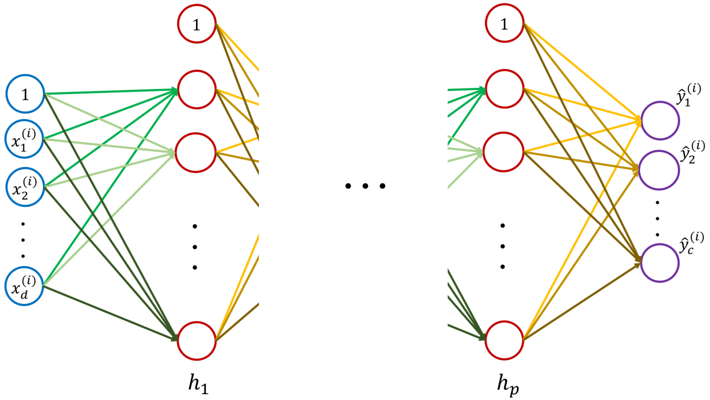
A single hidden layer network can be represented as,

$$\mathbf{q} = f_1(\mathbf{x}), \quad \mathbf{y} = f_2(\mathbf{q})$$

or,

$$\mathbf{y} = f_2(f_1(\mathbf{x}))$$

Motivation: Deep Neural Networks



A network with p number of hidden layers can be represented as,

$$\mathbf{y} = f_{p+1}(...f_2(f_1(\mathbf{x})))$$

Such **deep** overparameterized neural networks excel at learning on problems where large volumes of data are available.