CSE 411: Machine Learning

Multilayer Perceptrons

Dr Muhammad Abul Hasan

Department of Computer Science and Engineering
Green University of Bangladesh
muhammad.hasan@cse.green.edu.bd

Fall 2023

Outline

> *“Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don’t think AI (Artificial Intelligence) will transform in the next several years.*
>
> *– Andrew Ng”*

Next Up ...

Introduction

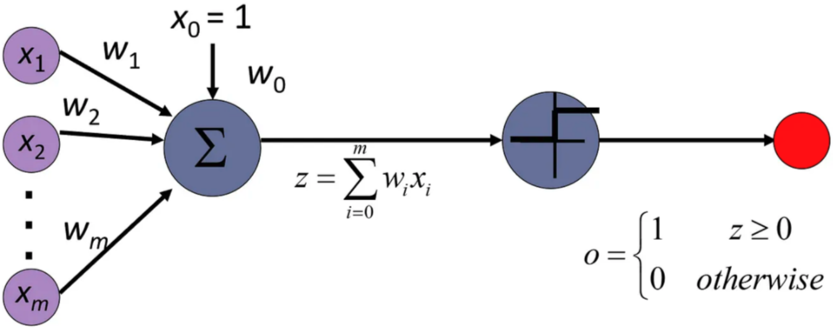
The perceptron is a type of linear classifier used in machine learning. It was introduced in 1957 by Frank Rosenblatt and is based on the idea of a single neuron in the brain.

The Perceptron

- First neural network learning model in the 1960's
- Simple and limited (single layer model)
- Basic concepts are similar for multi-layer models so this is a good learning tool

The Perceptron

- The perceptron has m binary inputs denoted by $x_1, \cdots, x_m$, which represent the incoming signals
- outputs a single binary value denoted if the perceptron is “firing” or not.



The Perceptron

The perceptron computes the weighted sum of its incoming signals.

$$z = w_1 x_1 + \cdots + w_m x_m \tag{1}$$

$$= \sum_{i=1}^{m} w_i x_i \tag{2}$$

- To simplify the learning process, a special input called bias is added (value 1).
- This neuron is typically denoted by x_0 and its connection weight is denoted by w_0.

$$z = \sum_{i=1}^{m} w_i x_i + \text{bias} \tag{3}$$

$$= \sum_{i=1}^{m} w_i x_i + w_0 x_0 = \sum_{i=0}^{m} w_i x_i \tag{4}$$

Next Up ...

Perceptron Algorithm

The perceptron algorithm is a supervised learning algorithm for binary classification. It learns a linear function by updating the weights of the input features based on the errors made by the model.

- Initialize the weights to zero or small random values.
- For each training example, calculate the predicted output.
- Update the weights based on the prediction error.
- Repeat until convergence or a maximum number of iterations is reached.

The perceptron algorithm is guaranteed to converge if the data is linearly separable.

Next Up ...

Example

Suppose we have the following training data:

Feature 1	Feature 2	Label
1	2	1
2	3	1
3	1	-1

We can use the perceptron algorithm to learn a linear classifier for this data. After several iterations, the algorithm converges to the following weights:
$w_0 = -3$, $w_1 = 2$, $w_2 = 1$
The final decision boundary is $-3 + 2x_1 + x_2 = 0$, which separates the positive examples from the negative examples.

Example

Let's play with perceptron learning:
https://dennis198.github.io/Perceptron-Visualizer/

Example

What is the limitation of single-layer perceptron learning?

Answer: Simplicity and Non-linearity (that means, it is not capable of learning complex non-linear function)

Example

What is the limitation of single-layer perceptron learning?
Answer: Simplicity and Non-linearity (that means, it is not capable of learning complex non-linear function)

Next Up ...

Multilayer Perceptrons

A multilayer perceptron represents an adaptable model $y(\cdot, w)$ able to map D-dimensional input to C-dimensional output:

$$y(\cdot, w) : \mathbb{R}^D \to \mathbb{R}^C, x \mapsto y(x, w) = \begin{pmatrix} y_1(x, w) \\ \vdots \\ y_C(x, w) \end{pmatrix}. \tag{5}$$

In general, a $(L + 1)$-layer perceptron consists of $(L + 1)$ layers, each layer l computing linear combinations of the previous layer $(l - 1)$ (or the input).

Multilayer Perceptrons – First Layer

On input $x \in \mathbb{R}^D$, layer $l = 1$ computes a vector $y^{(1)} := (y_1^{(1)}, \dots, y_{m^{(1)}}^{(1)})$ where

$$y_i^{(1)} = f\left(z_i^{(1)}\right) \quad \text{with } z_i^{(1)} = \sum_{j=1}^{D} w_{i,j}^{(1)} x_j + w_{i,0}^{(1)}. \tag{6}$$

i^{th} component is called "unit i"

where f is called activation function and $w_{i,j}^{(1)}$ are adjustable weights.

Multilayer Perceptrons – First Layer

What does this mean?

Layer $l = 1$ computes linear combinations of the input and applies a (non-linear) activation function ...

The first layer can be interpreted as generalized linear model:

$$y_i^{(1)} = f\left(\left(w_i^{(1)}\right)^T x + w_{i,0}^{(1)}\right). \tag{7}$$

Idea: Recursively apply L additional layers on the output $y^{(1)}$ of the first layer.

Multilayer Perceptrons – Further Layers

In general, layer l computes a vector $y^{(l)} := (y_1^{(l)}, \ldots, y_{m^{(l)}}^{(l)})$ as follows:

$$y_i^{(l)} = f\left(z_i^{(l)}\right) \quad \text{with } z_i^{(l)} = \sum_{j=1}^{m^{(l-1)}} w_{i,j}^{(l)} y_j^{(l-1)} + w_{i,0}^{(l)}. \tag{8}$$

Thus, layer l computes linear combinations of layer $(l-1)$ and applies an activation function ...

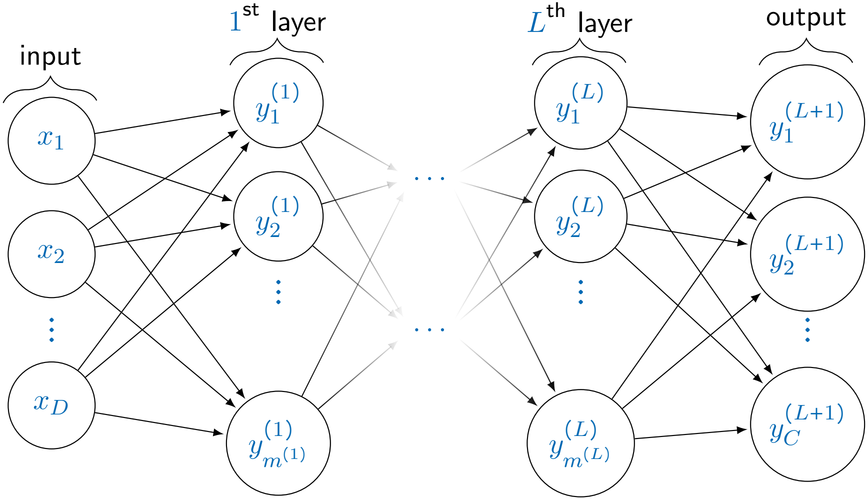
Multilayer Perceptrons – Output Layer

Layer $(L+1)$ is called output layer because it computes the output of the multilayer perceptron:

$$y(x, w) = \begin{pmatrix} y_1(x, w) \\ \vdots \\ y_C(x, w) \end{pmatrix} := \begin{pmatrix} y_1^{(L+1)} \\ \vdots \\ y_C^{(L+1)} \end{pmatrix} = y^{(L+1)} \tag{9}$$

where $C = m^{(L+1)}$ is the number of output dimensions.

Network Graph



Activation Functions – Notions

How to choose the activation function f in each layer?

- Non-linear activation functions will increase the expressive power: Multilayer perceptrons with $L + 1 \geq 2$ are universal approximators **?**!
- Depending on the application: For classification we may want to interpret the output as posterior probabilities:

$$y_i(x, w) \stackrel{!}{=} p(c = i|x) \tag{10}$$

where c denotes the random variable for the class.

Activation Functions – Notions

How to choose the activation function f in each layer?

- Non-linear activation functions will increase the expressive power: Multilayer perceptrons with $L + 1 \geq 2$ are universal approximators **?**!
- Depending on the application: For classification we may want to interpret the output as posterior probabilities:

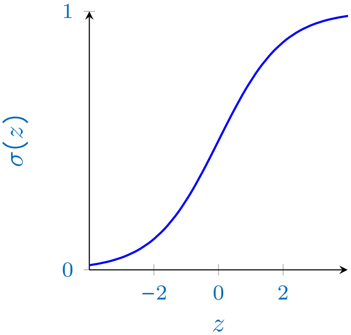
$$y_i(x, w) \overset{!}{=} p(c = i|x) \tag{10}$$

where c denotes the random variable for the class.

Activation Functions

Usually the activation function is chosen to be the logistic sigmoid:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



which is non-linear, monotonic and differentiable.

Activation Functions

Alternatively, the hyperbolic tangent is used frequently:

$$\tanh(z). \tag{11}$$

For classification with $C > 1$ classes, layer $(L+1)$ uses the softmax activation function:

$$y_i^{(L+1)} = \sigma(z^{(L+1)}, i) = \frac{\exp(z_i^{(L+1)})}{\sum_{k=1}^{C} \exp(z_k^{(L+1)})}. \tag{12}$$

Then, the output can be interpreted as posterior probabilities.

Activation Functions

Alternatively, the hyperbolic tangent is used frequently:

$$\tanh(z). \tag{11}$$

For classification with $C > 1$ classes, layer $(L+1)$ uses the softmax activation function:

$$y_i^{(L+1)} = \sigma(z^{(L+1)}, i) = \frac{\exp(z_i^{(L+1)})}{\sum_{k=1}^{C} \exp(z_k^{(L+1)})}. \tag{12}$$

Then, the output can be interpreted as posterior probabilities.

Next Up ...

Network Training – Notions

By now, we have a general model $y(\cdot, w)$ depending on W weights.

Idea: Learn the weights to perform

- regression,
- or classification.

We focus on classification.

Network Training – Training Set

C classes:
1-of-C coding scheme

Given a training set

$$U_S = \{(x_n, t_n) : 1 \leq n \leq N\}, \tag{13}$$

learn the mapping represented by U_S ...

by minimizing the squared error

$$E(w) = \sum_{n=1}^{N} E_n(w) = \sum_{n=1}^{N} \sum_{i=1}^{C} \left(y_i(x_n, w) - t_{n,i}\right)^2 \tag{14}$$

using iterative optimization.

Network Training – Training Set

C classes:
1-of-C coding scheme

Given a training set

$$U_S = \{(x_n, t_n) : 1 \leq n \leq N\}, \quad (13)$$

learn the mapping represented by U_S ...

by minimizing the squared error

$$E(w) = \sum_{n=1}^{N} E_n(w) = \sum_{n=1}^{N} \sum_{i=1}^{C} \left(y_i(x_n, w) - t_{n,i}\right)^2 \quad (14)$$

using iterative optimization.

Training Protocols

We distinguish ...

Stochastic Training A training sample (x_n, t_n) is chosen at random, and the weights w are updated to minimize $E_n(w)$.

Batch and Mini-Batch Training A set $M \subseteq \{1, \ldots, N\}$ of training samples is chosen and the weights w are updated based on the cumulative error $E_M(w) = \sum_{n \in M} E_n(w)$.

Besides, online training is possible, as well.

Training Protocols

We distinguish ...

Stochastic Training A training sample (x_n, t_n) is chosen at random, and the weights w are updated to minimize $E_n(w)$.

Batch and Mini-Batch Training A set $M \subseteq \{1, \dots, N\}$ of training samples is chosen and the weights w are updated based on the cumulative error $E_M(w) = \sum_{n \in M} E_n(w)$.

Besides, online training is possible, as well.

Training Protocols

We distinguish ...

Stochastic Training A training sample (x_n, t_n) is chosen at random, and the weights w are updated to minimize $E_n(w)$.

Batch and Mini-Batch Training A set $M \subseteq \{1, \ldots, N\}$ of training samples is chosen and the weights w are updated based on the cumulative error $E_M(w) = \sum_{n \in M} E_n(w)$.

Besides, online training is possible, as well.

Thank You!