Lecture 12: Multilayer Perceptron

COMP90049

Semester 2, 2021

Qiuhong Ke, CIS

About Me

Experience:

- 2020.01 - now: Lecturer in The University of Melbourne
- 2018.05-2019.12: Post-doc in Max Planck Institute for Informatics, Germany
- 2015.02-2018.04: PhD in The University of Western Australia

Research: computer vision and machine learning

Contact: qiuhong.ke@unimelb.edu.au

Roadmap

So far:

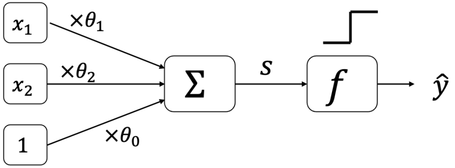
- Classifiers: KNN, Naive Bayes, Logistic Regression and Perceptron
- Feature Selection
- Evaluation

Today: Multilayer Perceptron

- Introduction
- Architecture
- Reflections

Introduction

Perceptron



$$\hat{y} = f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Single processing 'unit'
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)
- Perceptron vs Logistic Regression?

Linear classifiers

Perceptron

$$\hat{y} = f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

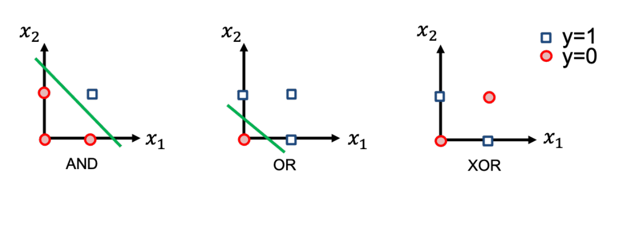
- Single processing ‘unit’
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)

Logistic Regression

$$P(y = 1|x; \theta) = f(\theta^T x) = \frac{1}{1 + \exp(-(\sum_{f=0}^{F} \theta_f x_f))}$$

- View 1: Model of $P(y = 1|x)$, maximizing the data log likelihood
- View 2: Single processing ‘unit’
- Activation: sigmoid (continuous, differentiable)

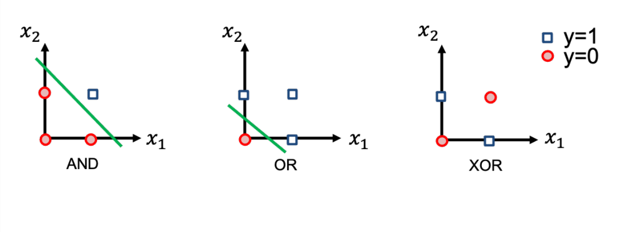
Limitations of linear classifiers



Linear classifier:

- Decision boundary is a linear combination of features $\sum_i \theta_i x_i$
- Cannot learn 'feature interactions' naturally
- can solve only linearly separable problems

Motivations of multilayer perceptron



Possible solution: composition

$$x_1 \text{ XOR } x_2 = (x_1 \text{OR } x_2) \text{ AND } [\text{NOT}(x_1 \text{AND } x_2)]$$

Non-linear separable data classification:

compose perceptrons $\rightarrow$ Multilayer perceptron

Neural Networks

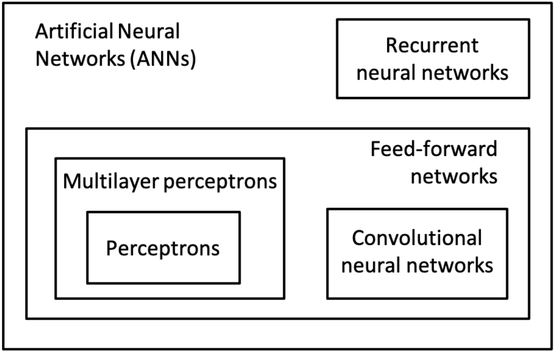
- Connected sets of many such units
- Connected into many layers $\rightarrow$ **Deep** neural networks

Multilayer Perceptron

- This lecture!
- One specific type of neural network
- Feed-forward
- Fully connected
- Supervised learner

THE UNIVERSITY OF MELBOURNE

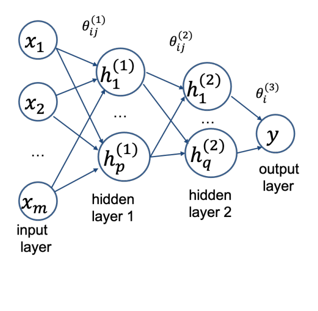
Neural networks:“Animals” in the zoo



art: OpenClipartVectors at pixabay.com (CC0)

Architecture

Multilayer Perceptron



Terminology

- depth: number of hidden layers and output layer.
- Each layer l has a number of units K_l. K_l is the **width** of layer l.
- Each layer l is **fully connected** to its neighboring layers $l-1$ and $l+1$

THE UNIVERSITY OF MELBOURNE

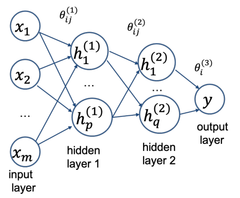
Prediction

$$h_j^{(1)} = \phi^{(1)}\left(\sum_{i=1}^{m} \theta_{ij}^{(1)} x_i + \theta_{0j}^{(1)}\right) \quad \text{or } h^{(1)} = \phi^{(1)}\left(\theta^{(1)T} x\right), \text{ assume } x_0 = 1$$

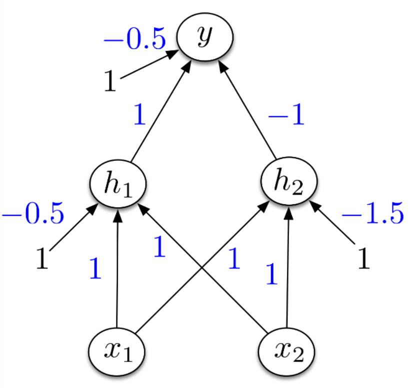
$$h_j^{(2)} = \phi^{(2)}\left(\sum_{i=1}^{p} \theta_{ij}^{(2)} h_i^{(1)} + \theta_{0j}^{(2)}\right) \quad \text{or } h^{(2)} = \phi^{(2)}\left(\theta^{(2)T} h^{(1)}\right), \text{ assume } h_0^{(1)} = 1$$

$$y = \phi^{(3)}\left(\sum_{i=1}^{q} \theta_{i}^{(3)} h_i^{(2)} + \theta_{0j}^{(3)}\right) \quad \text{or } y = \phi^{(3)}\left(\theta^{(3)T} h^{(2)}\right), \text{ assume } h_0^{(2)} = 1$$

- (non-linear) activation function for layer l as $\phi^{(l)}$
- one weight $\theta_{ij}^{(l)}$ for each connection ij (unit i in layer $l-1$ layer and unit j unit in layer l)



A Multilayer Perceptron for XOR



$$\phi(x) = \begin{cases} 1 & \text{if } x >= 0 \\ 0 & \text{if } x < 0 \end{cases}$$ and recall: $h_j^{(l)} = \phi^{(l)}\left(\sum_i \theta_{ij}^{(l)} h_i^{(l-1)} + \theta_j^{(l)}\right)$

Source: `https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L05%20Multilayer%20Perceptrons.pdf`

Multilayer Perceptron I: Inputs

Feature Engineering

- informative features, e.g., $\texttt{outlook} \in \{\textit{overcast}, \textit{sunny}, \textit{rainy}\}$, $\texttt{wind} \in \{\textit{high}, \textit{low}\}$ etc.
- Require **domain knowledge**
- Require feature selection

Example Classification dataset

Feature engineering on Weather dataset:

Outlook	Temperature	Humidity	Windy	True Label
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
...				

Raw data of Weather dataset:

Date	measurements					True Label
01/03/1966	0.4	4.7	1.5	12.7	...	no
01/04/1966	3.4	-0.7	3.8	18.7	...	no
01/05/1966	0.3	8.7	136.9	17	...	yes
01/06/1966	5.5	5.7	65.5	2.7	...	yes

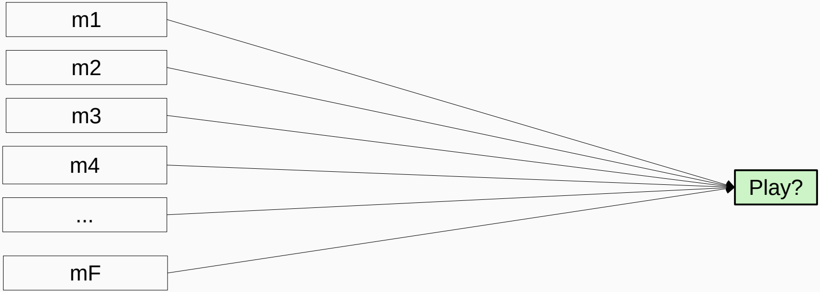
Multilayer Perceptron: Input layer

Feature learning:

- Neural networks can take as input ‘raw’ data
- Neural networks learn features themselves as intermediate representations
- feature engineering is replaced at the cost of additional parameter tuning (layers, activation functions, learning rates, ...)

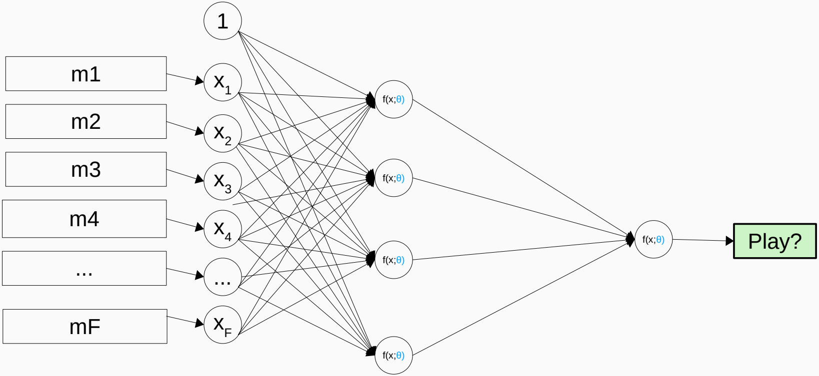
Multilayer Perceptron for Classification

Example Problem: Raw data of Weather Dataset



Multilayer Perceptron for Classification

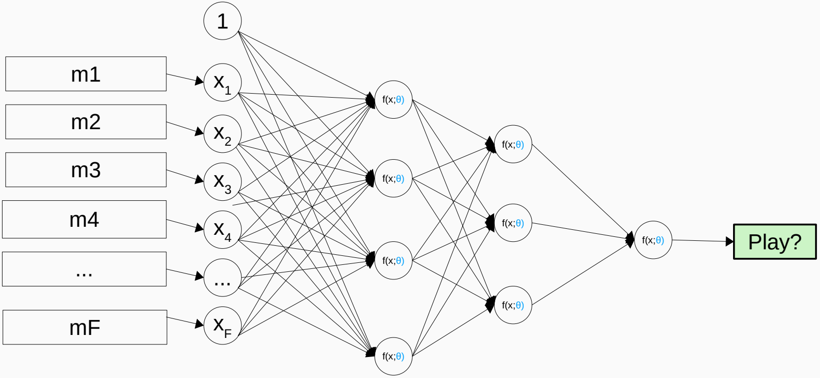
Example Problem: Raw data of Weather Dataset



Input layer, 1 hidden layer, output layer

Multilayer Perceptron for Classification

Example Problem: Raw data of Weather Dataset



Input layer, 2 hidden layers, output layer

Multilayer Perceptron I: Inputs

Inputs and feature functions

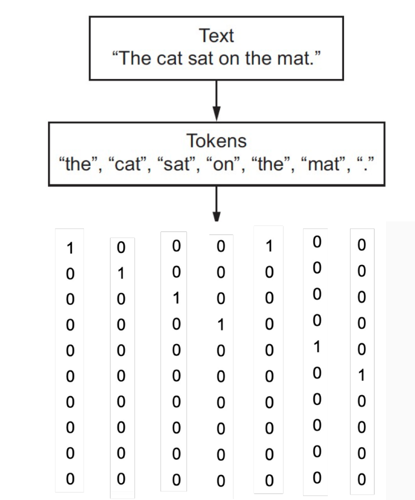
- x could be numerical measurements, e.g., $\{$blood pressure, height, age, weight, ...$\}$
- x could be a texts, i.e., a sequence of words
- x could be an image, i.e., a matrix of pixels

Non-numerical data need to be mapped to numerical

- For language:
  - 1-hot encoding: $dim(x) = V$ (words in the vocabulary)
  - pre-trained word embedding vectors: $dim(x) = k$ (embedding feature dimension $k < V$)
- For pixels, map to RGB, or other visual features

One-hot Embedding

dic={ “*the*” , “*cat*” , “*sat*” , “*on*” , “*mat*” , “.” , “*these*” , “*are*” , “*other*” , “*words*” }



Multilayer Perceptron II: Hidden Layer- Activation Functions

- Each layer has an associated activation function (e.g., sigmoid, ReLU, ...)
- Represents the extent to which a neuron is 'activated' given an input
- Each hidden layer performs a **non-linear transformation** of the input
- **the activation functions must be non-linear**, as without this, the model is simply a (complex) linear model

Popular activation functions

1. logistic (aka sigmoid) ("σ"):

$$f(x) = \frac{1}{1 + e^{-x}}$$

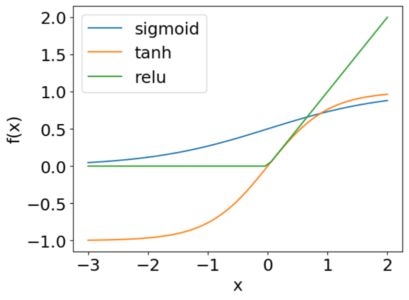
2. hyperbolic tan ("$\mathtt{tanh}$"):

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

3. rectified linear unit ("ReLU"):

$$f(x) = \max(0, x)$$

note not differentiable at $x = 0$



Multilayer Perceptron III: Output Functions

Neural networks can learn different concepts: **classification**, **regression**, ...
The **output function** depends on the concept of intereest.

- Binary classification:
  - one neuron, with sigmoid function
- Multiclass classification:
  - typically **softmax** to normalize K outputs from the pre-final layer into a probability distribution over classes

$$p(y_i = j | x_i; \theta) = \frac{exp(z_j)}{\sum_{k=1}^{K} exp(z_k)}$$

- Regression:
  - identity function
  - possibly other continuous functions such as sigmoid or tanh

Reflections

Linear vs Non-linear classifiers

Linear classifier

- Decision boundary is a linear combination of features $\sum_i \theta_i x_i$
- Cannot learn ‘feature interactions’ naturally
- can solve only linearly separable problems

Non-linear classifier

- Neural networks with at least 1 hidden layer and non-linear activations are non-linear classifiers
- Decision boundary is a non-linear function of the inputs
- Capture “feature interactions”

Pros and Cons of Neural Networks

Pros

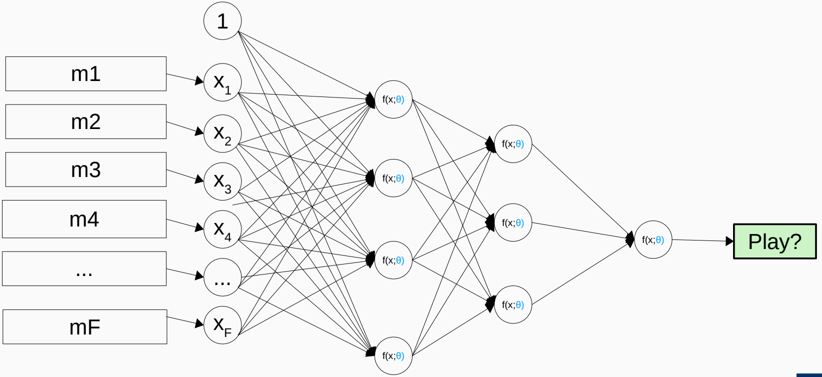
- Powerful tool!
- Neural networks with at least 1 hidden layer (with non-linear activation function) can approximate any (continuous) function.
- Automatic feature learning
- Empirically, very good performance for many diverse tasks

Cons

- Powerful model increases the danger of ‘overfitting’
- Requires large training data sets
- Often requires powerful compute resources (GPUs)
- Lack of interpretability

Lack of interpretability

Input units become indistinguishable: What input units lead to the output?



When is Linear Classification Enough?

- If we know our classes are linearly (approximately) separable
- If the feature space is (very) high-dimensional

  ...i.e., the number of features exceeds the number of training instances
- If the traning set is small
- If *interpretability* is important, i.e., understanding how (combinations of) features explain different predictions

Neural Networks Structure

Network Structure

- Sequence of hidden layers $l_1, \ldots, l_L$ for a netword of depth L
- Each layer l has K_l parallel neurons (breadth)
- Many layers (depth) vs. many neurons per layer (breadth)? Empirical question, theoretically poorly understood.

Advanced tricks include allowing for exploiting data structure

- convolutions (convolutional neural networks; CNN), Computer Vision
- recurrencies (recurrent neural networks; RNN), Natural Language Processing
- attention (efficient alternative to recurrencies)
- ...

Beyond the scope of this class.

THE UNIVERSITY OF MELBOURNE

Summary

Today

- From perceptrons to neural networks
- multilayer perceptron prediction
- Input layer (data), hidden layer (activation functions),output layer (output functions)
- features and limitations

Next Lecture

- Learning parameters of neural networks
- The Backpropagation algorithm

References

Jacob Eisenstein (2019). *Natural Language Processing*. MIT Press. Chapters 3 (intro), 3.1, 3.2. https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf

Dan Jurafsky and James H. Martin. *Speech and Language Processing*. Chapter 7.2, 7.3. Online Draft V3.0. https://web.stanford.edu/~jurafsky/slp3/

THE UNIVERSITY OF MELBOURNE