

Slides:

goo.gl/xzPpX3

Repo:

goo.gl/ge4SsY

Neural Networks

Machine Learning Workshop Series

Angelos Filos

November 28, 2017

Imperial College Data Science Society

Table of Contents

1. Theory
2. Application
3. Codelab

Theory

Problem Definition

Provided a set \mathcal{S} of (x_i, y_i) pairs:

$$\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$$

Find a function f that maps $x_i \rightarrow y_i$:

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

f is a:

- **(Discriminate) Model**
- **Map Function**
- **Function Approximation**

for \mathcal{S} , such that:

$$f(x_i) = \hat{y}_i \approx y_i, \quad \forall i$$

Storage vs. Approximation

Map (hashmap, dict, hash table)

$$f[xi] \leftarrow yi$$

Suitable for:

- Finite space
- Data storage

Model (Neural Network, Gaussian Process, etc)

$$f(xi) \approx yi$$

Suitable for:

- **Infinite** space
- Function approximation

Linear Model

f is a **Linear Model**, iff:

Math

For a finite \mathcal{S} where $|\mathcal{S}| = k$:

$$f(\mathbf{X}) = \hat{\mathbf{y}} = \mathbf{X} * \mathbf{w} + \mathbf{b} \quad (1)$$

where: $\mathbf{X} \in \mathbb{R}^{k \times n}$; $\hat{\mathbf{y}} \in \mathbb{R}^{k \times m}$; $\mathbf{w} \in \mathbb{R}^{n \times m}$; $\mathbf{b} \in \mathbb{R}^m$

Graph

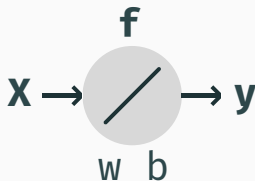


Figure 1: Linear Activation Function Neuron

Neural Network

f is a **Neural Network**, iff:

Math

For a finite \mathcal{S} where $|\mathcal{S}| = k$:

$$f(\mathbf{X}) = \hat{\mathbf{y}} = \sigma(\mathbf{X} * \mathbf{w}_1 + \mathbf{b}_1) * \mathbf{w}_2 + \mathbf{b}_2 \quad (2)$$

where: $\mathbf{X} \in \mathbb{R}^{k \times n}$; $\hat{\mathbf{y}} \in \mathbb{R}^{k \times m}$; $\mathbf{w}_1 \in \mathbb{R}^{n \times c}$; $\mathbf{b}_1 \in \mathbb{R}^c$; $\mathbf{w}_2 \in \mathbb{R}^{c \times m}$; $\mathbf{b}_2 \in \mathbb{R}^m$

Graph

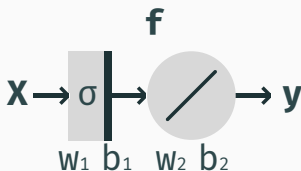


Figure 2: Single Hidden Layer Neural Network

Universal Approximation Theorem

Statement (Existence Theorem)

Any function can be approximated with **arbitrary precision** by some single hidden layer neural network.

Attention

This theorem does not suggest any systematic way of finding such a network, just a proof that it exists. There are other methods, such as:

- **Backpropagation Algorithm**
- general **Evolutionary Algorithms**

that are used for finding a neural network that satisfy this.

Mini Demo

- **Loss Function \mathcal{L} :** choose a loss function to evaluate the model.

$$\mathcal{J}(\mathbf{w}, \mathbf{b}) = \mathcal{L}(\mathbf{y} - f(\mathbf{x}_i; \mathbf{w}, \mathbf{b}))$$

| Application | Loss Function |
|----------------------------|---------------------------------|
| Regression | Mean Squared Error (MSE) |
| Multi-class Classification | Cross Entropy Error |

Table 1: Choosing Loss Function \mathcal{L}

- **Stochastic Gradient Descent:** minimize \mathcal{J} with respect to \mathbf{w}, \mathbf{b} .

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta \frac{\partial \mathcal{J}}{\partial \mathbf{w}} \quad \text{and} \quad \mathbf{b}^{(t+1)} = \mathbf{b}^{(t)} + \eta \frac{\partial \mathcal{J}}{\partial \mathbf{b}}$$

- **Backpropagation Algorithm:** efficient calculation of $\frac{\partial \mathcal{J}}{\partial \mathbf{w}_i}$ and $\frac{\partial \mathcal{J}}{\partial \mathbf{b}_i}$.

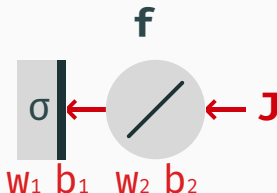
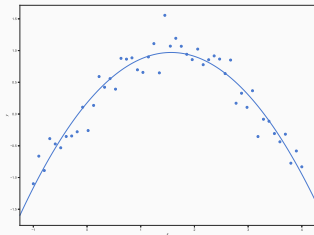


Figure 3: Backpropagation Algorithm

Backpropagation Algorithm relies on **Chain Rule**, reversing the computation of the gradients and caching them for efficient calculation of the gradients of the previous layers.

Application

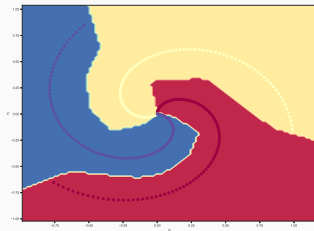
Regression



$$y \in \mathcal{R}$$

where \mathcal{R} is a **continuous** set.

Classification



$$y \in \mathcal{D}$$

where \mathcal{D} is a **discrete** set.

Approach i

Dataset

1. Get data \mathcal{S}
2. Define:
 - features matrix $\mathbf{X} \in \mathbb{R}^{k \times n}$
 - targets matrix $\mathbf{y} \in \mathbb{R}^{k \times m}$
3. (optional) Randomly split data to **train** & **test**

Neural Network Layers

1. `input_shape` = `n`, where `n` the number of features (columns) of \mathbf{X}
2. `hidden_layer_shape` = `??`
3. `output_shape` = `m`, where `m` the number of components of \mathbf{y}

Approach ii

Problem Nature

- $y \in \mathcal{D}$, \mathcal{D} : **discrete** \rightarrow **Classification**
- $y \in \mathcal{R}$, \mathcal{R} : **continuous** \rightarrow **Regression**

Loss Function

- **Classification** \rightarrow **Cross Entropy Error**
- **Regression** \rightarrow **Mean Squared Error**

Optimisation

1. **Stochastic Gradient Descent**
2. **Backpropagation Algorithm**

Checklist

| Step | Question | Answer |
|------|--------------------------------------|--------|
| 1 | Labeled Data? | |
| 2 | Number of features of \mathbf{X} | |
| 3 | Number of components of \mathbf{y} | |
| 4 | Size of hidden layer | |
| 5 | Problem Nature | |
| 6 | Loss Function | |
| 7 | Update Rules | |
| 8 | Accuracy | |

Table 2: Application using Neural Network Checklist

Questions?

Codelab

Setup

1. Create a Github account.
2. Sign-in cocalc using your Github credentials.
3. Create a new project in cocalc.
4. Clone (green button at top RHS) in zip format the **Neural Networks** repository.
5. Upload the zip file to newly created cocalc project.
6. Click on the zip file and extract the compressed files.
7. Navigate to the extracted folder
`Neural-Networks-master/notebooks/Demo.ipynb`
8. Change the kernel by:
`Kernel → Change Kernel → Python 3 (Anaconda)`
9. Run the project by:
`Kernel → Restart & Run All → Restart and Run All Cells`

Computer Vision

Multi-class classification of handwritten digits for MNIST[5] dataset using:

1. a Linear Model
2. a Single Layer Feedforward Neural Network

References



cocalc.

Collaborative Calculation in the Cloud, 2017.

Online; accessed 06 Nov 2017; available at
<https://cocalc.com/>.



A. Filos.

Linear Models, 2017.

Online; accessed 09 Nov 2017; available at
<https://goo.gl/H65adq>.



Github.

Built for developers, 2017.

Online; accessed 06 Nov 2017; available at
<https://github.com>.



M. Nielsen.

A visual proof that neural nets can compute any function, 2017.

Online; accessed 10 Nov 2017; available at

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the participants individually and, unless expressly stated to the contrary, are not the opinion or position of the ICDSS, its cosponsors, or its committees. The ICDSS does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented. Attendees should note that sessions are video-recorded and may be published in various media, including print, audio and video formats without further notice.