Multi Layer Perceptron

M. Saiful Bari

October 11, 2017

Table of Contents

Introduction

Feedforward Neural Network aka (**M**)ulti-(**L**)ayer (**P**)erceptron (**MLP**)

Series of **logistic regression** models **stacked** on top of each other, with the **final layer** being either another **logistic** or a **linear regression** model.

MLP : Example

Assume, we have **two layers**, and we are solving a **regression problem**, the model has the form,

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^{\mathsf{T}}\mathbf{z}(\mathbf{x}), \sigma^2)$$
$$\mathbf{z}(\mathbf{x}) = g(\mathbf{V}\mathbf{x}) = [g(\mathsf{v}_1^{\mathsf{T}}\mathsf{x}), ..., g(\mathsf{v}_{\mathsf{H}}^{\mathsf{T}}\mathsf{x})]$$

where,

- g is a non-linear activation or transfer function (commonly the logistic function).
- $\mathbf{z}(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{V})$ is called the hidden layer (a deterministic function of the input)
- H is the number of hidden units.
- V is the **weight matrix** from the inputs to the hidden nodes
- w is the **weight vector** from the hidden nodes to the output

MLP

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^{\mathsf{T}}\mathbf{z}(\mathbf{x}), \sigma^2)$$
$$\mathbf{z}(\mathbf{x}) = g(\mathbf{V}\mathbf{x}) = [g(\mathsf{v}_1^{\mathsf{T}}\mathsf{x}), ..., g(\mathsf{v}_H^{\mathsf{T}}\mathsf{x})]$$

It is important that g be nonlinear, otherwise the whole model collapses into a large linear regression model of the form
$\mathbf{y} = \mathbf{w}^{\mathsf{T}}(\mathbf{V}\mathbf{x})$

One can show that an **MLP** is a **universal approximator**, meaning it can model any suitably **smooth function**, given enough **hidden units**, to any **desired** level of accuracy. **(Hornik 1991)**

MLP : Example

To handle binary classification, we pass the output through a sigmoid, as in a GLM.

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^{\mathbf{T}}\mathbf{z}(\mathbf{x})))$$

We can easily extend the MLP to predict multiple outputs. For example, in the regression case, we have

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}|\boldsymbol{W}\phi(\mathbf{x}, \mathbf{V}), \sigma^2\boldsymbol{I})$$

MLP : Example

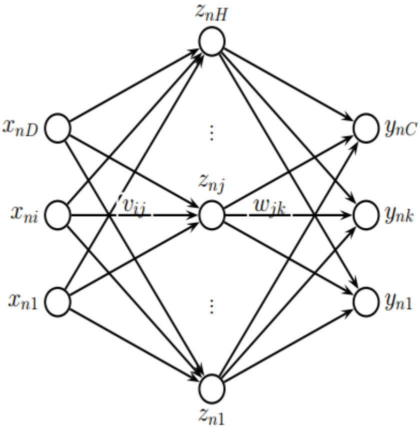

Figure 1: A neural network with one hidden layer

MLP : Example

If we add **mutual inhibition arcs** between the **output units**, ensuring that only one of them turns on, we can enforce a **sum-to-one** constraint, which can be used for **multi-class classification**.

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathrm{Cat}(y|\mathcal{S}(\boldsymbol{W}\mathbf{z}(\mathbf{x})))$$

The Backpropagation Algorithm

- Unlike a Generalized Linear Model**(GLM)**, the Negative Log Likelihood**(NLL)** of an MultiLayer Perceptron**(MLP)** is a **non-convex** function of its parameters.
- We can find a **locally optimal** ML or MAP estimate using **standard gradient-based optimization** methods.
- since **MLP**s have lots of parameters, they are often trained on very large data sets.
- **MLP**s have lots of parameters, they are often trained on very large data sets