# Multi-Layer Perceptrons and Deep Learning

Christos Dimitrakakis

November 7, 2023

# Outline

# Layering and features

## Fixed layers

- Input to layer $x \in R^n$
- Output from layer $\hat{y} \in R^m$.

## Intermediate layers

Combinations of

- Linear layer
- Non-linear activation function.

## Linear layers types

- Dense
- Sparse
- Convolutional

## Activation funnction

Simple transformations of previous output.

- Sigmoid
- Softmax

# Linear layers

## Example: Linear regression with $n$ inputs, $m$ outputs.

- Input: Features $\boldsymbol{x} \in \mathbb{R}^n$
- Dense linear layer with $\boldsymbol{\Theta} \in \mathbb{R}^{m \times n}$
- Output: $\hat{\boldsymbol{y}} \in \mathbb{R}^m$

## Dense linear layer

- Parameters $\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_m \end{pmatrix}$,
- $\boldsymbol{\theta}_i = [\theta_{i,1}, \ldots, \theta_{i,n}]$, $\boldsymbol{\theta}_i$ connects the $i$-th output $y_i$ to the features $\boldsymbol{x}$:

$$y_i = \boldsymbol{\theta}_i \boldsymbol{x}$$

- In compact form:

$$\boldsymbol{y} = \boldsymbol{\Theta} \boldsymbol{x}$$

# Sigmoid activation

## Example: Logistic regression

- Input $\boldsymbol{x} \in \mathbb{R}^n$
- Intermediate output: $z \in \mathbb{R}$,

$$z = \sum_{i=1}^{n} \theta_i x_i.$$

- Output $\hat{y} \in [0, 1]$.

## Definition

This activation ensures we get something we can use as a probability

$$f(z) = 1/[1 + \exp(z)].$$

Now $P_{\boldsymbol{\theta}}(y = 1|x) = \hat{y}$.

# Softmax layer

## Example: Multivariate logistic regression with $m$ classes.

- Input: Features $\boldsymbol{x} \in \mathbb{R}^n$
- Middle: Fully-connected Linear activation layer $\boldsymbol{z} = \boldsymbol{\Theta}\boldsymbol{x}$.
- Output: $\hat{\boldsymbol{y}} \in \mathbb{R}^m$

## Softmax output layer

We want to translate the real-valued $z_i$ into probabilities:

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

Now $P_{\boldsymbol{\Theta}}(y = i|\boldsymbol{x}) = \hat{y}$

# Random projections

- Features $x$
- Hidden layer activation $z$
- Output $y$

## Hidden layer: Random projection

Here we project the input into a high-dimensional space

$$z_i = \mathrm{sgn}(\boldsymbol{\theta}_i^\top x) = y_i$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_i]_{i=1}^m$.

## The reason for random projections

- The high dimension makes it easier to learn.
- The randomness ensures we are not learning something spurious.

# Background on back-propagation

## The problem

- We need to minimise a loss function $\ell$
- We need to calculate

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\theta}}[\ell] \approx \frac{1}{T} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} c(x_t, y_t, \boldsymbol{\theta}).$$

- However $c(x_t, y_t, \boldsymbol{\theta})$ is a complex non-linear function of $\boldsymbol{\theta}$.

## The solution

- [1673] Liebniz, the chain rule of differentiation.
- [1976] Rosenblat's perceptron without realising it!
- [1982] Werbos applied it to MLPs.
- [1986] Rummelhart, Hinton and Williams popularised it.

# Back-propagation

## The chain rule

$$f : X \to Z, \qquad g : Z \to Y, \qquad \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}, \qquad \nabla_x g = \nabla_f g \nabla_x f$$

## Linear regression

- $f_{\boldsymbol{\theta}}(x) = \sum_{i=1}^{n} \theta_i x_i$.
- $\mathbb{E}_{\boldsymbol{\theta}}[\ell] \approx \ell(D, \boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} c(\boldsymbol{\theta}, \boldsymbol{x}_t, y_t)$.

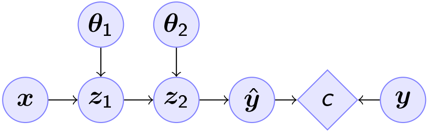
$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} c(\boldsymbol{\theta}, \boldsymbol{x}_t, y_t) &= \nabla_{\boldsymbol{\theta}} [\underbrace{f_{\boldsymbol{\theta}}(x_t) - y_t}_{z}]^2, \qquad g(z) = z^2 && (1)\\
&= \nabla_z g(z) \nabla_f z \nabla_{\boldsymbol{\theta}} f(x_t) && (2)\\
&= 2[f_{\boldsymbol{\theta}}(x_t) - y_t] \nabla_f [f_{\boldsymbol{\theta}}(x_t) - y_t] \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_t) && (3)\\
&= 2[f_{\boldsymbol{\theta}}(x_t) - y_t] \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_t) && (4)
\end{aligned}$$

# Gradient descent with *back-propagation*

## Inputs

- Dataset $D$, cost function $\ell = \sum_t c_t$
- Parametrised architecture with $k$ layers
  - Parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$
  - Intermediate variables: $\boldsymbol{z}_j = f_j(\boldsymbol{z}_{j-1}, \boldsymbol{\theta}_j)$, $\boldsymbol{z}_0 = \boldsymbol{x}$, $\boldsymbol{z}_k = \hat{\boldsymbol{y}}$.

## Dependency graph




## Backpropagation with steepest stochastic gradient descent

- Forward step: For $j = 1, \ldots, k$, calculate $\boldsymbol{z}_j = f_j(k)$ and $c(\hat{\boldsymbol{y}}, \boldsymbol{y})$
- Backward step: Calculate $\nabla_{\hat{\boldsymbol{y}}} c$ and $d_j \triangleq \nabla_{\boldsymbol{\theta}_j} c = \nabla_{\boldsymbol{\theta}_j} z_j d_{j+1}$ for $j = k \ldots, 1$
- Apply gradient: $\boldsymbol{\theta}_j -= \alpha d_j$.

# Linear layer

## Definition

This is a linear combination of inputs $x \in \mathbb{R}^n$ and parameter matrix

$$\boldsymbol{\Theta} \in \mathbb{R}^{m \times n} \text{ where } \boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_i \\ \vdots \\ \boldsymbol{\theta}_m \end{bmatrix} = \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,j} & \cdots & \theta_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \dots \\ \theta_{i,1} & \cdots & \theta_{i,j} & \cdots & \theta_{i,m} \\ \vdots & \ddots & \ddots & \ddots & \dots \\ \theta_{n,1} & \cdots & \theta_{i,j} & \cdots & \theta_{n,m} \end{bmatrix}$$

$$f(\boldsymbol{\Theta}, \boldsymbol{x}) = \boldsymbol{\Theta}\boldsymbol{x} \qquad f_i(\boldsymbol{\Theta}, \boldsymbol{x}) = \boldsymbol{\theta}_i \cdot \boldsymbol{x} = \sum_{j=1}^{n} \theta_{i,j} x_j,$$

## Gradient

Each partial derivative is simple:

$$\frac{\partial}{\partial \theta_{i,j}} f_k(\boldsymbol{\Theta}, \boldsymbol{x}) = x_j$$

# Sigmoid layer

$$f(z) = 1/(1 + \exp(-z))$$

## Derivative

So let us ignore the other inputs for simplicity:

$$\frac{d}{dz} f(z) = \exp(-z)/[1 + \exp(-z)]^2$$

# Softmax layer

$$y_i(\boldsymbol{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

## Derivative

$$\frac{\partial}{\partial z_i} y_i(\boldsymbol{z}) = \frac{e^{z_i} e^{\sum_{j \neq i} z_j}}{\left(\sum_j e^{z_j}\right)^2}$$

$$\frac{\partial}{\partial z_i} y_k(\boldsymbol{z}) = \frac{e^{z_i + z_k}}{\left(\sum_j e^{z_j}\right)^2}$$