STAT 535

10/26/23

# Lecture 9

2 layer NNs

Q2 Thu Nov 2

L II

L III NN posted (may be updates)

HW 4 TBPosted

# Lecture Notes III – Neural Networks

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2023

Two-layer Neural Networks 

Multi-layer neural networks

A zoo of multilayer networks

**Reading** HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

# Two-layer Neural Networks

- The **activation function** (a term borrowed from neuroscience) is any continuous, bounded and strictly increasing function on $\mathbb{R}$. Almost universally, the activation function is the **logistic** (or **sigmoid**)

$$\phi(u) = \frac{1}{1+e^{-u}} \tag{1}$$

because of its nice additional computational and statistical properties.

- We build a **two-layer neural network** in the following way:

| | | |
|---|---|---|
| Inputs | $x_k$ | $k = 1:d$ |
| Bottom layer[1] | $z_j = \phi(w_j^T x)$ | $j = 1:m$, $w_j \in \mathbb{R}^d$ |
| Top layer | $f = \beta^T z$ | $\beta \in \mathbb{R}^m$ |
| Output | $f \in \mathbb{R}$ | |

In other words, the neural network implements the function

$$f(x) = \sum_{j=1}^{m} \beta_j z_j = \sum_{j=1}^{m} \beta_j \phi\Big(\sum_{k=1}^{d} w_{kj} x_k\Big) \in (-\infty, \infty) \tag{2}$$

Note that this is just a linear combination of logistic functions.

$x \in \mathbb{R}^d$, $W_{1:m} \in \mathbb{R}^d$, $z_{1:m} = \varphi(W_{1:m}^T x) =$ logistic regression, $\beta \in \mathbb{R}^m$

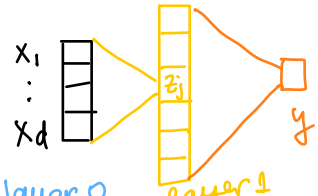
[1] In neural net terminology, each variable $z_j$ is a **unit**, the bottom layer is **hidden**, while top one is **visible**, and the units in this layer are called hidden/visible units as well. Sometimes the inputs are called **input units**; imagine neurons or individual circuits in place of each $x, y, z$ variable.

Variables = units

Layers

$z_j = \varphi(W_j^T x)$ ← logistic regression

$y = \beta^T z$




$m$ = hidden units

↳ new features

$\beta \in \mathbb{R}^m$ $y = \beta^T z$

parameters = weights $(W, \beta)$

hidden units = represent features

$$W = \begin{bmatrix} W_j^T \end{bmatrix} \in \mathbb{R}^{m \times d}$$

$$z = \varphi(W^T x)$$

↑ $\varphi$ of every element $W^T x$

architecture = structure = $m = ?$ how many layers ?

$r = 2$

etc.

# Output layer options

- **linear** layer as in (2) $f = \sum_j \beta_j z_j = \beta^T z \in \mathbb{R}$ regression
- **logistic** layer: in classification $f(x) \in [0,1]$ is interpreted as the probability of the $+$ class.

$$f(x) = \phi\left(\sum_{j=1}^{m} \beta_j z_j\right) = \phi\left(\sum_{j=1}^{m} \beta_j \phi(\sum_k w_{kj} x_k)\right) \in (0,1) \quad (3)$$

logistic regression ($z$)

- **softmax** layer in multiway classification

  The softmax function $\phi(z) : \mathbb{R}^r \to (0,1)^r$

$$\phi_k(u) = \frac{e^{u_k}}{\sum_{j=1}^{m} e^{u_j}} \quad (4)$$

$y = \{1, \dots r\}$, $y \in \mathbb{R}^r$

$k = 1:r$

- Properties
  - $\sum_{j=1}^{m} \phi_j(u) = 1$ for all $u$
  - for $u_k \gg u_j$, $j \neq k$ $\phi_k(u) \to 1$.
  - derivatives $\frac{\partial \phi_j}{\partial u_k} = \phi_k \delta_{jk} - \phi_j \phi_k$ Ex

sigmoid $\varphi'(u) = \varphi(u) - \varphi^2(u)$

$u \in \mathbb{R}$

# Softmax function $\varphi$ ← ARBITRARILY

Ex: $r=2$

$\varphi_1\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \varphi(u)_1$ ? (2)

$u \in \mathbb{R}^r$

$\varphi : \mathbb{R}^r \longrightarrow (0,1)^r$

$$\varphi_k = \frac{e^{u_k}}{\sum_{k'=1}^{r} e^{u_{k'}}}, \quad k = 1:r$$

class $k$

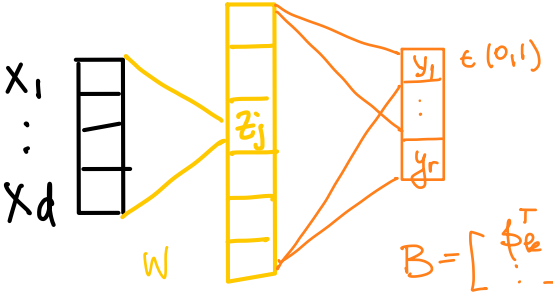
$\Rightarrow \varphi_k(z(x)) = P[y=k \mid x]$

confidence in $\hat{y}=k$

$$\hat{y}(x) = \underset{k=1:r}{\operatorname{argmax}}\ \varphi_k(x) = \underset{k=1:r}{\operatorname{argmax}}\ u_k(x)$$

$$\sum_{k=1}^{r} \varphi_k = 1$$

$\varphi_k(u) > 0$ for any $u$

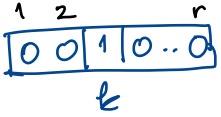



$u_k = \beta_k^T z$

$u = Bz \in \mathbb{R}^r$

Ideally $y_k(x) \approx 1$ for $k$ = true class

$y_{k'}(x) \approx 0 \quad k' \neq k$




← 1-hot encoding of $y \in \{1,..r\}$ (sparse coding)

# Generalized Linear Models (GLM)

A GLM is a regression where the "noise" distribution is in the exponential fami ly.

- $y \in \mathbb{R}$, $y \sim P_\theta$ with

$$P_\theta(y) = e^{\theta y - \ln \psi(\theta)} \tag{5}$$

- the parameter $\theta$ is a linear function of $x \in \mathbb{R}^d$

$$\theta = \beta^T x \tag{6}$$

- We denote $E_\theta[y] = \mu$. The function $g(\mu) = \theta$ that relates the mean parameter to the natural parameter is called the **link function**.

The log-likelihood (w.r.t. $\beta$) is

$$l(\beta) = \ln P_\theta(y|x) = \theta y - \ln \psi(\theta) \quad \text{where } \theta = \beta^T x \tag{7}$$

and the gradient w.r.t. $\beta$ is therefore

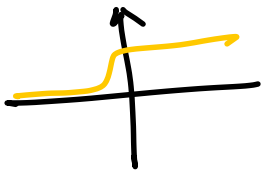
$$\nabla_\beta l = \nabla_\theta l \nabla_\beta(\beta^T x) = (y - \mu)x \tag{8}$$

This simple expression for the gradient is the generalization of the gradient expression you obtained for the two layer neural network in the homework. [Exercise: This means that the sigmoid function is the *inverse link function* defined above. Find what is the link function that corresponds to the neural network.]

# Hidden layer options

- sigmoidal functions $\phi$, *tanh*
- hinge functions RELU $= max(u, 0)$, softplus $= \ln(1 + e^u)$

bounded e.g. (0,1)



NO

↑ Exercise

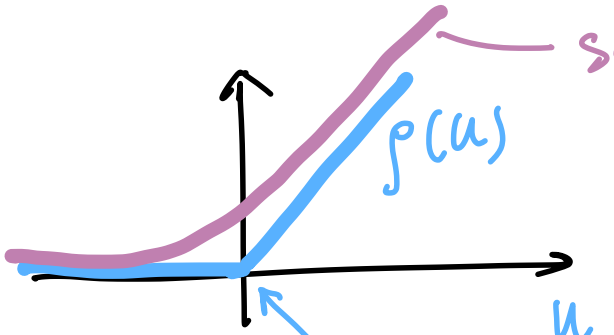
MUST : non-linear !

increasing

differentiable almost everywhere

unbounded

Rectified Linear Unit = Hinge




$\rho(u) = \max(0, u)$

$$\rho'(u) = \begin{cases} 1 & u > 0 \\ 0 & u < 0 \end{cases}$$

# What is 2 layer NN representing? (assume $y \in \{\pm 1\}$)

- $m = 1$

$f = \beta \varphi(w^T x)$

wlog $\beta = 1 \Rightarrow f = \varphi(w^T x)$

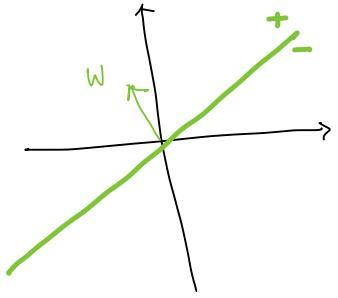

Logistic Regression

$f(x) = \beta^T z \in \mathbb{R}$

$z = \varphi(Wx) \in \mathbb{R}^m$

$\hat{y} = 1$ iff $f(x) > 0$

$\hat{y} = \text{sign}\left[f(x) - \frac{1}{2}\right]$




$x \in \mathbb{R}^2$

- $m > 1$

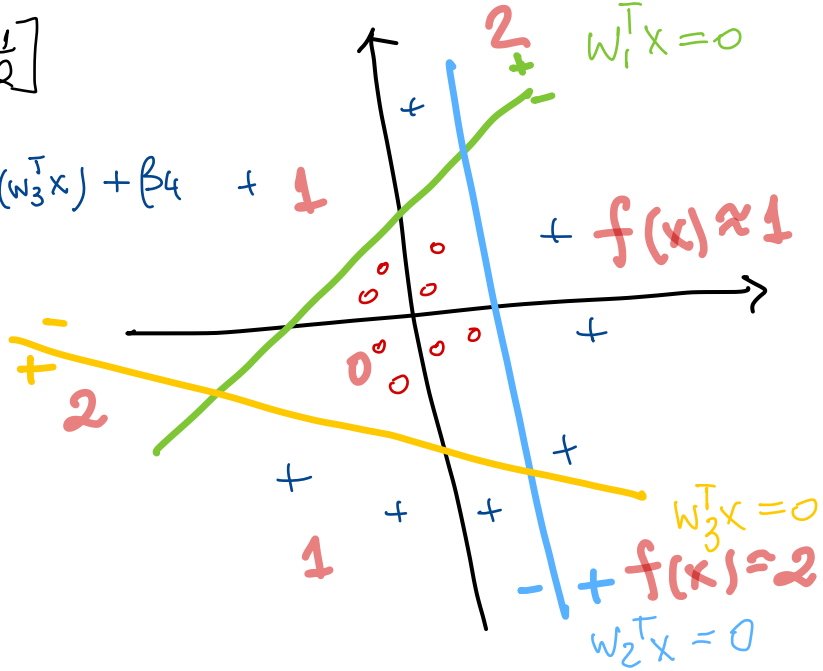
$m = 3$

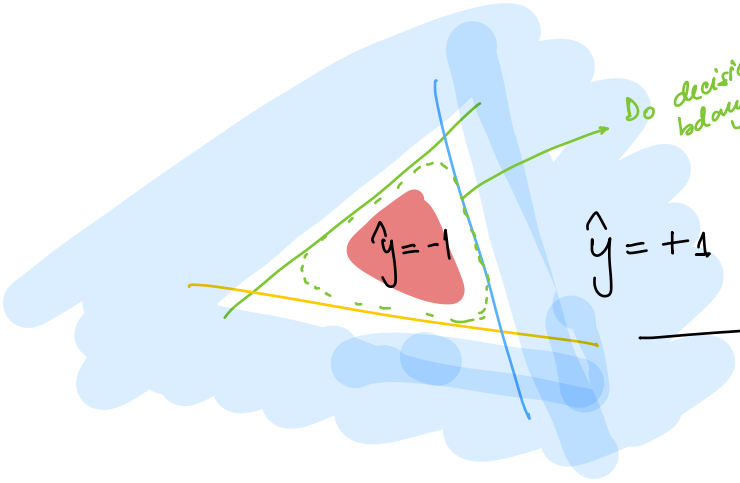
$f(x) = \beta_1 \varphi(w_1^T x) + \beta_2 \varphi(w_2^T x) + \beta_3 \varphi(w_3^T x) + \beta_4$

not through 0

$\beta_1 = \beta_2 = \beta_3 = 1$

$\beta_4 = -\frac{1}{2}$



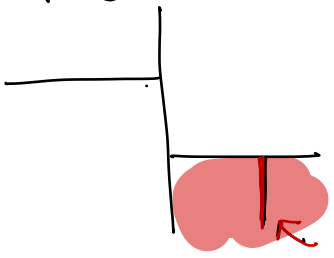



Do decision bdary ← global (over $\mathbb{R}^d$)

Do imprecise (where $f(x)=0$) slow transition $0 \to 1$

Do inaccurate (e.g true Do is ◯)

Decision Tree



local optimum in $\mathcal{X}$ space

Split depends only on

- Do imprecise (where $f(x) = 0$)
  slow transition $0 \to 1$

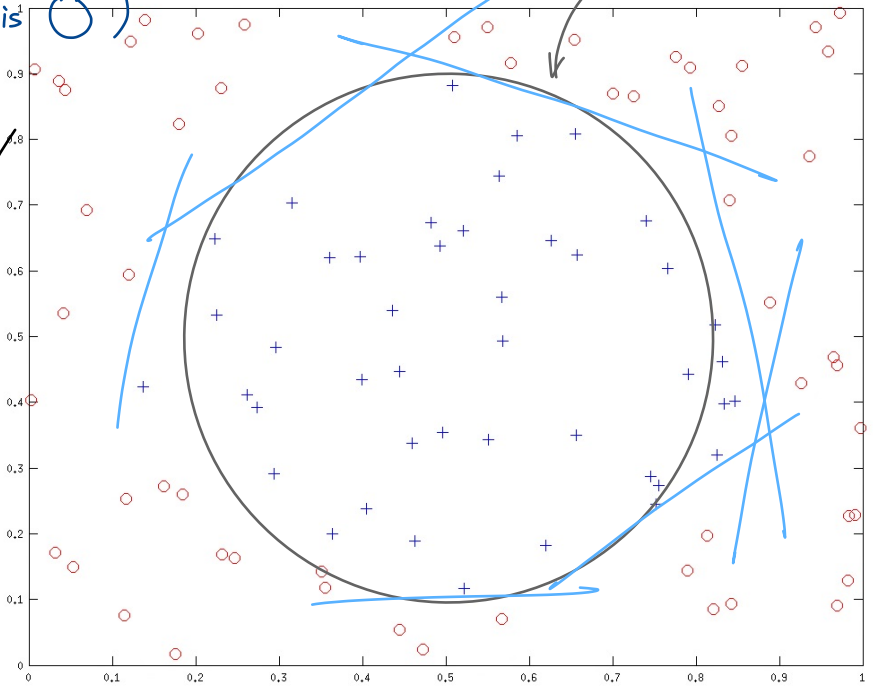
transitions sharper
$|W| \nearrow$

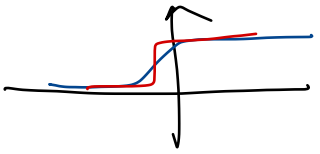
- Do inaccurate
  (e.g true $D_0$ is ○)

⇓

make $m$ larger

Bayes decision bdary



$w' > w$

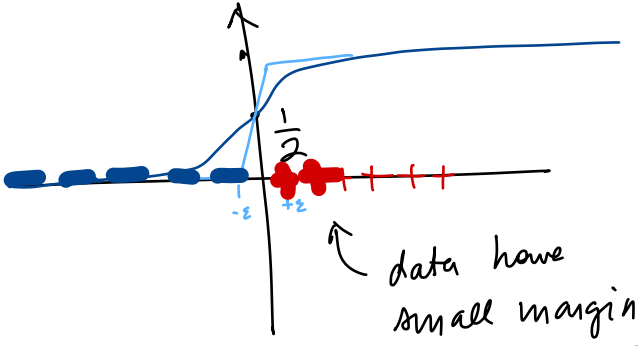
$$\varphi_{(u)} = \frac{1}{1 + e^{-wu}}$$

$w \in \mathbb{R}$

$w > 0$

$w = 0 \Rightarrow \varphi \simeq \frac{1}{2}$

$w = 1$

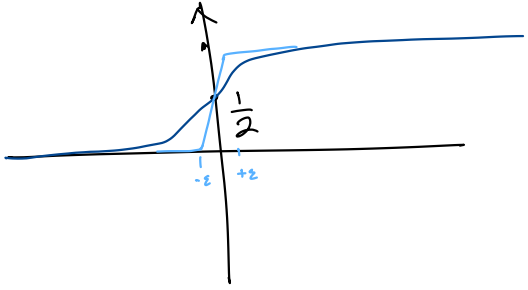



data have small margin ⇒ steep

$w = 1000$

$u = +\varepsilon \quad e^{-u} \simeq 0 \quad \varphi \simeq 1$

$u = -\varepsilon \quad e^{-u} \simeq \infty \quad \varphi = 0$



# Multi-layer/Deep neural networks

The construction can be generalized recursively to arbitrary numbers of layers.
Each layer is a linear combination of the outputs from a previous layer (a multivariate operation), followed by a non-linear transformation via the logistic function $\phi$. Let $x \equiv x^{(0)}, y \equiv x^{(L)}$, $n_0 = n, n_L = 1$ and define the recursion:

$$x_j^{(l)} = \phi\left((w_j^{(l)})^T x^{(l-1)}\right), \text{ for } j = 1 : n_l \tag{9}$$

The vector variable $x^{(l)} \in \mathbb{R}^{n_l}$ is the ouput of layer $l$ of the network. As before, the sigmoid of the last layer may be omitted.