

Department of Linguistics and Translation

香港城市大學 City University of Hong Kong

Computational Linguistics LT3233

Jixing Li Lecture 8: Feedforward neural networks

Slides adapted from Dan Jurafsky

Lecture plan

- Neural network unit
- The XOR problem
- Feedforward neural networks
- Short break (15 mins)
- Hands-on exercises

This is in your brain

Neural network unit This is not in your brain

Neural unit weight weigh bias term tional term in the sum called a bias term. Given a set of inputs *x*1*...xn*, a unit has Often it's more convenient to express this weighted sum using vector notation; recall vector from linear algebra that a vector is, at heart, **As a real valued in Eq. 7.2,** *Aleural* unit

Take weighted sum of inputs, plus a bias **blus** and an input vector **weighted** sum of inputs, plus a bias *blue* Take weighted sum of inputs, plus a bias

$$
z = b + \sum_{i} w_i x_i
$$

$$
z = w \cdot x + b
$$

Apply a nonlinear activation function f: extor a vector from the angle technical state a vector is, at the set of the set of the set of the set of the s
Anniv a nonlinear activation function f we have $\mathcal{L}(\cdot)$ Apply a nonlinear activation fur

$$
y = a = f(z)
$$

apply a non-linear function *f* to *z*. We will refer to the output of this function as

activation the activation value for the unit, *a*. Since we are just modeling a single unit, the

Non-linear activation functions

We're already seen the sigmoid for logistic regression: We're already coop the cigmoid for legistic regression

$$
u = 0.5
$$

What is the output *y* for the input *x*:

$$
x = [0.5, 0.6, 0.1]
$$

$$
y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}
$$

$$
= \frac{1}{1 + e^{-(.5 \cdot 2 + .6 \cdot 3 + .1 \cdot 9 + .5)}} = \frac{1}{1 + e^{-0.87}} = .70
$$

^y ⁼ ^s(*w· ^x*+*b*) = ¹

 $W_{\text{max}} = W_{\text{max}}$

What would this unit do with the following input vector:

Non-linear function besides sigmoid The simulation is significant and perhaps the most commonly used, in the most commonly used, is the most conta ReLU tified linear unit, also called the ReLU, shown in Fig. 7.3b. It's just the same as *z*

© Jixing Li

ez ⁺*e^z* (7.5)

The XOR problem some very simple functions of its input of its input of task of task of computing elementary of computing elementary \mathbf{r}

Can neural units compute simple functions of input? logical functions of two inputs, like AND, or, and XOR. As a reminder, here are also and \sim the truth tables for the truth term in the truth that the truth the truth the truth that the truth the truth t
Those functions in the truth of the truth that the truth that the truth the truth the truth the truth the truth

Perceptrons w, including the complete of the complete of \mathbf{r} ⇢ ⁰*,* if *^w· ^x*+*^b* ⁰ the proof by Minsky and Papert (1969) that a single neural unit cannot compute *y* = the proof by Minsky and Papert (1969) that a single neural unit cannot compute

A very simple neural unit $1 \t0, if w \cdot x + b \leq 0$ Binary output (0 or 1) and 1 A very simple neural unit $\begin{array}{ccc} 0, & \text{if } w \cdot x + b \leq 0 \\ p \end{array}$ functions of its binary inputs; \mathbf{F} is binary weights. The necessary weights. The necessary weights. The necessary weights. The necessary weights is a set of \mathbf{F} is a set of \mathbf{F} is a set of \mathbf{F} is a se $\mathbf{D}\mathbf{I}$ the truth tables for the truth tables functions: $\mathbf{D}\mathbf{I}$ $f(x) = \frac{1}{2} \int_{0}^{1} f(x) \, dx$

One of the most clever demonstrations of the need for multi-layer networks was

One of the most clever demonstrations of the need for multi-layer networks was

XOR with perceptrons? *x* and *x* and *b* and *x* and *b* and *y* and *b* and *y* an ⇢ ⁰*,* if *^w· ^x*+*^b* ⁰ *Perceptrons?* $\mathbf v$ and $\mathbf v$ ith more neural unit cannot compute some very simple functions of its input. Consider the task of computing elementary

$$
y = \begin{cases} 0, & \text{if } w \cdot x + b < 0 \\ 1, & \text{if } w \cdot \end{cases}
$$

Perceptrons are linear classifiers

Perceptron equation given x1 and x2, is the equation of a line

 $w_1x_1 + w_2x_2 + b = 0 \implies x_2 = (-w_1/w_2)x_1 + (-b/w_2)$

This line acts as a **decision boundary**

Solution to the XOR problem \overline{a} R prob = *.*70

some very simple functions of its input. Consider the task of computing elementary

A layered network of unit:

perceptron internal was first shown for the perceptron, which is a very simple neural shown for the perceptron, which is a very simple neural simple neural simple neural simple neural simple neural simple neural simple neu $y = \max(\lambda, 0)$ Activation function: ReLU $y = max(z, 0)$

The hidden representation h

A layered network of unit:

a) The original *x* space

b) The new (linearly separable) *h* space

hidden layers learn to form useful representations

The hidden representation h

A layered network of unit:

a) The original *x* space

b) The new (linearly separable) *h* space

hidden layers learn to form useful representations

Feedforward neural networks

Two-layer network with scalar output

Feedforward neural networks

Two-layer network with softmax output

The softmax function Like the sigmoid, it is an exponential function. For a vector *z* of dimensionality *k*, the softmax is defined as: **The softmax function** distribution, with each value in the range (0,1), and all the values summing to 1. Like the sigmoid, it is an exponential function.

the result softmax(*z*) is

distribution, with each value in the range (0,1), and all the values summing to 1.

Turns a vector $z = [z_1, z_2, ..., z_k]$ of k arbitrary values into probabilities P*^k ^j*=¹ exp(*zj*) **i** rbitrary values into ctor $z = [z_1, z_2, ...]$ Z_l *, ez*2 *,..., ezk* $+$ For a vector $z = [z_1, z_2, ..., z_k]$ of *k* arbitrary values is the software some software as

takes a vector *z* = [*z*1*,z*2*,...,zk*] of *k* arbitrary values and maps them to a probability

$$
\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^k \exp(z_j)} \quad 1 \le i \le k
$$
\n
$$
\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^k \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^k \exp(z_i)}, \dots, \frac{\exp(z_k)}{\sum_{i=1}^k \exp(z_i)} \right]
$$

$$
z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]
$$

softmax(z) = [0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]

© Jixing Li $@$ Jixing Li

P*^k*

Example

chinese_name major gender n1_male n2_male n1_uniqueness n2_uniqueness 林加敏 F 0.442 **LLA** -0.562 2.795 2.087

$$
x = [0.442, -0.562, 2.795, 2.087]
$$

\n
$$
W = [[1,1,1,1],
$$

\n
$$
[1,1,1,1]]
$$

\n
$$
b1 = [1,1], b2 = 1
$$

\n
$$
U = [1,1]
$$

\n
$$
b2 = Uh + b2
$$

\n
$$
V = \sigma(z)
$$

\n
$$
V = b2
$$

\n
$$
V = Uh + b2
$$

To do

• Optional reading: **SLP** Ch7.1-7.3