Deep Learning

3 MLP: Representation Power of an MLP

Dr. Konda Reddy Mopuri
Dept. of Artificial Intelligence
IIT Hyderabad
Jan-May 2023

So far

1. Any Boolean function of n inputs can be exactly represented with one hidden layer!

Universal Approximation (for real functions)

1. We can represent any continuous function $(f : \mathcal{R}^m \rightarrow \mathcal{R}^n)$ to any desired approximation $(|g(x) - f(x)| < \epsilon)$ with a linear combination of sigmoid neurons

Universal Approximation (for real functions)

1. We can represent any continuous function $(f : \mathcal{R}^m \rightarrow \mathcal{R}^n)$ to any desired approximation $(|g(x) - f(x)| < \epsilon)$ with a linear combination of sigmoid neurons
2. In other words, neural networks with a single hidden layer can be used to approximate any continuous function to any desired precision

Universal Approximation

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Neural Networks, Vol. 4, pp. 251–257, 1991

0893-6080/91 $3.00 + .00

ORIGINAL CONTRIBUTION

Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

Technische Universität Wien, Vienna, Austria

Universal Approximation

1. Let's look at the visual proof!

Universality with 1-input and 1-output

- Two hidden units and one output unit

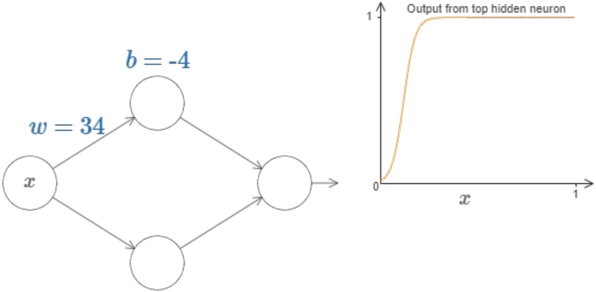


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Sigmoid neurons can closely approximate a step function!

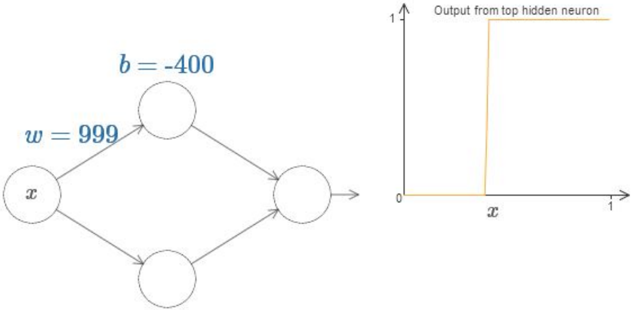


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Let's simplify the neuron representation with a single parameter (s)

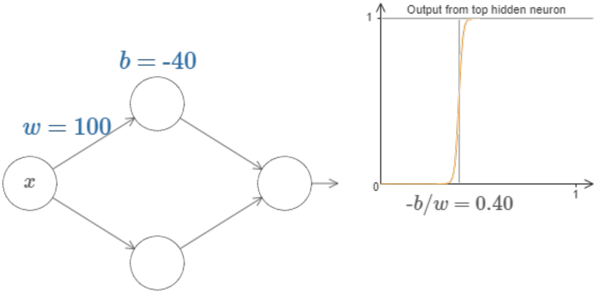


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Let's simplify the neuron representation with a single parameter (s)

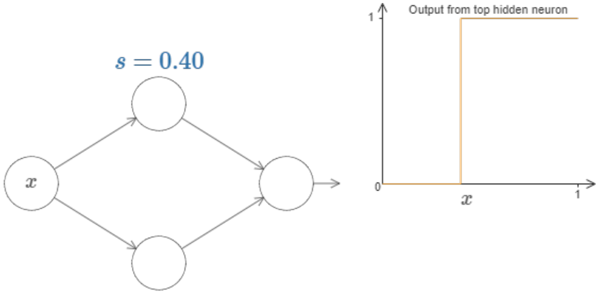
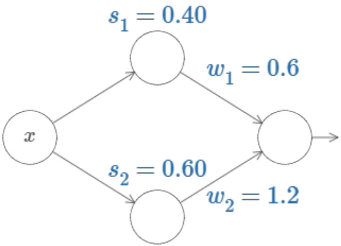


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Weighted output of hidden neurons



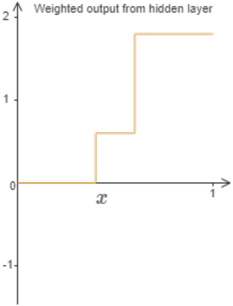


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Can output a pulse/tower of desired width and height!

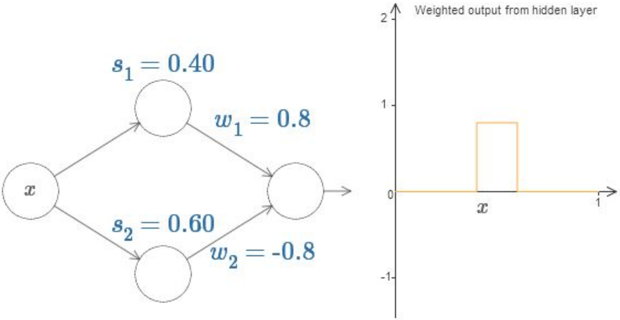


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Can output a pulse/tower of desired width and height!

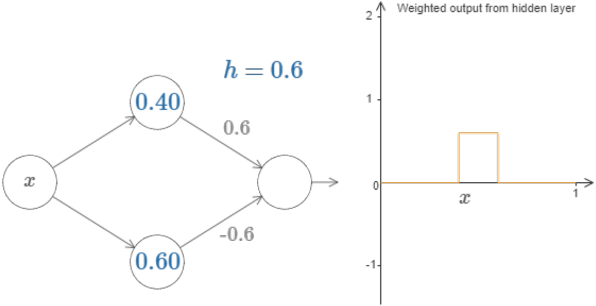


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Can output a pulse/tower of desired width and height!

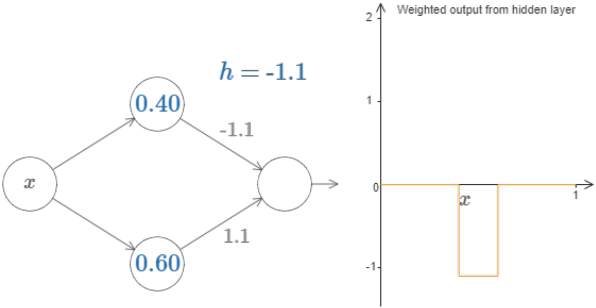


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- With more neurons in the hidden layer, more towers!

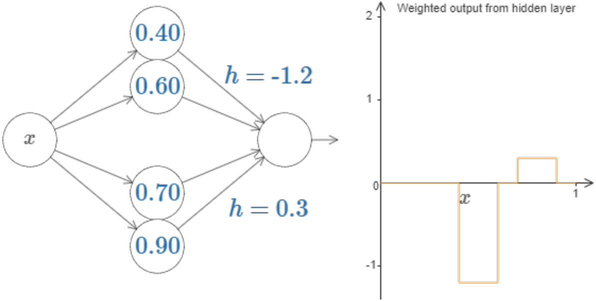


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- With more neurons in the hidden layer, more towers!

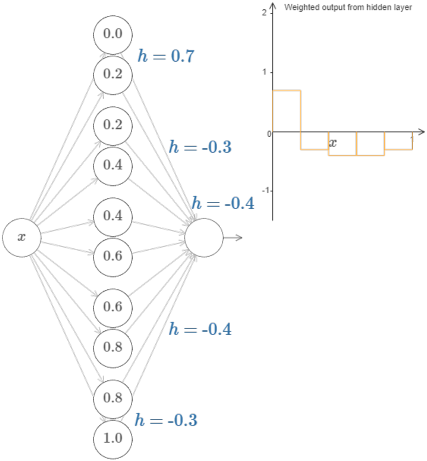


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

1. Note that we computed only the weighted sum of the hidden outputs

Universality with 1-input and 1-output

1. Note that we computed only the weighted sum of the hidden outputs
2. It's not the output of our MLP

Universality with 1-input and 1-output

- For approximating $f(x)$, the input to the output neuron has to be $\sigma^{-1}(f(x))$ (note that the bias is zero)

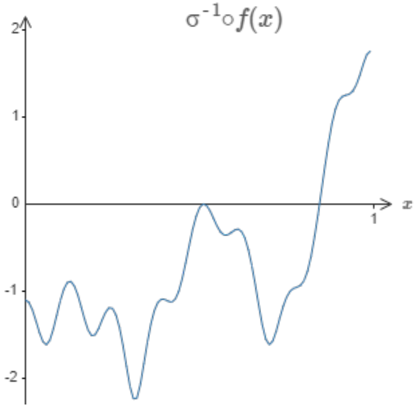


Figure from Michael Nielsen's NNDL textbook

Universality with 1-input and 1-output

- Manipulating the width and height of the towers $\rightarrow$ a better approximation of the function

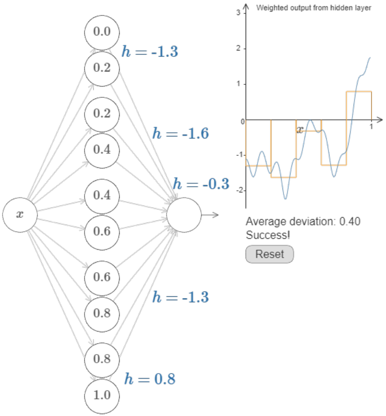


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Let's consider two input variables

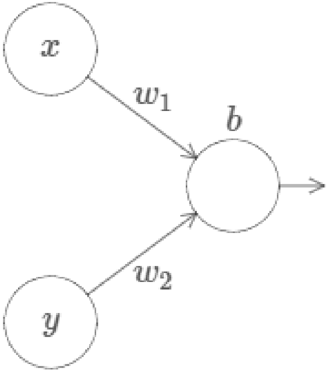


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Let's set $w_2 = 0$

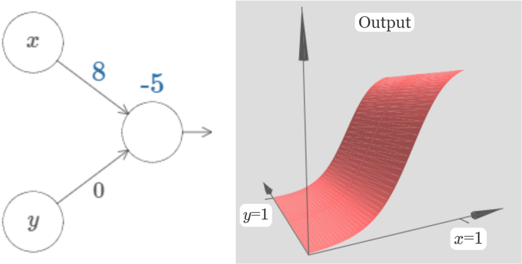


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- As seen earlier, let's approximate the step function

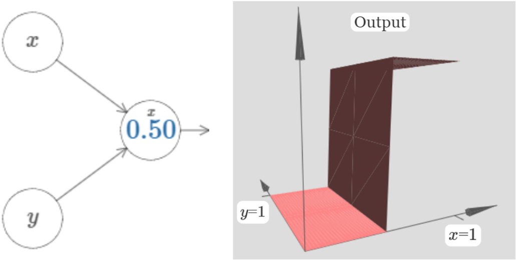


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- As seen earlier, let's approximate the step function
- Use a single parameter $s = -b/w$ to represent

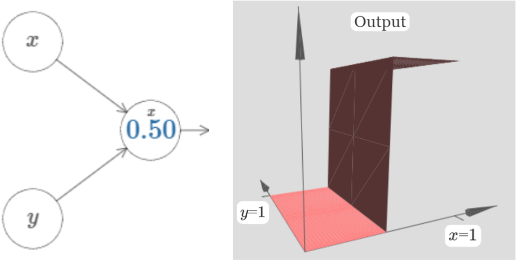


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- The step function in the y direction

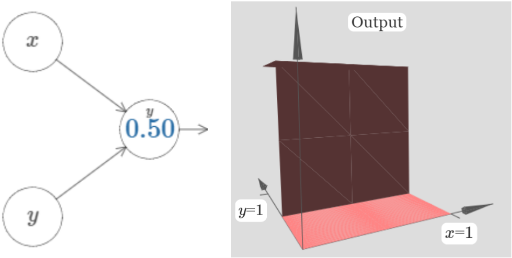


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Towards the tower in 3D

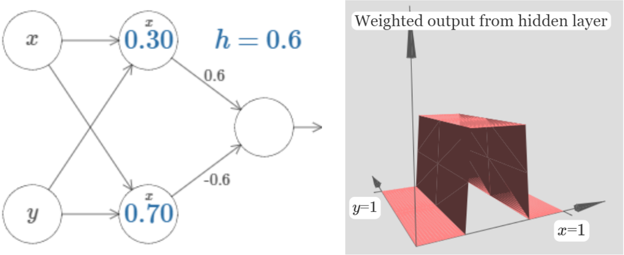


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Towards the tower in 3D

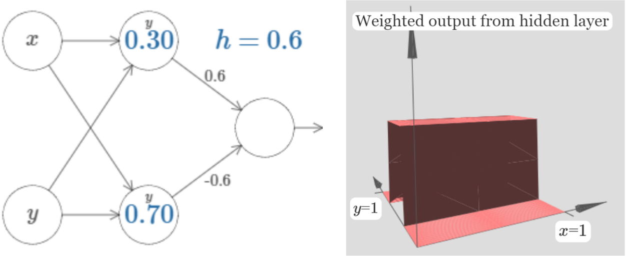


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Towards the tower in 3D

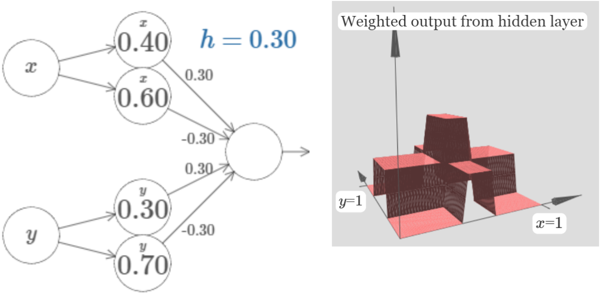


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Towards the tower in 3D

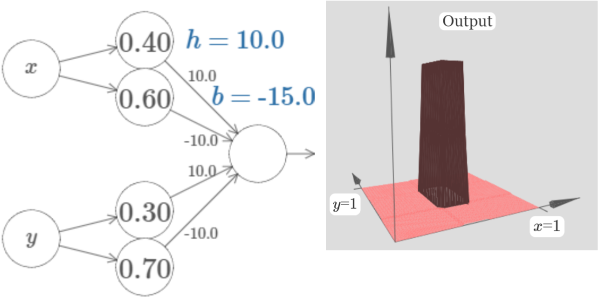


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Towards the tower in 3D

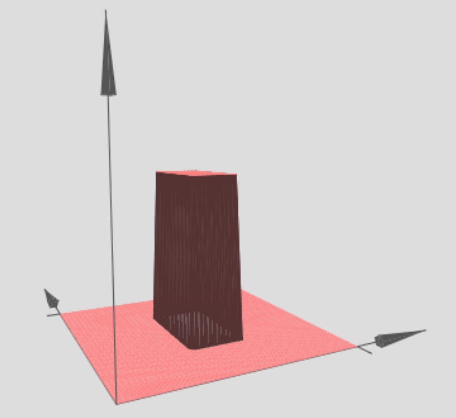


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

- Several of the towers can approximate arbitrary functions

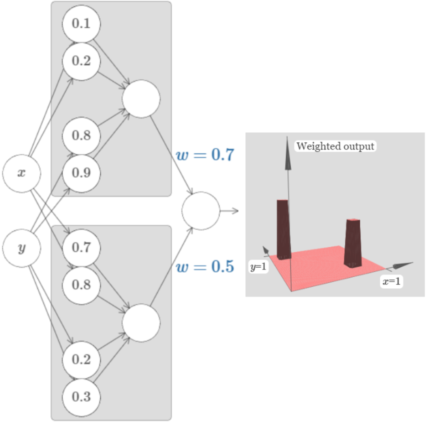


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Several of the towers can approximate arbitrary functions

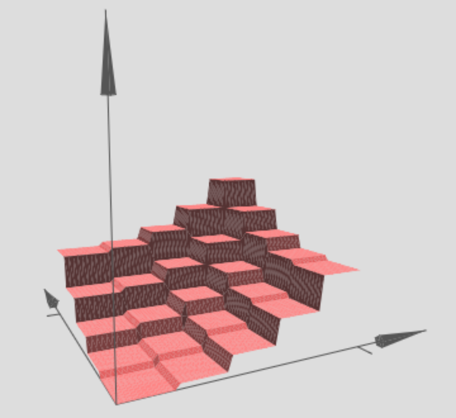


Figure from Michael Nielsen's NNDL textbook

Universality with multiple inputs

- Three input variables

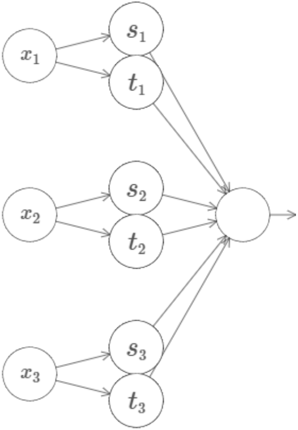


Figure from Michael Nielsen's NNDL textbook

Universality for vector-valued functions

- $f(x) : \mathcal{R}^m \rightarrow \mathcal{R}^n$

Universality for vector-valued functions

- $f(x) : \mathcal{R}^m \rightarrow \mathcal{R}^n$
- Can be regarded as n separate real-valued functions $f^1(x_1, \ldots, x_m), \ldots, f^n(x_1, \ldots, x_m)$

Universality for vector-valued functions

- $f(x) : \mathcal{R}^m \rightarrow \mathcal{R}^n$
- Can be regarded as n separate real-valued functions $f^1(x_1, \ldots, x_m), \ldots, f^n(x_1, \ldots, x_m)$
- Create a network approximating each function f^i and put them all together

Universal Approximation: Original form

Theorem 0.1 (UAT, [Cyb89, Hor91]). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:*

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} v_i \sigma\left(\boldsymbol{w}_i^T \boldsymbol{x} + b_i\right) = \boldsymbol{v}^\mathsf{T} \sigma\left(\boldsymbol{W}^\mathsf{T} \boldsymbol{x} + \boldsymbol{b}\right)$$

as an approximate realization of the function f; that is,

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon$$

for all $\boldsymbol{x} \in I_m$.

Universal Approximation: Later

- Target function may lie on a space other than the hypercube (has to be bounded)

Universal Approximation: Later

- Target function may lie on a space other than the hypercube (has to be bounded)
- Discontinuous targets can be approximated arbitrarily well

Universal Approximation: Later

- Target function may lie on a space other than the hypercube (has to be bounded)
- Discontinuous targets can be approximated arbitrarily well
- σ can be as general as any nonpolynomial function ($\sigma(z)$ well-defined and different $z \to \infty$ and $z \to -\infty$; at least one side bounded)

Universal Approximation

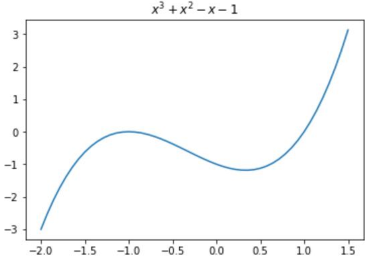
- Note that our visual proof had a network with two hidden layers

Universal Approximation

- Note that our visual proof had a network with two hidden layers
- One can show that a single hidden layer can do this

Universal Approximation using ReLU functions

1. Let's approximate the following function using a bunch of ReLUs:

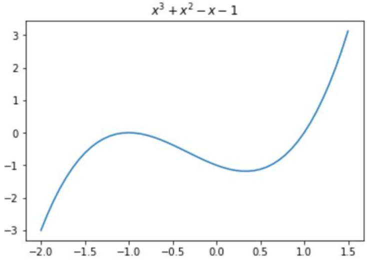


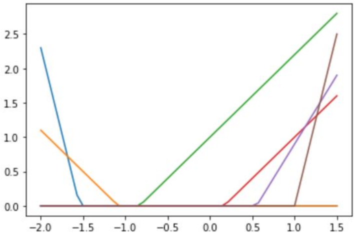
Example credits: Brendan Fortuner, and https://towardsdatascience.com/

Universal Approximation using ReLU functions

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. $n_1 = ReLU(-5x-7.7), n_2 = ReLU(-1.2x-1.3), n_3 = ReLU(1.2x+1), n_4 = ReLU(1.2x-0.2), n_5 = ReLU(2x-1.1), n_6 = ReLU(5x-5)$

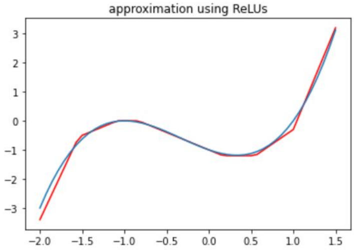




Example credits: Brendan Fortuner, and https://towardsdatascience.com/

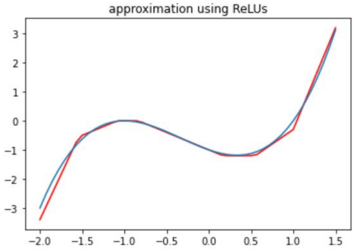
Universal Approximation using ReLU functions

1. Appropriate combination of these ReLUs:
$-n_1 - n_2 - n_3 + n_4 + n_5 + n_6$



Universal Approximation using ReLU functions

1. Appropriate combination of these ReLUs: $-n_1 - n_2 - n_3 + n_4 + n_5 + n_6$
2. Note that this also holds in case of other activation functions with mild assumptions.



If one hidden layer is good enough, why Deep learning?

1. May require an infeasible size for the hidden layer

If one hidden layer is good enough, why Deep learning?

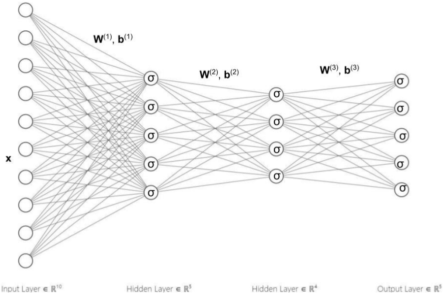
1. May require an infeasible size for the hidden layer
2. May not generalize well

If one hidden layer is good enough, why Deep learning?

1. May require an infeasible size for the hidden layer
2. May not generalize well
3. Doesn't enable the hierarchical learning

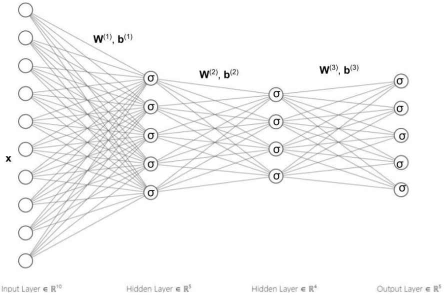
MLP for regression

1. Output is a continuous variable in $\mathcal{R}^D$
   - Output layer has that many neurons (When $D = 1$, regresses a scalar value)
   - May employ a squared error loss



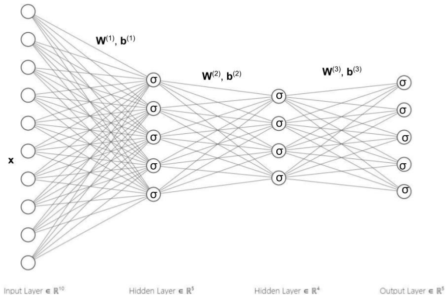
MLP for regression

1. Output is a continuous variable in $\mathcal{R}^D$
   - Output layer has that many neurons (When $D = 1$, regresses a scalar value)
   - May employ a squared error loss
2. Can have an arbitrary depth (number of layers)



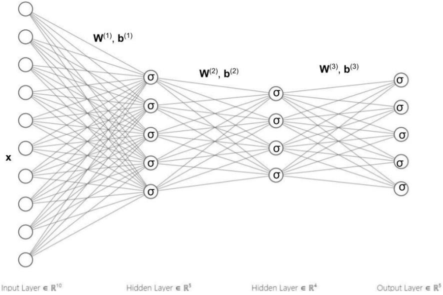
MLP for classification

1. Categorical output in $\mathcal{R}^C$ where C is the number of categories



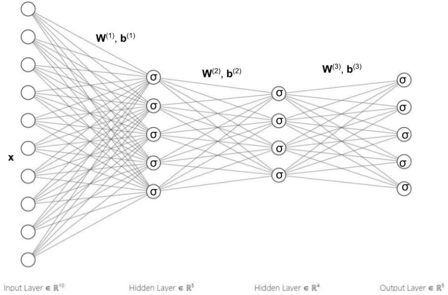
MLP for classification

1. Categorical output in $\mathcal{R}^C$ where C is the number of categories
2. Predicts the scores/confidences/probabilities towards each category
   - Then converts into a pmf
   - Employs loss that compares the probability distributions (e.g. cross-entropy)



MLP for classification

1. Categorical output in $\mathcal{R}^C$ where C is the number of categories
2. Predicts the scores/confidences/probabilities towards each category
   - Then converts into a pmf
   - Employs loss that compares the probability distributions (e.g. cross-entropy)
3. Can have an arbitrary depth



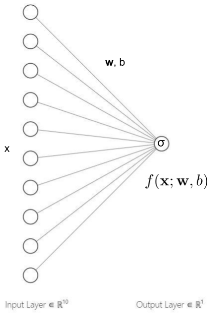
Extending Linear Classifier

1. Single class: $f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$ from $\mathcal{R}^D \rightarrow \mathcal{R}$ where $\mathbf{w}$ and $\mathbf{x} \in \mathcal{R}^D$

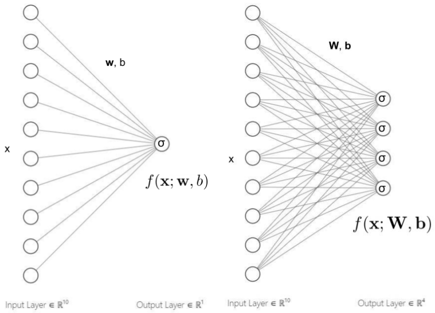
Extending Linear Classifier

1. Single class: $f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$ from $\mathcal{R}^D \rightarrow \mathcal{R}$ where $\mathbf{w}$ and $\mathbf{x} \in \mathcal{R}^D$
2. Multi-class: $f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ from $\mathcal{R}^D \rightarrow \mathcal{R}^C$ where $\mathbf{W} \in \mathcal{R}^{C \times D}$ and $\mathbf{b} \in \mathcal{R}^C$

Single unit to a layer of Perceptrons



Single unit to a layer of Perceptrons



Single unit to a layer of Perceptrons