Deep Learning

3 MLP: Representation power of a network of Perceptrons

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

Universal Approximation (for Boolean functions)

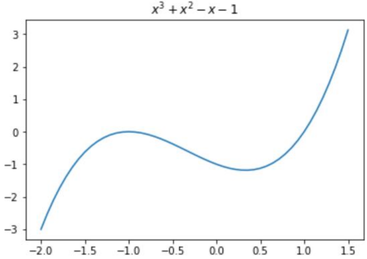
1. We can represent any Boolean function with a linear combination of perceptrons (an MLP with a single hidden layer)

Universal Approximation (for real functions)

1. We can represent any continuous function to any desired approximation with a linear combination of sigmoid neurons

Universal Approximation using ReLU functions

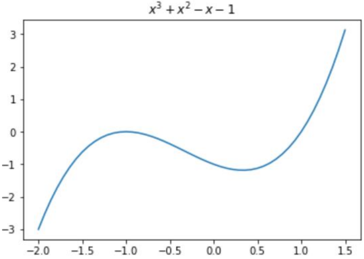
1. Let's approximate the following function using a bunch of ReLUs:

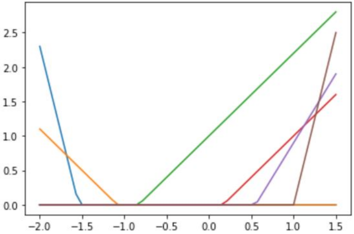


Example credits: Brendan Fortuner, and https://towardsdatascience.com/

Universal Approximation using ReLU functions

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

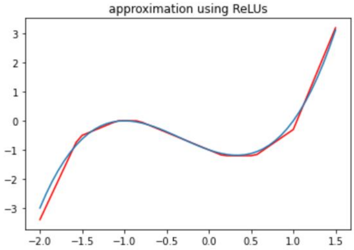




Example credits: Brendan Fortuner, and https://towardsdatascience.com/

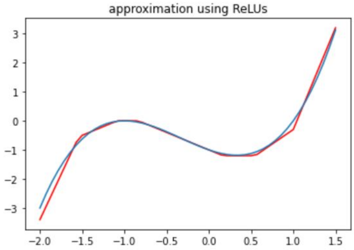
Universal Approximation using ReLU functions

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.



Universal Approximation Theorem

1. We can approximate any continuous function $\psi : \mathcal{R}^D \rightarrow R$ with one hidden layer of perceptrons

Cybenko (1989), Hornik (1991)

Universal Approximation Theorem

1. We can approximate any continuous function $\psi : \mathcal{R}^D \rightarrow R$ with one hidden layer of perceptrons
2. $\mathbf{x} \rightarrow \mathbf{w}^T \sigma(W\mathbf{x} + \mathbf{b})$
   $\mathbf{b} \in \mathcal{R}^C, W \in \mathcal{R}^{C \times D}, \mathbf{w} \in \mathcal{R}^C,$ and $\mathbf{x} \in \mathcal{R}^D$

Cybenko (1989), Hornik (1991)

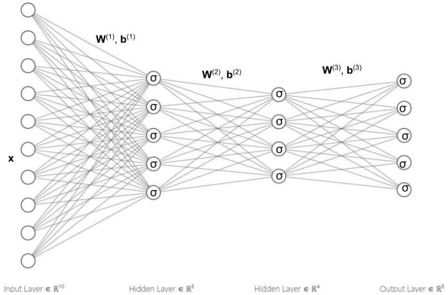
Universal Approximation Theorem

1. We can approximate any continuous function $\psi : \mathcal{R}^D \rightarrow R$ with one hidden layer of perceptrons
2. $\mathbf{x} \rightarrow \mathbf{w}^T \sigma(W\mathbf{x} + \mathbf{b})$
   $\mathbf{b} \in \mathcal{R}^C, W \in \mathcal{R}^{C \times D}, \mathbf{w} \in \mathcal{R}^C,$ and $\mathbf{x} \in \mathcal{R}^D$
3. However, the resulting NN
   - May require infeasible size for the hidden layer
   - May not generalize well

Cybenko (1989), Hornik (1991)

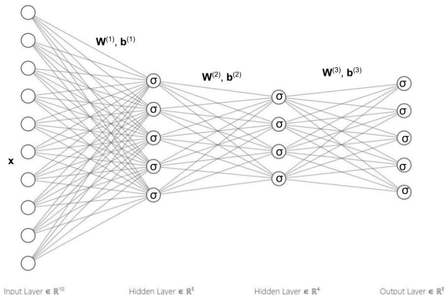
MLP for regression

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



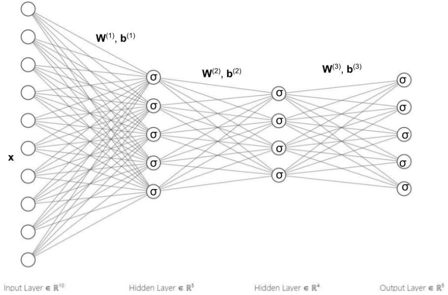
MLP for regression

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)



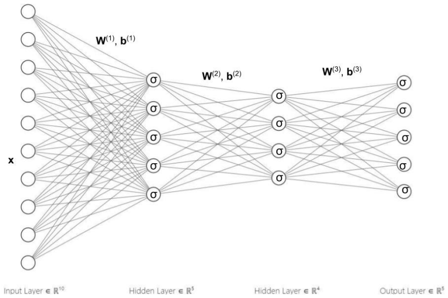
MLP for classification

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



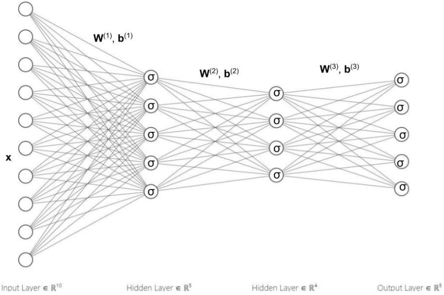
MLP for classification

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



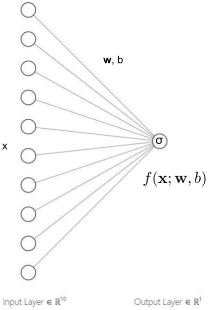
Extending Linear Classifier

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

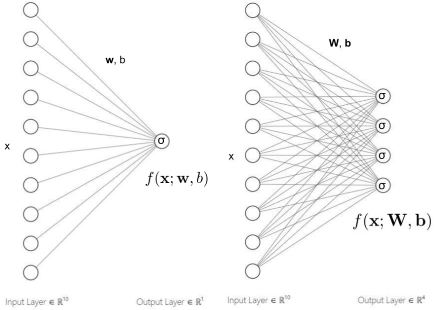
Extending Linear Classifier

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

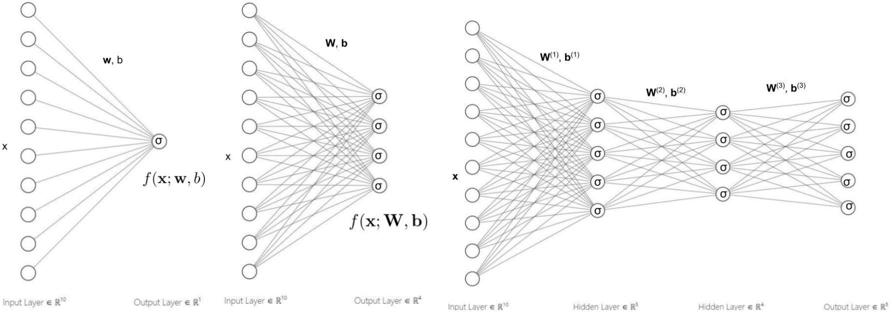
Single unit to a layer of Perceptrons



Single unit to a layer of Perceptrons



Single unit to a layer of Perceptrons



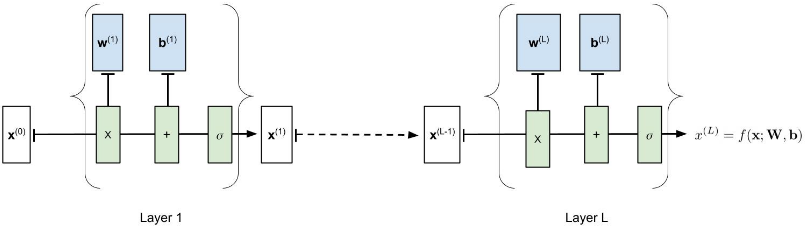
Formal Representation

1. Latter is known as an MLP: Multi-Layered Perceptron (i.e, Multi-Layered network of Perceptrons)

Formal Representation

1. Latter is known as an MLP: Multi-Layered Perceptron (i.e, Multi-Layered network of Perceptrons)
2. can be represented as:
   $\mathbf{x}^{(0)} = \mathbf{x}$,
   $\forall l = 1, \ldots, L, \quad \mathbf{x}^{(l)} = \sigma(\mathbf{W}^{(l)^T} \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)})$, and

MLP



$\mathbf{w}^{(1)}$
$\mathbf{b}^{(1)}$
$\mathbf{x}^{(0)}$
X
+
σ
$\mathbf{x}^{(1)}$
$\mathbf{x}^{(L-1)}$
$\mathbf{w}^{(L)}$
$\mathbf{b}^{(L)}$
X
+
σ
$x^{(L)} = f(\mathbf{x}; \mathbf{W}, \mathbf{b})$
Layer 1
Layer L

Nonlinear Activation

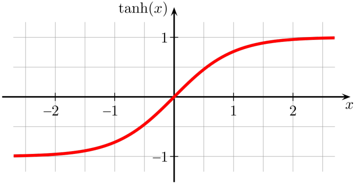
1. Note that σ is nonlinear

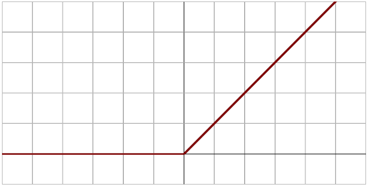
Nonlinear Activation

1. Note that σ is nonlinear
2. If it is an affine function, the full MLP becomes a complex affine transformation (composition of a series of affine mappings)

Nonlinear Activation

Familiar activation functions





Hyperbolic Tangent (Tanh) $x \rightarrow \frac{2}{1+e^{-2x}} - 1$ and
Rectified Linear Unit (ReLU) $x \rightarrow \max(0, x)$ respectively