

## List of Topics

### 01 Basics of Neural Network

- Basic functional form
- Universal function approximator

### 02 Technical difficulties (and solutions)

- Vanishing gradient problem
- Vanishing variance problem
- Ill-posed problem and overfitting

### 03 programming exercise 01

- Pseudo-code
- Framework comparison
- PyTorch hands-on

### 04 Exploring data properties and structure

- Recurrent neural network (RNN)
- vanilla, LSTM and variants
- Convolutional neural network (CNN)
- Notable architectures (AlexNet, VGGNet, GoogLeNet, ResNet, Xception)

### 05-programming exercise 02

- RNN and CNN implementation

### 06 NeuralNet zoo

- Google AutoML, Caffe model zoo
- Neural style transfer

### 07 Advanced usage 01 (into the deep layers)

- Transfer learning
- AutoEncoder
- U-net

### 08 Advanced usage 02 (generative models)

- Generative Adversarial network
- Variational autoencoder (VAE), conditional VAE
- variational inference

### 09 programming exercise 03

- implementation templates for lecture 07 and 08

### 10 Research trend 01: AlphaZero

- Monte Carlo Tree Search, reinforcement learning

### 11 Research trend 02: Deep Query Network (DQN)

- Algorithm cocktail (convLSTM, net-merging etc)

### 12-Research trend 03: NEAT

- Dynamic neural network
- model averaging

### 13-Final remark and tips

- "brainless ML" / step-by-step guide
- Research trend
- Unresolved topics / problems

## Machine learning

### Formal Definition:

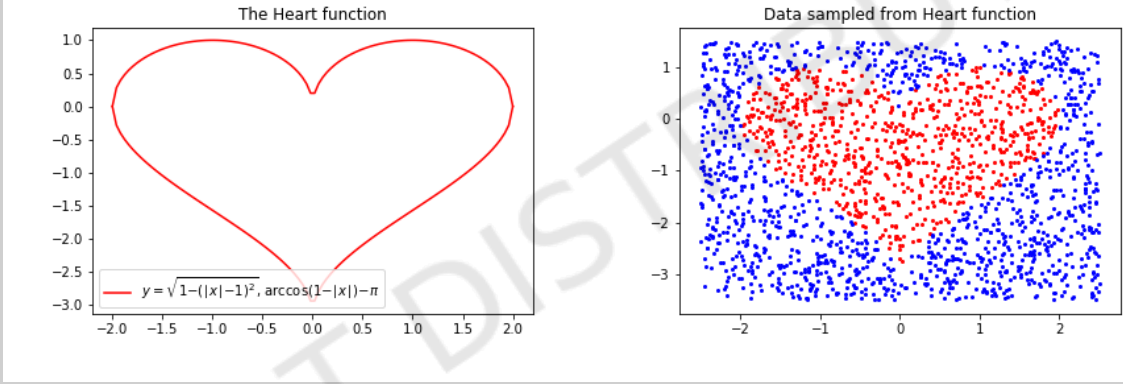
"A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ." - Mitchell, T. (1997).

### Loose Definition:

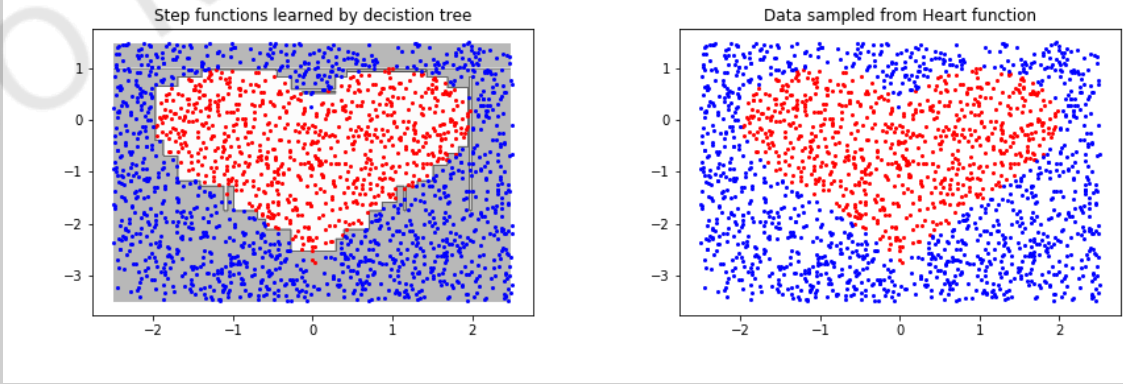
"Approximating the underlying function to a target task by synthesis of basis functions of simple form."

Classifier	basis function
Decision trees	Step function
Nearest neighbours	Delta function
Naive Bayes, mixture models	Gaussian
Neural networks	Logistic function

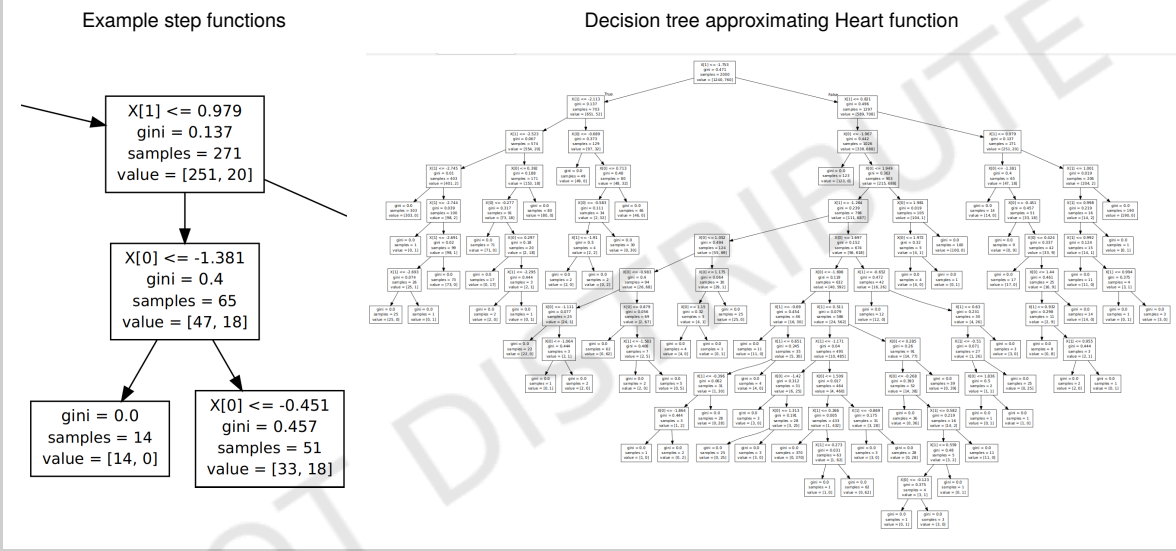
Function Approximation demonstration



Function Approximation demonstration



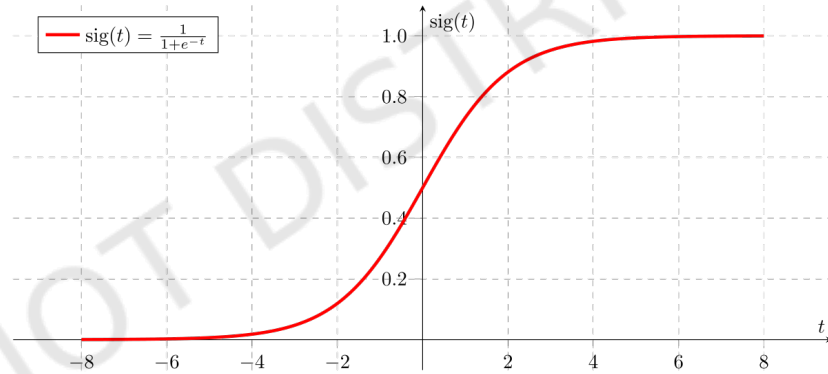
Function Approximation demonstration



## Logistic Regression

$$y = (1 + \exp(-\{\sum_{\forall i} w_i x_i + b_i\}))^{-1}$$

**Objective:** Find optimal  $\vec{w}_i, \vec{b}_i$  over training set  $(\vec{X}, \vec{Y})$



## Stacking Logistic Regression

- 2-layer:

$$y = (1 + \exp(-\{\sum_{\forall j} w_j \boxed{(1 + \exp(-\{\sum_{\forall i} w_i x_i + b_i\}))^{-1}} + b_j\}))^{-1}$$

**Objective:** Find optimal  $\vec{w}_i, \vec{w}_j, \vec{b}_i, \vec{b}_j$  over training set  $(\vec{X}, \vec{Y})$

**This is already a neural network !**

## Keep Stacking

- many-layer:

$$y = (1 + \dots \boxed{(1 + \exp(-\{\sum_{\forall j} w_j \boxed{(1 + \exp(-\{\sum_{\forall i} w_i x_i + b_i\}))^{-1}} + b_j\}))^{-1} \dots)^{-1}$$

**Objective:** Find optimal  $\vec{w}_i, \vec{w}_j, \dots, \vec{b}_i, \vec{b}_j, \dots$  over training set  $(\vec{X}, \vec{Y})$

**This is a \*deep\* neural network**

# Universal function approximator

"... the universal approximation theorem states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of  $R^n$ , under mild assumptions on the activation function."

~ Cybenko, G. 1989; Balázs Csanád Csáji 2001

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

**Mathematics of Control,  
Signals, and Systems**  
© 1989 Springer-Verlag New York Inc.

Neural Networks, Vol. 4, pp. 251–257, 1991  
Printed in the USA. All rights reserved.

0893-4000/91 \$3.00 + .00  
Copyright © 1991 Pergamon Press plc

ORIGINAL CONTRIBUTION

**Approximation Capabilities of Multilayer  
Feedforward Networks**

KURT HORNIK  
Technische Universität Wien, Vienna, Austria  
(Received 30 January 1990; revised and accepted 25 October 1990)

**Abstract**—We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to  $L^p(\mu)$  performance criteria, for arbitrary finite input environment measures  $\mu$ , provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.

**Keywords**—Multilayer feedforward networks, Activation function, Universal approximation capabilities, Input environment measure,  $L^p(\mu)$  approximation, Uniform approximation, Sobolev spaces, Smooth approximation.

**Approximation by Superpositions of a Sigmoidal Function\***

G. Cybenko†

**Abstract**. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of  $n$  real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

**Key words**. Neural networks, Approximation, Completeness.

## Discussion:

why not just use universal function approximator all the times?