Introduction to Program Synthesis (SS 25)

Chapter 4 - Advanced Methodologies

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany

Advanced Methodologies

Neural Program Synthesis vs Genetic Programming

- **Genetic Programming**
  - $\rightsquigarrow$ Gradient-free symbolic search heuristic
  - $\rightsquigarrow$ Direct search in symbolical spaces
  - $\rightsquigarrow$ Ill-conditioned, noisy search spaces
  - $\rightsquigarrow$ Interpretable symbolic solutions
- **Neural Program Synthesis**
  - $\rightsquigarrow$ Gradient-based search in the latent space of a neural network
  - $\rightsquigarrow$ Numerical, non-interpretable solutions
  - $\rightsquigarrow$ Turing complete programs?

- **Artificial Neural Network (ANN)** $\rightarrow$ model of computation inspired by biological brain topology and functioning
- Set of *artificial neurons* called **perceptron** and connections that mimic synapses
  - $\rightsquigarrow$ Perceptrons $\rightarrow$ **nodes**
  - $\rightsquigarrow$ Synapses $\rightarrow$ **edges**, **weights**
- ANN $\rightarrow$ directed, weighted and acyclic graph
- Perceptron $\rightarrow$ equipped with an activation function
  - $\rightsquigarrow$ Mimics firing of a neuron

Weights	—	Modelling of weighted transitions
Bias	—	Shifting of transitions
Backpropagation	—	Derivation-based update of parameters
Perceptron	—	Artificial neuron
Layer	—	Set of perceptrons
Transition function	—	Calculation of the neuron's output
Activation function	—	Decision on forward propagation (of the output)
Batch	—	Set of input samples
Epoch	—	Iteration of the training phase

Table: List of important terms which are commonly used in the field of deep learning

Advanced Methodologies
Deep Learning

φ	—	activation function
η	—	learning rate
β	—	batch size

Table: List of general symbols

Advanced Methodologies
Deep Learning

- 1943 McCulloch, Pitts: **Electronic Brain**
- 1950 Turing: **Learning Machine**
- 1958 Rosenblatt: **Perceptron**
- 1969 Minsky, Papert: **XOR Problem**
- 1982 Hopfield: **Recurrent Neural Network**
- 1986 Rumelhart, Hinton, Williams: **Multi-layered Perceptron**, **Backpropagation**
- 1989 LeCun: **Convolutional Neural Networks**
- 1997 Hochreiter, Schmidhuber: **Long Short Term Memory (LSTM)**
- 2006 Hinton, Rusian: **Deep Neural Networks**
- 2012 Ng, Dean: **Recognizing Cats on YouTube**
- 2017 Ashish et al.: **Transformer**

Advanced Methodologies
Deep Learning

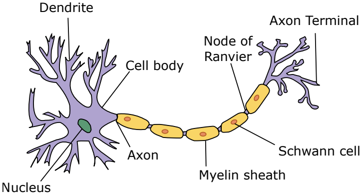


Figure: Biological neuron

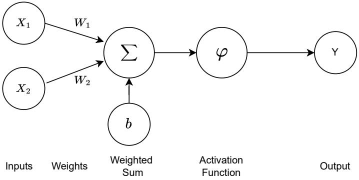
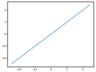
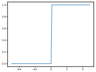
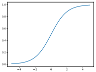
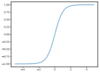
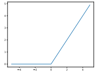


Figure: Artificial neuron (perceptron)

Definition (Perceptron)

A perceptron $\mathcal{P}$ is a composite function g of the transition function Σ and activation function φ that receives a signal $s = (s_1, ..., s_m)$ which is weighted by $w = (w_1, ..., w_m)$. The function $g(\sigma \circ \varphi)$ generates a one dimensional output y.

Identity	x		$(-\infty, \infty)$
Binary step	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$		$\{0, 1\}$
Logistic (sigmoid)	$\sigma(x) \doteq \frac{1}{1+e^{-x}}$		$(0, 1)$
Hyperbolic tangent (tanh)	$\tanh(x) \doteq \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$		$(-1, 1)$
Rectified linear unit (ReLU)	$x^{+} = \max(0, x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$		$[0, \infty)$

- **Linear activation functions**
  - $\rightsquigarrow$ Can only represent linear relationships $\rightarrow$ linear separation of data
  - $\rightsquigarrow$ Data from real world problems $\rightarrow$ often noisy and correlations are mostly non-linear
  - $\rightsquigarrow$ Used only at the output layer
- **Non-linear activation functions**
  - $\rightsquigarrow$ Enable the network to approximate a wider range of functions

- General topology of ANN's $\rightarrow$ input, hidden and output layers
- **Single layer neural networks** $\rightarrow$ one hidden layer
  - $\rightsquigarrow$ Traditional neural learning paradigm
- **Multilayer neural networks** $\rightarrow$ multiple hidden layers
  - $\rightsquigarrow$ Modern deep learning approach
- Hidden layer(s) $\rightarrow$ sets of artificial neurons
- Each artificial neuron k has a transition and activation function
  - $\rightsquigarrow$ $y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$
  - $\rightsquigarrow$ Transition function $\rightarrow$ typically sum of the input parameters

- General deep learning paradigm
  - $\rightsquigarrow$ Given a set of n samples $\mathbf{D} = \{[\mathbf{x_1}, ...\mathbf{x_n}], [\mathbf{y_1}, ..., \mathbf{y_n}]\}$, find a model that approximates a function $\mathbf{f}(\mathbf{x_i}) = \mathbf{y_i}$
  - $\rightsquigarrow$ **Forward propagation:** Model is feed with a sample data set D where the matrix $[x_1, ...x_n]$ is the input and $[\hat{y}_1, ..., \hat{y}_n]$ the output matrix
  - $\rightsquigarrow$ For each sample $(x_i, y_i) \in D$, the distance of the prediction to the actual is measured by the loss function $\mathcal{L}$
  - $\rightsquigarrow$ Parameters are adjusted in accordance to the error $\rightarrow$ **Backpropagation** via partial derivatives of the loss function
  - $\rightsquigarrow$ Efficient use of the chain rule $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$

- **Feed-forward neural network** $\rightarrow$ Commonly used type of ANN
  - $\leadsto$ Two modes with different directions of the computation flow
  - $\leadsto$ Multilayer perceptron model $\rightarrow$ **fully connected** (feed-forward) network with at least three layers
- **Forward pass:** Computation of predictions based on given observations or features
  - $\leadsto$ Forward propagation of the input data
  - $\leadsto$ Calculation of the loss $\mathcal{L}$ and cost $\mathcal{C}$
  - $\leadsto$ Quantification of the error $\mathcal{E}$ based on $\mathcal{L}$ and $\mathcal{C}$
- **Backward pass:** Backpropagation of the error
  - $\leadsto$ Derivation-based method
  - $\leadsto$ Adjustment of the parameters (weights and biases) in accordance to a learning rate η
  - $\leadsto$ Minimization of $\mathcal{E}$

Advanced Methodologies
Deep Learning

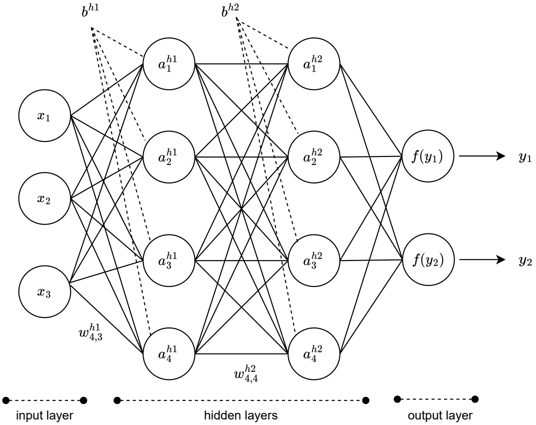


Figure: Feed-forward neural network with two hidden layers

- **Notation:**
  - $\mathbf{x_i}$ is the i-th feature of the input batch
  - $\mathbf{\hat{y}_i}$ is the i-th predicted output
  - $\mathbf{y_i}$ is the i-th desired output
  - $\mathbf{w_{j,i}^t}$ the weight of the connection between j-th node in layer t and i-th node in layer $t-1$
  - $\mathbf{b^t}$ is the bias for the layer t
    - Global for the respective hidden layer
  - $\mathbf{z^t}$ is the intermediate net-input for layer t
  - $\mathbf{a_i^t}$ is the result of the activation function of the i-th node in layer t

Advanced Methodologies

Deep Learning: Representation

- **Configuration:**
  - $\rightsquigarrow$ n: number of features (inputs)
  - $\rightsquigarrow$ k^t: number of nodes of layer t
  - $\rightsquigarrow$ l: number of hidden layers
  - $\rightsquigarrow$ o: number of outputs
- **Representation:**
  - $\rightsquigarrow$ Weights matrix $\rightarrow$ $\mathbf{W}$
  - $\rightsquigarrow$ Activation vectors $\rightarrow \mathbf{a^t} \in \mathbb{R}^{k^t}$
  - $\rightsquigarrow$ Biases $\rightarrow \mathbf{b} \in \mathbb{R}^l$
- **Dimensions:**
  - $\rightsquigarrow$ $\dim(W^{t=1}) = k^1 \times n$ (first hidden layer)
  - $\rightsquigarrow$ $\dim(W^{t>1}) = k^t \times k^t$
  - $\rightsquigarrow$ $\dim(W^o) = o \times k$ (output layer)

$$W^{h>1} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & & \vdots \\ w_{k,1} & w_{k,2} & \cdots & w_{k,k} \end{pmatrix}$$

- Intuition of backpropagation $\rightarrow$ analytical gradient
  - $\rightsquigarrow$ **Intermediate variables** $\rightarrow$ calculated by forward propagation
  - $\rightsquigarrow$ **Intermediate gradients** $\rightarrow$ used for backpropagation
- Chain rule $\rightarrow$ differentiation of a composite function
  - $\rightsquigarrow$ $h = f \circ g \rightarrow h'(x) = f'(g(x))g'(x)$
  - $\rightsquigarrow$ Analysis how the change of the rate of a composite function is affected

Advanced Methodologies
Deep Learning

- $\varphi = \sigma(x) = \frac{1}{1+e^{-x}}$
  - $\rightsquigarrow$ $\varphi' = \sigma'(x) = \sigma(x) * (1 - \sigma(x))$
- Intermediate variables (forward pass)
  - $\rightsquigarrow$ $\hat{y} = \varphi(x^T * W_1 + b_1) * (W_2 + b_2)$
  - $\rightsquigarrow$ $h_1 = x^T * W_1 + b_1$
  - $\rightsquigarrow$ $z_1 = \varphi(h_1)$
  - $\rightsquigarrow$ $z_2 = z_1 * W_2 + b_2$
  - $\rightsquigarrow$ $\mathcal{L} = (z_2 - y)^2$
- Intermediate gradients (backward pass)
  - $\frac{\partial \mathcal{L}}{\partial z_2} = 2 * (z_2 - y)$
  - $\rightsquigarrow$ $\frac{\partial z_2}{\partial z_1} = W_2^T$
  - $\rightsquigarrow$ $\frac{\partial z_1}{\partial h_1} = \varphi'(h_1) = z_1 * (1 - z_1)$
  - $\rightsquigarrow$ $\frac{\partial h_1}{\partial x} = W_1^T$

Advanced Methodologies

Deep Learning: MLP example

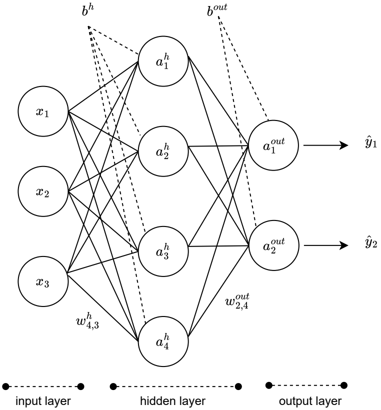


Figure: Feed-forward neural network with $n = 3$, $l = 1$, $k = 4$ and $o = 2$

Advanced Methodologies

Deep Learning: MLP example

- Activation:
  - $\varphi = \sum_{j=0}^{m} w_{kj} x_j$
- Forward propagation:
  - $\rightsquigarrow$ $z^t = W^t * z^{t-1} + b^t$
  - $\rightsquigarrow$ $a^t = \varphi(z^t)$
  - $\rightsquigarrow$ $\hat{y} = a^{out}$
- Backward propagation:
  - $\rightsquigarrow$ $\mathcal{L} = \frac{1}{\beta} \sum_{i=1}^{\beta} (\hat{y} - y)^2 \rightarrow$ Mean Squared Error (MSE)
  - $\rightsquigarrow$ $w_{i,j}^{new} = w_{i,j}^{old} - \eta \frac{\mathcal{L}}{\partial w_{i,j}}$
  - $\rightsquigarrow$ $b_t^{new} = b_t^{old} - \eta \frac{\mathcal{L}}{\partial b^t}$

Advanced Methodologies

Deep Learning: Backpropagation

- $\frac{\partial \mathcal{L}}{\partial w_{2,4}^{out}} = \frac{\partial \mathcal{L}}{\partial a_2^{out}} \frac{\partial a_2^{out}}{\partial w_{2,4}^{out}}$
- $\frac{\partial \mathcal{L}}{\partial w_{4,3}} = \left[\frac{\partial \mathcal{L}}{\partial a_1^{out}} \frac{a_1^{out}}{\partial a_4^h} \frac{a_4^h}{\partial w_{4,2}^h} + \frac{\partial \mathcal{L}}{\partial a_2^{out}} \frac{a_2^{out}}{\partial a_4^h} \frac{a_4^h}{\partial w_{4,2}^h} \right]$
- $\frac{\partial \mathcal{L}}{\partial b^{out}} = \left[\frac{\partial \mathcal{L}}{\partial a_1^{out}} \frac{\partial a_1^{out}}{\partial b^{out}} + \frac{\partial \mathcal{L}}{\partial a_2^{out}} \frac{\partial a_2^{out}}{\partial b^{out}} \right]$
- $\frac{\partial \mathcal{L}}{\partial b^h} \rightarrow$ left as exercise

Advanced Methodologies
Deep Learning: Backpropagation

- $\mathcal{E} = \hat{y} - y_i$
- $\delta_j \rightarrow$ error term for each unit
  - $\rightsquigarrow$ $\delta_j = \varphi'(a_j^t)$
- $\Delta w_{j,i} \rightarrow$ change in each weight
  - $\rightsquigarrow$ $\Delta w_{j,i} = \eta \cdot \delta_j \cdot a_j^t$
- $w_{new} = \Delta w + w_{old}$