Introduction to neural networks

Laboratory of robotics and control systems

ITT PROJECT MANAGEMENT SERVICES L.L.C

What is this course about?

Aim: provide students with basic skills in the field of deep learning computer vision in the context of robotics

Focus:

1. process of integrating and running neural nets on robots
2. optimizing model's performance and improving their quality
3. choosing model architecture for solving specific tasks related to the robot's perception of the environment

Caution: This course is more practical than fundamental. That's why some historical details, proofs, overview of some tasks and topics, not related to robotics, might be omitted

Course Structure

1. Into to DL, Neuron model, Perceptron, Losses, Backpropagation, PyTorch concpetions
2. Activation functions, Optimizers, Dataset splitting, Underfitting, Overfitting, Metrics
3. CV problems and data, Convolutional neural nets, More Pytorch Concepts, GPU
4. NNs architecutres: AlexNet, VGG, ResNet, Inception, MovileNet. Pretraining and Finetuning theory
5. Semantic segmentation: data, encoder-decoder, U-Net architecture, losses, metrics. SAM. Instance Segmentation
6. Object detection: data, metrics. Yolv3-v8 architecture
7. Object tracking
8. Pose Estimation
9. Point Clouds
10. Software integration: LibTorch, TorchScript, ONNX, OpenVINO. SAHR Vision Pipeline
11. Quantization
12. Pruning
13. Domain Adaptation, Self-supervised learning (DINO)

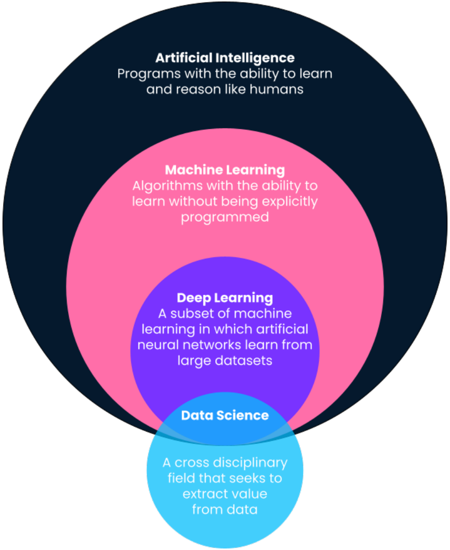
Course prerequisites

1. Basic Python, Numpy
2. Basic linear algebra, probability theory
3. CVR (Classic part) recommended

Plan

1. What is Machine learning?
2. Deep learning
3. Perceptron
4. Training process
5. PyTorch

Relation between AI, ML, DL



Types of Machine learning

Machine learning divides into:

- Supervised learning (have answers)
- Unsupervised learning (have no answers)
- Reinforcement learning (agent, environment, reward)
- Semi-supervised learning (partial labeling)
- Self-supervised learning (no answers, synthetic task in order to obtain good object representation)

Supervised learning

Provided:

- X - set of objects
- Y - set of answers
- $y : X \to Y$ - unknown dependency

We have to find $a : X \to Y$ that maps object x to expected output y.

Problems:

- Classification
- Regression
- Object detection
- Semantic segmentation

Regression vs Classification part 1

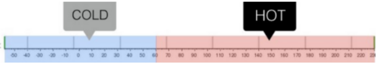
Regression

What will be the temperature tomorrow?

Fahrenheit

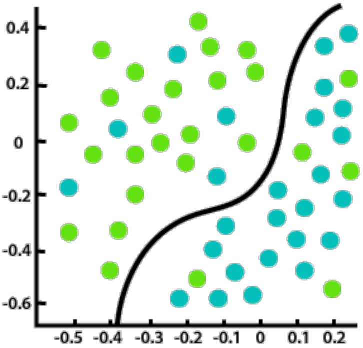
Classification

Will it be hot or cold tomorrow?



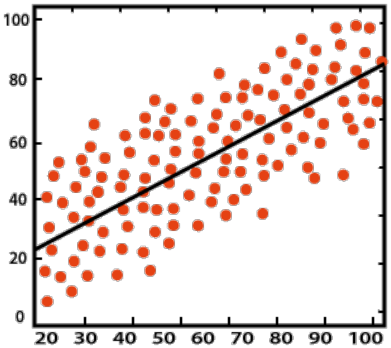
Fahrenheit

Regression vs Classification part 2



Classification

Given 2 features (x, y) predict dividing surface



Regression

Given 1 feature x predict y

Unsupervised learning

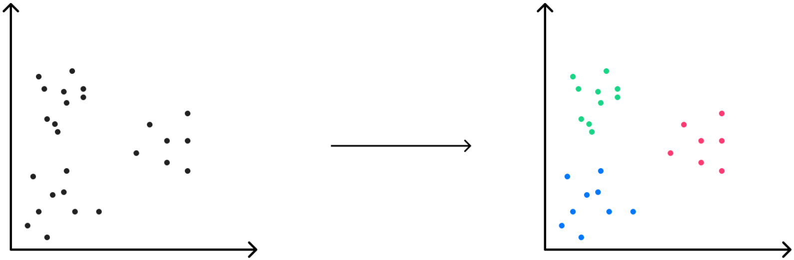
Features:

- Learns patterns exclusively from unlabeled data
- Due to missing labels, quality assessment is non-trivial

Problems:

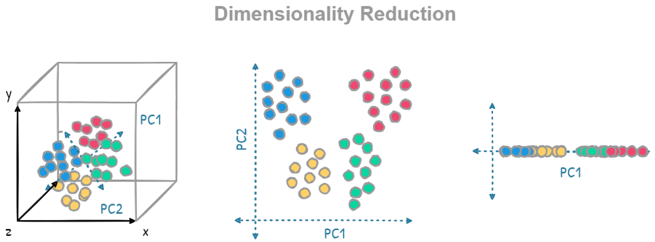
- Clustering
- Dimension reduction
- Anomaly detection

Clustering



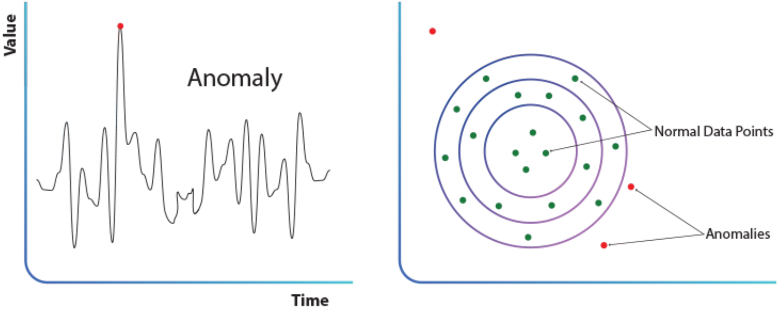
Given unlabeled data with 2 features (x, y), split objects into groups

Dimensionality reduction



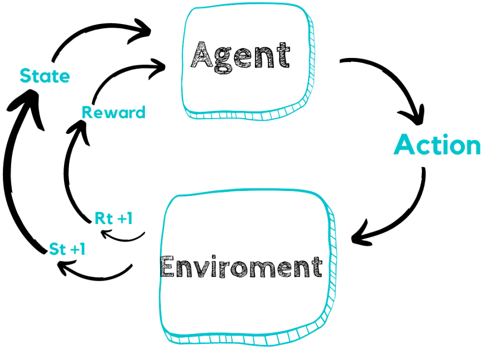
Remove excess information from data by projecting it into a lower dimensional space

Anomaly detection



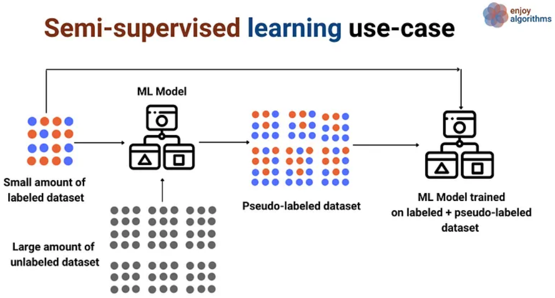
Find samples, that came from another data distribution

Reinforcement learning



- Agent performs action in environment
- Environment responds by giving it's new state and reward for action
- Environment serves as a teacher for the agent

Semi-supervised learning



Iterative process:

1. Label small amount of data
2. Train base model on labeled data
3. Generate pseudo-labels for unlabeled data
4. Take the most confident predictions
5. Add them to labeled partition
6. Repeat 2. - 6.

Self-supervised learning

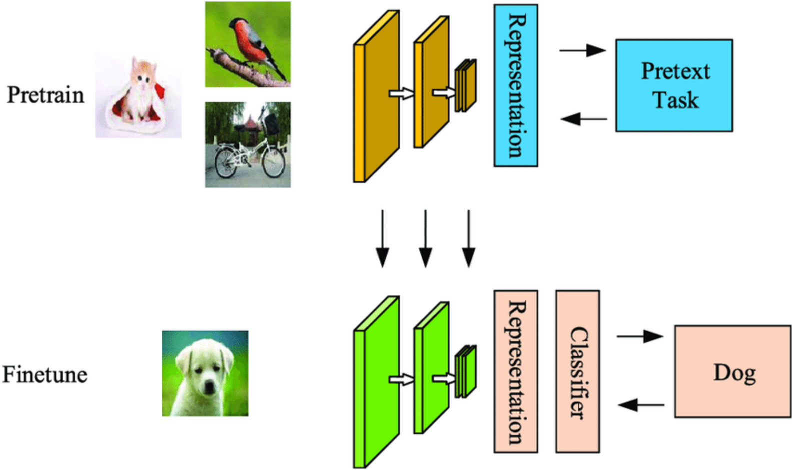
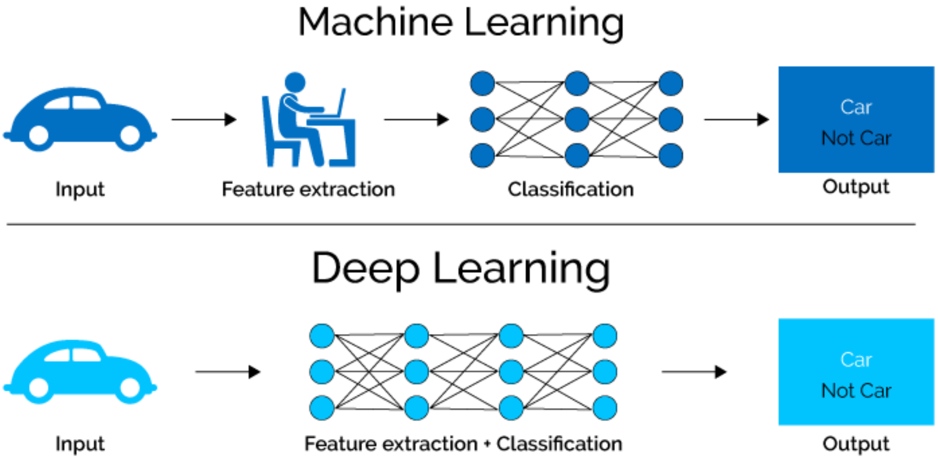
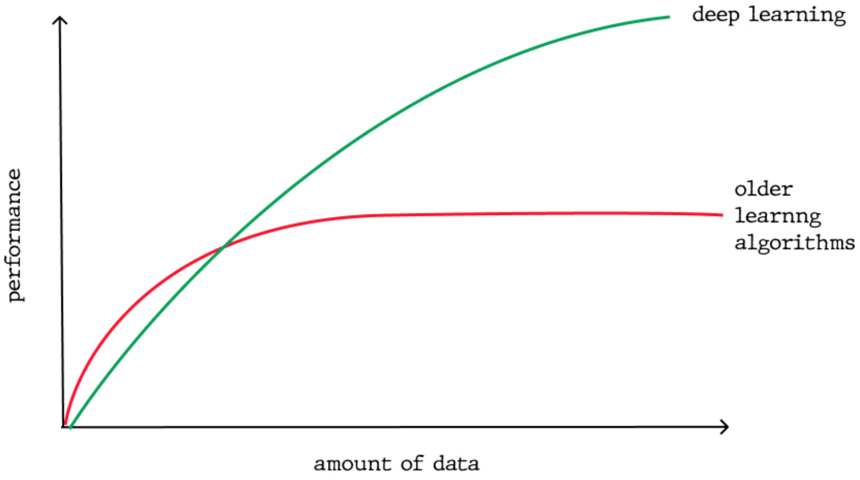


Table of Contents

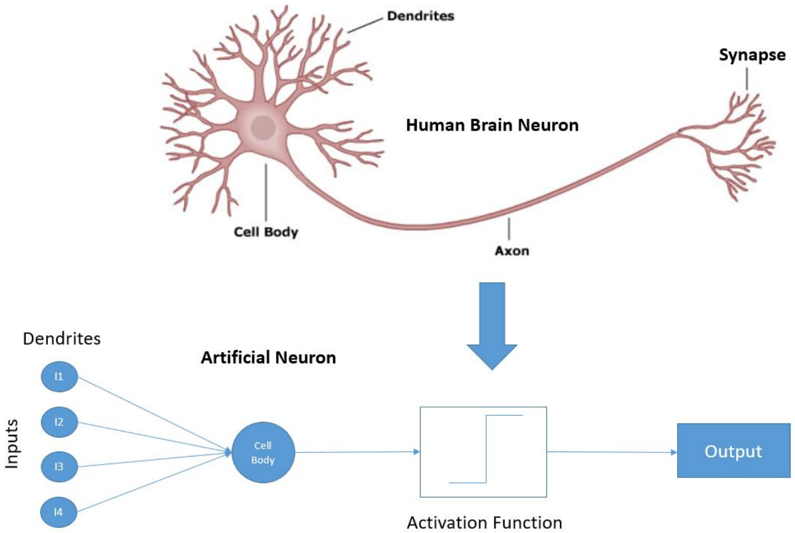
Classic ML vs Deep learning



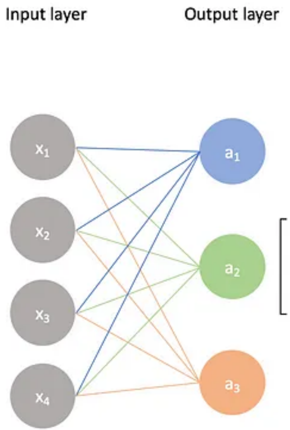
Comparison of scaling laws



Neuron structure



Linear layer



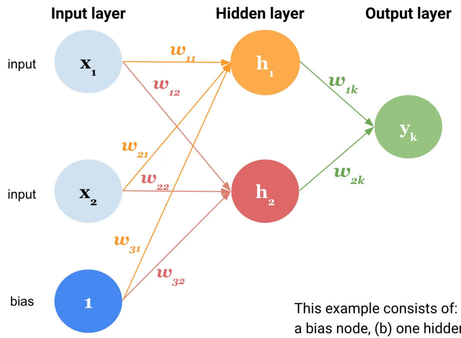
A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[\text{activation}]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Table of Contents

Multilayer perceptron (MLP)

Illustrative example of Multilayer perceptron, a Feedforward neural network



x_1, x_2 : *input data features*

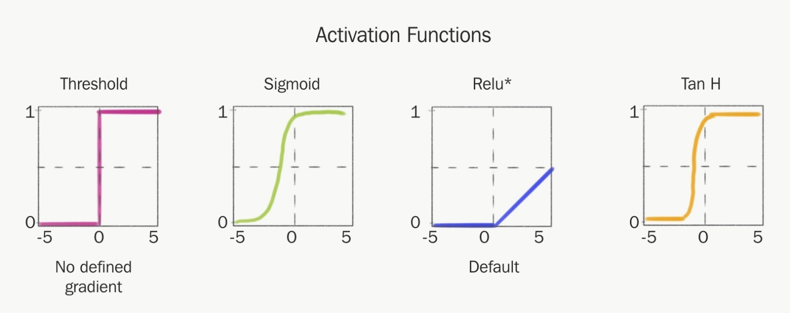
w_{ij} : *weights of the network*

h_1, h_2: *nodes in the hidden layer*

y_k : *output variable*

This example consists of: (a) an input layer with two input nodes and a bias node, (b) one hidden layer with two neurons, and (c) an output layer with one neuron

Some activation functions



Approximation theorems

Multilayer perceptron with 1 hidden layer and:

- arbitrary width, and sigmoid function (Cybenko, 1989)
- arbitrary width, and any activation function (Hornik, 1991)

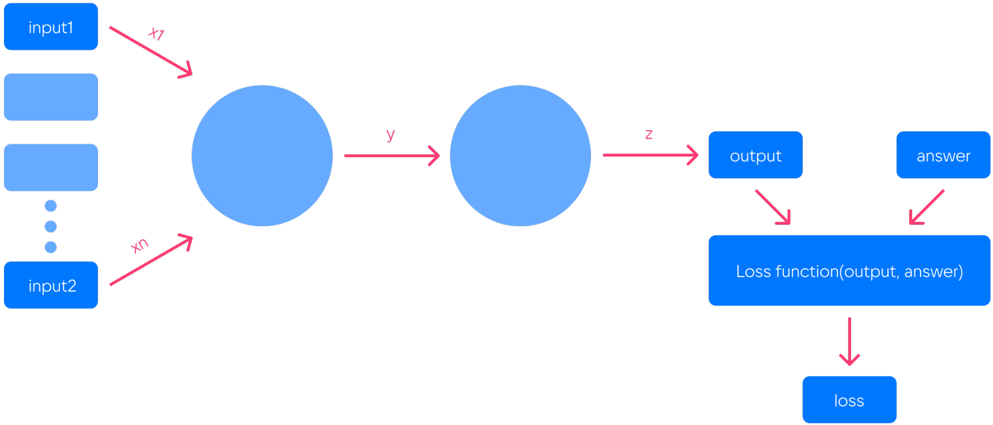
may approximate any continuous function with prefined precision ε

Weaknesses:

1. Theorems don't answer what is the width of MLP
2. Theorems don't answer how to find the parameters

Table of Contents

Forward Propagation



Loss function

- Estimates the quality of model's predictions
- Takes the model's prediction and the correct answer as an input
- Depends on the data type and the problem
- Is being minimized in the training process

Loss function examples

Regression:

Mean Squared Error: $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Mean Average Error: $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

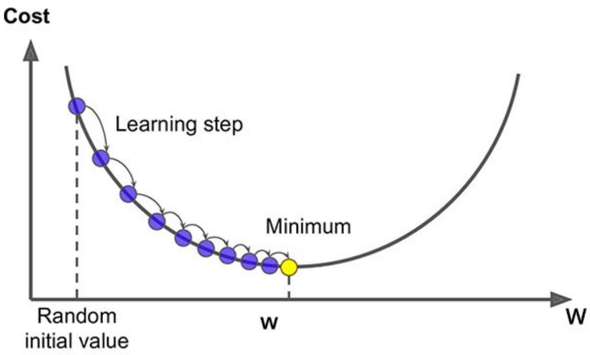
Mean absolute percentage error: $\frac{1}{n}\sum_{i=1}^{n}|\frac{y_i - \hat{y}_i}{y_i}|$

Binary Classification:

Cross Entropy Loss: $\frac{1}{n}\sum_{i=1}^{n} -(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$ $\quad (\hat{y}_i \in \{0, 1\})$

Hinge Loss: $\frac{1}{n}\sum_{i=1}^{n} max(1 - y_i\hat{y}_i)$ $\quad (\hat{y}_i \in \{-1, 1\})$

Gradient descent



$$F(\omega) = \sum_{i}^{N} \frac{L(x_i, y_i, \omega)}{N} + \lambda R(\omega)$$

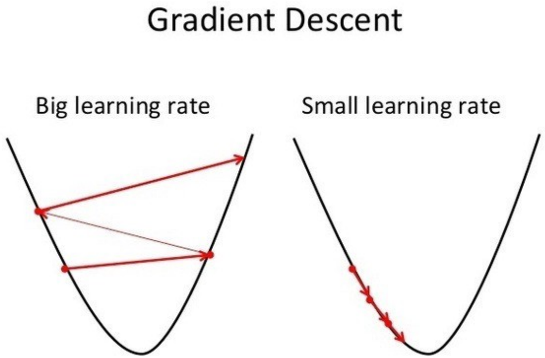
$$\nabla f(\omega_1, \ldots, \omega_n) = (\frac{\partial f}{\partial \omega_1}, \ldots, \frac{\partial f}{\partial \omega_n})$$

$$w := w - \alpha \nabla f$$

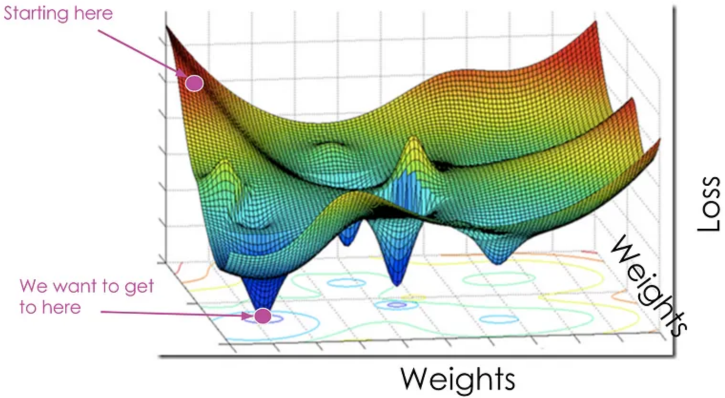
Learning rate

$$w := w - \alpha \nabla f$$

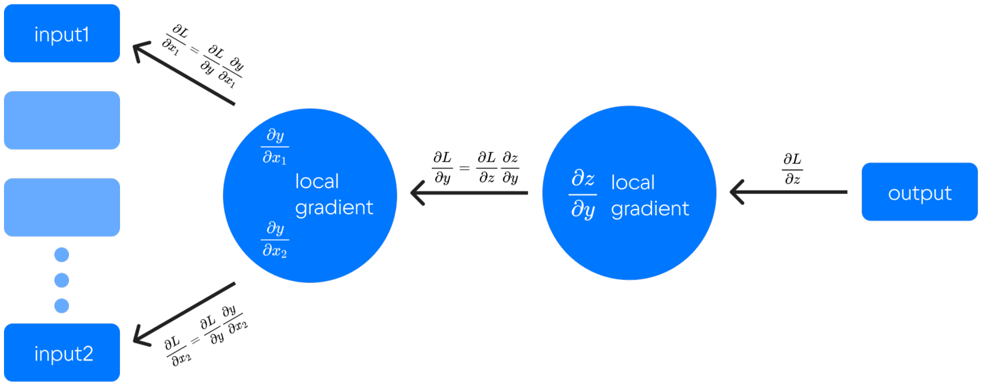
α - learning rate, controls convergence speed



Loss function of 2 variables example



Backpropagation



Backpropagation

$$L = L(O_n(...(O_i(O_{i-1}(...(O_1))))))$$

$$O_i = O_i(W_i, O_{i-1})$$

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial O_i}\frac{\partial O_i}{\partial W_i}$$

$$\frac{\partial L}{\partial O_{i-1}} = \frac{\partial L}{\partial O_i}\frac{\partial O_i}{\partial O_{i-1}}$$

$$O_n = Y_{pred}$$

$$O_1 = X$$

Table of Contents

Pytorch

Features:

- Automatic differentiation for building and training neural networks
- Native GPU and CPU usage
- Numpy-like Tensors
- Dynamic Computational graph
- Prototyping of neural networks using torch.nn and torchvision

PyTorch = Numpy + GPU + Autograd

Autograd

PyTorch’s automatic differentiation engine that powers neural network training

.backward()

- computes the gradients from each .grad_fn
- accumulates them in the respective tensor’s .grad attribute
- using the chain rule, propagates all the way to the leaf tensors

Autograd

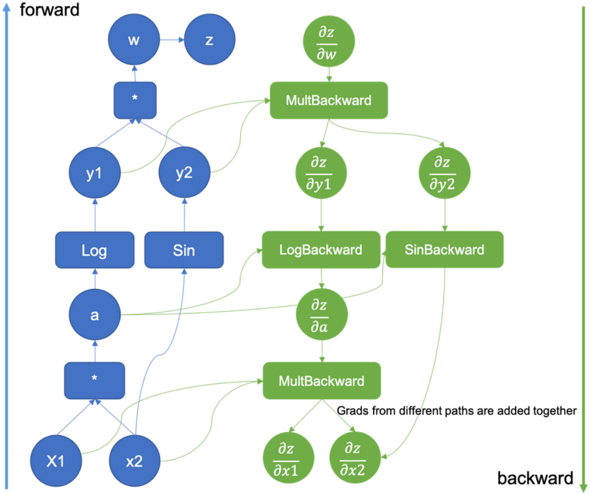


Figure 1: Example of an augmented computational graph

Thank you for your time

Laboratory of robotics and control systems
ITT PROJECT MANAGEMENT SERVICES L.L.C