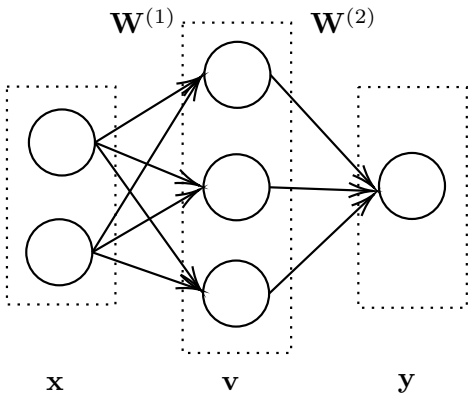
5 Optimisation

To obtain extra points for the written exam, prepare the solution for **Programming Task 5.3** using the Jupyter notebook provided and upload your work to StudOn. Students may discuss with each other while preparing the solution, but each student must submit their work individually. Refer the Supplements page in StudOn for details regarding the submission deadline.

5.1 Backpropagation

Consider the network shown in the figure below. The input to the network $\mathbf{x}$ is connected to a hidden layer. The hidden layer consist of a fully connected layer with weights given by the matrix $\mathbf{W}^{(1)}$ followed by a ReLU activation layer. The output of the hidden layer, denoted by $\mathbf{v}$, is connected to the output $\mathbf{y}$ via a linear fully connected layer with weights given by the matrix $\mathbf{W}^{(2)}$. For the sake of simplicity we assume no biases in fully connected layers. Using the network output $\mathbf{y}_{predicted}$ and the target output $\mathbf{y}_{target}$, the cost function in the network is expressed as:

$$C(\mathbf{y}_{predicted}, \mathbf{y}_{target}) = \|\mathbf{y}_{predicted} - \mathbf{y}_{target}\|_2^2$$



The initial weights are given by:

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & -2 \\ 1 & 5 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} \end{bmatrix} = \begin{bmatrix} -1 & 3 & 2 \end{bmatrix}. \tag{A}$$

The ReLU function $\sigma(x)$ and its derivative $\sigma'(x)$ is given by:

$$\sigma(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \qquad \sigma'(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0. \end{cases}$$

For this task, assume the input $\mathbf{x}$ and the corresponding target output $\mathbf{y}_{target}$ to be:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \mathbf{y}_{target} = \begin{bmatrix} 3 \end{bmatrix}. \tag{B}$$

i. Calculate the output of the hidden layer $\mathbf{v}$, the predicted output of the network $\mathbf{y}_{predicted}$ and the loss C. Assume that the initial weights in the network are given by (A) and the input and the target output are given by (B).

ii. Calculate the value of the partial derivative of the loss function with respect to weight element $w_{11}^{(1)}$ from the first layer.

iii. Calculate the value of the partial derivative of the loss function with respect to the weight matrix $W^{(2)}$ from the second layer.

iv. Using learning rate $\alpha = 0.05$, calculate the new weight matrix $\mathbf{W}_{updated}^{(2)}$ after one update of the gradient descent algorithm.

5.2 Programming Task: Pytorch Tutorial

The tutorial is divided into 3 parts:

i. In the first part, some basic operations in PyTorch are introduced. Some of the building blocks for neural networks such as activation functions and optimisers are also discussed here.

ii. In the second part, implementation of the network from the task 5.1 is provided. You may verify your answer from the previous task and also generate your own practice numerical tasks by applying different values for weights, activation functions, etc.

iii. Finally, an example of a feed-forward network used for detecting handwritten digits from images is provided. The MNIST dataset is used to train and test the network. You may use this example to understand how the network is structured using the PyTorch Framework.

5.3 Programming Task: Digit recognition using CNNs

The goal of this task to build and train a Convolutional Neural Network that classifies the handwritten digit images. The MNIST dataset from task 5.2 can be reused here. You may refer the PyTorch documentation to implement the layers of the network [`https://pytorch.org/docs/stable/nn.html`].

i. Complete the network architecture in the class `ConvNet` based on the characteristics given below:

- The input to the network is a grayscale image of size 28×28.

- The first layer of the network is a convolution layer that has a kernel size of 5×5, stride length 1 and has 20 output feature maps. The convolution layer output is connected to a ReLU activation layer.
- A max pooling layer with a 2×2 pooling window and stride length of 2 is applied on the output from the activation layer.
- The max-pool output is flattened into a long vector and is provided as input to a fully connected layer with 100 neurons, followed by a ReLU activation layer.
- The final predict layer is a fully connected layer that produces a vector containing the class probabilities for the 10 classes i.e. 10 digits.

ii. Train the CNN and observe the difference in performance in comparison to the feed-forward network from the task 5.2.

iii. Calculate the number of learnable parameters and the output shape in each layer. Verify your answers with model summary provided in the last cell of the notebook.