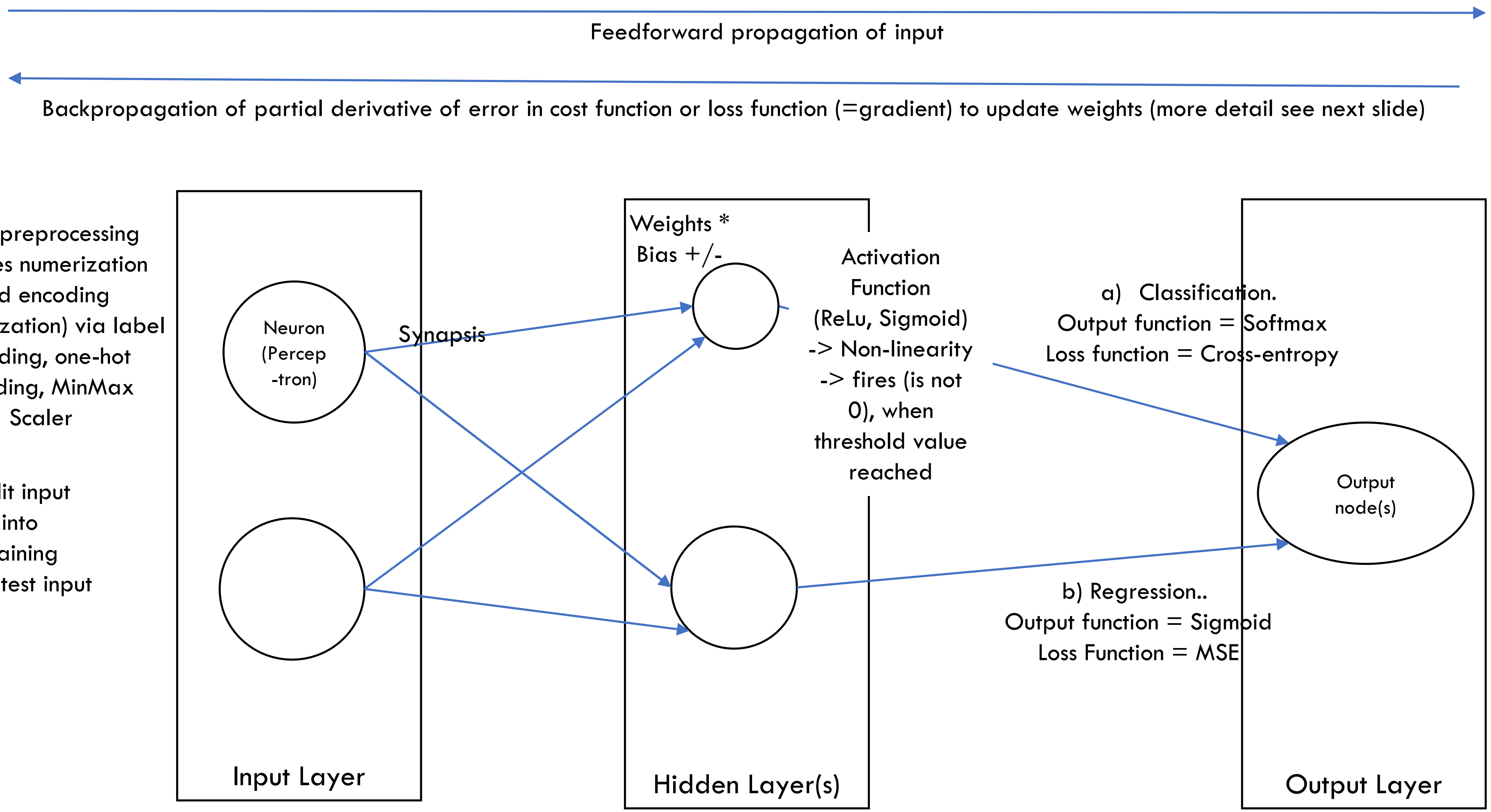
Artificial Neural Network (ANN) (Deep Learning = Many Hidden Layers)

ML Problems

- Supervised (labelled)
- Unsupervised
- Reinforcement
- Regression vs. classification



Feedforward propagation of input

Backpropagation of partial derivative of error in cost function or loss function (=gradient) to update weights (more detail see next slide)

Activation Function (ReLu, Sigmoid) -> Non-linearity -> fires (is not 0), when threshold value reached

a) Classification. Output function = Softmax Loss function = Cross-entropy

b) Regression.. Output function = Sigmoid Loss Function = MSE

Data preprocessing includes numerization and encoding (normalization) via label encoding, one-hot encoding, MinMax Scaler

Split input into training and test input

Hyperparamenter = parameter that is optimized by human (e.g. learning rate)

overfitting (model not generalizable because too much noise) prevention:

- regularization, normalization
- dropouts and ensembles

- 1 round = epoch
- mini-batch gradient descent vs. stochastic gradient descent (based on single observations) with Adam
- learning rate (pace of descent)
- Multi-class classification is based on each class having a separate node in the output layer

More on Updating the Weights

- Gradient descent: minimizes loss function by «descending into» minimum point in the function based on the gradients (derivatives for multi-variable functions, vector instead of scalar value for one-variable functions)
    - Partial Derivatives measure the rate of change of the slope of the function with respect to the change of a particular variable
    - learning rate controls how fast the weights are updated
    - Approach: initialize value for X, subtract (learning rate * result of partial derivative function) at X (i.e. for y = x^2 + 5x, derivative function is 2x + 5), iterate until difference between 2 subsequent X-values becomes smaller than threshold
- Backpropagation: used in deep neural networks to update the weights based on partial derivatives of loss function and the chain rule
    - let a function be y1 = f(x1, w1)
    - propagation: generating prediction $\widehat{y1}$ (forward propagation), calculating error(loss) term, compute delta output weights from layers based on their partial derivative and the chain rule (backward propagation)
    - weight update: multiply delta output weight with activation to find gradient, subtract gradient multiplied with learning rate from weight