Computational Intelligence

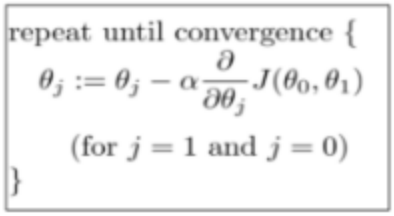
▼ Describe the difference between unsupervised and supervised learning including
differences in training datasets. Explain the principle of the supervised gradient learning, how
the gradient is calculated and hyperparameters of the algorithm.

- ▼ Supervised Learning
  1. algorithm is trained on a labeled dataset
  2. The goal of supervised learning is to learn a function that maps inputs to
     outputs, so that when given new input data, the algorithm can predict the
     correct output.
  3. If y is quantitative (numeric), we want to solve a regression problem. If y is
     qualitative (categorical), we want to solve a classification problem
  4. Examples of supervised learning include linear regression, logistic regression, and decision trees.
- ▼ Unsupervised Learning
  1. algorithm is trained on a unlabeled dataset
  2. The goal of unsupervised learning is to find patterns, structures, or relationships in the data that are not immediately obvious.
  3. Examples of unsupervised learning include clustering, dimensionality reduction, and anomaly detection.

- ▼ Principle of supervised gradient learning

The principle of supervised gradient learning is to iteratively adjust the model's parameters so as to minimize a cost function that measures the difference between the model's predictions and the true output values for the training data.
The process of training a model typically starts with randomly initializing the model's parameters. Then, for each iteration of the training process, the model makes predictions on the training data and the cost function is calculated.

The gradient of the cost function with respect to the model's parameters is then computed using backpropagation. The model's parameters are then adjusted in the opposite direction of the gradient, with the size of the adjustment determined by a learning rate. This process is repeated many times, with the model's parameters being adjusted at each iteration, until the cost function reaches a minimum value. At this point, the model is considered to be trained, and its parameters are considered to be the optimal values for making predictions on new, unseen input data.
The most commonly used optimization algorithms to find the optimal parameters are Stochastic Gradient Descent, Adagrad, Adadelta, Adam, and so on. These are variants of Gradient Descent algorithm which tries to optimize the cost function.



▼ hyperparameters of the algorithm

learning rate, epochs and batch size

Techniques like cross-validation and grid search can be used to search for optimal hyperparameter values efficiently.

▼ **Cross-Validation:**

Cross-validation is a resampling technique used to evaluate machine learning models on a limited data sample. It involves splitting the dataset into multiple subsets or folds, training the model on some of these folds, and evaluating it on the remaining fold(s). This process is repeated multiple times, with each fold serving as the validation set exactly once. There are different types of cross-validation, with k-fold cross-validation being the most common.

Hyperparameter Tuning: Different sets of hyperparameters are tested using cross-validation. The set of hyperparameters that yield the best average performance across all folds is selected as the optimal choice.

▼ Shallow and deep neural networks (DNNs). **Typical neurons, their activations** and typical
activation functions. Describe convolution, pooling and dense layers and their parameters.
Application of DNNs for object detection in images.

▼ Shallow neural networks:

1. **Definition**: Shallow neural networks have a small number of hidden layers, typically one or two.
2. **Architecture**: Shallow neural networks usually consist of an input layer, one or two hidden layers, and an output layer.
3. **Feature Representation**: They are limited in their ability to learn complex patterns and hierarchical representations from data due to their shallowness. They are often effective for simple tasks and datasets with relatively low complexity.
4. **Training and Optimization**: Training shallow neural networks is relatively faster compared to deep neural networks because there are fewer parameters to learn. However, they may struggle with capturing intricate relationships in complex datasets.
5. **Examples**: Logistic Regression, Perceptron, Single-layer Feedforward Neural Networks.

▼ Deep neural networks:

1. **Definition**: Deep neural networks have a large number of hidden layers, typically more than two.
2. **Architecture**: Deep neural networks can be characterized by their depth, with architectures containing multiple hidden layers stacked on top of each other
3. **Feature Representation**: Deep neural networks excel at learning hierarchical representations of data, where each layer captures increasingly abstract and complex features.
4. **Training and Optimization**: Training deep neural networks can be more challenging than training shallow networks due to the increased complexity. Techniques such as batch normalization, and advanced optimization algorithms (e.g., Adam, RMSprop) are often employed to facilitate training.
5. **Examples**: Convolutional Neural Networks (CNNs) for image recognition, Recurrent Neural Networks (RNNs) for sequence data

▼ Typical activation functions

Sigmoid function: This function maps any real-valued input to a value between 0 and 1, which makes it useful for binary classification tasks. The sigmoid function is defined as: 1/(1+e^(-x))

Hyperbolic tangent (tanh) function: This function maps any real-valued input to a value between -1 and 1, which also makes it useful for binary classification tasks. The tanh function is defined as: (e^x - e^(-x))/(e^x + e^(-x))

ReLU (Rectified Linear Unit) function: This function maps any real-valued input to the input value if it is positive and to 0 if it is negative. This makes it useful for tasks that require a non-linear decision boundary. The ReLU function is defined as: max(0,x)

Softmax function : It is used in the output layer of multi-class classification problems, it maps the output of the last layer to a probability

distribution over the classes.

▼ Convolution layer

A convolutional layer is **the main building block of a CNN**. It is used for tasks like image recognition, object detection, and image segmentation.

Here's a description of a convolutional layer and its parameters:

1. **Convolution Operation**: The core operation performed by a convolutional layer is the convolution operation. It involves applying a set of learnable filters (also known as kernels or convolutional kernels) to the input data. Each filter is a small-sized matrix that slides across the input data (image or feature maps) and computes the dot product between its weights and the local regions of the input data. This process extracts spatial patterns or features from the input.
2. **Parameters**:
    - **Number of Filters (or Kernels)**: This parameter defines the depth of the output volume produced by the convolutional layer. Each filter detects different features in the input data. For example, in the first layer of a CNN designed for image processing, the number of filters may correspond to different edge detectors, color detectors, or other low-level features.
    - **Filter Size (or Kernel Size)**: This parameter determines the spatial dimensions of the filters. Common filter sizes are 3×3, 5×5, or 7×7. Larger filter sizes capture more spatial information but increase the computational cost.
    - **Stride**: The stride specifies the number of pixels by which the filter shifts or slides over the input data. A stride of 1 means the filter moves one pixel at a time, while a larger stride skips pixels and reduces the spatial dimensions of the output feature maps.
    - **Padding**: Padding refers to the additional border added to the input data to preserve its spatial dimensions after convolution. It helps in retaining more information at the borders of the input image. Common padding options are "valid" (no padding) and "same"

(pad the input such that the output has the same spatial dimensions as the input).

▼ Pooling layer

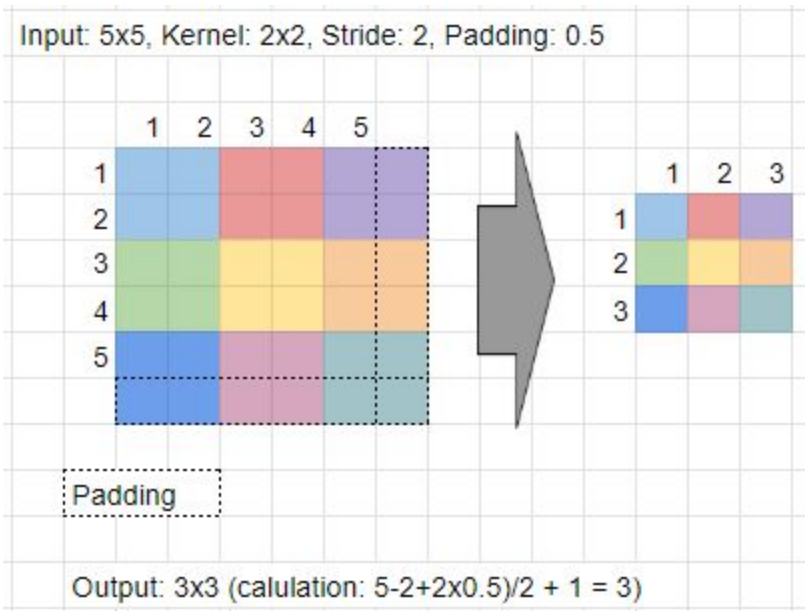
Pooling layers provide an approach to down-sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling.

Max pooling selects the maximum value of a group of adjacent pixels in the feature map and replaces them with this maximum value. It is typically used to retain the most important information in the feature map and to reduce the spatial size of the feature maps.

Average pooling, on the other hand, computes the average of a group of adjacent pixels in the feature map and replaces them with this average value. It is typically used to reduce the spatial size of the feature maps while also preserving the information present in the feature map.

Parameters

1. Pool size:Also known as the pooling window or kernel size, this parameter defines the spatial dimensions (width and height) of the pooling regions.
2. Padding
3. Strides
4. Pool type: max or avg pooling



▼ Dense layer

A dense layer, also known as a fully connected layer, is a type of layer commonly used in neural networks where each neuron in the layer is connected to every neuron in the preceding layer.

$$y_i = f\left(\sum_{j=1}^{n} w_{ij} \cdot x_j + b_i\right)$$

where:

- x_j is the input from the j^{th} neuron in the previous layer.
- w_{ij} is the weight associated with the connection between the j^{th} neuron in the previous layer and the i^{th} neuron in the dense layer.
- b_i is the bias term associated with the i^{th} neuron.
- f is the activation function applied element-wise to the weighted sum.

Parameters:

- **Number of Neurons**: The number of neurons in the dense layer determines the dimensionality of the output. It's a crucial hyperparameter that affects the capacity of the model to learn complex patterns.
- **Activation Function**: Dense layers typically include an activation function to introduce non-linearity into the network. Common

activation functions include ReLU, sigmoid, tanh, and softmax (for output layers).

- **Weights**: Each connection between neurons in the previous layer and neurons in the dense layer is associated with a weight parameter. These weights are learned during the training process to adjust the strength of connections and minimize the loss function.
- **Biases**: Each neuron in the dense layer has an associated bias term. The bias term allows the model to learn a constant offset in addition to the weighted sum of inputs.

▼ Application of DNN

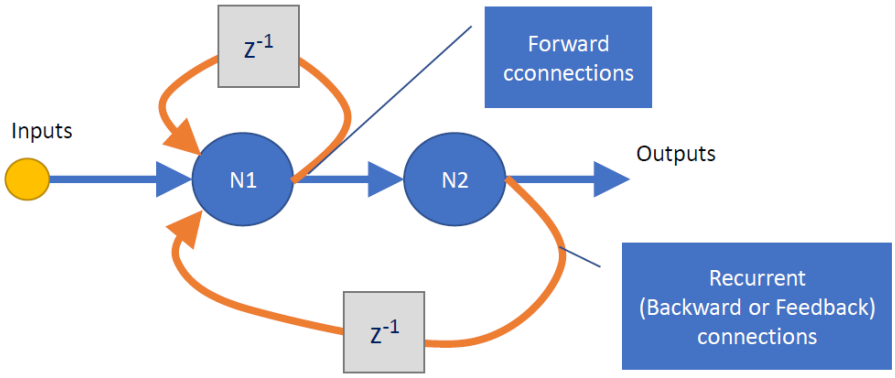
1. Computer Vision: DNNs have been used to perform image classification, object detection, semantic segmentation, and other computer vision tasks.
2. Natural Language Processing (NLP): DNNs have been used for tasks such as language translation, text generation, sentiment analysis, and named entity recognition.
3. Speech Recognition: DNNs have been used to build speech recognition systems that can transcribe speech to text with high accuracy.
4. Recommender Systems: DNNs have been used to build recommendation systems that can recommend items such as movies, music, and books based on user preferences

▼ Recurrent neural networks (RNN). Explain what the recurrent connections and necessity of
their specific evaluation are. Explain fully recurrent and LSTM neural architecture.

▼ Recurrent neural network:

A recurrent neural network (RNN) is a type of neural network that allows the information to flow in a cyclic manner, with feedback connections. This allows the network to maintain a hidden state, which can be used to store information about the previous inputs. This type of network is used for

tasks such as language modeling, speech recognition, and time series prediction.



▼ Recurrent connections:

Imagine you're reading a story, and as you go through each sentence, you remember what happened in the previous sentences. This ability to remember past information and use it to understand the current part of the story is similar to how recurrent connections work in neural networks.

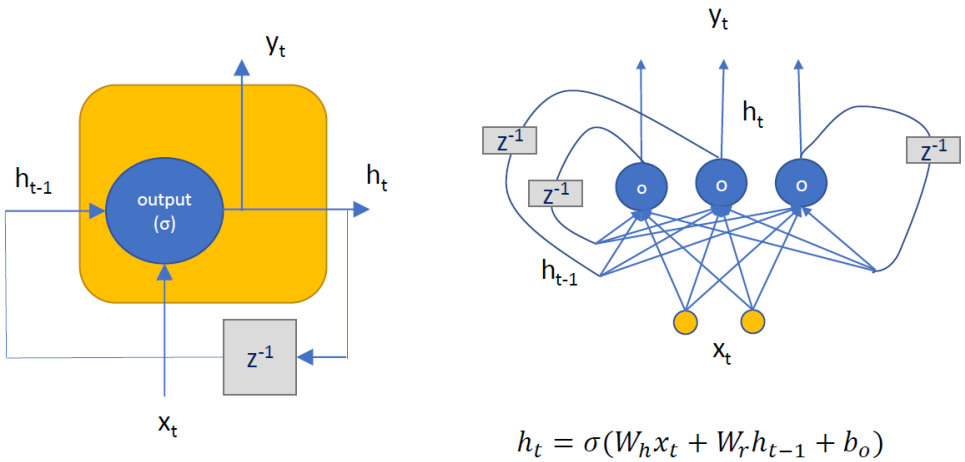
▼ necessity of their specific evaluation

1. To ensure the network remembers important information from previous steps. This memory is crucial for tasks like understanding language or predicting future values accurately.
2. Information in these connections can get too weak or too strong, causing issues. By evaluating them, we can catch problems early and fix them.
3. By evaluating how well recurrent connections are working, we can make adjustments to improve the overall performance of the neural network

▼ Fully recurrent network

1. **Basic Structure**: In a fully recurrent neural network, each neuron has a connection to itself, allowing it to pass its output from one time step to the next. This looping structure enables the network to maintain information over time, making it suitable for tasks where past context matters, such as language modeling or time series prediction.

2. **Memory and Computation**: At each time step, the network takes in an input vector and combines it with the output from the previous time step. This combined input is then passed through an activation function to produce the output for the current time step. The output also serves as the input for the next time step, creating a recurrent loop.
3. **Training Challenges**: While fully recurrent networks are powerful for handling sequential data, they suffer from challenges like vanishing or exploding gradients during training. These issues can hinder the network's ability to learn long-range dependencies effectively.



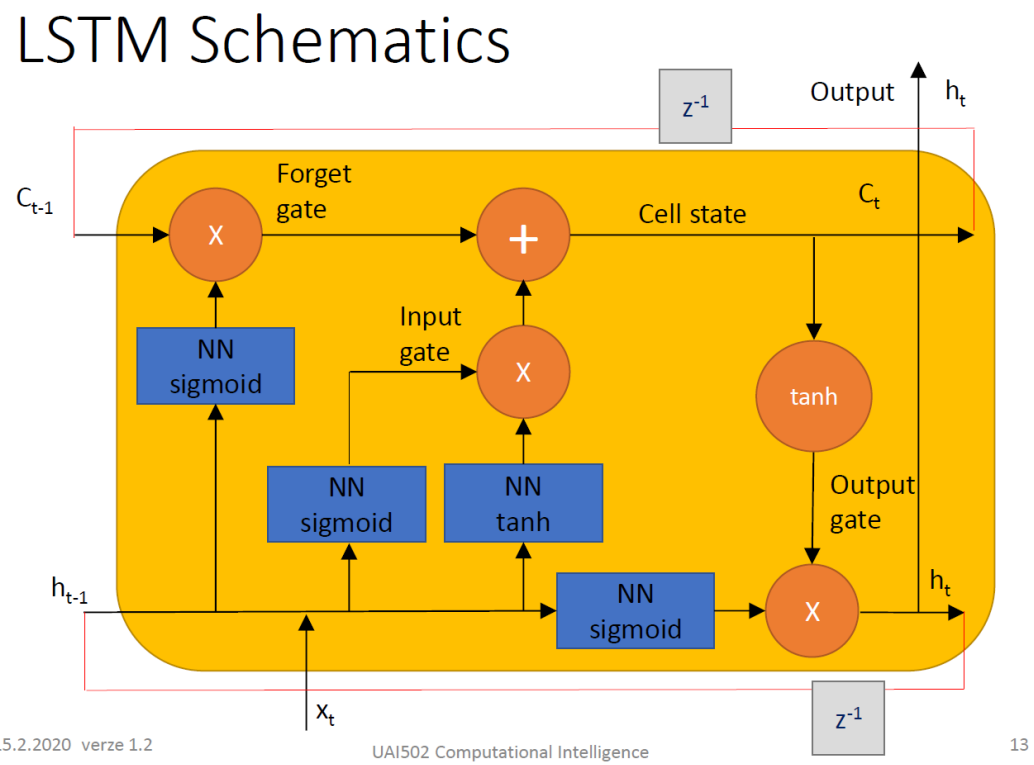
$$h_t = \sigma(W_h x_t + W_r h_{t-1} + b_o)$$

$$y_t = h_t \quad \text{or subset of } h_t$$

▼ LSTM

1. **Memory Cells**: The core component of an LSTM network is the memory cell. Unlike traditional neurons in fully recurrent networks, LSTM cells have a more complex structure that allows them to maintain a memory state over long sequences.
2. **Gates**: LSTM cells incorporate different types of gates, including input gates, forget gates, and output gates. These gates control the flow of information into and out of the memory cell, enabling the network to learn when to remember, forget, or use information.

3. **Long-Term Dependencies**: By incorporating mechanisms like forget gates and input gates, LSTM networks can effectively capture and retain information over many time steps. This ability to handle long-term dependencies makes them well-suited for tasks such as machine translation, speech recognition, and sentiment analysis.
4. **Training Stability**: LSTM networks are designed to mitigate the vanishing and exploding gradient problems encountered in fully recurrent networks. The gated structure helps the network to maintain stable gradients during training, enabling more effective learning of sequential patterns.



▼ Reinforcement learning. Explain the principle. What is the environment state, observation,
reward, policy? Describe the actor-critic architecture.

▼ Reinforcement learning

Learning system, that is based on an agent that observes his environment and

context.
- The agent can perform actions
- Actions get penalized or rewarded
- Aim is to learn a strategy that yields the highest reward over time

The principle of reinforcement learning (RL) is based on the idea of learning through interaction with an environment to achieve a goal or maximize some notion of cumulative reward. RL is inspired by the way humans and animals learn from experience by taking actions and receiving feedback from the environment.

▼ Components

Environment: The environment represents the external system with which the agent interacts. It encapsulates the dynamics of the problem the agent is trying to solve.

State (S): The state of the environment represents the current situation or configuration of the environment at a particular time step. It captures all relevant information needed for decision-making by the agent. In many cases, the state is not directly observable by the agent and may be partially observable or fully observable.

Observation (O): The observation refers to the data that the agent perceives from the environment at each time step. It may include the state information if the state is fully observable, or it may include partial information about the state if the state is partially observable. Observations are used by the agent to make decisions and take actions.

Action (A): An action represents the decision made by the agent at each time step based on the current observation. Actions are chosen from a set of possible actions available to the agent in the given environment. The goal of the agent is to learn a policy that maps observations to actions in a way that maximizes cumulative rewards over time.

Reward (R): The reward is a scalar feedback signal provided by the environment to the agent after each action. It indicates the immediate

benefit or cost associated with taking that action in that particular state. The agent's objective is to learn a policy that maximizes the cumulative sum of rewards over time, often referred to as the return.

Policy (π): The policy is a mapping from observations to actions, specifying the agent's behavior or strategy in the environment. It defines the agent's decision-making process and determines which action to take in each observed state.

▼ Actor-critic architecture

The actor-critic architecture is a popular framework in reinforcement learning that combines elements of both value-based and policy-based methods. It consists of two main components:

1. **Actor**: The actor is responsible for selecting actions based on the current observation. It learns a policy (π) that maps observations to actions by directly optimizing the policy through gradient ascent using techniques such as policy gradients.
2. **Critic**: The critic evaluates the actions taken by the actor by estimating the expected return or value of taking those actions in the given state. It learns a value function (V or Q) that approximates the expected cumulative reward of following a particular policy. The critic provides feedback to the actor by estimating the advantage or the difference between the observed rewards and the expected rewards.

During training, the actor learns to improve its policy based on the feedback from the critic, while the critic learns to better estimate the expected returns by minimizing the difference between predicted and observed rewards. The actor-critic architecture leverages the strengths of both value-based and policy-based methods, leading to more stable and efficient learning in reinforcement learning tasks.

▼ Nature inspired optimization. Explain the principle of the genetic algorithm and the particle
swarm optimization algorithm.

▼ Nature inspired optimization

Nature-inspired optimization algorithms are computational methods that mimic the behavior of natural systems to solve complex optimization problems. These algorithms are inspired by the way biological systems or natural phenomena, such as evolution, swarm behavior, and survival of the fittest, work to find optimal solutions.

▼ **Genetic Algorithm:**

The principle behind the genetic algorithm is inspired by Darwin's theory of natural selection and genetics. It starts with a population of potential solutions (represented as individuals or "chromosomes") to a problem. These solutions undergo a process similar to evolution, where better solutions have a higher chance of survival and reproduction.

Here's a simplified explanation of how a genetic algorithm works:

1. **Initialization**: Start with a random population of potential solutions.
2. **Evaluation**: Evaluate each solution's fitness, which represents how good it is at solving the problem.
3. **Selection**: Select individuals from the population based on their fitness. Solutions with higher fitness have a higher chance of being selected.
4. **Crossover**: Create new solutions by combining traits (or parts) of selected individuals. This simulates mating or crossover in genetics.
5. **Mutation**: Introduce small random changes to some solutions in the population. This mimics genetic mutation.
6. **Replacement**: Replace some solutions in the population with the newly created ones.
7. **Repeat**: Repeat the process (steps 2-6) for a certain number of iterations or until a stopping criterion is met (e.g., a satisfactory solution is found).

Through this iterative process of selection, crossover, and mutation, the population evolves over generations, and better solutions are gradually discovered.

▼ **Particle Swarm Optimization (PSO):**

Particle Swarm Optimization is inspired by the social behavior of bird flocking or fish schooling. In PSO, potential solutions to an optimization problem are represented as particles in a multidimensional space.

Here's how PSO works in simple terms:

1. **Initialization**: Randomly place a group of particles in the search space.
2. **Velocity and Position Update**: Each particle adjusts its velocity and position based on its own experience and the experiences of its neighbors.
    - Velocity Update: Particles adjust their velocities to move toward their own best-known position (personal best) and the best-known position of any particle in the swarm (global best).
    - Position Update: Particles move through the search space according to their velocities.
3. **Evaluation**: Evaluate the fitness of each particle's current position.
4. **Update Personal and Global Best**: Update each particle's personal best position and the global best position found by any particle.
5. **Repeat**: Repeat steps 2-4 until a stopping criterion is met.

Through the continuous adjustment of velocities and positions based on personal and social information, particles in the swarm gradually converge towards the optimal solution.