Patrick Reignier

Grenoble INP-Ensimag

2024-2025

Outline of the project

- Goal: construct a neural network (almost) from scratch in Java
  - Using the EJML java matrix manipulation library
  - Some source code is provided:
    - A class library with interfaces for the neural networks
    - Several applications to help training and evaluating the implemented networks

Outline of the project

- Goal: construct a neural network (almost) from scratch in Java
  - Using the EJML java matrix manipulation library
  - Some source code is provided:
    - A class library with interfaces for the neural networks
    - Several applications to help training and evaluating the implemented networks
- Use the neural network to solve a regression problem
  - The neural network will be used to price a financial product
  - Competition to select the best neural network

Outline of the project

- Goal: construct a neural network (almost) from scratch in Java
  - Using the EJML java matrix manipulation library
  - Some source code is provided:
    - A class library with interfaces for the neural networks
    - Several applications to help training and evaluating the implemented networks
- Use the neural network to solve a regression problem
  - The neural network will be used to price a financial product
  - Competition to select the best neural network
- Evaluation
  - The source code
    - Correctness of the implementation
    - Structure, maintainability, efficiency
    - Implemented features
  - (How well the best network performs)
    - The serialized version of the best network that was constructed

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations
- Hands-on 2: Backpropagation, gradient descent (1.5 hours)
  - Implement Network loading and Saving using the provided Serializer
  - Update of the code to make backpropagation and gradient descent work

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations
- Hands-on 2: Backpropagation, gradient descent (1.5 hours)
  - Implement Network loading and Saving using the provided Serializer
  - Update of the code to make backpropagation and gradient descent work
- Hands-on 3: Mini-batch, gradient acceleration (4.5 hours)
  - Update of the code to include batch size
  - Implementation of Momentum gradient acceleration

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations
- Hands-on 2: Backpropagation, gradient descent (1.5 hours)
  - Implement Network loading and Saving using the provided Serializer
  - Update of the code to make backpropagation and gradient descent work
- Hands-on 3: Mini-batch, gradient acceleration (4.5 hours)
  - Update of the code to include batch size
  - Implementation of Momentum gradient acceleration
- Hands-on 4: Regularization (1.5 hours)
  - Implementation of an L^2 layer

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations
- Hands-on 2: Backpropagation, gradient descent (1.5 hours)
  - Implement Network loading and Saving using the provided Serializer
  - Update of the code to make backpropagation and gradient descent work
- Hands-on 3: Mini-batch, gradient acceleration (4.5 hours)
  - Update of the code to include batch size
  - Implementation of Momentum gradient acceleration
- Hands-on 4: Regularization (1.5 hours)
  - Implementation of an L^2 layer
- **Pricing neural network** (6 hours)

Global timetable

- Hands-on 1: Forward propagation (1.5 hours)
  - Update of the code to make forward propagation work (`And.java`)
  - Validation: verify the and fonction is correcty computed
  - Refactoring of the code to reduce the number of instantiations
- Hands-on 2: Backpropagation, gradient descent (1.5 hours)
  - Implement Network loading and Saving using the provided Serializer
  - Update of the code to make backpropagation and gradient descent work
- Hands-on 3: Mini-batch, gradient acceleration (4.5 hours)
  - Update of the code to include batch size
  - Implementation of Momentum gradient acceleration
- Hands-on 4: Regularization (1.5 hours)
  - Implementation of an L^2 layer
- **Pricing neural network** (6 hours)
- See the README.md file for more explanations on what is provided