ANN
MM - AB
A*

Projets IA

A* | Min-Max | Alpha-Beta | ANN

Encadrés par : M. Toumi BOUCHENTOUF – M. Jamal BERRICH

Réalisés par :
AITELABD Salma – JABRI Abdelilah – MALKI Farah – TOUMI Ashraf

Plan

- Get Diplome: Application de l'heuristique A*
  - ❑ Planning du projet
  - ❑ L'algorithme A*
  - ❑ Greenfoot
  - ❑ Les classes utilisées
  - ❑ Démonstration
- Tic Tac Toe: Application de l'algorithme MIN-MAX et ALPHA-BETA
  - ❑ Planning du projet
  - ❑ L'algorithme Min-Max
  - ❑ L'algorithme Alpha-Beta
  - ❑ Alpha-Beta VS MIN-MAX
  - ❑ JADE
  - ❑ Agent en JADE
  - ❑ Les classes utilisées
  - ❑ Démonstration
- Song Year Prediction: Réseau de neurones artificiels & DL4J
  - ❑ Réseau de neurones
  - ❑ Deep Learning For Java
  - ❑ Description du program
  - ❑ La lecture du fichier de données & le changement de la structure de données de DL4J
  - ❑ La séparation en jeu de données d'entraînement et de test
  - ❑ La création et paramétrisation du réseau de neurones
  - ❑ L'entraînement
  - ❑ L'évaluation

ANN

MM - AB

A*

Get Diploma

Jeu 2D

Application de l'heuristique A*

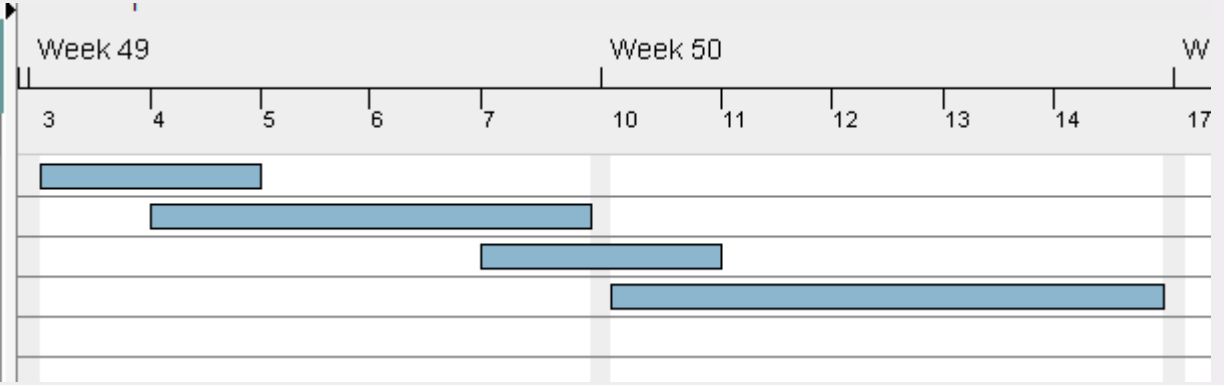
Planning du projet

ANN

MM - AB

A*

Name	Begin date	End date
Apprentissage de A*	12/3/18	12/4/18
Apprentissage de Greenfoot	12/4/18	12/7/18
Analyse & concepion des classes	12/7/18	12/10/18
Codage & Développement	12/10/18	12/14/18
Rédaction du présentation	1/21/19	1/23/19



L’algorithme A*

L'algorithme A* est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final. Il utilise une évaluation heuristique sur chaque nœud pour estimer le meilleur chemin y passant, et visite ensuite les nœuds par ordre de cette évaluation heuristique. C'est un algorithme simple, ne nécessitant pas de prétraitement, et ne consommant que peu de mémoire.

Greenfoot

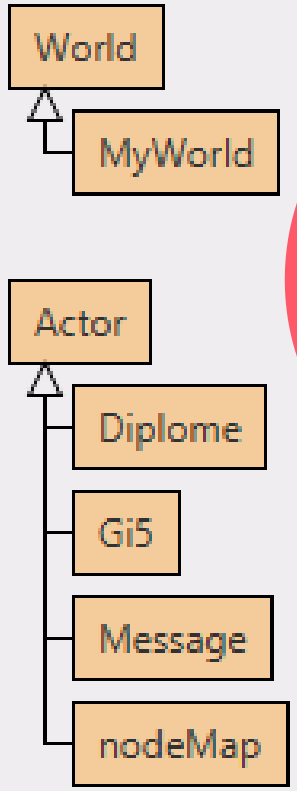
Greenfoot est un environnement de développement basé sur le langage de programmation Java. Il est principalement conçu pour faciliter l'apprentissage du langage Java et permet de facilement créer des jeux en 2D. Greenfoot est un logiciel libre (gratuit) et est disponibles pour la plupart des ordinateurs sous Microsoft Windows, Mac OS X, ou Linux.

Les classes utilisées

GreenFoot nous montre deux classes importantes:

-World: c'est le «monde» de notre jeu. L'objet World va dessiner l'écran. Notre Monde va hériter de la classe World.

-Actor: les différents éléments qui vont «vivre» dans le jeu. le point source et le point destination vont hériter de la classe Actor.



Les classes utilisées

ANN

MM - AB

A*

MyWorld

Représente l'écran de jeu.

Gi5

Représente le point source de départ. On a implementé l'algo A* dans cette classe.

Message

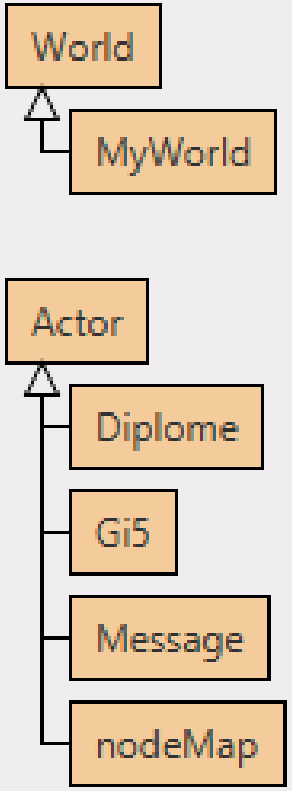
Représente les message de l'écran : Instruction – résultat – Game over…

Diplome

Représente le point destination.

nodeMap

Représente chaque noeud dans le jeu.



ANN

MM - AB

A*

Démonstration

ANN

Tic Tac Toe

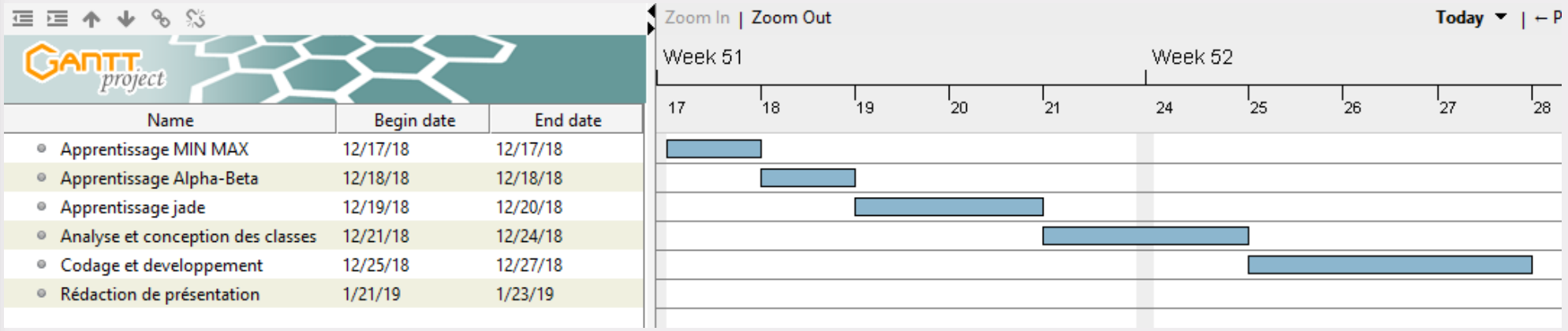
Jeu 2D

Application de l'algorithme MIN-MAX et ALPHA-BETA

MM - AB

A*

Planning du projet



Name	Begin date	End date
Apprentissage MIN MAX	12/17/18	12/17/18
Apprentissage Alpha-Beta	12/18/18	12/18/18
Apprentissage jade	12/19/18	12/20/18
Analyse et conception des classes	12/21/18	12/24/18
Codage et developpement	12/25/18	12/27/18
Rédaction de présentation	1/21/19	1/23/19

ANN

MM - AB

A*

L’algorithme Min-Max

L'algorithme mini-max est l'un des plus anciens algorithmes d'intelligence artificielle. Il utilise une simple règle de somme nulle pour déterminer quel joueur gagnera à partir d'une position actuelle. C'est sans doute l'outil le plus puissant et le plus fondamental pour la construction de jeux en jouant à l'intelligence artificielle. Il capture les intentions de jeux à deux joueurs se disputant. Minimax est le fondement d’algorithmes tels que l’élagage alpha-bêta et l’approfondissement itératif.

L’algorithme Alpha-Beta

L'algorithme Alpha-beta(élagage alpha – bêta est un algorithme de recherche qui cherche à réduire le nombre de nœuds évalués par l'algorithme minimax ,Appliqué à un arbre minimax standard, il renvoie le même mouvement que minimax, mais élimine les branches qui ne peuvent éventuellement pas influencer la décision finale.

Alpha-Beta VS MIN-MAX

Avec Alpha-beta ,Étant donné que des valeurs égales, il vous suffira de garder la "première meilleure action rencontrée". Cependant, avec Minimax, vous explorez tout de toute façon, vous pouvez donc décider de conserver le "dernier meilleur". C'est un cas où Minimax retournerait une action différente de celle d'Alpha-Beta. Mais ils sont toujours équivalents en ce qui concerne votre fonction de notation.

JADE

Java Agent Développent Framework est un Framework open source pour le développement d'un agent intelligent, implémenté en Java.

Le système JADE prend en charge la coordination entre plusieurs agents FIPA et fournit une implémentation standard du langage de communication FIPA-ACL, ce qui facilite la communication entre agents et permet la détection de services du système.

Agent JADE

Le cycle de vie d'un agent JADE suit le cycle proposé par la FIPA, passent par différents états définis comme suit: Initié, Actif, Suspended, En attente, Supprimé et Transit.

ANN

MM - AB

A*

Agent JADE

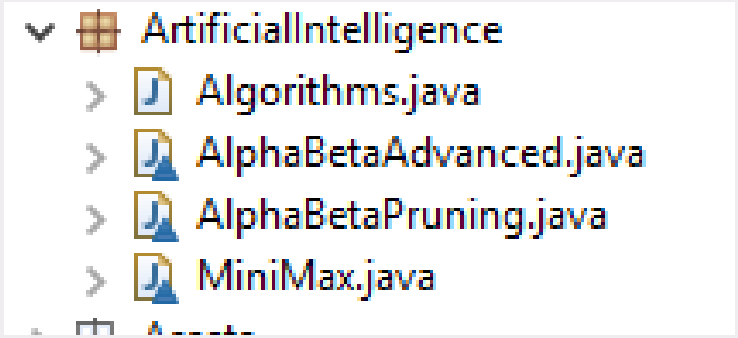
Le comportement des agents définit les actions sous un événement donné dans la configuration de la méthode à l'aide de la méthode addBehaviour. Les différents comportements que l'agent adoptera sont définis à partir de la classe abstraite Behavior. La classe Behavior contient les méthodes abstraites: action() , done (), onStart () et OnEnd().

Lorsqu'un agent est verrouillé, il peut être déverrouillé de différentes manières. Sinon, l'utilisateur peut remplacer les méthodes que possèdent l'agent onStart () et onEnd ().

Les classes utilisées

Package Artificial-Intelligence:

Algorithms.java : classe principale qui fait appel à l'algorithme choisi.



AlphabetaPruning.java : Utilise l'algorithme d'élagage alpha-bêta pour jouer un coup dans un jeu de Tic Tac Toe.

Alphabetadvanced.java : Utilise l'algorithme d'élagage alpha-bêta pour jouer un coup dans un jeu de Tic Tac Toe. mais inclut la profondeur dans la fonction d'évaluation.

MinMax.java Utilise l'algorithme MiniMax pour jouer un coup dans un jeu de Tic Tac Toe.

ANN

MM - AB

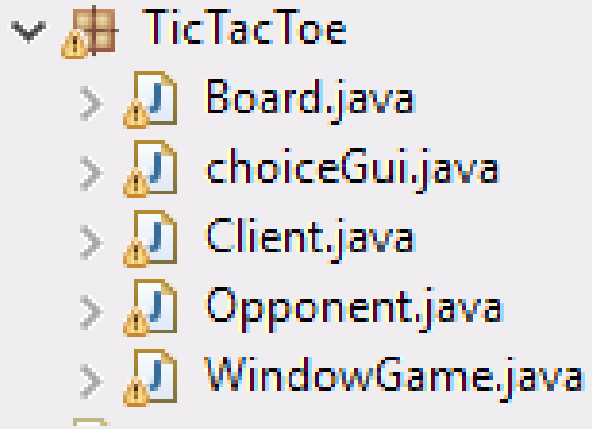
A*

Les classes utilisées

Package TicTacToe:

WindowGame.java : l'interface JFrame sur laquelle l'utilisateur va jouer.(le monde)

ChoiceGui.java : l'interface JFrame où on définis le niveau du jeu et l'algorithme utilisé pour commencer le jeu.



Client.java : agent de l'utilisateur hérite de Agent

Opponent.java : c'est l'agent IA qui joue contre l'utilisateur, hérite de Agent .

Board.java : classe hérite de JFrame qui génère les résultats du jeu (WINNER et le score)

ANN

MM - AB

A*

ANN

Démonstration

MM - AB

A*

Song Year Prediction

Réseau de neurones artificiels & DL4J

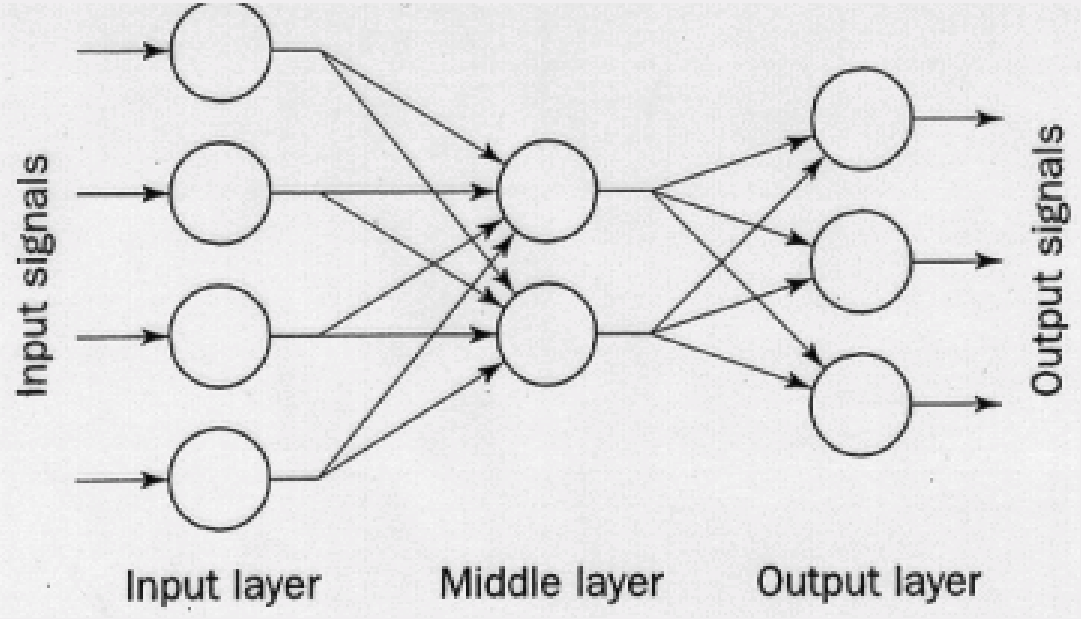
ANN

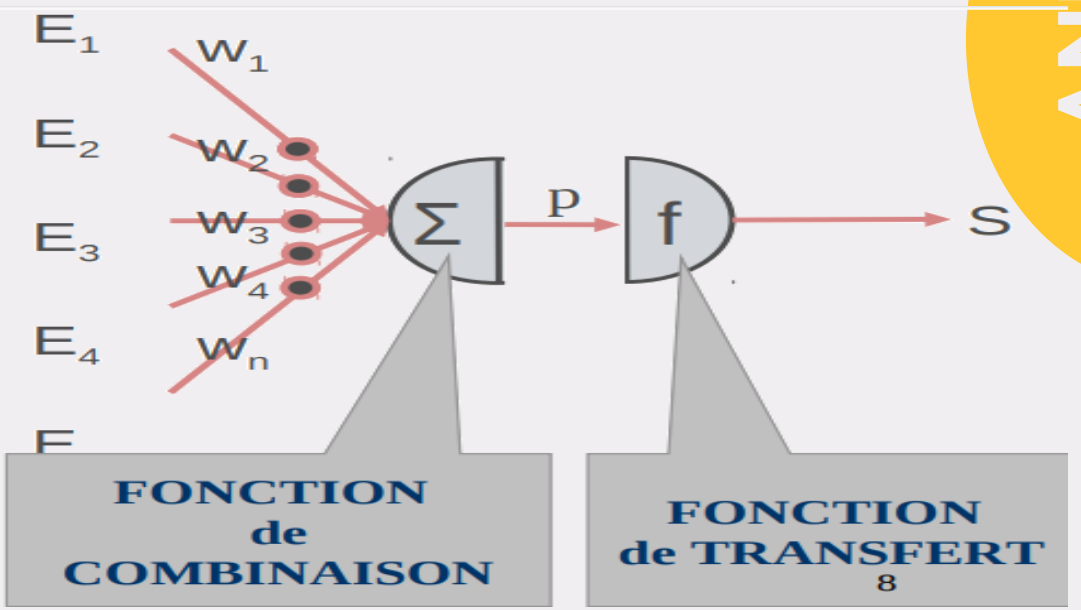
MM - AB

A*

Réseau de neurones

Un système dont la conception est à l'origine schématiquement inspirée des **neurones** biologiques.





ANN

MM - AB

A*

DL4J

Deep Learning For Java (DL4J) est une librairie open source qui permet de construire, entraîner et tester une grande diversité d'algorithmes de Deep Learning (depuis les réseaux standard, jusqu'aux réseaux convolutionels, en passant par des architectures plus complexes).

DEEPLEARNING4J

Description du program

Le program est divisé en 4 parties

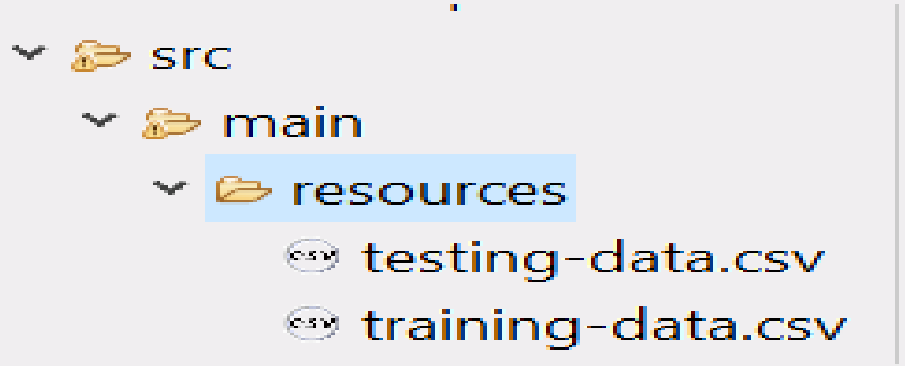
1. La lecture du fichier de données et Le changement de la structure de données de DL4J
2. La création et paramétrisat-ion du réseau de neurones
3. L'entraînement du réseau de neurones
4. L'évaluation du réseau de neurones

La lecture du fichier de données & le changement de la structure de données de DL4J

```
private static void transform(String in, String out) throws Exception {
    Schema.Builder builder = new Schema.Builder()
            .addColumnInteger("Year");

    for (int i = 0; i < 90; i++) {
        builder.addColumnDouble("feature" + i);
    }

    Schema inputDataSchema = builder.build();

    TransformProcess tp = new TransformProcess.Builder(inputDataSchema)
            .integerMathOp("Year", MathOp.Add, -1922)
            .build();
```

```
RecordReader recordReader = new CSVRecordReader();
recordReader.initialize(new FileSplit(new ClassPathResource("train-data.csv").getFile()));
```

La séparation en jeu de données d'entraînement et de test



La création et paramétrisation du réseau de neurones

```
int nEpochs = 200;
int       batchSize = 1;

 //Create the network
 int numInput = 90;
 int numOutputs = 90;
 int numHidden = 90;
 MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
         .seed(12345)
         .updater(new Nesterovs(0.01, 0.9))
         .list()
         .layer(0, new DenseLayer.Builder().nIn(numInput).nOut(numHidden)
                 .weightInit(WeightInit.XAVIER)
                 .activation(Activation.RELU)
                 .build())
         .layer(1, new OutputLayer.Builder()
                 .weightInit(WeightInit.XAVIER)
                 .activation(Activation.SOFTMAX)
                 .nIn(numHidden).nOut(numOutputs).build())
         .build();
```

L'entraînement

```
MultiLayerNetwork net = new MultiLayerNetwork(conf);
net.init();
net.setListeners(new EvaluativeListener(it,1));
```

```
Training model
Epoch 0
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
Epoch 11
```

L'évaluation

```
private static void evaluate(MultiLayerNetwork net,DataNormalization normalizer) throws Exception {
    RecordReader recordReader = new CSVRecordReader();
    recordReader.initialize(new FileSplit(new ClassPathResource("testing-data.csv").getFile()));

    DataSetIterator it = new RecordReaderDataSetIterator(recordReader, batchSize, 0, 90);

    it.setPreProcessor(normalizer);


    System.out.println("Evaluate model....");
    Evaluation eval = new Evaluation(90); // 90 number outputs
    while(it.hasNext()){
        DataSet t = it.next();
        INDArray features = t.getFeatures();
        INDArray lables = t.getLabels();
        INDArray predicted = net.output(features,false);

        eval.eval(lables, predicted);
    }

    System.out.println(eval.stats());
}
```