Using a neural network on "Coders Strike Back"

because why not

By bleuj

16 June 2016

Contents

Contents

What we try to achieve here

A light decent runner..

- Supervised learning on a good runner bot using a neural network.

What we try to achieve here

A light decent runner..

- Supervised learning on a good runner bot using a neural network.
- Constrained memory use of the neural network.

What we try to achieve here

A light decent runner..

- Supervised learning on a good runner bot using a neural network.
- Constrained memory use of the neural network.
- Low time cost with the neural network bot (to make all of this slightly less useless).

Contents

Machine learning problem

- Given a data set of inputs and associated outputs, machine learning and statistics techniques can help to generalize a data set.

Machine learning problem

- Given a data set of inputs and associated outputs, machine learning and statistics techniques can help to generalize a data set.
- Here we will try to imitate a bot.

Machine learning problem

- Given a data set of inputs and associated outputs, machine learning and statistics techniques can help to generalize a data set.
- Here we will try to imitate a bot.
- We must choose inputs and outputs to imitate a reference bot.

Machine learning problem

- Given a data set of inputs and associated outputs, machine learning and statistics techniques can help to generalize a data set.
- Here we will try to imitate a bot.
- We must choose inputs and outputs to imitate a reference bot.
- Then the neural network will be a function to generalize the data gathered about the reference bot.

Machine learning problem

- Given a data set of inputs and associated outputs, machine learning and statistics techniques can help to generalize a data set.
- Here we will try to imitate a bot.
- We must choose inputs and outputs to imitate a reference bot.
- Then the neural network will be a function to generalize the data gathered about the reference bot.
- Neural networks are one of many methods to do that.

Contents

Main parts of the work

- 1 - Coding a good normal bot.

Main parts of the work

- 1 - Coding a good normal bot.
- 2 - Using a simulator to generate data to learn.

Main parts of the work

- 1 - Coding a good normal bot.
- 2 - Using a simulator to generate data to learn.
- 3 - Learning the data on a neural network.

Main parts of the work

- 1 - Coding a good normal bot.
- 2 - Using a simulator to generate data to learn.
- 3 - Learning the data on a neural network.
- 4 - Importing the neural network on CodinGame.

Contents

Reference bot

- Bot that minimizes the distance to run, with a depth of 8.

Reference bot

- Bot that minimizes the distance to run, with a depth of 8.
- Minimization done with simulated annealing heuristic.

Reference bot

- Bot that minimizes the distance to run, with a depth of 8.
- Minimization done with simulated annealing heuristic.
- Extra points for reaching checkpoints early, for having some speed, and good angles at the end.

Reference bot

- Bot that minimizes the distance to run, with a depth of 8.
- Minimization done with simulated annealing heuristic.
- Extra points for reaching checkpoints early, for having some speed, and good angles at the end.
- Does not care about the opponent.

Contents

Simulating things and creating data outside of CodinGame

- Visualization to verify your bot seems correct outside of CodinGame (I use Qt).

Simulating things and creating data outside of CodinGame

- Visualization to verify your bot seems correct outside of CodinGame (I use Qt).
- Lots of simulations are run with random checkpoint positions, random speed and angles (speed and pod angles), picked out data from there.

Simulating things and creating data outside of CodinGame

- Visualization to verify your bot seems correct outside of CodinGame (I use Qt).
- Lots of simulations are run with random checkpoint positions, random speed and angles (speed and pod angles), picked out data from there.
- Data saved with C++ ofstream.

About inputs and outputs

- The inputs should be invariant by translation and rotation, because it doesn't restrict you, and otherwise there's useless extra difficulty to learn.

About inputs and outputs

- The inputs should be invariant by translation and rotation, because it doesn't restrict you, and otherwise there's useless extra difficulty to learn.
- The inputs must take into account at least the 2 next checkpoints.

About inputs and outputs

- The inputs should be invariant by translation and rotation, because it doesn't restrict you, and otherwise there's useless extra difficulty to learn.
- The inputs must take into account at least the 2 next checkpoints.
- Outputs are simply : chosen angle difference, and thrust choice.

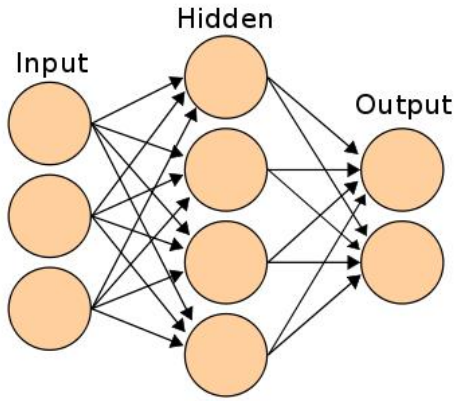
About inputs and outputs

- The inputs should be invariant by translation and rotation, because it doesn't restrict you, and otherwise there's useless extra difficulty to learn.
- The inputs must take into account at least the 2 next checkpoints.
- Outputs are simply : chosen angle difference, and thrust choice.
- Normalizing everything (for example between 0 and 1) helps a lot the learning part of the Neural network.

Contents

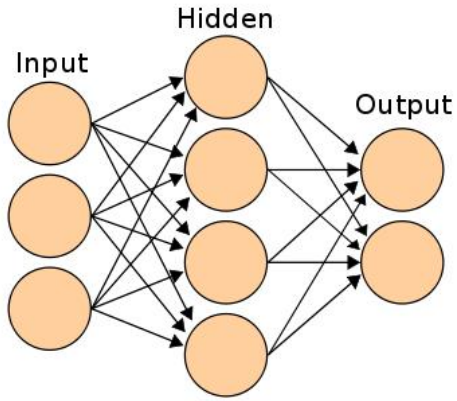
Neural network architecture

- The function defined by a neural network is easy to understand with a picture.



Neural network architecture

- The function defined by a neural network is easy to understand with a picture.
- Values go from inputs to outputs as follows : when you are on an arrow you are multiplied by a weight, on a neuron the value is the sum of the values that arrive. When you leave a hidden neuron, you apply a non linear function (for example tanh).



Neural network architecture

- I only use one hidden layer.

Neural network architecture

- I only use one hidden layer.
- I use bias (neurons always equal to one).

Neural network architecture

- I only use one hidden layer.
- I use bias (neurons always equal to one).
- The hidden layer is a rectified linear unit (ReLU) layer (Found its implementation on stack overflow).

Neural network architecture

- I only use one hidden layer.
- I use bias (neurons always equal to one).
- The hidden layer is a rectified linear unit (ReLU) layer (Found its implementation on stack overflow).
- Data saved with C++ ofstream.

Contents

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.
- There are lots of libraries to do that.

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.
- There are lots of libraries to do that.
- I use PyBrain on Python (first one that worked for me).

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.
- There are lots of libraries to do that.
- I use PyBrain on Python (first one that worked for me).
- It's very simple to use after reading the short tutorial on its website, but quite slow.

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.
- There are lots of libraries to do that.
- I use PyBrain on Python (first one that worked for me).
- It's very simple to use after reading the short tutorial on its website, but quite slow.
- Data produced with C++ code is opened with file = open('mydata.txt','r') and file.readline().

Learning a neural network function

- A neural network function relies on a lot of parameters (weights), and you have to optimize them so that it best fits a data set.
- It's quite tricky to optimize (back-propagation, stochastic gradient descent...), especially if you don't want to change all your code if you add a layer or something like that.
- There are lots of libraries to do that.
- I use PyBrain on Python (first one that worked for me).
- It's very simple to use after reading the short tutorial on its website, but quite slow.
- Data produced with C++ code is opened with file = open('mydata.txt','r') and file.readline().
- Saving the weights was a little tricky with PyBrain (a post on stack-overflow helped me).

Contents

Importing the neural network on CodinGame

- The functions to create the inputs of the data set are used again here.

Importing the neural network on CodinGame

- The functions to create the inputs of the data set are used again here.
- Weights arrays are read from 4 very ugly 1-dimensional arrays that are created by the python code, and that are just copy-pasted every time.

Importing the neural network on CodinGame

- The functions to create the inputs of the data set are used again here.
- Weights arrays are read from 4 very ugly 1-dimensional arrays that are created by the python code, and that are just copy-pasted every time.
- New arrays are something more understandable.

Importing the neural network on CodinGame

- The functions to create the inputs of the data set are used again here.
- Weights arrays are read from 4 very ugly 1-dimensional arrays that are created by the python code, and that are just copy-pasted every time.
- New arrays are something more understandable.
- Carefully implemented neural network on CodinGame using the definition/architecture of the neural network.

Contents

Checking the result

- Now we have a bot ready to compete with others

Checking the result

- Now we have a bot ready to compete with others
- The bot may fail to cross many checkpoints, so it can be a good idea to reduce checkpoint radius in simulation.

Checking the result

- Now we have a bot ready to compete with others
- The bot may fail to cross many checkpoints, so it can be a good idea to reduce checkpoint radius in simulation.
- Another trick is to exaggerate thrust and angle difference given by the neural network.