Neural RRT*

Wednesday, August 6, 2025 11:06 AM

Core Idea

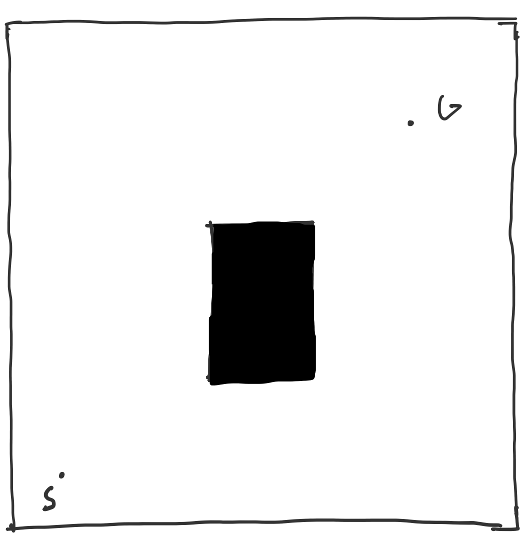
Neural RRT* is Neural Network + RRT*

- RRT* randomly samples points to build its tree
- if we are smart about where we sample, RRT* becomes much better
- if we are smart about which connections to explore, RRT* can converge to solution / optimal solution faster

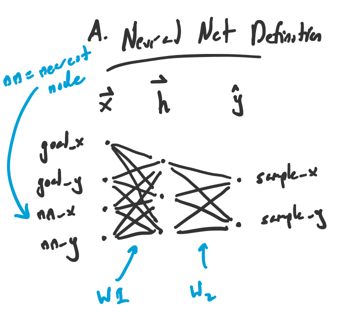
The Neural Net!

Step 1 - Initialization

Start = 1,1 | Goal = 8,8



A. Neural Net Definition



We define a set of vectors to store weights

W1: input → hidden

W2: hidden → output

b1: hidden bias

b2: output bias

B. Randomly Initialize weights

W1 = [[.5, -.3, .2]
[.1, .4, -.1]
[-.2, .3, .6]
[.4, -.2, .1]]

W2 = [[.8, .2]
[-.1, .7]
[.3, -.4]]

b1 = [.1, -.2, .3]

b2 = [0, 0]

C. Define the forward pass eqns

(matrix indices in weight vector)

$h_1 = relu(W_1[0,0] \cdot x_1 + W_1[1,0] \cdot x_2 + W_1[2,0] \cdot x_3 + W_1[3,0] \cdot x_4 + b_1[0])$

$h_2 = relu(W_1[0,1] \cdot x_1 + W_1[1,1] \cdot x_2 + W_1[2,1] \cdot x_3 + W_1[3,1] \cdot x_4 + b_1[1])$

$h_3 = relu(W_1[0,2] \cdot x_1 + W_1[1,2] \cdot x_2 + W_1[2,2] \cdot x_3 + W_1[3,2] \cdot x_4 + b_1[2])$

* $\forall h_i$ if < 0, then set $= 0$

$y_1 = W_2[0,0] \cdot h_1 + W_2[1,0] \cdot h_2 + W_2[2,0] \cdot h_3 + b_2[0]$

$y_2 = W_2[0,1] \cdot h_1 + W_2[1,1] \cdot h_2 + W_2[2,1] \cdot h_3 + b_2[1]$

where

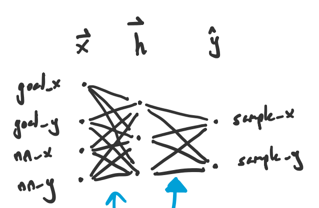
$\vec{x} = [x_1, x_2, x_3, x_4]$

$\hat{\vec{y}} = [y_1, y_2]$

D. Define Gradient Descent Functions

$L = \frac{1}{2}[(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2]$ ← basic MSE

$\frac{\partial L}{\partial y_1} = \hat{y}_1 - y_1 \quad \frac{\partial L}{\partial y_2} = \hat{y}_2 - y_2$



$\frac{\partial L}{\partial W_2[i,0]} = (\hat{y}_1 - y_1) \cdot h_{i+1}$

$\frac{\partial L}{\partial W_2[i,1]} = (\hat{y}_1 - y_1) \cdot h_{i+1}$

$\frac{\partial L}{\partial b_2[0]} = \hat{y}_1 - y_1 \quad \frac{\partial L}{\partial b_2[1]} = \hat{y}_2 - y_2$

$\frac{\partial L}{\partial h_1} = (\hat{y}_1 - y_1) \cdot W_2[0,0] + (\hat{y}_2 - y_2) \cdot W_2[0,1]$

$\frac{\partial L}{\partial h_2} = (\hat{y}_1 - y_1) \cdot W_2[1,0] + (\hat{y}_2 - y_2) \cdot W[1,1]$

$\frac{\partial L}{\partial h_3} = (\hat{y}_1 - y_1) \cdot W_2[2,0] + (\hat{y}_2 - y_2) \cdot W_2[2,1]$

$\frac{\partial L}{\partial W_1[i,j]} = \frac{\partial L}{\partial h_{j+1}} \cdot x_{i+1}$

$\frac{\partial L}{\partial b_1[i]} = \frac{\partial L}{\partial h_{i+1}}$

E. Weight update rules

$W_2[i,j] = W_2[i,j] - \alpha \cdot \frac{\partial L}{\partial W_2[i,j]}$

$W_1[i,j] = W_1[i,j] - \alpha \cdot \frac{\partial L}{\partial W_1[i,j]}$

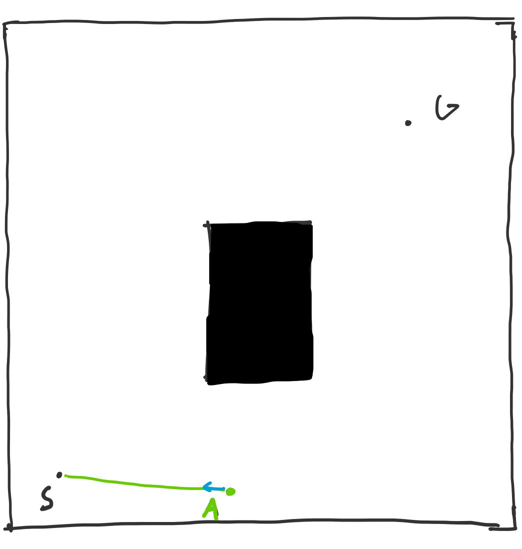
$b_2[i] = b_2[i] - \alpha \cdot \frac{\partial L}{\partial b_2[i]}$

$b_1[i] = b_1[i] - \alpha \cdot \frac{\partial L}{\partial b_1[i]}$

Step 2: Iteration 1

A. Analyze Tree

- Coverage - bottom left of grid
- 1 node (1,1)
- Distance to Goal: $d = \sqrt{(8-1)^2 + (8-1)^2} = 9.9$



B. Determine NN input

[goal_x, goal_y, nn_x, nn_y]
↓
[8, 8, 1, 1]

C. NN forward pass

Using calculator...

$\hat{y} = [4.55, .79]$ ← A

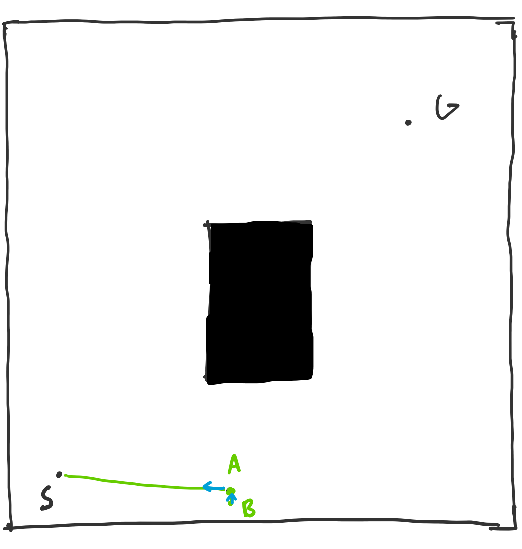
D. Run RRT*

- Query point - [A]
- Candidate Nodes - [S]
- $d = \sqrt{(4.55-1)^2 + (.79-1)^2} = 3.56$
- [S(3.56)] → so S is closest
- Check for collision → NO collision ✓

Step 3 - Iteration 2

A. Analyze Tree

- Nodes: 2 [A(7.99), S(9.90)] (Distance to goal)



B. NN input

- Use closest Node for expansion (A)

[8, 8, 4.55, .79]

C. NN Forward Pass

$\hat{y} = [4.487, .562]$

* Gut check - this is a bad guess!

D. Run RRT*

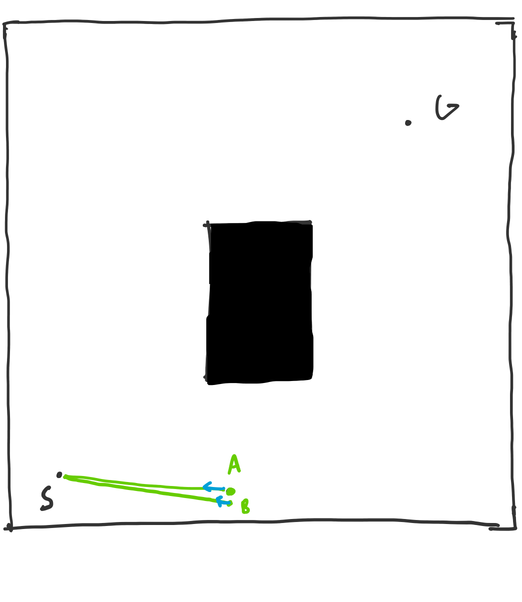
- Find Nearest Node → A
  [A(.26), S(3.46)]
- Collision check → NO collision

E. RRT* rewiring

- would connecting (4.437, .562) via other nodes be more optimal?

- Via (1,1)
  d = 0 + 3.46 = 3.46 ← Lower! Rewire

- via (4.55, .79) (current)
  d = 3.56 + .26 = 3.82



F. Performance Assessment

[A(7.99), B(8.25), S(9.9)]
(B: Further)

NN will not be able to reach goal...

Step 4 - Training

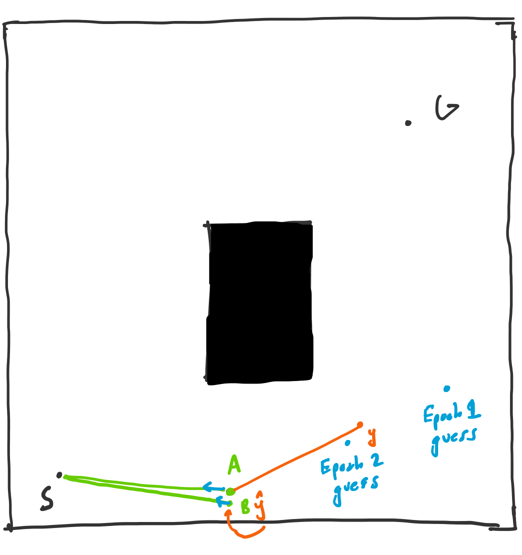
A. Collect Bad suggestions

Training Sample:

input state: [8, 8, 4.55, .79]

$\hat{y} = [4.437, .562]$

$y = [7, 2]$ we chose this ground truth



B. Calculate Loss

$L = \frac{1}{2}[(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2]$

$L = 4.318$

C. Backprop

$\alpha = .01$

This isn't a backprop tutorial, so... after the updates we re-run the same point

Initial prediction: [4.437, .562]

Epoch 1: [8.665, 2.745] closer

Epoch 2: [6.89, 1.78] VERY close!

Step 5 - Repeat

- It's clear now how this works
- once the net is trained, it will react appropriately to the given situation it's in