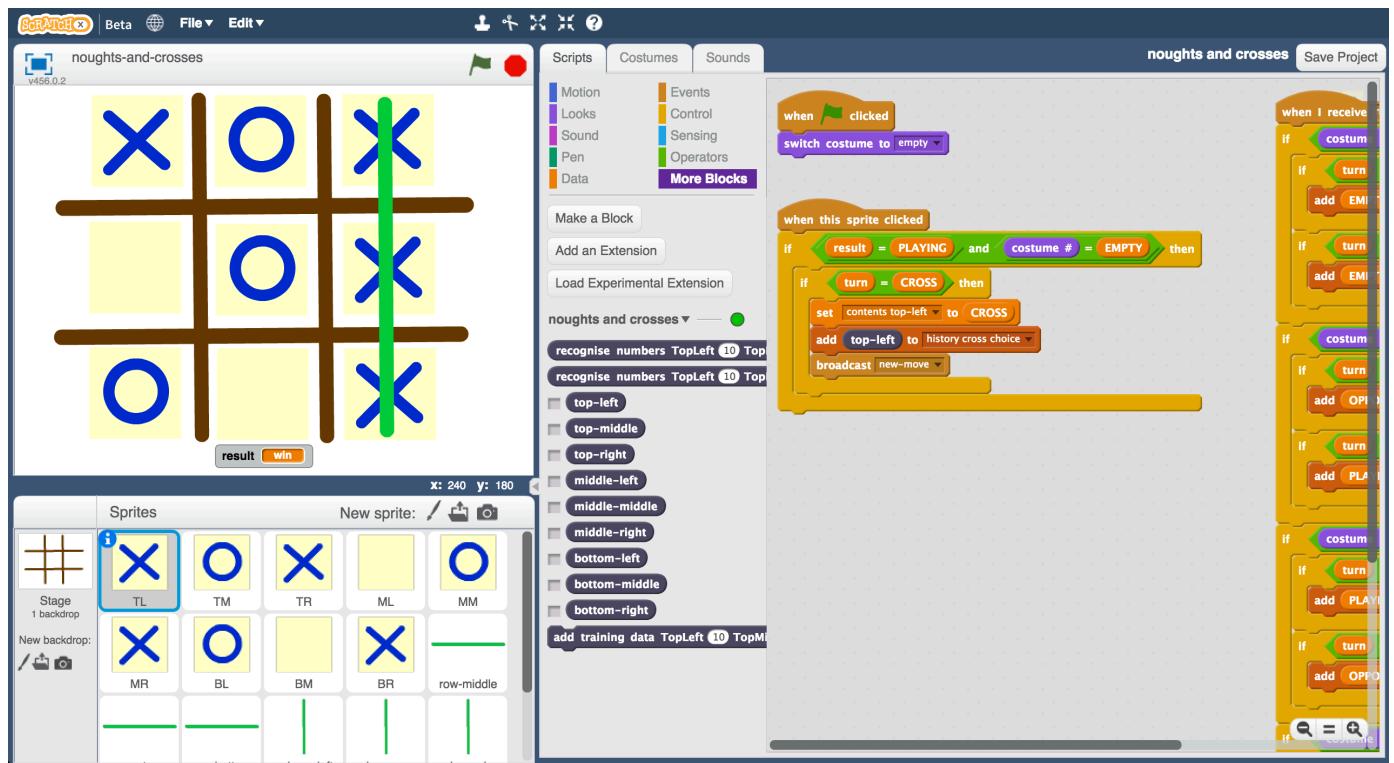


Noughts and Crosses

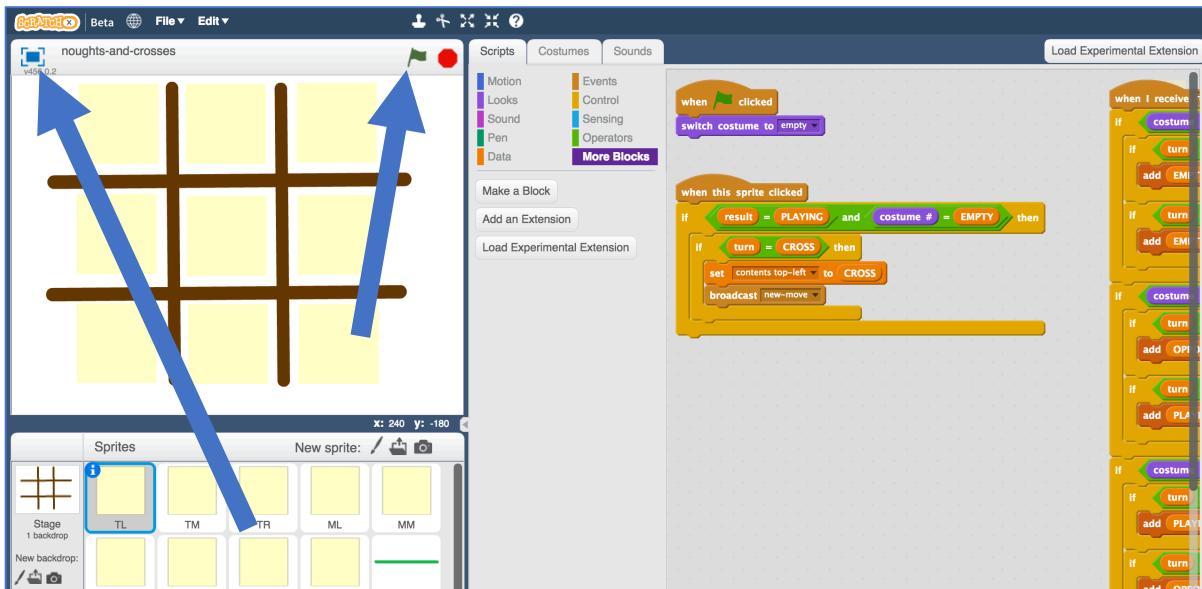
In this project you will create a noughts and crosses game in Scratch that is able to learn from how you play.

You won't give it instructions for how to play, or tell it what the objective or rules of the game are.

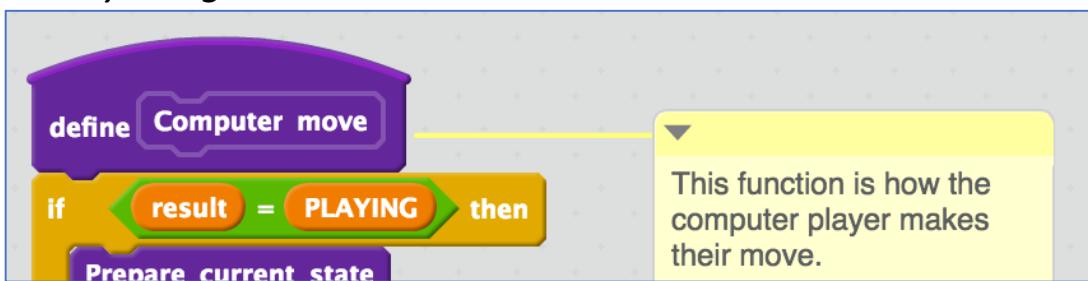
Instead, you'll show it examples of you playing the game. When it's seen enough examples to start trying to play for itself, you'll tell it when it beats you.



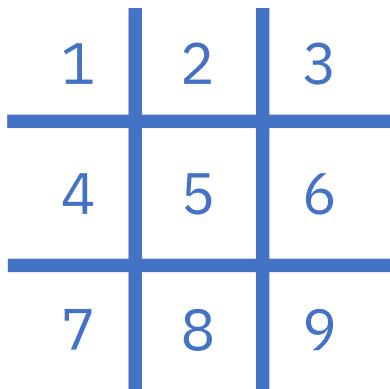
1. Go to <https://machinelearningforkids.co.uk/scratchx> in a browser.
2. Open the **noughts-and-crosses.sbx** starter file for this project.
Click File -> Load Project
If you haven't got this file, ask your teacher or group leader.
3. Click the **full-screen** button, and then click the Green Flag



4. Play a few games of noughts and crosses
You are CROSS (X), the computer is playing as NOUGHTS (O).
Click the green flag to start a new game, then click on the game board.
5. Can you see how the computer is choosing where to put its moves?
When you think you've worked out the computer's strategy, look at the Computer move block in the Stage
Were you right?

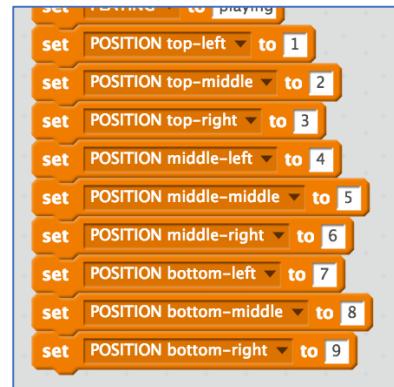


Representing noughts and crosses in Scratch



The positions of spaces on the noughts and crosses board are numbered from 1 to 9.

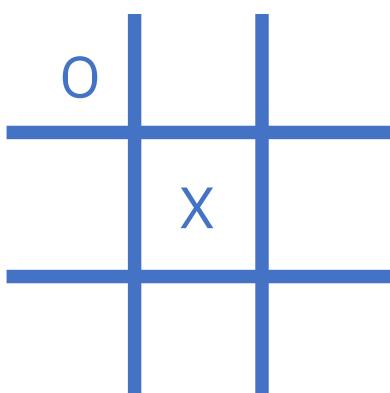
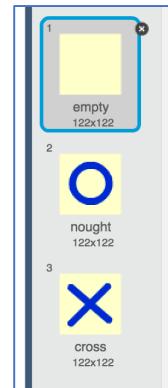
Data constants are used to make it easier to refer to them in scripts.



Empty = 1
O = 2
X = 3

An empty space is shown in costume 1.
A nought is shown in costume 2.
A cross is shown in costume 3.

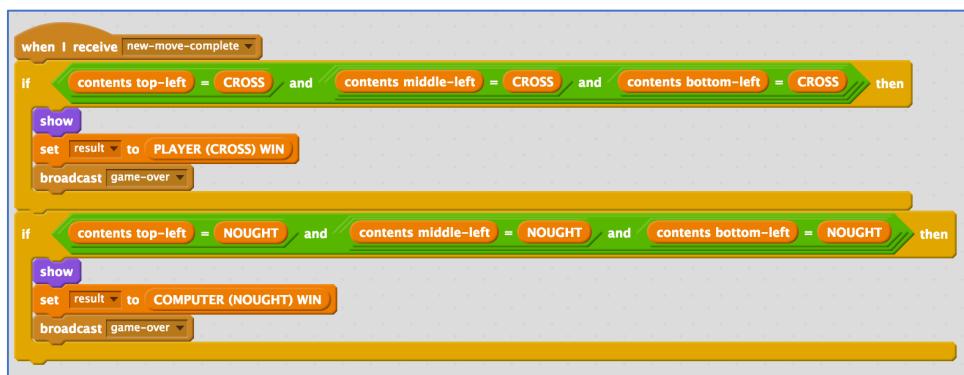
Data constants are used to make it easier to refer to these in scripts.



Variables are used to store the current state of the game.

For example, at this point:

contents top-left = 2
contents middle-middle = 3
contents bottom-right = 1



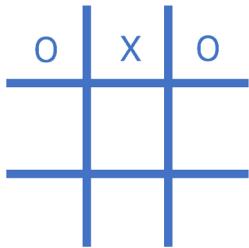
Each of the green row and column sprites check to see if someone has won.

This happens after every move.

What are you going to do?

You're going to train a computer to play noughts and crosses. You'll do this by showing it examples of how you play the game.

Imagine the board looks like this and it's X's turn.

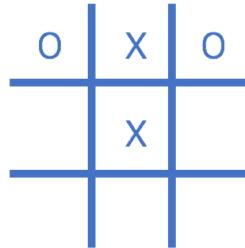


Imagine you decide to put your X in the centre space.

top-left	opponent
top-middle	player
top-right	opponent
middle-left	empty
middle-middle	empty
middle-right	empty
bottom-left	empty
bottom-middle	empty
bottom-right	empty

choice : middle-middle

Imagine the board looks like this and it's O's turn.



Imagine you decide to put your O in the bottom middle space.

top-left	player
top-middle	opponent
top-right	player
middle-left	empty
middle-middle	opponent
middle-right	empty
bottom-left	empty
bottom-middle	empty
bottom-right	empty

choice : bottom-middle

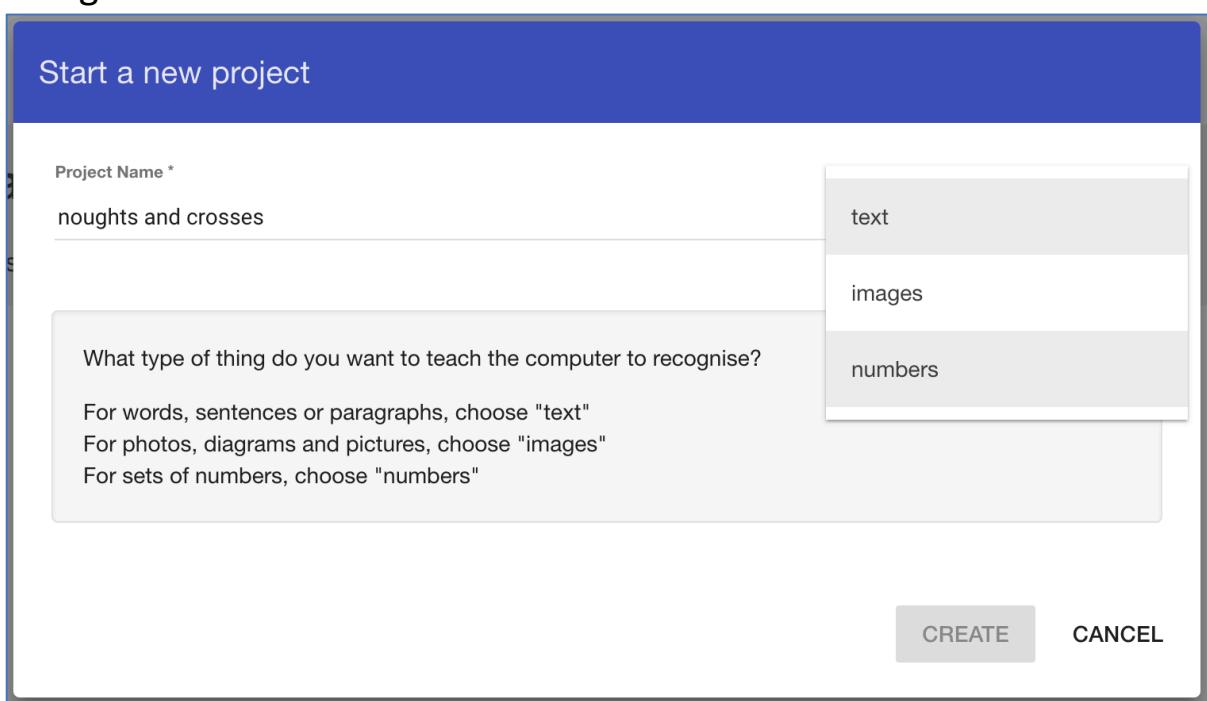
Using “opponent” and “player” instead of “nought” and “cross” means the computer can learn from both nought and cross moves.

You'll only use examples of moves from the player that wins the game.

If you (X) win, use your moves as examples to train the computer.
If the computer (O) wins, use the computer's moves to train with.

These **examples of moves that lead to winning** will teach the computer how to play to win!

- 6.** Close the Scratch window.
- 7.** Go to <https://machinelearningforkids.co.uk/> in a web browser
- 8.** Click on “**Get started**”
- 9.** Click on “**Log In**” and type in your username and password
If you don't have a username, ask your teacher or group leader to create one for you.
If you can't remember your username or password, ask your teacher or group leader to reset it for you.
- 10.** Click on “**Projects**” on the top menu bar
- 11.** Click on the “**+ Add a new project**” button.
- 12.** Name your project “noughts and crosses” and set it to learn how to recognise “**numbers**”



- 13.** Set “**how many numbers in each example**” to 9

14. Name the fields after the positions on the board

Each example is the state of the board before a move that led to a win.

*TopLeft, TopMiddle, TopRight, MiddleLeft, MiddleMiddl, MiddleRight,
BottomLeft, BottomMiddl, BottomRight*

(Two of these names have a letter missing off the end because they're too long!)

Start a new project

Project Name *	Recognizing *	
noughts and crosses	numbers	
How many numbers in each example?		
9		
Field 1 name *	Field 2 name *	Field 3 name *
TopLeft	TopMiddle	TopRight
Field 4 name *	Field 5 name *	Field 6 name *
MiddleLeft	MiddleMiddl	MiddleRight
Field 7 name *	Field 8 name *	Field 9 name *
BottomLeft	BottomMiddl	BottomRight

Each example for numbers projects can contain a set of numbers.
For example, if you want to train the computer to recognise the time of year based on weather data, you might want a few numbers - for

CREATE **CANCEL**

15. Click **Create**. You should see “noughts and crosses” show up in the list of your projects. Click on it.

ml-for-kids Welcome About Projects Worksheets News Help Log Out

Your machine learning projects

+ Add a new project

newspapers Recognising text as Daily_Express, Daily_Mirror or 2 other classes	
noughts and crosses Recognising numbers	

16. Click on Train

The screenshot shows a web interface for a machine learning project titled "noughts and crosses". At the top, there is a navigation bar with links: ml-for-kids, Welcome, About, Projects, Worksheets, News, Help, and Log Out. Below the title, there are three main options: "Train", "Learn & Test", and "Scratch". The "Train" option is highlighted with a blue button labeled "Train". A descriptive text below it says: "Collect examples of what you want the computer to recognise." The "Learn & Test" option has a blue button labeled "Learn & Test" and a description: "Use the examples to train the computer to recognise numbers." The "Scratch" option has a blue button labeled "Scratch" and a description: "Use the machine learning model you've trained to make a game in Scratch."

17. Click “+ Add new label” and create a label called “top-left”

Examples of making a move in the top-left space that leads to a win will go in this bucket.

The screenshot shows a web interface for a machine learning project titled "Recognising numbers as top-left". At the top, there is a navigation bar with links: ml-for-kids, Welcome, About, Projects, Worksheets, News, Help, and Log Out. Below the title, there is a link "< Back to project". On the right side, there is a button labeled "+ Add new label". In the center, there is a large rectangular area with a dark border. Inside this area, the text "top-left" is written in green. At the bottom left of this area, there is a small button labeled "+ Add example".

18. Click “+ Add new label” again and create labels for the other eight spaces on the board.

top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, bottom-right

The screenshot shows a web-based application for training a machine learning model. At the top, there's a navigation bar with links for 'ml-for-kids', 'Welcome', 'About', 'Projects', 'Worksheets', 'News', 'Help', and 'Log Out'. Below the navigation, the title 'Recognising numbers as top-left, top-middle or 7 other classes' is displayed. Underneath the title, there are nine rectangular boxes arranged in a 3x3 grid, each containing a label and a '+ Add example' button. The labels are: 'top-left', 'top-middle', 'top-right' in the first row; 'middle-left', 'middle-middle', 'middle-right' in the second row; and 'bottom-left', 'bottom-middle', 'bottom-right' in the third row. A small 'Add new label' button is located in the top right corner of the main content area.

19. Click the “< Back to project” link then click **Scratch**

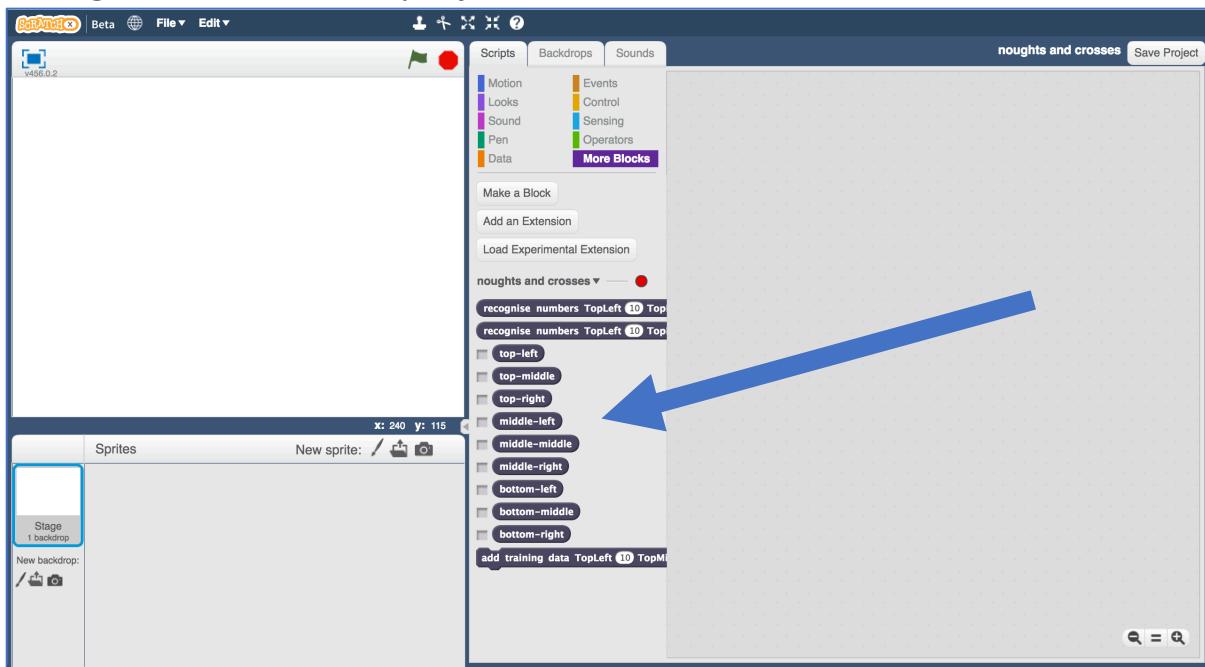
20. Click the **Open in Scratch** button

It will warn you that you haven't trained the computer yet – but that's okay, as you'll use Scratch to collect the training examples.

Click the “go straight into Scratch now” link.

The screenshot shows the 'More Blocks' tab in the Scratch script editor. On the left, there's a text box explaining that the project adds blocks to the 'More Blocks' tab in the Scripts category. It describes two main blocks: 'recognise numbers' which takes an input number and returns a label, and 'confidence' which provides a confidence score from 0 to 100. Below this, a list of labels ('top-left', 'top-middle', etc.) is shown. A note says these blocks represent the labels created in the project. A sample script is shown with a green dot next to the project name 'make me happy', indicating the model is trained. On the right, a preview window shows the Scratch stage with a green dot next to the project name. A legend at the bottom explains the color coding: a green dot means the model is trained, a yellow dot means it hasn't finished training, and a red dot means there was an error during training.

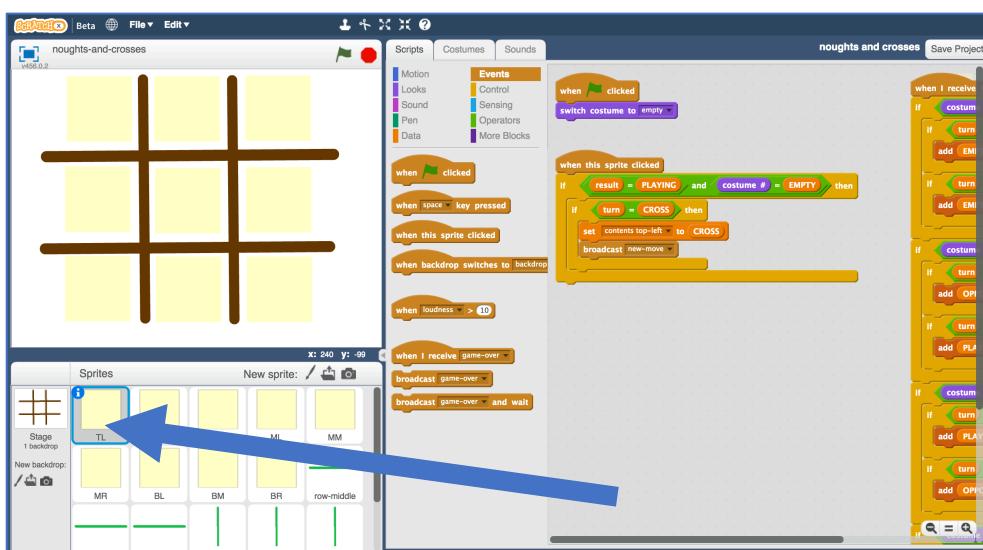
21. You should see new blocks in the “More blocks” section from your “noughts and crosses” project.



22. Open the “noughts-and-crosses.sbx” starter project file again.
*Click **File** -> **Load Project***

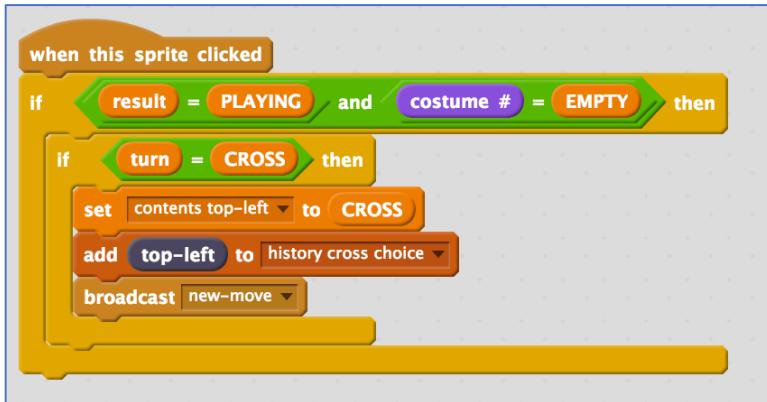
23. Click the **TL** (top-left space) sprite

You'll update the script to store when you click on this space for your move.



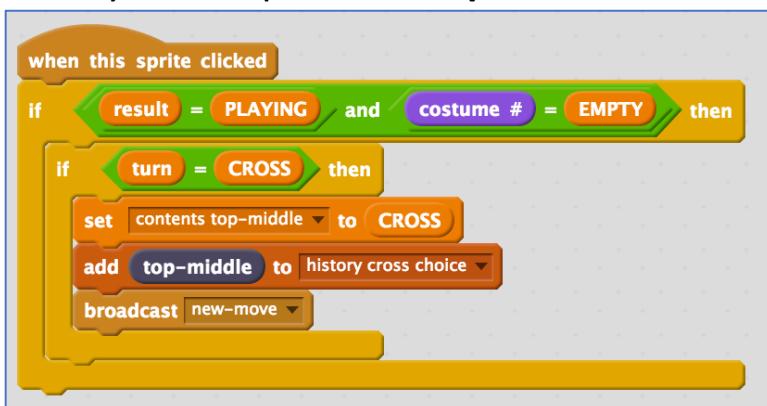
24. Modify the **When this sprite clicked** script to add **top-left** to the history of cross choices.

You only need to add a single block to get the script to look like this:



25. Click the **TM** (top-middle) sprite.

Modify the script to add **top-middle** to the history of cross choices.



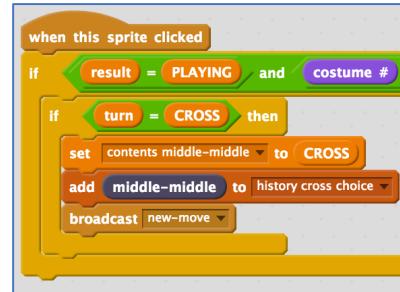
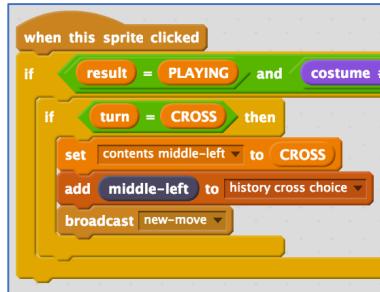
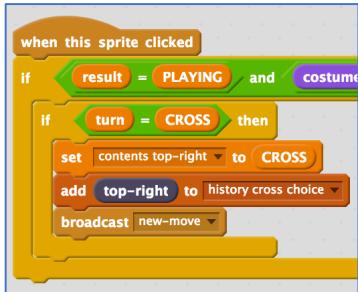
26. Repeat for the other seven board sprites

Add **top-right** to the **TR** sprite. Add **middle-left** to the **ML** sprite.

Add **middle-middle** to the **MM** sprite. Add **middle-right** to the **MR** sprite.

Add **bottom-left** to the **BL** sprite. Add **bottom-middle** to the **BM** sprite.

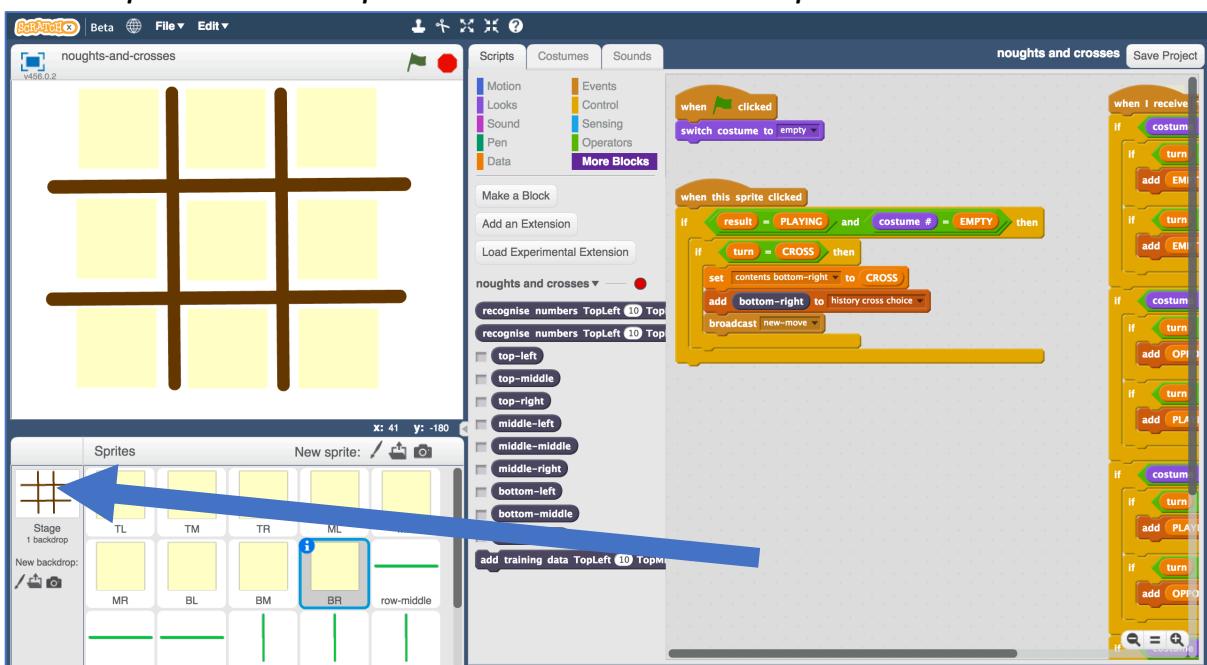
Add **bottom-right** to the **BR** sprite.



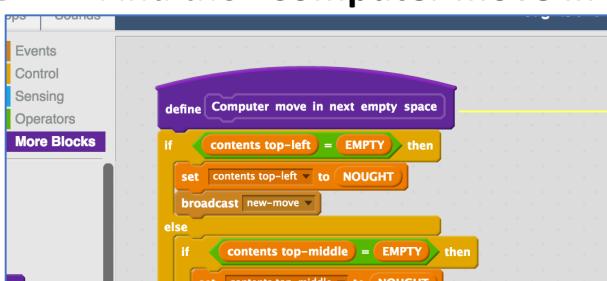


27. Click on the “Stage”

You'll update the script to store when the computer makes it's move



28. Find the “Computer move in next empty space” script



29. Modify the script to add each of the computer's moves to the history of nought choices.

*You only need to add the **add CHOICE** to 'history nought choice' blocks*

```

define Computer move in next empty space
  if contents top-left = EMPTY then
    set contents top-left to NOUGHT
    add top-left to history nought choice
    broadcast new-move
  else
    if contents top-middle = EMPTY then
      set contents top-middle to NOUGHT
      add top-middle to history nought choice
      broadcast new-move
    else
      if contents top-right = EMPTY then
        set contents top-right to NOUGHT
        add top-right to history nought choice
        broadcast new-move
      else
        if contents middle-left = EMPTY then
          set contents middle-left to NOUGHT
          add middle-left to history nought choice
          broadcast new-move
        else
          if contents middle-middle = EMPTY then
            set contents middle-middle to NOUGHT
            add middle-middle to history nought choice
            broadcast new-move
          else
            if contents middle-right = EMPTY then
              set contents middle-right to NOUGHT
              add middle-right to history nought choice
              broadcast new-move
            else
              broadcast new-move
            end
          end
        end
      end
    end
  end
end
else
  if contents bottom-left = EMPTY then
    set contents bottom-left to NOUGHT
    add bottom-left to history nought choice
    broadcast new-move
  else
    if contents bottom-middle = EMPTY then
      set contents bottom-middle to NOUGHT
      add bottom-middle to history nought choice
      broadcast new-move
    else
      if contents bottom-right = EMPTY then
        set contents bottom-right to NOUGHT
        add bottom-right to history nought choice
        broadcast new-move
      else
        set result to DRAW
        broadcast game-over
      end
    end
  end
end
end

```

The Scratch script defines a function 'Computer move in next empty space'. It iterates through all nine positions in a 3x3 grid. For each position, it checks if it is empty ('contents' followed by position = 'EMPTY'). If so, it sets the position to 'NOUGHT', adds it to a list named 'history nought choice', and broadcasts 'new-move'. If none of the positions are empty, it broadcasts 'new-move' directly. After the function ends, the script continues with an 'else' block. This block checks the bottom row for an empty position. If found, it sets the position to 'NOUGHT', adds it to the history, and broadcasts 'new-move'. If no empty position is found in the bottom row, it sets the 'result' variable to 'DRAW' and broadcasts 'game-over'.

30. Create the following script (still in the Stage)

This will add all of the history of moves made by the winning player to the training data that you will use to train the computer.



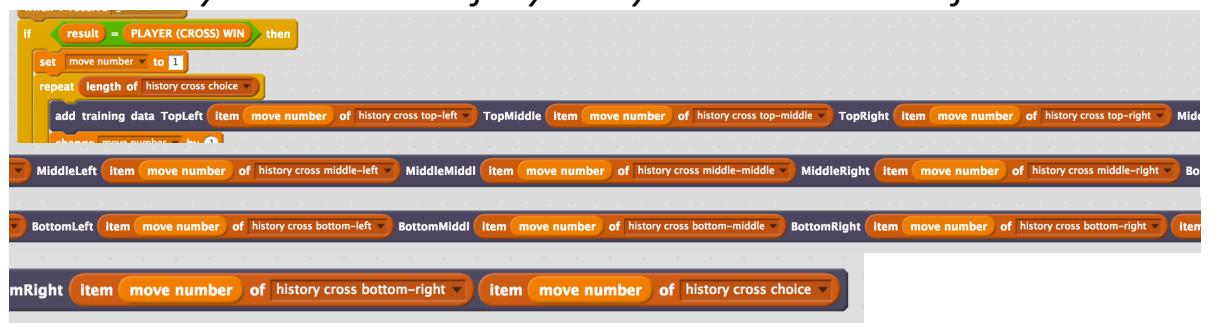
*If the game ends with the player (CROSS) winning, all the moves you want to add to the training data should be from the **cross** histories.*

*If the game ends with the computer (NOUGHT) winning, all the moves you want to add to the training data should be from the **nought** histories.*

The history item you should add should match the board state space (e.g. add top-left to the TopLeft item).

And you end with the choice of move that was made.

The 'add training data' block is very long! Duplicate can help save time. Make sure you do this carefully – any mistakes will confuse the training.





31. Save your project

Click File -> Save Project

32. Play a few games

Click on the Green Flag as you did before.

It is better to play in full-screen mode to avoid accidentally moving sprites.

33. Go back to the training page

Leaving the Scratch window open, go back to the training tool window.

Click the “< Back to project” link and then click “Train”

Category	Example 1	Example 2	Example 3
top-left	TopLeft 0 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 0 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 0 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0
middle-left			
bottom-left	TopLeft 0 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 0 TopRight 1 MiddleLeft -1 MiddleMiddle 1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 0 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0
top-middle	TopLeft 1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle -1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight -1	TopLeft 1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 1 BottomMiddle 0 BottomRight 0
middle-middle	TopLeft -1 TopMiddle 1 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 1 BottomMiddle 1 BottomRight 0	TopLeft -1 TopMiddle 1 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0
bottom-middle	TopLeft 0 TopMiddle 0 TopRight -1 MiddleLeft 0 MiddleMiddle 1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 0 TopMiddle 0 TopRight -1 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 1 TopRight 0 MiddleLeft 0 MiddleMiddle 1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0
top-right	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 1 BottomRight 0	TopLeft 1 TopMiddle -1 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 1 BottomRight 0
middle-right	TopLeft -1 TopMiddle 0 TopRight 1 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0
bottom-right	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 1 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft 1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0	TopLeft -1 TopMiddle 0 TopRight 0 MiddleLeft 0 MiddleMiddle 0 MiddleRight 0 BottomLeft 0 BottomMiddle 0 BottomRight 0

34. Look at your training so far

Each item is the state of the board at the time you or the computer made a move (in a game that you or the computer won).

The bucket that the item is in is the move that you made.

If there are any empty buckets – like middle-left in the screenshot before – that means you haven't made a move in that space in a game that you won yet.

35. Click the “< Back to project” link. Click the “Learn & Test” button.

36. If there is a “Train new machine learning model” button you can go to step 40.

By this point, you won't have enough examples to train the computer yet.

For the computer to know when it is a good idea to choose any space on the board, you need at least 5 examples of where you chose that space and ended up winning.

This page shows you how many examples you have so far. Look to see which one(s) you need more examples for.

The screenshot shows a web page titled "Machine learning models". At the top, there is a navigation bar with links: "ml-for-kids", "Welcome", "About", "Projects", "Worksheets", "News", "Help", and "Log Out". Below the title, there is a link "< Back to project". The main content area is divided into two sections: "What have you done?" and "What's next?".

What have you done?

You've collected examples of numbers for a computer to use to recognise when numbers are top-left, top-middle or 7 other classes.

You've collected:

- 3 examples of bottom-left,
- 2 examples of bottom-middle,
- 3 examples of bottom-right,
- 5 examples of middle-middle,
- 2 examples of middle-right,
- 3 examples of top-left,
- 3 examples of top-middle,
- 6 examples of top-right

What's next?

Keep going!

Go back to the [Train](#) page and collect more examples for each of the labels.

The more you can get, the better it should learn, but you need at least five examples of each as an absolute minimum.

37. Leave the “**Learn & Test**” window open. Go back to the **Scratch** window.

38. Play more games.

Try starting from a different position each time to get a variety of examples.

Try starting from positions that you know you need more examples of.

39. When you think you've got at least 5 examples of each space, go back to the “**Learn & Test**” window and **refresh** the page.

If there is still no “Train new machine learning model” button, you need to go back to step 37 and try again.

The screenshot shows a web-based application for collecting and training machine learning models. At the top, a navigation bar includes links for 'ml-for-kids', 'Welcome', 'About', 'Projects', 'Worksheets', 'News', 'Help', and 'Log Out'. Below the navigation is a main title 'Machine learning models'. A 'Back to project' link is visible. The page is divided into two main sections: 'What have you done?' and 'What's next?'. The 'What have you done?' section contains text about collected examples and a bulleted list of 13 specific examples. The 'What's next?' section contains text about starting training and a 'Train new machine learning model' button. At the bottom, there is a box labeled 'Info from training computer:' which contains the text 'Train new machine learning model'.

- 40.** Click on the “Train new machine learning model” button at the bottom of the page.

What have you done so far?

You’re teaching a computer to play noughts and crosses.

So far, you’ve updated a Scratch noughts and crosses game so that it can collect examples of how you play and add them to a set of examples. And you’ve used those examples to train a machine learning “model”.

The next step is to use that model to let the computer decide what move to make – instead of just going for the next empty space every time.

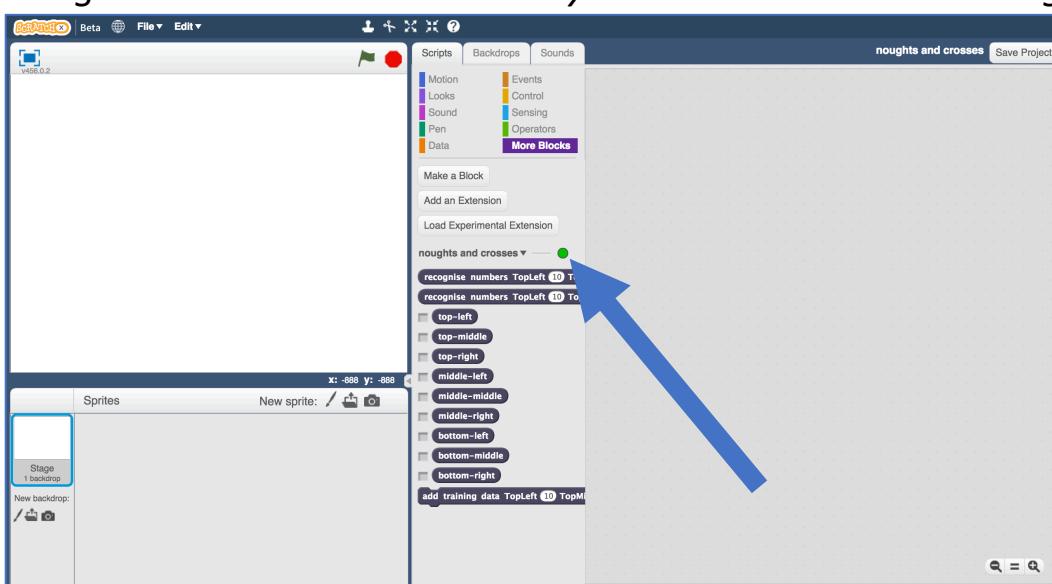
- 41.** Close the Scratch window.

Make sure you’ve saved your project first!

- 42.** Click the “< Back to project” link. Then click the **Scratch** button.

- 43.** Click the **Open in Scratch** button.

The green circle means this time you have a machine learning model.

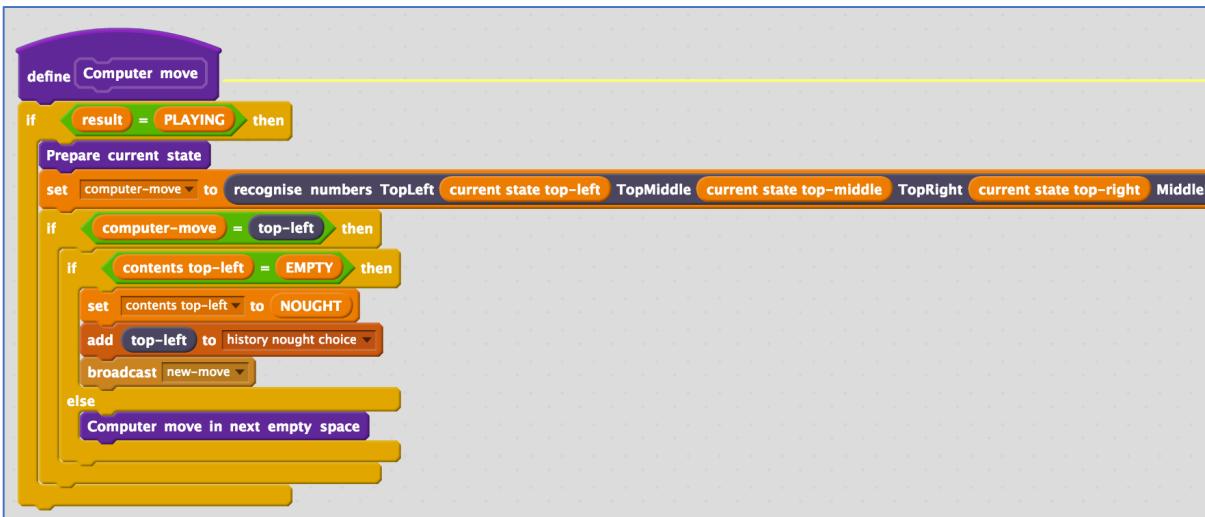


44. Open your saved project

*Click **File** -> **Load Project***

45. Click on the **Stage** and find the **Computer move** script

46. Modify the **Computer move** script so that it looks like this



The **recognise numbers** block is another very long one, but just match the space names with the "current state" data block that matches.



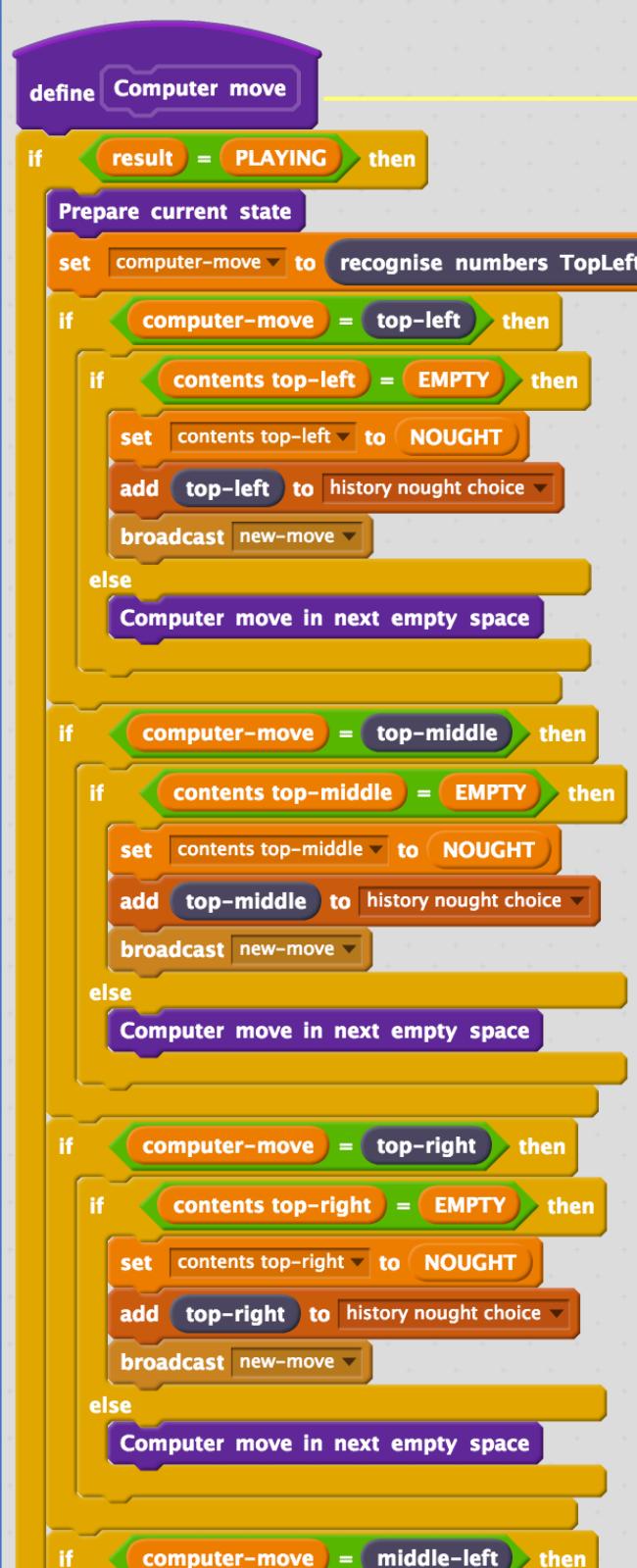
This script will use the machine learning model to let the computer choose where to have a move... (as long as it chooses top-left, for now).

While the computer is still learning, it may get things wrong. So there is also a check in the script to make sure that the computer has chosen an EMPTY space! If it's chosen a space that already has a nought or cross in, it'll just resort to picking the next empty space.

It will also add the choice it's made to the history, so it can be used in training data if it ends up winning.

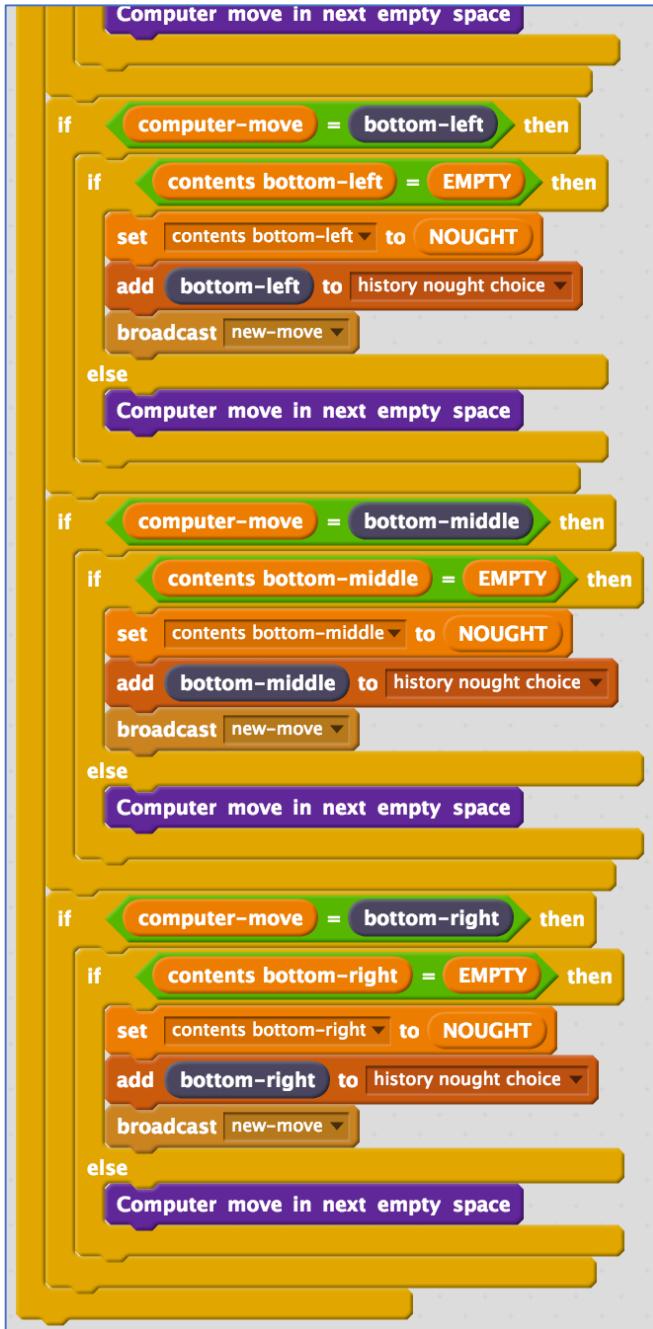
47. Update the **Computer move** script to handle all the other choices it could make. This is all one long script.

Use Duplicate on the right-click menu to save yourself some time.



The image shows a Scratch script titled "Computer move". It begins with a "define" block for "Computer move". Inside, there is a large "if" block where "result" is set to "PLAYING". The script then branches into six cases based on the value of "computer-move": "top-left", "top-middle", "top-right", "middle-left", "middle-right", and "bottom-left". Each case follows a similar pattern: it checks if the "contents" of a specific cell are "EMPTY", sets the cell to "NOUGHT", adds the cell to a "history nought choice" list, and broadcasts "new-move". If the cell is not empty, it instead says "Computer move in next empty space".

```
define Computer move
  if result = PLAYING then
    Prepare current state
    set computer-move to recognise numbers TopLeft
    if computer-move = top-left then
      if contents top-left = EMPTY then
        set contents top-left to NOUGHT
        add top-left to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
    if computer-move = top-middle then
      if contents top-middle = EMPTY then
        set contents top-middle to NOUGHT
        add top-middle to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
    if computer-move = top-right then
      if contents top-right = EMPTY then
        set contents top-right to NOUGHT
        add top-right to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
    if computer-move = middle-left then
      if contents middle-left = EMPTY then
        set contents middle-left to NOUGHT
        add middle-left to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
    if computer-move = middle-right then
      if contents middle-right = EMPTY then
        set contents middle-right to NOUGHT
        add middle-right to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
    if computer-move = bottom-left then
      if contents bottom-left = EMPTY then
        set contents bottom-left to NOUGHT
        add bottom-left to history nought choice
        broadcast new-move
      else
        Computer move in next empty space
    end
  end
end
```



48. Save your project

*Click **File** -> **Save Project***

49. Play against the computer by clicking on the **Green Flag**

Use full-screen to avoid moving sprites accidentally.

Try to avoid playing the same game over and over again.

Choose different spaces each time to give the computer a variety of examples of how to play.

50. When it starts to feel like you're playing the same games over and over again, go back to the Learn & Test screen, and use the new examples you've collected

*Click **train a new machine learning model** again*

The screenshot shows the 'Machine learning models' section of the website. At the top, there's a navigation bar with links: ml-for-kids, Welcome, About, Projects, Worksheets, News, Help, and Log Out. Below the navigation, the title 'Machine learning models' is centered. Underneath the title, there's a link '< Back to project'. The main content area is divided into two sections: 'What have you done?' on the left and 'What's next?' on the right. The 'What have you done?' section contains text about training a model to recognize numbers in various positions, a creation date of Friday, August 25, 2017 at 12:15 AM, and a list of 14 categories of examples collected. The 'What's next?' section contains instructions for testing the model with new numbers and a note about using Scratch to make a game if it's learning correctly. It also includes a button to train a new model.

51. Every time you train a new machine learning model, you will need to re-open the Scratch project so that it starts using the new model.

*Click **File -> Load Project***

Open your saved project, even though you already have it open!

*Click **OK** when it asks if you want to replace the current project.*

That will make sure you're using the latest model.

52. Go back to step 49 – and repeat

Play against the computer, and try to vary your playing as much as possible.

Once you're struggling to keep coming up with new games, click the Train new machine learning model button again, and re-load your saved project in Scratch.

Try to repeat this process a few times until your machine learning model starts getting good!

Tips

Don't be kind!

You might be tempted to go easy on the computer when you're playing against it, particularly when it's just starting to learn and is playing very badly.

For example, you might have two crosses-in-a-row next to a blank space and could win. But instead, you might feel sorry for it doing badly and put a cross somewhere else instead to give it a chance.

Don't.

It is learning from the way that you play. If you don't complete a three-in-a-row when you can, you will be teaching it that it should do that.

If you want it to get better quickly, **play as well as you can**.

Mix things up with your examples

Try to come up with lots of different types of examples.

For example, start from a different position on the board on every turn.

What have you done?

You've trained a computer to play noughts and crosses.

You didn't have to describe the rules to the computer.

You didn't tell it that it should try to get three noughts in a row.

You didn't describe the difference between rows, columns or diagonals.

(The rules are in the Scratch game, but that doesn't count – that wasn't used in the machine learning model).

Instead, you showed it how you play, by collecting examples of decisions that you made when you win.

When it makes decisions that leads to it winning, this is added to its training data, so it can be even more confident in that approach in future.

This is called “reinforcement learning” because when it does something good you are “reinforcing” this by rewarding it.

Did you know?

People have been learning about machine learning by training a computer to play noughts and crosses for decades!

One famous example was **Donald Michie** – a British artificial intelligence researcher. During World War II, Michie worked at Bletchley Park as a code breaker.

In 1960, he developed “**MENACE**” – the Machine Educable Noughts And Crosses Engine. This was one of the first programs able to learn how to play noughts and crosses perfectly.

As he didn’t have a computer he could use, Michie built MENACE using 304 matchboxes and coloured glass beads.

Each matchbox represented a possible state of the board – like the examples that you’ve been collecting in your training data.

He put beads in the matchboxes to show how often a choice led to a win – the number of beads in the matchbox was like the number of times an example shows up in one of the buckets you created for your training data.

