

OzoBot, the Musician

Zdeněk Tesař

NAIL052 Seminar on Artificial Intelligence II

Progress report

- First phase
 - Detailed information
- Second phase
 - Brief overview

■ First phase

- What is it about?
- OzoCodes
 - Datasets
 - Classifiers
 - Pros, cons & results
- Surface colors
 - Datasets
 - Pros, cons & results
- Surface colors as RGB
 - Dataset
 - Pros, cons & results
- Creating music sheet
- DEMO
 - OzoBlockly code

What is it about?

- Is about getting familiar with OzoBot capabilities
- Decode visited colors as inputs for machine learning algorithm which classify it as a note that is then played by OzoBot
 - Testing different ways of getting colors of line sections visited by OzoBot
 - OzoCodes, surface colors, surface colors as RGB values
 - Each way is coded to machine learning algorithms's inputs differently
 - Only selected combinations are tested
 - Testing direct implementations of machine learning algorithms' functions
- Subphases
 - OzoCodes
 - OzoCode to note
 - Surface colors
 - Surface color to note and octave (only four octaves are available)
 - Surface colors as RGB
 - Surface color to note, octave and length of play

OzoCodes

Datasets

- Only two-colors OzoCodes were used as an input
- Only notes from single octave were used as an output
- Each dataset prepared separately for each classifier
- Example dataset

OzoCode	Input	Class	Note
{ ■ , ■ }	1	C	SoundNote ["C"]
	0		
	8 total >		
{ ■ , ■ }	1	C#	SoundNote ["CSharp"]
	0		
	8 total >		
{ ■ , ■ }	1	D	SoundNote ["D"]
	0		
	8 total >		
{ ■ , ■ }	0	D#	SoundNote ["DSharp"]
	1		
	8 total >		
{ ■ , ■ }	0	E	SoundNote ["E"]
	1		
	8 total >		
{ ■ , ■ }	0	F	SoundNote ["F"]
	1		
	8 total >		
{ ■ , ■ }	0	F#	SoundNote ["FSharp"]
	0		
	8 total >		
{ ■ , ■ }	0	G	SoundNote ["G"]
	0		
	8 total >		
{ ■ , ■ }	0	G#	SoundNote ["GSharp"]
	0		
	8 total >		
{ ■ , ■ }	0	A	SoundNote ["A"]
	0		
	8 total >		

K < showing 1–10 of 12 > >

OzoCodes

Classifiers

- Classifiers for all parts of the first phase were created using Wolfram Mathematica
- Used classifiers
 - Decision tree

The decision tree classifier models class probabilities with a decision tree constructed using the CART algorithm.

- K-nearest neighbors

The nearest neighbors classifier infers the class of a new example by analyzing its nearest neighbors in the feature space. In its simplest form, it picks the commonest class amongst the "k"-nearest neighbors.

- Neural network

An neural networks is composed of layers of artificial neuron units. Each unit computes its value as a function of the unit values in the previous layer. Information is processed layer by layer from the feature layer to the output layer which gives the class probabilities. It is also called a feed-forward neural network or a multi-layer perceptron.

- Support vector machine

The support vector machine classifier separates the training data into two classes using a maximum-margin hyperplane.

The original feature space can be mapped into a higher dimensional space to improve linear separability. The multi-class classification problem is reduced to a set of binary classification problems.

- Simple example for demonstration

```
exampleClassifier = Classify[ozoCodesExampleDataset[All, "Input"] ->
  ozoCodesExampleDataset[All, "Class"], Method -> "DecisionTree"]
```

```
ClassifierFunction[   Input type: BooleanVector (length: 8)
  Number of classes: 12
  Method: DecisionTree
  Number of training examples: 12 ]
```

```
exampleClassifierInformation = ClassifierInformation[exampleClassifier]
```

Classifier information	
Input type	BooleanVector (length: 8)
Number of classes	12
Method	DecisionTree
Accuracy	2.78% ± 0.52%
Loss	2.89 ± 0.13
Single evaluation time	2.24 ms/example
Batch evaluation speed	99. examples/ms
Classifier memory	126. kB
Training examples used	12 examples
Training time	1.12 s

- Classifiers functions and logic can be exported and then manually rewritten to OzoBlockly language
 - Beware that this functionality is not working in the newest Wolfram Mathematica 11.3

OzoCodes

Pros, cons & results

Pros

- Simple classifiers that can be easily rewritten to OzoBlockly code
 - Note that neural networks used only one hidden layer with small number of neurons with simple threshold function etc.
- OzoBot can run some of these algorithms

Cons

- For such easy task machine learning classifiers are overkill
- Quality of “music” output based on selected method
- Need lot of if-else logic for parsing OzoCodes to input
 - 12 two-color OzoCodes
 - 36 three-color OzoCodes
 - ...

Results

- Some machine learning algorithms for classifying can be implemented and OzoBot is capable to run them
- Next subphase is doable
- Better to implement directly OzoCode to node table

Surface colors

Datasets

- Similar to OzoCodes subphase
- Two-colors line sections used as an input
- Notes from all available octaves were used as an output
- Each dataset prepared separately for each classifier

Surface colors

Pros, cons & results

Pros

- Simple classifiers that can be easily rewritten to OzoBlockly code
- OzoBot can run some of these algorithms
- Easier than OzoCodes to decode colors to inputs, because they can be read by color, not by whole combination

Cons

- Some classifier function created by Wolfram Mathematica were so complicated for OzoBot
- Quality of “music” output based on selected method

Results

- Similar to OzoCode subphase
- For some cases is if-else method bigger effort and cannot be such easily rewritten to OzoBlockly code as some machine learning algorithms

Surface colors as RGB

Dataset

- Similar to surface colors subphase
- Three-color line sections used as an input
 - First color used for note encoding/decoding
 - Second color was used for octave
 - Third color used as a note length
- I choose only one machine learning algorithm - combination of three decision trees
- Dataset was prepared especially for the chosen method

Surface colors as RGB

Pros, cons & results

Pros

- Better method of reading line colors

Cons

- Computation is too hard for OzoBot
- Long waiting before tone is produced - not pro musician, but beginner one

Results

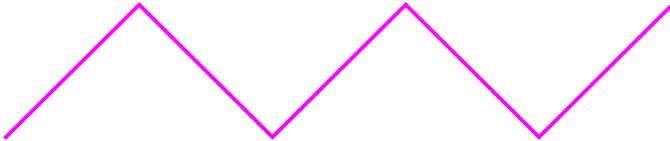
- Back to second subphase, but with newer color reading method

Creating music sheets

- Manually
- Microsoft Paint
- Wolfram Mathematica functionality
- Example of color line

Graphics[

```
{Thick, RGBColor[128, 0, 255], Line[{{1, 0}, {2, 1}, {3, 0}, {4, 1}, {5, 0}, {6, 1}}]}
```

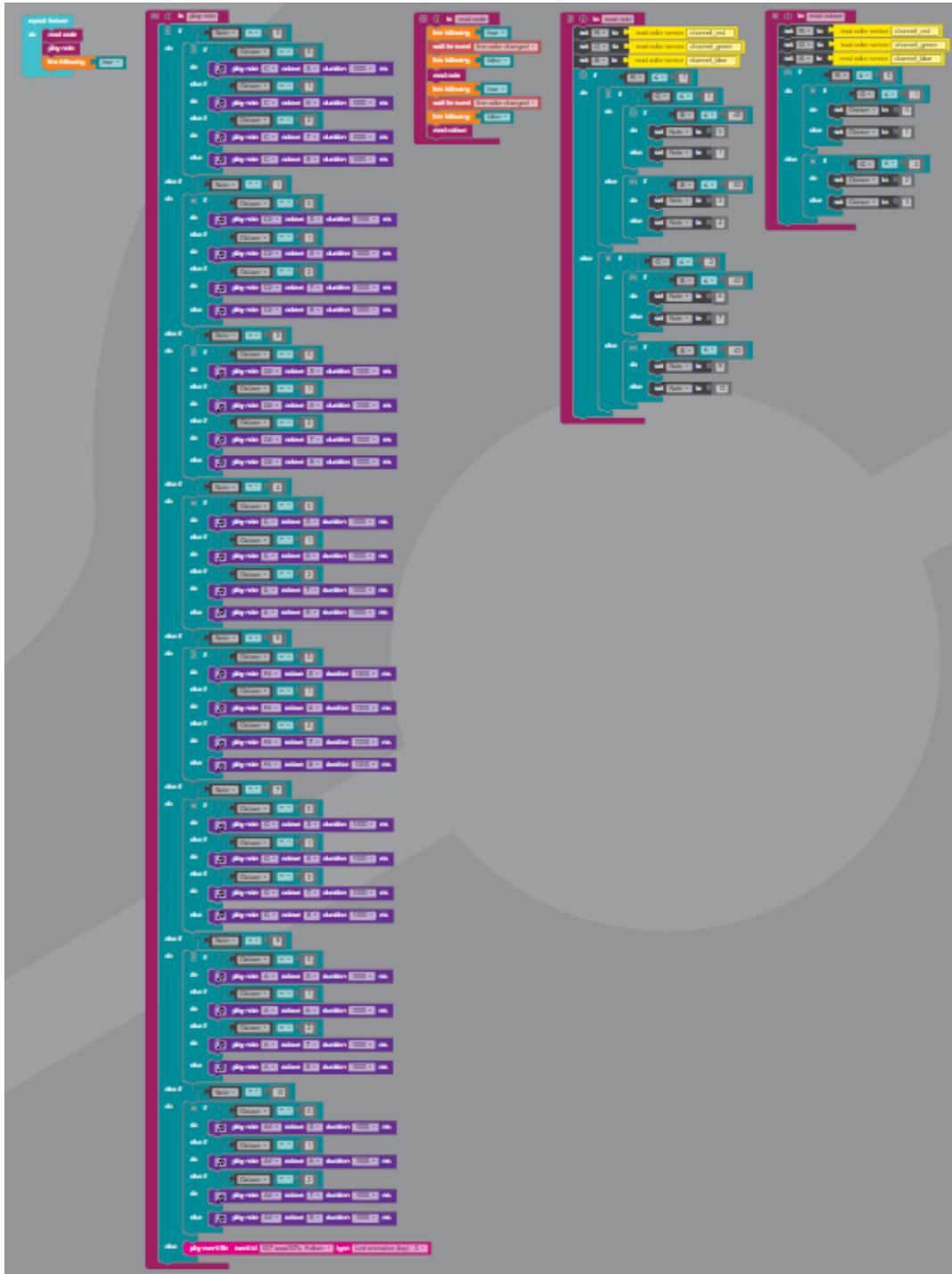


- For next progress presentation I will create Wolfram Mathematica cloud program that creates color lines music sheets from JSON config

DEMO

OzoBlockly code

- Visual OzoBlockly code



- OzoBlockly demo code
 - demo.ozocode

DEMO

■ Second phase

- What is it about?

What is it about?

- In first phase I tested direct implementation of learned machine learning algorithms
- In this phase I will try to implement som reinforcement learning algorithms
 - Temporal difference learning
 - Q-learning
 - SARSA
 - ...
- Currently working on, but unfortunately without presentable results yet

Thanks for your attention