Artificial Intelligence - Research Journal

Machine Learning

A new way of looking at Procedurally Generated Content (PCG) is to now train generators on existing content, so that the generator can make content of the same look or type. This was brought about by advancements in deep neural networks.

Generative Adversarial Nets – This is where a generative model is generated by simultaneously training two models: one is a generative model that records the data distribution and a discriminative model which that estimates the whether a sample has come from training data or whether it has been made by the generative model.

Generative Adversarial Nets and Variational Auto Encoders have made good progress in learning to create image of items such as bedrooms, cats or faces. Although these kinds of generative methods that are based off of machine learning can make decent renditions of some content such as music and images, some games content is a little more complicated. The main difference between generating content for a game and generating content for other domains, is that quite a lot of game content has specific structural constraints that must be fulfilled. For example a level must have a structure that allows the level to be completed.

The same programs that make almost correct images of bedrooms and animals, can’t be used as they still have structural discrepancies such as impossible angles or extra legs are less suitable for generating a map that needs to have an exit. This is why machine learning has not had much success in PCG for games. One other reason is that there is not much game content that the computer can train off of. This is however being researched and a lot of progress can be achieved in the next few years.

The main difference between PCG using machine learning and PCG that is search based **(RESEARCH**), is that the content is generated directly such as via sampling, from models that have been trained on game content. There are some search based PCG that have had their evaluation functions trained on game content such as Shaker et al (**RESEARCH**) or Liapis et al(**RESEARCH**) the actual content is actually generated based off search.

**N Gram Models**

The use of the n-gram model can be used for content that has many game levels. The way the n-gram model works is that you build conditional probability tables from strings and from the tables you take a sample when constructing new strings which is incredibly fast.

Dahlskog et al used an n-gram model to analyse the levels of super Mario Bros (Nintendo, 1985) and based on the information that the n-gram model took from the levels it could then build new levels based on this data. N-gram models are fundamentally one dimensional, the levels needed to be converted into strings so that the n-grams can be applied. The way this was done is by diving the level up into vertical “slices” where most of the slices overlap multiple times in a level. This can really only work if the level has a high level of redundancy which most games do. The models were trained using varying levels of *n* and while n = 0 basically created a level of random structures while n = 1 created levels that could be played but not very well n = 2 and n = 3 created levels that were well shaped and could be completed. Insert fig 4.10 here.

These models were then extended with the use of Monte Carlo tree search by Summerville et al. Instead of just using the learned conditional probabilities, they used the probabilities during rollouts which are generations of whole levels, they were then given an objective and scored on how well that objective was carried out by a designer. This meant that they could make the levels more or less difficult if they wanted. Through the use of Monte Carlo tree search the designer would have more control over how the level could be set out. Other than just editing the input corpus. This method could be seen as a mix between search-based method and machine learning.

Neural network architectures are highly useful in learning-based PCG. Hoover at al made levels using a concept called functional scaffolding for music composition (FSMC) this was originally made to compose music. The original FSMC says that 1) music can be constituted as a function of time and 2) musical voices in a given piece are functionally related. Through the use of NeuroEvolution of augmenting topologies which evolves neural networks, additional voices are evolved to be played simultaneously with an original human–composed voice. To represent Mario levels, the levels have to be seen as functions of time and the levels are broken down into a sequence of tile-width columns. The height of each of these columns extends to the height of the screen. The FSMC represents a unit by the length of an eighth-note, the unit of time in this example is the width of each of the columns. At each unit of time, the system checks with ANN to decide the height at which it should place the tile. FSMC then sends a musical note’s pitch and duration to ANN which is then translated into how high should the tile be placed while duration is translated into how many tiles are placed a that height. For a specific tile type or musical voice the information is fed into a neural network that is trained on two thirds of the existing human-authored levels so that the file type can be predicted at each column. The purpose of this is hopefully the neural network will realise hidden relationships between the specific tile types and the human authored levels used, so that humans can the construct entire levels from as little starting information as possible.