**Neural networks**

The goal of any machine learning system is to learn a generalised mapping from input to output data. This mapping then allows you to take instances where the output variables are unknown, but input data is available, and to make predictions.

In a neural network this is done by a series of algorithms which successively transform data through a series of layers. In each layer, the representation of the data becomes increasingly distant from the original data, but increasingly informative about the final result.

Neural networks have three kinds of layer:

* Input layer – provides information from the outside world to the network. Does not perform any computation; just passes on information to the hidden layer(s).
* Hidden layer(s) – perform all sorts of computation on the features entered through the input layer and transfer the results to the output layer.
* Output layer – brings the information learned by the network to the outer world.

Each layer consists of small individual units known as neurons. Each neuron runs the input data through some sort of non-linear activation function and produces an output value which is then passed onto the next layer in the network. Each neuron represents a different hypothesis on how to transform the data (i.e. take the values from the previous layer and spit out a number as an output).

A neuron contains the following components:

1. A mathematical function which is known as an activation function
2. Inputs
3. A vector of weights
4. A bias

First, the neuron computes the weighted sum of the inputs, and then it adds a bias (constant) to the weighted sum. This computed value is then fed to then activation function, which prepares an output.

The **activation function** defines how the weighted sum of the input is transformed from a node in a layer of the network. The purpose of the activation function is to add non-linearity to the neural network.

Each **neuron** in the network has a weight value and a bias value. At the start of the network’s training these are assigned to random values. The weight of the neuron determines how much of an influence it has on the neuron which it feeds into.

Each neuron also has a **bias**/constant/offset value. This is used to shift the results of the activation function to the positive or negative side (a bit like an intercept in a linear regression). Bias make up the difference between the function’s output and its intended output.

To control the output of a neural network a method is needed to measure how far the output is from what you’d expect. This is the job of the **loss (or objective) function**. The loss function takes the predictions of the network and the true target (what you wanted the network to output) and computes a distance score, capturing how well the network has done on this specific example.

Initially the **weights** of the network are assigned random values, so the network merely implements a set of random transformations. And so naturally the output is far from what it should ideally be. But with every example the network processes, the weights are adjusted a little in the correct direction and the loss score decreases. This is the training loop which, repeated a sufficient number of times (typically tens of iterations over thousands of examples) yields weight values that minimise the loss function.

**Recurrent neural network**

There are a number of types of neural network.

* Feed-forward neural network: the simplest kind. Information flows in one direction only, forward. The information starts at the input layer, goes to the hidden layers and ends at the output layer. The network does not have a loop
* Recurrent neural network: a multi-layered neural network that can store information in context nodes, allowing it to learn data sequences and output a number or another sequence. The network includes loops.

RNNs are good at processing sequential data. Whilst feed-forward neural networks have no memory of the input they receive and are bad at predicting what’s coming next, in RNNs the information cycles through a loop and when it makes a decision it considers the current input and also what it has learned from the previous inputs.

The project

Activity 4 – Deep Learning Models of Urban Dynamics. Validating agent-based models (i.e. Quantifying the extent to which they are able to accurately represent the system under study) is one of the key ongoing challenges for the discipline [12, 13]. Although data assimilation will reduce uncertainty in the model outcomes by “using all the available information” [54], it will be important to apply additional validation methods. “Docking” [2, 57] is commonly used, whereby a second model is used ascertain whether similar results can be replicated. This project proposes a novel approach drawing on the ongoing innovations in the field of machine learning to adapt a recurrent neural network (RNN) to model urban flows at an aggregate level. Unlike traditional neural networks, that are stateless, RNNs maintain information about a history of past inputs [30]. They use this historical state vector to predict future states. For example, RNNs have been used to generate new text in the style of the author that they were initialised with, e.g. Shakespeare [27]. Therefore data about population flows or densities (from cameras that count the number of passers-by, aggregate mobile phone counts, etc) will be used to initialise an RNN that can subsequently be used to make short-term predictions about future urban dynamics. This approach is novel and interesting in its own right –neural networks have not been applied to the problem of predicting short-term population flows – and will provide a valuable means of validating the research results.

--

Model produced won’t do direct validation, but will talk about opportunities that machine learning offer for real time footfall predictions.

Machine learning will be an aggregate model rather than an agent based model – but does it matter that it doesn’t have this richness? If machine learning model can predict it quite well, then why both with ABM.

End of project, report will say building ABM was too ambitious.

Innovation in it’s own right, people haven’t done it yet for modelling footfall. In a paper wouldn’t conceptualise it in terms of validating an ABM, but in report for funding report would go into this (but this isn’t something I need to worry about).

Roads – stuff I did previously was just a footfall count for the city centre. Camera is pointed at a particular road intersection. So could get road network data and then try and predict what the footfall on a particular road section would be on a particular time or day of the week. Can get whole road network for city. So could say on a particular day, this is how we expect people to be distributed across the roads. If that works well, could then make predictions for other roads in cities, as long as you capture the right independent variables. This would be assuming that the same variables drive footfall in the city centre and a cul-de-sac in Horsforth? Problem is that only have data for city centre, which isn’t very representative.

First thing to do would be to look for data in other cities – Singapore, Melbourne, Boston. Leeds is probably not the best place to get footfall or contextual data. Looking for cities with any kind of footfall data (counts of people per hour/day). [Ambient population? Different terminology for data].

Neural networks – suggested RNNs because they keep some kind of history. When it makes predictions it remembers recent changes, so might be good for footfall…if you have a model and want to predict the next hour, then a RNN might allow you to base your prediction based on the previous hour. Does it with loops…not exactly sure, some kind of feedback.

Part of project would be looking for new data, and looking into methods. Previous work – identified some issues with using NN on footfall data, due to non-linearity’s (e.g. effect of Xmas).

Look at machine learning for modelling urban flows/mobility. See what other people have done…probably very little? Nikolai Bode…don’t some stuff. He has a project looking at mobility or urban flows or something. Model-driven construction of city-level pedestrian traffic maps over time