Using deep learning to measure the strength of non-linear patterns

Donald Hobson

September 4, 2017

Abstract

When data is correlated in a linear manner, then standard statistical tests can detect and measure this correlation. The aim of this project is to detect relationships in the data more complex than as A increases then B increases. Techniques involving neural networks that can detect a wide range of complex patterns are demonstrated.

1 Approach 1

1.1 Method

It was assumed that the data came from a set of sensors $\beta_1, \cdots \beta_n$, each of which had a time series (eg temperature) $\phi(t)$ and some scalar (eg elevation or local tree cover)κ associated with it. The first pattern that was to be detected was a linear relation between κ and $f(\phi) = f(\phi(1), \phi(2), \cdots \phi(t_{max}))$ where f was some function. As there are an infinity of possible functions to use for f it was decided to find a suitable function by training a convolutional neural network on the $\phi(t)$ sequence to generate κ. To stop units or scaling making a difference to the algorithm, both $\phi(t)$ and κ are normalized so they have a mean of 0 and a mean-square of 1. To avoid spurious correlations coming from the network memorising random data, the sensors are split into a training set and a testing set, the network attempting to learn patterns on the training set and the strength of the pattern being measured on the test set. The network is trained to minimize L2 loss $mean((\kappa_p - \kappa_r)^2)$ between the predicted value κ_p and the real value κ_r. The loss for the testing dataset must be at least 1 if there is no pattern, due to the normalization, with this being achieved when $\kappa_p = 0$. If κ can be perfectly predicted from $\phi(t)$ then $\kappa_p = \kappa_r$ and loss equals 0. There is a tendency for the network to first learn any general patterns, then over-fit to random noise in the data. To help minimise this problem the error function is evaluated for the test data throughout the training process and the lowest value is taken. The whole process can be repeated.

1.2 Results

The algorithm performed well on samples of generated training data. When given random noise it output ≈ 1 and when given a strong correlation it outputs lower values. Tests on real data to follow.

2 Approach 2

2.1 Method

Method 1 only indicates how patterned the data is. Suppose the data is a mixture of strongly patterned data, $\phi(t)$ such that κ can be accurately predicted, coming from some sensors and unpatterned data, $\phi(t)$ that is unrelated to κ, from other sensors. In this case it would be usefull if the network could predict the error in its prediction $\kappa_e \geq 0$ as well as the values from before. To do this we use the error function $\frac{1}{2}mean((\kappa_p - \kappa_r)^2 \div \kappa_e + \kappa_e)$. This function is chosen because it is mathematically simple and again gives 1 for no relationship and 0 for a perfect prediction.

3 Approach 3

Suppose you have a set $\Sigma = \{s_1, s_2 \cdots s_n\}$ of datums. Each datum consists of two vectors chosen from vector spaces Φ and Ψ respectively. So $\Sigma \subset \Phi \times \Psi$ and $s_i = (x_i, y_i)\ for\ x_i \in \Phi, y_i \in \Psi$ Consider each datum to be an independant sampling of a random variable $S = (X, Y)$ We know that X and Y are statistically independant if $PDF(X|Y = y) = PDF(X)$ Let $\alpha = PDF(X = xandY = y)$ and let $\beta = PDF(X = x) \times PDF(Y = y)$ as implicit functions of x, y. Define $f(x, y) = \frac{\alpha}{\alpha+\beta}$ Given i, j such that $P(i = j) = \frac{1}{2}$ then

$$P(i = j|X = x_i, Y = y_j) = \frac{PDF(X = x_i, Y = y_j|i = j)P(i = j)}{PDF(X = x_i, Y = y_j)} = \frac{PDF(X = x_i, Y = y_i)\frac{1}{2}}{\frac{1}{2}(\alpha + \beta)} = \frac{\alpha}{\alpha + \beta} = f($$

We can then calculate the info of pairing $\nu = \begin{cases} -\log_2 f\ \ if\ \ i=j \\ -\log_2(1-f)\ \ else \end{cases}$. The expected amount of information gained about the pair (x_i, y_j) by knowing if $i = j$ is $1 - \nu$. The amount of info needed to say if $i = j$ given (x_i, y_j) is ν. $\nu = 1 \Rightarrow$statistical independance between X and Y. If $nu = 0$ then the product vector space can be split into disjoint subsets $A, B \in \Phi \times \Psi$ with $A \cup B = \emptyset$ Such that if $i = j$ then $P((X, Y) \in A) = 1$ and if $i \neq j$ then $P((X, Y) \in B) = 1$

This is a useful metric for the distribution, how can we estimate it with acess only to a random sample from the distribution. Here we encounter a problem in that we can fit the data to a simple probability distribution with a high ν or to a highly convoluted distribution with a low ν. We have to assume that the data is simple unless we have sufficient evidance otherwise. To avoid such overfiting problems, we will split the data into a training set and a testing set. During training the network is expected to learn the pattern, while the testing

removes any pattern that could be explained by overfitting. Note that the network is never presented with a pair that contains training and testing data.

$$\nu = -\int (\alpha \log_2 f + \beta \log_2 (1 - f) \tag{1}$$

$$= -\int \log_2 (f^\alpha (1 - f)^\beta) \tag{2}$$

$$= \int \log_2 (\frac{\alpha^\alpha \times \beta^\beta}{(\alpha + \beta)^{\alpha+\beta}}) \tag{3}$$

$$= \int ((\alpha + \beta) \log_2 (\alpha + \beta) - \alpha \log_2 \alpha - \beta \log_2 \beta) \tag{4}$$