Using Convolutional Neural Networks for Deconvolution: an Novel Approach for Understanding Datasets at Finer Timescales

Greg Weaver
CS 230 Project

Introduction

Finding the distribution of a random variable after it has been summed is hard. The process typically involves taking the inverse Fourier Transform of a distribution, which is a painstaking process. I propose a method using machine learning that determines what family of distribution best fits a dataset at a time scale that is 10 times finer than the dataset itself: something that deconvolution is trying to solve. However, this method has considerable advantages in its simplicity and generalizability.

Concept

The Central Limit Theorem states that a random variable $X \sim f$, when summed, will be best approximated by a normal distribution (fig 1a). In short, most random variables tends towards fitting to normal as they are summed. What behavior of fit do these density functions have towards other distributions? I show that this behavior is unique to each family of distribution (fig 1b), and thus can be approximated by a neural network.

Methods

Deconvolving Without a Filter

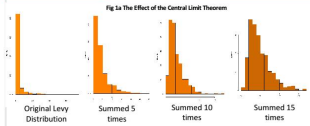
Given a set of P observations, denoted as v, I create a 100 x P matrix, denoted as A, where row 1 is v and rows 2-100 are permutations of v. From matrix A I create matrix B, where each row of B, $B_{[i,\ ]} = A_{[1,\ ]} + \cdots + A_{[i-1,\ ]} + A_{[i,\ ]}$. Each row in B is then a distributions of values. I then use maximum likelihood to fit each row to 5 different distributions (normal, lognormal, levy, Weibull, and logistic), which maximizes the probability $f(B_{[i,1]}, B_{[i,2]}, \ldots B_{[i,99]}, B_{[i,100]}|\theta) = \prod_{q=1}^{100} f(B_{[i,q]}|\theta)$, where f is one of the five density functions to be estimated, and θ is the set of distribution-specific parameters. I then take the Kolmogorov-Smirnov statistic for each distribution at each row to measure goodness of fit.

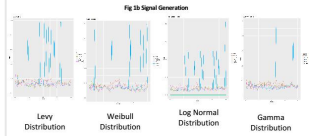
Doing this over all rows creates a central limit theorem "signal" unique to the family of the posterior distribution. Training a Convolutional Neural Network to classify the signal that is generated by successive summation provides an effective means for classifying the distribution to be deconvolved.

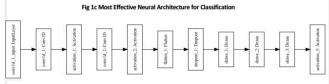
Predictions

The signal that is described above, represented in 5 100-length vectors, is put into a 1-d convolution neural network. The output is a label of what density function that this dataset came from.

CLT, Signals, and Neural Architecture







Discussion

The results are promising because it shows that the right neural architecture can classify this signal with 91% accuracy. It is important to note that this means that given a dataset that is the result of 10 summations, of which all look like a normal distribution. Given that it looks like the signal generated is unique to the original distribution, I hoped that a neural network would be able to tell apart the signals that accompany each distribution. It is important to note that the expressivity of having all five signals from each distribution made an a huge difference in network accuracy. Simply concatenating all five signals into a single vector resulted in random accuracy (20%).

Data

Using R libraries to create random variables and utilize MLE methods, I create a training dataset of 10000 examples with 2000 examples from each of the five classes that I am classifying. Each example consists of 5 100-length vectors that represent the behavior of fit to a distribution. Examples are labeled from ground truth. Inputs were not normalized as it drastically reduced accuracy. The signal is constructed via MLE methods and Kolmogorov statistics of a dataset.

Models

See figure 1c.

Results

Model	Train Acc.	Test Acc.	Tra. Size	Test Size
Vanilla, two-layer	.97	.87	10000	1000
2 Conv1D layers+relu	.92	.91	10000	1000
3 Conv1D layers+sig	.997	.8	10000	1000
2 Conv1D layers + str = 5	.772	.32	10000	1000

Discussion

The applications of this involve any time series that is altered through convolution. For example, what this method allows us to do is that if one has displacements of someone moving every 24 hours, we can use this method to see what function governs their movement every 2 hours.

LeCun, Bengio (1995). *Convolutional Networks for Images, Speech, and Time-Series*.