Yale University

Senior Thesis for the Mathematics and Computer Science major

Recurrent Neural Networks for Change Point Analysis

William Cai

supervised by
Prof. Sahand Negahban

December 15, 2016

Abstract

In this paper we present a synthetic time-series dataset which is motivated by vital measurements for patients in the ICU (MIMIC). At time zero, the time series moves according to one probability distribution. We then have some criterion (motivated by events such as cardiac arrest), which a fraction of the generated examples trigger, causing the probability distribution to change for the remainder of the time series. We then try to solve the problems of regression, predicting what the next measurement will be, and classification, predicting which generation function is currently generating the data, on this dataset. The models we consider include a random forest, single-layer perceptron, deep neural network, and a recurrent neural network. We give a comparison of the capability of these models to do this kind of change point analysis, and also consider their robustness in the face of missing data.

1 Introduction

-¿ Lot of interesting data is in time series
-¿ RNNs are really powerful for time series
-¿ Lot of interesting time series have their distributions change at some point
-¿ e.g. DYNACARE
-¿ Question: Can we use recurrent neural nets to forecast a hazard function?

2 Data

Our synthetic data takes the form of a n x T x d matrix where n is the number of patients, T is the number of measurements, and d is the dimensionality of the measurements.

Before we begin generating the data, we initialize d x d matrices A and B which have 2-norm .9 and are rank 2. We also initialize betashift, a d-dimensional vector where each entry is drawn from a normal distribution with mean 0 and variance 1 and normalized by $\sqrt{d}$. Then, the synthetic data for each patient is generated as follows:

P, the data for a single patient, is a T x d matrix. Let P_i be the measurement of that patient at time i.
P_0 is randomly initialized with all values drawn from a normal distribution with mean 0 and variance 1 and normalized by $\sqrt{d}$.
Then, $P_{t+1} = (1_A A + (1 - 1_A)B)P_t + \sigma_w \epsilon$ where:

σ_w is set to .2

ϵ is d-dimensional white noise with all values mean 0 variance 1

1_A is an indicator variable which is 1 if there is no $i \leq t$ s.t. $P_i \cdot$ betashift $> .5$

To feed our data into our models, we use the window technique. This means that each sample contains measurements from time i to time $i + x - 1$, where x is our window size.

For the models besides the recurrent neural net, this data is flattened into a vector of length xd, while recurrent neural networks take the data as an x by d matrix. For regression this is regressed against the measurement at time $i + x$, and for classification we want to do binary classification on the value of 1_A at time $i + x$. Our generated examples were split 80-20 into the training and validation sets.

3 Models

3.1 Random Forests

We used scikit-learn's RandomForestRegressor and RandomForestClassifier. We set the forest to have 40 estimators and features, except when the window only contained one set of measurements when we set it to 10 estimators.

3.2 Neural Network

Our Neural Networks were built with Keras. We used a single layer perceptron with 512 nodes using the rectified linear unit (ReLU) activator, with a dropout of .25. For regression, the neural network feeds into a layer of d nodes, which is the dimensionality of the measurements we are trying to predict. For classification it feeds into a single node which is converted to a classification using the sigmoid activation.

3.3 Deep Neural Network

We used a two layer perceptron where each layer consisted of 512 nodes using the rectified linear unit (ReLU) activator, with a dropout of .25. Again, for regression this feeds into a layer of d nodes and for classification it feeds into a single node which is converted to a classification using the sigmoid activation.

3.4 Recurrent Neural Network

We used a 16 node Gated Recurrent Unit using the rectified linear unit (ReLU) activator which fed into a perceptron with 128 nodes and a dropout of .5 which also used the rectified linear unit (ReLU) activator. Again, for regression this feeds into a layer of d nodes and for classification it feeds into a single node which is converted to a classification using the sigmoid activation.

4 Experiment

There are two prediction problems we are interested in:

1) Given P_0 through P_t, predict P_{t+1} (regression).
2) Given P_0 through P_t, predict 1_A (binary classification)

4.1 Regression

Given a time series of measurements, we would like to be able to predict what the next measurement will be. First, we give a brief description of a variety of models that we implemented to solve this task. All of these models use a fixed window of measurements as an input (i.e. the x previous measurements). For this entire section, we let the dimensionality d of the data be 10, the number of timesteps per patient T to be 175, and the number of patients N to be 200. For each model, we give a brief description of the model and show the model's predictions for a patient in the validation set. Then, at the end of this section, we give a comparison of the models over various values of x, as described previously.

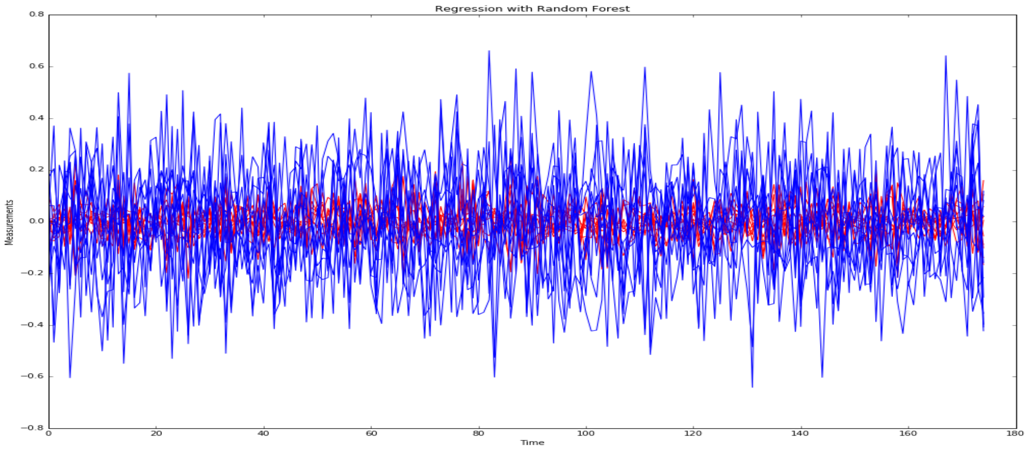
4.2 Regression Baseline

As a baseline, we check to see how our models work when A = B, or in other words, the patient measurements are generated by:

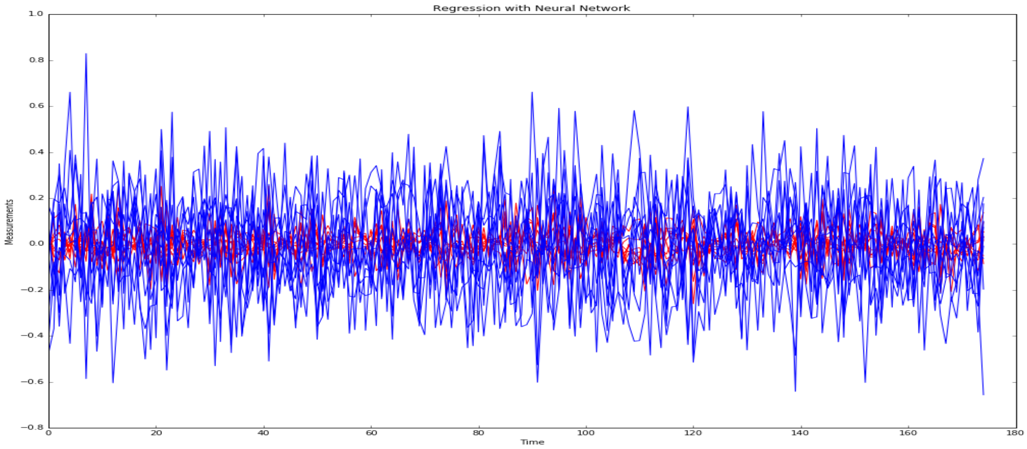
$$P_{t+1} = AP_t + \sigma_w \epsilon$$

In this case, the 'correct' model should predict $P_{t+1} = AP_t$.

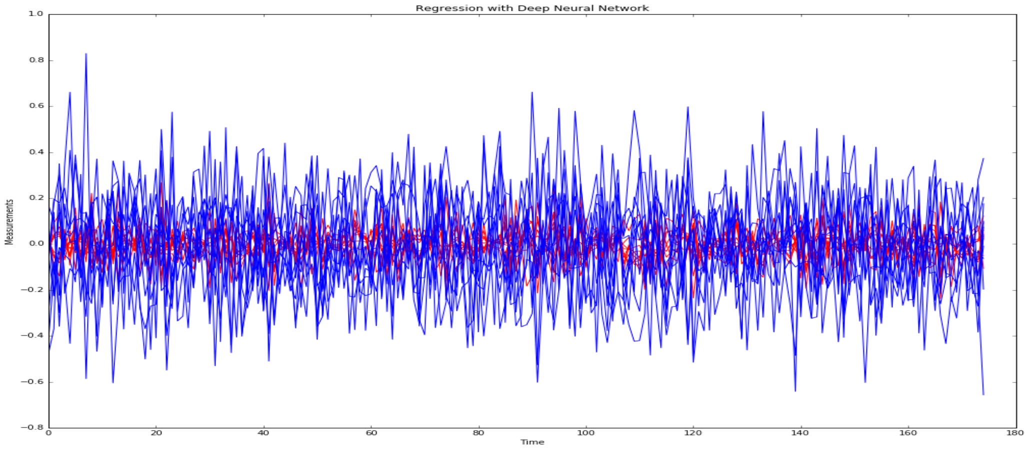
4.2.1 Random Forest Regression



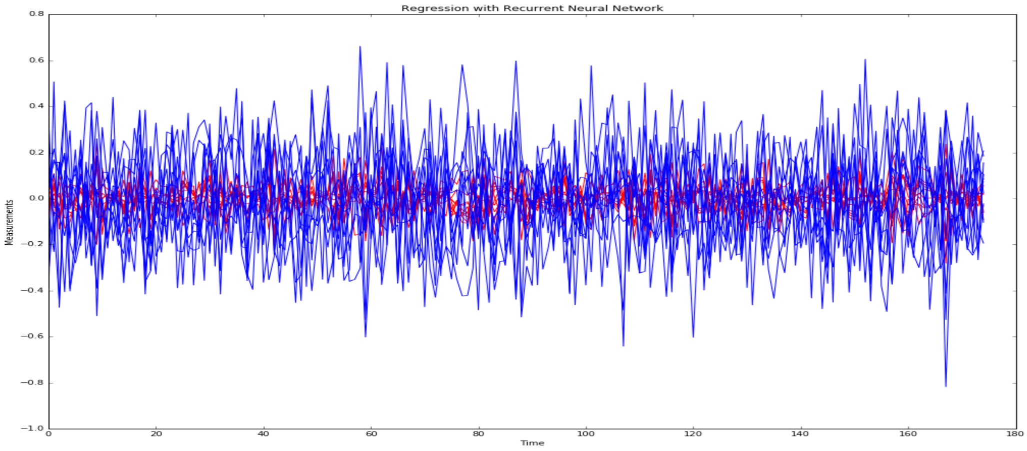
4.2.2 Neural Network



4.2.3 Deep Neural Network

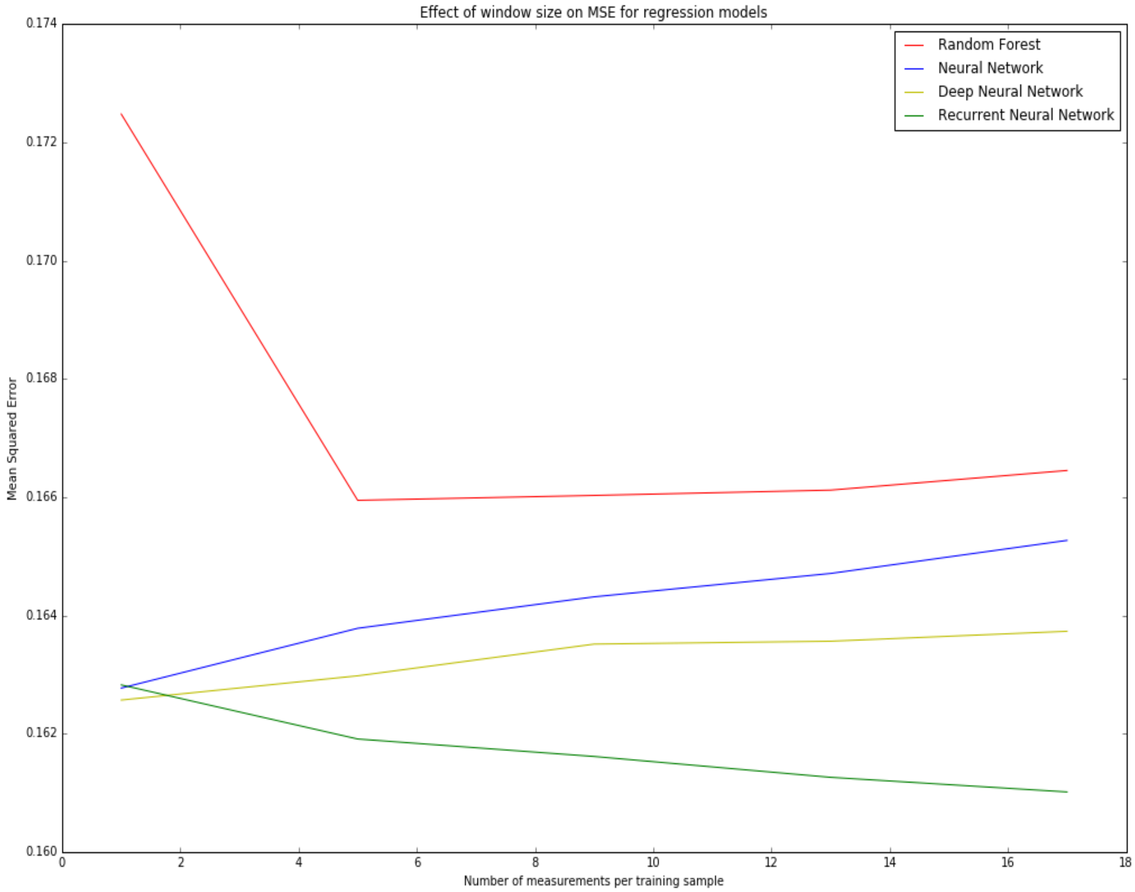


4.2.4 Recurrent Neural Network, window method



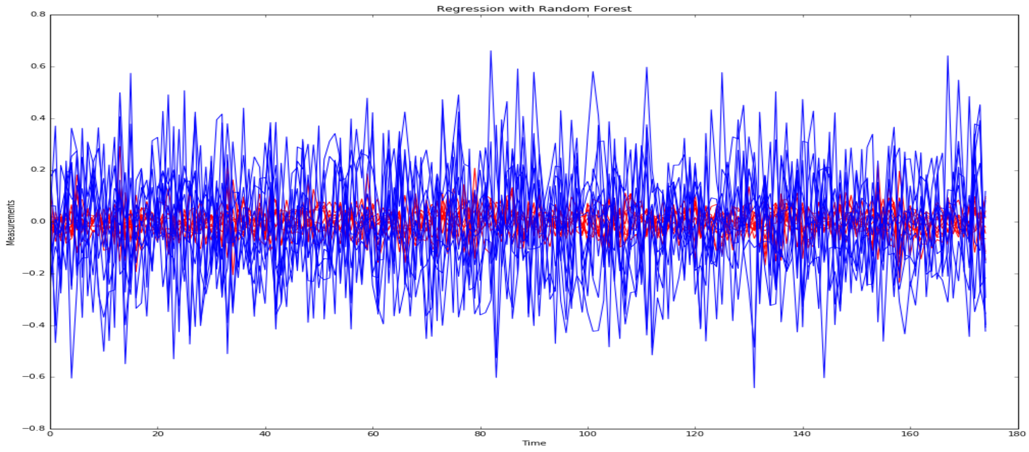
4.3 Regression

4.3.1 Comparison



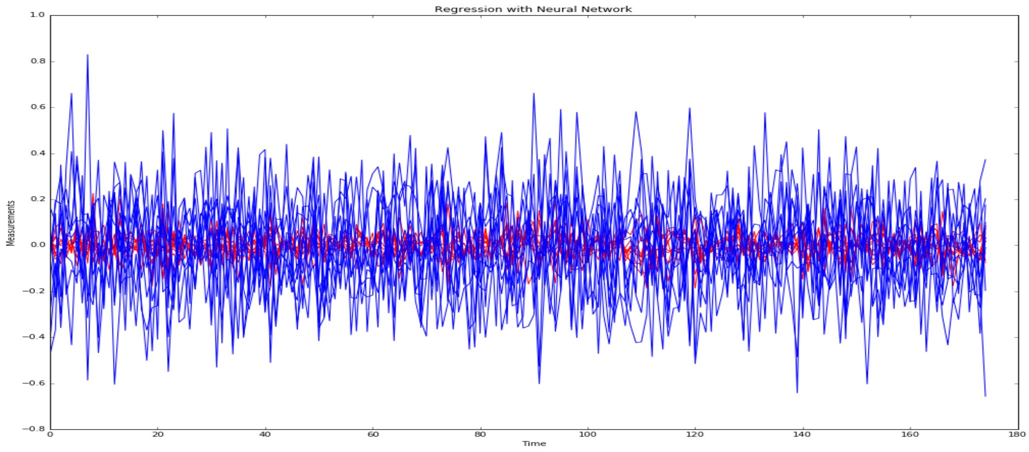
Interestingly, besides the jump when the random forest is run on windows of length 5 instead of windows of length 1, every one of the non-recurrent models does worse as we increase the window size. This suggests that the models struggle to train on the additional information that they are learning on. However, the recurrent neural network's error monotonically decreases as we increase the window size - this demonstrates the power of recurrent neural nets in learning temporal associations.

4.4 Random Forest Regression



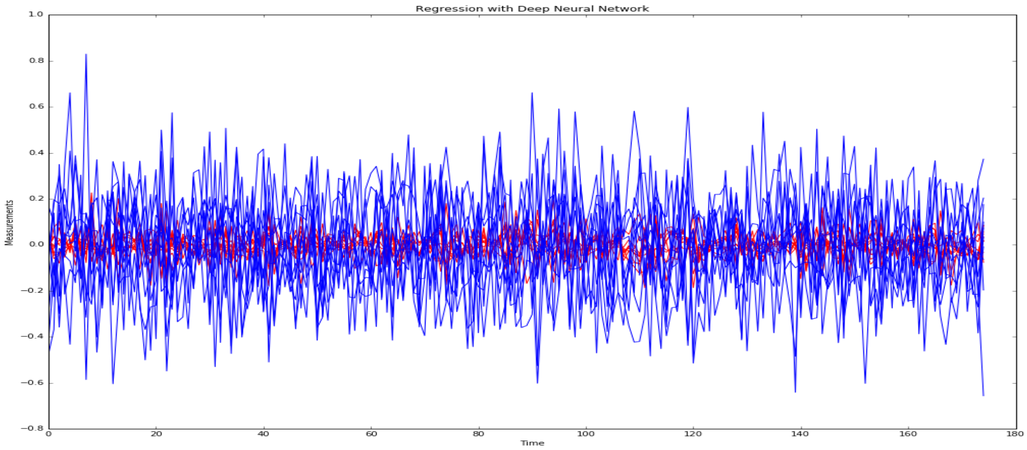
We used scikit-learn's RandomForestRegressor, trained with 40 estimators and features, except when we were using a sequence length of 1, in which case we only used 10.

4.5 Neural Network



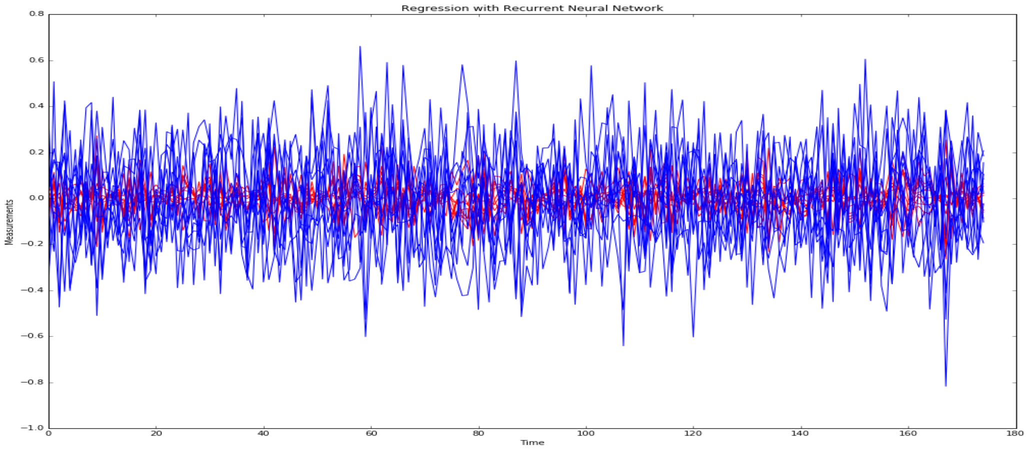
We used a single layer perceptron with 512 nodes using the rectified linear unit (ReLU) activator, with a dropout of .25.

4.6 Deep Neural Network



We used a two layer perceptron where each layer consisted of 512 nodes using the rectified linear unit (ReLU) activator, with a dropout of .25.

4.7 Recurrent Neural Network, window method



We used a 16 node Gated Recurrent Unit using the rectified linear unit (ReLU) activator which fed into a perceptron with 128 nodes and a dropout of .5 which also used the rectified linear unit (ReLU) activator.

4.8 Classification

4.8.1 Random Forest Classifier (30 trees, 30 features)

4.8.2 Neural Network (1 layer, 512 nodes)

4.8.3 Deep Neural Network

4.8.4 Recurrent Neural Network, window method

4.8.5 Recurrent Neural Network, stateful

5 Experiment

5.1 Regression where $A = B$

5.1.1 Results

Models:
All Zero Regressor:
0.168840895246
Oracle Regressor (knows the value of A):
0.159880463551
Random Forest Regressor
0.164463405344
Neural Net, 512 nodes
0.161696706556
Deep Neural Net, 2 layers of 512 nodes
0.160718555103
Recurrent Neural Net
0.160057111654

5.2 Regression where $A \neq B$

5.2.1 Introduction

Given a sequence of measurements from time 0 to 1, our goal is to predict the measurement at time t + 1. Now that $A \neq B$, our data is generated by

$$P_{t+1} = (1_A A + (1 - 1_A)B)P_t + \sigma_w \epsilon$$

which means our models have to deal with the fact that sometimes the data is being generated with A and sometimes with B.
All of these models were trained using the window method, where each training example is a sequence of 10 measurements and the result is the next measurement. The dataset was split 80-20 between training/validation.

5.2.2 Results

Models:
Zero predictor (just guess all-zeros):
MSE: 0.168869027809
Random Forest
MSE: 0.17058158425
Neural Net, 512 nodes
MSE: 0.161847476813
Deep Neural Net, 2 layers of 512 nodes
MSE: 0.160740647684
Recurrent Neural Net
MSE: 0.161418953358
The error of an oracle predictor which knows the value of A, B and flag should be similar to the error from the previous section but wasn't implemented because it's hard to figure out the value of flag for a given training example.

5.3 Classification where $A \neq B$

5.3.1 Introduction

Given a sequence of measurements from time 0 through t, our goal is to predict the value of flag at time t + 1.
All of these models were trained using the window method, where each training example is a sequence of 10 measurements and the result is the next value of flag. The dataset was split 80-20 between training/validation.

5.3.2 Results

Random Forest

```
[[5104 28]
[1271 197]]
```

Neural Net, 512 nodes

```
[[5009 123]
[ 196 1272]]
```

Deep Neural Net, 2 layers of 512 nodes

```
[[5011 121]
[ 212 1256]]
```

Recurrent Neural Net

```
[[4965 167]
[ 148 1320]]
```

6 Discussion