Time series analysis: the effect of adding an unsupervised layer to NN time series prediction

David Seetapun

December 14, 2012

Introduction and Synthetic Data

Let $\{Y_t\}$ be an observed time series where the interval between observations is fixed. We consider models which take a window of length d of the time series so that Y_{s+d} is the response to $\langle Y_s, Y_{s+1}, \ldots, Y_{s+d-1}\rangle$. We will train the models to predict the next observation in the time series. In order to predict an observation some number of periods f in the future we will predict the next observation and iterate the predictor. The method of stochastic sampling may be also used [1] but we will not consider it here.

An $ARMA(p,q)$ process is given by $Y_t = \sum^{p} \phi_i Y_{t-i} + \sum^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$. Following [2] we start the investigation with synthetic data as we will then have solutions for the distributions of the the Y_t and we can then compare $E(Y_t)$ to the model predictions. In what follows Y_t is a sample of 500,000 ARMA(2,2) with $\phi = .35, .45$ and $\theta = .15, .25$. Also we add .01 to each element so that $Z_t = \sum^{t} Y_i$ will have a non constant mean.

ARMA(2,2), Gaussian process models and FFNN

We use a set of 50000 observations, split by random selection into 75% training set, 25% validation set for the NN. The test set consists of the first 500 observations. We fit a Gaussian process model with exponential kernel (by optimizing the hyperparameters), a FFNN with hidden layer size 5 and window size 5 and an ARMA(2,2) model. The optimal predictions were obtained from the fitted ARMA(2,2) model. The results are shown in figure 1 below.

The FFNN produces the optimal predictions. The GP is unable to model the non zero mean (as expected for most kernels). Since GP with standard kernels cannot model non stationary processes, we will not consider them further.

ARIMA(2,1,2) and FFNN

We consider the times series $Z_t = \sum^{t} Y_i$ which is non stationary. We train the FFNN directly without differencing to the stationary Y_t. The results are in figure 2 for different size windows.

There is much more variability in the final model arising from with starting with random weights. From a qualitative standpoint, we will interpret this as the increased difficulty of modelling the joint distributions of the integrated time series.

Autoencoders

We know from the above that there is a representation of the time series Z_t, namely $Z_t - Z_{t-1}$ that is easily modelled by a small FFNN. So we can now ask if we can arrive at such representations with unsupervised learning and in particular if the method of training the layers of a multiple layer AE recursively results a good final fit for our synthetic data set.

We train two AE with one and two hidden layers each and then use the encoded data for the supervised training of a FFNN with hidden layer size 5. No further fine tuning is done. NN indicates where we train the overall network corresponding to one AE stack as a single network. The results are below in figure 3.

We see that the method of training each layer separately, does not produce a better result than training the network as a single entity. Further, if fine tuning the AE networks starting from the parameters obtained by training each layer separately are in the same basin of attraction as the parameters obtained by training the network as a whole, we will obtain results as shown on the plot.

We also plot the decoded representation of the time series along with the original times series in figure 4. We see that the effect of the AE is largely to smooth the time series. This will not remove the predictive difficulties associated with the nonstationarity. This concludes our experiments with synthetic data which we used to evaluate the difficulty of training NN with and without AE on data from a well understood time series. We will use our understanding of the tools and techniques employed above to investigate observed data.

Global Energy Forecasting Competition - Wind Forecasting on kaggle.

Initial experiments and observations

The data set and predictive challenge is described at www.kaggle.com/c/GEF2012-wind-forecasting. The problem is to predict the power output of seven wind farms in given 48 hour windows from 48 hour wind forecasts. Training data is provided in which 48 hour wind forecasts are given every 12 hours, the test data only has forecasts that would be available when the test prediction period starts. The wind forecasts are given as x and y (planar) components of the wind, and the polar (r, θ) representation is also included in the data set (or in the notation of the datasets provided u,v,ws,wd).

As a first experiment, we assume that the observed wind is normally distributed around the prediction with the variance increasing with the lead time, justifying the use of the latest prediction as the best estimator of the wind speed at any time a power prediction was required. The feature vector is then the x, y, r, θ of the latest forecast. We also add windows of previous power observations to each of the seven predictors. The results are given as Window-SP.

Window-SP	RMSE	Window-TP	RMSE	Window-TF	RMSE	INT-α	RMSE
0	0.17887	0	0.16618	5	0.16878	$\alpha = 1.0$	0.16301
6	0.18132	5	0.16878	12	0.17181	$\alpha = .775$	0.16368
12	0.18064	12	0.17181			$\alpha = .75$	0.16132
24	0.18396					$\alpha = .6$	0.16287
36	0.18739					$\alpha = .5$	0.16217

From this it is clear that using the windowing technique with power observations to capture autoregressive properties of the time series of power generation does not increase the accuracy of the predictor.

Now we predict the power output of each of the seven wind farms simultaneously using the seven most recent predictions in an attempt to capture any correlations that may be present. This predictive method yielded RMSE of 0.16618 or 25/134 on the leaderboard (the top score was 0.14567) indicating that we are benefitting from the covariance structure of the seven farms. Windows consisting of previous power observations and forecast observations are added. The results are Window-TP and Window-TF.

To refine our predictive model we have to address the issue related to the lead in the wind forecast for a required forecast date. Because of the way the forecast data is supplied, any date in the training set will have a forecast that is at most 12 hours old but dates in the test set can have most recent forecasts that are 48 hours old. As a next experiment we will construct a test set where the lead in the forecast time is a feature of the training examples. This only results in a small improvement to 0.16558.

We will use another approach to refine our model. Until now we have supposed that the observed wind is normally distributed around the latest prediction. The way that the wind forecasts are presented means

there are times T where the prediction for T has a long lead but T is very close to a date S for which there is a prediction with lead of one hour (giving a very good indication of the wind at time S). Using a simple linear interpolation to modify the forecasts does not improve the model, which obtains 0.24955. Thus we introduce a decay term so that the influence of the S forecast is more local. Thus if F_T is a recent forecast for time T and F_{T-j} is an old forecast for $T-j$ we use $Ke^{-\alpha j}F_T + (1 - Ke^{-\alpha j})F_{T-j}$ for the forecast at $T-j$. With such a term α the results are in the table above under INT$-\alpha$ We also add the time of day and day of year to capture seasonal effects. With $\alpha = .75$ we obtain an improvement to 0.16044. In what follows we will include these seasonal features.

Random Forests and Boosted Decision Trees

RF α	RMSE	GBM-5CV	RMSE
0.75	0.15529	1000	0.16485
0.6	0.15563	5000	0.16196
0.5	0.15420	10000	0.16283
0.4	0.15560		

Having settled on feature vectors which give good performance, we will experiment with the underlying regression technique. Using a two layer NN has no effect on performance nor does predicting each farm individually from the whole feature vector. Using the same α parameterized interpolation above with a Random Forest of 250 trees we obtain the results in the table. With gradient boosting we show the results in the table with 5 fold cross validation.

Autoencoders

In an effort to further improve performance, we will address the issue of the lead in the forecast. We will use an autoencoder on the time series of the most recent forecasts. We will do this for each component separately and then recalculate the polar representation. The motivation here is to obtain a representation of the forecasts in the test sets consistent with the forecasts in the training set and also interpolate the data as above. This results in 0.26672 which is comparable with linear interpolation with no decay factor. In what follows we use the α interpolation scheme above as we require the smoothing to be local for good performance.

WINDOW-α-TP	RMSE	WINDOW-α-AE1	RMSE	WINDOW-α-AE2	RMSE
2	0.16049	8	0.23306	8	0.22929
4	0.16445	16	0.23621	16	0.23870
8	0.16454	32	0.23858	32	0.24208

If we use as a feature vector the planar forecast of each wind farm, we can again use the windowing technique to provide a temporal context. However the length of the feature vector will be 14 times the length of the window (here we omit the polar representation). For small window lengths the results are in the table above. For larger window lengths, we use an autoencoder on the features for dimensionality reduction and non supervised feature selection. The autoencoder is trained with L-BFGS and we optimise over the whole training set and not smaller batches as the number of examples is small. For one and two hidden layers, the results are under WINDOW-α-AE1,WINDOW-α-AE2 respectively.

Conclusions

The best result is achieved with a random forest and α interpolation giving RMSE 0.15420. This result places in the top 10%. From the experiments for both the wind data and the synthetic data above, we failed to achieve any performance gains with multilayer NN with the use of windowing to provide temporal context.

It may be that the method is more suited to time series where there are more directly observable predictive variables rather than time series where there are few predictive variables such as he ones considered here.

References

[1] Volker Tresp, Reimar Hofmann, *Graphical Models: Foundation of Neural Computing*. Eds, M Jordan and T. Sejnowski MIT Press, Massachusetts, 2001.

[2] Jerome T. Conner, R. Douglas Martin, L. E. Atlas, *Recurrent Neural Networks and Robust Time Series Prediction*. IEEE Transactions on Neural Networks, Vol. 5 No. 2 March 1994.

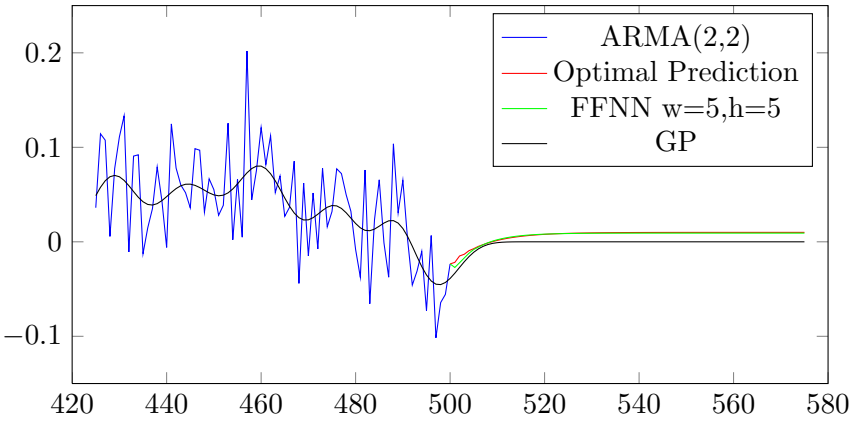

Figure 1: FFNN and GP

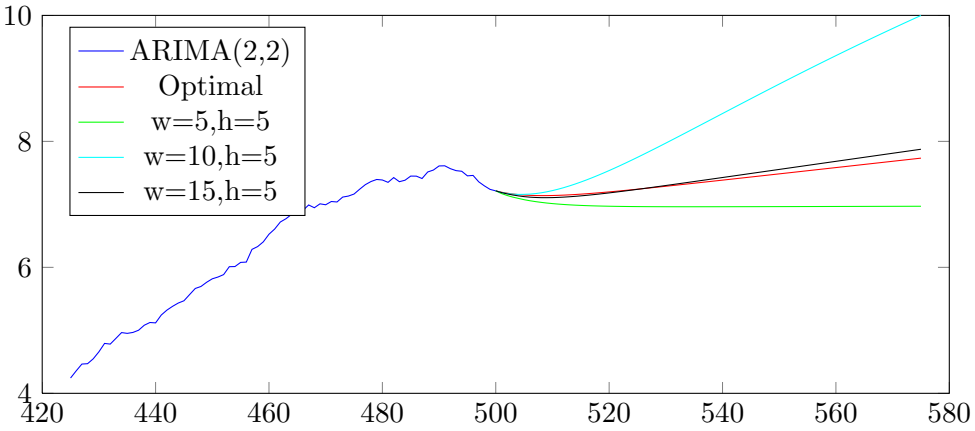

Figure 2: FFNN and ARIMA

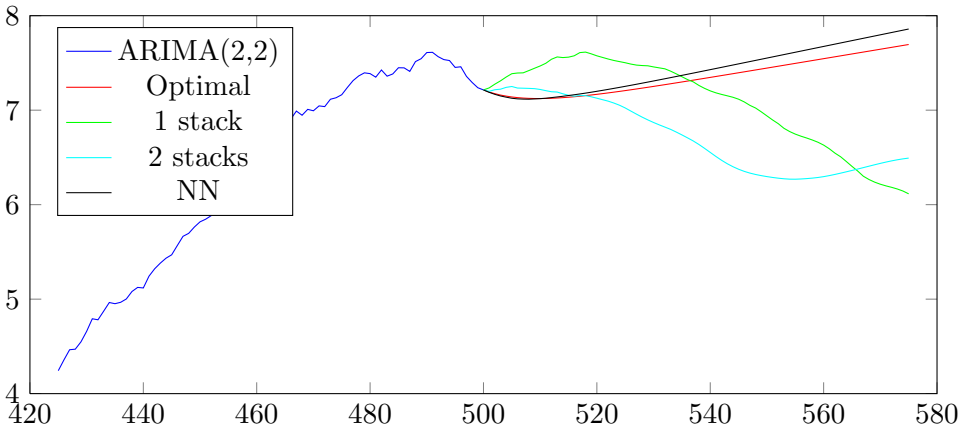

Figure 3: AE predictions

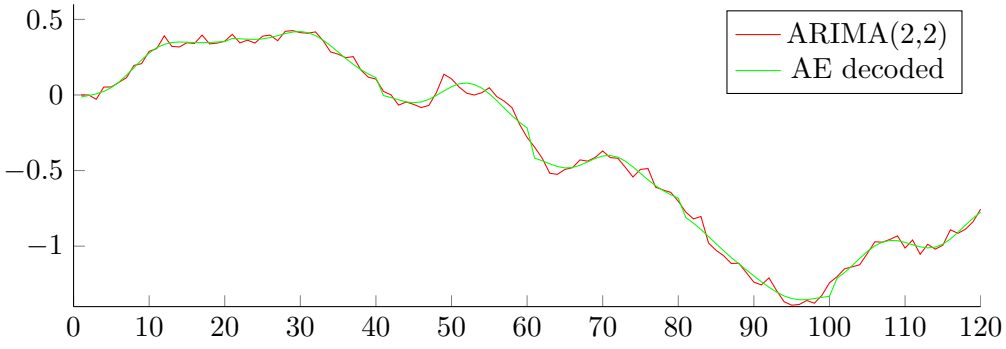

Figure 4: Result of AE