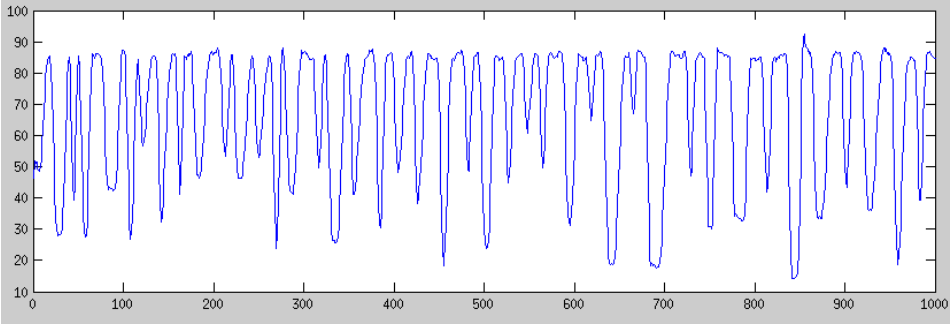
Time Series Ahead Prediction in Neural Network (N Point Ahead Prediction) Large Scale Iterative Training

(N=90) Point ahead Prediction using Neural Network:

I am trying to predict 3 minutes ahead i.e. 180 points ahead. Because I compressed my time series data as taking the mean of every 2 points as one, I have to predict (N=90) step-ahead prediction.

My time series data is given in seconds. The values are in between 30-90. They usually move from 30 to 90 and 90 to 30, as seen in the example below.



My data could be reach from: https://www.dropbox.com/s/uq4uix8067ti4i3/17HourTrace.mat

I am having trouble in implementing neural network to predict N points ahead. My only feature is previous time. I used elman recurrent neural network and also newff.

In my scenario I need to predict 90 points ahead. First how I separated my input and target data manually: For Example:

```
data_in = [1,2,3,4,5,6,7,8,9,10]; //imagine 1:10 only defines the array index valu
N = 90; %predicted second ahead.
P(:, :)         T(:)     it could also be(2 theta time)  P(:, :)         T(:)
[1,2,3,4,5]    [5+N]              |                     [1,3,5,7,9]     [9+N]
[2,3,4,5,6]    [6+N]              |                     [2,4,6,8,10]    [10+N]
...
```

until it reaches to end of the data

I have 100 input points and 90 output points in Elman recurrent neural networks. What could be the most efficient hidden node size?

```
input_layer_size = 90;
NodeNum1 =90;

 net = newelm(threshold,[NodeNum1 ,prediction_ahead],{'tansig', 'purelin'});
net.trainParam.lr       = 0.1;
net.trainParam.goal     = 1e-3;
```

//At the beginning of my training I filter it with kalman, normalization into range of [0,1] and after that I shuffled the data. 1) I won't able to train my complete data. First I tried to train complete M data which is around 900,000, which didn't gave me a solution.

2) Secondly I tried iteratively training. But in each iteration the new added data is merged with already trained data. After 20,000 trained data the accuracy start to decreases. First trained 1000 data perfectly fits in training. But after when I start iterativelt merge the new data and continue to training, the training accuracy drops very rapidly 90 to 20. For example.

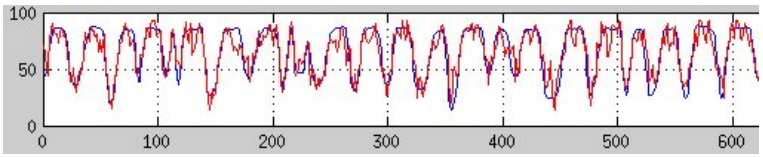
```
P = P_test(1:1000) T = T_test(1:1000) counter = 1;
     while(1)
      net = train(net,P,T, [], [] );%until it reaches to minimum error I train it.
      [normTrainOutput]    = sim(net,P,         [], [] );

      P = [ P P(counter*1000:counter*2000)]%iteratively new training portion of th
    counter = counter + 1; end
```

This approach is very slow and after a point it won't give any good resuts.

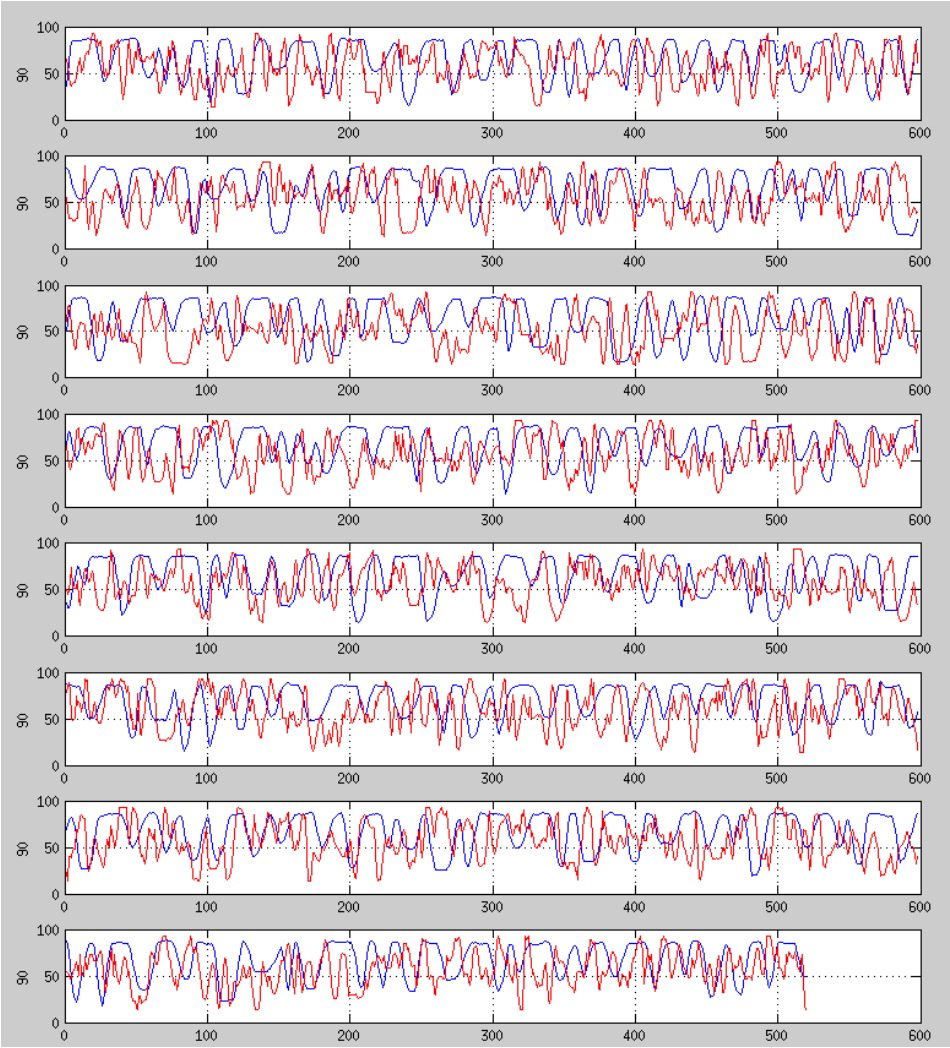
My third approach was iteratively training; It was similar to previous training but in each iteration, I do only train the 1000 portion of the data, without do any merging with previous trained data.For example when I train first 1000 data until it gets to minimum error which has >95% accuracy. After it has been trained, when I have done the same for the second 1000 portion of the data;it overwrites the weight and the predictor mainly behave as the latest train portion of the data.

```
> P = P_test(1:1000) T = T_test(1:1000) counter = 1;
      while(1)
>       net            = train(net,P,T, [], [] ); % I did also use adapt()
>       [normTrainOutput]    = sim(net,P,         [], [] );
>
>       P = [ P(counter*1000:counter*2000)]%iteratively only 1000 portion of the d
>     counter = counter + 1;
end
```

Trained DATA: This figure is snapshot from my trained training set, blue line is the original time series and red line is the predicted values with trained neural network. The MSE is around 50.



Tested DATA: On the below picture, you can see my prediction for my testing data with the neural network, which is trained with 20,000 input points while keeping MSE error <50 for the training data set. It is able to catch few patterns but mostly I doesn't give the real good accuracy.



I wasn't able to successes any of this approaches. In each iteration I also observe that slight change on the alpha completely overwrites to already trained data and more focus onto the currently trained data portion. I won't able to come up with a solution to this problem. In iterative training should I keep the learning rate small and number of epochs as small.

And I couldn't find an efficient way to predict 90 points ahead in time series. Any suggestions that what should I do to do in order to predict N points ahead, any tutorial or link for information.

What is the best way for iterative training? On my second approach when I reach 15 000 of trained data, training size starts suddenly to drop. Iteratively should I change the alpha on run time?

==========

Any suggestion or the things I am doing wrong would be very appreciated.

I also implemented recurrent neural network. But on training for large data I have faced with the same problems.Is it possible to do adaptive learning(online learning) in Recurrent Neural Networks for(newelm)? The weight won't update itself and I didn't see any improvement.

If yes, how it is possible, which functions should I use?

```
net = newelm(threshold,[6, 8, 90],{'tansig','tansig', 'purelin'});
net.trainFcn               = 'trains';
batch_size                 = 10;
while(1)
       net = train(net,Pt(:, k:k+batch_size ) , Tt(:, k:k+batch_size)   );
end
```

machine-learning | neural-network | time-series | prediction | supervised-learning

edited Apr 16 at 0:03

community wiki
18 revs
Avatar

add comment

2 Answers

Have a look at Echo State Networks (ESNs) or other forms of Reservoir Computing. They are perfect for time series prediction, very easy to use and converge fast. You don't need to worry about the structure of the network at all (every neuron in the mid-layer has random weights which do not change). You only learn the output weights.

If I understood the problem correctly, with Echo State Networks, I would just train the network to predict the next point AND 90 points ahead. This can be done by simply forcing the desired output in the output neurons and then performing ridge regression to learn the output weights.

When running the network after having trained it, at every step *n*, it would output the next point (*n+1*), which you would feed back to the network as input (to continue the iteration), and 90 points ahead (*n+90*), which you can do whatever you want with - i.e: you could also feed it back to the network so that it affects the next outputs.

Sorry if the answer is not very clear. It's hard to explain how reservoir computing works in a short answer, but if you just read the article in the link, you will find it very easy to understand the principles.

If you do decide to use ESNs, also read this paper to understand the most important property of ESNs and really know what you're doing.

EDIT: Depending on how "predictable" your system is, predicting 90 points ahead may still be very difficult. For example if you're trying to predict a chaotic system, noise would introduce very large errors if you're predicting far ahead.

answered Feb 27 at 17:09

community wiki
user131521

I checked the examples on the paper and related work. All examples are based on MickeyTime series(delay=17). I couldn't find any example that I can attach my own input and output data set, independent from the mathematical equations. – Avatar Mar 6 at 2:05

1 The formulae are only used because we can compute the desired output easily and that facilitates testing. You could of course plug in your own data instead. On chapter 7 of this (neuron-ai.tuke.sk/bundzel/diploma_theses_students/2006/…) paper, there's an example of training an ESN to generate text. at each step n, you plug in the nth letter as input and the n+1th as output, so it learns to predict the next letter. there's no delay here but it might help to see how you can plug in your own data. – user131521 Mar 6 at 9:45

I tried to work on from simple example (reservoir-computing.org/node/129). It seems like it is only working for MickeyTime series(delay=17), which is form from a pattern. I wasn't able to use my own input time series, to make predictions. – Avatar Mar 8 at 20:01

Can you explain what exactly is going wrong? What are you doing, what do you expect and what is actually happening? If you want to discuss on details, you can also send me an e-mail at tomaspllaha at gmail dot com. ESNs should work very well on any time series prediction task as long as the network size is big enough, the state update functions are correct, and the input and output is represented appropriately. – user131521 Mar 9 at 22:14

add comment

use fuzzy logic using membership function to predict the future data. will be efficient method.

answered Jan 25 at 20:40

community wiki
pinak

add comment

Not the answer you're looking for? Browse other questions tagged machine-learning neural-network time-series prediction supervised-learning **or ask your own question.**