Echo state networks with multiple readout modules

Abstract: We propose a new readout architecture for echo state networks where multiple linear readout modules are activated at distinct time points to varying degrees by a separate controller module. The controller module, like the reservoir of the echo state network, can be initialized randomly. All linear readout modules are trained through simple linear regression, which is the only adaptive step in the modified algorithm. The resulting architecture provides modest improvements on a variety of time series processing tasks (between 5 to 50% in performance metric depending on the task studied). The novel architecture is guaranteed to perform at least as accurately as a conventional linear readout. It can be utilized as a general purpose readout method when augmentations to performance relative to the standard method is needed.

Function approximation methods seek to learn mappings from the input feature space x to the output feature space y. In parametric methods, a general mapping function is used which can approximate a wide variety of different functions depending on the precise value of its parameters. The values of the parameters are learned from example data. Perhaps the simplest parametric mapping function is linear regression, where the output is expressed as a linear combination of input signals. When linear regression fails to provide adequate precision, a common remedy involves calculating a nonlinear expansion f(x) of the input feature space and performing linear regression between f(x) and y [1]. For time series analysis, echo state networks (ESNs) provide one such expansion function [2]. ESNs have been successfully applied to a wide variety of time series processing problems such as chaotic time series prediction, nonlinear system identification and classification.
In general, the precise conditions on the true mapping function from x to y for which a linear readout of the nonlinear expansion is sufficient are not known for many expansion methods (but see 3). We first present a heuristic argument that a single linear readout is expected to be insufficient for some classes of data streams. Then, we propose an elaboration of the basic ESN design that provides greater expressive power than a single linear readout while retaining the property that the only adaptive step in the training process involves simple linear regression.

Motivation

For concreteness, we consider the task of predicting the future behavior of an animal. For many simple animals such as worms, flies and fish, a large fraction of their time is spent performing stereotypical behavioral programs such as crawling, grooming, walking, eating etc [4]. Every class of stereotyped behavior is conceptualized as motion along a low-dimensional manifold (typically 1D in the intrinsic coordinate system). Switching between programs corresponds to a switch from one manifold to another. The behavior of the animal can be described in terms of the magnitudes of its intrinsic coordinates (muscle activities) or some more easily observed proxies such as joint angles, body postures etc. For motion along some one-dimensional manifold, any intrinsic coordinate x evolves according to x=x1(t), where t represents both time and the natural manifold coordinate. Predicting future behavior dt time steps ahead requires finding a mapping from x1(t) to x1(t+dt). Because t is a function of x ($t=x1^{-1}(x)$), we can express t as a Taylor series in x:

$$t = \sum \frac{x^n}{n!} \frac{d^n x1^{-1}(0)}{dx^n}$$

where $x1^{-1}$ is the inverse function from x to t. As x is also a Taylor series of t:

$$x(t) = \sum \frac{t^n}{n!} \frac{d^n x1(0)}{dt^n}$$

we can write x(t+dt) as a Taylor series of x:

$$x(t+dt)=\sum \frac{\left(\left(\sum \frac{x^m}{m!}\frac{d^m x1^{-1}(0)}{dx^m}\right)+dt\right)^n}{n!}\frac{d^n x1(0)}{dt^n}$$

Such a Taylor series corresponds to a linear readout of a polynomial expansion of x.
When motion of the animal switches from one manifold to another, the predictive linear readout of the expansion generally changes, because the functional form of the derivatives (which are directly related to the values of the linear readout coefficients) also change. Thus, for perfect prediction of the trajectories of an object whose motion switches between manifolds, different linear readouts at different times are required.

The architecture

Extending the polynomial expansion analogy to Echo state networks, we might expect mappings of complicated dynamical systems to be better approximated by a combination of linear readouts used to different extents at different times. If the reservoir state at time t is row vector R(t), we can write the predicted output $y_p(t)$ as a linear sum of the predictions of individual readout modules $y_p(t)=\Sigma p_i(t)y_{pi}(t)$, where $p_i(t)$ is the time dependent module weight which lies between 0 and 1 and $y_{pi}(t)$ represents the prediction made by module i at time t. We restrict ourselves to a scalar output $y_p(t)$ for notational clarity, the generalization to a vector output is trivial.
Each $y_{pi}(t)=R(t)W_i$, where W_i is the column vector of linear readout weights for module i. The weights $p_i(t)$ are calculated by a separate softmax control module: $p_i(t)=\exp(R(t)*w_i)/\Sigma(\exp(R(t)*w_j)$. The lower case w_i denote the column vector of weights for the softmax of each module. For a suitable choice of w_i, each module i will dominate the the predicted output $y_p(t)$ for those time periods where $p_i(t)$ has a high value. This allows the individual readout modules W_i to de facto specialize on predicting outputs at certain epochs, while their possibly erroneous contributions will be ignored for other epochs where their $p_i(t)$ acquires low values.
The overall aim of the architecture is to reduce the squared error between the desired y and predicted output $y_p(t)$: $C=\Sigma(y^j-y^j_p)^2$, where sum over index j runs over all the training data examples. The weights w_i and W_i can in principle be trained by joint gradient descent to reduce the cost function. However, a simpler approach works just as well in practice wherein w_i are initialised randomly and all the W_i are trained jointly as a least squares problem.
If we fix the w_i, minimizing the cost function can be converted to an ordinary least squares regression problem. The column vector $y_p(t)$ can be expressed as a the matrix product R_p*W, where W is the columnwise concatenation of the individual readout modules $W=[W_1' \; W_2' \ldots W_n']'$. Row t of the matrix R_p corresponds to a row-wise concatenation of vectors $p_i(t)R(t)$: $R_p(t)=[p_1(t)*R(t) \; p_2(t)R(t) \; \ldots \; p_n(t)R(t)]$.
The p_i needed to form the matrix R_p are calculated based on the fixed w_i and reservoir neuron activities. The softmax weights w_i are initialized randomly from a uniform distribution between -1 and 1 and then multiplied by a constant. A good choice of the constant is critical for effective results. For low values of the constant, all weights $p_i(t)$ remain near the value 1/n during all time points, thus preventing the readout modules from specializing. For very high values of the constant, the weights $p_i(t)$ jump rapidly between 0 and 1, creating a lack of smoothness. For a suitable initialization, where weights $p_i(t)$ smoothly vary over time between values of 0.1 to 0.9, the network performance is very close to the best results achieved by gradient descent, while the time required to estimate the final W_i is dramatically shortened. Finally, we note that the new network is guaranteed to work as well as a single module readout. If all W_i are set to equal the single module readout W_s, then the prediction produced by the new network is equivalent to the prediction produced by W_s, because the weights $p_i(t)$ sum to unity for each time step.

Results

We first tested the performance of the new architecture on a behavior prediction task. We created a simulated lamprey, which switches between different behavior modes of swimming, digging and struggling. The lamprey was chosen because its multisegment body shows remarkably reliable behavior well-described by simple mathematical equations [5-7]. The body of the model lamprey consisted of 20 segments. The equation describing swimming was written as $a_i=\sin(2\pi i/l-2\pi t/T)$, where i is segment number, ai the activity of segment i and l and 1/T are wavelength and frequency of the wave respectively. Like in the real lamprey, the wavelength was kept fixed while the period T varied between 20 and 200. The equation for digging was $a_i=\sin(2\pi i/40)\sin(2\pi ft)$, where f varied between 0.05 and 0.0125 cycles/s. In the real lamprey, struggling consists of a periodic contortion of the body into convoluted shape, but is less well understood mathematically. We chose to model struggling using the equation $a_i=\sin(2\pi(i-10)/(20+10\sin(2\pi ft))$; see figure 1 for graphical summary of all modes. A small amount of uniform random noise with amplitude 0.1 was added to each a_i at all time steps.

Each behavioral mode had a duration of 300 time steps. After the completion of 300 time steps, the lamprey switched between the modes randomly. For each 300 step epoch, the equation control parameters 1/T or f remained fixed, but between epochs the parameters switched randomly.

The task of the echo state network was to predict the body posture of the model lamprey 10 time steps ahead of the present posture of the lamprey. For this task it was trained with a randomly initialized time series of 10000 time steps and test error was measure on a differently initialized time series of equal length and a linear model was fitted to relate R(t) (calculated at each time step using R(t-1) and all the $a_i(t)$) to each $a_i(t+10)$. For a reservoir with 100 neurons (connection probability 0.05, largest connectivity matrix eigenvalue 0.95), which was augmented with a further set of 100 features found by calculating the square of each reservoir neuron activation at every time step [8], the echo state network achieved a normalized mean square prediction error 0.360+/-0.014 (for worm motion prediction all nmse reported as the averages of 50 random reservoir initializations +/- s.e.m), while a 2-module readout give an improvement of 30% (nmse 0.25+/-0.0065) and a 3-module readout give an improvement of 35% (0.234+/-0.007) compared to the single module prediction.

A 3-module readout has 600*20 adjustable free parameters compared to the 200*20 free parameters of a single module architecture. Another possible comparison can thus be made to a 300 neuron reservoir with quadratic feature augmentation, a model which also has 600*20 adjustable parameters. The mean test set nmse on 50 trials (both input and reservoirs randomly reinitialized at each trial) gave a prediction improvement of 35% (0.233+/-0.009) relative to the 100 neuron single-readout reservoir- a result statistically indistinguishable from the improvement given by the 3-module readout with a 100 neuron reservoir. However, the 100 neuron reservoir with 3 readout modules has a considerably smaller run-time complexity. To get the next reservoir state, 300*300 multiplications must be carried out at run time for 300 neuron reservoir compared to the 100*100 multiplications that must be carried out for use of the 100 neuron 3-module method.

Three methods (feature augmentation with polynomials of reservoir activities, increasing the reservoir size or introducing multiple readout modules) give complementary improvements to each other. For equivalent number of introduced adjustable parameters, they provide comparable improvements to performance. For another example, a 100 neuron reservoir without feature augmentations gives a test-set nmse of 0.44+/-0.0095, which is improved by quadratic feature augmentation to 0.36, while a 2-module readout (without quadratic augmentation) gives an nmse of 0.38+/-0.015 and a 200-neuron reservoir with a single readout module and no quadratic augmentation gives an nmse of 0.35+/-0.012. Conceptually, the three methods of improving ESN performance provide different benefits. While they all increase the number of adjustable parameters, larger reservoirs provide a larger working memory

[9], while multi-module readouts allow prediction modules to fine tune their predictions for distinct phases of behavior. Both quadratic augmentation and multiple readout modules provide improved performance with only marginal increases in model run-time complexity, especially when compared to increases in reservoir size.

Next, we tested the new architecture on the standard Mackey-Glass (MG) chaotic time-series prediction task [10]. For MG(17) with reservoir size 1000, no significant improvements were found when comparing a 2-module method with the standard single module method. When the reservoir size was reduced to 500, the 2-module method gave significant improvements. We measure the performance of the method by calculating the time when the difference between the predicted and actual continuation became larger than 0.1, which we call the divergence time. The divergence time rose from 990 to 1500 when comparing the 1-module case with the 2-module system. Also, the standard deviation of the divergence time decreased significantly from 500 to 350 for the 2-module readout, this despite an increase of the divergence time itself for the 2-module case. A re-implementation of Jaeger 2004 (where ESN prediction of the MG system was first reported) with 1000 reservoir neurons found divergence time of around 1640 with a standard deviation of 400. Thus, a 2-module readout method is able to achieve performance that is only 10% inferior to a 1000 neuron reservoir but has a 4 times lower run-time complexity.

Another time series prediction problem involves forecasting electricity loads. We obtained freely available data from the 2012 Global Energy Forecasting Competition on kaggle.com. The dataset contained 20 timeseries of hourly loads for 20 different utility zones spanning a period of four years. Hourly temperature data was available for 11 stations as well. We used ESNs to predict the summed load of the 20 utility stations using as inputs the time of day, the month, and temperature signals. Table 1 shows the test set nmse values (average of 50 reservoir initializations) for a reservoirs with 100, 200 and 400 neurons and compares these to a 2-module readout and quadratic augmentation for N=100 and N=200 neuron reservoirs. For this dataset-comparing methods with equal numbers of free parameters-quadratic augmentation gives the largest improvements, followed by the 2-module architecture. Increasing the reservoir size performs worst in this dataset.

Table 1:

method	**reservoir size**	**num params.**	**nmse**	**s.d**
standard	100	100	0.168	0.0043
standard	200	200	0.1526	0.0065
standard	400	400	0.1373	0.0067
quad. Augmentation	100	200	0.111	0.0041
quad. Augmentation	200	400	0.107	0.003
2-module	100	200	0.1365	0.01
2-module	200	400	0.125	0.009

On the Santa Fe time series prediction task D, the 2-module readout gave a modest 5% improvement over the 1-module method for prediction delays between 2 to 5 time steps.

Finally, we tested the new architecture on a motor coordinate transformation task. Here, a model human with two joints had to move his arm between randomly chosen points on the x,y plane. A sequence of 500 points were chosen randomly within a circular quadrant that was within reach of the model arm. The task was to move the arm with uniform speed from one point to the next in 10 time steps, then hold the hand steady for 10 steps until moving forward to the next point. The input positions were specified in normalized x,y coordinates normalized to the 0 to 1 range, whereas the output was specified in radian joint angle coordinates from which 1.4 was subtracted to remove most of the mean. Performance of a single module ESN with 500 neurons gave a small NMSE of 0.017. A 3-module ESN improved

performance by 50%. Note: in this task the inertia of the model arm was not modeled, so the task could in principle have been solved by a feed forward, not a recurrent neural net. It was studied primarily because its sequences of dynamics and stability might pose problems for an echo state network whose neurons can show ongoing dynamics also during stable phases of the task.

Discussion and summary

Many simple tricks have been proposed to augment the performance of ESNs. These include increasing non-linearity by augmenting the non-linear expansion with polynomial functions of reservoir activities [8], increasing the reservoir size[10], averaging predictions from many reservoirs[10], introducing delay lines into the read out system [11], providing neurons with a diversity of time constants [11] and having the reservoir adapt to input statistics via intrinsic plasticity [12]. The new multi-module readout architecture proposed is in its simplest form equivalent to introducing an additional layer of fixed non-linearity into the readout layer for improved performance. However, in principle, the new layer of non-linearity is trainable and it might still be the case that for certain tasks, gradient descent of the softmax weights w_i produces far superior results to the random initialization. In its randomly initialized form, it is most likely useful as an out-of-the-box non-linearity which can be tried when the traditional tricks run short of providing the required performance. It is guaranteed to work as well as a single readout module or better with very little additional training cost. We have demonstrated its modest usefulness on a variety of data series processing tasks.

References:

1.Pattern recognition and machine learning. C.M.Bishop. Springer 2006. Chapter 3: Linear models for regression.
2.Reservoir computing approaches to recurrent neural network training. M. Lukoševicious, H. Jaeger. Computer Science Review. 2009
3.Real-time computing without stable states: A new framework for neural computation based on perturbations. W. Maass, T. Natschlaeger, H. Markram. Neural Computation. 2002.
4.Mapping the stereotyped behavior of freely moving fruit flies. GJ. Berman, DM. Choi, W. Bialek, JW. Shaevitz. Interface. 2014.
5.Fictive locomotion in the lamprey spinal cord in vitro compared with swimming in the intact and spinal animal. P. Wallen, T. Williams. J. Physiol. 1984.
6.Functional significance and neural basis for larval lamprey startle behavior. SN. Currie, RC Carlsen. JEB. 1987.
7.Adaptive variations of undulatory behaviors in larval lamprey: comparison of swimming and burrowing. Pagett, Gupta, McCellan. Exp. Brain Research. 1998.
8.Adaptive nonlinear system identification with Echo State networks. H. Jaeger. Advances in neural information processing systems. 2002.
9.Short term memory in echo state networks. H. Jaeger. GMD-Report 152, GMD - German National Research Institute for Computer Science. 2002.
10.Harnessing non-linearity: predicting chaotic systems and saving energy in wireless communication.
11.Echo state networks with filter neurons and a delay&sum readout. G. Holzman, H. Hauser. Neural Networks. 2010.
12.Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning. J. Steil. Neural Networks. 2007.

Order of figures: Figure 1, Figure 2, Figure 3

Figure text:

Figure 1: illustration of the three modes of worm behavior- swimming (a translating sinusoid), digging (a vibrating string) and struggling (a contracting deformation of the body). The delay between the blue and red and the red and black traces is seven time steps. In the prediction task, the time delay was 10 time steps.

Figure 2: example performance on the MG time series prediction (blue signals) for a 500 neuron reservoir with 2 and 1 readout modules respectively. The 2-module readout system clearly tracks the correct continuation (red signals) for a longer duration. The zero time point marks the end of training and the beginning of the prediction phase.

Figure 3: Prediction (blue) versus actual signal shown for test data for a 3-module readout system in the coordinate transformation task.