State-Denoised Recurrent Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Introduction

Noise robustness is a fundamental challenge for every information processing system. A traditional approach to handling noise is to design preprocessing stages that estimate and filter noise from an input signal [Boll, 1979]. More recently in machine learning, loss functions have been explored to achieve invariance to task-irrelevant perturbations in the input [Simard et al., 1992, Zheng et al., 2016]. Such methods are suitable for handling external noise, but we argue in this article that *internal* noise can be an even greater challenge. To explain what we mean by internal noise, consider a deep-net architecture in which representations are transformed from one hidden layer to the next. To the degree that the network weights are not precisely tuned to extract only critical features of the domain, irrelevant features may be selected and have the potential to interfere with subsequent processing. Recurrent architectures are particularly fragile because internal-state dynamics can amplify noise over the course of sequence processing [**?**]. In this article, we propose a suppression method that improves the generalization performance of deep nets. We focus on recurrent nets, where the potential benefits are most significant.

Our approach draws inspiration from a notably robust and flexible information processing system—the human brain. Although the brain operates in an unfathomably high-dimensional continuous state space, the (conscious) mind is driven to categorize and to interpret the world via language. We argue that categorization and interpretation processes serve to suppress noise and increase the reliability of behavior. Categorization treats instances that vary in incidental ways as identical: a chair is something you can sit on regardless of whether it is made of wood or metal. Language plays a similar role in cognition. For example, Witthoft et al. [2003] have demonstrated that color terms of one's native language influence a perceptual-judgment task—selecting a color patch that best matches a reference patch. The linguistic label assigned to a patch is the basis of matching, not the low-level perceptual data.

Suppressing variability facilitates *communication* in information-processing systems. Whether we are talking about groups of humans, regions within the brain, components of an articulated neural net architecture, or layers of a feedforward network, in each case information must be communicated from a *sender* to a *receiver*. To ensure that the receiver interprets a message as intended, the sender should limit its messages to a canonical form that the receiver has interpreted successfully in the past. Language serves this role in inter-human communication. We explore noise suppression methods to serve this role in intra-network communication in deep-learning models.

In the next section, we describe a recurrent neural network architecture for cleaning up noisy representations—an *attractor net*. We then propose integrating attractor networks into deep networks, specifically sequence-processing networks, for denoising internal states. We present a series of experiments on small-scale problems to illustrate the methodology and analyze its benefits, followed by experiments on more naturalistic problems which also show reliable and sometimes substantial benefits of denoising, particularly in data-limited situations.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

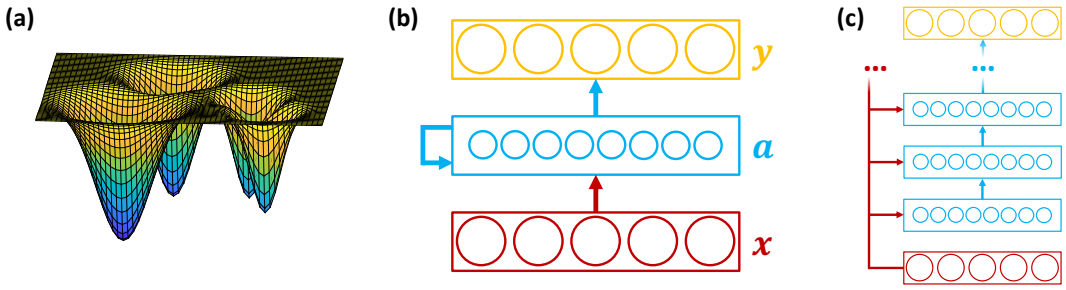

Figure 1: (a) energy landscape, (b) attractor net, (c) unfolded attractor net.

2 Noise Suppression Via Attractor Dynamics

The attractor nets we explore are discrete-time nonlinear dynamical systems. Given a static n-dimensional input $\boldsymbol{c}$, the network state at iteration k, $\boldsymbol{a}_k$, is updated according to:

$$\boldsymbol{a}_k = f\left(\boldsymbol{W}\boldsymbol{a}_{k-1} + \boldsymbol{c}\right), \tag{1}$$

where f is a nonlinearity and $\boldsymbol{W}$ is a weight matrix. Under certain conditions on f and $\boldsymbol{W}$, the state is guaranteed to converge to a *limit cycle* or *fixed point*. A limit cycle of length λ occurs if $\lim_{k\to\infty} \boldsymbol{a}_k = \boldsymbol{a}_{k+\lambda}$. A fixed point is the special case of $\lambda = 1$.

Attractor nets have a long history beginning with the work of Hopfield [1982] that was partly responsible for the 1980s wave of excitement in neural networks. Hopfield defined a mapping from network state, $\boldsymbol{a}$, to scalar *energy* values via an energy (or Lyapunov) function, and showed that the dynamics of the net perform local energy minimization. The shape of the energy landscape is determined by weights $\boldsymbol{W}$ and input $\boldsymbol{c}$ (Figure 1a). Hopfield's original work is based on binary-valued neurons with asynchronous update, but since then similar results have been obtained for certain nets with continuous-valued nonlinear neurons acting in continuous [Hopfield, 1984] or discrete time [Koiran, 1994].

We adopt Koiran's 1994 framework, which dovetails with the standard deep learning assumption of synchronous updates on continuous-valued neurons. Koiran shows that with symmetric weights, $w_{ji} = w_{ij}$, nonnegative self connections, $w_{ii} \geq 0$, and a bounded nonlinearity f that is continuous and strictly increasing except at the extrema (e.g., tanh, logistic, or their piecewise linear approximations), the network asymptotically converges over iterations to a fixed point or limit cycle of length 2. As a shorthand, we refer to convergence in this manner as *stability*.

Attractor nets have been used for multiple purposes, including content-addressable memory, information retrieval, and constraint satisfaction [Mozer, 2009, Siegelmann, 2008]. In each application, the network is given an input containing partial or corrupted information, which we will refer to as the *cue*f denoted $\boldsymbol{c}$; and the cue is mapped to a canonical or *well-formed* output in $\boldsymbol{a}_\infty$. For example, to implement a content-addressible memory, a set of vectors, $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots\}$, must first be stored. The energy landscape is sculpted to make each $\boldsymbol{\xi}_i$ an attractor via a supervised training procedure in which the target output is the vector $\boldsymbol{\xi}_i$ for some i and the input is a noise-corrupted version of the target, e.g., $\boldsymbol{c} = \boldsymbol{\xi}_i \circ \boldsymbol{\eta}$, where $\circ$ denotes the Hadamard product and $\eta_j \sim \text{Bernoulli}(0.5)$. Following training, noise-corrupted inputs should be 'cleaned up' to reconstruct the state.

In the model we described, the attractor dynamics operate in the same representational space as the input and output. Historically, this is the common architecture. By projecting the input to a higher dimensional latent space, we can design attractor nets with greater representational capacity. Figure 1b shows our architecture, with an m-dimensional input, $\boldsymbol{x} \in [-1, +1]^m$, an m-dimensional output, $\boldsymbol{y} \in [-1, +1]^m$, and an n-dimensional attractor state, $\boldsymbol{a}$, where typically $n > m$ for representational flexibility. The input $\boldsymbol{x}$ is projected to the attractor space via an affine transform:

$$\boldsymbol{c} = \boldsymbol{W}^{\text{IN}}\boldsymbol{x} + \boldsymbol{v}^{\text{IN}}. \tag{2}$$

The attractor dynamics operate as in Equation 1, with initialization $\boldsymbol{a}_0 = \boldsymbol{0}$ and $f \equiv \tanh$. Finally, the asymptotic attractor state is mapped to the output:

$$\boldsymbol{y} = f^{\text{OUT}}\left(\boldsymbol{W}^{\text{OUT}}\boldsymbol{a}_\infty + \boldsymbol{v}^{\text{OUT}}\right), \tag{3}$$

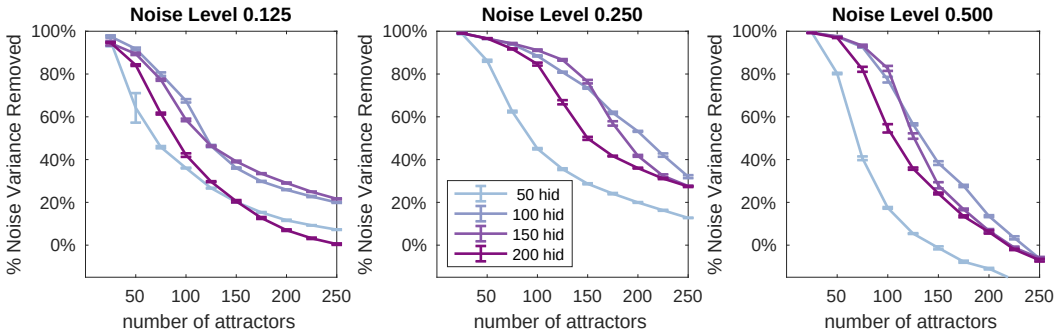

Figure 2: Percentage noise variance suppression by an attractor net trained on 3 noise levels. In each graph, the number of hidden units and number of attractors to be learned is varied. Evaluation is with noise level 0.250, 50-dimensional inputs, and 10 replications. Error bars indicate ± 1 SEM.

where f^{OUT} is an output activation function and the $\boldsymbol{W}^*$ and $\boldsymbol{v}^*$ are free parameters of the model.

One way to conceptualize the operation of this network is that $\boldsymbol{W}^{\text{IN}}$ copies the m input features forward and the attractor net uses m degrees of freedom in its state representation to maintain these *visible* features. The other $n - m$ degrees of freedom can be used as *latent* features that impose higher-order constraints among the visible features (in much the same manner as the hidden units in a restricted Boltzmann machine). When the attractor state is mapped to the output, $\boldsymbol{W}^{\text{OUT}}$ transfers just the visible features.

As Equations 1 and 2 indicate, the input $\boldsymbol{x}$ biases the attractor state $\boldsymbol{a}_k$ at every iteration k, rather than—as in the standard recurrent net architecture—being treated as an initial value, e.g., $\boldsymbol{a}_0 = \boldsymbol{x}$. Effectively, there are short-circuit connections between input and output to avoid vanishing gradients that arise in deep networks (Figure 1c). As a result of the connectivity, it is trivial for the network to copy $\boldsymbol{x}$ to $\boldsymbol{y}$—and thus to propagate gradients back from $\boldsymbol{y}$ to $\boldsymbol{x}$. For example, the network will implement the mapping $\boldsymbol{y} = \boldsymbol{x}$ if: $m = n$, $\boldsymbol{W}^{\text{IN}} = \boldsymbol{W}^{\text{OUT}} = I$, $\boldsymbol{v}^{\text{IN}} = \boldsymbol{v}^{\text{OUT}} = \boldsymbol{0}$, $\boldsymbol{W} = \boldsymbol{0}$, and f^{OUT} is the identity mapping or an identity mapping bounded at -1 and $+1$.

There is an alternative variation on the architecture that facilitates the pass through of $\boldsymbol{x}$ to $\boldsymbol{y}$ which consists of: the input $\boldsymbol{x}$ in Equation 2 being replaced with $\tanh^{-1}(\boldsymbol{x})$, $f^{\text{OUT}} \equiv \tanh$, and the nonlinearity in the attractor dynamics shifted back one iteration, i.e., Equation 1 becomes $\boldsymbol{a}_k = \boldsymbol{W} f(\boldsymbol{a}_{k-1}) + \boldsymbol{c}$. To ensure numerical stability, one can treat the input as $\tanh^{-1}(0.999\boldsymbol{x})$.

3 State-Denoised Recurrent Neural Networks

L2 regularization of attractor weights

for each minibatch, one update of the task weights, update the attractor weights for up to 10 steps until loss drops below 1.i

Training the attractor net: and y as the target output $x = s(s^{-1}(y) + \eta)$ with $\eta \sim \mathcal{N}(0, \sigma^2)$.

how do we detect stability? $|\boldsymbol{a}_{k+2} - \boldsymbol{a}_k|_\infty < \theta$

4 Simulations

4.1 Parity

Task: length $L = 10$ or $L = 12$ bit sequences train on all 2^L sequences. It would be nice to have two testing sets: (1) the same 2^L sequences with additional Gaussian noise added. Choose noise level so that standard RNN is not at floor or ceiling in performance (e.g., 75% correct). (2) longer noise-free sequences, of length $L + L\prime$, and chose $L\prime$ such that performance of the standard RNN is not at floor or ceiling. Note that the performance levels you have in your results so far are TOO LOW. I expect that if you make the task a bit easier, you'll see more difference between the different architectures.

Compare (a) basic tanh RNN, (b) RNN with attractor architecture, single training objective, (c) RNN with attractor architecture, trained for noise robustness. Unless you have a better name, call these: RNN, ARNN (attractor RNN), SDRNN (state denoised RNN).

100 replications. Compute Mean, median, std dev of generalization performance. Perform t-tests to determine reliability. Show separate results for the noisy sequences and the longer sequences.

NOTE: this simulation requires that you pick 4 model paramters: elative learning rate for attractor dynamics, relative noise level in attractor training, and forgetting (L2 regularization) rate, and the dimensionality of the attractor space. To tell a really nice story, you'd have found the optimum by searching in this 4D parameter space, and use this optimum for the basic parity simulation. Then for each of the simulations below in which you manipulate one parameter at a time, you'd freeze the other parameters to be the optima. Since it would be a huge simulation to pick the conjunction of the 4 parameters optimally, it's probably good enough for you to pick the parameters you expect to be best based on your past simulations.

4.1.1 Interpreting model

Show that SDRNN has lower entropy than RNN. I'm not sure of the best way to visualize. Try lots of things. You've made neat graphs showing entropy reduction over the course of training, but this may be TMI for a paper or talk. Perhaps just the entropy of the trained net, on both training and test sets?

For this simulation and the ones that follow, I can't say for sure that we'll want to present results from both the training and test sets. I think we need to make and see the figures to decide what we want to present and what best conveys our conclusions from a simulation.

4.1.2 Relative learning rate

Explore relative learning rate for attractor dynamics: vary from 0 to some upper bound. training and testing performance. 100 replications.

4.1.3 Attractor basins

Vary noise level, which affects the size of the attractor basins. Vary from 0 to some upper bound. training and testing performance. 100 replications.

4.1.4 Forgetting

Vary L2 weight decay of the recurrent attractor weights. Vary from 0 to some upper bound. training and testing performance. 100 replications.

4.1.5 Attractor dimensionality

Vary the number of units in the space where attractor dynamics are operating. You may have a better name than "attractor dimensionality". Let's look at training and testing performance. 100 replications.

4.2 Majority task

Figure out what length L to use, train on a small subset of the sequences, test on a large random subset. Compare the same 3 architectures: RNN, ARNN, SDRNN. (Remember, please use TANH neurons, not GRU.) For this simulation, we just need to compare generalization performance of the 3 models for a set of model parameters that you choose based on the parity simulation.

4.3 Simple finite-state machine

Let's use the architecture I studied that seemed to get reasonable results. If we want an alternative task, weould have a mult-class task which is to count the length of the longest sequence of 1's in the input, with all sequences having between 2 and 5. Maybe train on sequences of length $L = 15$ and a relatively small training set to make the task hard.

4.4 Larger, Naturalistic Data Sets

3-4 larger scale data sets: POS, Sentiment, Reuters Newswire topics classification, IMDB, etc. You don't need to explore model parameters, but you could use a validation set to pick the best SDRNN parameters. Don't cheat and pick based on the test set.

5 Relationship to Other Literature

Varieties of consciousness: access and phenomenal degrees of consciousness

Global workspace (Dehaene 2017 article)

Conscious states are interpretations (Dehaene 31. T. I. Panagiotaropoulos, G. Deco, V. Kapoor, N. K. Logothetis, Neuron 74, 924–935 (2012).j 32. N. K. Logothetis, Philos. Trans. R. Soc. Lond. B Biol. Sci. 353, 1801–1818 (1998).

stability – attractor states

Relationship to Ba's paper? - symmetric update rule will lead to attractor dynamics - learning within each sequence, vs. across sequences

http://www.offconvex.org/2018/02/17/generalization2/ - understanding deep nets in terms of noise suppression

A. Graves. "Adaptive Computation Time for Recurrent Neural Networks", CoRR, Vol abs/1603.08983. 2016.

? Learning with latent language – assumes pretrained language interpretation module using the space of natural language strings as a parameter space is an effective way to capture natural task structure.

Relation to Bengio Consciousness prior - proposal for a simplified or reduced description of an agent's complex, internal state that selectis a small subset of the information in the internal state, characterized in terms of factors, variables, concepts, dimensions - emphasis on low dimensionality (e.g., key-value representation where only one key is selected) - instead, i focus on coherence of state such that all the bits fit together, but may involve multiple factors (e.g., dog – hair, size, sound, etc.) - bengio focus is on isolating factors; my goal is to build sharper internal representations - bengio's hope is to have a conscious state that can be transformed into natural language sentences in a fairly simple manner. in my case, it's more of a criterion for what makes the conscious state. - reduced states. us: interpretations. can be complex but packaged as atoms

Consciousness as a posterior: prior is bias to familiar states, interpretation is posterior given evidence

References

S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, Apr 1979. ISSN 0096-3518. doi: 10.1109/TASSP.1979.1163209.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

P. Koiran. Dynamics of discrete time, continuous state hopfield networks. *Neural Computation*, 6(3): 459–468, 1994.

M. C. Mozer. Attractor networks. In T. Bayne, A. Cleeremans, and P. Wilken, editors, *Oxford Companion to Consciousness*, pages 88–89. Oxford University Press, Oxford UK, 2009.

H. T. Siegelmann. Analog-symbolic memory that tracks via reconsolidation. *Physica D: Nonlinear Phenomena*, 237(9):1207 – 1214, 2008. doi: https://doi.org/10.1016/j.physd.2008.03.038.

P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop - a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903. Morgan-Kaufmann, 1992.

N. Witthoft, J. Winawer, L. Wu, M. Frank, A. Wade, and L. Boroditsky. Effects of language on color discriminability. In R. Alterman and D. Kirsh, editors, *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society*, volume Pts 1 and 2, pages 1247–1252. Erlbaum Associates, 2003.

S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.