university of groningen / faculty of science and engineering

Hard Clustering of ESN Dynamics Using Conceptors

Bachelor's Project Thesis

Joris Peters, s4001109, j.peters.13@student.rug.nl,
Supervisors: G.Pourcel & Prof Dr H. Jaeger

Abstract: For data analysis, explainability, and more, it can be of interest to identify groups and hierarchies within the activity dynamics of recurrent neural networks (RNNs). Using conceptors to represent RNN dynamics, this study aims to cluster the dynamics of Echo State Networks (ESNs), a variant of RNNs. Conceptors were computed from the ESN-response to phoneme recordings of the TIMIT dataset. These conceptors were then used to perform phoneme classification with evidences (supervised), clustering with an adaptation of K-means (unsupervised), and hierarchical clustering with average linkage (partly supervised). Conceptor-based phoneme classification reached a reasonable [In addition to classification, I show that conceptors may be used for clustering into groups and hierarchies.] accuracy and clustering produced groups and hierarchies that resemble existing phonetical taxonomies. I conclude that conceptors are well-suited to classifying and clustering ESN-dynamics and the time-series that induced these dynamics.

1 Introduction

1.1 Echo State Networks

Jim Button clasped Luke's hand as they walked through the valley of twilight. He was petrified by the voices that sounded like hundreds of Jims murmuring at once through the valley. But Luke calmed him: "Don't worry, what you hear is just your echo". Similarly but less frighteningly, echo state networks (ESNs), under the right conditions, *echo* their input in form of a high-dimensional response.

What distinguishes ESNs from other Recurrent Neural Networks (RNNs) is their large size (many internal neurons), low density (low connections-to-neurons ratio), and randomly initialized and untrained internal- and input weights Yildiz, Jaeger, and Kiebel (2012). Figure 1.1 depicts a typical ESN as it is driven by an input sequence u (like Jim Button's voice) and elicits a high-dimensional response sequence x, the sequence of internal states of the ESN's reservoir (the echos in the valley). An output layer is commonly added to the ESN for (re)productive purposes but omitted here since only the internal states of the network will be of interest as will become apparent shortly.

Let us formalize the above on an example ESN.

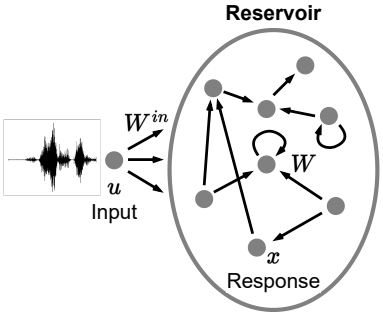

Figure 1.1: Sketch of an ESN driven with an input signal. State vector x is a high-dimensional response to the input signal u.

* Let N be the number of internal neurons. As mentioned, N will typically be large relative to the dimensionality d of the input. The input weight matrix $W^{in} \in \mathbb{R}^{N \times d}$, the bias vector $b \in \mathbb{R}^N$, and the internal weight matrix $W \in \mathbb{R}^{N \times N}$ are randomly initialized, the latter of which will typically contain many zeros to implement the internal sparsity of the reservoir. When driven by a discrete time-series u of length L, the ESN elicits an internal state sequence x of dimensionality N and of the same length as the input. The ESN's update equation is that of classical discrete-time RNNs:

$$x(n+1) = \tanh(Wx(n) + W^{in}u(n+1) + b), \quad (1.1)$$

where $x(n)$ and $u(n)$ are the internal reservoir state- and input column vectors at time step n and *tanh* is the hyperbolic tangent. We observe that any ESN state depends on the current input and the ESN's previous internal state. Moreover, the response x is, indeed, a high-dimensional expansion of the input if $N >> d$, and the system's dynamics are non-linear due to the application of tanh. [**Here you remarked "Clumsy: just say high dimensional expansion of the input... empirically and theoretically studied". Wdym with 'empirically and theoretically studied'?**] For later convenience, we define function r from the set of input sequences to the set of response sequences where $r(u)$ is the response that results from driving the ESN with u. The used ESN and the parameters will be clear from the context where r is used.

[Improve transition]

1.2 Conceptors

The primer on conceptors that is about to follow is based on the detailed report Jaeger (2014).

Definition4 and Intuition Loosely speaking, a conceptor is a "'fingerprint' of the activity of [a] network" over a period of time (Jaeger, 2014). Given any sequence of network states $x = (x(1), ..., x(L))$ that may have arose from running the above ESN †, the conceptor matrix C computed from x minimizes the following loss function $\mathcal{L}$:

$$\mathcal{L}(C|x, \alpha) = \sum_{n=1,...,L} ||x(n) - Cx(n)||^2/L + \alpha^{-2}||C||^2$$
$$C = \arg\min_C \mathcal{L}(C|x, \alpha), \quad (1.2)$$

where $\alpha \geq 0$ is the conceptor's aperture (further explained below) and $||C||^2$ is the sum of all of C's squared entries (squared Frobenius norm). The conceptor C that minimizes $\mathcal{L}(C|x, \alpha)$ may be analytically computed via the following procedure:

1. Concatenate the states in x column-wise in an $N \times L$ collection matrix $X = [x(1)|...|x(L)]$.
2. Compute the correlation matrix $R = XX'/N \approx \text{corr}(X)$.
3. Obtain the conceptor $C(R, \alpha) = R(R + \alpha^{-2}I)^{-1}$ with $N \times N$ identity matrix I.

The conceptor's capacity to *fingerprint* the sequence of states is reflected in its loss function. When minimizing $\mathcal{L}$, the term $||x(n) - Cx(n)||^2$ nudges C toward realizing an identity mapping for the states $x(n)$. However, the regularization term $a^{-2}||C||^2$ attaches a cost to the magnitude of its entries, limiting the numerical resources available to realize this identity mapping. Thus, the conceptor is drawn away from the identity matrix toward the zero matrix, especially on those axes that can account for little of the variance of x, i.e., where the minimization of $||x(n) - Cx(n)||^2$ is less beneficial. The amount of regularization applied to the conceptor depends on its aperture α. The higher the aperture, the closer the resulting conceptor maps all the members of x to themselves. The lower the aperture, the more selective the conceptor becomes, realizing a "good" identity-mapping only along the important axes that explain most of the sequence's variance.

A geometric perspective might extend this intuition. The reservoir states being a point cloud in state space $(0, 1)^N$, their conceptor can be represented as a hyperellipsoid of a similar shape as the point cloud but constrained to the unit circle of $\mathbb{R}^N$. The directions and lengths of the hyperellipsoid's axes are given by the singular vectors and values of

*For an index of some of the mathematical notations used throughout the paper, please refer to Appendix B

†We consider *sequences* of network states for practicality, but they can be thought of as *sets* because their states need not necessarily be ordered; any ordinal information is lost during the computation of conceptors. Neither do the states need necessarily stem from the same ESN run.

the conceptor, respectively, and "represent the principle components of the state cloud" (Vlegels, 2022). However, the shape of the hyperellipsoid slightly deviates from that of the point cloud as the regularization *squashes* it along its shorter axes. With a low aperture, the hyperellipsoid would be very squashed, extended only along a few main axes. With a large aperture, the conceptor would be little squashed and approach the unit hypersphere.

This squashing in space may be seen as a spatial compression as it systematically reduces the variance of the point cloud along its less important axes. Furthermore, conceptors are a temporal compression of ESN state sequences. Mapping sequences of vectors in state-space to conceptor-space turns variable-length objects into constant-sized objects. When the state sequence is a time-series, this mapping removes the temporal dimension, whence it may be seen as a temporal compression. [**Is there a better way to describe this compression?**]

Unused section (perhaps useful for intuition): [In practice, conceptors often capture meaningful features of the driving pattern or other circumstances that gave rise to the reservoir states. Any set of reservoir states occupies some linear subspace, some region, of the state space. For example, reservoir states produced by driving the ESN with an utterance of vowel a would populate a particular linear subspace. Under the right circumstances, some geometric [**better word: spatial?**] features of this linear subspace reflect "real", meaningful, features of this sound like the fact that it is a vowel. Responses to recordings of other vowels may fall close to this region, whereas the responses to recordings of consonants would tend to populate some different subspace. However, like a reflection in a dirty mirror, this information is hidden behind noise and distributed over the time steps of x. Here, a conceptor may serve to extract relevant, potentially meaningful, geometric features of the linear subspace while filtering out the less relevant, potentially noisy, parts. What parts of subspace are relevant depends on the task and it can be important to adapt the conceptor to the task by setting its aperture parameter.]

Conceptor operations The following section describes several useful operations on conceptors.

Similarity function Conceptors may be used to compute the similarities of state sequences. If C_a and C_b be the conceptors derived from the two sequences of states to be compared, their similarity may be defined as:

$$\mathrm{Sim}(C_a, C_B) = \frac{|(S_a)^{1/2}(U_a)'(U_b)(S_b)^{1/2}|^2}{|\mathrm{diag}(S_a)||\mathrm{diag}(S_b)|}, \quad (1.3)$$

where $U_a S_a (U_a)'$ is the SVD of C_a and $U_b S_b (U_b)'$ is the SVD of C_b. It is a function of the squared cosine similarity of the conceptors that measures the angular alignment between all pairings of singular vectors of the two conceptors weighted by the corresponding singular values.

Aperture The aperture of a conceptor can be set during its computation or when given a precomputed conceptor C [**Is it okay to use the name 'C' to refer to other conceptors than this one. Or do I need to name every object differently e.g., C_2.**], its aperture can still be adapted by any factor of $\gamma > 0$ using the aperture-adaptation function φ that returns the aperture-adapted conceptor C_{new}:

$$C_{new} = \varphi(C, \gamma) = C(C + \gamma^{-2}(I - C))^{-1} \quad (1.4)$$

Logical Operations on Conceptors Several logical operations have been meaningfully defined on conceptors. Given the arbitrary conceptors C and B, we have the following definitions and semantics:

1. **Negation ($\neg$)**

$$\neg C := I - C \quad (1.5)$$

   It returns a conceptor that describes the linear subspace complementary to that of C.

2. **Conjunction ($\wedge$)**

$$C \wedge B := (P_{R(C)\cap R(B)}(C^\dagger + B^\dagger - I)P_{R(C)\cap R(B)})^\dagger \quad (1.6)$$

   See Appendix A.1 for details on the computation of $P_{R(C)\cap R(B)}$. It returns a conceptor that describes the intersection of the linear subspaces of C and B.

3. **Disjunction ($\vee$)**

$$C \vee B := \neg(\neg C \wedge \neg B), \quad (1.7)$$

by De Morgan's law. It returns a conceptor that describes the union of the linear subspaces of C and B.

Although a simpler method for computing con- and disjunction exists, that method relies on the inversion of B and C and thus fails when B or C contain singular values of 0. Such singular values may occur in practice, for example, due to rounding or through the negation of conceptors with unit singular values. Therefore, the above method is recommended whenever the absence of null and unit singular values cannot be ensured.

Conceptors in practice The following presents some previous use-cases of conceptors.

Classification Conceptors may be used for time-series classification. The concrete procedure is outlined in Section 2.4, but the guiding idea is the following. Under a condition termed the echo state property (ESP), the ESN responds uniquely to each input sequence, i.e., there is a functional relationship between inputs and responses **[Why is it not bijective?]**. Then, any classification of a reservoir response can be seen as a classification of the input sequence that induced that response; the reservoir response is a higher-dimensional representation of the input. Now, given a set of signals from a set of classes, one may compute a conceptor per class from the responses to the signals in that class. The closer any unforeseen response is to state cloud represented by any of the conceptor, the more likely it is of the conceptor's class. [A bit wordy] [Elaborate on ESP showing how ESN states relate to the input signals]

Jaeger (2014) demonstrated this method for time-series classification on the *Japanese Vowels* dataset to an accuracy of about 99.99%. **[Should I further describe this experiment?]** [Describe the advantages of this classification method.] [Perhaps, mention Vlegels (2022), Chatterji (2022)]

Leading in & RQ These promising results give hope to improving on other classical machine learning tasks with conceptors. The machine learning paradigm may be split into supervised, unsupervised, and reinforcement learning. Reinforcement learning will not be considered in this paper. Supervised learning is a type of algorithm that typically aims to model a function given a set of sample input-output pairs. Jaeger (2014) shows that conceptors can be successfully used on the supervised task of time-series classification. Unsupervised learning algorithms are trained on data without sample outputs. They are typically used to analyze the structure present in data, for example, clusters (groups), hierarchies of clusters, important axes, or relationships. In an attempt to extend preexisting unsupervised methods using conceptors, the current study aims to answer the question of *whether conceptors can be used to cluster ESN dynamics.*

Why ESNs Four aspects make ESNs attractive for this study. First, their large size renders explainability particularly challenging and working on it relevant. Time-complexity questions are important for the large ESNs and will be addressed in the discussion. Second, the insights gained on ESNs may hold for other types of RNN. Third, ESNs allow focus on the network's activities without the need for concern with their training. Fourth, ESNs, especially with the ESP, perform high-dimensional non-linear expansions of the input. Therefore, any clusters found among internal states may coincide with meaningful clusters among their respective inputs. On the one hand, this correspondence allows for evaluating the activity clusters using prior knowledge about clusters among the inputs. On the other hand, previously unknown clusters among the inputs could be thus discovered using the ESN responses. Thus, ESNs will be used for demonstration; yet, much of the following will generalize to other types of NNs.

1.3 Background

Conceptors have been used on unsupervised tasks before. (Bricman, Jaeger, and van Rij-Tange, 2022) **[You previously marked the Bricman citation without explanation.. is there something to be improved?]** [...] (Mossakowski, Diaconescu, and Glauer, 2019)[...]

However, these studies have several limitations. They do not provide generalized algorithms for clustering reservoir states and only implement fuzzy (soft) clustering.

In continuation to (Bricman et al., 2022) and (Mossakowski et al., 2019), the current study will

adapt two classical unsupervised learning algorithms, K-means and hierarchical agglomerative clustering (HAC), to cluster ESN dynamics. [Defend the choice of K-means and HAC ... HAC was done, since it is a well-known and relatively robust agglomerative clustering algorithm and provides...]

K-means K-means is a clustering algorithm aimed at identifying a set of K clusters among the input data. A partitioning is good when the data points tend to be similar to data points from their cluster and dissimilar to data points from other clusters. K-means works by iteratively assigning each data point to the nearest centroid - the mean position of all the data points in a cluster - and recalculating the centroids based on the newly formed clusters. This process continues until all centroids converged in their position or a maximum number of iterations is reached. Concretely, its implementation depends on three components besides hyperparameter K:

1. A *set of points*, typically, in Euclidean space.
2. A *function that provides the centroid of a given cluster*, typically, the mean of the points assigned to the cluster.
3. A *function that provides the distance from a point to a cluster*, typically, the Euclidean distance between the data point and cluster centroid. During the assignment step, points are assigned to their closest cluster minimizing this function.

`http://videolectures.net/epsrcws08_teh_hc/`

K-means makes several assumptions on the dataset, which can impact its performance: [Verify or remove if irrelevant]

- Data distribution: The data points are distributed around their respective centroids in a Gaussian or normal distribution.
- Feature Scaling: The features have the same variance and are equally important. Therefore, the data is often normalized or standardized beforehand.
- Independence of features: The features of the dataset are independent of each other, or equivalently, k-means may not work well with datasets where features are correlated.

[Generalize k-means to Bregman Divergences according to Banerjee, Merugu, Dhillon, Ghosh, and Lafferty (2005) if applicable]

HAC HAC is a clustering algorithm aimed at identifying a hierarchy of clusters among the input data. HAC works by initializing one cluster per data point and then iteratively merging the two closest clusters until a single cluster or a desired number of clusters is obtained. The result is a hierarchical structure of nested clusters that can be represented as a dendrogram. Concretely, its implementation depends on three components:

1. A *set of points*, typically, in Euclidean space.
2. A *function that provides the distance between two points*, typically, the Euclidean distance between the points.
3. A *function that provides the distance between two clusters*. A common choice is the mean distance between all pairs of points (as defined above) in the two clusters. This variant of HAC is coined average linkage.

[Briefing on time-series clustering using K-means and HAC: 1. Use-cases 2. Methods] However, while ESN state sequences are time-series, existing methods for clustering time-series may not be suitable to cluster ESN activities.

[Limitations of time-series clustering (may struggle with high dimensionality for infinite time series, lack of online algorithms that process as the data points arrive, problems with variable length time series which is mediocrely solved using DTW) and comparison to advantages of conceptor-based classification to forecast the advantages of the following methods]

1.4 Phonemes

The developed methods will be demonstrated on phonemic speech recordings (speech := spoken communication). Within a speaker's vocal tract, the place and manner of production can be varied (articulatory variations) to create different sounds (acoustic variations). From all possible sounds that humans may create, a fraction is used for speech. Furthermore, the space of speech sounds is commonly

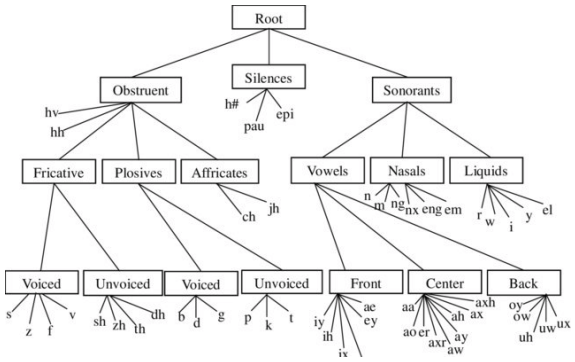

Figure 1.2: A taxonomy of all phones present in TIMIT Pfeifer and Balik (2011)

discretized into phones, the smallest still distinguishable units of speech. For example, [b], [ɾ], and [a][‡] are phones (recognizable by their square-bracketed narrow transcription), where each of them encompasses a subspace of speech sounds. Moreover, while listeners can distinguish between many phones, some of the differences are irrelevant for speech recognition, whence phones are commonly grouped into phonemes, the mental representations of phones. For every phoneme like /t/ (recognizable by their slanted-bracketed broad transcription), the phones associated with it ([t^h], [t], [ɾ], etc.) are termed allophones of that phoneme. Thus, phonetics – the branch of linguistics concerned with speech – has come up with ways to group speech sounds into phones and phones into phonemes, yet further organizations exist. For example, Figure 1.2 depicts a possible taxonomy of the phonemes present in the TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) which contains clusters like vowels and nasals[§].

The availability of this knowledge from phonetics makes phoneme recordings a good domain for evaluating clustering algorithms. These phonetic classes and taxonomies can be used as ground truth clusterings to determine the performance of the developed clustering algorithms. Conversely, the results from this study may provide an empirical verification or falsification of those phonetic theories; whereas the groups and taxonomies of phones were largely developed based on their articulatory features, the following clustering Experiments will investigate whether these groups can be found on a purely acoustic basis (phone recordings).

[Phonemes clustering literature on TIMIT] (refer to methods to avoid redundancy)

[Below (without supervision) vs above (with) phoneme]

Phoneme classification For clustering ESN responses to phoneme recordings, the ESN must be suited for the task. The following assumption that affects the experimental design choices is made; The more accurately activity groups can be distinguished by a supervised classifier, the better the used ESN is fit to responding to the data. It is somewhat like setting the aperture right to the sweet spot where signal to noise ratio is maximized. Because tuning the ESN's hyperparameters may increase classification performance **[citation]**, hyperparameters were optimized in Experiment 1. Thus, I assume that tailoring the ESN to the TIMIT data and classification task will improve clustering on the same data. Moreover, performing the classification task in Experiment 1 has the positive side-effect of demonstrating the application of conceptors to the phoneme classification on the TIMIT dataset, whereas a previous study applied conceptors to TIMIT for phoneme recognition Chatterji (2022). The difference between phoneme classification and recognition is that the former estimate the phonemes of pre-segmented signals (one label per segment), whereas the latter estimates the phonemes within unsegmented speech signals (multiple labels per signal over time) Lopes and Perdigao (2011). The following classification methods build on that of Jaeger (2014). The best accuracy for phoneme classification on TIMIT is held by **[...]** that used **[...]**.

Conclusion Concluding, the main aim of this study will be to identify groups and hierarchies within the activity dynamics of ESNs. Success will be if (1) such structures could be successfully identified within a reasonable time and (2) they correspond to groups that are to be expected given the input to the networks. Concretely, Experiments 2. and 3. will attempt to apply the K-means and HAC, respectively, to cluster ESN dynamics using conceptors. Experiment 1 will be a classification

[‡]Following the Internation Phonetic Alphabet (IPA)

[§]The TIMIT labels, also shown in Figure 1.2 and listed in Table A.1, are from IPA, although not written in Greek letters. The translation keys are given in the documentation.

task used to find the right hyperparameters to be used in Experiments 2 and 3.

2 Methods and Results

2.1 Dataset

The TIMIT Acoustic-Phonetic Continuous Speech Corpus was chosen as the data source for its diverse and phonetically annotated speech signals. Each of 630 American native speakers (from eight dialect regions and of which 30% were female) read ten sentences – five phonetically-compact ("SX" in the dataset), three phonetically-diverse ("SI"), and two dialect ("SA") sentences – amounting to a total of 6300 utterances. Each utterance comes with a phonetic transcription that indicates which of 64 phones is uttered at any time of the signal. Moreover, the corpus is split into a training (73 % of the speech material) and a test set, which were used as such in the experiments (Experiment 3 only uses the training set).

2.2 Pre-processing

For all experiments, the following pre-processing steps were performed. The utterances were segmented according to the phonetic transcriptions into $n = 241225$ segments ($n_{train} = 177080$ and $n_{test} = 64145$), each a vocalization of one phone (Figure 2.1). From each segment, the first $d = 14$ Mel Frequency Cepstral Coefficients (MFCCs) were extracted. This representation ought to isolate the information most relevant to speech analysis. To compute the MFCCs, the Librosa Python library (McFee, Raffel, Liang, Ellis, Mcvicar, Battenberg, and Nieto, 2015) was used with one MFCC vector computed every 1 ms from a 2.5 ms long sliding window.

The resulting time series were normalized in amplitude and time. First, the amplitudes varied strongly across the channels [the lowest mean amplitude of an MFCC channel ($\mu_1 = -593.2$) is 24.2 times lower than that of the second lowest channel ($\mu_4 = -22.4$)]. To grant each MFCC a similarly strong effect on the ESN dynamics under the identically distributed input weights, each channel was normalized to a range of $[-0.5, 0.5]$ across all samples. Second, to account for differences in utterance

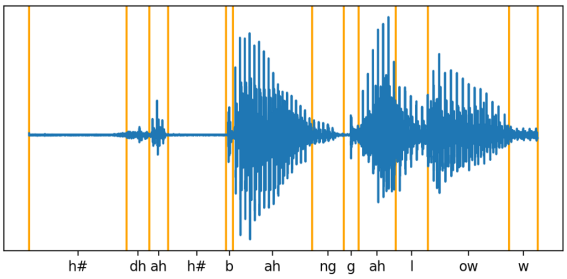

Figure 2.1: Example of the twelve first segments of one of the utterances. [will be identified more precisely]

speeds, the series were normalized in time by fitting each channel with a cubic spline and sampling it at $L = 10$ equidistant points.

Lastly, the phonemic labels (transcriptions) p_i $(i = 1, ..., c)$ were mapped from the original set of 61 phones to a subset of 39 phonemes P. Initially proposed by Lee and Hon (1989) and largely adopted by the field, this mapping amounts to folding stress-related variations and allophonic variations of phonemes (e.g., /em/ and /m/) into the same classes. The concrete mappings are given in appendix Table A.1, along with the distribution of the resulting classes within the dataset. The mapping serves to reach reasonable classification and clustering performances, render the computations feasible, and make the results comparable to previous studies.

Thus, the resulting data consisted of tuples $D = \{(s_i, p_i)|i = 1, ..., c\}$ with MFCC time series s_i, phone labels $p_i \in P$, and the set of phones P after folding ($|P| = 39$). The ready-made train-test split from TIMIT was used giving D_{train} and D_{test} of respective lengths n_{train} and n_{test}.

2.3 ESN

The following ESN was used for all three experiments. It consisted of $N = 100$ neurons with a connection density of $r = 10\%$. The entries of W^{in} and b were randomly sampled from a standard normal distribution and rescaled by factors of $k_{W^{in}} = 1.1$ and $k_b = 0.6$, respectively. W was obtained by random sampling from a standard normal distribution and rescaling the result to a spectral radius of $\rho = 2.57$. The spectral radius of an internal weight matrix like

W is its largest absolute eigenvalue. The larger ρ, the farther W transforms the internal state during the state update along its first eigenvector, which tends to lead to a more chaotic behavior of the dynamic system. An ESN's spectral radius may be adapted by rescaling its old internal weight matrix W_{old} to $W_{new} = \frac{\rho_{new}}{\rho(W_{old})} W_{old}$ where W_{new} has the desired spectral radius ρ_{new} instead of the previous ρ_{old}. Thus, despite the random initialization of its weights, some properties of the system are controlled. [**Effects on ESP**]

The above hyperparameters were picked by hand based on their effects on the accuracy in Experiment 1, previous research, and resource constraints. All parameters were initially set to the values used in the demonstration experiments of (Jaeger, 2014) (Section 4.1, p. 161). N was kept as a largest possible value still feasible under the available computational resources. Larger sizes would likely improve performance after adapting the other hyperparameters but may increase the risk for overfitting (Lukoševičius, 2012). The remaining hyperparameters, r, $k_{W^{in}}$, k_b, and ρ, were adjusted by hand with the objective to maximize the testing accuracy of phoneme classification in Experiment 1. Moreover, automated hyperparameters optimization was attempted but eventually not used due to its large computational cost and slow convergence. Its method and results are in the Appendix.

The resulting ESN was driven independently on each input signal s_i $(i = 1, ..., c)$ producing the reservoir state sequence x_i $(i = 1, ..., c)$. Concretely, each run started from the same state $x(0)$ sampled once from a standard normal distribution not to introduce meaningless between-sample differences while providing the network with an initial excitation. Indeed, using this normally distributed starting state led to a greater accuracy in Experiment 1 than when using the null vector as the starting state **[Can/Should I just state this without reporting the experiment? And is it tolerated to make such design decisions using the test set given the risk of overfitting?]**. The following states $x_i(t)$ $(t = 1, ..., L)$ were computed via update Equation 1.1 and collected column-wise in $N \times L$ matrix $X_i = [x_i(1)|...|x_i(L)]$. Concluding, an ESN response collection matrix X_i was computed for each training sample .

2.4 Experiment 1: Phoneme Classification

Objective The following experiment aimed to classify the pre-processed phoneme recordings using only their corresponding ESN responses. The purpose of this experiment was to tune ESN hyperparameters in preparation for the following clustering experiments and to demonstrate conceptor-based time-series classification on TIMIT. Concretely, the objective was to enable the classifier to assign the correct phoneme label to as many unforeseen speech samples of the test set D_{test} as possible.

2.4.1 Training

Training amounted to computing one *positive conceptor* C_p^+ and one *negative conceptor* C_p^- from the ESN states of the phoneme classes $p \in P$ [¶]. For each class p, its positive conceptor C_p^+ models [!] the linear state subspace that ESN states of class p tend to occupy, and it was computed as follows. Let η_p be the number of training instances of class p. The state collection matrices corresponding to signals of class p were concatenated column-wise into a phoneme-level collection matrix $X_p = [X_1|X_2|...|X_{\eta_p}]$ from which C_p^+ was computed with an initial aperture of $\alpha = 1$ by steps 2 and 3 of the procedure for conceptor computation. This was repeated for each class to obtain the set of preliminary positive conceptors $C_{pre}^+ = \{C_p^+ | p \in P\}$.

Aperture adaptation After computing the conceptors in C_{pre}^+ with the initial aperture of $\alpha = 1$, their apertures were adapted to the task. First, one aperture was chosen for all positive conceptors that would maximize the following optimality heuristic. The aim is to create a good balance between admitting information to the conceptor and limiting its size. This balance is reflected in the regularization term $||C||^2$ (for any conceptor C) that continuously changes, in a sigmoidal fashion [Test this, by plotting the values, and perhaps, add the figure], between N^2 (admitting all information) and 0 (admitting no information). We conjecture the optimal aperture to be found at the inflection point of $||C||^2$. [**I**

[¶]Not the pre-processed speech recordings s_i $(i = 1, ..., \eta_p)$ are used for classification but the states collected when driving the ESN with them.

should verify the above explanations.] To find the inflection point, we maximize the ∇-*criterion* (Jaeger, 2014), the gradient of the Frobenius norm of the aperture-adapted conceptor with respect to the logarithm of γ [‖]:

$$\nabla(C,\gamma) = \frac{d}{d\ \log(\gamma)}||\varphi(C,\gamma)||^2 \qquad (2.1)$$

To approximate the optimal γ_p for each positive conceptor C_p^+, various candidate values $\gamma_{candidate}$ in the interval $[0.001, 1000)$ were swept through. The $\gamma_{candidate}$ that maximized a numerical approximation of $\nabla(C_p^+, \gamma_{candidate})$ (with a delta in γ of $h = 10^{-4}$ **[Needs further explanation?]**) was taken for γ_p. $|P| = 39$ values γ_p ($p \in P$) resulted. Finally, the apertures of all positive conceptors were adapted using the mean $\gamma_{opt} = \frac{1}{|P|}\sum_{p\in P}\gamma_p$. Let C_{opt}^+ be the resulting set of aperture-optimized positive conceptors.

The reason for adapting all conceptors with the same γ instead of adapting each conceptor with their γ_p is to avoid classification bias. The larger a conceptor's trace (the sum of its singular values), the larger the expected *evidence* **[Should I add the proof to appendix?]**, the metric later used for classifying new sequences (further described under Testing). Thus, if there are differences between the traces of the conceptors, the classifier will be biased toward the classes whose conceptors have greater traces. Since the trace is a growing function of the conceptor's aperture [Add citation/proofx], equating the apertures between classes helps to debias the classifier.

However, although the conceptors were adapted to the same aperture, some variation in their traces remained (See the divergence of traces at $x = 1$ in Figure 2.2). To remove it, each conceptor's trace was adapted to reach a shared target value $\bar{tr}$, the mean trace among the conceptors:

$$\bar{tr} = \frac{1}{|C^+|}\sum_{C\in C_{opt}^+}\mathrm{tr}(C) \qquad (2.2)$$

To adapt each conceptor's trace to this target value with an error tolerance of ϵ (here, $\epsilon = 0.01$), Algorithm 2.1 was developed. The iterative procedure adapts the conceptor's aperture by a factor of the current trace error ratio until reaching the target trace. This works because, again, trace and aperture are positively related. We shall call $\psi(C, tr_{target}, \epsilon)$ the function implemented by the algorithm.

Algorithm 2.1 Adapt the trace of a conceptor

Require:
Conceptor C whose trace is to be adapted to tr_{target}
Target trace tr_{target}
Error tolerance ϵ

while TRUE **do**
  if $|tr_{target} - tr(C)| < \epsilon$ **then**
    break
  end if
  $\gamma \leftarrow tr/\mathrm{tr}(C)$
  $C \leftarrow \varphi(C, \gamma)$
end while

As shown in Figure 2.2 for $x > 1$, applying $\psi(\cdot, \bar{tr}, 0.01)$ to each of the conceptors in C_{opt}^+ normalizes their traces. Thus, via the above two aperture adaptation procedures, the apertures of the conceptors in C_{pre}^+ were optimized and their traces were normalized resulting in the final set of positive conceptors C^+.

Negative conceptors From C^+, the set of negative conceptors $C^- = \{C_p^- | p \in P\}$ was computed. Each negative conceptor models the linear state subspace complementary to those of all other classes, the region that states "not from any of the other classes" are expected to occupy, and it was thus computed:

$$C_p^- = \neg\bigvee\{C_q^+ | q \in P, q \neq p\}, \qquad (2.3)$$

where $\bigvee S$ is the disjunction of the $|S|$ conceptors in set S computed by the associative disjunction of its elements:

$$\bigvee S = ((C_1 \vee C_2) \vee C_3) \vee ... \ C_{|S|} \qquad (2.4)$$

2.4.2 Testing

The combined evidence E was used to classify unforeseen speech samples via the corresponding ESN responses. The combined evidence $E(x, p)$, that a

[‖] In this case, $\gamma = \frac{\alpha_{new}}{\alpha} = \alpha_{new}$, where α_{new} is the candidate target aperture and $\alpha = 1$ is the current aperture.

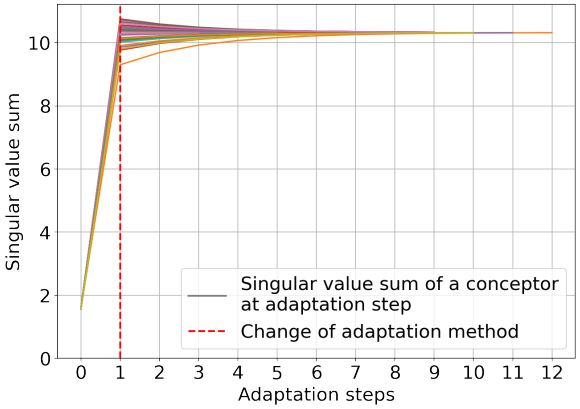

Figure 2.2: The changes in the traces of the conceptors in function of the adaption steps. The first adaptation step, at the red, dotted, line, is the aperture adaptation using the ∇-criterion. The increase in aperture caused the traces to increase and diverge. The next steps ($x > 1$) are the results of normalizing the traces using Algorithm 2.1. [Redo figure by replacing "sum of singular values" by "trace".]

state x corresponds to class p, is a measure of similarity between that state and the positive and negative conceptors of class p. Therefore, E is the combination of a positive evidence $E^+(x,p)$, computed using the positive conceptor C_p^+, and a negative evidence $E^-(x,p)$, computed using the negative conceptor C_p^-:

$$\begin{aligned} E(x,p) &= E^+(x,p) + E^-(x,p), \\ E^+(x,p) &= x' C_p^+ x \\ E^-(x,p) &= x' C_p^- x \end{aligned} \tag{2.5}$$

$E(x,p)$ is large when x is close to the linear subspace modeled [!] by conceptor C_p^+, but far from the linear subspace which the other conceptors model (see definition of C_p^-).

To classify a state x, the positive evidence is maximized over all classes:

$$\text{Class}'(x) = \arg\max_{p \in P} E(x,p) \tag{2.6}$$

To classify a point cloud X (a column-wise state collection matrix), the mean evidence for all states (columns of the collection matrix) is taken:

$$\text{Class}(X) = \arg\max_{p \in P} \frac{1}{L} \sum_{i=1}^{L} E(X[:,i],p), \tag{2.7}$$

where L is the width of X.

Finally, given a speech sample s_{new}, $\text{Class}(X_{new})$ returns the classification estimate, where $X_{new} = [r(s_{new})(1), ...|r(s_{new})(L)]$ is the column-wise concatenation of the states collected when driving the ESN with s_{new}.

2.4.3 Results

This testing procedure resulted in accuracies of 56.16% on the training set and 56.04% on the test set. In an attempt to improve the accuracy, the experiment was repeated in slight adaptation to the type of data (Appendix A) which led to a training accuracy of 63.64% but test accuracy of 49.13.

2.5 Experiment 2: Below-phoneme

2.5.1 Activity Clustering

In this and the next experiment, the traditional K-means and HAC algorithms were adapted to the clustering of ESN activities. Classically, these algorithms are used on data in Euclidian space; their components are well-defined for that space (mean, Euclidian distance, manhattan distance, etc.). However, the activities within the ESN are subjected to non-linear transformations like *tanh* at each time step as is the case for most neural networks. I hypothesize that Euclidian distance functions may not be well-suited to capturing the distances between network states. Moving to conceptor space will be an attempt to better represent, compare, and merge clusters of ESN activities.

Transfer of clustering in Euclidian space to clustering in conceptor-space [Add an illustration here: From signal space to state space to conceptor space to clusters conceptor space]

Objective The following experiment aimed to group the ESN responses to the pre-processed phoneme recordings into K clusters using an adaptation of the K-means algorithm. These clusters were then tested against the provided phonemic classes.

Dataset For computational constraints, only a subset $D' \subset D_{train}$ of $n' = 105$ samples was used for this experiment. To simplify the task, D' was constrained to 15 utterances of each of the phonemes

/ih/, /ae/, /ah/, /iy/, /aa/, /eh/, and /uh/. These phonemic classes were similar to those used in (Lerato and Niesler, 2012). Each utterance was randomly picked from the phonetically compact sentences (less phonetic variability) of 48 randomly sampled speakers with an equal male-female ratio and an equal ratio across the 8 dialect regions. Thus, whether the algorithm clusters by phoneme, speaker gender, or dialect region, the classical K-means assumption of equal-sized clusters was met.

Conditions The experiment was performed in a *Conceptor* and *Control* condition. For each condition, K-means' main components were customized as follows.

In the *Control* condition, the traditional version of K-means was implemented clustering the MFCC time series directly without using an ESN.

1. The set of points contains the unlabeled MFCC time series from D':

$$D'_{ctrl} = \{s_i | 0 < i \leq n'\} \tag{2.8}$$

2. Given a cluster Cl, its centroid is the mean of its members:

$$\text{centroid}_{ctrl}(Cl) = \frac{1}{|Cl|} \sum_{p \in Cl} p \tag{2.9}$$

3. The distance from point p to cluster Cl is the Euclidian distance between p and Cl's centroid:

$$d_{ctrl}(p, Cl) = ||p - \text{centroid}_{ctrl}(Cl)|| \tag{2.10}$$

In the *Conceptor* condition, an adaptation of K-means was implemented that uses conceptors as embeddings for the MFCC time series.

1. The set of points contains the conceptor embeddings of the MFCC time series (D'_{ctrl}) computed as follows. A new ESN with the same parameters as in Experiment 1 was driven independently with the MFCC time series s_i producing the set of responses x_i ($0 < i \leq n'$). From each x_i, a conceptor C_i was derived, like in Experiment 1, with a preliminary aperture of 1, to then be adapted to 21.4, the value that maximized the gamma criterion, and again adapted to normalize the traces among the conceptors. We get the following the set of points:

$$D'_{con} = \{C_i | i = 1, ..., n'\} \tag{2.11}$$

2. Given a cluster Cl, its centroid is the aperture-adapted disjunction of the cluster's members:

$$\text{centroid}_{con}(Cl) = \psi(\bigvee Cl, \bar{tr}_D, 0.01), \quad \text{where } \bar{tr}_D = \frac{1}{|D|} \sum_{C \in D} \text{tr}(C) \tag{2.12}$$

The conceptor $\text{centroid}_{con}(Cl)$ may equivalently be computed from the set of ESN responses that underlie the members of Cl, i.e., applying Procedure 1.2 to $\{r(s_i) | i = 1, ..., n'\}$ with an aperture of 1. With either approach, disjunction or recomputation, the resulting conceptor needs to be aperture-adapted to match the mean trace of the points in D'_{con} using Algorithm 2.1. Although the above computations would produce the same conceptor, the latter was used for being cheaper.

3. The distance from point p to cluster Cl is the complement with respect to 1 of the conceptor similarity of p and Cl's centroid:

$$d_{con}(p, Cl) = 1 - \text{Sim}(p, \text{centroid}_{ctrl}(Cl)) \tag{2.13}$$

The ranges of d_{con} and Sim are $[0, 1]$, and smaller values of d_{con} indicate a higher degree of similarity between point p and the centroid of cluster Cl. **[Big problem: Not a bregman divergence, because cosine similarity does not fulfill the triangle inequality. Therefore, this algorithm is not guaranteed to converge.]**

With these components set up for both conditions, hard clustering was performed. Thus, for each condition, its components (1,2,3) were substituted in the designated places of the generalized K-means algorithm. This algorithm uses the same iterative relocation scheme of K-means, but with additional input parameters for the centroid computation function and divergence function.

The centroids were initialized in a K-means++ fashion. With this method, the initial centroids are sampled from a distribution that tends to spread them evenly across the point space. Compared to the random centroid-initialization, as classically used in K-means, this method tends to converge faster and more consistently [cite AI2?].

Algorithm 2.2 Generalized K-means clustering algorithm. The numbers in brackets indicate the condition-specific components.

Require:
Number of clusters K
[1] Set of points $D = p_1, p_2, ..., p_n$
[2] Centroid computation function centroid(Cl)
[3] Divergence function $d(p_i, p_j)$

Initialize K cluster centroids via procedure 2.3: $\mu = \mu_1, \mu_2, ..., \mu_K$

while TRUE **do**
Reset all clusters: $Cl = \emptyset, \emptyset, ..., \emptyset$

Assignment step:
for p_i in D **do**
$k \leftarrow \arg\min_j d(p_i, \mu_j)$
Assign p_i to cluster k: $Cl_k \leftarrow Cl_k \cup p_i$
end for

Centroid update step:
for $j = 1$ to K **do**
$\mu_j \leftarrow \text{centroid}(Cl_j)$
end for

if Cluster assignments didn't change **then**
break (because converged)
end if
end while

return Set of clusters $Cl = Cl_1, Cl_2, ..., Cl_K$

Algorithm 2.3 K-means++ improved initialization of centroids.

Require:
Number of clusters K
Set of points $D = \{p_1, p_2, ..., p_n\}$
Divergence function $d(p_i, p_j)$

Choose first centroid μ_1 uniformly at random from D

for $k = 2$ to K **do**
For each point p_i, compute distance to nearest centroid**:
$d_{min}(p_i) = \min_{0<j<k} d(p_i, \mu_j)^2$

Choose $p_i \in D$ as the next centroid μ_k with probability $\frac{d_{min}(p_i)}{\sum_{j=1}^{n} d_{min}(p_j)}$
end for

return Set of centroids $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$

Evaluation The Normalized Mutual Information (NMI) and Silhouette Coefficient (SC) were used to evaluate the clustering performance.

First, the NMI measures the similarity between a clustering result $Cl = \{Cl_1, Cl_2, ..., Cl_K\}$, where Cl_k is the set of points assigned to cluster k, and the ground-truth phonemic groups $G = \{G_1, G_2, ..., G_{|P|}\}$, where G_p is the set of points with label p in the original dataset D'. Its values range from 0 for completely dissimilar clusterings and 1 for identical clusterings. Concretely, the NMI is computed as I, the mutual (shared) information between Cl and G, normalized by H, the mean entropy (uncertainty) within each clustering:

$$I(G, Cl) = \sum_{G_p \in G} \sum_{Cl_k \in Cl} P(G_p \cap Cl_k) \log \frac{P(G_p \cap Cl_k)}{P(G_p)P(Cl_k)}$$

$$H(G') = -\sum_{G'_p \in G'} P(G'_p) \log P(G'_p), \text{ for any clustering } G'$$

$$NMI(G, Cl) = \frac{I(G, Cl)}{\frac{1}{2}[H(G) + H(Cl)]} \tag{2.14}$$

Second, the SC measures the quality of a clustering Cl by comparing the cluster cohesion (intra-cluster divergence) and cluster separation (inter-

cluster divergence). Thus, it complements the NMI by not being informed by the original dataset and relying on the divergence functions used by each condition. The SC ranges from -1 to 1, where higher values indicate better clustering quality. It is computed as follows:

- Let $a(p_i)$ be the mean divergence, using either d_{con} or d_{ctrl}, between data point p_i and all other points assigned to the same cluster according to Cl.
- Let $b(p_i)$ be the smallest mean divergence between p_i and all points in some other cluster.
- $s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$ represents the silhouette value for each data point p_i.
- The overall SC for a clustering Cl is the mean silhouette value of all data points.

Run The experiment was run as follows. For additional evaluation of the algorithm, the numbers of clusters K was varied $K = 1$ to $K = 18$. For each K, the algorithm was run on each condition for two trials. The NMIs and SCs were averaged across the trials.

Results The NMI and SC for different values of K are shown in Figures A.1 and A.2, respectively. [..]

2.6 Experiment 3: Above-phoneme

The second hard clustering algorithm aims to structure the phonemic classes P in a hierarchy through an adaptation of HAC. Whereas Experiment 2 was concerned with finding clusters of individual samples, this experiment aims to build a hierarchy of the classes P of samples.

Method The following three components were used to adapt traditional HAC to the clustering of conceptors:

1. The set of points to be clustered consisted of one conceptor embedding per phonemic class of the corresponding samples in the training set. Therefore, the set of positive conceptors from Experiment 1 was reused:
$$D_{HAC} = C^+ = \{C_p^+ | p \in P\} \tag{2.15}$$
2. The dissimilarity measure between points p_i and p_j was the negative logarithm of the conceptor similarity of p_i and p_j:
$$d(p_i, p_j) = 1 - \mathrm{Sim}(p_i, p_j) \tag{2.16}$$
This function fulfils several desirable properties for a dissimilarity measure: It is non-negative, symmetric, and equals 0 for identical inputs.
3. The linkage function $d_{link}(Cl_i, Cl_j)$ between clusters Cl_i and Cl_j is the mean pairwise dissimilarity of the points in the clusters. This is the defining function of *average* linkage algorithms:
$$d_{link}(Cl_i, Cl_j) = \frac{1}{|Cl_i||Cl_j|} \sum_{p_x \in Cl_i, p_y \in Cl_j} d_{HAC}(p_x, p_y) \tag{2.17}$$

From these components, the HAC adaptation was performed as detailed in Algorithm 2.4.

Algorithm 2.4 Generalized HAC

[1] Set of points $D = \{p_1, p_2, ..., p_n\}$
[2] Dissimilarity function $d(p_i, p_j)$
[3] Linkage function $d_{link}(Cl_i, Cl_j)$

Initialize a cluster for each point: $Cl \leftarrow D$

while number of clusters > 1 **do**
  Find the two most similar clusters:
    $Cl_i, Cl_j \leftarrow \underset{Cl_i \neq Cl_j}{\arg\min}\, d_{link}(Cl_i, Cl_j)$
  Update clusters in Cl:
    Remove Cl_i and Cl_j
    Add $Cl_{merged} = Cl_i \cup Cl_j$
end while

Results The resulting hierarchy is depicted as a dendrogram in Figure 2.3. The following groups can approximately be identified within this dendrogram:

3 Discussion

Exp 1: Phoneme Classification But how is it that the hyperparameters from the classification

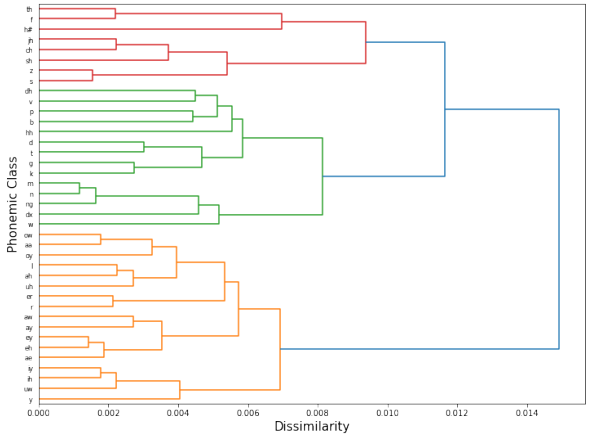

Figure 2.3: Dendrogram of the hierarchy that resulted from the adapted average linkage agglomerative clustering algorithm.

algorithm generalize to clustering? [...One of the conditions of experiment 2 uses similar mechanisms (same distance function) as the classification algorithm which effectively isolates one iteration of the clustering algorithm, reducing the computational cost needed to tune the hyperparameters.]

Moreover, Experiment 1 illustrates the effective use of conceptors for phoneme classification. [Comparison to phoneme classification benchmark...] [Comparison to Jaeger's experiments...]

Aperture normalization The methods of conceptor based time series classification used in Experiment 1 were largely inspired by (Jaeger, 2014) with the exception of the aperture adaptation. The first deviation here was to adapt explicitly only the apertures of the positive conceptors C^+..... **[normalizing/optimizing the aperture of the negative conceptors does not help]** It was made to avoid to give conceptors with larger singular values an advantage during classification. To understand this effect, consider the example of a 2×2 conceptor matrix A. **[...]** To avoid such classification errors, the apertures were iteratively adapted until approaching a target singular value sum using the following procedure. Indeed this normalization further improved classification.

It must be remarked that, by that process, the final apertures will deviate from the one originally chosen aperture via the ∇-criterion. Figure 2.2 depicts this normalization process for the classification experiment.

Dataset Without normalization in time, performance decreases... how to handle different utterance speeds? Add leakiness to ESN to integrate time scales?

Non-stationary signals [z condition worse due to overfitting, or perhaps exponential reason (AI2)Z-condition] Thus, the size of z and the associated computational costs are larger than for the former method, and it tends to lead to better *training* classification accuracy, but much worse test accuracy indicating overfitting.

Exp 2: Advantages of clustering method Conceptors do not [...]. As constant-sized matrices, they do not scale with the number of time steps or modeled states. This is a leap for analyzing time series since previous algorithms typically scale poorly with longer samples; for example, computing the distance between time series in dynamic time warping (DTW) scales quadratically with sample length dynamic. Notwithstanding, the quality of conceptors depends on the number of states used for their computation [...]. [Comparing responses of different lengths would require workarounds like Dynamic Time Warping (DTW).]

Third, ESNs deal with time-extended inputs causing several difficulties for preexisting clustering algorithms. For example, the euclidian distance can only be used to compare samples of different durations (for clustering algorithms that require distance functions) and have time-.

[Time complexity of clustering algorithms: especially Experiment 2]

[Mention esp might not be satisfied]

[Effects of spectral radius on ESP]

It should be stressed that, in the current work, *ESN activities* are classified and clustered using conceptors; Not speech signals, but an ESN's responses to these signals are used as input to classification and clustering algorithms.

[BPTT] To evaluate how well a classifier can distinguish between a set of classes, one may resort to performance metrics or investigate the degree of certainty in its predictions. However, such an evaluation relies only on the outputs and disregards

the internal states that led to the predictions. For complex models or whole pipelines, it may be of interest to find groups or clusters present in the RNN's activations or within a subset thereof, e.g., corresponding to only one part of a pipeline. This study ought to be an exploratory step toward answering the following questions: How clustered are the activities within a NN? When, during training, do class differences in network activities arise?

Nonetheless, a phonetic interpretation about the speech signals will be made which, however, requires an assumption. Commonly, the echo state property is used to ensure a functional relationship between driver signal (phoneme recording) and RNN response Yildiz et al. (2012). However, for the echo state to be reached requires left-infinite or at least considerably long input sequences, for the starting state to be washed out. In our case, inputs will be downsampled to a length of 10 rendering the claim of the ESP impossible based on current theory. The analyzed states will most likely still correspond to the network's initial transient period. Thus, functionality between input and response cannot be guaranteed but only assumed. Hence, we shall nonetheless use the results of clustering and classification of ESN activities from the transient period for an interpretation about the input signals.

Adv of conceptor-based clustering to classical: Moreover, they compress one or more network states into a mathematical object (a matrix). These Another advantage, This hurdle can m and one advantage applying conceptors to time series is, as will be further argued, that they capture their information in a static mathematical compressing the temporal dimension and

To summarize, the three main motivations of the following clustering methods are to provide additional.

1. Explainability: Clustering RNN dynamics could help explain what groups of activity are present in the network. For example, both algorithms could be helpful tools for interpreting BPTT-trained RNNs. For example, Experiment 2 could show how classes become more distinct or numerous within the RNN (form activity clusters) while training.

2. Data analysis: Clustering time series with traditional clustering algorithms can be challenging. For instance, due to varying input lengths, methods for embedding time-series data are rather limited (e.g., latent-space encodings). This paper shows that the clusters present in RNN dynamics can be tied back to clusters within the inputs used to produce these dynamics.

3. Conceptor Classification: To the end of successful clustering, a phoneme classification experiment (Experiment 1) will be performed. In this experiment, the current methods for conceptor-based classification are applied and extended.

Conceptors as state Embeddings The temporal and spatial compression realized by conceptors, allows them to be used as embeddings of ESN responses. Unifying responses in conceptor-space, independently of their length and maintaining only their most important information, allows state sequences to be related and compared. [Examples of the use of conceptors as embeddings]

... We attempt to solve this mismatch by mapping the network states to be clustered to conceptors since conceptors offer several tools for comparing and analyzing network states. By moving to conceptor space, the following adaptations were made.

1. plus becomes or: Firstly, the operations of Euclidian vector space were changed.

2. division become aperture adaptation

3. Distance becomes the one proposed by Jaeger (2014) OR...

Problems. Distance metric is rather unclear, lack of semantics. Average linkage: Linkage algorithms: Pros and Cons slide of video

Categories may be due to coarticulation / correlationss

4 Conclusions

Conceptors are well-suited to apply clustering algorithms to time-series and ESN dynamics.

References

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, Joydeep Ghosh, and John Lafferty. Clustering with bregman divergences. *Journal of machine learning research*, 6(10), 2005.

Paul Bricman, Dr Herbert Jaeger, and Dr Jacolien van Rij-Tange. Nested state clouds: Distilling knowledge graphs from contextual embeddings. Bachelor's thesis, University of Groningen, 8 2022.

Satchit Chatterji. Cut the carp! using context to disambiguate similar signals using conceptors. Bachelor's thesis, University of Groningen, 8 2022.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Herbert Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.

K-F Lee and H-W Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

Lerato Lerato and Thomas Niesler. Investigating parameters for unsupervised clustering of speech segments using timit. In *Twenty-Third Annual Symposium of the Pattern Recognition Association of South Africa*, page 83, 2012.

Carla Lopes and Fernando Perdigao. Phoneme recognition on the timit database. In Ivo Ipsic, editor, *Speech Technologies*, chapter 14. IntechOpen, Rijeka, 2011. doi: 10.5772/17600. URL `https://doi.org/10.5772/17600`.

Mantas Lukoševičius. A practical guide to applying echo state networks. *Neural Networks: Tricks of the Trade: Second Edition*, pages 659–686, 2012.

Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015. doi: 10.25080/Majora-7b98e3ed-003.

Till Mossakowski, Razvan Diaconescu, and Martin Glauer. Towards fuzzy neural conceptors. *IfCoLog Journal of Logics and their Applications*, 6(4):725–744, 2019. URL `https://collegepublications.co.uk/ifcolog/?00033`.

Vaclav Pfeifer and Miroslav Balik. Comparison of current frame-based phoneme classifiers. *Advances in Electrical and Electronic Engineering*, 9, 12 2011. doi: 10.15598/aeee.v9i5.545.

Jamie Vlegels. Multivariate time series classification using conceptors: Exploring methods using astronomical object data. Bachelor's thesis, University of Groningen, 2022.

Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

A Appendix

A.1 Computing Projector Matrix

The following algorithm for computing projector matrix $P_{R(C)\cap R(B)}$ is taken from Jaeger (2014). C and B are conceptors of size N. We begin by computing the basis matrix $B_{R(C)\cap R(B)}$ as follows:

1. Compute the SVDs $C = U\text{diag}(s_1, ..., s_l, s_{l+1}, ..., s_N)U'$ and $B = V\text{diag}(t_1, ..., t_m, t_{m+1}, ..., t_N)V'$, where l and m are the indices of the last singular values if C and B not below the arbitrary threshold $\epsilon_{impr} = 10^{-10} \approx 0$; This approximation is to account for computational imprecision [**Is this the right word?**], but in theory, $s_{l+1}, ..., s_N$ and $t_{m+1}, ..., t_N$ should all equal to 0.

2. Let $U_{>l}$ be the submatrix of U made from the last $N - l$ columns of U, and similarly, let $V_{>m}$ consist of the last $N - m$ columns of V.

3. Compute the SVD $U_{>l}(U_{>l})' + V_{>m}(V_{>m})' = W\Sigma W'$, where Σ =diag$(\sigma_1, ..., \sigma_k, \sigma_{k+1}, ..., \sigma_N)$. Again, k is the index of the last singular value of C not below ϵ_{impr}.

4. $B_{R(C)\cap R(B)} = W_{>k}$, the submatrix of W consisting of the last $N - k$ columns of W.

Finally, $P_{R(C)\cap R(B)} = B_{R(C)\cap R(B)}B'_{R(C)\cap R(B)}$.

A.2 Dataset

Table A.1 lists the phone classes and their frequencies within the processed TIMIT dataset.

A.3 Results

Figures A.1 and A.2 relate the number of clusters to cluster quality as measured by the NMI and the SC scores, respectively.

A.4 Extension of Experiment 1: Hyperparameter Optimization

Hyperparameters $k_b, k_{W^{in}}, r$ and ρ were also tuned automatically using bayesian optimization procedure of the Bayesian Optimization python package. The optimization objective was to maximize

	Phone	Folded	#Training	#Test
	iy		6953	2710
	ih	ix	13693	4654
	eh		3853	1440
	ae		3997	1407
	ah	ax-h ax	6291	2343
	uw	ux	2463	750
	uh		535	221
	aa	ao	6004	2289
	ey		2282	806
	ay		2390	852
	oy		684	263
	aw		729	216
	ow		2136	777
	l	el	6752	2699
	r		6539	2525
	y		1715	634
	w		3140	1239
	er	axr	5453	2183
	m	em	4027	1573
	n	en nx	8762	3112
	ng	eng	1368	419
	ch		822	259
	jh		1209	372
	dh		2826	1053
	b		2181	886
	d		3548	1245
	dx		2709	940
	g		2017	755
	p		2588	957
	t		4364	1535
	k		4874	1614
	z		3773	1273
	v		1994	710
	f		2216	912
	th		751	267
	s		7475	2639
	sh	zh	2389	870
	hh	hv	2111	725
d	h# (silence)	dcl tcl kcl bcl pcl pau epi q gcl	39467	14021
Σ	39	22	177080	64145

Table A.1: Phone labels and their frequencies. The first column contains the phone labels used as classes during classification and for the evaluation of the clustering algorithms. The second column lists the phones that were originally present in TIMIT but were folded into a class with the phone on the left during preprocessing. The third and fourth columns depict the number of speech samples of that class. The last row contains, respectively, the number of phones after folding, the number of phones that disappeared through folding, the total number of samples in the training set and the total number of samples in the test set.

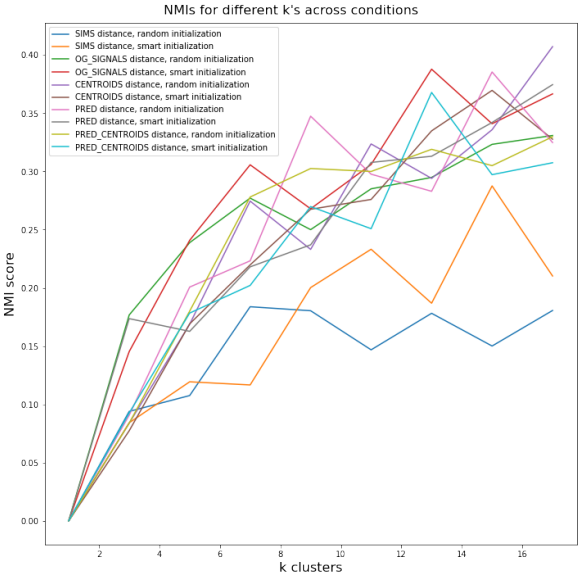

Figure A.1: NMI scores by values of k across conditions

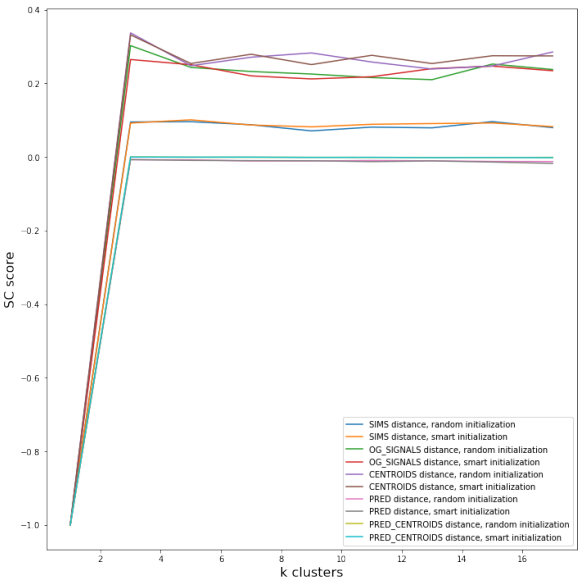

Figure A.2: SC scores by values of k across conditions

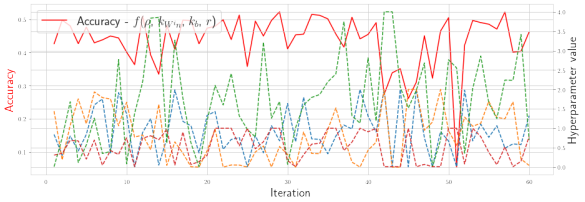

Figure A.3: Hyperparameter tuning with the bayesian optimizer and accuracy [Still no progress seems to occur. This experiment will need to be repeated.]

the testing accuracy of phoneme classification in Experiment 1. Concretely, after 10 initial exploration steps, 40 optimization steps were taken. At each optimization step, a set of hyperparameters is sampled from a promising region of the hyperparameter space trying to maximize an estimated surrogate $\bar{f}$ for the unknown objective function f [$f(\rho, k_{W^{in}}, k_b, r)$:=accuracy]. After training and testing the phoneme classifier with these hyperparameters on the training set (with a train-test split), the surrogate estimate is improved (for a detailed review of bayesian optimization, see Frazier). The hyperparameter space was restricted to:

- bias scaling parameter $b \in (0, 2)$
- input weight scaling parameter $k_{W^{in}} \in (0.01, 0.99)$
- spectral radius $r \in (0.01, 4)$
- internal weight density $\rho \in (0.01, 1)\%$

The progress of the bayesian optimizer is depicted in Figure A.3. Bayesian optimization was preferred over the more straightforward grid search, since, under consideration of the high computational complexity of training the classifier (about 30 minutes on my computer), the reduced number of training steps outweighed the overhead added by the bayesian optimizer.

A.5 Extension of Experiment 1: Inclusion of input states

Experiment 1 was repeated in slight adaptation of its methods to the stationary type of data. Jaeger (2014) demonstrated that conceptors may be used

to classify signals produced by stationary and non-stationary processes. Stationary processes produce the same kind of signal (with the same probability distribution) over time; for example, white noise or sin waves are the results of stationary processes. Meanwhile, non-stationary processes change their properties over time leading to signals like speech whose probability distributions change over time. Meanwhile in the current classification method, the order within a sequence of states does not affect the resulting conceptor since it is lost when computing the correlation matrix. This works fine on signals from stationary sources where temporal order is of no relevance. However, for non-stationary sources, states can have different probabilities of occurring depending on their position in the sequence, information not accounted for in the current classification method. Jaeger (2014) approached this limitation by unrolling the ESN response $x(n)_n = 1, ..., L$ into a vector z reserving a dimension for each step in time. Moreover, the input signal s is appended to z for additional information. We have $z = [x(0); s(0); x(1); s(1); ...; x(L); s(L)]$. For z, the same classification procedure applies. New hyper-parameters were picked by hand; the ESN size was reduced to $N' = 40$ neurons due to the larger computational complexity of this method, but the same density of $r' = 10\%$ was used. Scaling factors were changed to $k'_{W^{in}} = 1.5$ and $k'_b = 0.2$. A spectral radius of $\rho' = 1.5$ was used. A training accuracy of 63.64% and test accuracy of 49.13 were reached.

B Appendix

B.1 Mathematical notations

Notation	Meaning
A' or x'	Transposes of matrix A or vector x
I	$N \times N$ identity matrix, N to be inferred from context
$[x\|y]$	Matrix resulting from the column-wise concatenation of vectors x and y
$[x; y]$	Matrix resulting from the row-wise concatenation of vectors x and y
$\|S\|$	Cardinality of set S
$\|\|A\|\|$	Frobenius norm of matrix A
$\text{diag}(A)$	Vector containing the main diagonal of matrix A
$\|x\|$	Magnitude of vector x
$A^\dagger$	Pseudo-inverse of square matrix A
$R(A)$	Range of matrix A
$tr(A)$	Trace of square matrix A
P_S	$N \times N$ Projector matrix on the linear subspace S of $\mathbb{R}^N$, N to be inferred from context
$S \cap Z$	Intersection of linear spaces S and Z

Table B.1: List of some of the mathematical notations used throughout the paper.