



# CLUSTERING ESN DYNAMICS USING CONCEPTORS

Bachelor's Project Thesis

Joris Peters, s4001109, j.peters.13@student.rug.nl,

Supervisors: G.Pourcel & Prof Dr H. Jaeger

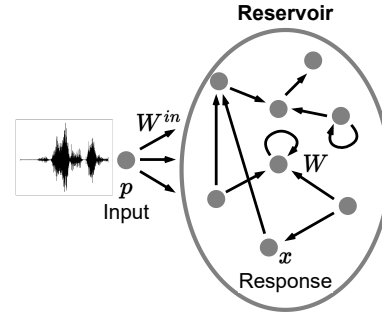
**Abstract:** For data analysis, explainability, and more, it can be of interest to identify groups and hierarchies within the activity dynamics of recurrent neural networks (RNNs). Using conceptors to represent RNN dynamics, this study aims to cluster the dynamics of Echo State Networks (ESNs), a variant 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 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 Background

#### 1.1.1 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 inputs in a higher-dimensional space. ESNs are a type of recurrent neural network (RNN) composed of a reservoir of large size (many internal neurons), low density (low connections-to-neurons ratio), and random and fixed internal and input weights (random initialization and no training) Yildiz, Jaeger, and Kiebel (2012). Figure 1.1 depicts a typical ESN as it is driven by an input sequence  $u$  (that may well be a recording of Jim Button's voice) and elicits a higher-dimensional response sequence  $x$ , the series of internal states in the ESN's reservoir (the echos in the valley). An output layer is commonly added to the ESN for (re)productory purposes but omitted here, since only the internal states of the network will be of interest as will become apparent shortly.



**Figure 1.1:** Sketch of an ESN with an input driver signal. State vector  $x$  is a high-dimensional response to the input signal  $p$ .

Let us declare some notations. Let  $N$  be the number of neurons of an example ESN. As mentioned,  $N$  will typically be large relative to the input dimensions. 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 dimensionality  $d$  and 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 the 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 vectors at time  $n$ , and  $\tanh$  is the hyperbolic tangent. It is apparent that any  $x(n)$  depends on the current input time-step and the ESN's previous internal state itself. Moreover, the response  $x$  can, indeed, be seen as a high-dimensional expansion of the equally long input if  $N \gg d$ , and the system's dynamics are non-linear due to the application of  $\tanh$ .

### 1.1.2 Conceptors

Conceptors provide a variety of options for analysing, controlling, and otherwise using ESNs (and other neural networks). The primer on conceptors that is about to follow is based on the detailed report Jaeger (2014).

**Definition** Loosely speaking, a conceptor is a "fingerprint" of the activity of [a] network during [a] period [of time]" Jaeger (2014). It can be computed from any sequence of network states  $x = (x(1), \dots, x(L))$  \*. A positive semi-definite conceptor matrix derived from  $x$  minimizes the following loss function  $\mathcal{L}$ :

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

where  $C$  is the conceptor,  $\|C\|^2$  is the sum of all of  $C$ 's squared entries, and  $\alpha \leq 0$  is the conceptor's aperture (further explained below). The conceptor's capacity to *characterize* the sequence of states is reflected in this loss function; Firstly, when minimizing  $L$ , the term  $\|x(n) - Cx(n)\|^2$  nudges  $C$  toward realizing an identity mapping of  $x(n)$  to  $x(n)$  for all states  $x(n)$ . Secondly, the added regularization term,  $\alpha^{-2}\|C\|^2$ , attaches a cost to the magnitude of matrix entries, requiring the conceptor to concentrate (**better word?**) on the axes of

\*We consider *sequences* of network states for practicality, but these states need not necessarily be chronologically ordered or chronologically continuous, and can be thought of as *sets*.

greatest variance in state space, distilling the most defining features from the states. Thus, the resulting conceptor is characteristic of the sub-space that the state sequence occupies. A procedure to analytically compute a conceptor from the state sequence  $x$  and with an aperture of  $\alpha$  is as follows.

1. Concatenate states  $x(n)$  of  $x$  column-wise in an  $N \times L$  collection matrix  $X = [x(0)|x(1)|\dots|x(L)]$ .
2. Compute correlation matrix  $R = XX'/N \approx \text{corr}(X)$ .
3. Obtain conceptor  $C(R, \alpha) = R(R + \alpha^{-2}I)^{-1}$  with  $N \times N$  identity matrix  $I$ .

**Aperture** The aperture parameter determines the amount of regularization of conceptors. "With large apertures, the resulting conceptor will tend toward the identity matrix  $I$ ; for shrinking apertures it will come out closer to the zero matrix." [Paraphrase this citation and develop this paragraph to match my methods.]

**Similarity function** Conceptors may be used to compute the similarities of state sequences. If  $C^i$  and  $C^j$  be the conceptors derived from the two sequences of states to be compared, their similarity may be defined as:

$$\text{Sim}_{i,j}^\alpha = \frac{|(S^i)^{1/2}(U^i)^T(U^j)(S^j)^{1/2}|^2}{|\text{diag}(S^i)||\text{diag}(S^j)|}, \quad (1.3)$$

where  $US^jU'$  is the SVD of  $C^j$ . It intuitively corresponds to a measure of angular alignment between all pairings of singular vectors of the two conceptors weighted by the corresponding singular values.

**Classification** Another use of conceptors is for time-series classification. The concrete procedure is outlined in Section 2.4, but guiding idea is the following. Reiterating, conceptors describe the linear subspace occupied by the cloud of reservoir states used for its computation. Any conceptor can, subsequently, be used to compute the distance of an unclassified reservoir state to the linear subspace the conceptor describes. This distance computation allows conceptors to be used to classify reservoir states. Furthermore, under the echo state property (ESP), a reservoir's response to an input sequence

is functional. Thus, the label assigned to a reservoir states sequence may also be reasonably assigned to the input that induced the reservoir response. However, this method for time-series classification assumes the ESP: **[Describe ESP]** Jaeger (2014) demonstrated this method for time-series classification to an accuracy of about 99.99% (its methods come close to the methods outlined in Section 2.4).

**RQ** [if classification of timeseries works, how about clustering?]

Three aspects make ESNs attractive for this study. First, their large size renders explainability particularly challenging and working on it relevant, whilst it also poses interesting time-complexity questions that will be addressed in the discussion. Second, they allow focus on the network's activities without the need for concern with their training. Third, ESNs, under certain conditions, quite literally, echo their input; their states become high-dimensional non-linear expansions of whatever input the networks are fed. 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.

This paper will investigate methods for clustering NN activations. When a NN is used for whatever task, the computations leading to the output are obscure. For example, 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, this evaluation relies solely on the outputs. How may one evaluate the classes present within the model before the output? For complex models or whole pipelines, it may be of interest to find groups or clusters present in the NN's activations (or within a subset thereof, e.g., corresponding to only one part of a pipeline). This will be the aim of this paper. It will be an exploratory step toward answering the following questions: How clustered are the activities within a NN? When, during train-

ing, do class differences in network activities arise? To analyze structures in network activities with little or without prior information, it is useful to look toward unsupervised learning. This field provides techniques for identifying patterns within unlabeled data: clustering algorithms. The two types of clustering methods considered and adapted to the current problem are average linkage and K-means.

- Clustering algorithms aim to identify groups within data.

- Clustering time series faces additional challenges [...]. **Average linkage is a type of hierarchical agglomerative clustering.** ... – Hierarchical clustering requires 3 things:

1. points
2. function defining the distance between points
3. function defining the distance between clusters (sets of points)

- **Kmeans is...** – Kmeans clustering requires 3 things:

1. points
2. function to compute centroids from clusters
3. function to decide which cluster to assign a given point to

**Clustering Algorithm** – Clustering algorithms aim to identify groups within data.

- Clustering time series faces additional challenges [...].

- Hierarchical clustering requires 3 things:

1. points
2. function defining the distance between points
3. function defining the distance between clusters (sets of points)

- Kmeans clustering requires 3 things:

1. points
2. function to compute centroids from clusters
3. function to decide which cluster to assign a given point to



for phoneme classification on TIMIT is held by [...] that used [...].

## 2 Methods and Results

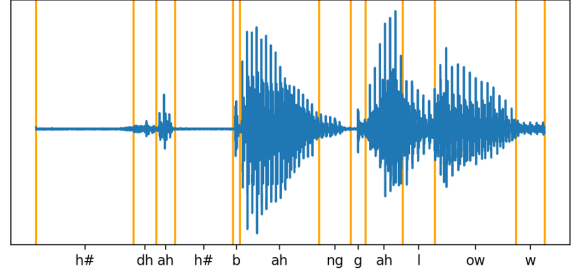
### 2.1 Dataset

The TIMIT Acoustic-Phonetic Continuous Speech Corpus was chosen as the data source, for it contains diverse and phonetically annotated speech signals. Each of 630 American native speakers (from eight dialect regions and of which 30% female) read ten sentences – five phonetically-compact (SX), 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 Preprocessing

For all experiments, the following preprocessing steps were performed. The utterances were segmented according to the phonetic transcriptions into  $c = 241225$  segments ( $c_{train} = 177080$  and  $c_{test} = 64145$ ), each of which 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, Mcvcar, Battenberg, and Nieto (2015) was used with one MFCC feature vector computed every 1 ms from a 2.5 ms long sliding window.

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



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

for differences in utterance 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)  $p^i_{i=1,\dots,c}$  were mapped from the original set of 61 phones to a subset set 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 (e.g., /em/ and /m/) into the same classes. The concrete mappings are given in appendix Table ??, along with the distribution of the resulting phonemes within the dataset. It serves to reach reasonable classification and clustering performances, render the computations feasible, and, for all late-comers, make their results comparable to the existing research body.

Thus, the resulting data consisted of tuples  $D = \{(s^i, p^i)_{i=1,\dots,c}\}$  with MFCC time series  $s^i \in [-0.5, 0.5]^{d \times L}$ , phone labels  $p^i \in P$ , and the set of phones  $P$  after mapping ( $|P| = 39$ ). The ready-made train-test split from TIMIT was used giving  $D_{train}$  and  $D_{test}$ .

### 2.3 From MFCCs to ESN states

The following ESN was used for all three experiments. It consisted of  $N = 100$  neurons with a connection density of  $r = 10\%$ .  $W^{in}$  was randomly sampled from a standard normal distribution and rescaled by a factor of  $k_{W^{in}} = 1.1$ .  $b$  was sampled from  $\mathcal{N}_{N \times 1}(0, 1)$  but rescaled by a factor of  $k_b = 0.6$ .  $W$  was randomly initialized and rescaled to a spectral radius of  $\rho = 2.57$ .

The above were picked by hand.  $N$  was chosen as a reasonably large value under the available computational resources. Any larger size would likely improve performance after adapting the other hyperparameters. **[arguments for other parameters, ref. Lukosevicius]**. Moreover, automated hyperparameters optimization was attempted to maximize the accuracy of phoneme classification but was eventually not used due to large computational costs and slow convergence (Appendix B).

The resulting ESN was driven independently on each input signal  $s^i(t)_{t=1,\dots,L}$  producing the reservoir state sequences  $x^i(t)_{t=1,\dots,L}$ . Concretely, each run started from the same state  $x(0)$  sampled once from a standard normal distribution to not 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. The following states  $x^i(n)_{n>0}$  were computed via update Equation 1.1. Finally, the states  $x^i(t)_{t=1,\dots,L}$  were collected column-wise in  $N \times L$  matrices  $X^i = [x^i(0)|x^i(1)|\dots|x^i(L)]$ .

## 2.4 Experiment 1: Phoneme Classification

The above steps of driving the ESN on an input signal and collecting its internal states (the response) is a straightforward computation and will be summarized in function  $r : \mathbb{R}^{dL} \rightarrow \mathbb{R}^{NL}$  with  $r(u) = X$ . Lastly, the spectral radius  $\rho(W)$  of an ESN needs some introduction as it largely affects the dynamic behavior of the ESN that, without it, would be left to chance as weights are randomly initialized.  $\rho(W)$  corresponds to the largest eigenvalue of the internal weight matrix  $W$ . The larger  $\rho$ , the stronger  $W$  transforms the internal state during the state update, and the more chaotic the ESN will be. An ESN's spectral radius may be adapted by simply rescaling the internal weight matrix as  $W \leftarrow W \times \frac{\rho_{target}}{\rho(W)}$  where  $\rho_{target}$  is the desired spectral radius.

**Classification using Conceptor** Conceptors may be used to classify discrete-time signals. The objective is to assign the correct label  $y^{new} \in \text{Classes}$  to an unforeseen discrete-time sample  $u^{new}$

its phonemic class  $p^{new}$ .

### 2.4.1 Training

Given is a training set  $D_{train} = (u^i, y^i)_{i=1,\dots,c}$  consisting of  $c$  tuples of time series  $u^i$  and labels  $y^i$ . Training the classifier amounts to computing a "positive" conceptor  $C_y^+$  and a "negative" conceptor  $C_y^-$  for each class  $y \in \text{Classes}$  based on the  $\eta_y$  training samples  $U^y = s_{p,i=1,\dots,\eta_y}$  of that class.

$C_y^+$  is computed as follows. The ESN is driven independently on each input signal  $u^i(t)_{t=1,\dots,L^i, i=1,\dots,\eta_y} \in U^y$  producing the reservoir state sequences  $x^i(t)_{t=1,\dots,L^i}$ , where  $L^i$  is the length of  $u^i$ . All states of all these state sequences were concatenated column-wise into a collection matrix  $X_y = [x^1(1)|x^1(2)|\dots|x^1(L^1)|x^2(1)|x^2(2)|\dots|x^2(L^2)|\dots|x^{\eta_y}(1)|x^{\eta_y}(2)|\dots|x^{\eta_y}(L^{\eta_y})]$  from which  $C_y^+$  is computed with an initial aperture of  $\alpha = 1$  by the above procedure. These steps are repeated for each label resulting in the set of positive conceptors  $C^+$ .

**Aperture adaptation [...] [normalizing/optimizing the aperture of the negative conceptors does not help]**

After computing the conceptors in  $C^+$  with  $\alpha = 1$ , the apertures are optimized. Various optimization approaches exist, but the method of choice for the current methods is the  $\nabla$ -rule **[Missing description...]**.

**Negative conceptors** From the adapted positive conceptors  $C^+$ , the set of negative conceptors  $C^- = C_{y,y \in \text{Classes}}^-$  is computed with  $C_y^- = \neg \bigvee \{C_y^+ | y \neq z, z \in \text{Classes}\}$ .

**Testing** A new sample  $u^{new}$  is classified using the mean combined evidence  $\bar{E}(x, y)$ . First, the combined evidence  $E(x, y)$  will be introduced; it is defined as the sum of a positive evidence  $E^+(x, y)$  computed using the positive conceptor  $C_y^+$  and a negative evidence  $E^-(x, y)$  computed using the negative conceptor  $C_y^-$ :

$$\begin{aligned} E(x, y) &= E^+(x, y) + E^-(x, y), \\ E^+(x, y) &= x' C_y^+ x \\ E^-(x, y) &= x' C_y^- x \end{aligned} \tag{2.1}$$

where  $x'$  is the transpose of  $x$ . Thus, the combined evidence  $E(x, y)$  is a measure of similarity between a state vector  $x$  and a pair of positive and negative conceptors that correspond to label  $y$ . It is large if  $x$  is close to the point cloud used to compute conceptor  $C_y^+$  and far from the point cloud used to compute the other conceptors (see definition of  $C_y^-$ ).

To classify a single state  $x$ , the positive evidence is to be maximized over all classes:

$$Class(x) = \arg \max_{y \in Classes} E(x, y) \quad (2.2)$$

To classify a point cloud  $X$  (a state collection matrix), the mean of the evidences for all state points (columns of the collection matrix) is taken:

$$Class(X) = \arg \max_{y \in Classes} \frac{1}{L} \sum_{0 < i < L} E(X[:, i], y), \quad (2.3)$$

where  $L$  is the width of  $X$ . Finally, if  $X = r(u^{new})$ ,  $Class(X)$  will return the classifier's estimate for the class of  $u^{new}$ .

Jaeger (2014) demonstrated that this method applies to the classification of signals whether were produced by stationary or non-stationary processes. Stationary processes are ones that produce the same kind of signal whatever the time (with the same probability distribution); 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. The above method works fine on signals from stationary sources because the temporal information that is lost when concatenating the ESN responses  $x(n)$  into a collection matrix and computing the correlation from it, is of no relevance. But for non-stationary sources, observations have different probabilities of occurring depending on the time which needs to be accounted for during classification. Jaeger (2014) solved this by unrolling the ESN response  $x(n)_n = 1, \dots, L$  into a vector  $z$  reserving having each time step at a constant point in time. Additionally, the input is introduced for additional information. We have  $z = [x(0); u(0); x(1); u(1); \dots; x(L); U(L)]$ . For  $z$ , the same classification procedure applies. While the large size of  $z$  and the associated computational costs are larger than for the former method, it tends

to lead to better classification, especially of signals from non-stationary sources.

The procedure described above was used with some deviation to train a conceptor-based classifier. Here,  $D_{train}$  was used as the training set. The steps that deviated from the above procedure merely concerned the aperture adaptations. After adapting the apertures to the optimal ones found via the  $\nabla$ -rule and before computing  $N^-$ , the apertures of each conceptor were adapted to normalize the sums of all conceptors' singular values:

1. The target sum of singular values  $s_{target}$  was computed as the average of the sums singular values of each conceptor in  $C^+$ :

$$s_{target} = \frac{1}{|P|} \sum_{p \in P} trace(S^p), \quad (2.4)$$

with  $S^p$  being the Note that the trace of

2. This target was approached up to an error of  $\epsilon = 0.01$  reaching by iteratively adapting the aperture of each conceptor by a fraction of the current sum and the target sum:

---

**Algorithm 2.1** Set sum of singular values of conceptor  $C$

---

```

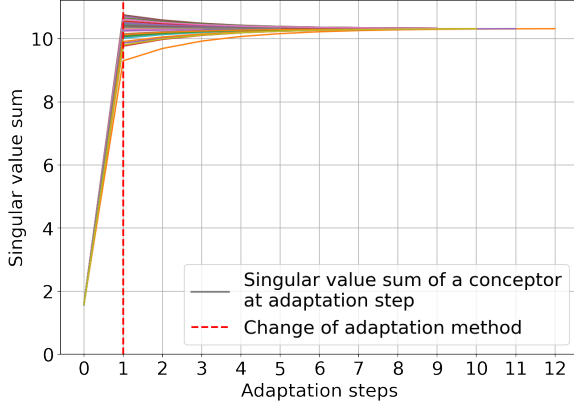
while TRUE do
   $U, S, U' \leftarrow svd(C)$ 
  if  $|s_{target} - trace(S)| < \epsilon$  then
    break
  end if
   $\gamma \leftarrow s_{target}/trace(S)$ 
   $C \leftarrow \psi(C, \gamma)$ 
end while

```

---

This was important since conceptors with larger singular values are given an advantage during classification. To understand the reason, consider the example of a  $2 \times 2$  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 lambda rule. Figure 2.2 depicts this normalization process for the classification experiment.



**Figure 2.2:** The changes in the sum of singular values of the conceptors in function of the adaptation steps. The first adaptation step is done using the  $\nabla$ -rule. The following are done via normalization to the target mean sum (red dotted line).

Finally, the testing procedure outlined in the introduction was applied on the training and test sets.

**Results** This resulted in an accuracy of 56.16% on the training set and 56.04% on the test set.

**Extension: Inclusion of input states** In an attempt of achieving the highest possible accuracy for bench-marking the current method, the experiment was repeated slightly differently. [Concatenation of states and input vectors]. Due to the added computational complexity, the hyperparameters were picked by hand: Here, the ESN size was reduced to  $N = 40$  neurons with a density of  $r = 10\%$ . Scaling factors were changed to  $k_{Win} = 1.5$  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 achieved [for discussion: overfit].

## 2.5 Activity Clustering

For k-means and average linkage to be applied to network activities, their distance functions need to be adapted. Classically, these clustering algorithms are used on data in euclidian space, and their distance functions are defined on that space (euclidian distance, manhattan distance, etc.). Hence, to

apply these algorithms to network activities, the distance functions may need be adapted.

Conceptors may be a means to adequate distance functions between states.

The activities within the ESN are subjected to non-linear transformations like  $\tanh$  at each new time step. While this, for instance, allows for more complex decision boundaries, it means that euclidian distance functions may not be well-suited to capturing the distances between network states. However, after mapping the two sets of states one wishes to compare to conceptors, a potentially more adequate distance function between the states was proposed by Jaeger (2014).

### 2.5.1 Below-phoneme

The below-phoneme clustering was performed in five different conditions [Will be renamed]:

- (a) SIM: Conceptor centroids and conceptor similarity function
- (b) PRED: Conceptor centroids and predictions distance function
- (c) CENTROIDS: Mean reservoir state centroids and euclidian distance function
- (d) OG.SIGNALS: Mean signals centroids and euclidian distance function
- (e) PRED.CENTROIDS: A combination of b. and c.

[I would really like to reduce this number of conditions!]

[Generalized kmeans pseudocode]

#### Conditions

1. points: samples  $D = \{(s^i, p^i)_{i=1, \dots, c}\}$
2. function to compute centroids from clusters (4 conditions a,b,c,d):

$$\begin{aligned}
 (a) d(C^i, C^j) &= 1 - \text{sim}(C^i, C^j) \\
 (b) & \\
 (c) & \\
 (d) & \\
 (e) &
 \end{aligned} \tag{2.5}$$



- 3.
4. function to decide which cluster to assign a given point to (4 conditions a,b,c,d):

$$\begin{aligned}
 (a) & d(Cl^i, Cl^j) \\
 &= \frac{1}{|Cl^i||Cl^j|} \sum_{s^x \in Cl^i, s^y \in Cl^j} d(s^x, s^y) \\
 (b) & \\
 (c) & \\
 (d) & \\
 (e) &
 \end{aligned} \tag{2.6}$$

**Cluster Evaluation** [Explanation of Normalized Mutual Information (NMI) and Silhouette Coefficient (SC)]

**Results** The NMI and SC for different values of  $k$  are shown in Figures A.1 and A.2, respectively.

### 2.5.2 Above-phoneme

The second clustering algorithm aims to build a hierarchy of conceptors through agglomerative clustering. It clusters ESN activities above the level of phonemes by grouping those phonemes whose conceptors are most similar. Specifically, average linkage was used since it is a well-known and relatively robust agglomerative clustering algorithm. In terms of the requirements of clustering, we have:

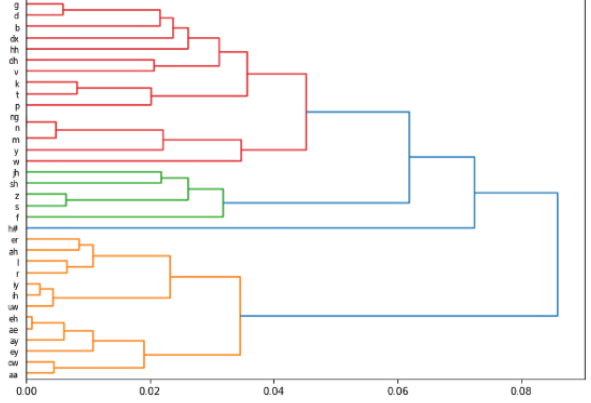
1. points: conceptors  $\{C^p\}_{p \in P}$
2. function defining the distance between points:

$$d(C^i, C^j) = 1 - \text{sim}(C^i, C^j) \tag{2.7}$$

3. linkage function defining the distance between clusters:

$$d_{link}(Cl^i, Cl^j) = \frac{1}{|Cl^i||Cl^j|} \sum_{C^x \in Cl^i, C^y \in Cl^j} d(C^x, C^y) \tag{2.8}$$

From these definitions, the agglomerative clustering was done as follows. The initial clusters are the conceptors. From this set of 39 clusters, always the two closest clusters by the linkage function  $d_{link}$  are grouped into a new cluster. This is repeated until



**Figure 2.3: Dendrogram of the hierarchy that resulted from the average linkage agglomerative clustering algorithm. [still the old picture... to be updated]**

reaching a state of only one cluster. The resulting hierarchy is shown in Figure 2.3.

When used to identify "above-phoneme" clusters: clusters of phoneme classes. Second, phonemic groups are themselves classes that can be used as ground truths to evaluate clustering algorithms when used to identify "below-phoneme" clusters: clusters of individual phoneme utterances. Specifically, phonemic transcriptions – the labels given to utterances commonly provided between brackets, such as [a], and standardized in the International Phonetic Alphabet – will entail the clusters by which the utterances would ideally be grouped.

## 3 Discussion

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

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

## Non-stationary signals

### 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 [...].

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-.

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.

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, the second experiment 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.

## 4 Conclusions

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

## References

- Satchit Chatterji. Cut the carp! using context to disambiguate similar signals using conceptors. *Bachelor's Thesis, Artificial Intelligence*, 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.
- 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.
- 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.
- Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

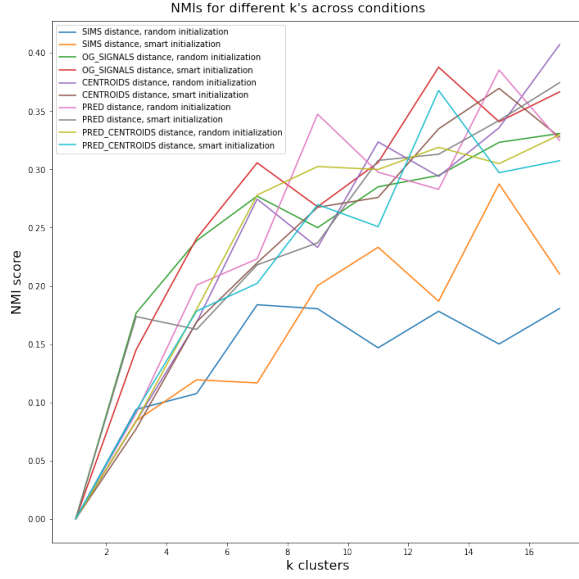
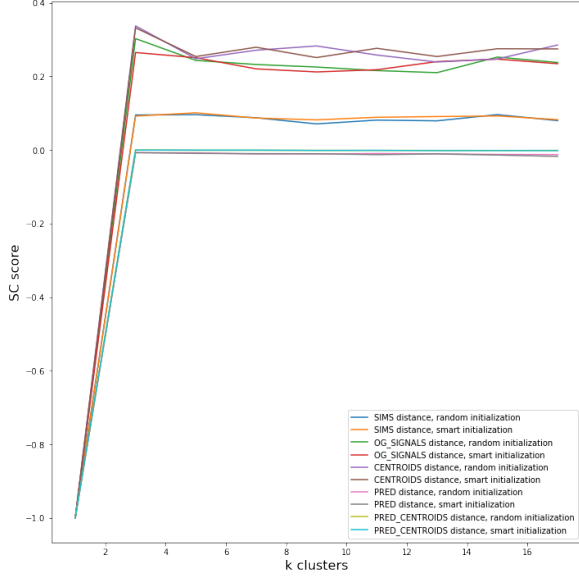


Figure A.1: NMI scores by values of  $k$  across conditions

## A Appendix

| 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<br>(silence) | h# dcl tcl kcl |           |       |
|                | bcl pcl pau    | 39467     | 14021 |
|                | epi q gcl      |           |       |
| $\Sigma$       | 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.

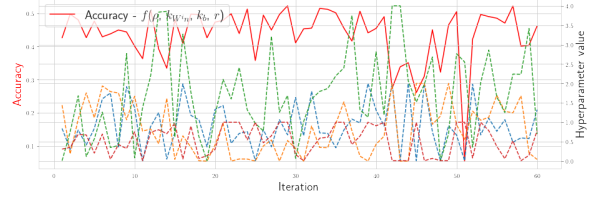


**Figure A.2: SC scores by values of  $k$  across conditions**

## B Appendices

**Hyperparameter Optimization** Then, hyperparameters  $k_b, k_{Win}, r$  and  $\rho$  were tuned to optimize the accuracy of the phoneme classifier. Therefore, I used the bayesian optimization procedure of the Bayesian Optimization python package. 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  $\hat{f}$  for the unknown objective function  $f$  [ $f(\rho, k_{Win}, k_b, r) := \text{accuracy}$ ]. 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  $k_{Win} \in (0.01, 0.99)$
- spectral radius  $r \in (0.01, 4)$
- internal weight density  $\rho \in (0.01, 1)\%$



**Figure B.1: Hyperparameter tuning with the bayesian optimizer and accuracy [Still no progress seems to occur. This experiment will need to be repeated.]**

The progress of the bayesian optimizer is depicted in Figure B.1. 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.