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 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 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 (like Jim Button's voice) and elicits a higher-dimensional response sequence x, the sequence of internal states in 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.

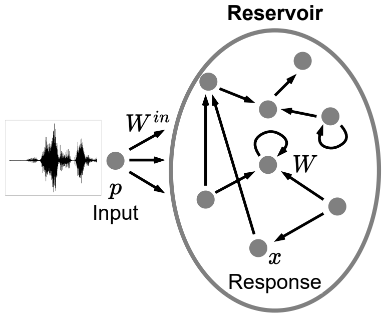

Figure 1.1: Sketch of an ESN driven with an input signal. State vector x is a high-dimensional response to the input signal p.

Let N be the number of 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 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 column vectors at time n and *tanh* is the hyperbolic tangent. 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*. [**Previous 2 sentences necessary?**] For later convenience, we define function $r : S_u \longrightarrow S_X$ from the set of input sequences S_u to the set of response collection matrices $S_X = [x(1)|...|x(L)] \subset \mathbb{R}^{N \times L}$ that are the column-wise concatenations of states $x(n)_{1,...,L}$ that result from driving the ESN (to be infered from the context where r is used) with x. [**Should I use this function e.g. in section 1.2 for conceptor computation or does it overly complicate understanding for the reader? If so, is calling r a function accurate?**]

[Improve transition]

1.2 Conceptors

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

Definition 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 $x = (x(1), ..., x(L))$ [*], the conceptor matrix C computed from x minimizes the following loss function $\mathcal{L}$:

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

where $||C||^2$ is the sum of all of C's squared entries (squared Frobenius norm), and $\alpha \geq 0$ is the conceptor's aperture (further explained below). C may be analytically computed with the following procedure given x and α:

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

[*]We consider *sequences* of network states for practicality, but they can be thought of as *sets* because their states need not necessarily be ordered. Neither do the states need to stem from the same ESN run.

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 parameter 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, only realizing a "good" identity-mapping 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 a hyperellipsoid of a similar shape as the point cloud within 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 the conceptor, respectively. However, the shape of the hyperellipsoid slightly deviates from that of the point cloud as the regularization *squashes* it along its short 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 is a spatial compression as it systematically reduces the variance of the point cloud along its unimportant 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.

[†]Like a Principle Component Analysis (PCA) of the states that provides more accurate scores on the main, high-scored, axes.

When the state sequence is a time-series, this mapping removes the temporal dimension, whence it may be seen as a temporal 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^i and C^j be the conceptors derived from the two sequences of states to be compared, their similarity may be defined as:

$$Sim_{i,j} = \frac{|(S^i)^{1/2}(U^i)^T(U^j)(S^j)^{1/2}|^2}{|diag(S^i)||diag(S^j)|}, \tag{1.3}$$

where US^jU' is the SVD of C^j. It corresponds to a measure of 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 same name C later on again for something else?**], its aperture can still be adapted by any factor of $\gamma > 0$ [**Correct domain for γ?**] using the aperture-adaptation function φ that returns the aperture-adapted conceptor C_{new}:

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

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 ($\neg$)**
$$\neg C := I - C, \tag{1.5}$$
where I is the identity matrix of the same size as the conceptors. It returns a conceptor that describes [**Correct word?**] the linear subspace complementary to that of C. [**How is this true, given my understanding of conceptor negation which returns the volume/area on the outside of C (drawing)?**]

2. **Conjunction ($\wedge$)**
$$C \wedge B := (P_{R(C)\cap R(B)}(C^\dagger + B^\dagger - I)P_{R(C)\cap R(B)})^\dagger, \tag{1.6}$$
where $A^\dagger$ is the pseudo-inverse and $R(A)$ the range of a matrix A, and $P_{R(C)\cap R(B)}$ is the projector matrix on $R(C) \cap R(B)$ (See Appendix A.1 for its computation). It returns a conceptor that describes the intersection of the linear subspaces of C and B.

3. **Disjunction ($\vee$)**
$$C \vee B := \neg(\neg C \wedge \neg B), \tag{1.7}$$
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.

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]

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. The distance of an unclassified reservoir state to the linear subspace that the conceptor describes can be computed. This distance computation can be used to classify the reservoir states to the most fitting conceptor. Furthermore, if the network's responses input sequences are unique, i.e., if there is a functional relationship between inputs and responses, a condition termed the echo state property (ESP), then, the label assigned to a sequence of reservoir states may reasonably be given to the input sequence that induced that response.

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 hexperiment?**]

[Describe the advantages of this classification method.]

[Elaborate on ESP showing how ESN states relate to the input signals]

Leading in 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. Supervised learning is a type of algorithm that typically models a function from a set of sample points, input-output pairs. Jaeger (2014) shows that conceptors can be successfully used on the supervised task of time-series classification, but unsupervised and reinforcement learning tasks have not yet been tackled using conceptors. Reinforcement learning will not be considered in this paper.

Define unsupervised learning and clustering 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, axes, or relationships. K-means and hierarchical agglomerative clustering (HAC) are instances of unsupervised learning that will be further explained [Defend the choice of K-means and HAC].

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 centroids. During the assignment step, points are assigned to their closest cluster minimizing this function.

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.

[Generalization of k-means to Bregman Divergences according to Banerjee, Merugu, Dhillon, Ghosh, and Lafferty (2005)]

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: 1. Use-cases 2. Methods]

[Limitations of time-series clustering and comparison to advantages of conceptor-based classification] Clustering time series faces additional challenges...

Briefing on clustering esn-dynamics A particular kind of time-series are ESN-state sequences, that may be clustered for explainability purposes. This paper will investigate methods for clustering NN activations. 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 training, 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 HAC and K-means.

[Potentially, cite a paper that clusters ESN-states]

[Describe the Problem I am solving: No clustering methods for reservoir states seem to exist.]

RQ [Can conceptors be used to cluster ESN dynamics?]

[Why ESNs]

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.

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 task used to find the right hyperparameters to be used in Experiments 2. and 3. (more on this below).

1.3 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 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).

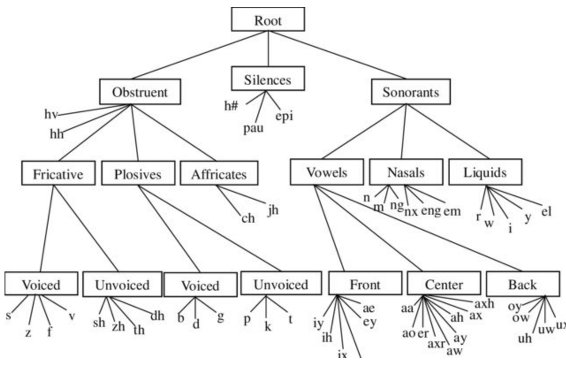

Figure 1.2: A taxonomy of all phones present in TIMIT Pfeifer and Balik (2011)

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

[explain 2 levels: below (without supervision), above (with)] 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.

e

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,

[‡]These notations follow 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.

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 **[...]**.

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), 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, Mcvicar, Battenberg, and Nieto, 2015) was used with one MFCC vector computed every 1 ms from a 2.5 ms long sliding window.

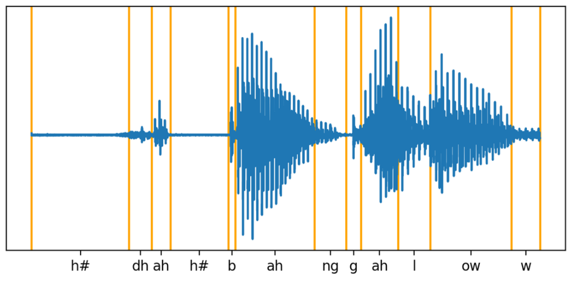

Figure 2.1: Example of the twelve first segments of one of the utterances. [will be identified more precisely]

The resulting time series were normalized in amplitude and time. First, since the amplitudes vary 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 ($\mu_4 = -22.4$)], each channel was normalized to a range of [-0.5,0.5] across all samples. This normalization ought to grant each MFCC a similar effect on the ESN dynamics under the identically distributed input weights to make no assumptions on the channels' relative importance. Second, to account 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 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 $s^i \in [-0.5, 0.5]^{d \times L}$, 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 $r = 10\%$. 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 a internal weight matrix like W is its largest absolute eigenvalue and affects the dynamic behavior of the system that is largely left to chance given the random initialization of ESN weights. The larger ρ, the farther W transforms the internal state during the state update (at least along its first eigenvector), and the more chaotic the ESN tends to be. An ESN's spectral radius may be adapted by rescaling its old internal weight matrix W_{old} to $W_{new} = \frac{\rho_{new}}{\rho(W_{old})} \cdot W_{old}$ where W_{new} will have the desired spectral radius ρ_{new} instead of the old ρ_{old}. [**Elaboration on spectral radius really necessary? I would think so for the discussion on the 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) (reported in section 4.1, p. 161). N was kept as a large enough 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 $s^i(t)_{t=1,...,L}$ producing the reservoir state sequences $x^i(t)_{t=1,...,L}$. 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 I just state this without reporting the experiment?**]. The following states $x^i(n)_{n>0}$ were computed via update Equation 1.1. Finally, the states $x^i(t)_{t=1,...,L}$ were collected column-wise in $N \times L$ matrices $X^i = [x^i(0)|x^i(1)|...|x^i(L)]$.

2.4 Experiment 1: Phoneme Classification

Classification using Conceptor The following experiment aims to classify the phoneme recordings using the above ESN's responses. Alongside, ESN-hyper-parameters will be tuned to prepare for the following clustering experiments. Concretely, the objective is to assign the correct label $y^{new} \in Classes$ to an unforeseen discrete-time sample u^{new} its phonemic class p^{new}. [Specify what *correct label* means]

2.4.1 Training

Given is a training set $D_{train} = (u^i, y^i)_{i=1,...,c}$ 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 $y \in Classes$ based on the η_y training samples $U^y = s_{p,i=1,...,\eta^y}$ of that class.

C_y^+ is computed as follows. The ESN is driven independently on each input signal $u^i(t)_{t=1,...,L^i,i=1,...,\eta^y} \in U^y$ producing the reservoir state sequences $x^i(t)_{t=1,...,L^i}$, where L^i is the length of u^i. All states of all these state sequences were concatenated column-wise into a collection matrix $X_y = [x^1(1)|x^1(2)|...|x^1(L^2)|x^2(1)|x^2(2)|...|x^2(L^2)|...|x^{\eta_y}(1)|x^{\eta_y}(2)|...|x^{\eta_y}($ 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 ∇-rule [**Missing description...**].

After adapting the apertures to the optimal ones found via the ∇-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^+:

$$s_{target} = \frac{1}{|P|} \sum_{p \in P} trace(S^p), \tag{2.1}$$

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 normalization of apertures among the class conceptors was a deviation from (Jaeger, 2014). 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 lambda rule. Figure 2.2 depicts this normalization process for the classification experiment.

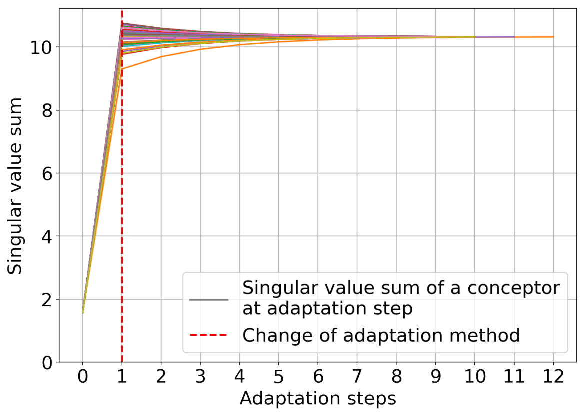

Figure 2.2: The changes in the sum of singular values of the conceptors in function of the adaptation steps. The first adaptation step is done using the ∇-rule. The following are done via normalization to the target mean sum (red dotted line).

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

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:

$$\begin{aligned} E(x, y) &= E^+(x, y) + E^-(x, y), \\ E^+(x, y) &= x' C^+_y x \\ E^-(x, y) &= x' C^-_y x \end{aligned} \tag{2.2}$$

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:

$$Class(x) = \underset{y \in Classes}{\arg\max}\ E(x, y) \tag{2.3}$$

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:

$$Class(X) = \underset{y \in Classes}{\arg\max}\ \frac{1}{L} \sum_{0<i<L} E(X[:, i], y), \tag{2.4}$$

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}.

Finally, the testing procedure outlined above 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 with a slight change to adapt to the type of data. Jaeger (2014) demonstrated that conceptors may be used to classify signals whether they 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. Importantly, 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 $x(n)_n = 1, ..., L$ into a vector z reserving a dimension for each time step. Moreover, the input signal is appended to z for additional information. We have $z = [x(0); u(0); x(1); u(1); ...; x(L); U(L)]$. For z, the same classification procedure applies.

[Concatenation of states and input vectors]. Due to the added computational complexity, the hyper-parameters 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 $k_{W^{in}} = 1.5$ 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 achieved. [Discussion:] 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.

2.5 Activity Clustering

[Parallel clustering in Euclidian space to clustering in conceptor-space] For k-means and average linkage to be applied to network activities, they may need to be adapted. Conceptors may be a means to adequate distance functions between states. 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.). However, 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. 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 was defined by Jaeger (2014).

- plus becomes or
- division become aperture adaptation

[Sketch of clusters in conceptor space]

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) OG_SIGNALS: Mean signals centroids and Euclidian distance function

[Generalized k-means 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):

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

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

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

Cluster Evaluation [Explaination 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:

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

3. linkage function defining the distance between clusters:

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

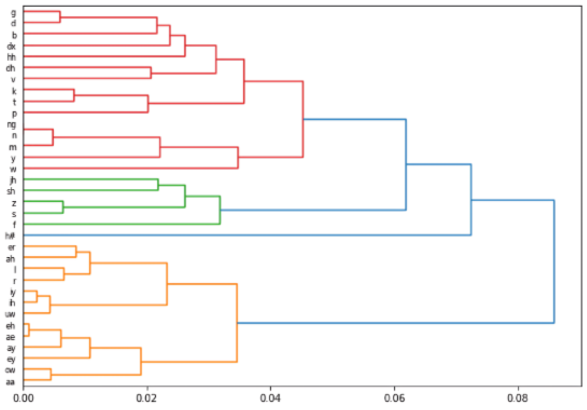

Figure 2.3: Dendrogram of the hierarchy that resulted from the average linkage agglomerative clustering algorithm. [still the old picture... to be updated

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

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 [z condition worse due to overfitting, or perhaps exponential reason (AI2)Z-condition]

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.

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.

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.

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.

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.

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.

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 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 the testing accuracy of phoneme classification in

	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.

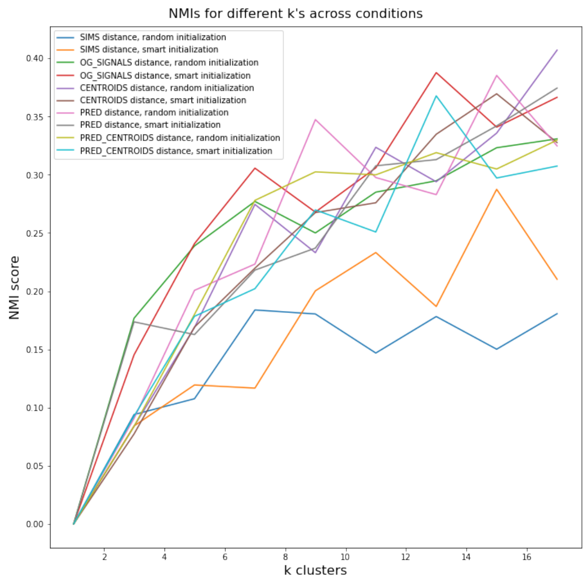

Figure A.1: NMI scores by values of k across conditions

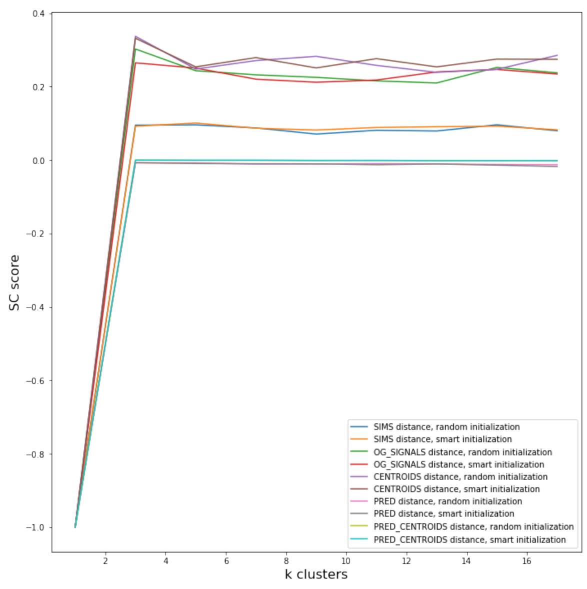

Figure A.2: SC scores by values of k across conditions

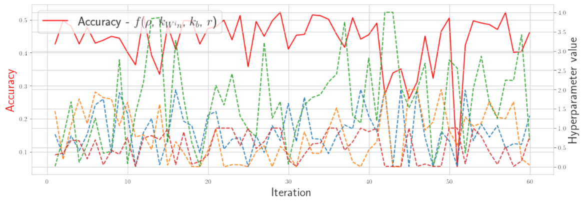

Figure A.3: Hyperparameter tuning with the bayesian optimizer and accuracy [Still no progress seems to occur. This experiment will need to be repeated.]

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.