When does the leading eigenvector agree with cluster means? A fixed-point analysis of Oja's and SoftHebb's streaming rules

Ovidiu-Alexandru Argherie

Supervisors: Stephanie Tan, Yaqi Guo

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

June 21, 2026

Name of the student: Ovidiu-Alexandru Argherie
Final project course: CSE3000 Research Project
Thesis committee: Inald Lagendijk, Stephanie Tan, Yaqi Guo

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

When does the leading eigenvector agree with cluster means? A fixed-point analysis of Oja's and SoftHebb's streaming rules

Ovidiu-Alexandru Argherie
Faculty Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6 2628 XE Delft

Abstract

Backpropagation-free learning rules depict an affinity towards neuromorphic and energy-constrained hardware, yet the final representations that they learn remain not well understood. We dive deep on two local Hebbian rules that appear to compute distinct objectives: (i) Oja's rule computes the first principal component; (ii) SoftHebb extends it to a soft winner-take-all network whose fixed points are normalized component means. In the batch setting, Ding and He (2004) have shown that K-means and PCA are strongly related, that is, the subspace spanned by the cluster centroids coincides with the span of the first $K-1$ principal directions of the data covariance. We analyze if the same correspondence survives sample by sample in a streaming setting, where updates are noisy and the weight vectors are renormalized. As such, we first provide a self-contained fixed-point analysis, which we are going to use it as the common lens for both rules. Second, on controlled two dimensional Gaussian data, we assess some geometric conditions under the rules agree or disagree, yielding an actionable criterion for predicting, on a given dataset, whether the rules converge to the same representation. Third, we show the disagreement is not as the naive picture suggests, that is, an expected divergence does not hold and is replaced with a quantitative account depicted by a ratio of the cluster width to the inter cluster offset.

1 Introduction

Deep learning has achieved remarkable results in the whole spectrum of computer science, all centered around the backpropagation algorithm, which computes gradients through the standard chain rule to update network weights. Even though backpropagation is highly successful, it suffers from several fundamental limitations. First, it is **biologically implausible** [1], since the biological brain does not seem to send error signals backward, along the same connections it used to go forward. Second, it encounters a **locking problem** [1], since every layered operation happens sequentially, that is, layers must wait for each other. Third, backpropagation has a **memory bottleneck** [1], since it stores all intermediate computations during the forward pass, hence it becomes impractical for large models on a setting where resources are limited.

These properties make backpropagation poorly suited for neuromorphic hardware, or, in essence, any setting where computation has to be energy efficient. Therefore, there is a motivation of a growing family of backpropagation-free alternatives [1], such as predictive coding, target propagation, and Hebbian learning.

Hebbian learning is the oldest and the most biologically grounded of the aforementioned methods. Hebb [2] proposed that synaptic connections are strengthened when the pre-synaptic and post-synaptic neurons co-activate, which results in an unstable Hebbian learning method, since the weights can grow indefinitely. Despite this problem, Oja [3] fixed this issue by adding a normalising decay term which makes the weight vector to converge to the top eigenvector of the input covariance matrix. This resulted in computing the first principal component from purely local information [4]. Furthermore, SoftHebb has extended Oja's rule to a soft winner-take-all network [5]. That is, it does not pick one single hard winner, but it assigns to every neuron a graded responsibility for each input and the plasticity rule drives each neuron's weights towards a cluster center (i.e., its unit normalized mean). Both of these make SoftHebb behave analogously to k-means.

As such, PCA looks for directions of large variance. K-means looks for cluster centers. The objectives are not similar, yet Oja's rule and SoftHebb's are assembled from the same Hebbian parts. When do they end up learning the same representation and when do they diverge? In the batch setting, Ding and He [6] have shown that the subspace spanned by the cluster centroids coincides with the span of the first principal directions of the data. Whether that agreement is kept in the streaming[1] rules of Oja and SoftHebb, where updates are noisy and the weight vectors are continuously renormalized, has not been formally characterized, hence this paper aims to analytically and empirically bridge part of this open gap.

To properly assess this aforementioned lacuna, we first outline a precise notion of what each rules converges to. In the streaming setting, the weights are updated one noisy sample at a time, therefore the converged representation cannot be read off the batch objective. That is the direction at which the online dynamics settle, or, to put it differently, the rule's stable fixed point. As such, we begin from a fixed-point analysis recipe: we explain what it is and how a reader could apply it to their own learning rule. The analysis then helps us as follows. On the one hand, it provides the common framework, along with the stability vocabulary, in which each rule's known stable fixed point becomes the predicted convergence target, thus any agreement question reduces to a concrete comparison of two directions. On the other hand, we are able to predict from the geometry of data, by instantiating those fixed points on the data covariance and the centroids, when those aforementioned directions coincide.

We turn this into two testable hypothesis on controlled two dimensional Gaussian mixtures. The former lays out isotropic clusters that are separated along a single axis, where the direction of the largest variance coincides with the cluster structure. In this case, we predict that Oja and SoftHebb converge to the same direction, which is then confirmed experimentally. The latter depicts an anisotropic setup where clusters are elongated and separated orthogonally to the elongation, therefore the variance axis and the cluster axis form a perpendicular angle. The natural expectation would be a fixed right angle divergence, where Oja is locked to the elongation, whilst SoftHebb is to the separation. This expectation turns to be wrong, since SoftHebb acts on unit normalized inputs rather than the raw data, therefore the clusters collapse into overlapping arcs once the spread orthogonal to the separation is large. As such, SoftHebb's split is in agreement with Oja and the disagreement is therefore not a fixed right angle, but ruled by a bounded window, which we can define by a quantitative claim of a single governing ratio $r = \frac{\sigma_1}{d}$. This represents the cluster width to the inter cluster offset, which makes the rules agree for small and large r and diverge only inside the predictable interval.

The rest of the paper is organized as follows. Section 2 constructs the background a reader needs to know to understand the theoretical analysis and the experiments. Section 3 describes the methods used and the experimental setup. Section 4 highlights the theoretical predictions, where we derive the agreements and

[1] Hereinafter we use the terms *online* or *streaming* to indicate that data samples arrive one at a time and the weights update happen after each one, with no access to the full dataset.

disagreements for the synthetic data. Section 5 reports the experimental results and discusses them. Section 6 concludes and outlines possible directions for future work. Section 7 provides a reflection to what has been concluded in terms of the ethical aspects of the work.

2 Background and Preliminaries

This section highlights the background knowledge that lay out a foundation on the theoretical predictions and the experimental outcomes. Section 2.1 depicts the fixed-point analysis recipe. The next two subsections briefly introduce Oja's rule and SoftHebb. The reader may also check the Appendix for additional mathematical background. That is, section A.1 outlines the law of total covariance, section A.2 describes the error function erf and its complement, and section A.3 portrays the modified Bessel function and the Dirac Delta.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, where Ω is the sample space, $\mathcal{F}$ is the σ-algebra of the events and $\mathbb{P}$ is the probability measure with $\mathbb{P} : \mathcal{F} \to [0, 1],\ \mathbb{P}(\Omega) = 1$. Hereinafter consider the input be a random vector $\boldsymbol{x} = (x_1, x_2)^\top : \Omega \to \mathbb{R}^2$, where $x_1, x_2 \colon \Omega \to \mathbb{R}$ are two random variables with real values. The dataset consists of N samples $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ drawn independently and identically distributed (i.i.d.) from a fixed distribution p on $\mathbb{R}^2$. We write $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ for the mean and assume the data is *centered*, that is, $\boldsymbol{\mu} = \mathbf{0} \in \mathbb{R}^2$.

2.1 Fixed-point Analysis

This section aims to present the thought process we had to understand what fixed-point analysis is. In essence, we are going to write the learning rule as a stochastic update, average over the input distribution to obtain an Ordinary Differential Equation (ODE) and then solve for fixed points and stability. This approach follows the ODE method of stochastic approximation [7], applied to plasticity rules as in [3].

When we train a rule online, every step nudges the weight vector $\boldsymbol{w}$ by a small amount $\Delta \boldsymbol{w}$. Each such change depends on two factors: first, the current weight; second, the random data point $\boldsymbol{x}$ that happens to arrive at that step. Since the input is drawn randomly from the data distribution, the direction of $\Delta \boldsymbol{w}$ is, as well, random. Assume that we run the update infinitely many times and the weight draws a path in the weight space. Even if the steps are tiny, the direction is biased by the current weight and the data distribution, therefore the trajectory is a stochastic process, or, to put it differently, a biased random walk we would want to characterize: after enough time, which region of the weight space does the walk concentrate at?

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \cdot \Delta \boldsymbol{w}(\boldsymbol{w}_t, \boldsymbol{x}_t) \tag{1}$$

Before we continue, let us discuss an analogy that will hereby be leveraged. Suppose we walk 100 km along a motorway that is straight, that is, with negligible curvature. We advance in many tiny steps and eventually reach our destination. Looking back over the path, we could see that our net movement was predominantly forward. We call this averaged, consistent forward progress made at each step the *deterministic push*.

Furthermore, looking back over the same path, we notice that the individual steps were not perfectly forward. At some times, we drifted slightly to the left of our heading, and, at other times, slightly to the right. These sideways deviations had no consistent direction, thus, over the full 100 km path, they mainly cancelled and left our arrival at the destination unaffected. We call these small, directionless fluctuations the *random jitter*.

Now, let us take a look at eq. (1) and apply the trick to the plasticity rule of adding and subtracting its averaged value. Formally, that is $\Delta \mathbf{w}(\boldsymbol{w}_t, \boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x} \sim p}[\Delta \boldsymbol{w}(\boldsymbol{w}_t, \boldsymbol{x}_t)] + (\Delta \boldsymbol{w}(\boldsymbol{w}_t, \boldsymbol{x}_t) - \mathbb{E}_{\boldsymbol{x} \sim p}[\Delta \boldsymbol{w}(\boldsymbol{w}_t, \boldsymbol{x}_t)])$. The first term of the addition is the deterministic push, since we are looking at the same direction every time. The latter is the random jitter, or the deviation of $\Delta \boldsymbol{w}$ from its average. The noise has zero mean by construction, but, visually, the reader may imagine the motorway analogy where each left step is going be partially cancelled out eventually by a right step. The mathematical proof may be also found in the Appendix.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta\, \boldsymbol{f}(\boldsymbol{w}_t) + \eta\, \boldsymbol{\xi}(\boldsymbol{w}_t, \boldsymbol{x}_t) \tag{2}$$

Substituting the trick in eq. (1) yields eq. (2), as shown previously, where we defined $\boldsymbol{f}\colon \mathbb{R}^2 \to \mathbb{R}^2$ to take a weight vector and return the expected update vector, and $\boldsymbol{\xi}\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ to take both the weight and input vectors and return the fluctuation. The reader may note that the former depends only on $\boldsymbol{w}$, since the expectation has integrated out $\boldsymbol{x}$ (i.e., it integrates over all values of $\boldsymbol{x}$, weighted by their probabilities).

Let us now look at eq. (2) and substract both sides by $\boldsymbol{w}_t$ and then divide by η. The result, which can be seen in eq. (4), can be read as the change in $\boldsymbol{w}$ at each step, divided by the size of the step equals the drift plus the noise. Therefore, should we compare, side by side, the result with the derivative formula, we can argue that their structure is very similar, with the only discrepancy that our update rule is inherently discrete.

$$\frac{dp}{dt} = \lim_{\Delta t \to 0} \frac{p(t+\Delta t) - p(t)}{\Delta t} \tag{3}$$

$$\frac{\boldsymbol{w}_{t+1} - \boldsymbol{w}_t}{\eta} = \boldsymbol{f}(\boldsymbol{w}_t) + \boldsymbol{\xi}(\boldsymbol{w}_t, \boldsymbol{x}_t) \tag{4}$$

We can now define a continuous time variable that takes both η and t, such that, as η becomes infinitesimal, our new time variable will become a continuous coordinate on the real axis. Formally, let us introduce $\alpha = \eta \cdot t$. Then, $\boldsymbol{w}_t$ becomes a function of α, written as $\boldsymbol{w}(\alpha)$, and the next step $\boldsymbol{w}_{t+1}$ becomes $\boldsymbol{w}(\alpha + \eta)$. Substituting eq. (4) with the aforementioned time variable and taking η under the limit of 0, we obtain eq. (5). By interpreting it, we could conclude that, at every instant of continuous time α, the velocity of the weight vector equals drift plus noise.

$$\frac{d\boldsymbol{w}}{d\alpha} = \lim_{\eta \to 0} \frac{\boldsymbol{w}(\alpha+\eta) - \boldsymbol{w}(\alpha)}{\eta} = \boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{\xi}(\boldsymbol{w}, \boldsymbol{x}) \tag{5}$$

A question that we could ask ourselves after visualizing the motorway analogy is the following: in eq. (5), can't we drop the latter term in the long run? That is, isn't noise insignificant? To see if we could simplify the dynamics, let us take a look at the discrete weight update. Formally, that is $\boldsymbol{w}_T = \boldsymbol{w}_0 + \eta \sum_{t=0}^{T-1} \boldsymbol{f}(\boldsymbol{w}_t) + \eta \sum_{t=0}^{T-1} \boldsymbol{\xi}(\boldsymbol{w}_t, \boldsymbol{x}_t)$. The drift is deterministic, thus it has some approximate magnitude D. As such, we have $\|\eta \sum_t \boldsymbol{f}(\boldsymbol{w}_t)\| \approx \eta\, D\, T$, where $\|\cdot\|$ denotes the vector's norm and T is the discrete time variable.

The noise is a sum of T zero mean, approximately[2] independent random variables, thus, by the Central Limit Theorem, the variances add, hence the standard deviation of the sum grows as the square root of time. Formally, that is $\|\eta \sum_t \boldsymbol{\xi}(\boldsymbol{w}_t, \boldsymbol{x}_t)\| \approx \eta\, \sigma_\xi \sqrt{T}$. In comparison with our motorway example, imagine that, since the left and right steps partially cancel out, the noise grows much more slowly than the drift. Let us then take the limit of the magnitude of the noise over the magnitude of the drift, as follows.

$$\lim_{T \to \infty} \frac{\|\eta \sum_t \boldsymbol{\xi}(\boldsymbol{w}_t, \boldsymbol{x}_t)\|}{\|\eta \sum_t \boldsymbol{f}(\boldsymbol{w}_t)\|} \approx \lim_{T \to \infty} \frac{\eta\, \sigma_\xi \sqrt{T}}{\eta\, D\, T} = \lim_{T \to \infty} \frac{\sigma_\xi}{D\sqrt{T}} = 0 \tag{6}$$

The limit being zero implies that, in the long run behaviour, drift dominates whilst the jitter remains bounded, hence we can drop the noise term from eq. (5) and arrive to the Ordinary Differential Equation the weight trajectory follows, which makes it the backbone of the fixed-point analysis.

$$\frac{d\boldsymbol{w}}{d\alpha} = \boldsymbol{f}(\boldsymbol{w}) = \mathbb{E}[\Delta \boldsymbol{w}(\boldsymbol{w}, \boldsymbol{x})] \tag{7}$$

We can finally address the "velocity of the weight vector" we have mentioned earlier. As such, $\boldsymbol{f}(\boldsymbol{w})$ depicts a vector field, where, at every point $\boldsymbol{w}$ in space, the function tells us the direction $\boldsymbol{w}$ wants to move to and how fast. A *fixed point* is a point $\boldsymbol{w}^*$ where the velocity is zero. Formally, that is:

$$\boldsymbol{f}(\boldsymbol{w}^*) = \mathbb{E}[\Delta \boldsymbol{w}(\boldsymbol{w}^*, \boldsymbol{x})] = \boldsymbol{0} \tag{8}$$

[2]We say *approximately* because the fluctuation is a function of $\boldsymbol{w}_t$, which inherently depends on past inputs, therefore the terms are not strictly independent. Nonetheless, since the inputs are drawn i.i.d., for small η, the noise tends to behave as independent.

Equation (8) represents one half of the fixed-point analysis. By averaging the plasticity rule and setting it to the zero vector, we find all the points where the weight update is at rest. We highlight that *"convergence can take place only if the noise is rejected by paying less and less attention to the noisy observations"* [7, p. 552], thus a vanishing learning rate is essential. Were it to be purely constant, the weights would settle in the neighbourhood of $\boldsymbol{w}^*$ with residual fluctuation.

Nonetheless, there still is one question that remains: what happens if we were to nudge the weight vector off its resting point? Answering it is the stability analysis. A fixed point belongs to one of three types, only one of them (i.e., the stable one) is a resting point of the dynamics. A fixed point $\boldsymbol{w}^*$ is *stable* if every small perturbation decays. That is, for any small displacement ϵ, the drift $\boldsymbol{f}(\boldsymbol{w}^* + \epsilon)$ points back towards $\boldsymbol{w}^*$. Moreover, it is a repeller if every direction of the perturbation grows and a saddle if some directions decay whilst others grow. Those are both *unstable*, although saddle points may still shape the transient trajectory.

To make the random walk converge, we are interested in the stable points. After solving the eq. (8) for fixed points, we can compute the Jacobian of the drift. Formally, that is $\boldsymbol{J}(\boldsymbol{w}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}$. Then, we plug in each fixed point and compute the eigenvalues of $\boldsymbol{J}^* = \boldsymbol{J}(\boldsymbol{w}^*)$. The sign of the real parts of these eigenvalues then determines the nature of each fixed point, as summarized in table 1, thus concluding the fixed-point analysis.

Table 1: Classification of a fixed point $\boldsymbol{w}^*$ based on the eigenvalues of the Jacobian $\boldsymbol{J}^*$ [8].

Eigenvalue condition	**Type**	**Stability**
$\mathrm{Re}(\lambda_i) < 0$ for all i	Attractor	Stable
$\mathrm{Re}(\lambda_i) > 0$ for at least one i	Repeller / Saddle	Unstable
$\mathrm{Re}(\lambda_i) = 0$ for some i, none positive	Inconclusive	Requires nonlinear analysis

For Oja and SoftHebb, we do not rederive their fixed points, as they had already been obtained in [3] and [5], respectively. The recipe instead hereby supplies the principle that a rule's stable fixed point is the direction its online dynamics converge to, therefore, in addition with the shared stability vocabulary, the agreement question could be posed precisely. The following subsections formulate each rule's fixed points and stability.

2.2 Oja's Rule [3]

Oja's rule starts with the simple setup of one single linear neuron with output $y = \boldsymbol{w}^\top \boldsymbol{x}$, for an input $\boldsymbol{x}$ and a presynaptic weight $\boldsymbol{w}$. The problem with the standard Hebbian learning method is that neurons can grow indefinitely, thus Oja adds a normalizing decay that shrinks the weight proportional to the output squared, to bound this aforementioned growth. Formally, $\Delta \boldsymbol{w}^{(\text{Oja})} = \eta \cdot y \cdot (\boldsymbol{x} - y\, \boldsymbol{w})$, where η is the learning rate.

The paper states that the learning rule can stop only at an eigenvector of the covariance matrix C. It further shows that the only stable resting point of the rule is the top eigenvector of the covariance matrix, whilst the others are unstable equilibria.

2.3 SoftHebb's Rule [5]

SoftHebb "*is similar to Oja's rule*"[5, p. 8] and applies this idea within a soft winner-take-all network of K neurons. That is, instead of a hard winner-take-all setting where there is one absolute winner, neurons are updated proportionally to their activation level by giving them a soft score. This assignment depicts how strongly neuron k claims the current input relative to the other neurons.

Inputs and weights are normalized: $\boldsymbol{x}^* = \boldsymbol{x}/\|\boldsymbol{x}\|$ and $\boldsymbol{w}_k^* = \boldsymbol{w}_k/\|\boldsymbol{w}_k\|$, therefore the preactivation $u_k = \boldsymbol{w}_k^* \cdot \boldsymbol{x}^*$ becomes the cosine similarity. The output is the softmax posterior $y_k = Q(C_k \mid \boldsymbol{x}, \boldsymbol{w}) = \dfrac{\pi_k\, b^{u_k}}{\sum_l \pi_l\, b^{u_l}}$,

where b is the softmax base, π_k is the prior weight of cluster k, and the denominator $\sum_l \pi_l \, b^{u_l}$ is the unnormalized model density $Z\, q(\boldsymbol{x})$. The weight update has the Hebbian factor is the nonlinear soft WTA posterior y_k and the weight decay uses the preactivation u_k. Formally, $\Delta \boldsymbol{w}_k^{(\text{SoftHebb})} = \eta \cdot y_k \cdot (\boldsymbol{x} - u_k \, \boldsymbol{w}_k)$.

The paper states that the learning rule can stop at the normalized component means, under the model's input assumption that the mixture components have narrow supports that do not overlap. Furthermore, the learning rule drives the weights towards the values that make the network's internal probabilistic model reflect the data as closely as possible. It is measured by the Kullback-Leibler divergence $D_{KL}(p \,||\, q)$, which shows how far one probability distribution (i.e., the model q) sits from another (i.e., the data p). We could rewrite KL as $D_{KL}(p \,||\, q) = -H(p) - \mathbb{E}_p[\log q(\boldsymbol{x})]$, where $H(p)$ is the entropy of the data. As it does not depend on the weights, minimizing KL means maximizing the expected log-likelihood.

3 Methodology

3.1 Dataset Generation

We test the agreement question on controlled two dimensional, two cluster Gaussian mixtures rather than on real data, such that the theory makes a pointwise prediction about the converged weight directions. As such, synthetic data let us fix every other factor (e.g., dimension, sample size) and only $r = \frac{\sigma_1}{d}$ varies. Real datasets would entangle it with uncontrollable confounds, therefore they are not used within this scope.

A single generator produces every dataset, that is, a balanced two component Gaussian mixture with a shared within cluster covariance $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2)$ and two centers placed symmetrically about the origin on one coordinate axis. Hereinafter we write d for the distance from the origin to each center. We use N i.i.d. samples per cluster, stack them and shuffle. The following two separation conventions are used: (i) isotropic clusters, with centers at $(\pm d, 0)$ and $\sigma_1 = \sigma_2$; (ii) anisotropic clusters, with centers at $(0, \pm d)$ and $\sigma_1 > \sigma_2$. The former's clusters are circular and the top eigenvector of the total covariance lies along the x-axis. The latter's within cluster elongation is along the x-axis, whilst the separation is along the y-axis.

Table 2: The exact dataset configurations.

Key	$\boldsymbol{\sigma_1}$	$\boldsymbol{\sigma_2}$	$\boldsymbol{d}$	**Sep. axis**	$\boldsymbol{r = \sigma_1 / d}$	N	**Role**
Isotropic	1.0	1.0	3.0	x	0.33	2500	agreement, section 4.1
Anisotropic	$r \cdot d$	0.1	1.0	y	$0.7 \to 2.2$	1200	divergence sweep, section 4.2

3.2 Experimental Details

We use the streaming forms of both rules, updating after each sample over shuffled epochs. We added two numerical safeguards, which are harmless to the rules' behaviour: first, $K = 2$ initial weights are placed on evenly spaced directions under one random global rotation, therefore no neuron starts "dead" (i.e., with zero responsibility); second, biases w_0 are in $[-50, 50]$ to prevent e^{-w_0} from overflowing. The latter is harmless since a dead neuron's prior is $e^{-50} \approx 0$. Table 3 shows the exact hyperparameters that are used.

We have also provided the results of some custom, targeted experiments for $r = 0.7, 1.1, 2$, which can be found in the Appendix, section C.1. Furthermore, for comparing directions, we use the unoriented angle between any $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, that is, $\arccos |\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle| \in [0°, 90°]$. Oja converges to $\pm \boldsymbol{v}^*$, whilst SoftHebb's neurons are antipodal and may swap labels between seeds, thus the importance is in the line, not in its arrow.

Table 3: Hyperparameters.

Symbol	Value	Meaning	Symbol	Value	Meaning
b	200	SoftHebb softmax base	epochs (sweep)	15	passes per r
η (SoftHebb)	0.03	SoftHebb learning rate	seeds (isotropic)	30	seeds for violin plot
η (Oja)	0.005	Oja learning rate	seeds (sweep)	12	seeds per r
K	2	# SoftHebb neurons	seeds (regimes)	10	seeds for regimes
epochs	30	passes, isotropic/regimes	w_0 clip	± 50	bias overflow guard

We furthermore have a tolerance called *cf_threshold* that turns the question if SoftHebb found the y split into a binary decision, by taking the mean of the unoriented angles between each neuron weight and the separation axis e_y. Should that be below $9.43°$, the run is counted as having converged to the y split. This is a measurement convenience and is not a theoretical quantity, chosen purely to absorb seed-to-seed jitter, as follows. We run a multi-seed experiment for a fixed $r = 1.1$, in which SoftHebb has a clear convergence to the y split. The unoriented angles between its converged weights and the eigenaxis e_y were recorded. The mean $\mu \approx 3.85$ and the standard deviation $\sigma \approx 1.86$ were calculated, results which could be also seen in the Appendix under table 6. Finally, the threshold was obtained from the cutoff $\mu + 3 \cdot \sigma \approx 9.43$.

4 Theoretical Analysis

4.1 The Isotropic Clusters

Let us begin with finding Oja's convergence. In order to solve for convergence, we have to look at the overall spread of the data cloud. By the law of total covariance, the total spread is the sum of the within and between spreads of the clusters. Let us note this by $C_{total} = C_{within} + C_{between}$.

Since the data is isotropic gaussian, it means that C_{within} is rotational symmetric. To put it differently, the cluster cannot favour any direction, therefore adding the spread to the total covariance implies adding the same variance to all directions, hence $C_{total} \sim C_{between}$. Formally, we find the within spread by:

$$C_{within} = \mathbb{E}\left[Cov\left(\boldsymbol{x} \mid K\right)\right] = \frac{1}{2}\sigma^2 I_2 + \frac{1}{2}\sigma^2 I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{9}$$

K represents the randomly assigned cluster label, thus it is a random variable, hence $\mathbb{E}[\boldsymbol{x} \mid K]$ becomes a random variable, as well. Let us denote this expectation with $\boldsymbol{\Gamma}$. Then, the overall mean is $\mathbb{E}[\boldsymbol{\Gamma}] = \bar{\boldsymbol{\mu}} = P(K_1)\,\boldsymbol{\mu}_1 + P(K_2)\,\boldsymbol{\mu}_2 = \frac{1}{2}(-3, 0)^\top + \frac{1}{2}(3, 0)^\top = (0, 0)^\top$. The between spread is:

$$C_{between} = Cov\big(\mathbb{E}[\boldsymbol{x} \mid K]\big) = \mathbb{E}\big[(\boldsymbol{\Gamma} - \bar{\boldsymbol{\mu}})(\boldsymbol{\Gamma} - \bar{\boldsymbol{\mu}})^\top\big] \tag{10}$$

$$C_{between} = \sum_k P(K = k)\,(\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})^\top = \begin{pmatrix} 9 & 0 \\ 0 & 0 \end{pmatrix} \tag{11}$$

By adding eq. (9) and eq. (11), we obtain $C_{total} = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}$, which is a diagonal matrix, hence its eigenvectors are $\boldsymbol{v_1} = (1, 0)^\top$ and $\boldsymbol{v_2} = (0, 1)^\top$, with eigenvalues $\lambda_1 = 10$, respectively $\lambda_2 = 1$. By section 2.2, we know that the stable fixed point of Oja's rule is the top eigenvector of the covariance matrix of the data, hence Oja's weight vector converges to the x-axis, or, formally, $\boldsymbol{w}^*_{\text{Oja}} = \boldsymbol{v}_1 = (1, 0)^\top$.

For SoftHebb, the work is much simplier. By section 2.3, its weight vectors converge to the normalized component means. As such, for our synthetic data, we have $\boldsymbol{w}^*_{\text{SH-k}} = \mu_k / \|\mu_k\|$, which yields $\boldsymbol{w}^*_{\text{SH1}} = (-1, 0)^\top$ and $\boldsymbol{w}^*_{\text{SH2}} = (1, 0)^\top$. Both lie on the x-axis, thus, under the unoriented angle convention, every pairwise angle between Oja's and SoftHebb's converged weight vectors is zero, hence we have reached alignment.

4.2 The Anisotropic Clusters

4.2.1 Naive Expectation and Its Failure

Were we to apply the derivation as we have analogously did for the isotropic clusters, we would obtain that SoftHebb rests its weights along the y axis. As Oja lands on the variance axis $\pm e_x$ (i.e., as derived in section 4.2.5), this would naively predict a theoretical $90°$ divergence, which matches the natural expectation pointed out for the second hypothesis in the introduction. Nonetheless, the claim can be falsified both empirically and analytically, since it hides one assumption: SoftHebb does not act on the raw input, but actually on its projection onto the unit circle, as shown in fig. 1.

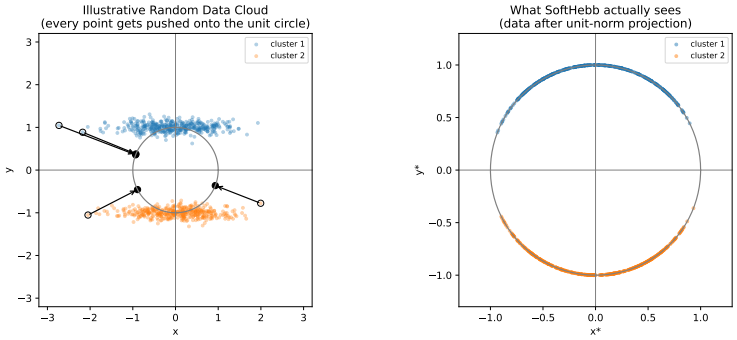


Figure 1: SoftHebb operates on unit normalized inputs. Left: an illustrative anisotropic two cluster data, with black arrows showing example points projected onto the unit circle. Right: the same points after unit normalization, that is, the distribution SoftHebb actually operates on.

When the within cluster width, which is orthogonal to the separation axis, is large, then the two clusters are flattened into overlapping arcs near $\pm e_x$. SoftHebb's best two way split then becomes left and right instead of top and bottom. To put it differently, the spread that is orthogonal to the separation destroys the clusters projected onto the circle, since that is what pushes the points sideways, towards $\pm e_x$. The y axis separation, or, informally, the cluster split, remains an equilibrium of the rule, yet it is simply no longer the one the dynamics select, since the partition that maximizes SoftHebb's alignment score becomes left and right.

In this scenario, σ_1 is the spread orthogonal to the separation (i.e., the standard deviation along x-axis). The spread along the separation is σ_2 (i.e., along y-axis), which tickens the arcs without merging them. Therefore, the quantity that decides the convergence outcome is the ratio $r = \frac{\sigma_1}{d}$. We will use σ_2 merely as a second order correction, that is, by first deriving SoftHebb's behaviour in the $\sigma_2 = 0$ limit and restore it in the end.

4.2.2 Reducing SoftHebb's Objective

We can then reduce SoftHebb's distribution objective to a simple alignment score for the two cluster data as follows. Writing the model density with prior weights π_k, a softmax base b and a normalizer Z, we have $\log q(\boldsymbol{x}) = \log \sum_k \pi_k b^{\boldsymbol{w}_k \cdot \boldsymbol{x}} - \log Z$. The expression in the former term is a log-sum-exp [9, p. 72] with the bound $\max_k a_k \leq \log \sum_{k=1}^K e^{a_k} \leq \max_k a_k + \log K$, for all $a_k \in \mathbb{R}$.

We analyze first the hard winner-take-all limit, where each point is claimed entirely by its single best matching neuron and restore the base $b = 200$ later in the subsection. By the log-sum-exp bound, the gap $\log K/\ln b$ shrinks to zero as the base grows, that is, $\frac{1}{\ln b} \log \sum_k \pi_k b^{\boldsymbol{w}_k \cdot \boldsymbol{x}} \xrightarrow[b\to\infty]{} \max_k \boldsymbol{w}_k \cdot \boldsymbol{x}$. The remaining term $-\frac{1}{\ln b} \log Z$ is the same for every unit weight direction, thus it does not affect maximization. Hence maximizing the expected log likelihood reduces to $\max_{\{\boldsymbol{w}_k\}} \mathbb{E}\left[\max_k \boldsymbol{w}_k \cdot \boldsymbol{x}\right]$.

The data is invariant not only under the point reflection $\boldsymbol{x} \mapsto -\boldsymbol{x}$, but also under each axis reflection $\boldsymbol{x} \mapsto (-x_1, x_2)$ and $\boldsymbol{x} \mapsto (x_1, -x_2)$. Therefore, for two neurons, the symmetric solution places them antipodally on an unit circle. Formally, that is $\boldsymbol{w}_{SH1} = \boldsymbol{\omega}(\theta) = (\cos\theta, \sin\theta)^\top$ and $\boldsymbol{w}_{SH2} = -\boldsymbol{\omega}(\theta)$. As such, we know that a score of a point is how well it is "winning" neuron matches it, that is, the point is assigned to one of the two neurons it aligns with the best, therefore the score is that best alignment. Consider we have a normalized point $\boldsymbol{p}^* = (x^*, y^*)^\top$. Then, since the neurons are antipodal, the larger of a number and its negative its just the absolute value, hence $\max(\boldsymbol{w}_{SH1} \cdot \boldsymbol{p}^*, \boldsymbol{w}_{SH2} \cdot \boldsymbol{p}^*) = \max(\boldsymbol{\omega} \cdot \boldsymbol{p}^*, -\boldsymbol{\omega} \cdot \boldsymbol{p}^*) = |\boldsymbol{\omega} \cdot \boldsymbol{p}^*|$.

The original problem had two weight vectors (i.e., 4 numbers) to optimize, but, as per the antipodal symmetry, it reduces to only one direction $\boldsymbol{\omega}(\theta)$, thus the score of every point becomes $|\boldsymbol{\omega} \cdot \boldsymbol{p}^*| = |\cos\theta\, x^* + \sin\theta\, y^*|$. Averaging over the data, we can derive an objective function that depends only on a single angle θ, that is:

$$S(\theta) = \mathbb{E}\left[|\cos\theta\, x^* + \sin\theta\, y^*|\right] \tag{12}$$

Having the objective function set, we now want to show that there are only two candidate splits that matter: first, the x-split, formally, $S(0) = \mathbb{E}\left[|x^*|\right]$; second, the y-split or the cluster split, formally, $S(\frac{\pi}{2}) = \mathbb{E}\left[|y^*|\right]$. Let us be focused on the cluster split condition (i.e., having $\mathbb{E}\left[|y^*|\right] \geq \mathbb{E}\left[|x^*|\right]$). The inequality depends inherently on two parameters, σ_1 and d, but geometrically only their ratio matters, since scaling the whole picture does not change which split wins, since everything lives on the unit circle after projection. Therefore, we ought to find a way to collapse the condition to depend on a single number, that is the ratio $r = \frac{\sigma_1}{d}$, which implies we will have a one parameter function and asks ourselves for which r the clusters split is winning.

4.2.3 The Governing Ratio and the Function $g(r)$

Let us take a point drawn from the top cluster. It has a horizontal coordinate of $X \sim \mathcal{N}(0, \sigma_1^2)$ and a fixed vertical coordinate d, since the vertical spread is, as previously mentioned, zero. Therefore, the distance from the origin is $\sqrt{X^2 + d^2}$ and projecting to the unit circle by dividing by that aforementioned length yields:

$$|x^*| = \frac{|X|}{\sqrt{X^2 + d^2}}, \qquad |y^*| = \frac{d}{\sqrt{X^2 + d^2}}.$$

By the linearity of the expectation, the condition becomes $\mathbb{E}\left[\frac{d - |X|}{\sqrt{X^2 + d^2}}\right] > 0$. We can set $t = X/d$, and, since $\sigma_1 = rd$, it is thus drawn from $\mathcal{N}(0, r^2)$. Substituting yields the aforementioned one parameter function, that is $g : (0, \infty) \to \mathbb{R}$, $g(r) = \mathbb{E}_{t\sim\mathcal{N}(0,r^2)}\left[\frac{1-|t|}{\sqrt{1+t^2}}\right]$. Since the denominator is always positive, should $1 - |t|$ be positive, the condition would be true, thus the point "votes" for the cluster split.

Let us prove g is strictly decreasing. Since the integrand depends on t through $|t|$ and t^2, we can show it is an even function, therefore its entire behaviour is determined by what happens at $t \geq 0$. Furthermore, let us take the function $h : (0, \infty) \to \mathbb{R}$, $h(t) = \frac{1-t}{\sqrt{1+t^2}}$. As the function is a composition of elementary functions, it this continuous, thus differentiable. Then, $h'(t) = -\frac{1+t}{(1+t^2)^{3/2}} \leq 0,\ \forall\, t \geq 0$, therefore h is strictly decreasing in $|t|$. Increasing r spreads the Gaussian towards larger $|t|$, making $\mathbb{E}[h]$ lower, hence g is strictly decreasing, thus it has a unique root $r^\star$. By taking the limits to the endpoints yields table 4, the sign and variation of g.

Table 4: The sign and variation table for the function g. When $r \in (0, r^\star)$, the y-axis split is preferred.

r	0^+			$r^\star$			$+\infty$
$g'(r)$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$g(r)$	1	$\searrow$	$\searrow$	0	$\searrow$	$\searrow$	-1
		cluster split					

We can now evaluate $\mathbb{E}[|x^*|]$ and $\mathbb{E}[|y^*|]$ in their closed form, in order to obtain the point $r^\star$ where $g(r^\star) = 0$. Starting from the definition with $t \sim \mathcal{N}(0, r^2)$, the expectation over the x part is as follows.

$$\mathbb{E}[|x^*|] = \int_{-\infty}^{\infty} \frac{|t|}{\sqrt{1+t^2}} \cdot \frac{1}{r\sqrt{2\pi}} e^{-t^2/(2r^2)}\, dt = \frac{2}{r\sqrt{2\pi}} \int_0^{\infty} \frac{t}{\sqrt{1+t^2}} e^{-t^2/(2r^2)}\, dt, \tag{13}$$

As we used the even integrand property, the absolute value goes away. By following the substitutions $v = t^2$, $w = 1 + v$, $w = q^2$, $l = \frac{q}{r\sqrt{2}}$, we obtain the closed form $\mathbb{E}[|x^*|] = e^{1/(2r^2)} \operatorname{erfc}\Big(\frac{1}{r\sqrt{2}}\Big)$. For the y part, we can begin by substituting $t = ru$, where $u \sim \mathcal{N}(0, 1)$, as the original distribution is not altered. Then, the expectation over the y part becomes $\mathbb{E}[|y^*|] = \mathbb{E}_u\left[\frac{1}{\sqrt{1+r^2u^2}}\right]$. As we cannot integrate directly the outer power that is wrapped around a Gaussian, we have to turn it into an integral and cancel the Gaussian. One way is to use the Schwinger parameterization trick [10, eqs. (4) and (7)], in order to turn any outer power into:

$$c^{-1/2} = \frac{1}{\sqrt{\pi}} \int_0^{\infty} s^{-1/2} e^{-cs}\, ds \tag{14}$$

$$\mathbb{E}[|y^*|] = \frac{1}{\sqrt{\pi}} \mathbb{E}_u \int_0^{\infty} s^{-1/2} e^{-s} e^{-r^2u^2 s}\, ds. \tag{15}$$

As such, by substitution of $c = 1 + r^2u^2$, we arrive to the latter equation. As everything is positive, we may swap the expectation and the integral and arrive to the inner $\mathbb{E}_u[e^{-u^2r^2s}]$, which is an average over a standard normal u. As u^2 follows a chi-squared distribution with one degree of freedom and the argument (i.e., $-r^2s$) is negative, we can substitute in the chi-squared's moment generating function[11, Table of Common Distributions] and arrive to $\mathbb{E}_u[e^{-u^2r^2s}] = \frac{1}{\sqrt{1+2r^2s}}$, making the random variable to go away. Hence,

$$\mathbb{E}[|y^*|] = \frac{1}{\sqrt{\pi}} \int_0^{\infty} s^{-1/2} e^{-s} (1 + 2r^2 s)^{-1/2}\, ds. \tag{16}$$

We can then pull out r of the last factor, merge the square roots, substitute $s = \frac{1}{4r^2}(\cosh t - 1)$ and finally obtain the closed form of the y part, that is $\mathbb{E}[|y^*|] = \frac{1}{r\sqrt{2\pi}} e^{1/(4r^2)} K_0\big(\frac{1}{4r^2}\big)$. By setting $\mathbb{E}[|x^*|] = \mathbb{E}[|y^*|]$ and solving numerically, we obtain $r^\star \approx 1.6872$. Therefore, for any $r < r^\star$, the cluster split is the global optimum for S and, for any $r > r^\star$, the x split wins.

4.2.4 Stability of the Candidate Splits

As SoftHebb does not magically jump to the best answer, each step nudges the angle θ in the direction that increases the objective function, thus it climbs uphill on S from wherever it happens to start, like a ball rolling uphill to the nearest peak, settling at whatever local maximum it can reach. Even when the cluster split is the highest peak, should the x split be also a peak, some seeds will get stuck on it.

To find if it is either a peak or a valley, let us check its curvature. From eq. (12), let us denote the inside of the expectation with q, that is $S(\theta) = \mathbb{E}[|q|]$. Differentiating the objective function gives $S'(\theta) = \mathbb{E}[\text{sgn}(q)q']$. Differentiating the second time yields $S''(\theta) = -S(\theta) + 2\,\mathbb{E}[\delta(q)(q')^2]$. The Dirac Delta counts how much data sits on the dividing line $L = 0$, which makes every point sitting on the boundary to contribute a little upwards push to the curvature, thus lots of points on the boundary implies that the axis becomes a valley, whilst no points on the boundary implies no push, hence it is a peak. Hereby we analyze two scenarios.

Cluster split ($\theta = \pi/2$). The point $q = y^* = \pm d/\sqrt{X^2+d^2}$ only approaches zero under the limit $|X| \to \infty$, where no probability mass lives, thus there is not any data that sits on the boundary and the Delta Dirac vanishes, hence $S''(\frac{\pi}{2}) = -S(\frac{\pi}{2}) < 0$. This implies that the cluster split is a local maximum for every r.

X split ($\theta = 0$). The point $q = x^* = X/\sqrt{X^2+d^2}$ crosses zero when $X = 0$, that is, the horizontal center of each cluster, where the density is plenty. Thus, Delta Dirac term survives, yielding $\mathbb{E}[\delta(x^*)(q')^2] = \rho_{x^*}(0) \cdot (y^*)^2|_{X=0} = \rho_{x^*}(0)$. As the map $X \mapsto \frac{X}{\sqrt{X^2+d^2}}$ is strictly increasing, the density of $q = x^*$ at 0 follows from the change of variables $q = X(X^2+d^2)^{-1/2}$ with $\rho_{x^*}(0) = \frac{\rho_X(0)}{|dq/dX|} = \rho_X(0)\,d = \frac{1}{r\sqrt{2\pi}}$, which yields $S''(0) = -\mathbb{E}[|x^*|] + \frac{2}{r\sqrt{2\pi}}$. By plotting the graph of this function we can see that, for small r, it is positive, thus the x split is a local minimum, whereas for larger values of r, the function is negative, thus the x split is a local maximum. The function moves monotonically with r and by solving numerically for $S''(0) = 0$ yields the boundary $r_{loc} \approx 1.3302$. The plot of $S''(\theta)$ can be seen in fig. 6 from the Appendix.

4.2.5 Finite Base/Width Corrections & The Three Zones

In order to derive the zones in which $r^\star$ and r_{loc} can be combined, let us derive Oja's resting point for this scenario. Analogously to the derivation from the isotropic clusters, we obtain that the total spread C is $\text{diag}(\sigma_1^2, \sigma_2^2 + d^2)$, thus the eigenvectors are the x/y axes, with eigenvalues σ_1^2 and $\sigma_2^2 + d^2$. Oja takes the larger, hence $w^*_{\text{Oja}} = \pm e_x \iff \sigma_1^2 > \sigma_2^2 + d^2$, which is equivalent to $r > \sqrt{1 + (\sigma_2/d)^2} \xrightarrow{\sigma_2 \to 0} r > 1$.

Let us move from the hard winner take all to a finite base. As the two neurons are placed antipodally with equal priors, a point with its projection $\hat{\boldsymbol{v}} = \boldsymbol{\omega} \cdot \boldsymbol{p}^*$ has its weight mixture term $\frac{1}{2}\exp(\ln b \cdot \hat{\boldsymbol{v}}) + \frac{1}{2}\exp(-\ln b \cdot \hat{\boldsymbol{v}}) = \cosh(\ln b \cdot \hat{\boldsymbol{v}})$, hence the finite base objective becomes $S_{\ln b}(\theta) = \frac{1}{\ln b}\mathbb{E}[\log\cosh(\ln b \cdot p)]$. The base b controls how sharp the kink is at $\hat{\boldsymbol{v}} = 0$ and, since the cluster split places no mass at that point whilst the x split does, it implies that only the x split's curvature is affected.

We quantify each correction by resolving the same two root conditions (i.e., $g(r^\star) = 0$ and $S''(0) = 0$ for r_{loc}). Replacing the hard winner-take-all limit with $b = 200$ moves the cutoffs $(r_{loc}, r^\star) \approx (1.330, 1.687)$ to $(1.351, 1.668)$. That is, $+1.6\%$ and -1.1%, respectively. Furthermore, we reinstated $y = d + Y$, with $Y \sim \mathcal{N}(0, \sigma_2^2)$ and averaged the objective over Y, therefore we restored the width that is along the separation, hence the finite base thresholds were shifted to $(1.349, 1.661)$. That is, -0.13% and -0.44%, respectively. Additionally, Oja's flip condition becomes $\sqrt{1 + (\sigma_2/d)^2} = 1.005$, which yields a $+0.5\%$ change.

Combining Oja's flip with SoftHebb's thresholds, we may partition the r axis into three distinct zones. **Zone I** $(1.005 < r < r_{loc} \approx 1.35)$, in which the x split is a local minimum and the cluster split is the only maximum, thus SoftHebb converges cleanly to the y split. **Zone II** $(r_{loc} \approx 1.35 < r < r^\star \approx 1.66)$, where the x split is a competing local maximum whilst the y split is a global one, therefore the split SoftHebb settles upon is seed dependent. **Zone III** $(r > r^\star \approx 1.66)$, in which the x split overtakes the y split as the global maximum, thus SoftHebb converges to $\pm e_x$. For $r < 1.005$, Oja is not flipped and rests, along with SoftHebb, on $\pm e_y$. As such, agreement holds at small and large values of r and divergence is bounded by the window $(1.005, \approx 1.66)$, with a seed dependent region starting at $r_{loc} \approx 1.35$. This concludes the theoretical analysis.

5 Experimental Results & Discussion

5.1 Isotropic Agreement

We test the isotropic two cluster model of section 4.1 by running the fixed configuration over 30 seeds at softmax base $b = 200$ and, for each seed, record the converged weights and the eigenaxis e_x as a reference.

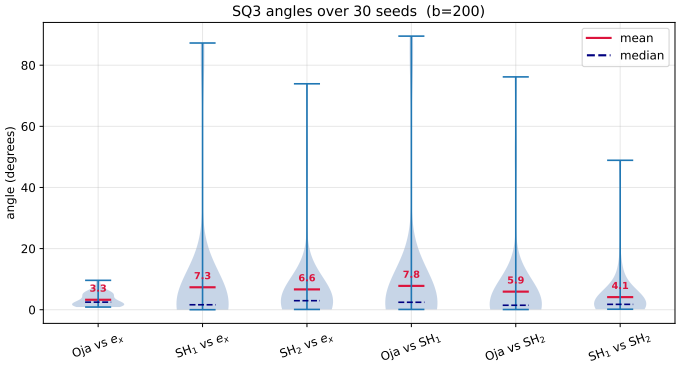


Figure 2: Violin plot showing the distribution of all six pairwise unoriented angles across 30 seeds.

Looking at fig. 2, we can see the isotropic prediction where Oja's top eigenvector and both SoftHebb's centroids collapse to the same x-axis direction, therefore, ideally, every unoriented pairwise angle should be zero. The relatively large eigengap $\lambda_1 = 10$ and $\lambda_2 = 1$ makes Oja's direction consistent across seeds. That is, around $\approx 3.3^\circ$ to e_x, the smallest mean deviation of the six. SoftHebb, as it has two centroids to place instead of one, accumulates a little more seed to seed angular jitter.

With hindsight, the isotropic experiment did agree not because it was different, but because its geometry places it in the $r_{\text{isotropic}} = 0.33 < 1$ regime where the variance axis and the cluster axis are the same. Therefore, both experiments are not two phenomena, but two points on the same r axis, with the two caveats: first, one is isotropic, whereas the other one is derived in anisotropic, zero width limit, thus the qualitative mapping (i.e., small r implies agreement) holds, but the exact thresholds were computed for a different covariance structure; second, the deformation is different, since pushing r upwards means stretching one axis in anisotropic, conversely to both axis in isotropic, hence we have different perturbations in two geometries.

5.2 Anisotropic Bounded Divergence

We test the anisotropic two cluster model of section 4.2 by sweeping the ratio r across $[0.7, 2.2]$ at a fixed softmax base $b = 200$ and recording, at each r, the converged direction of each rule. We therefore report three views on this sweep: first, the rate at which SoftHebb prefers the cluster split (i.e., fig. 3); second, the magnitude of the divergence (i.e., fig. 4a); third, the eigenaxis each rule settles on (i.e., fig. 4b). The reader may also check section C.1 for multi-seed runs for values of $r = 0.7, 1.1, 2$ for a more in-depth visualization.

The convergence axis sweep isolates the mechanism most directly, as Oja flips almost discontinuously from $\pm e_y$ to $\pm e_x$ at $r \approx 1.005$. SoftHebb remains on $\pm e_y$ longer and switches towards $\pm e_x$ gradually, between $r \approx 1.4$ and $r \approx 1.8$. The divergence plot quantifies this gap. The closest unoriented Oja and SoftHebb angle is around zero for $r < 1$, jumps at Oja's flip to $\approx 57^\circ$ and stands at $\approx 80^\circ$ across $r \in [1.05, 1.32]$ before decaying through $r_{loc} \approx 1.35$ and collapsing to around 15° after $r^\star \approx 1.66$. The empirical divergence window coincides with the predicted one and the fall through Zone II is gradual rather than a single step.

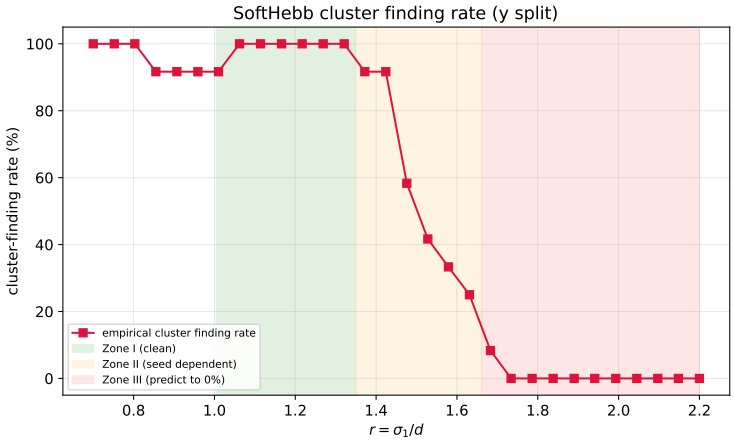
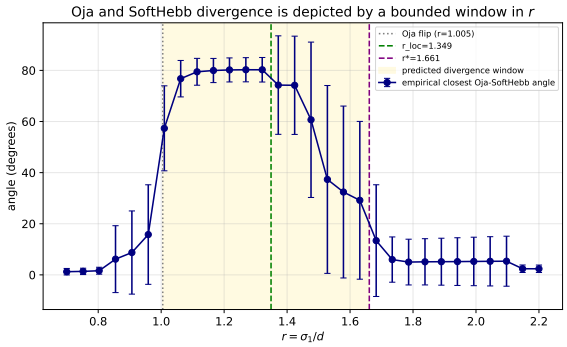
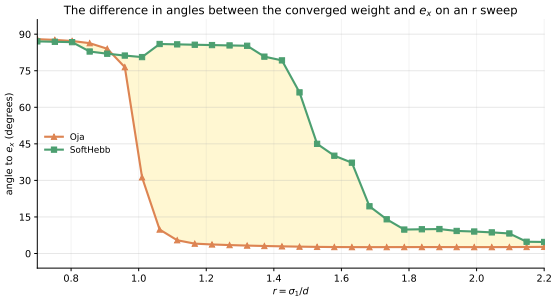


Figure 3: SoftHebb's cluster finding rate across the r sweep. That is, the fraction of seeds for which SoftHebb converges to the y split ($\sigma_1 = r \cdot d$, $\sigma_2 = 0.1$, $d = 1$, separation along y; 12 seeds per r, $b = 200$). A run counts as cluster found when its mean neuron to e_y angle is below $cf\_threshold = 9.43°$.



(a) Divergence magnitude, that is, the unoriented angle between Oja's weight and its nearest SoftHebb neuron, across r sweep (mean $\pm$ std over 12 seeds).



(b) The eigenaxis each rule settles on, shown as the angle between each converged weight and the e_x eigenaxis across the r sweep (mean over 12 seeds; spread is omitted for clarity, for variance the reader can see fig. 4a).

Figure 4: Two views of the anisotropic r sweep ($\sigma_1 = r \cdot d$, $\sigma_2 = 0.1$, $d = 1$, separation along y; 12 seeds, $b = 200$). The former depicts the magnitude of the Oja and SoftHebb divergence, whilst the latter portrays the eigenaxis each rule converges to, through the angular difference between the converged weight and e_x.

The cluster finding curve measures whether SoftHebb stills prefers the cluster split, or, in other words, if it still acts as k-means. The same zone structure as described in section 4.2.5 is further observed. That is, $\approx 100\%$ for Zone I, a monotone drop to $\approx 25\%$ by the end of Zone II and continuing to decline through Zone III till $\approx 0\%$, near $r \approx 1.75$. Hence, all three views report the same mechanism: Oja switches as the elongation overtakes the separation and SoftHebb, whilst acting on unit normalized inputs, holds the cluster split until $r^\star \approx 1.66$, when the orthogonal spread is finally large to collapse the two clusters into overlapping arcs near $\pm e_x$. This falsifies the naive expectation and replaces it with a quantitative amount controlled by r.

Two deviations may be worth stating. First, the divergence saturates near $\approx 80°$ rather than a perfect right angle split. We read this as a finite step convergence and seed-to-seed spread rather than a theoretical effect.

We claim no more than the figures show. Second, the zone boundaries are soft crossovers, not steps: the cluster finding rate decays smoothly and reaches 0% near $r \approx 1.75$ rather than snapping at $r^\star$. Both are consistent with the approximation behind the thresholds, but hereby it accumulates into a smooth transition.

Taken together, the three views point to a single transportable insight rather than a hard fact about the two dimensional data. The sweeps' isolation is not a property of this particular mixture, but the consequence of the two rules reading distinct geometries of the same data. Oja's resting direction is the top eigenvector of the covariance [3], which is fixed by the raw data in Euclidean space, SoftHebb has it fixed by the directional distribution on the unit circle, as it acts on $\frac{\boldsymbol{x}}{||\boldsymbol{x}||}$ rather than $\boldsymbol{x}$ [5]. The targets coincide exactly when those coincide and part only when the variance axis and the dominant axis of the normalized cloud disagree. That is the reason why the divergence is a bounded window in r rather than a naive angle of ninety degrees.

The batch correspondence of [6] is recovered at small r, where the separation dominates the orthogonal spread and the centroid subspace collapses to the leading variance direction. At large r, the within cluster elongation overtakes the separation and both rules follow the e_x eigenaxis. Thus, they agree, but, in the latter case, on the elongation rather than the generative centroids. For the neuromorphic, energy constrained setting that motivates backpropagation free learning rules, the reading is that, on the structured cluster data we tested, the two rules converge to the same direction, with the exception of the identifiable interval in r. We make no claim that this holds for arbitrary cluster geometries, but what we expect to survive is the qualitative claim in higher dimensional structured data. That is, agreement at small and large r with a bounded divergence window in between, with the higher dimensional generalization left open in section 6.

6 Conclusions and Future Work

We set forth from a batch setting result [6], where it was shown that K-means and PCA are tightly linked, with the subspace spanned by the cluster centroids to coincide with the span of the leading principal directions of the data. As such, our question was whether this correspondence survives once the computation is moved online, a regime that matters for neuromorphic and energy constrained hardware, where backpropagation-free alternatives are attractive. To answer it, we first laid out a fixed-point analysis recipe, thus the agreement question could be posed precisely and then made the question testable through two controlled experiments.

The first hypothesis held. When two isotropic clusters are separated along a single axis, it is not only the direction of the largest variance, but also the cluster axis, therefore Oja and SoftHebb converge to the same direction. The second hypothesis did not, as the naive picture (i.e., elongating the clusters orthogonally to the separation converges Oja to the elongation and SoftHebb to the separation) is false, because SoftHebb acts on unit normalized inputs rather then raw data. Therefore, once the orthogonal spread is large enough, the two clusters collapse into overlapping arcs and the SoftHebb's split flips back to agree with Oja.

We replaced the naive claim with a single quantity $r = \frac{\sigma_1}{d}$ that acts as a single organizing coordinate. That is, the rules agree below it, diverge within a bounded window, and agree above it, with a seed dependent caveat where the x split becomes a competing local maximum. The batch correspondence therefore survives in the streaming rules, but conditionally, as it holds at small r and breaks inside a predictable interval. For large r, the rules agree on the elongation axis, thus they no longer recover the cluster structure.

Our analysis was controlled and its scope inherently marks the next steps. The experiments live in two dimensions with two clusters. Thus, extending the streaming analysis to higher dimensions and more clusters would test whether r still organizes agreement or disagreement. A second direction is the boundary itself, since the middle regime is seed dependent and the zone edges behave as soft crossovers. This makes describing the transition analytically rather reading it from seed statistics to remain open. Finally, extending Oja's to Sanger's rule [4] implies learning more than purely the first principal component, which would bring our experiment closer to a more in-depth PCA vs K-means correspondence analysis.

7 Responsible Research

7.1 Reproducibility

The experiments are layed out by a single frozen configuration object, thus every hyperparameter is fixed in one place before anything runs. The exact runtime environment, that is, the Python, NumPy, SciPy, and Matplotlib versions, are recorded at execution time, along with the platform's version (i.e., Windows-11-10.0.26200-SP0). All randomness is fully seeded and each reported run is enumerated by an explicit seed. The closed form expressions (i.e., $\mathbb{E}[|x|]$ and $\mathbb{E}[|y|]$) are additionally double checked by numerical integration within the same notebook, therefore the reported thresholds could be generated independently of the derivations. The complete code, including the data generator, both learning rules, and the scrips that produce the figures, is released in a public GitHub repository[3].

7.2 Usage of AI

We have used AI tools for the preparation of this paper. A large language model was leveraged for rephrasing in written sections and also for providing synonyms or contextual terminologies for specific use cases (e.g., synonyms for the action of describing something, the term "*confound*" for causing confusion when acting against expectations, etc.). Moreover, the model was used as a programming assistant for both the experiments and the scripts for plotting, for instance with code formatting and debugging.

Furthermore, the large language model was used to generate boilerplate LaTeXcode for standard mathematical syntax, tables, and figures. We emphasize that the model was **not** utilized to format BibTeX code for the used references, as they were entirely copied from Google Scholar's citing feature. We further highlight that not only did **not** the large language model replace critical thinking, but its content was always reviewed and modified to ensure high accuracy and reflect our true intent.

7.3 Bias & Scope

The conclusions are conditioned on the data generating process itself, which is a source of bias. We use only balanced Gaussian mixtures of two components, in two dimensions. They have near zero spread along the separation. We confine our claims to this regime and leave the generalization as open in section 6. Additionally, to limit confirmation bias, both predictions were derived analytically before the sweeps were run. We further reported the case in which the prediction failed (i.e., the naive claim) rather than only the confirming one. Finally, the study uses no human, personal, nor demographic data, therefore fairness bias should not arise.

Lastly, in terms of evaluation level bias, we could assess the $cf\_threshold$ cutoff. It is calibrated from a single regime where $r = 1.1$, which shows near optimal convergence. Both the experiment's configuration and its multi-seed results (i.e., in order to compute the cutoff by $\mu + 3 \cdot \sigma$) are listed in the Appendix under section C.1. SoftHebb converging cleanly makes the cluster finding rate inherit the variability of calibration and we emphasized that we report it as a measurement convenience rather than a theoretical quantity.

References

[1] R. Ye, C. Ye, C. Huang, M. Tang, and Y. Liu, "Beyond-backpropagation training: Methods, applications, and perspectives," TechRxiv, vol. 2026, no. 0103, 2026. [Online]. Available: https://www.techrxiv.org/doi/abs/10.36227/techrxiv.176740426.63642005/v1

[2] D. O. Hebb, The organization of behavior: A neuropsychological theory. Psychology press, 2005.

[3] `https://github.com/stan-group/ovidiu_argherie`

[3] E. Oja, "Simplified neuron model as a principal component analyzer," Journal of mathematical biology, vol. 15, no. 3, pp. 267–273, 1982.

[4] B. A. Olshausen, "Linear hebbian learning and pca," 1998.

[5] T. Moraitis, D. Toichkin, A. Journé, Y. Chua, and Q. Guo, "Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks," Neuromorphic computing and engineering, vol. 2, no. 4, p. 044017, 2022.

[6] C. Ding and X. He, "K-means clustering via principal component analysis," in Proceedings of the twenty-first international conference on Machine learning, 2004, p. 29.

[7] L. Ljung, "Analysis of recursive stochastic algorithms," IEEE transactions on automatic control, vol. 22, no. 4, pp. 551–575, 1977.

[8] S. H. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, 2nd ed. CRC Press, 2015. [Online]. Available: https://doi.org/10.1201/9780429492563

[9] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2004.

[10] D. M. Bradley, A. Natian, and S. M. Stewart, "Integration using schwinger parametrization," The American Mathematical Monthly, vol. 131, no. 5, pp. 371–389, 2024. [Online]. Available: https://doi.org/10.1080/00029890.2024.2308457

[11] G. Casella and R. Berger, Statistical inference. Chapman and Hall/CRC, 2024.

[12] G. N. Watson, A treatise on the theory of Bessel functions. The University Press, 1922, vol. 3.

A Background and Preliminaries

A.1 Covariace Matrix & The Law of Total Covariance

The covariance matrix is usually defined as $C = Cov(\boldsymbol{x}) = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})\,(\boldsymbol{x} - \boldsymbol{\mu})^\top]$, but, since our 2D data is centered, we can reduce it to $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]$, thus, entrywise, $C = \begin{pmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_1, x_2) & \text{Var}(x_2) \end{pmatrix}$. The covariance matrix depicts a "summary" of the shape of the data cloud, where the diagonal entries show the spread along each axis, whilst the other outline how the cloud is tilted in comparison with the axis.

Before explaining the law of total covariance, let us start with an analogy. The reader can imagine two professional teams of football, in which the average height of the players is 170 cm, respectively 190 cm. We measure every player and ask ourselves what makes the heights vary? First, there is an inside team variation, which we call *within spread*. Some players might have 165 cm, whilst others might have 175 cm. Second, even if every player within a team has the exact height, the gap between the 170 cm and 190 cm will still create an overall spread, which we call *between spread*.

In statistics, the analogy depicts the law of total variance, which, were we to have vectors instead of scalars, it would become the law of total covariance, which is the sum between the within and between spreads. Formally, that is $Cov(\boldsymbol{x}) = \mathbb{E}\left[Cov\left(\boldsymbol{x} \mid K\right)\right] + Cov(\mathbb{E}\left[\boldsymbol{x} \mid K\right])$, where the former depicts the averaged noise within a cluster and the latter depicts the spread of the cluster centers, provided the expectation exists.

A.2 The Error Function erf & Its Complement

Were the reader to be familiar with probability theory, then they would have likely encountered the bell shaped curve e^{-t^2}. Assume we have a Gaussian random variable Z, that has mean 0 and the standard deviation $1/\sqrt{2}$. Then, the central probability $P(-z \leq Z \leq z)$ equals the error function erf. Formally, that is $\text{erf} : z \mapsto \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$. Furthermore, the complementary error function is the leftover area on the tails of the bell shaped curve, that is $\text{erfc} : z \mapsto 1 - \text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2}\, dt$. Figure 5 shows the functions' plots.

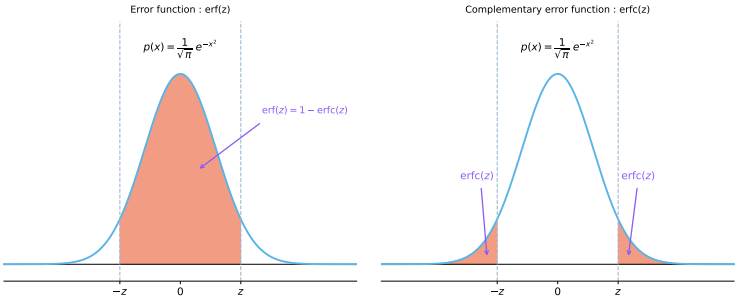


Figure 5: Probability density function $p(x) = \frac{1}{\sqrt{\pi}} e^{-x^2}$ with the regions corresponding to $\text{erf}(z)$ (left) and $\text{erfc}(z)$ (right) that are shaded in orange.

A.3 Modified Bessel Function & Dirac Delta

Similar to how trigonometric functions $sin(x)$ and $cos(x)$ play for straight lines, Bessel functions play for geometry that is shaped like a tube. The reader can imagine, as an analogy, the heat that is spreading through a cylinder. This is an use case where Bessel functions arise naturally. The family has a few branches, but the one we are interested in is K_0, the modified Bessel function of the second kind, order 0. For our use case, its integral representation is sufficient. Thus, as shown in [12, eq. (5), p. 181], $K_\nu(z) = \int_0^\infty e^{-z \cosh t} \cosh \nu t\, dt$, where the hyperbolic cosine $\cosh(x)$ is equal to $(e^x + e^{-x})/2$. Setting $\nu = 0$, the integral reduces to:

$$K_0(z) = \int_0^\infty e^{-z \cosh t}\, dt \tag{17}$$

The Dirac Delta $\delta(x)$ is a generalized function[4] that is zero except at zero, where it is infinite. The integral over $\mathbb{R}$ is one. For our use case, the reader has to know that, when the argument x is a random quantity, then the expectation over the Dirac Delta of x is the probability density of p, evaluated at 0. Formally, that is $\mathbb{E}[\delta(p)] = \rho_p(0)$. We use Dirac Delta because of the kink from the absolute value we will use in our analysis. The function $t \mapsto |t|$ has a sharp corner at zero, and its slope is -1 for negative numbers and 1 for positive numbers, hence its derivative is the sign function $\text{sgn}(t)$. The sign function's graph has two horizontal lines that has a discontinuity at zero, thus it results in a spike whose area is two, hence $\frac{d}{dt} \text{sgn}(t) = 2\delta(t)$.

A.4 Fixed-point analysis

Proof. Since $\boldsymbol{w}$ is fixed, there is only one source of randomness, that is $\boldsymbol{x}$, and the expectation $\mathbb{E}[\,\cdot\,]$ is taken over its distribution. Define $\boldsymbol{m} = \mathbb{E}[\Delta \boldsymbol{w}(\boldsymbol{w}, \boldsymbol{x})] \in \mathbb{R}^2$ and, since the expectation has averaged out the

[4]Mathematical objects that extend regular functions on the real or complex space to treat discontinuities smoothly.

randomness in $\boldsymbol{x}$, the vector $\boldsymbol{m}$ becomes constant. Hence, by the linearity of expectation,

$$\mathbb{E}[\,\Delta \boldsymbol{w}(\boldsymbol{w}, \boldsymbol{x}) - \boldsymbol{m}] = \mathbb{E}[\Delta \boldsymbol{w}(\boldsymbol{w}, \boldsymbol{x})] - \mathbb{E}[\boldsymbol{m}] = \boldsymbol{m} - \boldsymbol{m} = \mathbf{0}, \tag{18}$$

Substituting back the definition of $\boldsymbol{m}$ yields the claim. □

B Theoretical Analysis

B.1 The Anisotropic Clusters

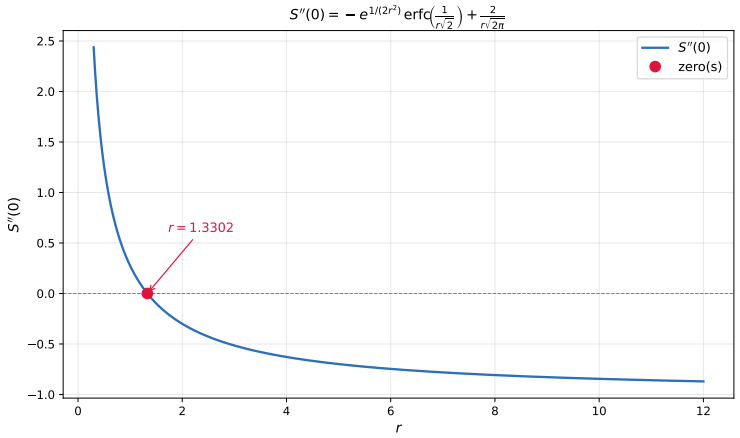


Figure 6: The plot of $S''(\theta)$, the second derivative of the objective function.

C Experimental Results & Discussion

C.1 Anisotropic Bounded Divergence

Table 5: Dataset configurations 2. `npc` = samples per cluster.

Key	$\boldsymbol{\sigma_1}$	$\boldsymbol{\sigma_2}$	$\boldsymbol{d}$	**Sep. axis**	$\boldsymbol{r = \sigma_1/d}$	`npc`	**Role**
Regime A	1.1	0.1	1.0	y	1.1	2500	clean divergence point
Regime B	0.7	0.1	1.0	y	0.7	2500	below Oja flip
Regime C	2.0	0.1	1.0	y	2.0	2500	above r^* collapse

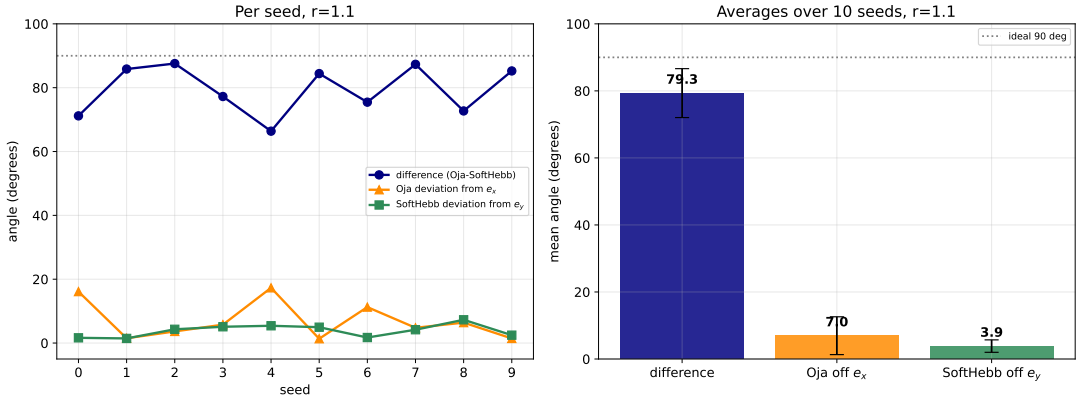


Figure 7: Regime A. The difference between Oja and SoftHebb convergence when the control ratio r is set to 1.1.

Table 6: Recovery angle (deg) vs. seed for $r = \sigma_1/d = 1.100$

Seed	Difference (deg)	Oja $\to e_y$ (deg)	SoftHebb $\to e_y$ (deg)
0	71.17	16.12	1.64
1	85.84	1.49	1.45
2	87.57	3.64	4.31
3	77.24	5.76	5.10
4	66.38	17.30	5.41
5	84.42	1.35	4.96
6	75.46	11.26	1.73
7	87.32	4.78	4.17
8	72.71	6.40	7.29
9	85.25	1.44	2.47
Mean	79.34	–	3.85
Std	7.30	–	1.86

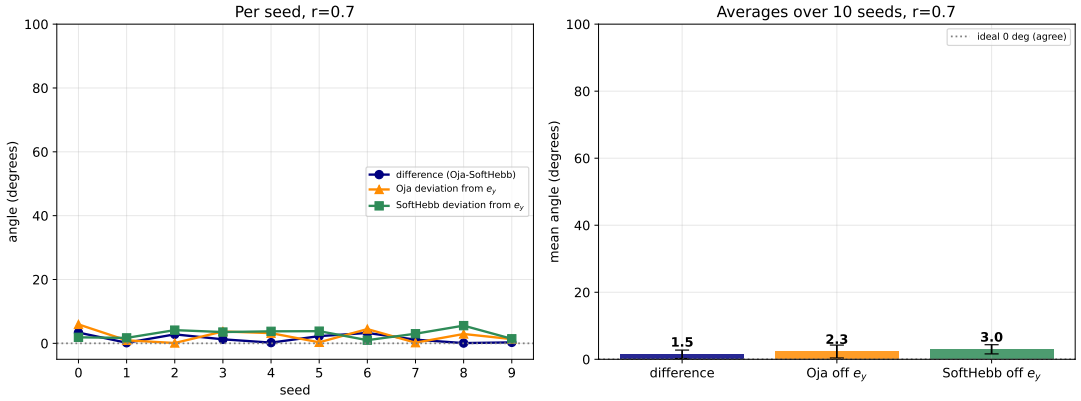


Figure 8: Regime B. The difference between Oja and SoftHebb convergence when the control ratio r is set to 0.7.

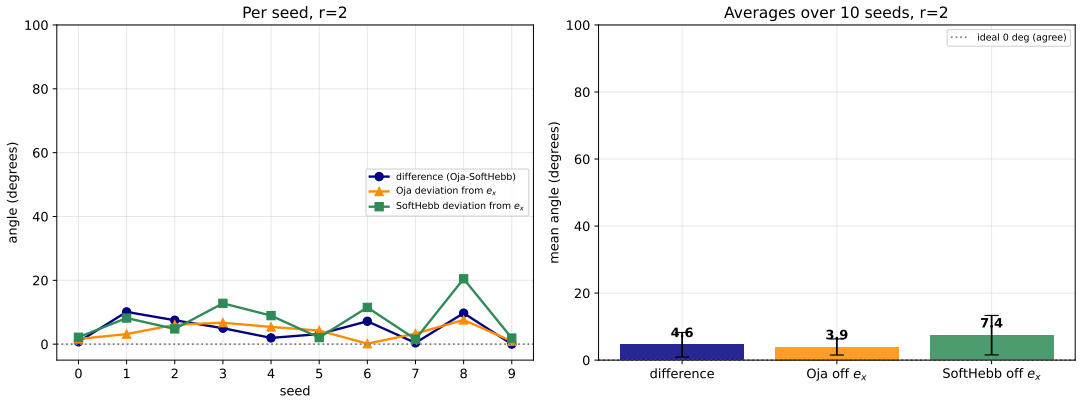


Figure 9: Regime C. The difference between Oja and SoftHebb convergence when the control ratio r is set to 2.