The Hebbian Descent Learning Rule

Laurenz Wiskott

Jan Melchior

INSTITUT FÜR NEUROINFORMATIK

RUHR UNIVERSITÄT BOCHUM

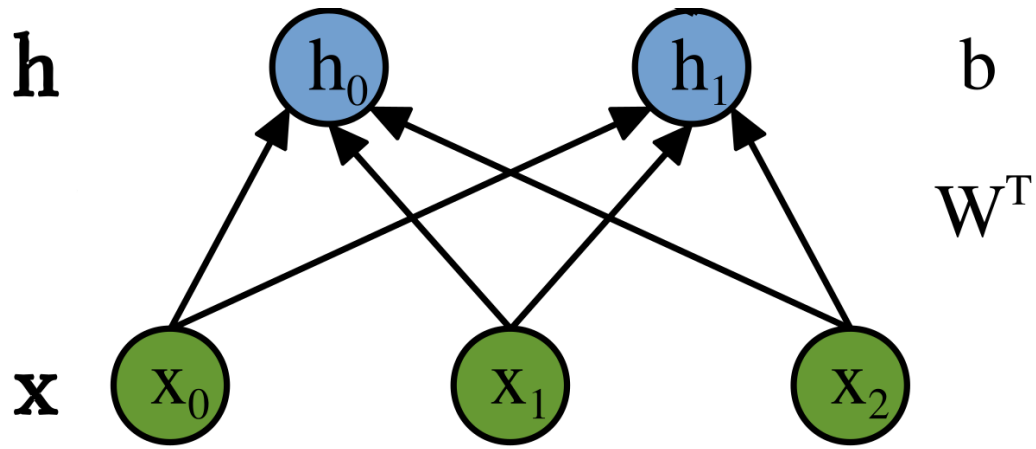
RUB

supported by:
Ruhr-University Bochum

Single layer network

Single layer network: $\boldsymbol{h} = \phi(\boldsymbol{a}) = \phi(\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{b})$

Input $\boldsymbol{x}$, offset $\boldsymbol{\mu}$, bias $\boldsymbol{b}$, weights $\boldsymbol{W}$, activation function $\boldsymbol{\phi}$, output $\boldsymbol{h}$.



(Melchior & Wiskott, 2019, arXiv:1905.10585)

Gradient descent

Single layer network: $\boldsymbol{h} = \phi(\boldsymbol{a}) = \phi(\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{b})$

Input $\boldsymbol{x}$, offset $\boldsymbol{\mu}$, bias $\boldsymbol{b}$, weights $\boldsymbol{W}$, activation function $\boldsymbol{\phi}$, output $\boldsymbol{h}$.

Loss function: $\mathcal{L}(\boldsymbol{t}, \boldsymbol{h})$ (with target values $\boldsymbol{t}$)

Gradient descent:

$$\begin{aligned} \Delta_{GD}\boldsymbol{W} &= -\eta(\boldsymbol{x} - \boldsymbol{\mu}) \left(\frac{\partial \mathcal{L}(\boldsymbol{t}, \boldsymbol{h})}{\partial \boldsymbol{h}} \odot \phi'(\boldsymbol{a}) \right)^T \\ \Delta_{GD}\boldsymbol{b} &= -\eta \left(\frac{\partial \mathcal{L}(\boldsymbol{t}, \boldsymbol{h})}{\partial \boldsymbol{h}} \odot \phi'(\boldsymbol{a}) \right) \end{aligned}$$

$\odot$ indicates element-wise multiplication.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian descent

Single layer network: $\boldsymbol{h} = \phi(\boldsymbol{a}) = \phi(\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{b})$

Input $\boldsymbol{x}$, offset $\boldsymbol{\mu}$, bias $\boldsymbol{b}$, weights $\boldsymbol{W}$, activation function ϕ, output $\boldsymbol{h}$.

Loss function: $\mathcal{L}(\boldsymbol{t}, \boldsymbol{h})$ (with target values $\boldsymbol{t}$)

Hebbian descent:

$$\Delta_{GD}\boldsymbol{W} = -\eta(\boldsymbol{x} - \boldsymbol{\mu}) \left(\frac{\partial \mathcal{L}(\boldsymbol{t}, \boldsymbol{h})}{\partial \boldsymbol{h}} \odot \phi'(\boldsymbol{a}) \right)^T$$

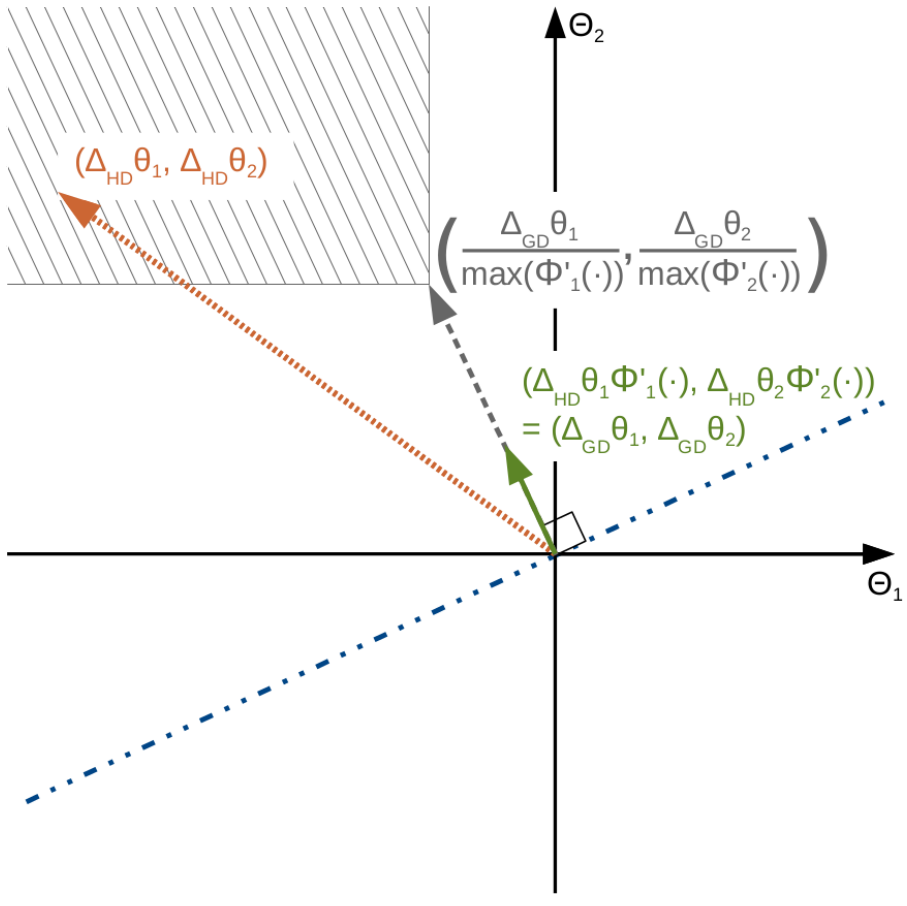
$$\Delta_{GD}\boldsymbol{b} = -\eta \left(\frac{\partial \mathcal{L}(\boldsymbol{t}, \boldsymbol{h})}{\partial \boldsymbol{h}} \odot \phi'(\boldsymbol{a}) \right)$$

$\odot$ indicates element-wise multiplication.

- Hebbian descent differs from gradient descent by dropping the derivative $\phi'(\boldsymbol{a})$.
- For non-negative $\phi'(\boldsymbol{a})$, Hebbian descent points within 90° of the gradient.
- Hebbian descent keeps going even if $\phi'(\boldsymbol{a}) = 0$.

(Melchior & Wiskott, 2019, arXiv:1905.10585;
cf. Kakade, Kalai, Kanade & Shamir, 2011, arXiv:1104.2018v1; Kanade, 2018, lecture notes)

Hebbian vs gradient descent

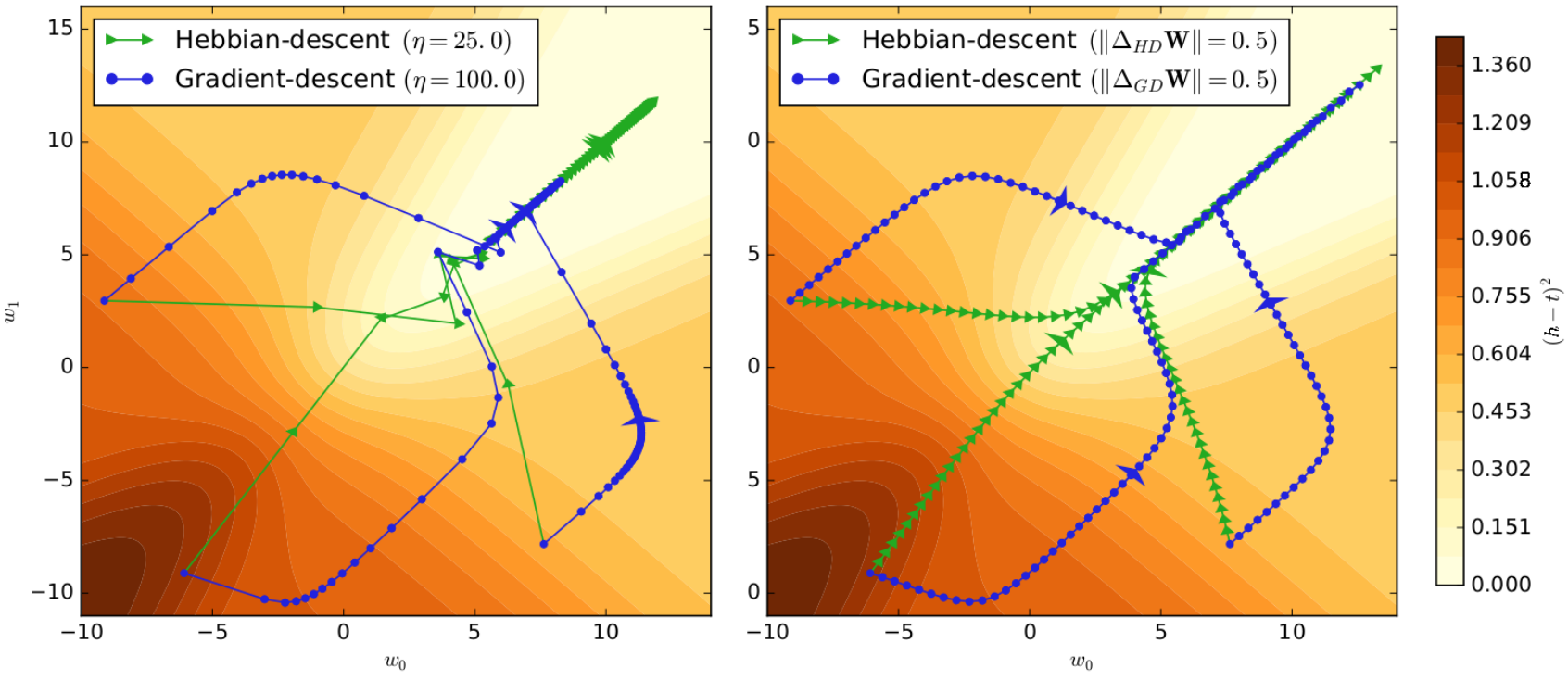


θ stands for W or b.

- Hebbian descent is confined to the quadrant at $\Delta_{GD}\theta_i / \max(\Phi'_i(\cdot))$.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian vs gradient descent



Two input units, one output unit, trained on $\{(1,0),(0,1),(1,1)\} \rightarrow \{(0),(0),(1)\}$ with L2 loss for 50 full-batch updates (wider arrowheads at 25 updates) with hand tuned learning rates avoiding oscillations. Natural step size (left), constant step size (right). Equivalent results with online learning.

- In this example, Hebbian descent finds a more direct path and converges faster.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian descent loss

Gradient descent: $$\Delta_{GD}\boldsymbol{W} = -\eta(\boldsymbol{x}-\boldsymbol{\mu})\left(\frac{\partial\mathcal{L}(\boldsymbol{t},\boldsymbol{h})}{\partial\boldsymbol{h}}\odot\boldsymbol{\phi}'(\boldsymbol{a})\right)^T$$

Hebbian descent as gradient descent (for Φ with $\Phi' > 0$)

$$\Delta_{HD}\boldsymbol{W} = -\eta(\boldsymbol{x}-\boldsymbol{\mu})\ \frac{\partial\mathcal{L}(\boldsymbol{t},\boldsymbol{h})}{\partial\boldsymbol{h}}^T = -\eta(\boldsymbol{x}-\boldsymbol{\mu})\left(\frac{\partial\mathcal{H}(\boldsymbol{t},\boldsymbol{h})}{\partial\boldsymbol{h}}\odot\boldsymbol{\phi}'(\boldsymbol{a})\right)^T$$

on different loss $$\mathcal{H}(\boldsymbol{t},\boldsymbol{h}) = \sum_j^M \int \frac{\partial\mathcal{L}(\boldsymbol{t},\boldsymbol{h})}{\partial\boldsymbol{h}} \oslash \boldsymbol{\phi}'(\boldsymbol{a})\,\mathrm{d}h_j$$

$\odot$ indicates element-wise multiplication; $\oslash$ indicates element-wise division.

- For instance, L2 loss with sigmoid leads to cross entropy loss:

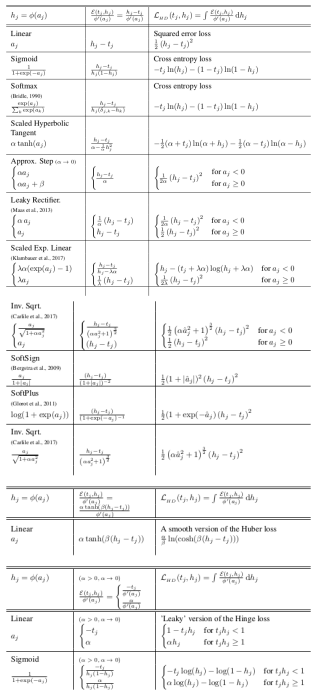
  $$\mathcal{L}(t_j,h_j) = \tfrac{1}{2}(h_j-t_j)^2 \quad\Rightarrow\quad \mathcal{H}(t_j,h_j) = -t_j\ln(h_j) - (1-t_j)\ln(1-h_j)$$

- Hebbian descent generalizes this to arbitrary losses and activation functions.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian descent losses

$h_j = \phi(a_j)$	$\frac{\mathcal{E}(t_j,h_j)}{\phi'(a_j)} = \frac{h_j - t_j}{\phi'(a_j)}$	$\mathcal{L}_{HD}(t_j, h_j) = \int \frac{\mathcal{E}(t_j,h_j)}{\phi'(a_j)}\,\mathrm{d}h_j$
Linear a_j	$h_j - t_j$	Squared error loss $\frac{1}{2}(h_j - t_j)^2$
Sigmoid $\frac{1}{1+\exp(-a_j)}$	$\frac{h_j - t_j}{h_j(1-h_j)}$	Cross entropy loss $-t_j \ln(h_j) - (1-t_j)\ln(1-h_j)$
Softmax (Bridle, 1990) $\frac{\exp(a_j)}{\sum_k \exp(a_k)}$	$\frac{h_j - t_j}{h_j(\delta_{j,k} - h_k)}$	Cross entropy loss $-t_j \ln(h_j) - (1-t_j)\ln(1-h_j)$
Scaled Hyperbolic Tangent $\alpha \tanh(a_j)$	$\frac{h_j - t_j}{\alpha - \frac{1}{\alpha}h_j^2}$	$-\frac{1}{2}(\alpha + t_j)\ln(\alpha + h_j) - \frac{1}{2}(\alpha - t_j)\ln(\alpha - h_j)$
Approx. Step $(\alpha \to 0)$ $\begin{cases} \alpha a_j \\ \alpha a_j + \beta \end{cases}$	$\begin{cases} \frac{h_j - t_j}{\alpha} \end{cases}$	$\begin{cases} \frac{1}{2\alpha}(h_j - t_j)^2 & \text{for } a_j < 0 \\ \frac{1}{2}(h_j - t_j)^2 & \text{for } a_j \geq 0 \end{cases}$
Leaky Rectifier. (Maas et al., 2013) $\begin{cases} \alpha\, a_j \\ a_j \end{cases}$	$\begin{cases} \frac{1}{\alpha}(h_j - t_j) \\ h_j - t_j \end{cases}$	$\begin{cases} \frac{1}{2\alpha}(h_j - t_j)^2 & \text{for } a_j < 0 \\ \frac{1}{2}(h_j - t_j)^2 & \text{for } a_j \geq 0 \end{cases}$
Scaled Exp. Linear (Klambauer et al., 2017) $\begin{cases} \lambda\alpha(\exp(a_j) - 1) \\ \lambda a_j \end{cases}$	$\begin{cases} \frac{h_j - t_j}{h_j - \lambda\alpha} \\ \frac{1}{\lambda}(h_j - t_j) \end{cases}$	$\begin{cases} h_j - (t_j + \lambda\alpha)\log(h_j + \lambda\alpha) & \text{for } a_j < 0 \\ \frac{1}{2\lambda}(h_j - t_j)^2 & \text{for } a_j \geq 0 \end{cases}$



(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian descent with L2 loss

Single layer network: $\boldsymbol{h} = \phi(\boldsymbol{a}) = \phi(\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{b})$

Input $\boldsymbol{x}$, offset $\boldsymbol{\mu}$, bias $\boldsymbol{b}$, weights $\boldsymbol{W}$, activation function ϕ, output $\boldsymbol{h}$.

Loss function: $\mathcal{L}(\boldsymbol{t}, \boldsymbol{h}) = \frac{1}{2}\|\boldsymbol{h} - \boldsymbol{t}\|^2$; $\frac{\partial \mathcal{L}(\boldsymbol{t}, \boldsymbol{h})}{\partial \boldsymbol{h}} = (\boldsymbol{h} - \boldsymbol{t})$

Hebbian descent:

$$\begin{aligned} \Delta_{HD}\boldsymbol{W} &= \underbrace{\eta(\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{t}^T}_{\text{sup. Hebb}} - \underbrace{\eta(\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{h}^T}_{\text{unsup. Hebb}} \\ \Delta_{HD}\boldsymbol{b} &= \eta\boldsymbol{t} - \eta\boldsymbol{h} \end{aligned}$$

- Difference between supervised and unsupervised Hebbian learning makes Hebbian descent stable, no learning if the desired output is achieved.
- Thus, no normalization is required.
- Thus, gradient descent benefits from multiple presentations of patterns (in contrast to Hebbian learning and the covariance rule).

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian descent includes Hebb's and covariance rule

Hebbian descent with L2 loss:

$$\Delta_{HD,sup.}\boldsymbol{W} = \underbrace{\eta(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{t}^T}_{\text{sup. Hebb}} \underbrace{-\eta(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{h}^T}_{\text{unsup. Hebb}} \quad (1)$$

For $\mu = 0$ we get Hebb's rule:

$$\begin{aligned} & \Delta_{HD,sup.}\boldsymbol{W} \\ = \; & \eta \boldsymbol{x}\boldsymbol{t}^T \\ = \; & \Delta_{Hebb}\boldsymbol{W} \end{aligned}$$

For $\mu = \langle \boldsymbol{x} \rangle$ we get the covariance rule:

$$\begin{aligned} & \langle \Delta_{HD,sup.}\boldsymbol{W} \rangle \\ = \; & \eta \langle (\boldsymbol{x} - \langle \boldsymbol{x} \rangle)\boldsymbol{t}^T \rangle \\ = \; & \eta \langle \boldsymbol{x}\boldsymbol{t}^T \rangle - \langle \boldsymbol{x} \rangle \langle \boldsymbol{t} \rangle^T \underbrace{-\langle \boldsymbol{x} \rangle \langle \boldsymbol{t} \rangle^T + \langle \boldsymbol{x} \rangle \langle \boldsymbol{t} \rangle^T}_{=0} \\ = \; & \eta \langle \boldsymbol{x}\boldsymbol{t}^T - \boldsymbol{x}\langle \boldsymbol{t} \rangle^T - \langle \boldsymbol{x} \rangle \boldsymbol{t}^T + \langle \boldsymbol{x} \rangle \langle \boldsymbol{t} \rangle^T \rangle \\ = \; & \eta \langle (\boldsymbol{x} - \langle \boldsymbol{x} \rangle)(\boldsymbol{t} - \langle \boldsymbol{t} \rangle)^T \rangle \\ = \; & \langle \Delta_{cov}\boldsymbol{W} \rangle \end{aligned}$$

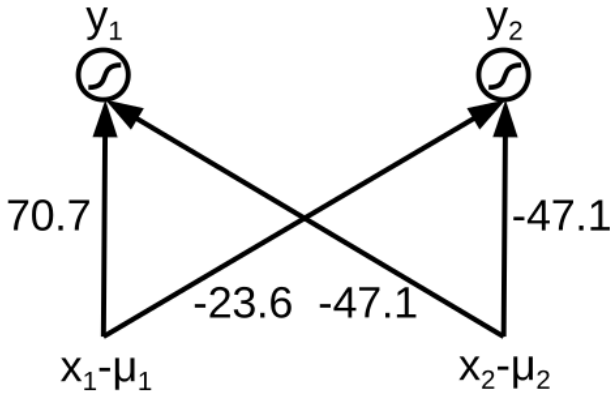
- The first term of Hebbian descent for the L2 loss includes Hebb's rule and covariance rule as special cases.
- The second term performs a normalization, lacking in Hebb's and covariance rule.

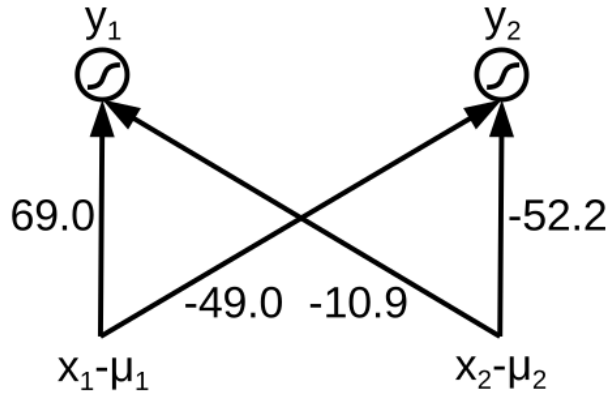
(Melchior & Wiskott, 2019, arXiv:1905.10585)

Limitation of Hebb's and covariance rule

target $\{(0,1),(1,0),(1,1),(1,1)\}$ $\{(0,1),(1,0),(1,1),(1,1)\}$

output $\{(0,0),(0,0),(1,1),(1,1)\}$ $\{(0,1),(1,0),(1,1),(1,1)\}$





input $\{(0,1),(1,1),(1,0),(1,0)\}$ $\{(0,1),(1,1),(1,0),(1,0)\}$

(a) Hebb-rule / Covariance rule (b) Hebbian-descent

Sigmoid units, one update per pattern, $\eta = 10$, $\boldsymbol{b} = \mathbf{0}$, $\boldsymbol{W}$ normalized to 100, ouput values rounded to two digits after the point.

- Hebb's and covariance rule do not deal well with ...
  - ... correlated patterns and
  - ... repeated patterns.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Data sets

- MNIST (LeCun et al., 1998) : 70,000 gray scale images of handwritten digits 0 to 9 with 28×28 pixels and pixel values in [0, 1].
- CONNECT (Larochelle et al., 2010) : 67,587 binary 126-dimensional game-state patterns from the game Connect-4.
- ADULT (Larochelle et al., 2010) : 32,561 binary 123-dimensional patterns of census data to predict whether a person's income exceeds 50,000 dollar per year.
- CIFAR (Krizhevsky, 2009) : 60,000 color images of various objects with 32×32 pixels, converted to gray scale and rescaled to [0, 1].
- RAND / RANDN : Random patterns with a size of 200 pixels. RAND has binary values with probabilities 0.5. RANDN has normally distributed values within [0,1] with mean 0.5 and standard deviation 0.1.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Experiments

Data sets

- RAND $\rightarrow$ ADULT
- RANDN $\rightarrow$ CONNECT
- ADULT $\rightarrow$ MNIST
- CONNECT $\rightarrow$ CIFAR
- MNIST $\rightarrow$ CONNECT
- CIFAR $\rightarrow$ CONNECT
- ...

with the L2 loss.

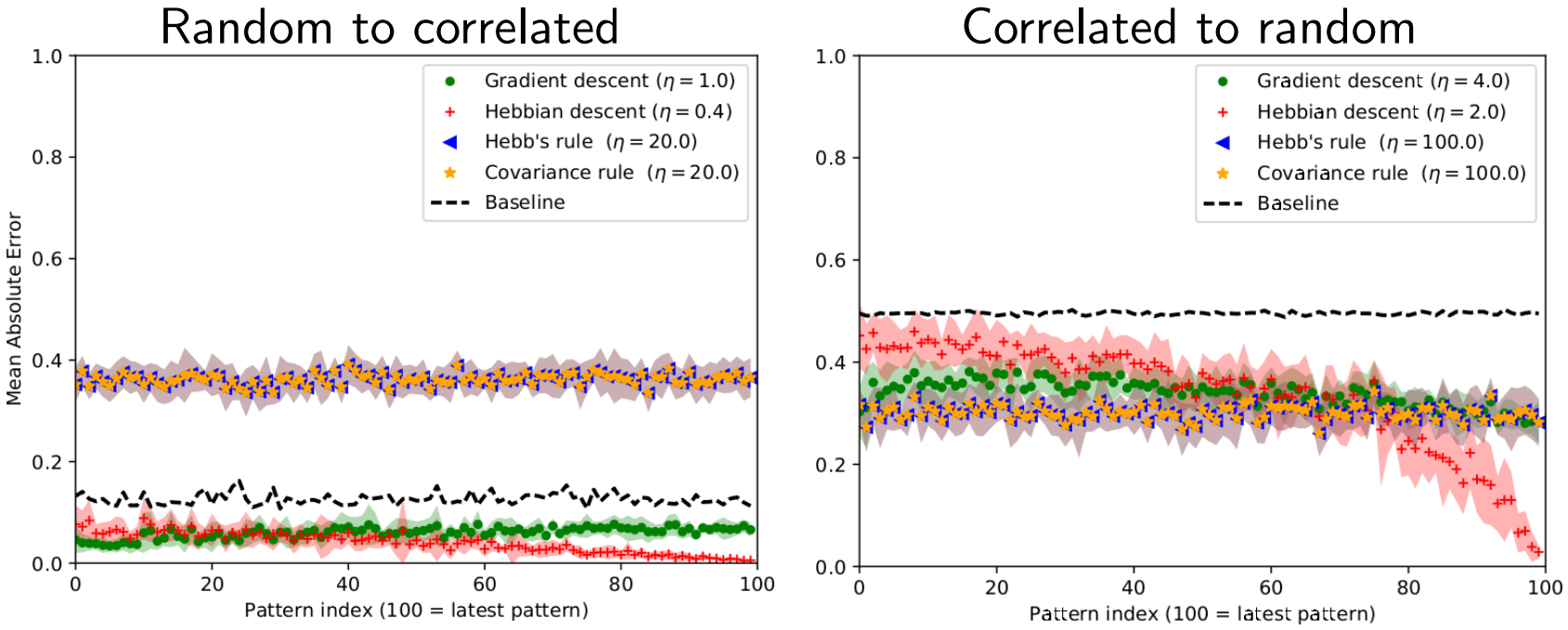
with

Activation functions

- Linear
- Exponential Linear
- Rectifier
- Sigmoid
- Step

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Associating random and correlated patterns



100 binary random patterns associated with 100 patterns from the ADULT dataset (binary census data) (left) and vice versa (right), one pair at a time, each pair trained once, with centering, parameters optimized for last 20 patterns.

- It is generally much easier to map random onto correlated patterns than the other way around.
- Hebb's rule and covariance rule fail due to correlations in the output.

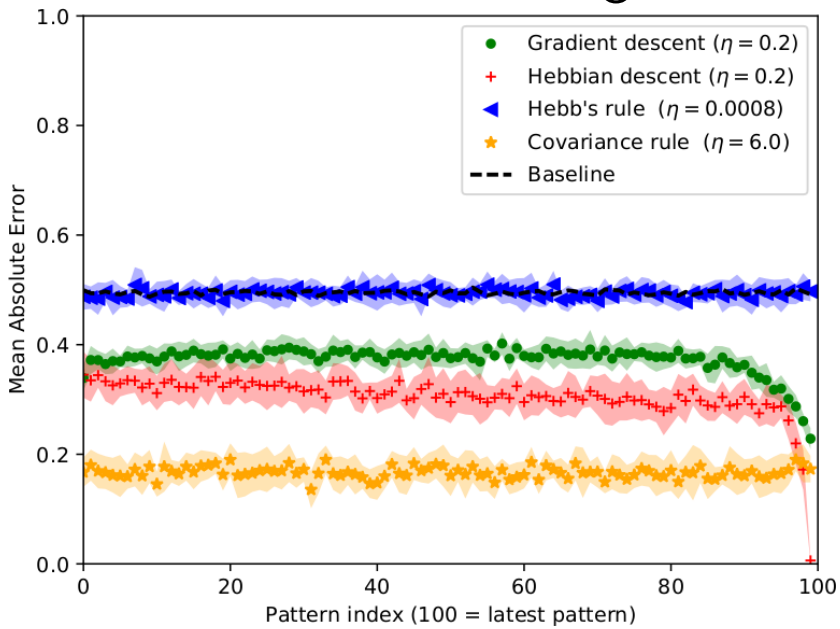
(Melchior & Wiskott, 2019, arXiv:1905.10585)

Centering

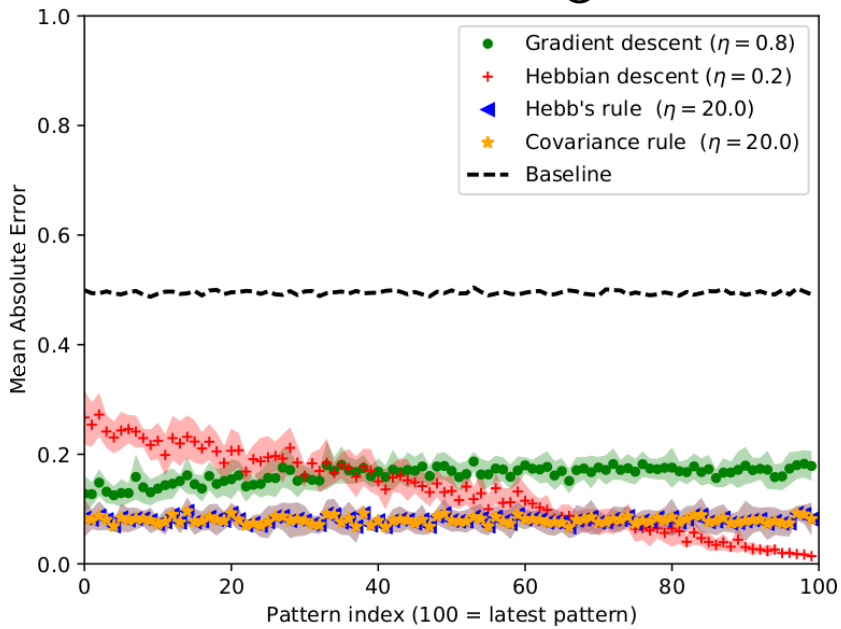
Centering sets $\mu = \langle \boldsymbol{x} \rangle$ in $\boldsymbol{h} = \phi(\boldsymbol{a}) = \phi(\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{b})$

(LeCun et al, 1998; Montavon & Müller, 2012; Melchior et al., 2016)

Without centering



With centering

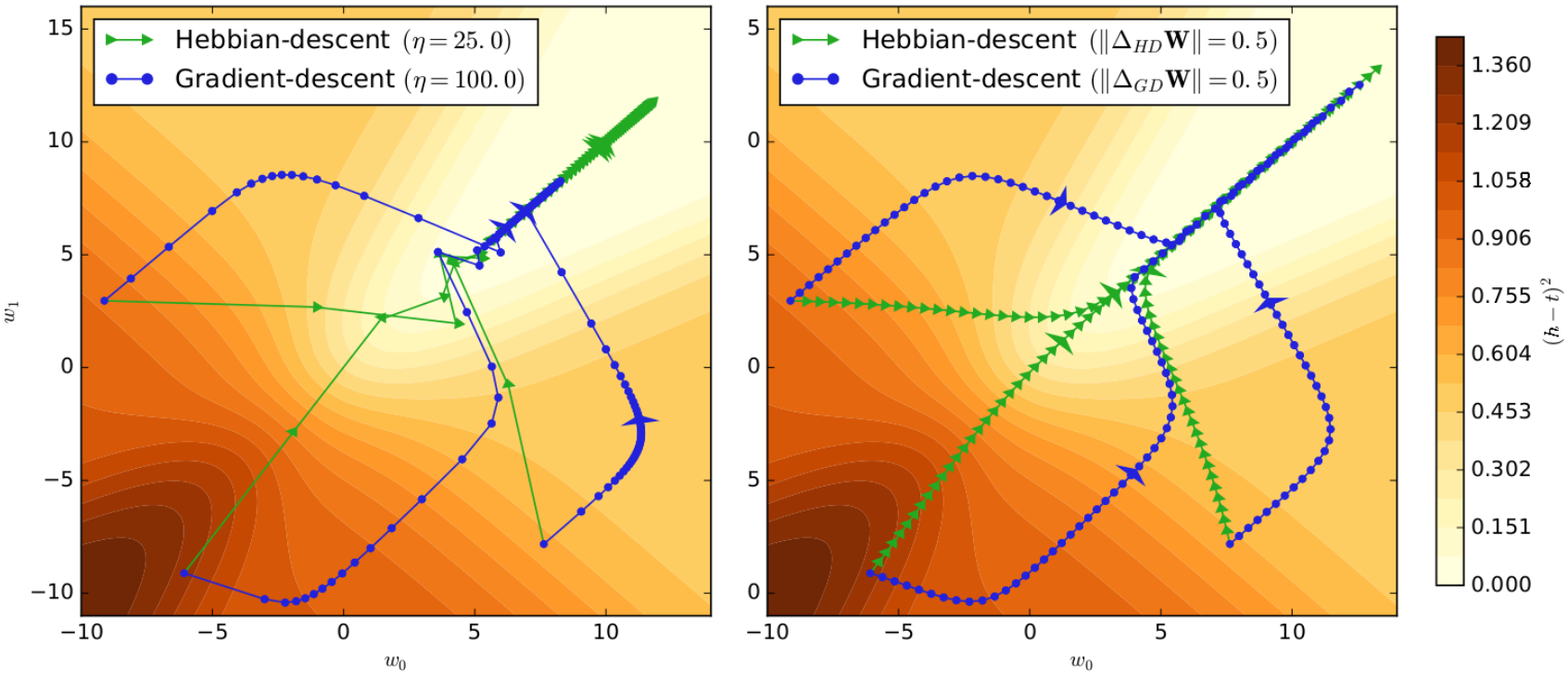


100 binary random patterns associated with 100 binary random patterns, one pair at a time, each pair trained once.

- Centering removes the mean. Bias and weights are more independent.
- With centering Hebb's rule and covariance rule become equivalent.
- Covariance rule does not do centering since the mean is subtracted only for learning.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian vs gradient descent

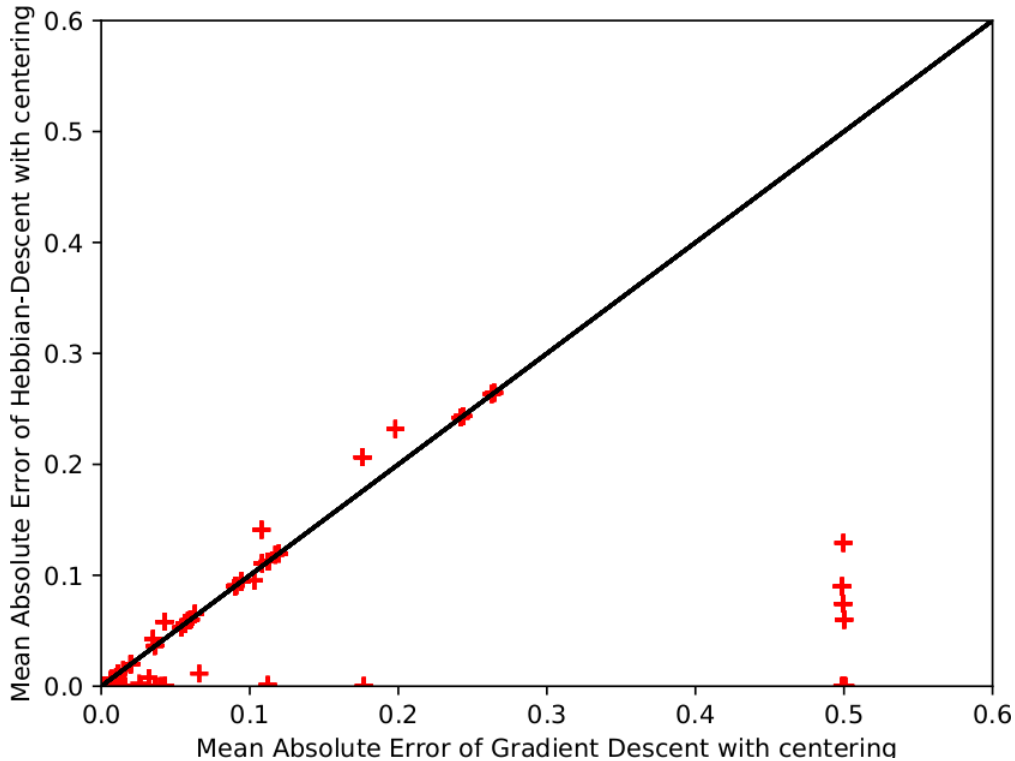


Two input units, one output unit, trained on $\{(1,0),(0,1),(1,1)\} \rightarrow \{(0),(0),(1)\}$ with L2 loss for 50 full-batch updates (wider arrowheads at 25 updates) with hand tuned learning rates avoiding oscillations. Natural step size (left), constant step size (right). Equivalent results with online learning.

- In this example, Hebbian descent finds a more direct path and converges faster.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian vs gradient descent (multi epoch)



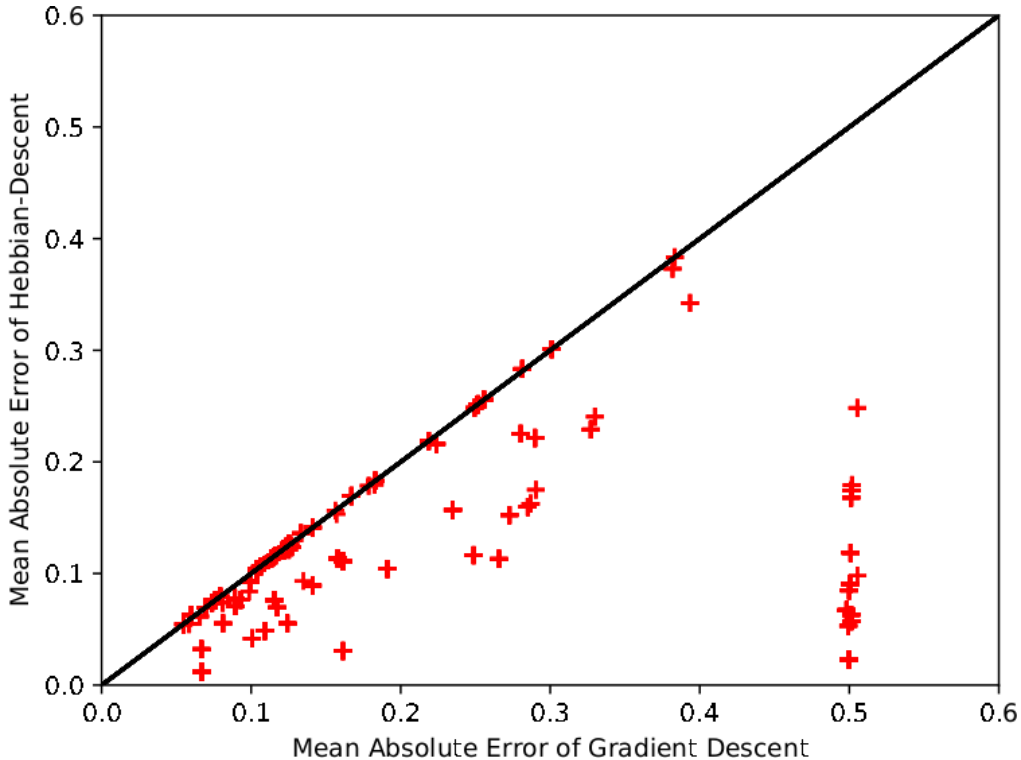
Multi epoch (with batch size one) hetero-association of 100 patterns of one data set onto 100 patterns of another data set with centering, parameters optimized for last 20 patterns on each experiment individually, 10 repetitions per experiment, mean absolute error on last 20 patterns.

For multi epoch learning:

- Hebbian descent and gradient descent perform equally well in most experiments.
- Hebbian descent has an advantage for binary target patterns.
- Gradient descent does not converge at all for the step function (values at 0.5).

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Hebbian vs gradient descent (online)



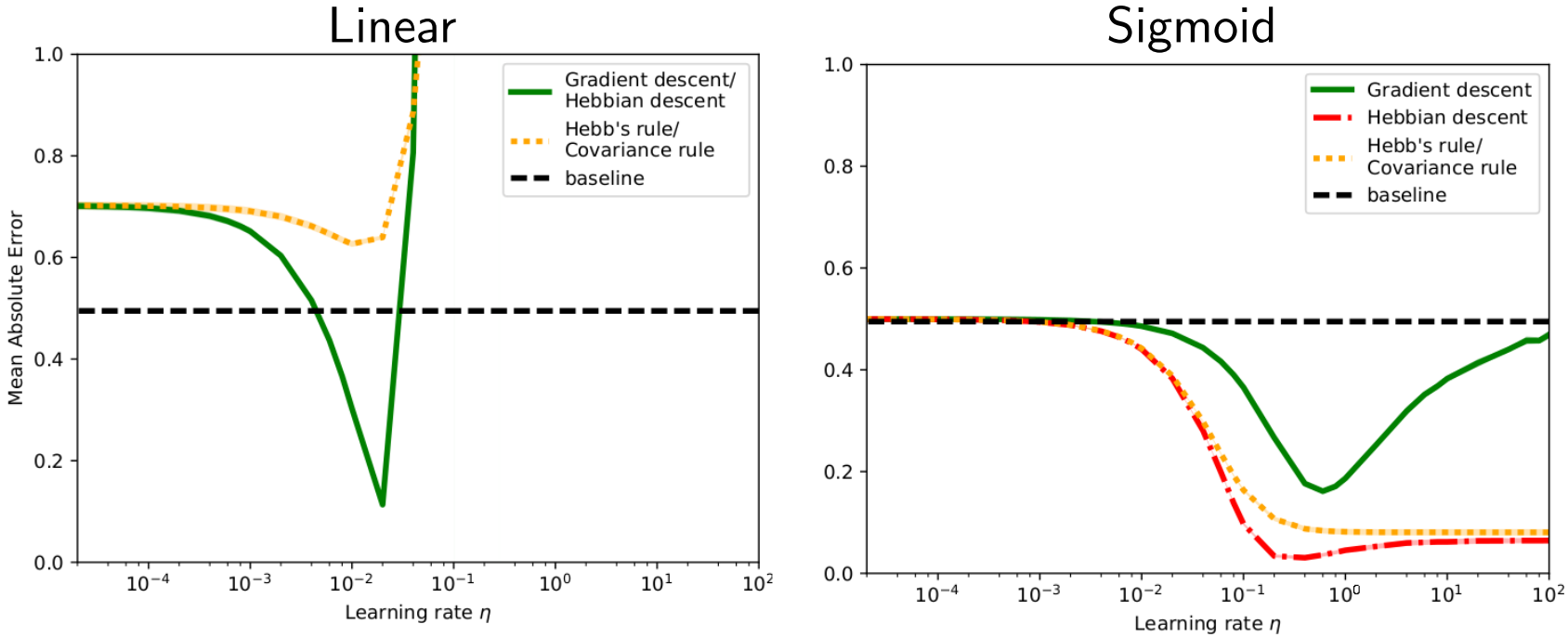
Online hetero-association of 100 patterns of one data set onto 100 patterns of another data set with centering, parameters optimized for last 20 patterns on each experiment individually, 10 repetitions per experiment, mean absolute error on last 20 patterns.

For online learning:

- Hebbian descent performes better than gradient descent in many experiments.
- The derivative of the activation function makes gradient descent inflexible.
- Gradient descent does not converge at all for the step function (values at 0.5).

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Saturating activation functions are important



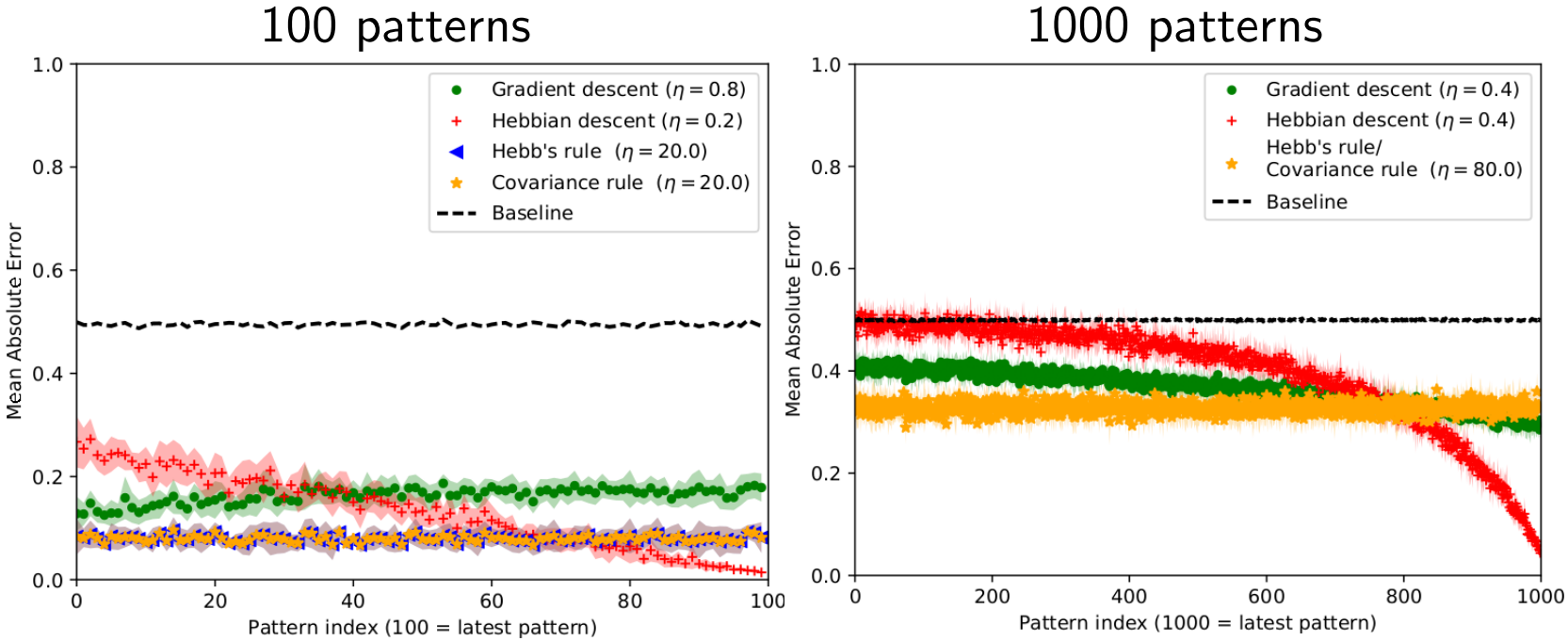
100 binary random patterns associated with 100 binary random patterns, one pair at a time, each pair trained once, with centering, parameters optimized for the last 20 patterns.

Nonlinear activation functions ...

- ... adapt the output range to the target values, thereby simplifying the learning problem.
- ... make the system robust against overshooting.
- ... allow to use a wide range of large learning rates.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Gradual forgetting



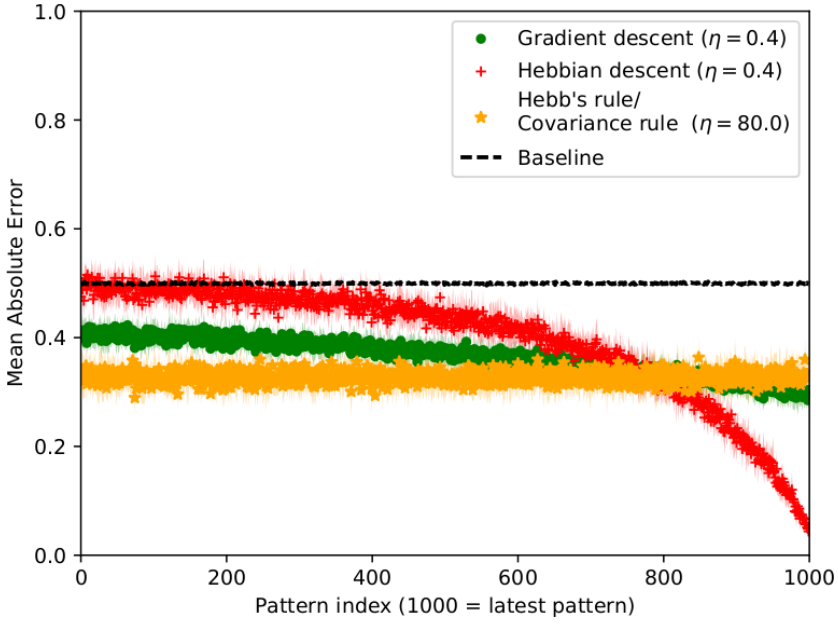
100 (left) or 1000 (right) binary random patterns associated with 100/1000 binary random patterns, one pair at a time, each pair trained once, with centering, parameters optimized for last 100 (left) or 20 (right) patterns (very similar results for last 1, 20 or all 100 patterns).

- If many more patterns are trained on than the network can store, ...
  - ... Hebbian descent gradually forgets old patterns and stores recent patterns well.
  - .... the other rules just store all patterns more or less poorly = catastrophic interference.

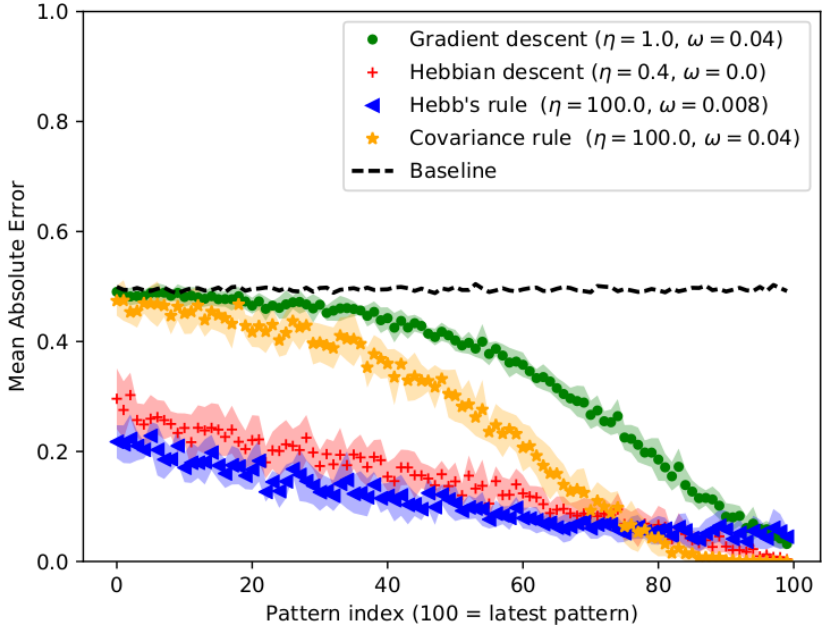
(Melchior & Wiskott, 2019, arXiv:1905.10585)

Weight decay

1000 patterns without weight decay



100 patterns with weight decay

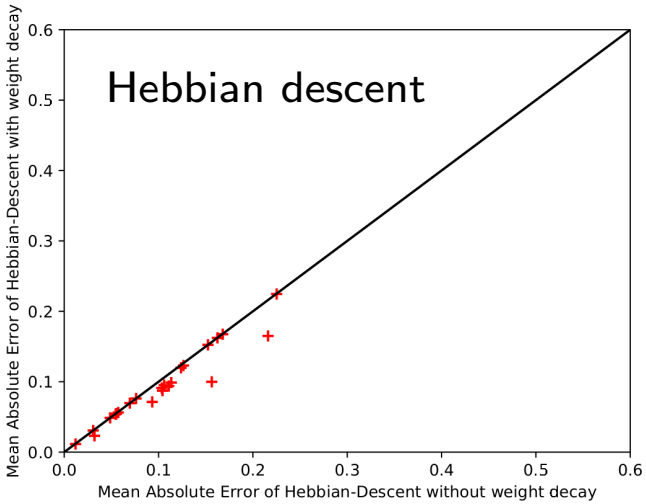


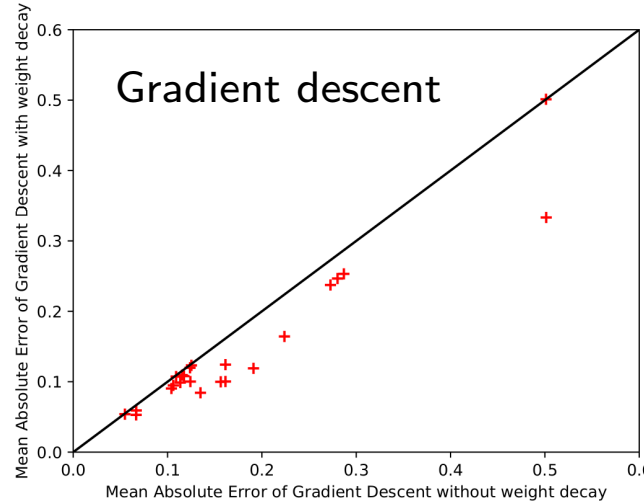
1000 (left) and 100 (right) binary random patterns associated with 1000/100 binary random patterns, one pair at a time, each pair trained once, with centering, parameters optimized for last 20 patterns.

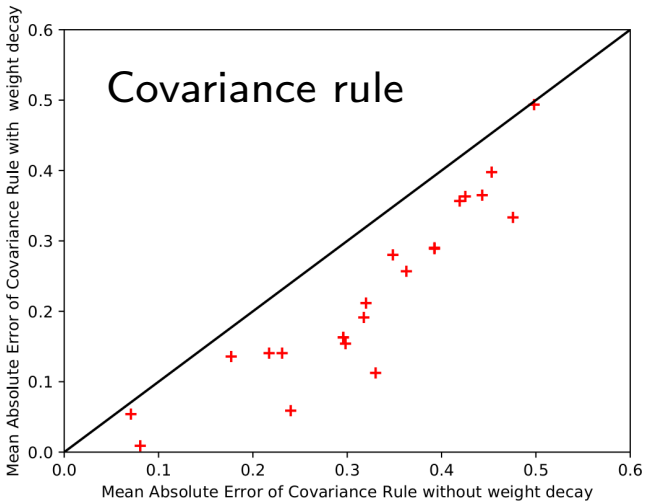
- All but Hebbian descent (optimal $\omega = 0.0$) profit from weight decay for the most recent patterns.

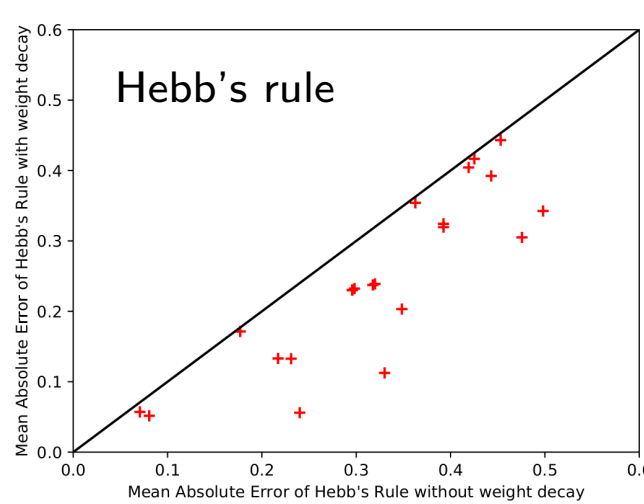
(Melchior & Wiskott, 2019, arXiv:1905.10585)

With weight decay vs without weight decay









Various experiments with different data sets and different activation functions, performance optimized for and measured on last 20 patterns.

- Hebbian descent profits least from weight decay but has overall best performance.
- Presumably Hebbian descent forgets more specifically than weight decay.
- Thus, weight decay is no alternative to Hebbian descent for gradual forgetting.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Summary

For hetero-association we have shown analytically:

1. **Hebbian-descent (HD) is gradient descent with dropped $\phi'(a)$.**
2. **HD equals gradient descent on a different loss function, if $\phi'(a) > 0$, thus**
3. **HD provably converges for $\phi'(a) > 0$.**
4. **HD for L2 loss with sigmoid leads to cross entropy loss. HD generalizes this idea.**

Furthermore, for L2 loss we have shown:

5. **HD is the difference of a supervised and an unsupervised Hebb/covariance rule.**
6. HD can also be viewed as contrastive learning where the hidden units are clamped to the same values in positive and negative phase.
7. HD optimizes a generalized linear model, if $\phi(a)$ is invertible and integrable, thus
8. HD loss can be seen as the general log-likelihood loss.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Summary

Our empirical results suggests that:

9. **Centering always helps.**
10. **HD outperforms Hebb's/covariance rule in general.**
11. **HD performs like gradient decent in multi-epoch learning.**
12. **HD performs clearly best in online/one-shot learning.**
13. **HD even works with the step function, for which $\phi'(a) = 0$ for $a \neq 0$.**
14. **HD shows an inherent curve of forgetting, better than weight decay.**

For auto-associative learning we have shown:

15. HD is related to contrastive learning in restricted Boltzmann machines.
16. HD corresponds to the encoder related part of gradient descent.
17. HD can be understood as non-linear Oja's/Sanger's rule.
18. HD does not have any objective function, but still it converges.
19. HD learns a hidden representation with equal average activity across units.
20. HD performs slightly worse than gradient descent
    but is computationally cheaper and biologically more plausible.

(Melchior & Wiskott, 2019, arXiv:1905.10585)

Thank you!