Machine Intelligence 2

1.2 Online-PCA / Hebbian Learning

Prof. Dr. Klaus Obermayer

Fachgebiet Neuronale Informationsverarbeitung (NI)

SS 2017

Recap: PCA

assuming centered data ($\underline{\mathbf{m}}_a = 0$), we have complex features $\underline{\mathbf{u}}_a = \underline{\mathbf{X}}\underline{\mathbf{e}}_a$

$$\sigma_\alpha^2 = \frac{1}{p}\underline{\mathbf{u}}_a^T\underline{\mathbf{u}}_a \qquad = \frac{1}{p}(\underline{\mathbf{e}}_a^T\underline{\mathbf{X}}^T)\cdot(\underline{\mathbf{X}}\underline{\mathbf{e}}_a) \qquad = \underline{\mathbf{e}}_a^T\underline{\mathbf{C}}\underline{\mathbf{e}}_a$$

Goal:

$$\underline{\mathbf{e}}_a^* = \underset{\underline{\mathbf{e}}_a}{\operatorname{argmax}}\left(\sigma_a^2\right) \qquad \text{with} \qquad \|\underline{\mathbf{e}}_a\| = 1$$

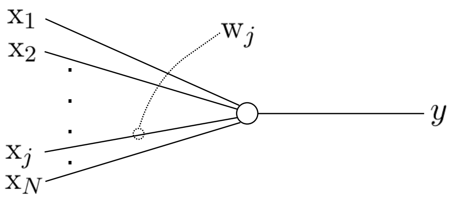
$$\underbrace{\underline{\mathbf{e}}_a^T\underline{\mathbf{C}}\underline{\mathbf{e}}_a}_{\text{objective}} - \lambda\underbrace{\left(\underline{\mathbf{e}}_a^T\underline{\mathbf{e}}_a - 1\right)}_{\text{constraints}} \overset{!}{=} \max$$

Constrained optimization $\rightsquigarrow$ Eigenvalue problem

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_a = \lambda\underline{\mathbf{e}}_a$$

$\Rightarrow$ **Principal Components**: normalized eigenvectors $\underline{\mathbf{e}}_\alpha$ of $\underline{\mathbf{C}}$

Linear connectionist neurons



$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

observations: $\underline{\mathbf{x}}^{(\alpha)}, \quad \alpha = 1, \dots, p, \quad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

Hebbian learning (Donald Hebb,1949): "fire together - wire together"

Hebbian learning

```
initialization of weights (e.g. to small numbers)
choose learning rate $\varepsilon$
begin loop
    choose an observation $\underline{\mathbf{x}}^{(\alpha)}$
    change weights according to:

        $$\Delta \underline{\mathbf{w}} = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \underline{\mathbf{x}}^{(\alpha)}$$

end
```

$\Rightarrow$ weights increase (decrease), if input and output are correlated (anticorrelated)

Proposition

Weight vector converges to the Principal Component with the largest eigenvalue.

Hebbian learning

assumption: centered data
small learning steps $\longrightarrow$ average over patterns:

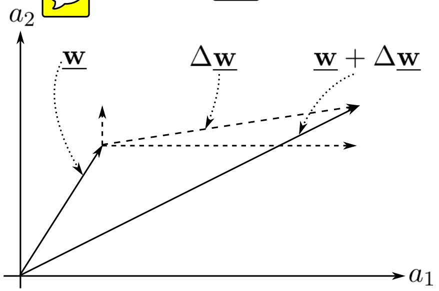
$$
\begin{aligned}
\Delta \mathrm{w}_j &\approx \frac{\varepsilon}{p} \sum_{\alpha=1}^{p} y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{x}_j^{(\alpha)} \\
&= \frac{\varepsilon}{p} \sum_{\alpha=1}^{p} \sum_{k=1}^{N} \mathrm{w}_k \mathrm{x}_k^{(\alpha)} \mathrm{x}_j^{(\alpha)} \\
&= \varepsilon \sum_{k=1}^{N} \mathrm{w}_k \left\{ \frac{1}{p} \sum_{\alpha=1}^{p} \mathrm{x}_k^{(\alpha)} \mathrm{x}_j^{(\alpha)} \right\} \\
&= \varepsilon \sum_{k=1}^{N} \mathrm{w}_k C_{kj}
\end{aligned}
$$

$$\Delta \underline{\mathbf{w}} \approx \varepsilon \underline{\mathbf{C}\mathbf{w}}$$

Hebbian learning

eigenvectors of $\underline{\mathbf{C}}$: $\underline{\mathbf{e}}_1, \underline{\mathbf{e}}_2, \ldots, \underline{\mathbf{e}}_N$ corresponding eigenvalues: $\lambda_1 > \lambda_2 > \ldots > \lambda_N$

$$\underline{\mathbf{w}} = a_1\underline{\mathbf{e}}_1 + a_2\underline{\mathbf{e}}_2 + \ldots + a_N\underline{\mathbf{e}}_N$$


a_2
$\underline{\mathbf{w}}$
$\Delta\underline{\mathbf{w}}$
$\underline{\mathbf{w}} + \Delta\underline{\mathbf{w}}$
a_1

$$\Delta\underline{\mathbf{w}} = \varepsilon\underline{\mathbf{C}\mathbf{w}}$$

$$\Delta a_j = \varepsilon\lambda_j a_j$$

(see blackboard)

consequence

- $t \to \infty \qquad \leadsto |\underline{\mathbf{w}}| \to \infty$
- $\underline{\mathbf{e}}_{\mathrm{w}} = \frac{\underline{\mathbf{w}}}{|\underline{\mathbf{w}}|}$ converges to $\underline{\mathbf{e}}_1$ (eigenvector with the largest eigenvalue)

Implicit normalization: Oja's rule

- adaptive tracking of the direction of largest variance: "on-line" PCA
- implicit normalization

Oja's rule

$$\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \left\{ \underbrace{\mathrm{x}_j^{(\alpha)}}_{\text{Hebbian learning}} - \underbrace{y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{w}_j}_{\text{decay term}} \right\}$$

decay = Zerfall

Proposition

Oja's rule converges to the unit vector which points into the direction of the largest variance.

Derivation of Oja's rule

- Let $y^{(\alpha)} = y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}(t))$.
- Normalization of $\left|\underline{\mathbf{w}}(t)\right| = 1 \; \forall t$ is achieved by the learning rule

$$\underline{\mathbf{w}}(t+1) = \frac{\overbrace{\underline{\mathbf{w}}(t) + \varepsilon y^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}^{\text{Hebbian learning}}}{\underbrace{\left|\underline{\mathbf{w}}(t) + \varepsilon y^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}\right|}_{\text{Euclidean weights normalization}}}$$

However

$\rightsquigarrow$ Multiplicative constraint requires computation of the norm in each step

Derivation of Oja's rule

- Small learning step ε:
  Taylor expansion around $\varepsilon = 0$ gives ($\rightarrow$ calculation: exercise sheet)

$$\underline{\mathbf{w}}(t+1) = \underline{\mathbf{w}}(t) + \varepsilon \left\{ y^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}(t) y^{(\alpha)} \left(\underline{\mathbf{w}}(t)^T \underline{\mathbf{x}}^{(\alpha)} \right) \right\} + \mathcal{O}(\varepsilon^2)$$

Oja's rule

$$\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \left\{ \underbrace{\mathrm{x}_j^{(\alpha)}}_{\text{Hebbian learning}} - \underbrace{y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{w}_j}_{\text{decay term}} \right\}$$

Oja's rule = Hebbian Learning with weight normalization

Convergence properties of Oja's rule

a) The learning rule analyzes the covariance matrix $\underline{\mathbf{C}}$ of the data

Small learning steps $\rightsquigarrow$ average over all patterns

$$\Delta\underline{\mathbf{w}} \approx \frac{1}{p}\sum_{\alpha=1}^{p} \overbrace{\varepsilon y^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)} - y^{(\alpha)}\underline{\mathbf{w}}\right)}^{\text{Oja's rule}} = \varepsilon \left(\underbrace{\underline{\mathbf{C}}\underline{\mathbf{w}}}_{\text{Hebbian rule}} - \underbrace{\overbrace{\left(\underline{\mathbf{w}}^T\underline{\mathbf{C}}\underline{\mathbf{w}}\right)}^{\geq 0}\underline{\mathbf{w}}}_{\text{decay term}} \right)$$

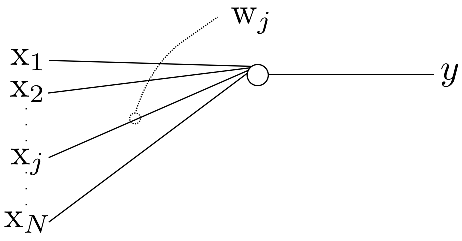
b) stationary states $\underline{\mathbf{w}}^*$ of Oja's rule $\hat{=}$ normalized eigenvector $\underline{\mathbf{e}}_j$ of $\underline{\mathbf{C}}$

Proof: See supplementary material

c) the stationary state $\underline{\mathbf{w}}^* = \underline{\mathbf{e}}_j$ is stable if and only if $\underline{\mathbf{e}}_j = \pm\underline{\mathbf{e}}_1$, i.e., if $\underline{\mathbf{e}}_j$ is the eigenvector with the largest eigenvalue λ_1

Proof: See supplementary material

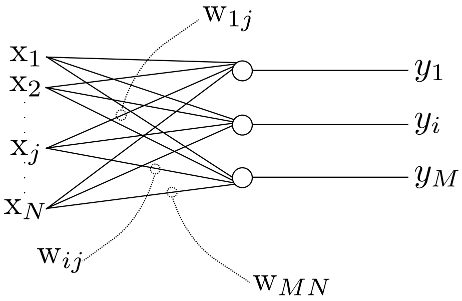
Hebbian PCA (generalized Hebbian algorithm)



One linear neuron: $y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$
observations: $\underline{\mathbf{x}}^{(\alpha)}, \quad \alpha = 1, \ldots, p, \quad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

Weight vector converges to the first principal component

Hebbian PCA (generalized Hebbian algorithm)



M linear neurons: $y_i = \underline{\mathbf{w}}_i^T \underline{\mathbf{x}} = \sum_{j=1}^{N} \mathrm{w}_{ij} \mathrm{x}_j$, $i = 1, \dots, M$
observations: $\underline{\mathbf{x}}^{(\alpha)}, \quad \alpha = 1, \dots, p, \quad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

The (feedforward) neural network extracts the M PCs with the largest eigenvalues

$\rightsquigarrow$ online-PCA for data with time-varying statistics

Hebbian PCA (generalized Hebbian algorithm)

Extended learning (Sanger's rule)

$$\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \underbrace{\mathrm{x}_j}_{\text{Hebbian rule}} - \underbrace{\sum_{k=1}^{i} \mathrm{w}_{kj} y_k}_{\sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k \text{ is added to Oja's rule}} \right\}$$

$\rightsquigarrow$ weights converge to the M eigenvectors with the largest eigenvalues

$$\begin{array}{ccc} \underline{\mathbf{w}}_1 & \rightarrow & \underline{\mathbf{e}}_1 \\ \underline{\mathbf{w}}_2 & \rightarrow & \underline{\mathbf{e}}_2 \\ & \vdots & \\ \underline{\mathbf{w}}_M & \rightarrow & \underline{\mathbf{e}}_M \end{array}$$

$\rightsquigarrow$ $y_i = \underline{\mathbf{e}}_i^T \underline{\mathbf{x}} =: a_i$ after learning

Learning: Oja's rule & Gram-Schmidt orthonormalization

Sanger's rule: $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \mathrm{x}_j - \sum_{k=1}^{i} \mathrm{w}_{kj} y_k \right\}$

- Define $\hat{\mathrm{x}}_j^{(i)} := \mathrm{x}_j - \sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k$
- Then $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \hat{\mathrm{x}}_j^{(i)} - y_j \mathrm{w}_{ij} \right\} \longrightarrow$ Oja's rule with modified input

Case $i = 1$:

$\hat{\mathrm{x}}_j^{(1)} = \mathrm{x}_j \quad \rightsquigarrow$ original form of Oja's rule

$\rightsquigarrow$ $\underline{\mathbf{w}}_1$ converges to eigenvector $\pm \underline{\mathbf{e}}_1$

Learning: Oja's rule & Gram-Schmidt orthonormalization

Sanger's rule: $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \mathrm{x}_j - \sum_{k=1}^{i} \mathrm{w}_{kj} y_k \right\}$

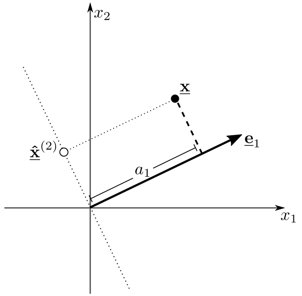
- Define $\hat{\mathrm{x}}_j^{(i)} := \mathrm{x}_j - \sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k$
- Then $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \hat{\mathrm{x}}_j^{(i)} - y_j \mathrm{w}_{ij} \right\} \longrightarrow$ Oja's rule with modified input

Case $i = 2$:

$\hat{\mathrm{x}}_j^{(2)} = \mathrm{x}_j - \mathrm{w}_{1j} y_1$
$\underline{\mathbf{w}}_1 = \underline{\mathbf{e}}_1 \rightarrow y_1 = \underline{\mathbf{x}}^T \underline{\mathbf{e}}_1 =: a_1$
& $\hat{\mathrm{x}}_j^{(2)} = \mathrm{x}_j - (\underline{\mathbf{e}}_1)_j \, a_1$

$\rightsquigarrow$ $\hat{\underline{\mathbf{x}}}^{(2)}$ is the projection of $\underline{\mathbf{x}}$ onto subspace orthogonal to $\underline{\mathbf{e}}_1$:

- $\rightsquigarrow$ $\underline{\mathbf{w}}_2$ converges to $\pm \underline{\mathbf{e}}_2$ by Oja's rule since $\underline{\mathbf{e}}_2$ is the direction of largest variance in that subspace



x_2
$\underline{\mathbf{x}}$
$\underline{\mathbf{e}}_1$
$\hat{\underline{\mathbf{x}}}^{(2)}$
a_1
x_1

Learning: Oja's rule & Gram-Schmidt orthonormalization

Sanger's rule: $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \mathrm{x}_j - \sum_{k=1}^{i} \mathrm{w}_{kj} y_k \right\}$

- Define $\hat{\mathrm{x}}_j^{(i)} := \mathrm{x}_j - \sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k$
- Then $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \hat{\mathrm{x}}_j^{(i)} - y_j \mathrm{w}_{ij} \right\} \longrightarrow$ Oja's rule with modified input

Case $i = 3$:

$\hat{\mathrm{x}}_j^{(3)} = \mathrm{x}_j - \mathrm{w}_{1j} y_1 - \mathrm{w}_{2j} y_2$

$\underline{\mathbf{w}}_1 = \underline{\mathbf{e}}_1$ & $\underline{\mathbf{w}}_2 = \underline{\mathbf{e}}_2 \rightarrow y_1 = a_1, y_2 = \underline{\mathbf{x}}^T \underline{\mathbf{e}}_2 =: a_2$

& $\hat{\mathrm{x}}_j^{(3)} = \mathrm{x}_j - a_1 \left(\underline{\mathbf{e}}_1\right)_j - a_2 \left(\underline{\mathbf{e}}_2\right)_j$

$\rightsquigarrow \hat{\underline{\mathbf{x}}}^{(3)}$ is the projection of $\underline{\mathbf{x}}$ onto subspace orthogonal to $\operatorname{span}\{\underline{\mathbf{e}}_1, \underline{\mathbf{e}}_2\}$

$\rightsquigarrow \underline{\mathbf{w}}_3$ converges to $\pm \underline{\mathbf{e}}_3$ by Oja's rule

Learning: Oja's rule & Gram-Schmidt orthonormalization

Sanger's rule: $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \mathrm{x}_j - \sum_{k=1}^{i} \mathrm{w}_{kj} y_k \right\}$

- Define $\hat{\mathrm{x}}_j^{(i)} := \mathrm{x}_j - \sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k$
- Then $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \hat{\mathrm{x}}_j^{(i)} - y_j \mathrm{w}_{ij} \right\} \longrightarrow$ Oja's rule with modified input

$\vdots$

Case $i = M$:

$\hat{\mathrm{x}}_j^{(M)} = \mathrm{x}_j - \sum_{k=1}^{M-1} \mathrm{w}_{kj} y_k$

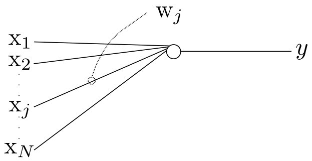
$\underline{\mathbf{w}}_k = \underline{\mathbf{e}}_k$ for $k = 1, \ldots, M-1 \to y_k = \underline{\mathbf{x}}^T \underline{\mathbf{e}}_k =: a_k$

& $\hat{\mathrm{x}}_j^{(M)} = \mathrm{x}_j - \sum_{k=1}^{M-1} a_k \left(\underline{\mathbf{e}}_k\right)_j$

$\rightsquigarrow$ $\underline{\hat{\mathbf{x}}}^{(M)}$ is the proj. of $\underline{\mathbf{x}}$ onto subspace orthogonal to $\mathrm{span}\left\{\underline{\mathbf{e}}_1, \ldots, \underline{\mathbf{e}}_{M-1}\right\}$

$\rightsquigarrow$ $\underline{\mathbf{w}}_M$ converges to $\pm \underline{\mathbf{e}}_M$ by Oja's rule

Summary of Hebbian learning



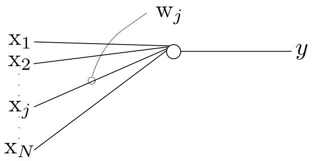
$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

I. Hebbian learning without constraint

$$\underbrace{\Delta \underline{\mathbf{w}} = \varepsilon y \underline{\mathbf{x}}}_{\text{Hebb's rule}} \rightsquigarrow \lim_{t \to \infty} \underline{\mathbf{w}} || \underline{\mathbf{e}}_1 \text{ (orthogonal)}$$

$\rightsquigarrow$ weights converge to direction of largest variance in the data
$\rightsquigarrow$ but: $|\underline{\mathbf{w}}| \to \infty$ for $t \to \infty$

Summary of Hebbian learning



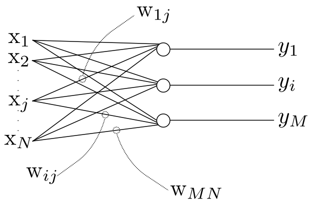
$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

II. Hebbian learning with normalization

$$\underbrace{\Delta \underline{\mathbf{w}} = \varepsilon y \left(\underline{\mathbf{x}} - y \underline{\mathbf{w}} \right)}_{\text{Oja's rule}} \rightsquigarrow \lim_{t \to \infty} \underline{\mathbf{w}} \in \{ +\underline{\mathbf{e}}_1, -\underline{\mathbf{e}}_1 \}$$

$\rightsquigarrow$ weights remain finite: $|\underline{\mathbf{w}}| = 1$

Summary of Hebbian learning



$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

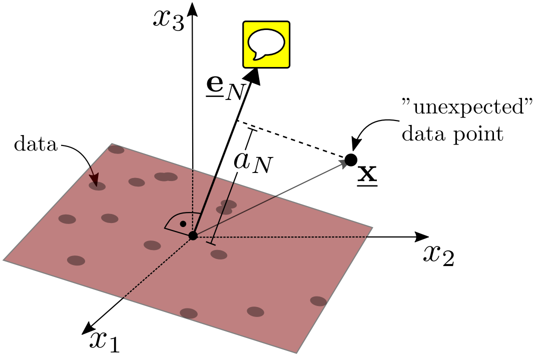
III. Hebbian PCA with M neurons and normalization

$$\underbrace{\Delta w_{ij} = \varepsilon y_i \left\{ x_j - \sum_{k=1}^{i} w_{kj} y_k \right\}}_{\text{Sanger's rule}} \leadsto \lim_{t \to \infty} \underline{\mathbf{w}}_i \in \left\{ +\underline{\mathbf{e}}_i, -\underline{\mathbf{e}}_i \right\}, i = 1, \ldots, M$$

$\leadsto$ combination of Oja's rule & Gram-Schmidt-orthonormalization

Novelty Filter

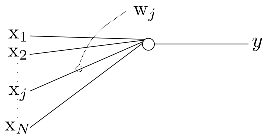
Reminder



$y = \underline{\mathbf{e}}_N^T \underline{\mathbf{x}} =: a_N$ after learning $\rightsquigarrow$ projection onto smallest PC

$\rightsquigarrow$ large output for unexpected data $\rightarrow$ Novelty Filter

Novelty Filter: On-line Learning



$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

Anti-Hebbian rule:

$$\Delta \mathrm{w}_j = \overbrace{-}^{\text{"Anti"-Hebbian}} \varepsilon y^{(\alpha)} \mathrm{x}_j^{(\alpha)}$$

	y small	y large
x_j small	$\rightarrow$	$\rightarrow$
x_j large	$\rightarrow$	$\downarrow$

Novelty Filter: On-line Learning

Conjecture:

$\underline{\mathbf{w}}$ converges to the direction of smallest eigenvector.

Proof:

Learning rule:

$$\Delta \mathrm{w}_j = -\varepsilon y^{(\alpha)} \mathrm{x}_j^{(\alpha)}$$

Assume small learning steps $\rightarrow$ average over all patterns

$$\Delta \mathrm{w}_j = -\frac{\varepsilon}{p} \sum_{\alpha=1}^{p} y^{(\alpha)} \mathrm{x}_j^{(\alpha)} = -\frac{\varepsilon}{p} \sum_{\alpha=1}^{p} \mathrm{x}_j^{(\alpha)} \sum_{k=1}^{N} \mathrm{x}_k^{(\alpha)} \mathrm{w}_k = -\varepsilon \sum_{k=1}^{N} C_{jk} \mathrm{w}_k$$

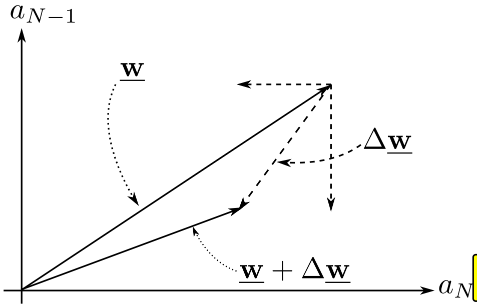
$$\Delta \underline{\mathbf{w}} = -\varepsilon \underline{\mathbf{C w}}$$

Novelty Filter: On-line Learning

Proof cont.:

$$\Delta\underline{\mathbf{w}} = -\varepsilon\underline{\mathbf{C}\mathbf{w}}$$

Transformation into eigenbasis of covariance matrix:



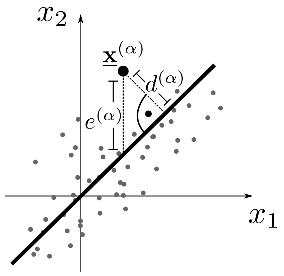
a_{N-1}
$\underline{\mathbf{w}}$
$\Delta\underline{\mathbf{w}}$
$\underline{\mathbf{w}} + \Delta\underline{\mathbf{w}}$
a_N

$$\underline{\mathbf{w}} = a_1\underline{\mathbf{e}}_1 + a_2\underline{\mathbf{e}}_2 + \cdots + a_N\underline{\mathbf{e}}_N$$

$$\Delta a_j = -\varepsilon\lambda_j a_j$$

- $\rightsquigarrow$ for $\lambda_j > 0 : a_j \to 0$, constraints required
- $\rightsquigarrow$ for $\lambda_j = 0 : a_j$ remains unchanged
- $\rightsquigarrow$ weights converge to the direction of smallest eigenvalue

Novelty Filter and linear regression



ordinary least squares:

$$\frac{1}{p}\sum_{\alpha=1}^{p}\left(e^{(\alpha)}\right)^2 \overset{!}{=} \text{min.}$$

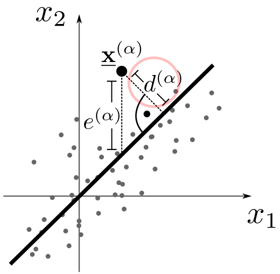
$\rightsquigarrow$ correct if data points are noisy along x_2-component only

$\rightsquigarrow$ wrong if data points are also noisy along x_1-component

Novelty Filter and linear regression

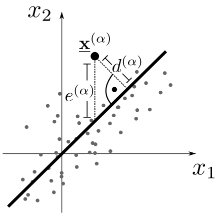
total least squares:

$$\frac{1}{p}\sum_{\alpha=1}^{p}\left(d^{(\alpha)}\right)^2 \overset{!}{=} \text{min.}$$



centered data $\rightarrow \underline{\mathbf{w}}^T \underline{\mathbf{x}} = 0$

Novelty Filter and linear regression



Cost function:

$$\mathbb{E}(\underline{\mathbf{w}}) = \frac{1}{p}\sum_{\alpha=1}^{p}\left(d^{(\alpha)}\right)^2 \overset{!}{=} \min_{\underline{\mathbf{w}}} \quad \text{s.t.}|\underline{\mathbf{w}}| = 1$$

$$\underline{\mathbf{e}}_{\mathrm{w}}^{T}\underline{\mathbf{C}\mathbf{e}}_{\mathrm{w}} \overset{!}{=} \min_{\underline{\mathbf{w}}} \quad \text{s.t.}|\underline{\mathbf{w}}| = 1$$

solution:
$\underline{\mathbf{w}}$ is the normalized eigenvector to the smallest eigenvalue of the covariance matrix.

Novelty Filter with normalization

$$\Delta \underline{\mathbf{w}} = -\varepsilon \frac{y^{(\alpha)} \left\{ \underline{\mathbf{x}}^{(\alpha)} - y^{(\alpha)} \underline{\mathbf{w}} \right\}}{\left| \underline{\mathbf{w}} - \varepsilon y^{(\alpha)} \left\{ \underline{\mathbf{x}}^{(\alpha)} - y^{(\alpha)} \underline{\mathbf{w}} \right\} \right|}$$

Anti-Hebbian version of Oja's rule:

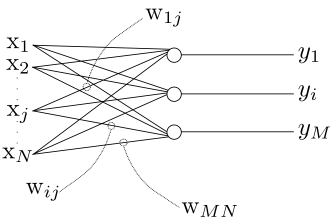
$$\Delta \mathrm{w}_j = -\varepsilon y^{(\alpha)} \Big\{ \overbrace{\mathrm{x}_j^{(\alpha)}}^{\text{Anti-Hebbian learning}} \underbrace{- y^{(\alpha)} \mathrm{w}_j \Big\} + \varepsilon \Big\{ 1 - \overbrace{\sum_{k=1}^{N} \mathrm{w}_k^2}^{=|\underline{\mathbf{w}}|^2} \Big\} \mathrm{w}_j}_{\text{normalization}}$$

$\rightsquigarrow$ $\underline{\mathbf{w}}$ converges to $\pm \underline{\mathbf{e}}_N$

Feedforward network as Novelty Filter

Extension of the learning rule to N neurons:

$$\Delta \mathrm{w}_{ij} = -\varepsilon y_i^{(\alpha)} \left\{ \mathrm{x}_j^{(\alpha)} - \overbrace{\sum_{k=1}^{i} \mathrm{w}_{kj} y_k^{(\alpha)}}^{-\sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k^{(\alpha)} \text{ is added}} \right\} + \varepsilon \left\{ 1 - \sum_{k=1}^{N} \mathrm{w}_{ik}^2 \right\} \mathrm{w}_{ij}$$



$\rightsquigarrow$ result:

$\underline{\mathbf{w}}_1 \to \underline{\mathbf{e}}_N$ (PC with smallest eigenvalue)

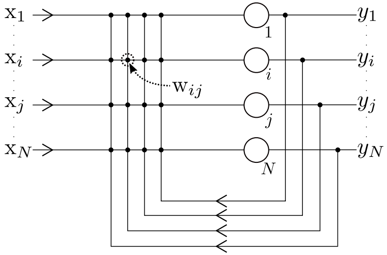
$\underline{\mathbf{w}}_2 \to \underline{\mathbf{e}}_{N-1}$ $\vdots$

$\vdots$ $\vdots$

$\underline{\mathbf{w}}_M \to \underline{\mathbf{e}}_{N-M+1}$ (PC with largest eigenvalue, if $N = M$)

Sequential calculation of eigenvectors (cf. Sanger's rule).

Recurrent network as Novelty Filter



$$y_i^{(\alpha)}(t+1) = \sum_{j=1}^{N} w_{ij} y_j^{(\alpha)}(t) + x_i^{(\alpha)}$$

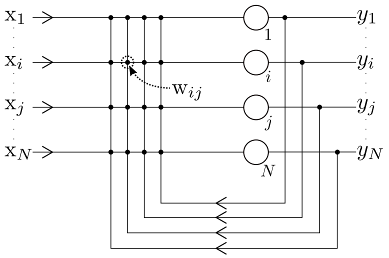
$$\underline{\mathbf{y}}^{(\alpha)}(t+1) = \underline{\mathbf{W}}\,\underline{\mathbf{y}}^{(\alpha)}(t) + \underline{\mathbf{x}}^{(\alpha)}$$

Stationary state:

$$\underline{\tilde{\mathbf{y}}}^{(\alpha)}(t+1) = \underline{\tilde{\mathbf{y}}}^{(\alpha)}(t) =: \underline{\tilde{\mathbf{y}}}^{(\alpha)}$$

$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \left(\underline{\mathbf{I}} - \underline{\mathbf{W}}\right)^{-1} \underline{\mathbf{x}}^{(\alpha)}$$

Recurrent network as Novelty Filter



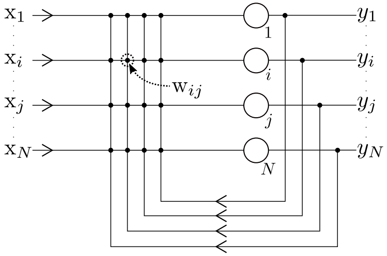
$$y_i^{(\alpha)}(t+1) = \sum_{j=1}^{N} w_{ij} y_j^{(\alpha)}(t) + x_i^{(\alpha)}$$

$$\underline{\mathbf{y}}^{(\alpha)}(t+1) = \underline{\mathbf{W}}\underline{\mathbf{y}}^{(\alpha)}(t) + \underline{\mathbf{x}}^{(\alpha)}$$

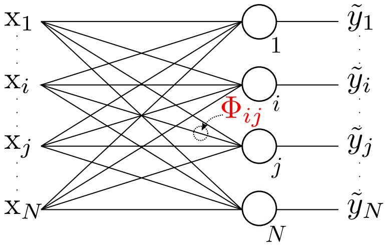
Learning rule:

$$\Delta w_{ij} = -\varepsilon \tilde{y}_i \tilde{y}_j$$

Recurrent network as Novelty Filter

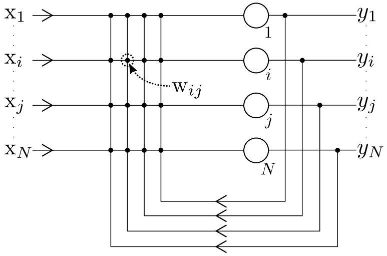


$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = (\underline{\mathbf{I}} - \underline{\mathbf{W}})^{-1}\, \underline{\mathbf{x}}^{(\alpha)}$$

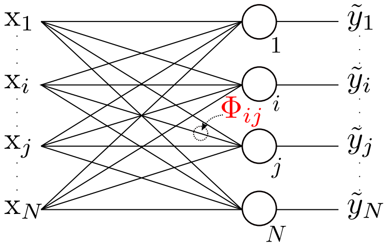


$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \underline{\mathbf{\Phi x}}^{(\alpha)}$$

Recurrent network as Novelty Filter



$$\Delta \underline{\mathbf{w}} = -\varepsilon \underline{\tilde{\mathbf{y}}}^{(\alpha)} \left(\underline{\tilde{\mathbf{y}}}^{(\alpha)} \right)^T$$



$$\Delta \underline{\mathbf{\Phi}} = -\varepsilon \underline{\mathbf{\Phi}}^2 \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} \right)^T \underline{\mathbf{\Phi}}^2$$

Recurrent network as Novelty Filter

```
initialization of $\underline{\mathbf{\Phi}}$ with identity matrix, $\underline{\mathbf{\Phi}} = \underline{\mathbf{I}}$
repeat
    choose an observation $\underline{\mathbf{x}}^{(\alpha)}$
    change weight matrix according to:
        $\Delta\underline{\mathbf{\Phi}} = -\varepsilon\underline{\mathbf{\Phi}}^2\underline{\mathbf{x}}^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{\Phi}}^2$
until convergence
```

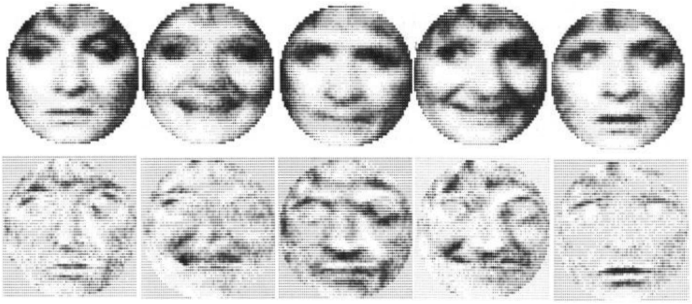
$\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ converges to a matrix, which projects onto the subspace orthogonal to the training data

example:

training data $\left\{\underline{\mathbf{x}}^{(1)}, \dots, \underline{\mathbf{x}}^{(p)}\right\} \subseteq \operatorname{span}\left\{\underline{\mathbf{e}}_1, \dots, \underline{\mathbf{e}}_{N-1}\right\}$

$$\underline{\mathbf{\Phi}} \xrightarrow[t\to\infty]{} \underline{\mathbf{I}} - \sum_{k=1}^{N-1}\underline{\mathbf{e}}_k\underline{\mathbf{e}}_k^T \rightsquigarrow \underline{\mathbf{\Phi}}\underline{\mathbf{e}}_j = \underline{\mathbf{e}}_N\delta_{jN} \rightsquigarrow \underline{\mathbf{\Phi}}\underline{\mathbf{x}} = \underbrace{\left(\underline{\mathbf{e}}_N^T\underline{\mathbf{x}}\right)}_{=:a_N}\underline{\mathbf{e}}_N$$

Recurrent network as Novelty Filter

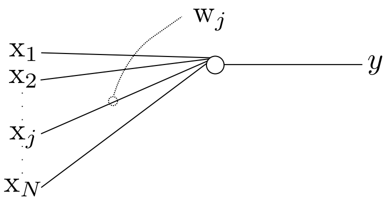


(taken from Kohonen 1989)

- training data: "neutral" facial expressions (not shown here)
  - $\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ projects into space orthogonal to this data
- top row: faces $\underline{\mathbf{x}}^{(\beta)}$ with different expressions
- bottom row: projection $\underline{\mathbf{\Phi}\mathbf{x}}^{(\beta)}$

Novelty Filter: Summary

One neuron:



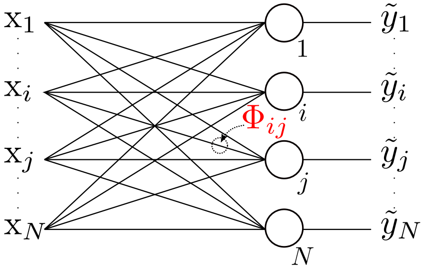
$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

Anti-Hebbian learning rule

$$\Delta w_j = -\varepsilon y^{(\alpha)} \Big\{ \overbrace{x_j^{(\alpha)}}^{\text{Anti-Hebbian learning}} \underbrace{- y^{(\alpha)} w_j \Big\} + \varepsilon \Big\{ 1 - \overbrace{\sum_{k=1}^{N} w_k^2}^{=|\underline{\mathbf{w}}|^2} \Big\} w_j}_{\text{normalization}}$$

Novelty Filter: Summary

N **neurons:**



$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \underline{\mathbf{\Phi}\mathbf{x}}^{(\alpha)}$$

Learning rule:

$$\Delta\underline{\mathbf{\Phi}} = -\varepsilon\underline{\mathbf{\Phi}}^2\underline{\mathbf{x}}^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{\Phi}}^2$$

$\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ converges to projection matrix onto subspace orthogonal to training data