Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

Neural Networks

Lecture 3

Linear auto-associative memories

Igor Farkaš

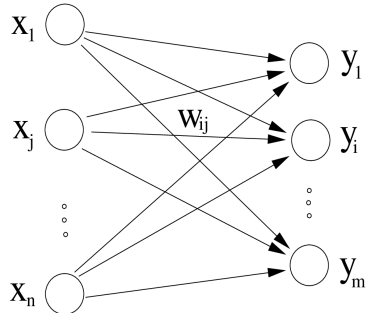
2019

Linear NN models

Input vector: $\boldsymbol{x} = [x_1, x_{2,\ldots}, x_n]^T$

Output vector: $\boldsymbol{y} = [y_1, y_{2,..}, y_m]^T$

Weight matrix: $\mathbf{W}$ ~ type $[m \times n]$



Linear transformation $\varphi : \Re^n \rightarrow \Re^m$, $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$

☹ ignores saturation property of neurons
☺ allows to find analytic solutions using linear algebra.

(Kohonen, 1970; Anderson, 1972; Cooper, 1973)

- Adding layers in a linear NN does not appear reasonable (since no complexity is added).
- But: It allows nonlinear learning dynamics in linear deep nets (Saxe, 2015).

Linear NN – analytic solution

Let's consider the train set: $A_{train} = \{(\boldsymbol{x}^{(p)}, \boldsymbol{y}^{(p)}), p = 1,2,...,N\}$.

We look for a matrix $\mathbf{W}$ that satisfies $\boldsymbol{y}^{(p)} = \mathbf{W}\boldsymbol{x}^{(p)}$, for each p.

In matrix notation: $\mathbf{Y} = \mathbf{W}\,\mathbf{X}$

$$\underset{(m\times N)}{[\boldsymbol{y}^{(1)}\ \boldsymbol{y}^{(2)} \ldots \boldsymbol{y}^{(N)}]} = \underset{(m\times n)}{\mathbf{W}} \times \underset{(n\times N)}{[\boldsymbol{x}^{(1)}\ \boldsymbol{x}^{(2)} \ldots \boldsymbol{x}^{(N)}]}$$

If $\mathbf{X}$ was regular (i.e., square matrix with $N = n$, linearly independent rows) then $\mathbf{X}^{-1}$ would exist and $\mathbf{W} = \mathbf{Y}\mathbf{X}^{-1}$.

However, in general we cannot assume $N = n$, nor linear independence of input vectors ($\Rightarrow \mathbf{X}^{-1}$ does not exist). Only generalized solutions exists:

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^{+}$$

$\mathbf{X}^{+}$ is called (Moore-Penrose) pseudoinverse matrix of $\mathbf{X}$. Theorem: $\forall\ \mathbf{X}$, $\exists\ \mathbf{X}^{+}$ with properties: 1) $\mathbf{X}\mathbf{X}^{+}\mathbf{X} = \mathbf{X}$, 2) $\mathbf{X}^{+}\mathbf{X}\mathbf{X}^{+} = \mathbf{X}^{+}$, 3) symmetric $\mathbf{X}\mathbf{X}^{+}$ and $\mathbf{X}^{+}\mathbf{X}$.

a) $\mathbf{X}^{+} = \mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}$, if $n < N$ and $rank(\mathbf{X}) = n$.

b) $\mathbf{X}^{+} = (\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}$, if $n > N$ and $rank(\mathbf{X}) = N$.

Auto-associator case

Let's consider $n > N$ (i.e. few examples of large dimension) and the autoassociative case: $\boldsymbol{y}^{(p)} = \boldsymbol{x}^{(p)}$, $m = n$

Model is supposed to remember N prototypes $[\boldsymbol{x}^{(1)}\ \boldsymbol{x}^{(2)} \ldots \boldsymbol{x}^{(N)}]$.

Goal: train on prototypes and then submit a corrupted version of a prototype. Model should be able to reconstruct it.

Since $\mathbf{Y} = \mathbf{X}$, then $\mathbf{W} = \mathbf{X}\mathbf{X}^{+}$. How to interpret $\mathbf{W}$?

In a special case, which is too restrictive ($N = n$, linearly independent inputs), we would have a trivial solution $\mathbf{W} = \mathbf{I}$ (identity).

How about a general case?

Basics of linear vector spaces

Let's have a linear space $\Re^n$.

Linear manifold $\mathscr{L} = \{\boldsymbol{x} \in \Re^n \,|\, \boldsymbol{x} = a_1\boldsymbol{x}^{(1)} + a_2\boldsymbol{x}^{(2)} + \ldots + a_N\boldsymbol{x}^{(N)},\ a_p \neq 0\}$

$\mathscr{L} \subset \Re^n$

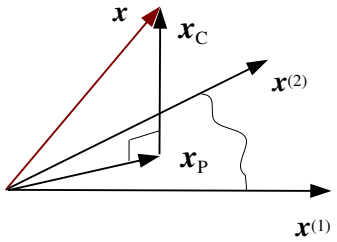
Orthogonal complement $\mathscr{L}^\perp = \{\boldsymbol{x} \in \Re^n \,|\, \boldsymbol{x} \perp \mathscr{L}\}$

Hence, $\mathscr{L} \oplus \mathscr{L}^\perp = \Re^n$

Each vector $\boldsymbol{x} \in \Re^n$ can be uniquely decomposed:

$$\boldsymbol{x} = \boldsymbol{x}_P + \boldsymbol{x}_C$$

where $\boldsymbol{x}_P \in \mathscr{L}$ and $\boldsymbol{x}_C \in \mathscr{L}^\perp$.



What does an autoassociative NN do?

Training set $A_{train} = \{\boldsymbol{x}^{(p)},\ p = 1,2,...,N\}$ forms the linear manifold $\mathscr{L}$.

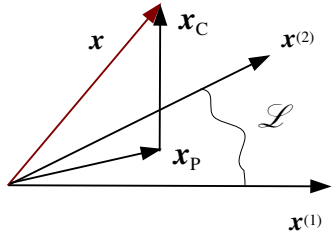
NN considers every departure $\boldsymbol{x}$ from $\mathscr{L}$ as added noise that needs to be filtered out by projecting $\boldsymbol{x}$ to $\mathscr{L}$:

We need to show that output $\mathbf{W}\boldsymbol{x} = \mathbf{X}\mathbf{X}^+\boldsymbol{x} = \boldsymbol{x}_P$ (filtered version of $\boldsymbol{x}$), i.e. that operator $\mathbf{W} = \mathbf{X}\mathbf{X}^+$ makes an orthogonal projection to $\mathscr{L}$.

Alternatively, the NN model with operator $\mathbf{W} = \mathbf{I} - \mathbf{X}\mathbf{X}^+$ is called novelty detector, where $\mathbf{W}\boldsymbol{x} = \boldsymbol{x}_C \in \mathscr{L}^\perp$.

Now assume: you learned N patterns, and want to add $(N+1)$-st pattern.

How to change $\mathbf{W}$ efficiently?



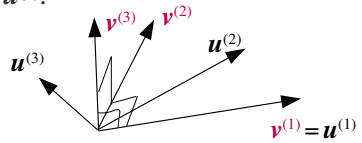
Gram-Schmidt orthogonalization process

Let's have a base $\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(k)} \in \mathscr{L}$, for which we want to create an orthogonal base $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)} \in \mathscr{L}$.

Procedure:

Set $\boldsymbol{v}^{(1)} = \boldsymbol{u}^{(1)}$. In space with base $\boldsymbol{v}^{(1)}$, $\boldsymbol{u}^{(2)}$ let's find vector $\boldsymbol{v}^{(2)}$ such that $\boldsymbol{v}^{(2)} \perp \boldsymbol{v}^{(1)}$. Hence, $\boldsymbol{v}^{(2)} = a_1\boldsymbol{v}^{(1)} + a_2\boldsymbol{u}^{(2)}$.

$$\boldsymbol{v}^{(2)} = \boldsymbol{u}^{(2)} - \frac{\boldsymbol{v}^{(1)T}\boldsymbol{u}^{(2)}}{\left|\boldsymbol{v}^{(1)}\right|^2}\boldsymbol{v}^{(1)}$$



Recursive formula: we have $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k-1)}$ and compute $\boldsymbol{v}^{(k)}$ such that $\boldsymbol{v}^{(k)} \perp \boldsymbol{v}^{(i)}$, $i = 1,2, \ldots, k\text{-}1$

$$\boldsymbol{v}^{(k)} = \boldsymbol{u}^{(k)} - \sum_{i=1}^{k-1} \frac{\boldsymbol{v}^{(i)T}\boldsymbol{u}^{(k)}}{\left|\boldsymbol{v}^{(i)}\right|^2}\boldsymbol{v}^{(i)}$$

How to use this recursion for a GI model?

General Inverse model

We have patterns $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}$ and the associated orthogonal base (via Gram-Schmidt process) $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(N)}$

$\mathbf{W}$ is computed recursively, upon adding a new input $\boldsymbol{x}$.

1. Initialize $\mathbf{W}^{(0)} = \mathbf{0}$, $i = 1$
2. $\boldsymbol{z}^{(i)} = \boldsymbol{x}^{(i)} - \boldsymbol{W}^{(i-1)}\boldsymbol{x}^{(i)}$

$$\boldsymbol{W}^{(i)} = \boldsymbol{W}^{(i-1)} + \frac{\boldsymbol{z}^{(i)}\boldsymbol{z}^{(i)T}}{\left|\boldsymbol{z}^{(i)}\right|^2}$$

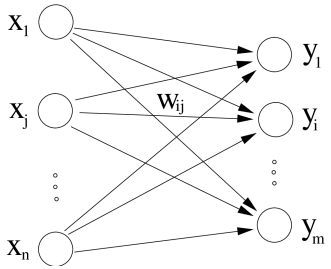
$i = i+1$

Correlation Matrix Memory

Let's have train set: $A_{train} = \{(\boldsymbol{x}^{(p)}, \boldsymbol{y}^{(p)}), p = 1,2,...,N\}$.

and let weights be set using Hebb's (1949) hypothesis:

$$w_{ij} = \sum_{p=1}^{N} x_j^{(k)} y_i^{(k)} \qquad \boldsymbol{W} = \sum_{p=1}^{N} \boldsymbol{y}^{(p)} \boldsymbol{x}^{(p)T} = \boldsymbol{Y}\boldsymbol{X}^T$$



Let's focus again on auto-associative case: $\mathbf{W} = \mathbf{X}\mathbf{X}^T$.

$$x_i = \sum_{p=1}^{N} w_{ij} x_j^{(p)}$$

Recursive computation:

$$\boldsymbol{W}(N+1) = \boldsymbol{W}(N) + \boldsymbol{x}^{(N+1)} \boldsymbol{x}^{(N+1)T}$$

Cross-talk in CMM

Response to input $\boldsymbol{x}^{(p)}$, $p \in \{1,2,...,N\}$:

$$\boldsymbol{W}\boldsymbol{x}^{(p)} = \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{x}^{(p)} = \sum_{k=1}^{N} \boldsymbol{x}^{(k)} \boldsymbol{x}^{(k)T} \boldsymbol{x}^{(p)} = \boldsymbol{x}^{(p)} \boldsymbol{x}^{(p)T} \boldsymbol{x}^{(p)} + \sum_{k=1;\, k\neq p}^{N} \boldsymbol{x}^{(k)} \boldsymbol{x}^{(k)T} \boldsymbol{x}^{(p)}$$

$$\boldsymbol{W}\boldsymbol{x}^{(p)} = \boldsymbol{x}^{(p)} \left|\boldsymbol{x}^{(p)}\right|^2 + C(p)$$

noise vector due to "cross-talk" from other inputs

If the inputs are orthogonal, then $C(p) = 0$ and $\mathbf{X}^T = \mathbf{X}^{-1} = \mathbf{X}^+$, i.e. CMM = GI.

Max. capacity of both GI and CMM equals n.

For sufficiently dissimilar inputs, CMM is a good faster alternative to GI.

Interpretation of CMM behavior

Let's consider inputs $\boldsymbol{x}^{(1)}\ \boldsymbol{x}^{(1)} \ldots \boldsymbol{x}^{(N)}$ as realizations of a random variable $X = (X_1, X_2, \ldots, X_n)^T$, with zero means, i.e. $\langle X_i \rangle = 0$, $i = 1,2,...,n$. Then

$$w_{ij} = \sum_{k=1}^{N} x_j^{(k)} x_i^{(k)} \propto cov(X_i, X_j) \qquad \text{[unbiased estimate]}$$

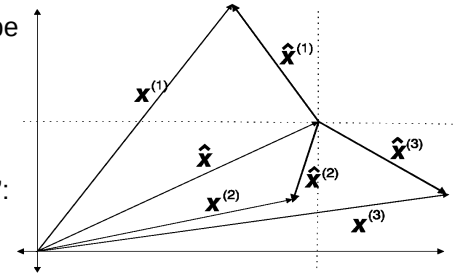
$$cov(X_i, X_j) = \langle (X_i - \langle X_i \rangle).(X_j - \langle X_j \rangle) \rangle = \langle X_i X_j \rangle$$

w_{ij} carries information about linear correlations between nodes *i* and *j*.

Reduced (noisy) input component x_i can be recovered (bootstrapped) by other active components strongly correlated with *i*.

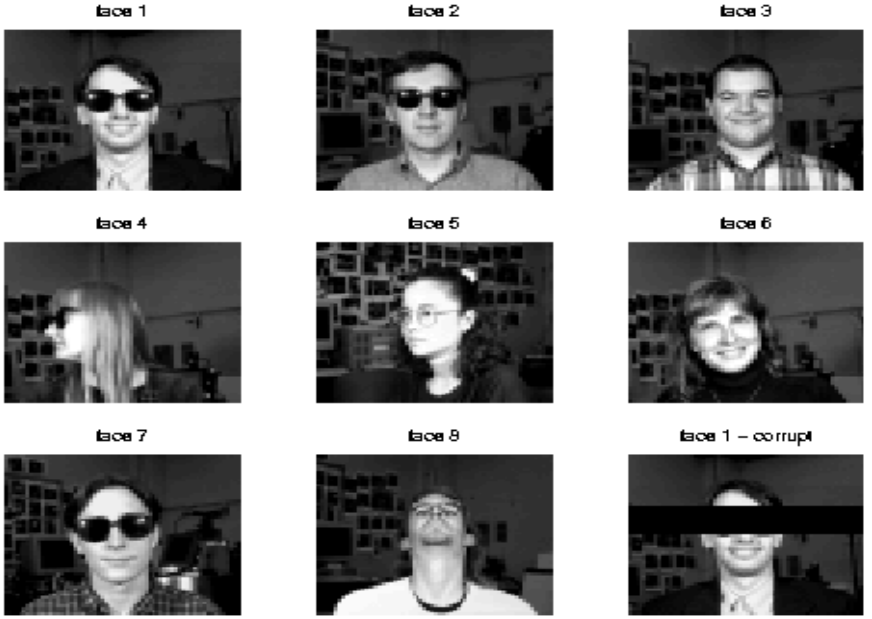
Shifting coordinates decreases "cross-talk":

$$\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{x}^{(i)} - \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{x}^{(k)}$$

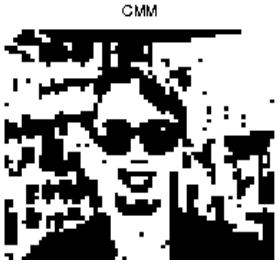


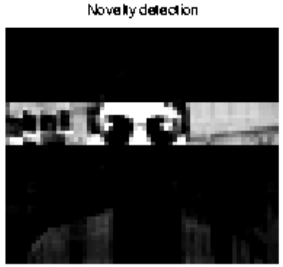
(Kvasnička et al, 1997)

Example: 8 faces from CMU image data repository



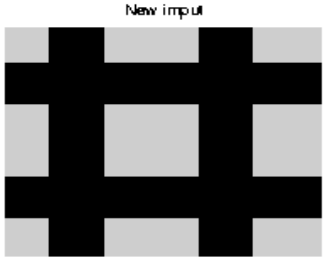
Recall by GI and CMM

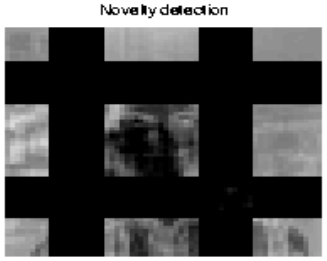




◄► GI's novelty detection

Response to a new input





Summary

- Linear models were studied during connectionist depression in the 1970s
- Single layer models as auto-associative memories
- Analytic solutions possible
- General inverse model – noise filtering by projection to linear manifold (of the training data)
- GI – as novelty detector
- Correlation Matrix Memory – Hebbian-based learning, subject to cross-talk
- GI better in general, for sufficiently dissimilar inputs both models are comparable