



Neural Networks

Lecture 3

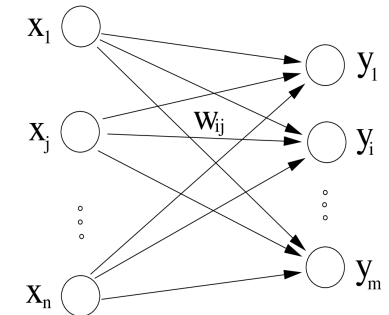
Single-layer models

Linear NN models

Input vector: $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$

Output vector: $\mathbf{y} = [y_1, y_2, \dots, y_m]^T$

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



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

- ☹ 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).

Analytic solution

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

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

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

$$\begin{matrix} [\mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(N)}] & = & \mathbf{W} & \times & [\mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(N)}] \\ & & & & & (m \times n) & & & & & (n \times N) \end{matrix}$$

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

Auto-association in a linear network

Let us now look at autoassociation: $\mathbf{y}^{(p)} = \mathbf{x}^{(p)}$, $\dim(\mathbf{y}) = \dim(\mathbf{x}) = n$

This can be useful for a memory, if $n > N$ (i.e. few examples of large dimension). We want to remember N **prototypes** $[\mathbf{x}^{(1)} \mathbf{x}^{(2)} \dots \mathbf{x}^{(N)}]$.

Goal: train a linear model 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, restrictive case ($N = n$, 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 \mathbb{R}^n .

Linear manifold $\mathcal{L} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = a_1 \mathbf{x}^{(1)} + a_2 \mathbf{x}^{(2)} + \dots + a_N \mathbf{x}^{(N)}, a_p \neq 0\}$
 $\mathcal{L} \subset \mathbb{R}^n$

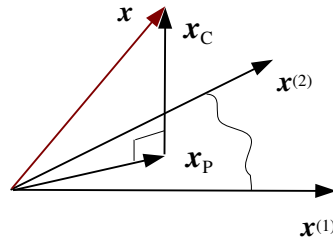
Orthogonal complement $\mathcal{L}^\perp = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \perp \mathcal{L}\}$

Hence, $\mathcal{L} \oplus \mathcal{L}^\perp = \mathbb{R}^n$

Each vector $\mathbf{x} \in \mathbb{R}^n$ can be uniquely decomposed:

$$\mathbf{x} = \mathbf{x}_p + \mathbf{x}_c$$

where $\mathbf{x}_p \in \mathcal{L}$ and $\mathbf{x}_c \in \mathcal{L}^\perp$.



Gram-Schmidt orthogonalization process

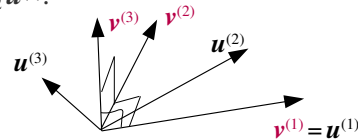
Let's have a base $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(k)} \in \mathcal{L}$, for which we want to create an orthogonal base $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(k)} \in \mathcal{L}$.

Procedure:

Set $\mathbf{v}^{(1)} = \mathbf{u}^{(1)}$. In space with base $\mathbf{v}^{(1)}, \mathbf{u}^{(2)}$ let's find vector $\mathbf{v}^{(2)}$ such that

$\mathbf{v}^{(2)} \perp \mathbf{v}^{(1)}$. Hence, $\mathbf{v}^{(2)} = a_1 \mathbf{v}^{(1)} + a_2 \mathbf{u}^{(2)}$.

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



Recursive formula: we have $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(k-1)}$ and compute $\mathbf{v}^{(k)}$ such that

$\mathbf{v}^{(k)} \perp \mathbf{v}^{(i)}, i = 1, 2, \dots, k-1$

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

How to use this recursion for a GI model?

What does an autoassociative NN do?

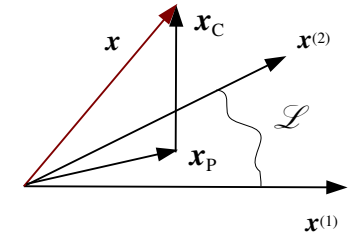
Training set $A_{\text{train}} = \{\mathbf{x}^{(p)}, p = 1, 2, \dots, N\}$ forms the linear manifold \mathcal{L} .

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

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

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

Now assume: you learned N patterns, and want to add $(N+1)$ -st pattern.
 How to change \mathbf{W} efficiently?



5

6

General Inverse model

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

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

1. Initialize $\mathbf{W}^{(0)} = \mathbf{0}, i = 1$

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

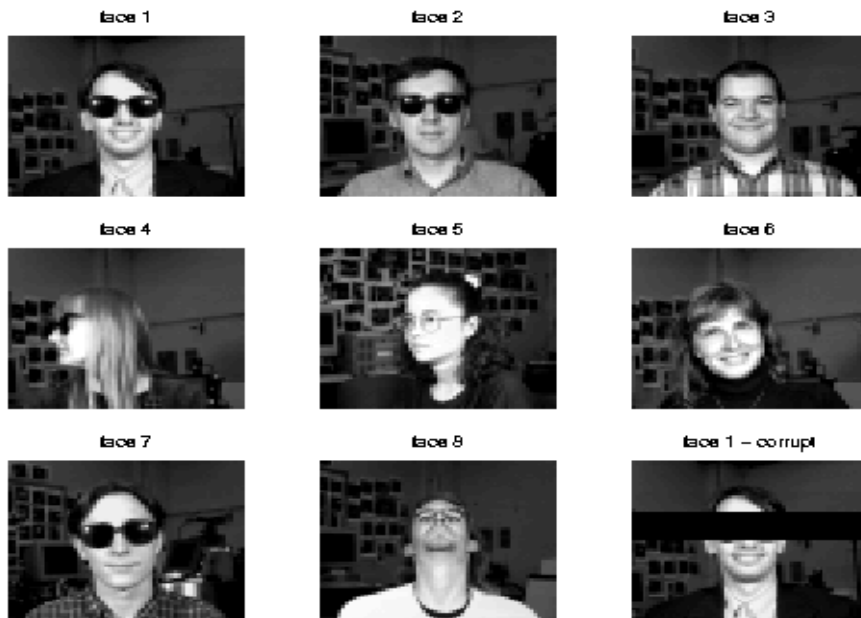
$$\mathbf{W}^{(i)} = \mathbf{W}^{(i-1)} + \frac{\mathbf{z}^{(i)} \mathbf{z}^{(i)T}}{|\mathbf{z}^{(i)}|^2}$$

$i = i+1$

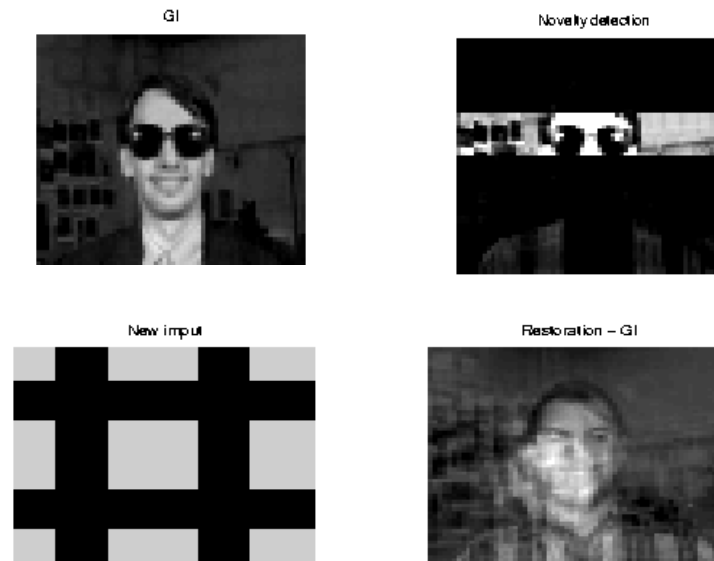
7

8

Example: 8 faces from CMU image data repository



Recall by GI and novelty detection

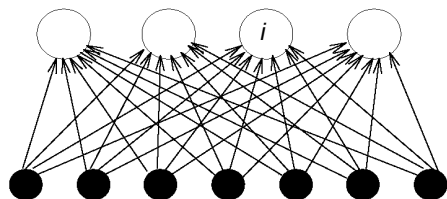


9

10

Single-layer perceptrons

- $i = 1, 2, \dots, m$ neurons work independently
- Each neuron linearly separates input space into two subspaces



- For sigmoidal neurons we get learning rule:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha (d_i - y_i) f'_i(\text{net}) x_j$$
- for linear neurons we get the least-means-square (LMS) learning rule

$$w_{ij}(t+1) = w_{ij}(t) + \alpha (d_i - y_i) x_j$$
- perceptron learning \leftrightarrow adaptive filtering (in signal processing)

Softmax regression

- Softmax – useful for classification
- Output yields conditional probabilities for each class

$$y_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)}$$
- Link to probabilistic modeling: maximizing $P(Y|X)$, i.e. minimizing $-\log P(Y|X)$, leads to optimal predictions.
- This yields the loss function $e_{CE} = -P(y|\mathbf{x}) = -\sum_i d_i \log y_i$
- $e_{CE} = -\sum_i d_i \log \sum_k \exp(o_k) - \sum_i d_i \exp(o_j) = -\log \sum_k \exp(o_k) - \sum_i d_i \exp(o_j)$
- The derivative:

$$\frac{d e_{CE}}{d o_i} = \frac{\exp(o_i)}{\sum_k \exp(o_k)} - d_i = P(y=i|\mathbf{x}) - d_i$$
- has the same effect as in case of squared error

11

12

Link to information theory

- assume a random discrete event that can take values j
- Entropy: $H(p) = -\sum_j p(j) \log p(j)$, reaches max. for uniform distribution
- In order to encode data drawn randomly from the distribution p , we need at least $H(p)$ “nats” to encode it (analogy to a “bit”).
- $-\log p(j)$ quantifies surprisal observing an event j having assigned it a (subjective) probability $p(j)$
- Cross entropy $H(p, q)$ is the expected surprisal of an observer with subjective probabilities q upon seeing data that was actually generated according to probabilities p .
- Kullback-Leibler divergence measures the distance b/w two distributions: $D(p, q) = H(p, q) - H(p) = -\sum_j p(j) \log (p(j) / q(j))$
- Minimizing $D(p, q)$ corresponds to minimizing $H(p, q)$

13

Sequential and batch modes of training

Sequential mode

- on line (example-by-example), stochastic
- able to track small changes in training data
- easier to implement, requires less local storage
- difficult to establish theoretical conditions for convergence

Batch mode

- adaptation performed at the end of each epoch, deterministic
- provides an accurate estimate of gradient vector
- parallelization possible

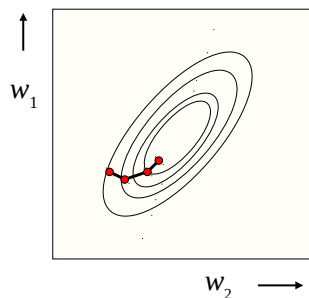
$$E_{av} = 1/(2N) \sum_{p=0}^N (d^{(p)} - y^{(p)})^2 \quad \Delta w_i \propto -\partial E_{av}(t) / \partial w_i$$

Mini-batches – best of two worlds (typical minibatch size 50–256)

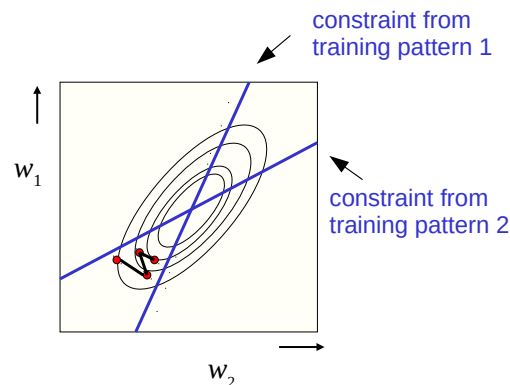
14

Online versus batch learning

Batch learning - does steepest descent on the error surface



Online learning - zig-zags around the direction of steepest descent



Summary

- Linear models were studied during connectionist depression in the 1970s
- Single layer models are linear in parameters
- Hence, analytic solutions possible (via pseudoinverse)
- Tasks: regression, classification, auto-associative memory
- Data classification – soft assignment
- General inverse – noise filtering by projection to linear manifold (of the training data)
- GI – as novelty detector
- GI better in general, for sufficiently dissimilar inputs both models are comparable

15

16