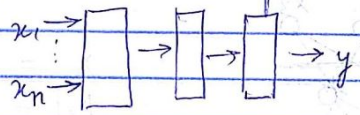
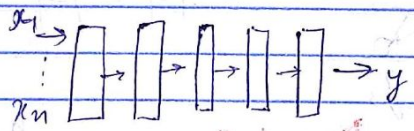
Feb 7th 2017

MLP's have been around for many years.

Shallow networks: 

Deep MLP's 

"intractable" { for chess: 2^{400} ($n = 400$)
for face recognition: $n = 500,000$.

Sentiment: MLP's are useless.

These problems are called NP-hard

Eg. n = # of weights = 1000
Say this has $5 \times 200 = 1000$ weights. If we assume the weights are binary, we have $2^n = 2^{1000}$

At least 5 hidden layers : Deep

MLPs are useless :
because their traing is impossible.

Deep learning
↳ deep Autoencoders
↳ Convolutional neural networks
} → Supervised

the auto encoder is supervised cuz the desired output is known. i.e, the desired output is the input.

2 types of ANNs

Discriminative Classification — SVM, MLP, CNNs

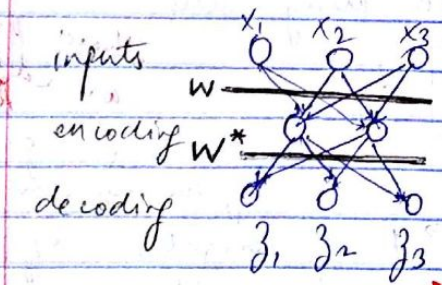
Generative models the data — AE, DBNs.

Auto Encoders.

inputs $x \in [0,1]^d$ → encoding $y \in [0,1]^{d'}$ → decoding $z = g(w^* y + b^*)$

$$y = g(wx + b)$$

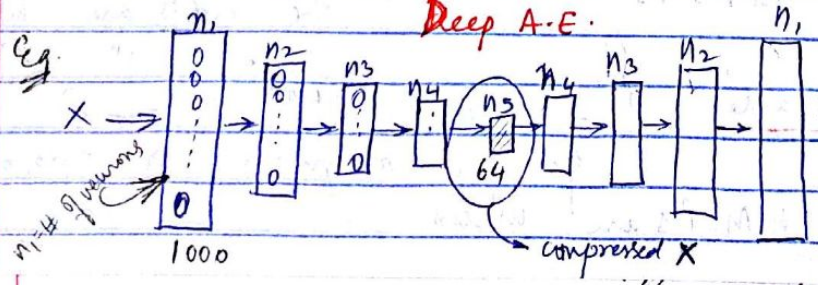
$$g(x) = \frac{1}{1+e^{-x}}$$



$x = z \Rightarrow$ auto encoder

$|x| = |z|$

Deep A.E.



$n_1 > n_2 > n_3 > n_4 > n_5$
and $n_2 = n_1/2 \ldots$
(becomes half everytime & build it back up)

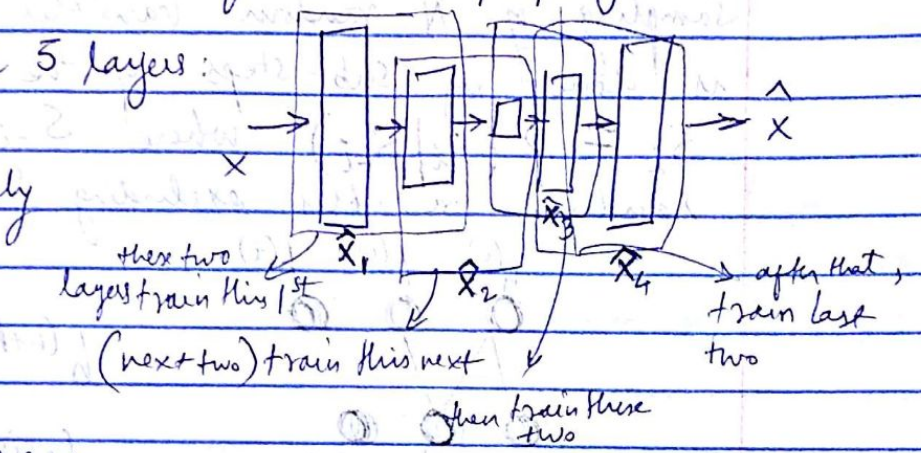
$L(x, \hat{x}) = \|x - \hat{x}\|^2$

If we interpret inputs as bit vectors / bit probability

$$(x, z) = \sum_{k=1}^{d} \left[x_k \log z_k + (1 - x_k) \log (1 - z_k) \right]$$

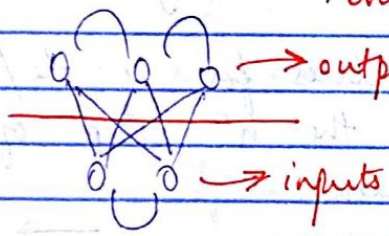
Problem: We cannot train a deep MLP. MLP can pick up the structure if it is properly initialized.

Lets say I have 5 layers:

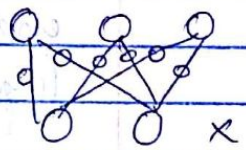


train two separately

Restricted Boltzmann Machines or RBMs.



everything is connected to everything.



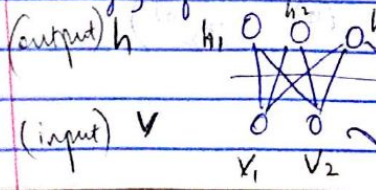
Boltzmann machine ↑ Remove inter layer connections ↗ This is "Restricted Boltzmann machine"

- unsupervised.
- less to more ∴ generative

(some is discriminative not generative)

RBMs: - special form of Markov Random Fields.
- uses energy function.
- it's unsupervised & shallow
- introduced by Smolensky in 1986 : Harmonium

Energy of RBM



$$E(v,h) = -b^T v - c^T h - h^T W v$$

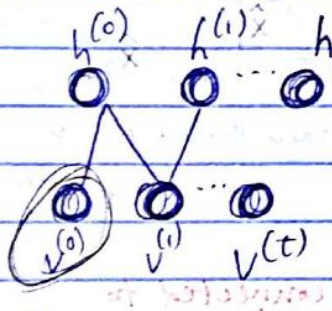
weights

$$\Delta W_{ij} = \eta \left[\langle v_i, h_j \rangle - \langle \hat{v}_i, \hat{h}_j \rangle \right]$$

data — sampled data

Gibbs Sampling It obtains a sample distribution $p(x)$ by running a Markov chain to convergence.

Sampling of N random variables $S = (S_1, S_2 \dots S_N)$ is done by sub-steps of the form $S_i \simeq p(S_i | S_{-i})$ where S_{-i} contains the $N-1$ random variables excluding S_i.



evidence (input)

$$h^{(n+1)} \simeq g\left(W^T v^{(n)} + c\right)$$

$$v^{(n+1)} \simeq g\left(W h^{(n+1)} + b\right)$$

$v^{(n)}$, $h^{(n)}$ values for visible / hidden units at the n^{th} step of the Markov chain.

Algorithm: "Contrastive Divergence". → published on LEARN

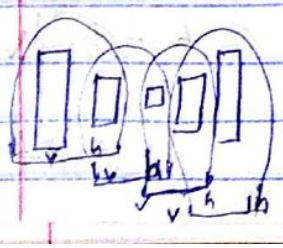
If we have distribution of random variables ↓ and we select z^1 at time n, $= z_1^{(n)}$ | $p(z_1, z_2, z_3)$
z^2 at time n, $= z_2^{(n)}$
z^3 at time n. $= z_3^{(n)}$

then according to the algorithm:

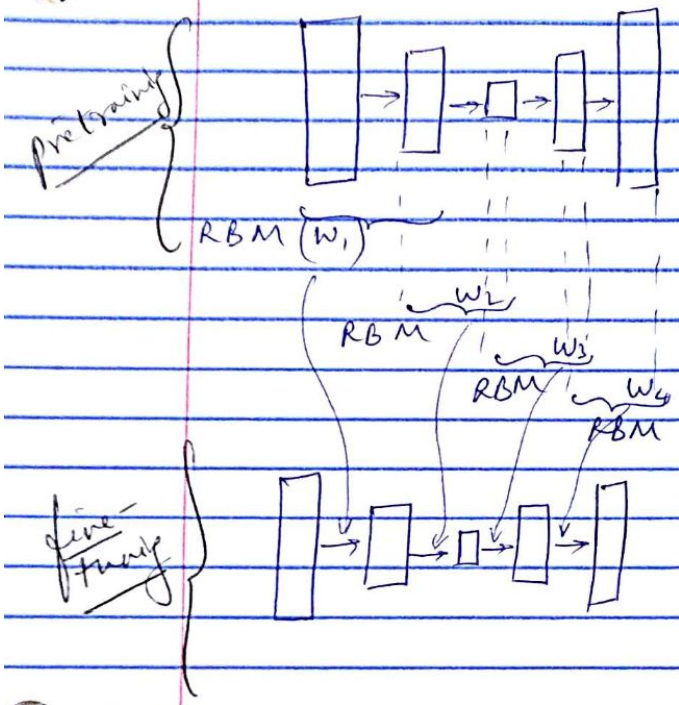
$$z_1^{(n+1)} = P\left(z_1 \mid z_2^{(n)}, z_3^{(n)}\right)$$

$$z_2^{(n+1)} = P\left(z_2 \mid z_1^{(n)}, z_3^{(n)}\right)$$

$$z_3^{(n+1)} = P\left(z_3 \mid z_1^{(n)}, z_2^{(n)}\right)$$



(deeplearning.net)

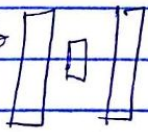


The first 2 layers: we do RBM & get a set of weights W_1.
Similarly, we get $W_1 \cdots W_4$.

Trained as usual (Back propagation) initialize with weights obtained in previous step.

Several A.E.

→ Denoising. A.E.

$x \rightarrow$ add noise $\rightarrow \tilde{x} \rightarrow$  $\rightarrow x$

→ Stacked A.E.

train shallow A.E.s separately & stack them.