Autoencoder

Autoencoder

- **Unsupervised** learning where Neural networks are subject to the task of **representation learning** (i.e. Encoding of input data / inner structure of the data)
- Impose a **bottleneck** in the network
- The bottleneck forces a **compressed** knowledge representation of the input

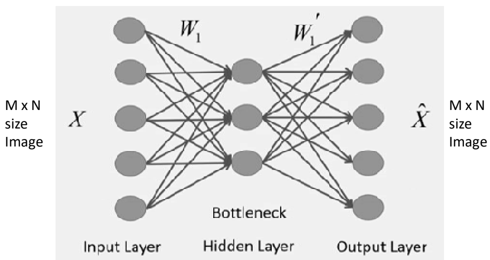
Autoencoder

- Input – network – re constructed output (should be identical to the input)
  - Network may eventually learn Identity mapping
  - Does not learn the representation
- Compressed representation is required
  - Done by the bottleneck layer

Autoencoder

- Assumption - High degree of correlation exists in the data for the compressed domain representation
- Two different functions required
  - Encoding (input to compressed representation)
  - Decoding (compressed representation to the reconstructed output)

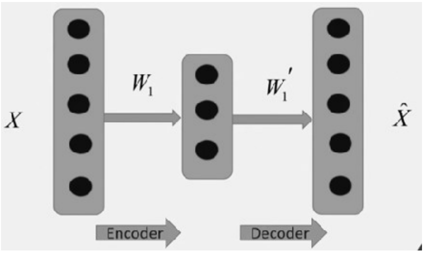
Autoencoder



Autoencoder

- No. of nodes in the bottleneck layer is much less than the no. of nodes in the input layer
  - For d dimension, it should be d nodes
- No. of nodes in the input layer is the same as the number of nodes in the output layer

Autoencoder



Expectation

- Sensitive enough to input for accurate reconstruction

Expectation

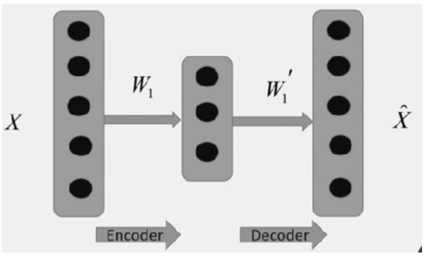
- Sensitive enough to input for accurate reconstruction
- Insensitive enough that it does not memorize or overfit the training data

⇓

Loss function = L (X, X̂) + Regularizer

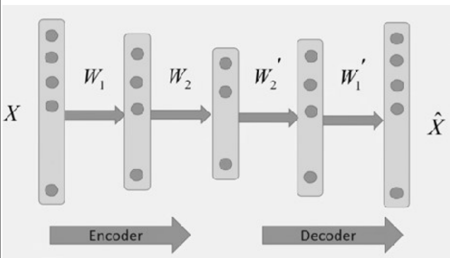
Regularizer learns salient features

Undercomplete Autoencoder

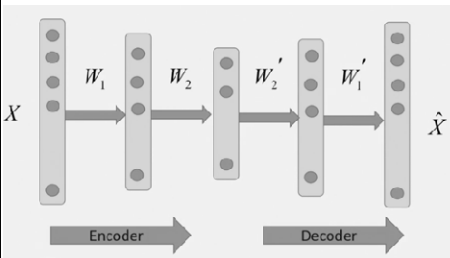


$$L(X,\hat{X})=\frac{1}{2}\sum_N \|X-\hat{X}\|^2$$

Stacked Autoencoder

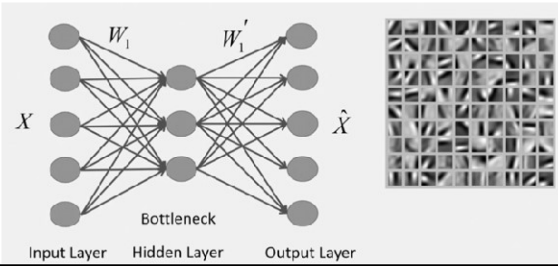


Stacked Autoencoder

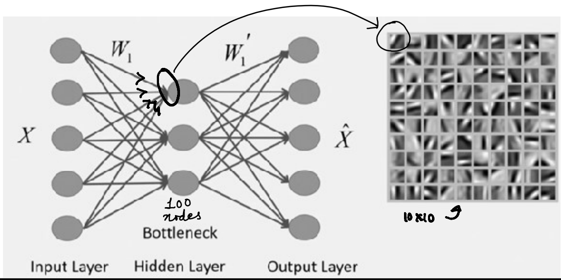


$$L(X,\hat{X})=\frac{1}{2}\sum_N \|X-\hat{X}\|^2$$

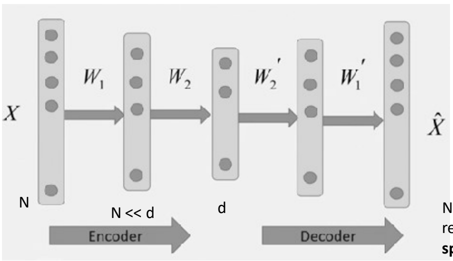
Autoencoder



Autoencoder



Deep Autoencoder



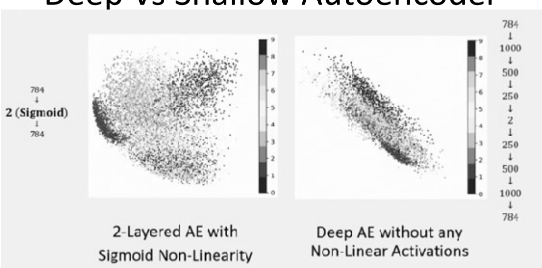
$$L(X,\hat{X}) = \frac{1}{2}\sum_N \left\| X - \hat{X} \right\|^2$$

N -> d compressed representation (**Latent space representation**)

Autoencoder

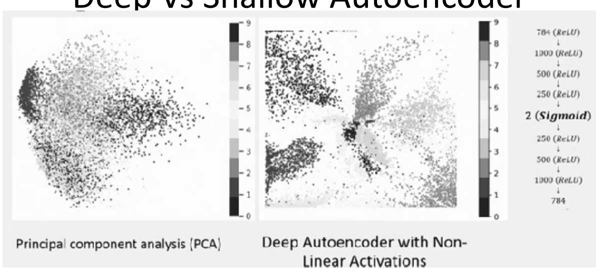
- Higher dimensional data is represented in to Lower dimensional space representation
- One of the functionality of Autoencoder is dimensionality reduction of the input data

Deep Vs Shallow Autoencoder



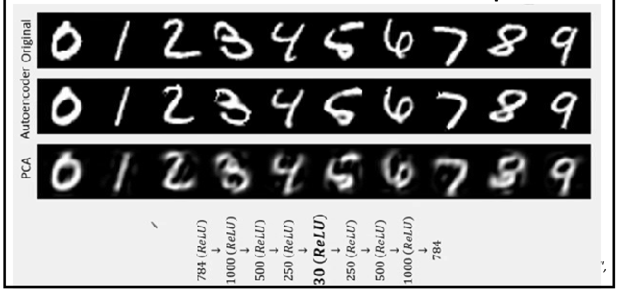
Source : G. E. Hilton and R. R. Salakhutdinov: "Reducing the Dimensionality of Data with Neural Networks", Science, Vol 313, July 2006, pp. 504-507

Deep Vs Shallow Autoencoder

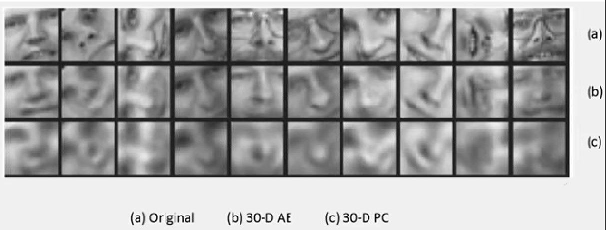


Source : G. E. Hilton and R. R. Salakhutdinov: "Reducing the Dimensionality of Data with Neural Networks", Science, Vol 313, July 2006, pp. 504-507

Reconstruction from Latent space

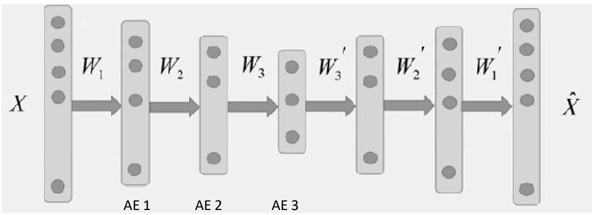


Reconstruction from Latent space



Source : G. E. Hilton and R. R. Salakhutdinov: "Reducing the Dimensionality of Data with Neural Networks", Science, Vol 313, July 2006, pp. 504-507

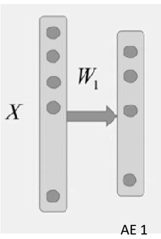
Autoencoder Training



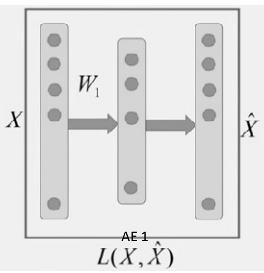
Autoencoder Training

- Layer by Layer Pre training
  - To reduce complexity in learning
  - To reduce no. of weights to learn at a time

Autoencoder Training



Autoencoder Training

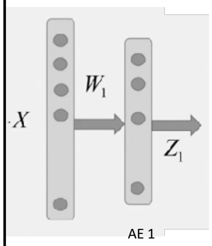


To learn weight vector W1, use back propagation

It will reduce error between X and $\hat{X}$

Once the weight vector W1 is learnt, we discard the first level of Decoder that we have used

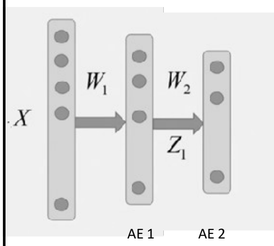
Autoencoder Training



Latent space vector Z1 is the output.

Next level of the Encoder will try to Encode Z1

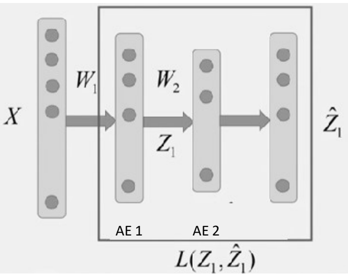
Autoencoder Training



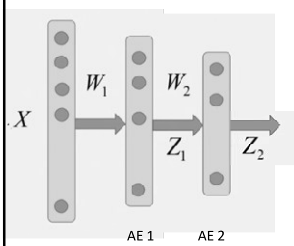
AE 2 will try to encode Z1

And will try to learn weight matrix W2

Autoencoder Training



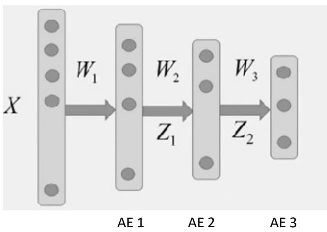
Autoencoder Training



Decoder is removed

Output of AE 2 is Z2

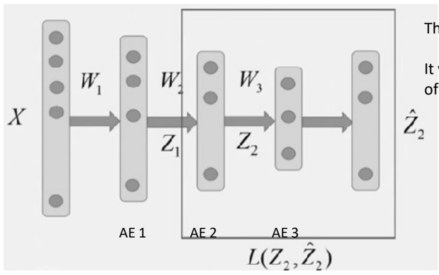
Autoencoder Training



Third Autoencoder AE 3 is put

Weight matrix W3 is to learn

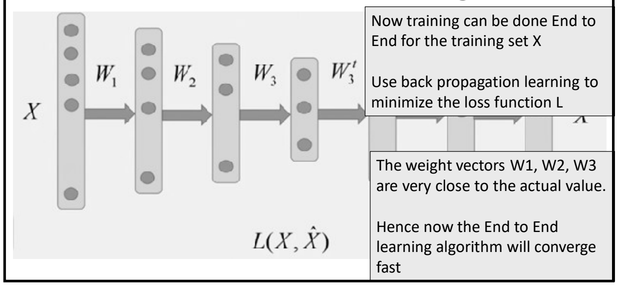
Autoencoder Training



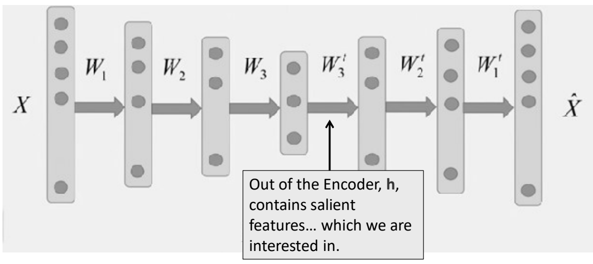
Third Decoder is added.

It will decode the output of AE 3

Autoencoder Training



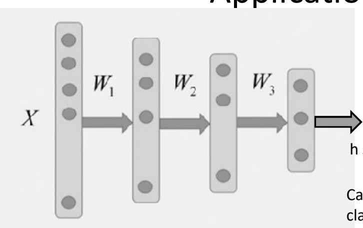
Autoencoder Training



Autoencoder Training

- The salient features (reduced features) can be used for various applications
  - Classification
  - Segmentation (pixel classification)

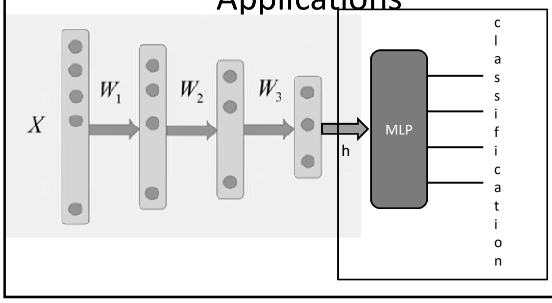
Applications



h salient features

Can be fed to a classifier for classification
(SVM/DT/ANN MLP etc.)

Applications



Autoencoder

- Sparse Autoencoder

Autoencoder

- Sparse Autoencoder
  - Interesting features can be learnt even when number of nodes in the hidden layer is large
  - Introduces sparsity constraint on the hidden layer nodes that penalize activations within a layer
  - Network learns encoding-decoding that relies on activating a small number of neurons
  - It regularize the activations, not the weights

Autoencoder

- Sparse Autoencoder – Sparsity constraint

Sigmoid activation function

$a_j^h \rightarrow$ Activation of j^{th} Neuron in hidden layer h

$a_j^h \rightarrow 1 \Rightarrow$ Neuron is active

Average activation $\rightarrow \hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} a_j^h(x_i)$

Constraint $\rightarrow \hat{\rho}_j = \rho$

$\rho \rightarrow$ sparsity parameter (typically a small value)

Sparsity parameter (ρ)

degree of sparsity that we want to impose on the network layer

Usually kept very low

Autoencoder

- Sparse Autoencoder – Sparsity constraint

$$\text{Regularizer}: \sum_{j=1}^{N_h}\left[\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] \Rightarrow \boxed{\sum_{j=1}^{N_h} KL(\rho \parallel \hat{\rho}_j)}$$

Regularization term :

KL Divergence between two distributions (ρ and ρ̂)

$$\boxed{J_{sparse}(W) = L(X, \hat{X}) + \lambda\sum_j KL(\rho \parallel \hat{\rho}_j)}$$

Complete Loss function

Back propagation algorithm can be applied to optimize the overall function

Autoencoder

- Sparse Autoencoder – Sparsity constraint

Back propagation

$$\delta_i^k = O_i^k(1-O_i^k)\sum_{j=1}^{M_{k+1}} \partial_j^{k+1} W_{ij}^{k+1}$$

$$\delta_i^k = O_i^k(1-O_i^k)\left[\left(\sum_{j=1}^{M_{k+1}} \partial_j^{k+1} W_{ij}^{k+1}\right) + \lambda\left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}\right)\right]$$

Autoencoder

- Sparse Autoencoder – Sparsity constraint
- Even though the number of nodes in hidden layer is large, but still it learns compressed domain representation
  - Because with the help of sparsity constraint, number of nodes to be active is controlled
  - It learns salient features and not a simple identity mapping
  - Different inputs having particular feature will make a particular node active in the hidden layer

Autoencoder

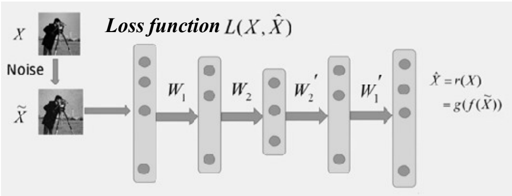
- Denoising Autoencoder

Autoencoder

- Denoising Autoencoder
- The autoencoder learns generalizable encoding decoding scheme
  - While training, use corrupt/noisy data as input but output as uncorrupted original data
  - The model can not memorize the training data as input and target output is not same any more

Autoencoder

- Denoising Autoencoder



Autoencoder

- Contractive Autoencoder

Autoencoder

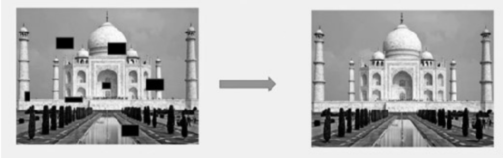
- Contractive Autoencoder
  - For similar inputs – learned encoding (compress domain representation) should also be similar
  - Hidden layer activation variation with input data should be small
  - Effectively the Model learns to contract a neighbourhood of inputs to a small neighbourhood of outputs

Autoencoder

- Contractive Autoencoder
  - A contractive autoencoder makes this encoding less sensitive to small variations in its training dataset.
  - This is accomplished by adding a regularizer, or penalty term, to whatever cost or objective function the algorithm is trying to minimize.
  - The end result is to reduce the learned representation's sensitivity towards the training input.

Applications

- Image inpainting



- Autoencoder learns salient features. From them, it can reconstruct the input
- It can remove corrupted region from the input

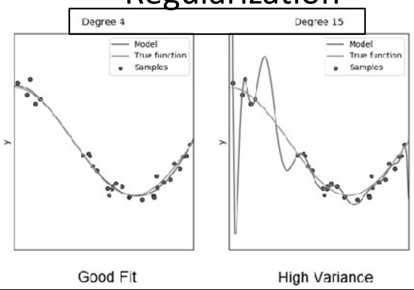
Applications

- To detect/identify Abnormal Event
  - On a road, only pedestrians are allowed to walk
  - While training, the encoder is given only the proper sequences, i.e. only pedestrians are present
  - While in testing, if any car comes on that pedestrian road, car can not be properly reconstructed (as it was never given while training)
  - So, in output the region where the reconstructed error is very large, that region has some Abnormal Event

Regularization

- Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain
  - L1 regularizer
  - L2 regularizer
  - Dropout

Regularization



Regularization

- Regularization refers to a set of different techniques that lower the complexity of a neural network model during training, and thus prevent the overfitting