

18AIC301J: DEEP LEARNING TECHNIQUES

B. Tech in ARTIFICIAL INTELLIGENCE, 5th semester

Faculty: **Dr. Athira Nambiar**

Section: A, slot:D

Venue: TP 804

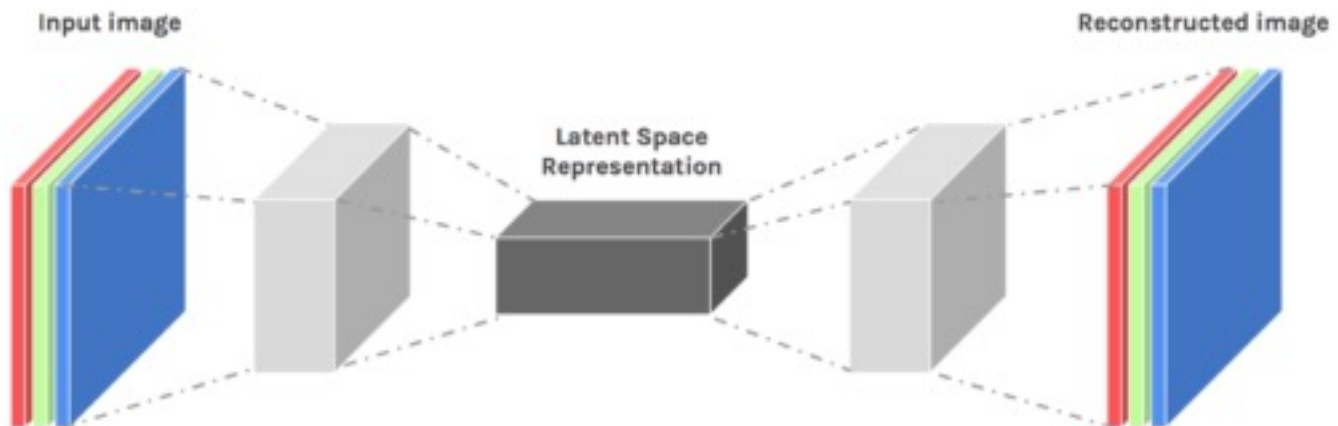
Academic Year: 2022-22

UNIT-2

Limitations of gradient descent learning algorithm, Contour maps,
Momentum based gradient descent, Nesterov accelerated gradient Descent, AdaGrad, RMSProp, Adam learning Algorithm, Stochastic gradient descent
Implement linear regression with stochastic gradient descent
Mini-batch gradient descent, Bias Variance tradeoff, Overfitting in deep neural networks,
Hyperparameter tuning, Regularization: L2 regularization, Dataset Augmentation and Early stopping
Implement linear regression with stochastic mini-batch gradient descent and compare the results with previous exercise.
Dimensionality reduction, Principal Component Analysis, Singular value decomposition
Autoencoders, Relation between PCA and Autoencoders, Regularization in Autoencoders
Optimizing neural networks using L2 regularization, Dropout, data augmentation and early stopping.

Autoencoders

Autoencoders are designed to reproduce their input, especially for images. Key point is to reproduce the input from a learned encoding.



Autoencoders

Compare PCA/SVD

- PCA takes a collection of vectors (images) and produces a usually smaller set of vectors that can be used to approximate the input vectors via linear combination.
- Very efficient for certain applications.
- Neural network autoencoders
 - Can learn nonlinear dependencies
 - Can use convolutional layers
 - Can use transfer learning

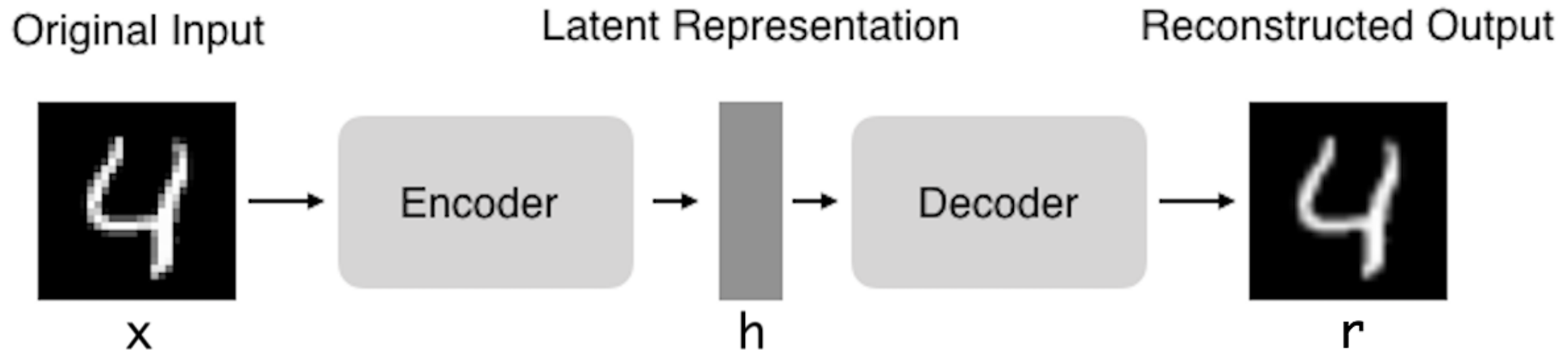
Autoencoders: Structure

Encoder: compress input into a latent-space of usually smaller dimension.

$$h = f(x)$$

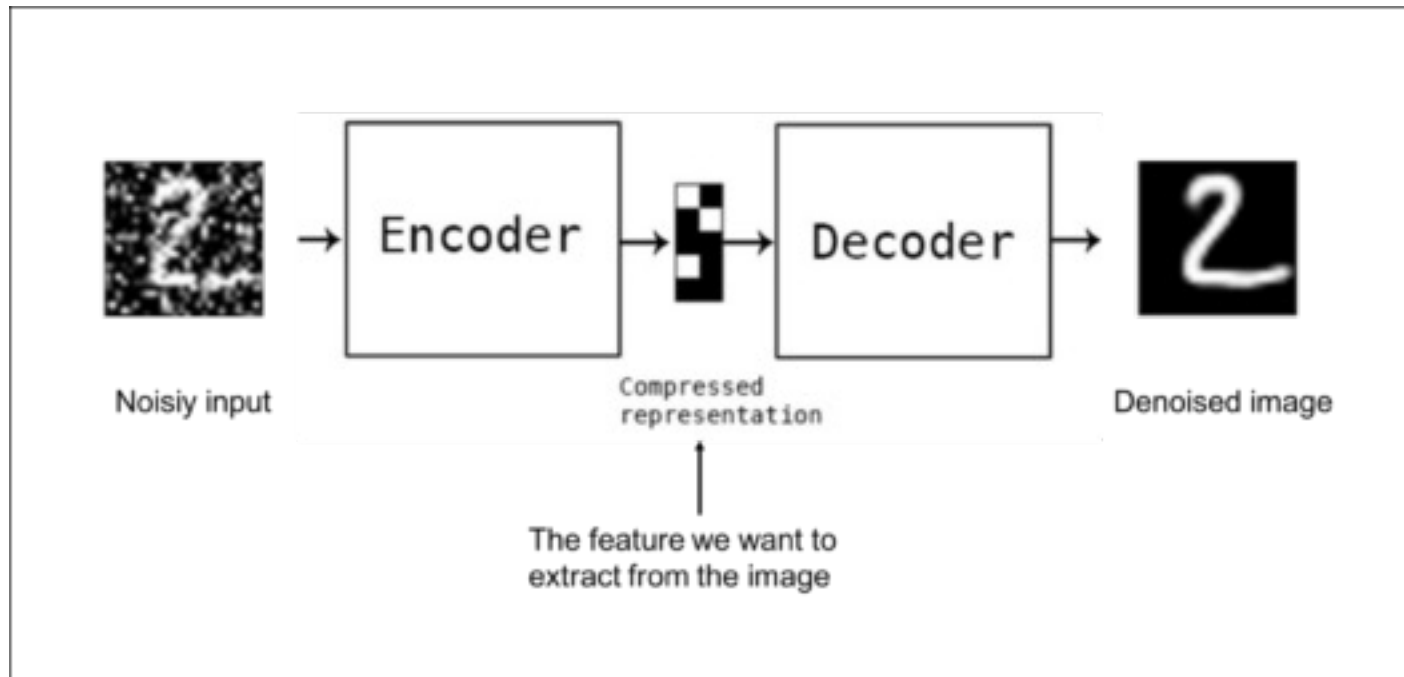
Decoder: reconstruct input from the latent space.

$$r = g(f(x)) \text{ with } r \text{ as close to } x \text{ as possible}$$



Autoencoders: Applications

Denoising: input clean image + noise and train to reproduce the clean image.



Autoencoders: Applications

Image colorization: input black and white and train to produce color images



Autoencoders: Applications

Watermark removal



Properties of Autoencoders

Data-specific: Autoencoders are only able to compress data similar to what they have been trained on.

Lossy: The decompressed outputs will be degraded compared to the original inputs.

Learned automatically from examples: It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

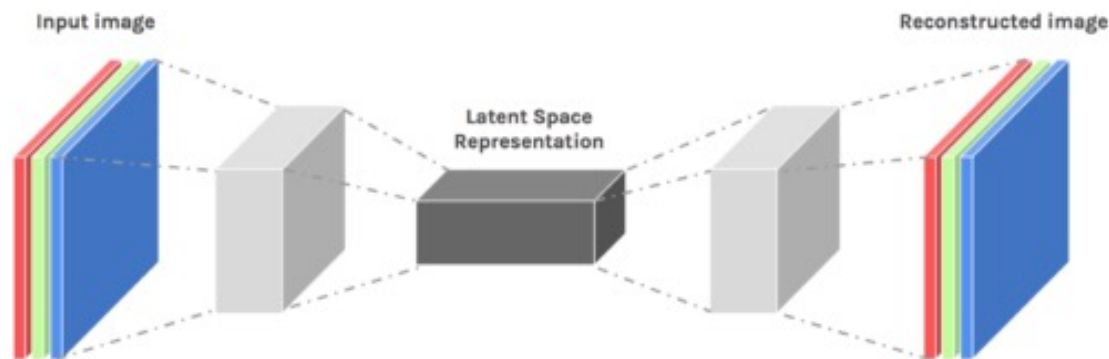
Capacity

- As with other NNs, overfitting is a problem when capacity is too large for the data.
- Autoencoders address this through some combination of:
 - Bottleneck layer – fewer degrees of freedom than in possible outputs.
 - Training to denoise.
 - Sparsity through regularization.

Autoencoder

A reliable autoencoder must make a tradeoff between two important parts:

- Sensitive enough to inputs so that it can accurately reconstruct input data
- Able to generalize well even when evaluated on unseen data



The first part is the **loss function** (e.g. mean squared error loss) calculating the difference between input data and output data while the second term would act as **regularization term** which prevents autoencoder from overfitting.

$$Obj = L(x, \hat{x}) + regularizer$$

Loss function of autoencoder

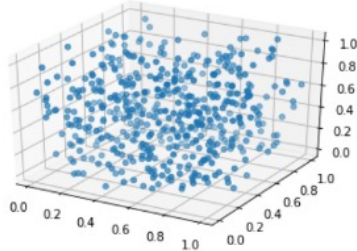
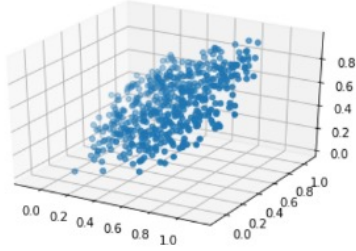
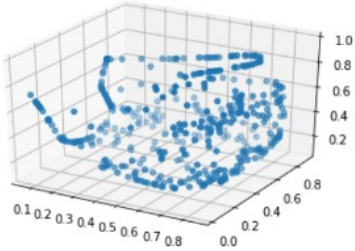

Relation between PCA and Autoencoders

PCA vs Autoencoder

- Although PCA is fundamentally a linear transformation, auto-encoders may describe complicated non-linear processes.
- Because PCA features are projections onto the orthogonal basis, they are completely linearly uncorrelated. However, since autoencoded features are only trained for correct reconstruction, they may have correlations.
- PCA is quicker and less expensive to compute than autoencoders.
- PCA is quite similar to a single layered autoencoder with a linear activation function.
- Because of the large number of parameters, the autoencoder is prone to overfitting. (However, regularization and proper planning might help to prevent this).
- If the features have non-linear relationship with each other then autoencoder will be able to compress the information better into low dimensional latent space leveraging its capability to model complex non-linear functions.

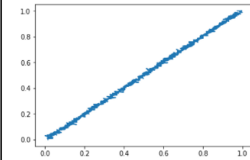
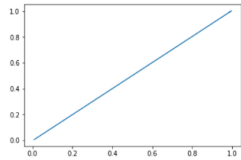
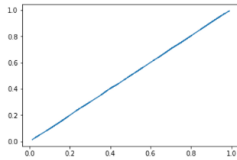
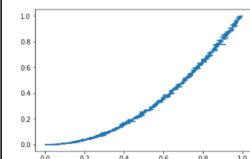
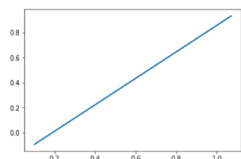
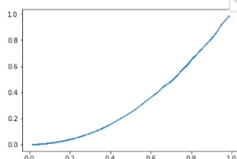
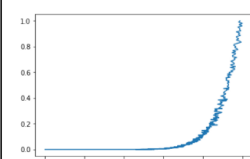
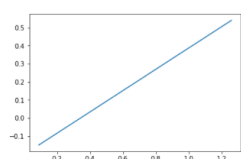
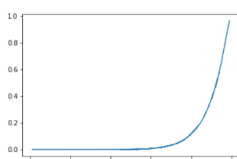
Relation between PCA and Autoencoders

Consider a random data without any collinearity. All features are independently sampled from a uniform distribution and have no relationship with each other. A two-dimensional latent space is used for both PCA and Autoencoder.

	Feature Space	PCA Reconstruction	<u>Auto Encoder</u> Reconstruction
Random Data			
Reconstruction Cost (MSE) 		0.024	0.010

PCA is able to retain the projection onto the plane with maximum variance, and loses a lot of information because the random data did not have a underlying 2 dimensional structure. Autoencoder also does poorly since there was no underlying relationship between features.

Relation between PCA and Autoencoders

Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
$y=mx+c$			
$y=mx^2+c$			
$y=mx^8+c$			

It is evident if there is a non linear relationship (or curvature) in the feature space, autoencoded latent space can be used for more accurate reconstruction. Where as PCA only retains the projection onto the first principal component and any information perpendicular to it is lost.

Regularization in Autoencoders

Regularized autoencoders

There are other ways to constrain the reconstruction of an autoencoder than to impose a hidden layer of smaller dimensions than the input.

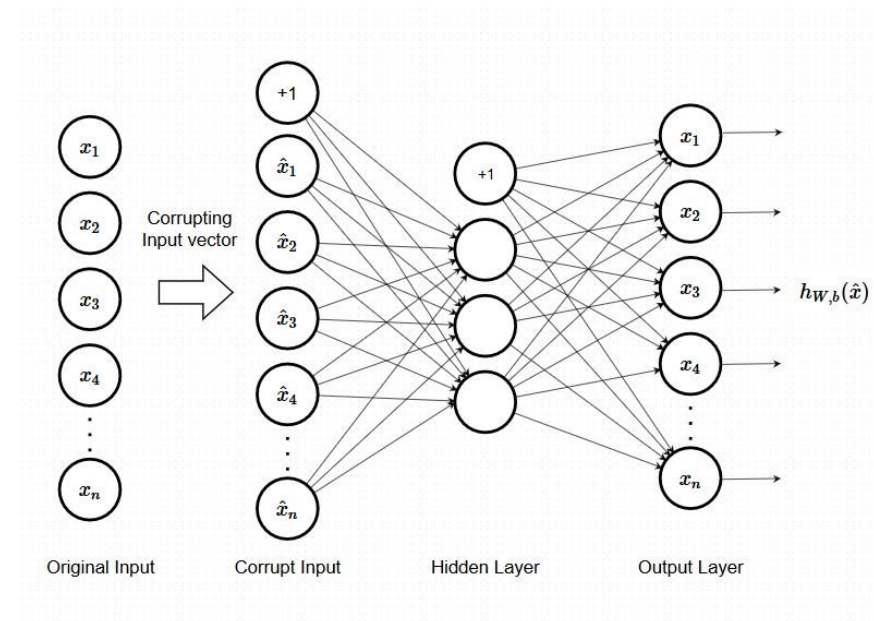
Regularized autoencoders use a loss function that encourages the model to have other properties besides copying its input to its output.

We can generally find two types of regularized autoencoder:

(i) denoising autoencoder and (ii) sparse autoencoder.

Denoising autoencoders

- A **Denoising** Autoencoder is a modification on the autoencoder to prevent the network learning the identity function.
- Specifically, if the autoencoder is too big, then it can just learn the data, so the output equals the input, and does not perform any useful representation learning or dimensionality reduction.
- Denoising autoencoders solve this problem by corrupting the input data on purpose, adding noise or masking some of the input values.



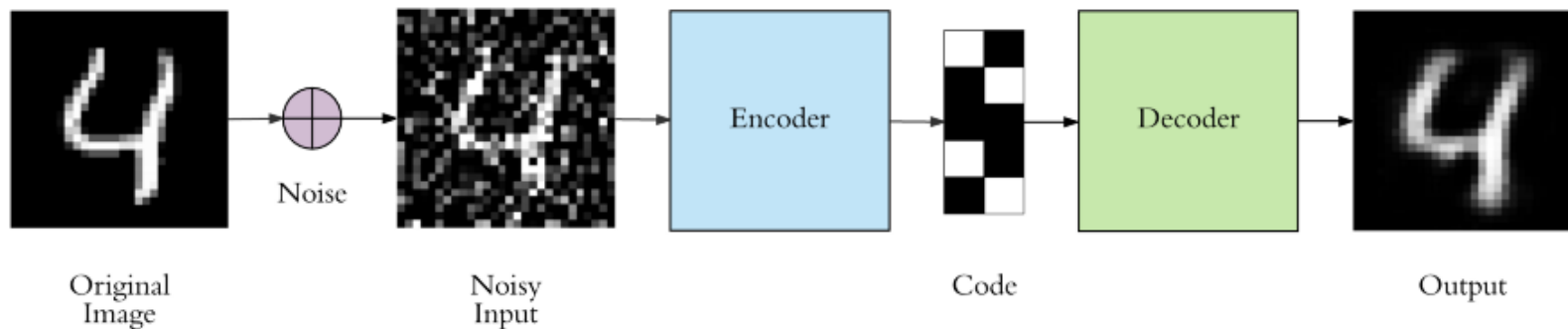
Denoising autoencoders

- Basic autoencoder trains to minimize the loss between x and the reconstruction $g(f(x))$.
- Denoising autoencoders train to minimize the loss between x and $g(f(x+w))$, where w is random noise.
- Same possible architectures, different training data.



Denoising autoencoders

- The denoising encourages the encoder to keep important information but forget about spurious information about the input.
- Then the hidden representation can be viewed as preserving useful features from the input.

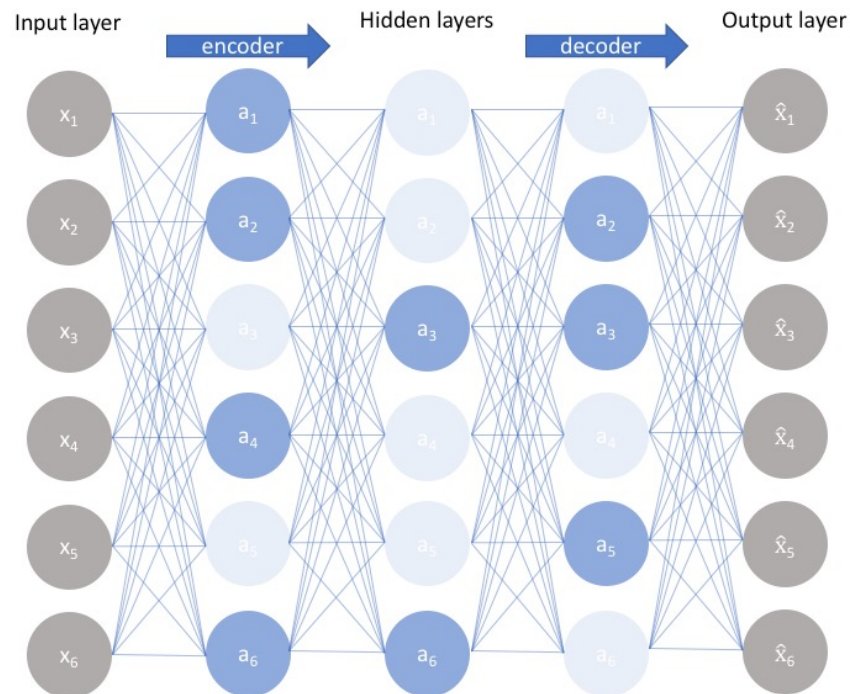


Sparse autoencoders

- Sparse autoencoders introduces an information bottleneck without requiring a reduction in the number of nodes at hidden layers.
- A sparse autoencoder is simply an autoencoder whose training criterion involves a **sparsity penalty**.
- It encourages network to learn an encoding and decoding which only relies on activating a small number of neurons. Interestingly, this is a different approach towards regularization, as we normally regularize the weights of a network, not the activations!

Sparse autoencoders

- The loss function is constructed by penalizing **activations** of hidden layers so that only a few nodes are encouraged to activate when a single sample is fed into the network.
- Specifically the loss function is constructed so that activations are penalized within a layer. The sparsity constraint can be imposed with L1 regularization or a KL divergence between expected average neuron activation to an ideal distribution p .



Sparse autoencoders

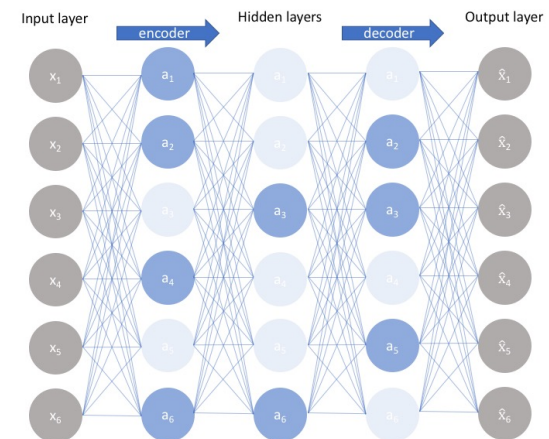
Construct a loss function to penalize *activations* the network.

L1 Regularization: Penalize the absolute value of the vector of activations a in layer h for observation l

KL divergence: Use cross-entropy between average activation and desired activation

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}|$$

$$\mathcal{L}(x, \hat{x}) + \sum_j KL(\rho || \hat{\rho}_j)$$



Learning Resources

- Charu C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- <https://www.jeremyjordan.me/autoencoders/>
- <https://www.codingninjas.com/codestudio/library/regularization-of-autoencoders>

Thank you