From AE to VAE

DSE 3121 DEEP LEARNING

Dr. Rohini Rao & Dr. Abhilash K Pai
Department of Data Science and Computer Applications
MIT Manipal

Credits: Most of the slides are adapted from CS7015 Deep Learning, Dept. of CSE, IIT Madras

Unsupervised Learning

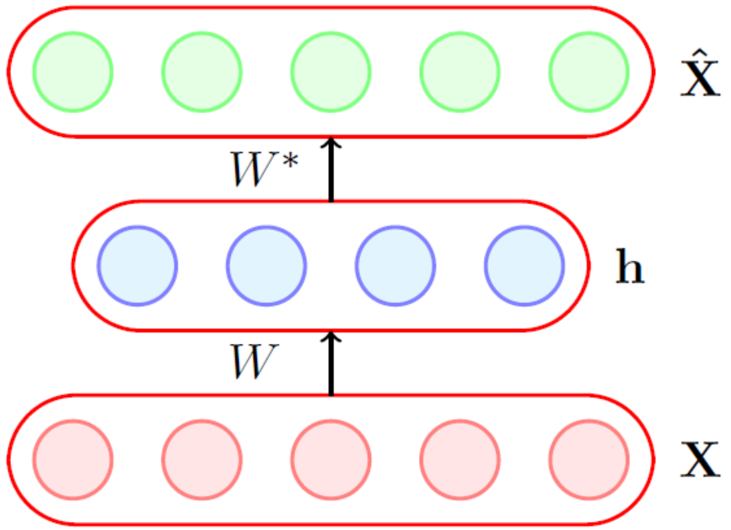
Unsupervised leaning : only use the inputs $X^{(t)}$ for learning.

- Automatically extract meaningful features/representations for the data.
- Compress data while maintaining the structure and complexity of the original dataset (dimensionality reduction).
- Leverage the availability of unlabeled data (which can be used for unsupervised pre-training).

- Some well-known neural networks for unsupervised learning are:
  - Restricted Boltzmann Machines (RBM)
  - Sparse Coding Model
  - **Autoencoders (AE)**

Autoencoders: Introduction

- Autoencoders are special type feed forward neural networks which are trained to reproduce their input at the output layer.



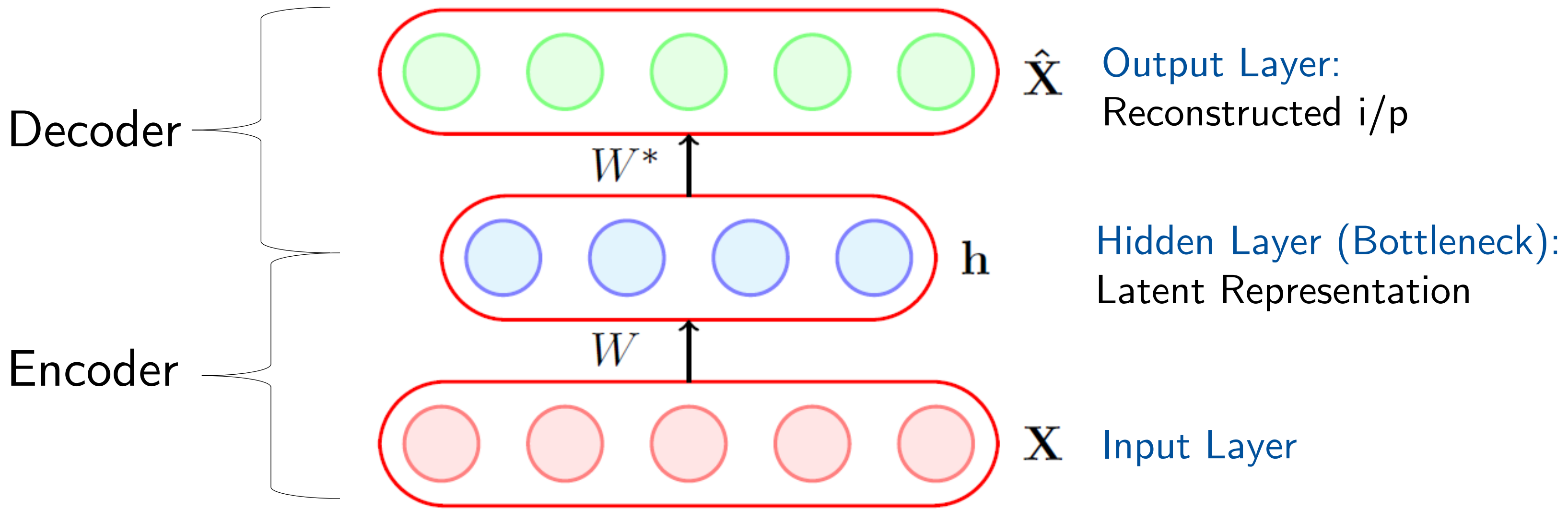
Output Layer:
Reconstructed i/p

Hidden Layer (Bottleneck):
Latent Representation

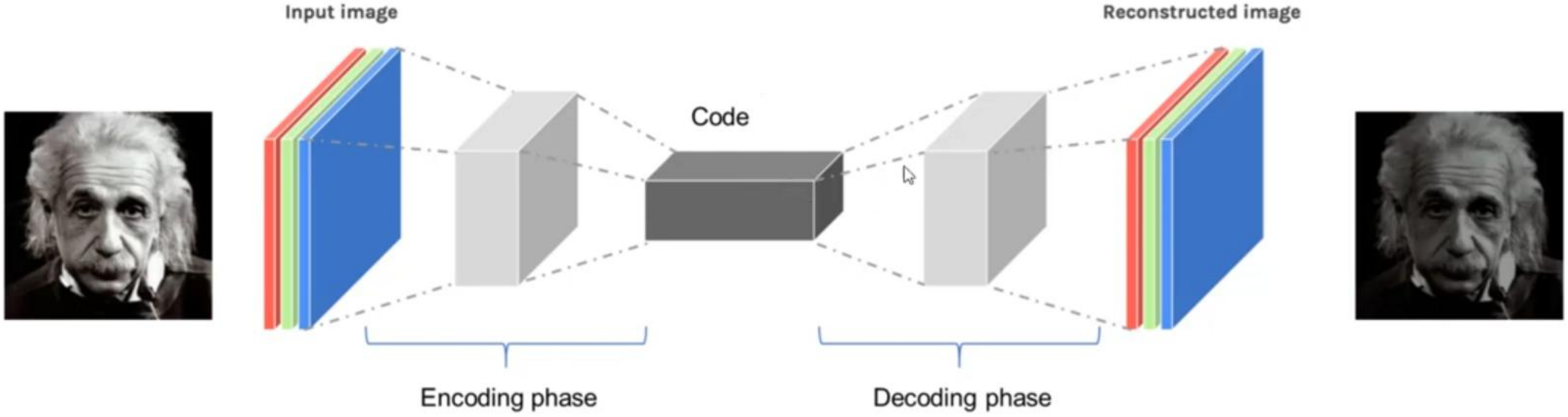
Input Layer

Autoencoders: Introduction

- It consists of an ENCODER that encodes its input X into a hidden representation h, and a DECODER which decodes the input again from this hidden representation.

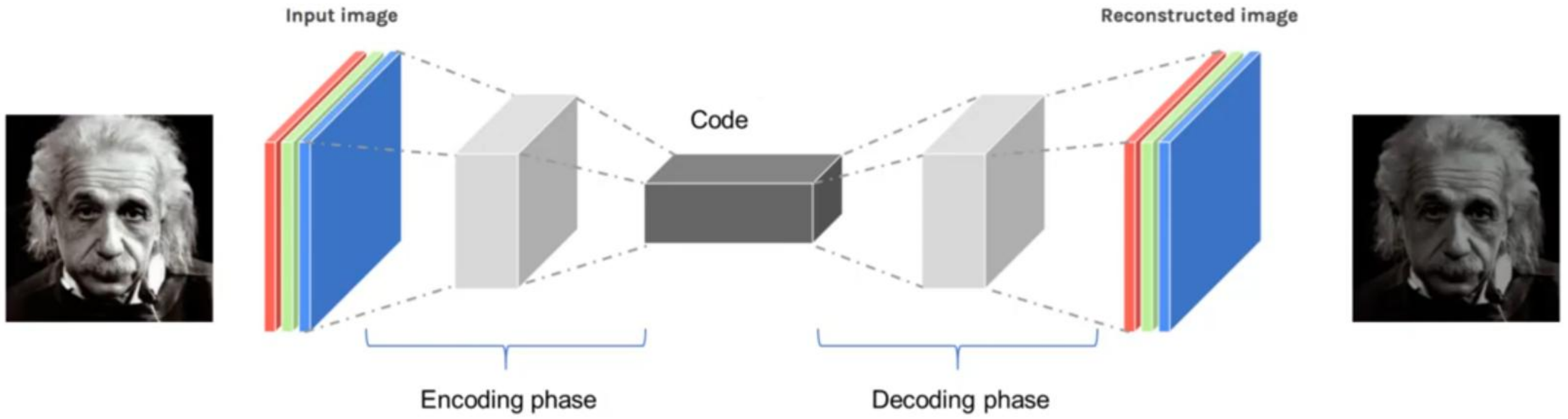


Autoencoders (AE: Introduction



Source: 85b - An introduction to autoencoders - in Python

Autoencoders: Introduction



```
model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))
```

```
model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
```

```
model.fit(x, x, epochs=100, shuffle=True)
```

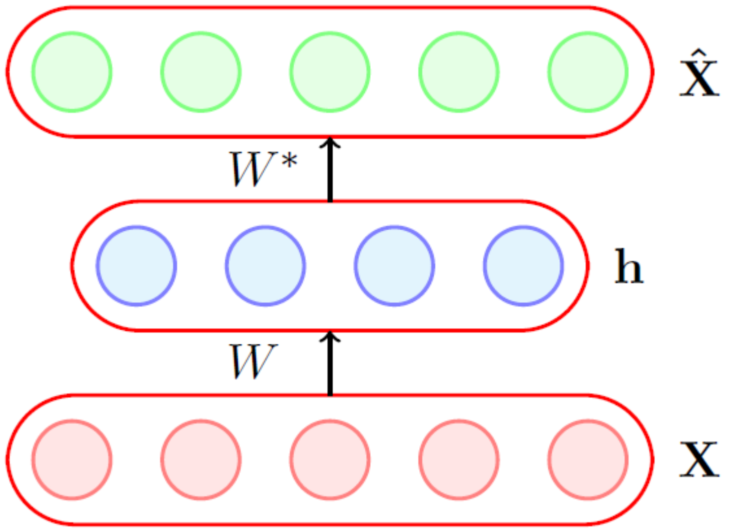
Source: 85b - An introduction to autoencoders - in Python

Autoencoders: Introduction

- The autoencoder model is trained to minimize a certain loss function which will ensure that $\mathbf{X}$ is close to $\hat{\mathbf{X}}$

$$\hat{\mathbf{X}} = f(W^*\mathbf{h} + \mathbf{c})$$

$$\mathbf{h} = g(W\mathbf{X} + \mathbf{b})$$



Output Layer: Reconstructed i/p

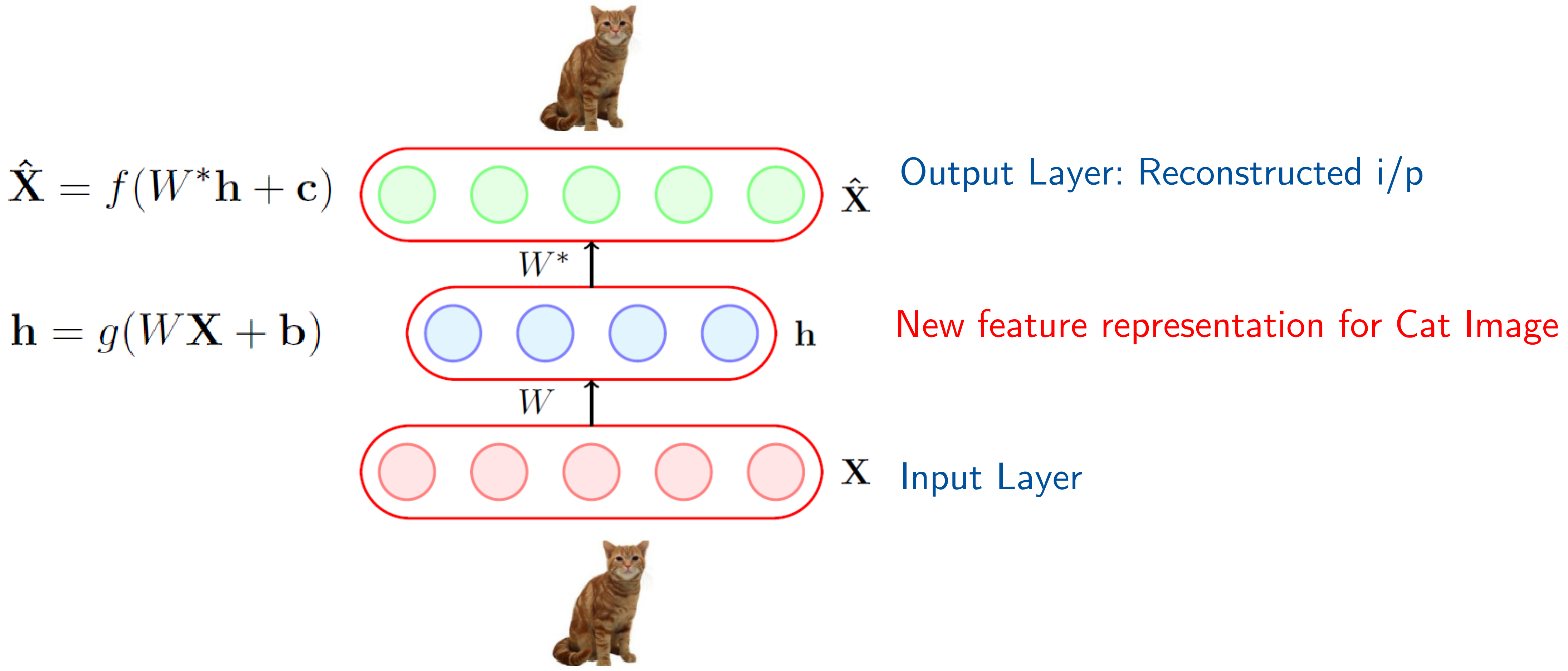
Hidden Layer: Latent Representation

Input Layer

W, W*, b, c are the parameters of the network

- As the hidden layer could produce a reduced representation of the input, autoencoders (as the one shown above) can be used for dimensionality reduction.

Autoencoders: Introduction

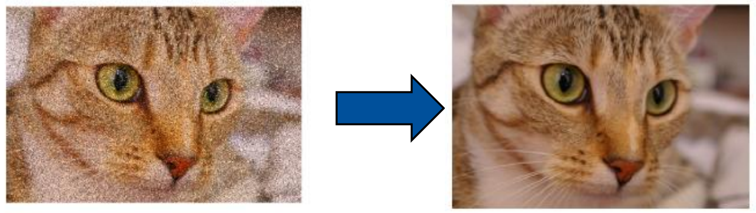


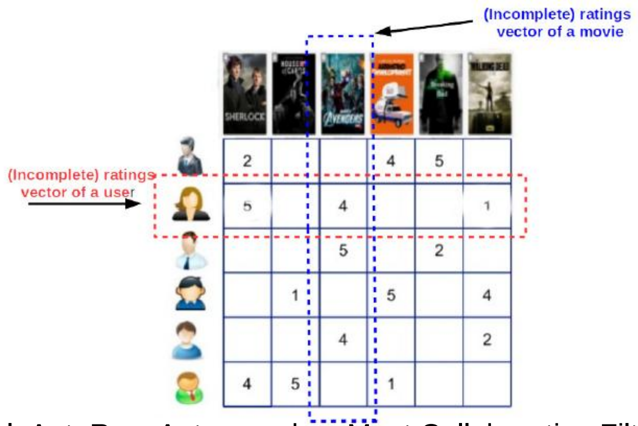
$\hat{\mathbf{X}} = f(W^*\mathbf{h} + \mathbf{c})$
$\hat{\mathbf{X}}$
Output Layer: Reconstructed i/p
W^*
$\mathbf{h} = g(W\mathbf{X} + \mathbf{b})$
$\mathbf{h}$
New feature representation for Cat Image
W
$\mathbf{X}$
Input Layer

- Also, the latent representation can be used as a new feature representation of the input X.

Autoencoders: Applications

- Feature Learning and Dimensionality reduction
- Pretraining of Deep Neural Networks
- Denoising and Inpainting
- Generate Images
- Anomaly Detection
- Recommender Systems (e.g. matrix completion)




(Incomplete) ratings vector of a movie
(Incomplete) ratings vector of a user
SHERLOCK
AVENGERS

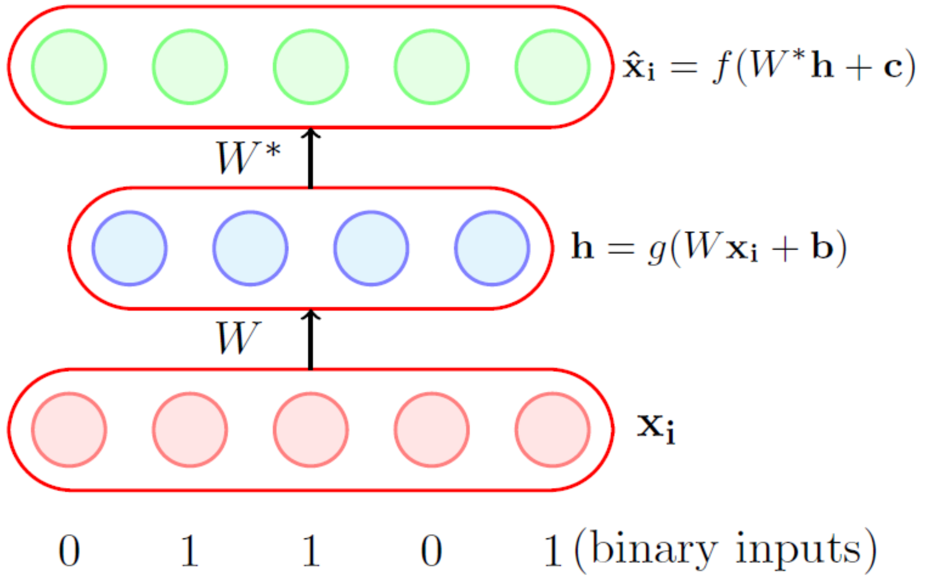
Sedhain et al, AutoRec: Autoencoders Meet Collaborative Filtering, WWW 2015

Choice of Loss and Activation functions

- For **binary inputs** (each $X_{ij} \in \{0,1\}$):
  - The activation function "f" (o/p layer) is chosen as **Sigmoid**.
  - The loss function is L([W,W*,c,b]) is **(Cross Entropy):**

$$\min\{-\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1-x_{ij})\log(1-\hat{x}_{ij}))\}$$

  - The activation function "g" is typically chosen as Sigmoid.

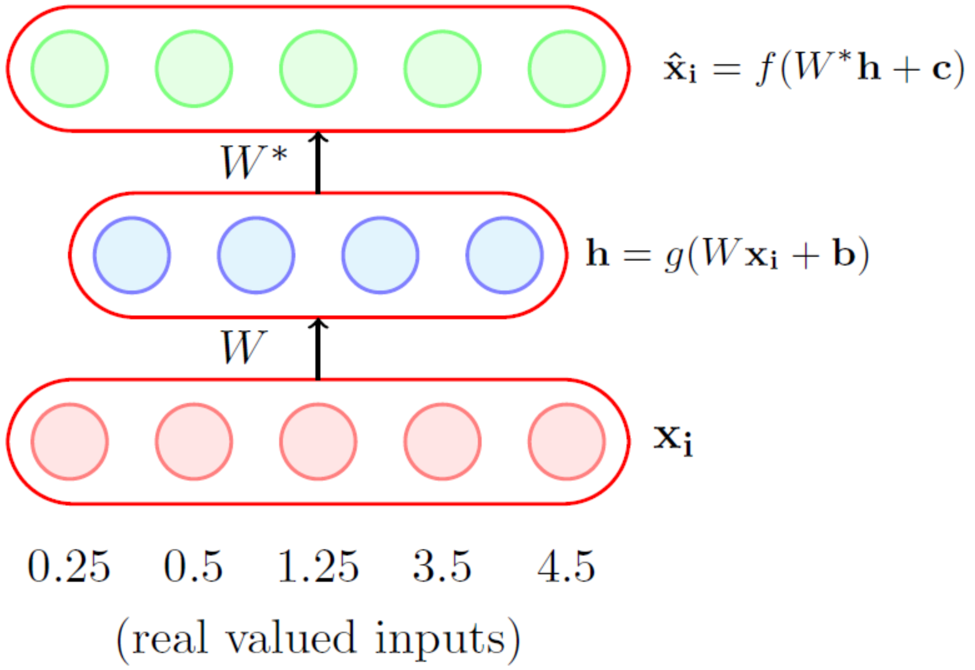

$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
$W*$
$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$
W
$\mathbf{x}_\mathbf{i}$
0 1 1 0 1 (binary inputs)

Choice of Loss and Activation functions

- For **real valued inputs** (each $X_{ij} \in R$):
  - The activation function "f" is chosen as **Linear**.
  - The loss function is L([W,W*,c,b]) is **(SSD):**

$$\min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2$$

  - The activation function "g" is typically chosen as Sigmoid.

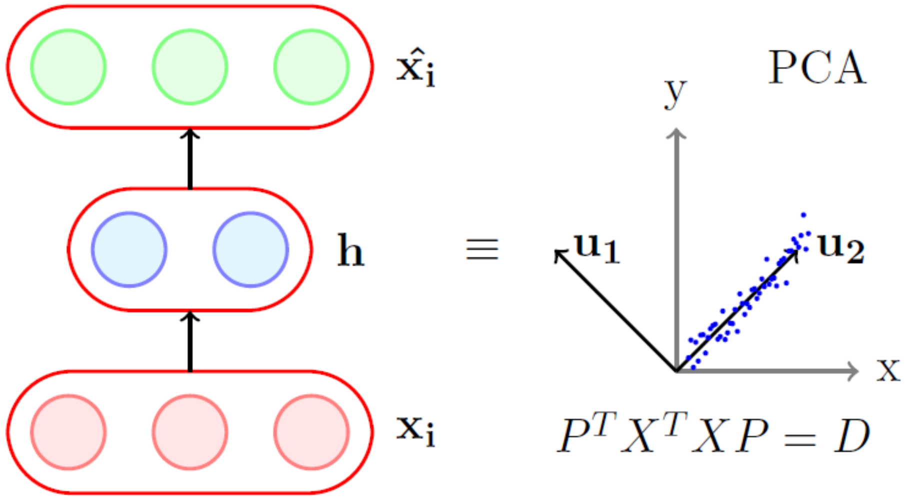


$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
$W*$
$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$
W
$\mathbf{x}_\mathbf{i}$
0.25
0.5
1.25
3.5
4.5
(real valued inputs)

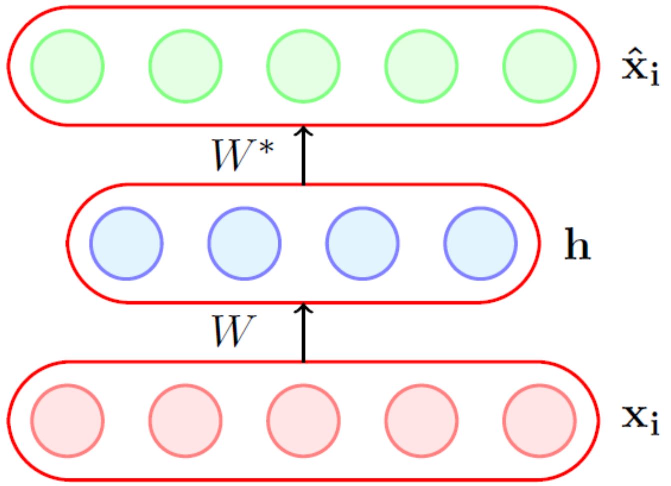
Autoencoders and PCA

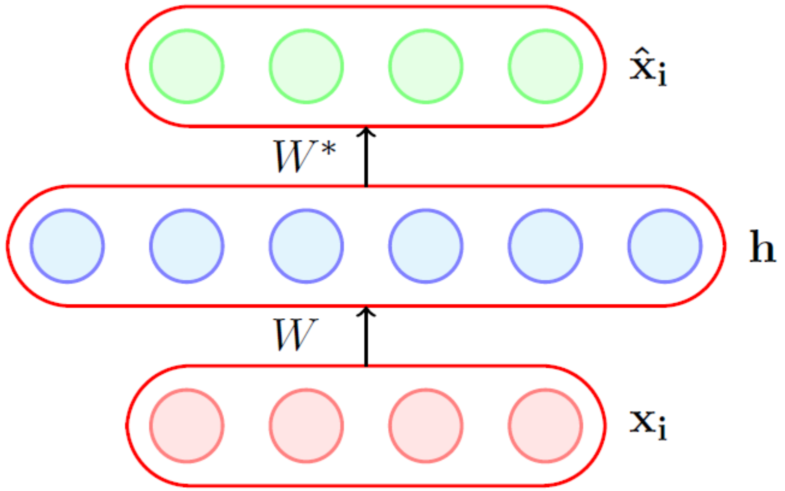
- An autoencoder's **encoder part** is **equivalent to PCA** if we:
  - Use a Linear Encoder.
  - Use a Linear Decoder.
  - Use a Squared Error Loss function.
  - Normalize the inputs to:

$$\hat{x}_{ij} = \frac{1}{\sqrt{m}} \left(x_{ij} - \frac{1}{m} \sum_{k=1}^{m} x_{kj} \right)$$



Undercomplete and Overcomplete Autoencoders





Undercomplete AE
- Dimension of h is less than dimension x

Overcomplete AE
- Dimension of h is greater than dimension x

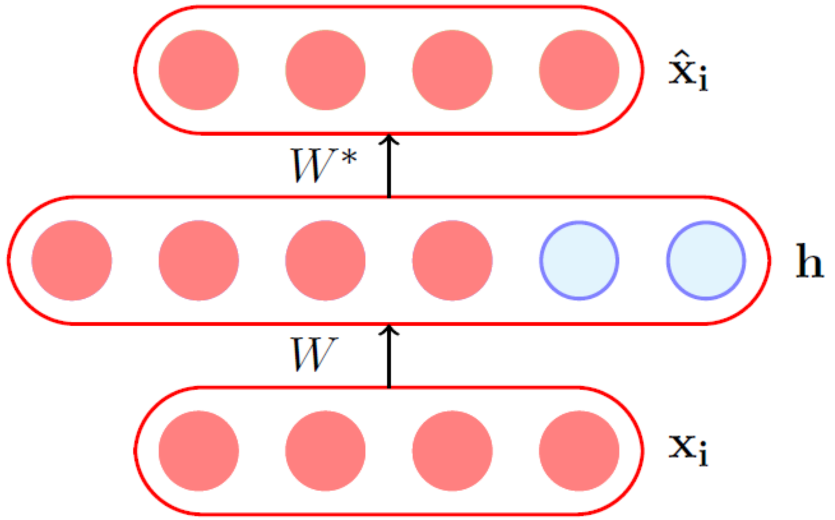
Regularization in Autoencoders

- An overcomplete autoencoder would learn to copy the inputs x_i to hidden representation h_i and then copy h_i to $\hat{x}_i$ (learn an identity function).

- Such an identity encoding is useless in practice as it does not really tell us anything about the important characteristics of the data. This is a case of overfitting!

- Overfitting (Poor generalization) could happen even in undercomplete autoencoders.

- To avoid poor generalization, we need to introduce regularization.



Regularization in Autoencoders

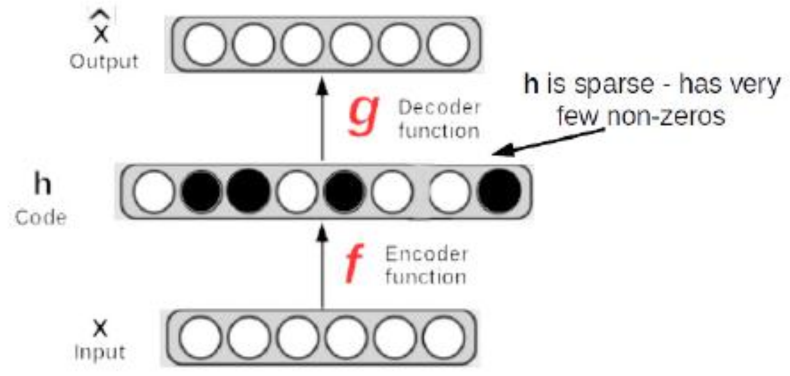
- The simplest solution is to add an L_2 regularization term to the objective function:

$$\min_{\theta, w, w^*, \mathbf{b}, \mathbf{c}} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2 + \boxed{\lambda ||\theta||^2}$$

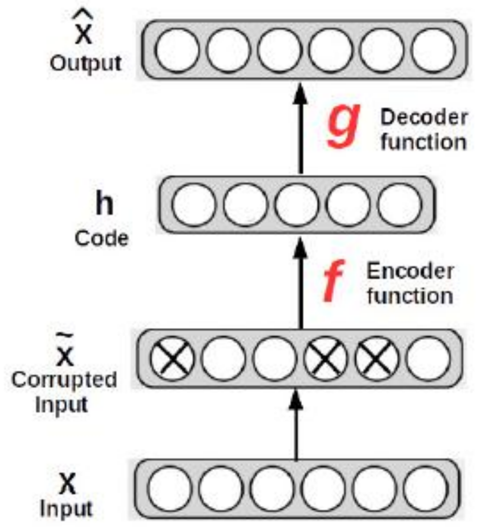
- Another trick is to tie the weights of encoder and decoder ($W^* = W^T$). This effectively reduces the number of parameters and acts as a regularizer.

Regularized Autoencoders

- Several ways to regularize the model, e.g.
  - Make the learned code sparse (Sparse Autoencoders)
  - Make the model robust against noisy/incomplete inputs (Denoising Dutoencoders)



Sparse Autoencoder



Denoising Autoencoder

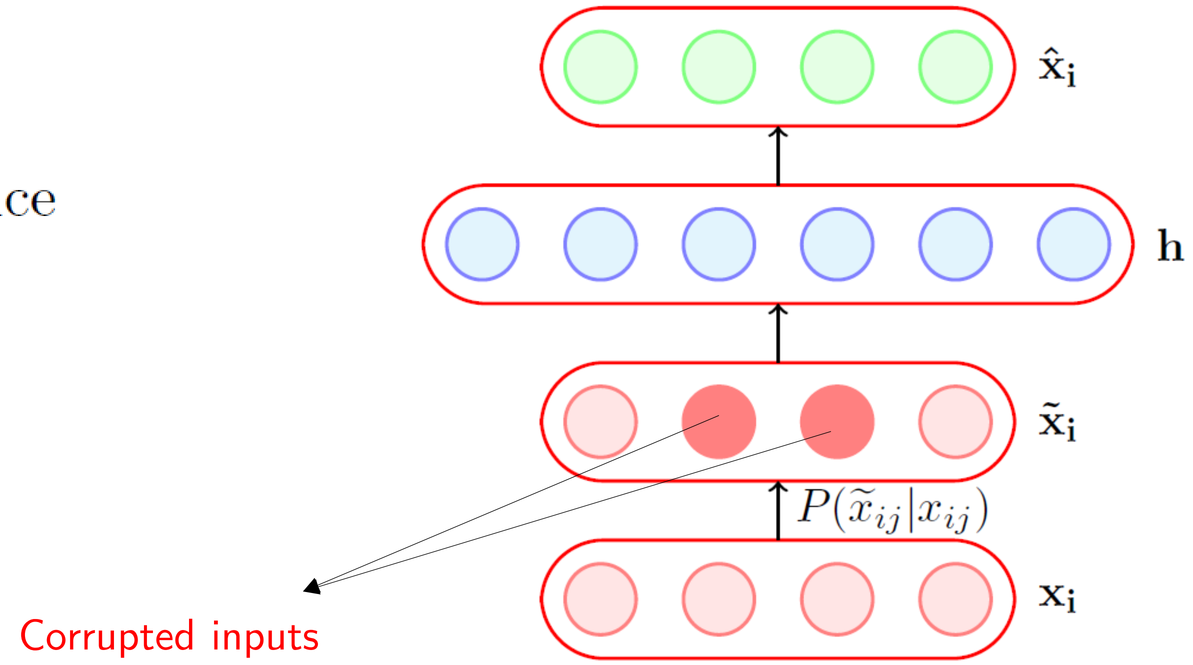
  - Make the model robust against small changes in the input (Contractive Autoencoders)

Denoising Autoencoders

- A denoising encoder simply corrupts the input data using a probabilistic process $(P(\widetilde{x}_{ij}|x_{ij}))$ before feeding it to the network.

A simple $P(\widetilde{x}_{ij}|x_{ij})$ used in practice is the following

$$P(\widetilde{x}_{ij} = 0|x_{ij}) = q$$
$$P(\widetilde{x}_{ij} = x_{ij}|x_{ij}) = 1 - q$$



- In other words, with probability q the input is flipped to 0 and with probability (1 - q) it is retained as it is.

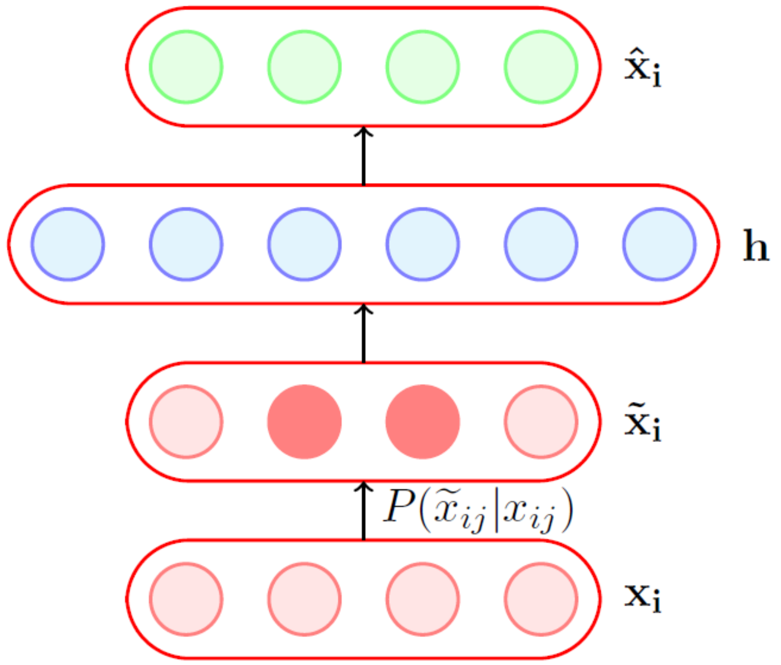
Denoising Autoencoders

- How does this help?
  - This helps because the objective is still to reconstruct the original (un-corrupted) x_i :

$$\arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2$$

  - It no longer makes sense for the model learn to copy corrupted input to h (the objective function will not be minimized by doing so)

  - Instead, the model will now have to capture the characteristics of the data correctly.

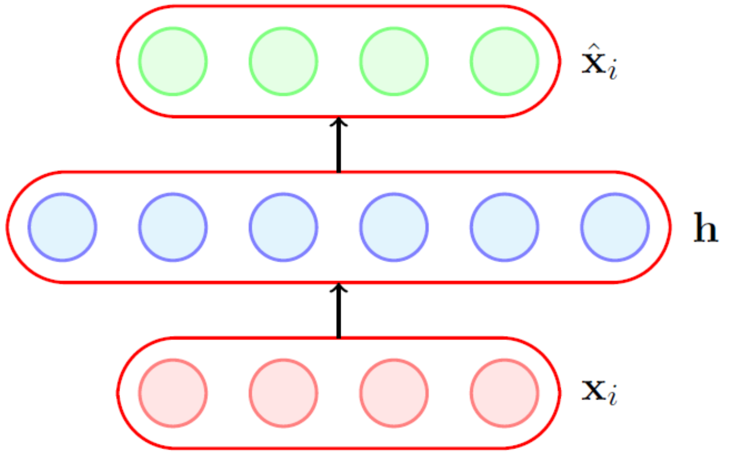


Sparse Autoencoders

- A hidden neuron with sigmoid activation will have values between 0 and 1.
- We say that the neuron is activated when its output is close to 1 and not activated when its output is close to 0.
- A sparse autoencoder tries to ensure the neuron is inactive most of the times.

The average value of the activation of a neuron l is given by

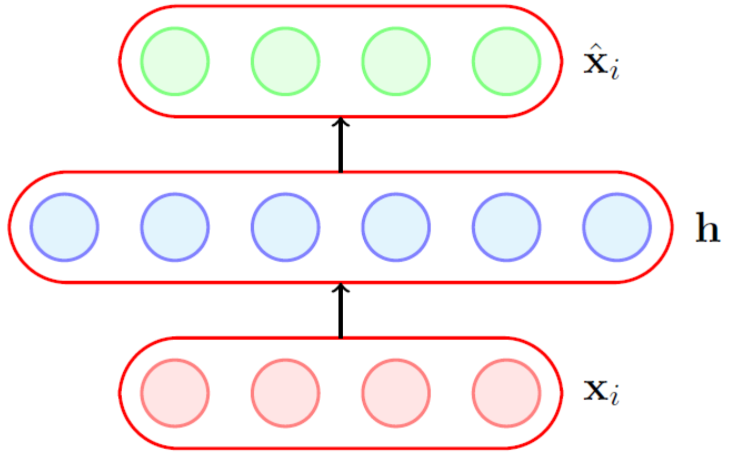
$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i)_l$$



Sparse Autoencoders

- If the neuron l is sparse (i.e. mostly inactive) then $\hat{\rho}_l \rightarrow 0$
- A sparse autoencoder uses a sparsity parameter ρ (typically very close to 0, say, 0.005) and tries to enforce the constraint $\hat{\rho}_l = \rho$
- One way of ensuring this is to add the following term to the objective function

$$\Omega(\theta) = \sum_{l=1}^{k} \rho \log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_l}$$



The average value of the activation of a neuron l is given by

$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i)_l$$

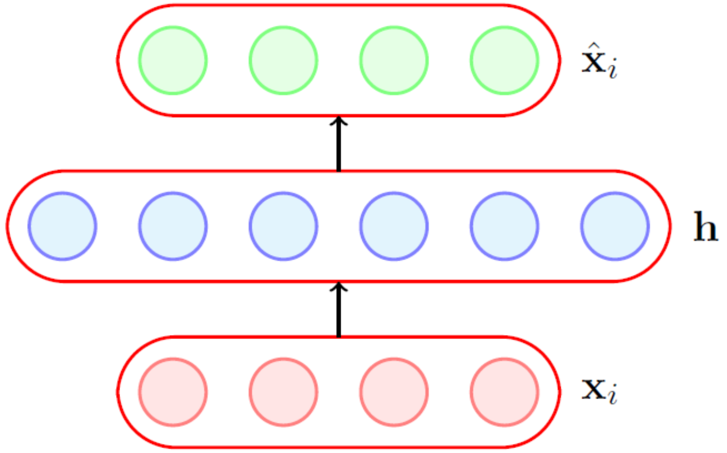
Sparse Autoencoders

- Now, the equation for the loss function will look like:

$$L'(\Theta) = L(\Theta) + \Omega(\Theta)$$

$$\Omega(\theta) = \sum_{l=1}^{k} \rho \log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_l}$$

- When will this term ($\Omega(\Theta)$) reach its minimum value and what is the minimum value?



The average value of the activation of a neuron l is given by

$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i)_l$$

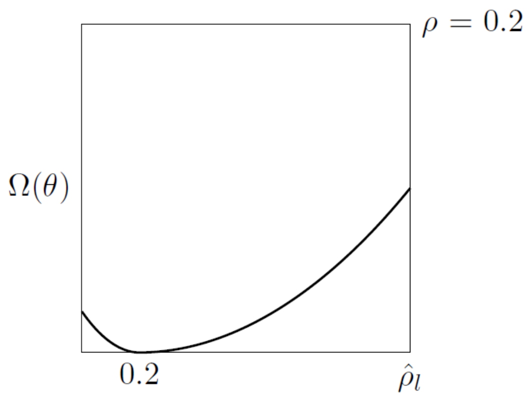
Sparse Autoencoders

- Now, the equation for the loss function will look like:

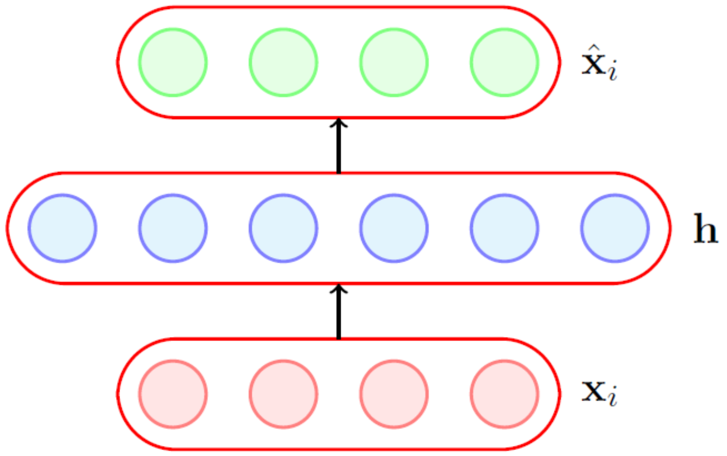
$$L'(\Theta) = L(\Theta) + \Omega(\Theta) \longrightarrow \text{Sparsity constraint}$$

$$\Omega(\theta) = \sum_{l=1}^{k} \rho \log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_l}$$

- When will this term ($\Omega(\Theta)$) reach its minimum value and what is the minimum value?



$\Omega(\Theta)$ will reach min. value when $\hat{\rho}_l = \rho$



The average value of the activation of a neuron l is given by

$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i)_l$$

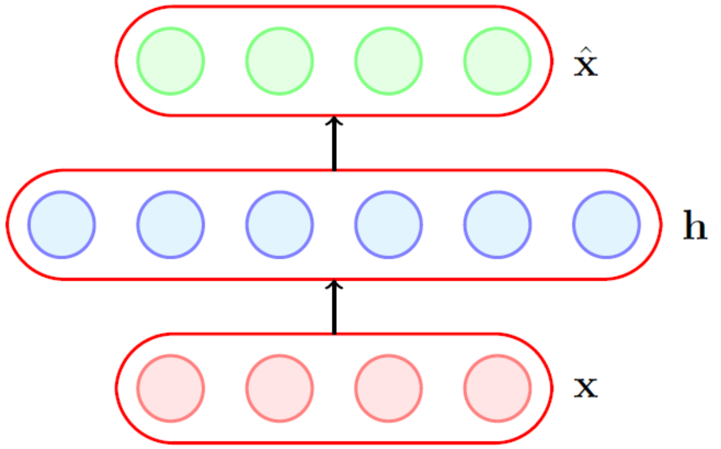
Contractive Autoencoders

- A contractive autoencoder also tries to prevent an overcomplete autoencoder from learning the identity function.
- It does so by adding the following regularization term to the loss function:

$$\Omega(\theta) = \underbrace{\|J_{\mathbf{x}}(\mathbf{h})\|_F^2}$$

Frobenius Norm

Jacobian of the encoder



Variation in output of 2nd neuron in the hidden layer with a small variation in the 1st input.

$$J_{\mathbf{x}}(\mathbf{h}) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & & \ddots & & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_k}{\partial x_n} \end{bmatrix}$$

$$\|J_{\mathbf{x}}(\mathbf{h})\|_F^2 = \sum_{j=1}^{n} \sum_{l=1}^{k} \left(\frac{\partial h_l}{\partial x_j} \right)^2$$

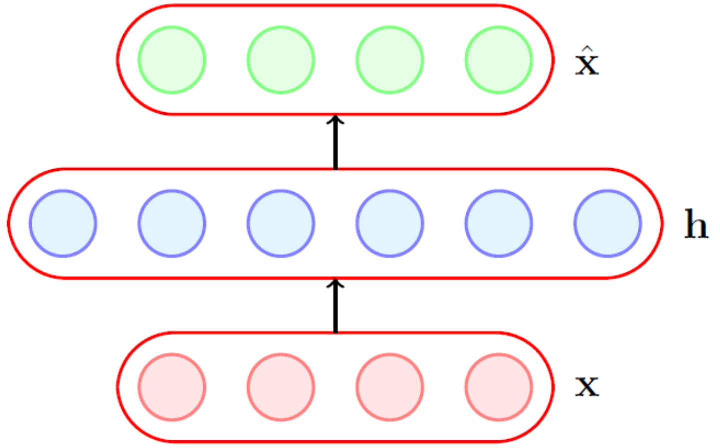
Contractive Autoencoders

- What is the intuition behind this ?
- Consider $\frac{\partial h_1}{\partial x_1}$, what does it mean if $\frac{\partial h_1}{\partial x_1} = 0$
- It means that this neuron is not very sensitive to variations in the input x_1.

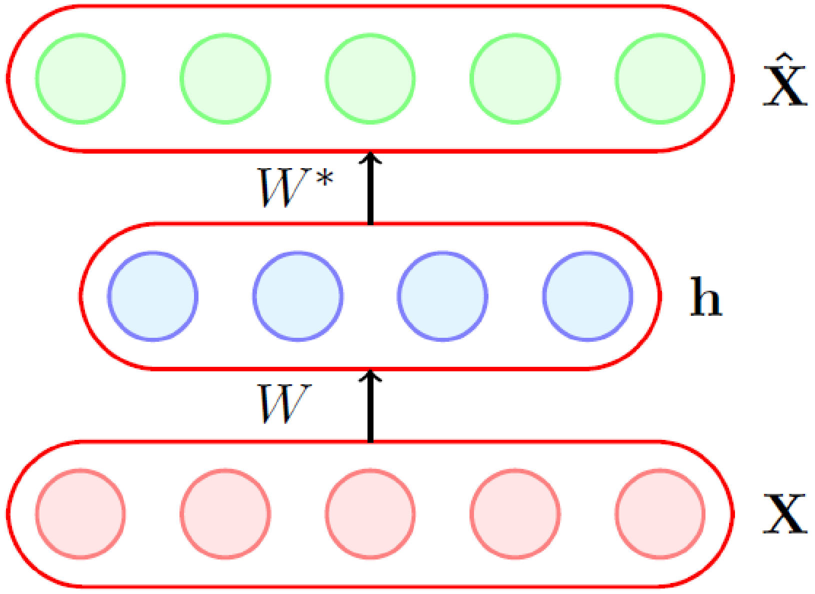
$$\hat{\mathscr{L}}(\theta) = \mathscr{L}(\theta) + \Omega(\theta)$$

- $\mathcal{L}(\theta)$ - capture important variations in data
- $\Omega(\theta)$ - do not capture variations in data
- Tradeoff - capture only very important variations in the data

$$\|J_{\mathbf{x}}(\mathbf{h})\|_F^2 = \sum_{j=1}^{n}\sum_{l=1}^{k}\left(\frac{\partial h_l}{\partial x_j}\right)^2$$

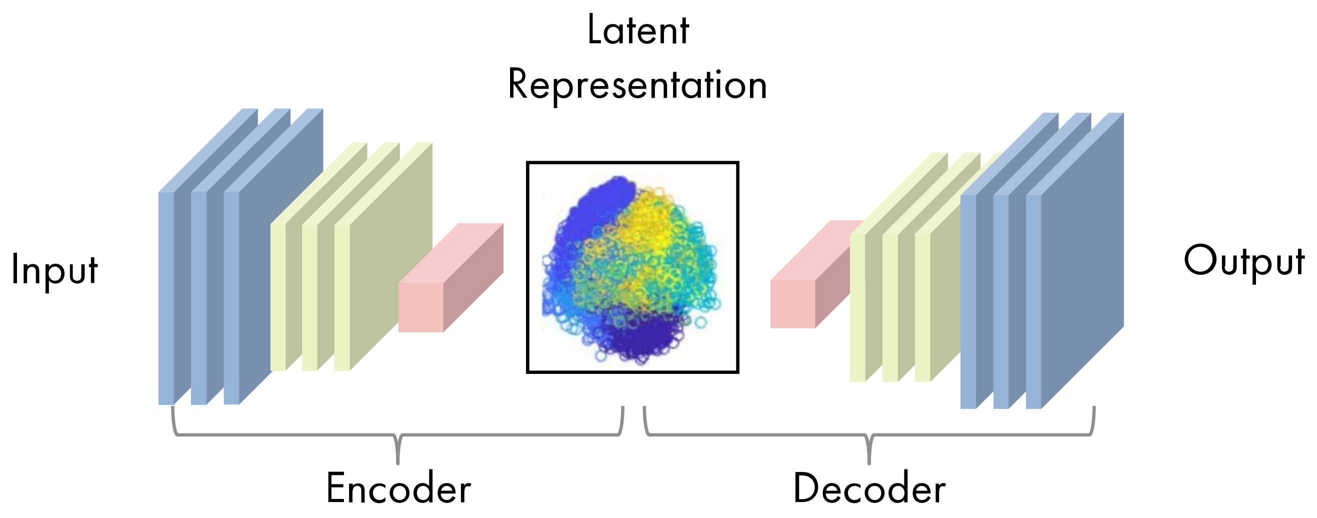


Generative Models: Introduction

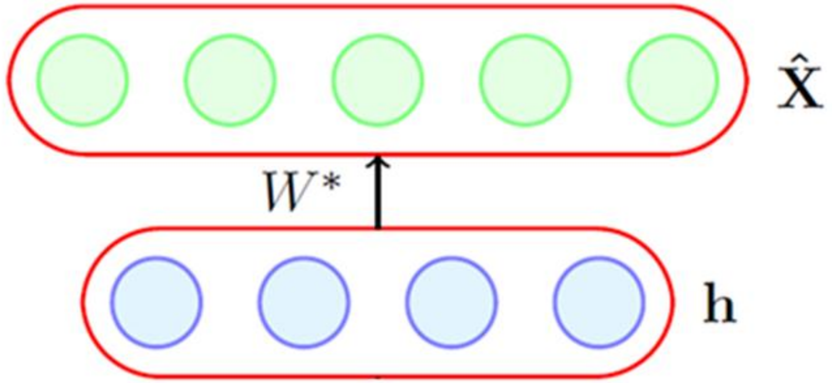


$$\mathbf{h} = g(W\mathbf{X} + \mathbf{b})$$
$$\hat{\mathbf{X}} = f(W^*\mathbf{h} + \mathbf{c})$$

- An autoencoder contains an encoder which takes the input $\boldsymbol{x}$ and maps it to a hidden representation.
- The decoder then takes this hidden representation and tries to reconstruct the input from it as $\hat{\boldsymbol{x}}$.
- The hidden layer produces a good abstraction of the input, in the form of h.

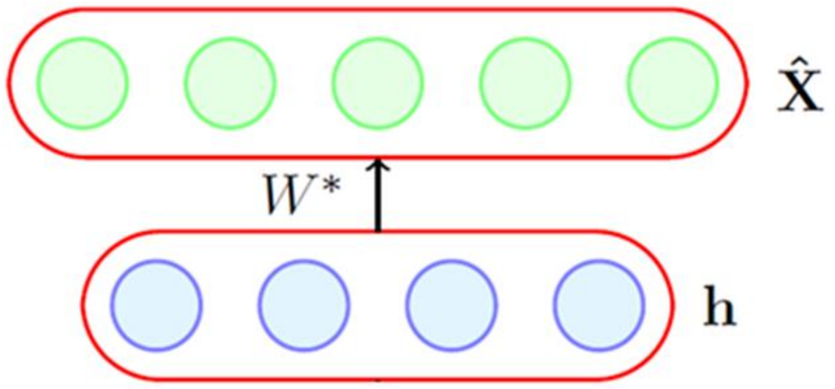


Generative Models: Introduction



- An autoencoder contains an encoder which takes the input $\boldsymbol{x}$ and maps it to a hidden representation.
- The decoder then takes this hidden representation and tries to reconstruct the input from it as $\hat{\boldsymbol{x}}$.
- The hidden layer produces a good abstraction of the input, in the form of h.
- Now, once the autoencoder is trained can we remove the encoder, feed a hidden representation h to the decoder and decode a $\hat{\boldsymbol{x}}$ from it ?
- In other words, can we do generation with autoencoders?

Generative Models: Introduction



- Remember classic autoencoders are deterministic i.e., for an X, the encoder will give the same hidden representation every time.

- In principle, yes! But in practice there is a problem with this approach.
- h is a very high dimensional vector and only a few vectors in this space would actually correspond to meaningful latent representations of our input.
- So, of all the possible value of h which values should I feed to the decoder?
- Ideally, we should feed only those values of h which are highly *likely*.
- In other words, we are interested in sampling from a distribution over h's (P(h|X)) so that we pick only those h's which have high probability.
- Let's understand this concept clearly with an example.

Generative Models: Introduction

Example: Movie Review

M1: An unexpected and necessary masterpiece

M2: Delightfully merged information and comedy

M3: Director's first true masterpiece

M4: Sci- perfection, truly mesmerizing film

M5: Waste of time and money

M6: Best Lame Historical Movie Ever

- Consider a movie critic who writes reviews for movies
- For simplicity let us assume that he always writes reviews containing a maximum of 5 words
- Further, let us assume that there are a total of 50 words in his vocabulary
- Each of the 5 words in his review can be treated as a **random variable** which takes one of the 50 values

Generative Models: Introduction

Example: Movie Review

M1: An unexpected and necessary masterpiece

M2: Delightfully merged information and comedy

M3: Director's first true masterpiece

M4: Sci- perfection, truly mesmerizing film

M5: Waste of time and money

M6: Best Lame Historical Movie Ever

- Given many such reviews written by the reviewer we could learn the joint probability distribution

$$P(X_1, X_2, \ldots, X_5)$$

- Consider a movie critic who writes reviews for movies
- For simplicity let us assume that he always writes reviews containing a maximum of 5 words
- Further, let us assume that there are a total of 50 words in his vocabulary
- Each of the 5 words in his review can be treated as a **random variable** which takes one of the 50 values

Generative Models: Introduction

Example: Movie Review

M1: An unexpected and necessary masterpiece

M2: Delightfully merged information and comedy

M3: Director's first true masterpiece

M4: Sci- perfection, truly mesmerizing film

M5: Waste of time and money

M6: Best Lame Historical Movie Ever

- We can think of a very simple factorization of this model by assuming that the i th word only depends on the previous 2 words and not anything before that:

$$P(X_1, X_2, \ldots, X_5) = P(X_1) \prod_{i=2:5} P(X_i | X_{i-1}, X_{i-2})$$

- Let us consider one such factor:

$$P(X_i\text{=time} \mid X_{i\text{-}2}\text{=waste}, X_{i\text{-}1}\text{=of})$$

- This could be estimated as:

$$\frac{\text{count (waste of time)}}{\text{count (waste of)}}$$

- And the two counts mentioned above can be computed by going over all the reviews

Generative Models: Introduction

- What can we do with this joint distribution?

Joint distribution

w	$P(X_i = w \mid, X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid, X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	...
than	0.61	0.01	0.20	...
as	0.12	0.10	0.16	...
for	0.14	0.09	0.05	...
real	0.01	0.50	0.01	...
the	0.02	0.12	0.12	...
life	0.05	0.11	0.33	...

Generative Models: Introduction

M7: More realistic than real life

w	$P(X_i = w\|, X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w\|, X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w\| X_{i-2} = than, X_{i-1} = real)$	$\dots$
than	0.61	0.01	0.20	$\dots$
as	0.12	0.10	0.16	$\dots$
for	0.14	0.09	0.05	$\dots$
real	0.01	0.50	0.01	$\dots$
the	0.02	0.12	0.12	$\dots$
life	0.05	0.11	0.33	$\dots$

- What can we do with this joint distribution?
- Given a **review**, **classify** if this was written by the reviewer

$$\begin{aligned}P(M7) = P(X_1 = more).&P(X_2 = realistic|X_1 = more).\\ &P(X_3 = than|X_1 = more, X_2 = realistic).\\ &P(X_4 = real|X_2 = realistic, X_3 = than).\\ &P(X_5 = life|X_3 = than, X_4 = real)\\ &= 0.2 \times 0.25 \times 0.61 \times 0.50 \times 0.33 = 0.005\end{aligned}$$

Generative Models: Introduction

M7: More realistic than real life

w	$P(X_i = w \mid, X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid, X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	$\ldots$
than	0.61	0.01	0.20	$\ldots$
as	0.12	0.10	0.16	$\ldots$
for	0.14	0.09	0.05	$\ldots$
real	0.01	0.50	0.01	$\ldots$
the	0.02	0.12	0.12	$\ldots$
life	0.05	0.11	0.33	$\ldots$

- What can we do with this joint distribution?
- Given a review, classify if this was written by the reviewer
- **What else can we do?**

Generative Models: Introduction

M7: More realistic than real life

w	$P(X_i = w \mid, X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid, X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	...
than	0.61	0.01	0.20	...
as	0.12	0.10	0.16	...
for	0.14	0.09	0.05	...
real	0.01	0.50	0.01	...
the	0.02	0.12	0.12	...
life	0.05	0.11	0.33	...

- What can we do with this joint distribution?
- Given a review, classify if this was written by the reviewer
- What else can we do?
- **Generate** new reviews which would look like reviews written by this reviewer

Generative Models: Introduction

M7: More realistic than real life

w	$P(X_i = w \mid, X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid, X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	...
than	0.61	0.01	0.20	...
as	0.12	0.10	0.16	...
for	0.14	0.09	0.05	...
real	0.01	0.50	0.01	...
the	0.02	0.12	0.12	...
life	0.05	0.11	0.33	...

- What can we do with this joint distribution?
- Given a review, classify if this was written by the reviewer
- What else can we do?
- **Generate** new reviews which would look like reviews written by this reviewer
- **How to do this?**

Generative Models: Introduction

w	$P(X_1 = w)$			
the	0.62			
movie	0.10			
amazing	0.01			
useless	0.01			
was	0.01			
⋮	⋮			

- How does the reviewer start his reviews (what is the first word that he chooses)?

Generative Models: Introduction

w	$P(X_1 = w)$			
the	0.62			
movie	0.10			
amazing	0.01			
useless	0.01			
was	0.01			
⋮	⋮			

The

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review

Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$		
the	0.62	0.01		
movie	0.10	0.40		
amazing	0.01	0.22		
useless	0.01	0.20		
was	0.01	0.00		
⋮	⋮	⋮		

The movie

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?

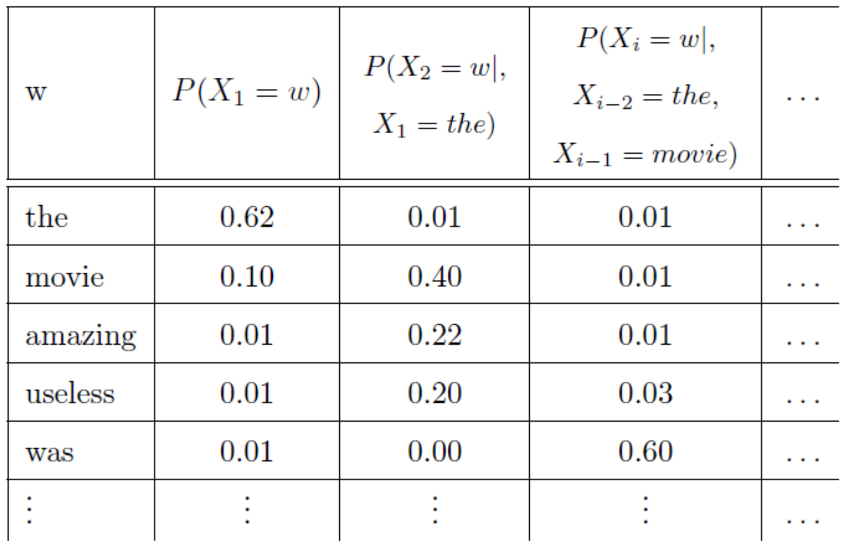
Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$
the	0.62	0.01	0.01
movie	0.10	0.40	0.01
amazing	0.01	0.22	0.01
useless	0.01	0.20	0.03
was	0.01	0.00	0.60
⋮	⋮	⋮	⋮

The movie was

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?
- Having selected first two words what is the most likely third word that the reviewer uses?

Generative Models: Introduction



w	$P(X_1 = w)$	$P(X_2 = w\|, X_1 = the)$	$P(X_i = w\|, X_{i-2} = the, X_{i-1} = movie)$	$\dots$
the	0.62	0.01	0.01	$\dots$
movie	0.10	0.40	0.01	$\dots$
amazing	0.01	0.22	0.01	$\dots$
useless	0.01	0.20	0.03	$\dots$
was	0.01	0.00	0.60	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$

The movie was really amazing

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?
- Having selected first two words what is the most likely third word that the reviewer uses?
- And so on ...

Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$	$\dots$
the	0.62	0.01	0.01	$\dots$
movie	0.10	0.40	0.01	$\dots$
amazing	0.01	0.22	0.01	$\dots$
useless	0.01	0.20	0.03	$\dots$
was	0.01	0.00	0.60	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$

The movie was really amazing

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?
- Having selected first two words what is the most likely third word that the reviewer uses?
- And so on ...
- But, if we select the most likely word at each time step, then it will give us the same review again and again

Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \vert, X_1 = the)$	$P(X_i = w \vert, X_{i-2} = the, X_{i-1} = movie)$	...
the	0.62	0.01	0.01	...
movie	0.10	0.40	0.01	...
amazing	0.01	0.22	0.01	...
useless	0.01	0.20	0.03	...
was	0.01	0.00	0.60	...
⋮	⋮	⋮	⋮	...

The movie was really amazing

We should instead **sample** from this distribution!

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?
- Having selected first two words what is the most likely third word that the reviewer uses?
- And so on ...
- But, if we select the most likely word at each time step, then it will give us the same review again and again

Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$	$\ldots$
the	0.62	0.01	0.01	$\ldots$
movie	0.10	0.40	0.01	$\ldots$
amazing	0.01	0.22	0.01	$\ldots$
useless	0.01	0.20	0.03	$\ldots$
was	0.01	0.00	0.60	$\ldots$
is	0.01	0.00	0.30	$\ldots$
masterpiece	0.01	0.11	0.01	$\ldots$
I	0.21	0.00	0.01	$\ldots$
liked	0.01	0.01	0.01	$\ldots$
decent	0.01	0.02	0.01	$\ldots$

- Suppose there are 10 words in our vocabulary, and we have computed the joint probability distribution over all the random variables.
- Now, consider that we need to generate the 3rd word in the review, given the first two words of the review.
- There are 10 possibilities :

Index	Word
0	the
1	movie
2	amazing
3	useless
4	was
5	is
6	masterpiece
7	I
8	liked
9	decent

Generative Models: Introduction

w	$P(X_1 = w)$	$P(X_2 = w \vert, X_1 = the)$	$P(X_i = w \vert, X_{i-2} = the, X_{i-1} = movie)$	$\dots$
the	0.62	0.01	0.01	$\dots$
movie	0.10	0.40	0.01	$\dots$
amazing	0.01	0.22	0.01	$\dots$
useless	0.01	0.20	0.03	$\dots$
was	0.01	0.00	0.60	$\dots$
is	0.01	0.00	0.30	$\dots$
masterpiece	0.01	0.11	0.01	$\dots$
I	0.21	0.00	0.01	$\dots$
liked	0.01	0.01	0.01	$\dots$
decent	0.01	0.02	0.01	$\dots$

- Suppose there are 10 words in our vocabulary, and we have computed the joint probability distribution over all the random variables.
- Now, consider that we need to generate the 3rd word in the review, given the first two words of the review.
- There are 10 possibilities :
- We can think of this as a 10-sided dice where each side correspond to a word and has a certain probability to show up.

Index	Word	$P(X_i = w \vert, X_{i-2} = the, X_{i-1} = movie)$	.
0	the	0.01	.
1	movie	0.01	.
2	amazing	0.01	.
3	useless	0.03	.
4	was	0.60	.
5	is	0.30	.
6	masterpiece	0.01	.
7	I	0.01	.
8	liked	0.01	.
9	decent	0.01	.

Generative Models: Introduction

Generated Reviews

- the movie is liked decent
- I liked the amazing movie
- the movie is masterpiece

Such models which tries to estimate the probability P(X) from a large number of samples are called as generative models

- Suppose there are 10 words in our vocabulary, and we have computed the joint probability distribution over all the random variables.

  Now, consider that we need to generate the 3rd word in the review, given the first two words of the review.

  There are 10 possibilities :

  We can think of this as a 10-sided dice where each side correspond to a word and has a certain probability to show up.

Index	Word	$P(X_i = w\|, X_{i-2} = the, X_{i-1} = movie)$	.
0	the	0.01	.
1	movie	0.01	.
2	amazing	0.01	.
3	useless	0.03	.
4	was	0.60	.
5	is	0.30	.
6	masterpiece	0.01	.
7	I	0.01	.
8	liked	0.01	.
9	decent	0.01	.

- Roll this dice and pick the side (word) which comes up.
- Every run will now give a different review.

Generative Models: Introduction

What about images?

- For 32x32 images we want to learn: **$P(V_1, V_2, \ldots\ldots V_{1024})$** where V_i is a random variable corresponding to each pixel, which could possibly have values from 0-255.
- We could factorize this joint distribution by assuming that each pixel is dependent on its neighboring pixels.

Generative Models: Introduction

Input Image

Output Images

- Again, what can we do with joint distribution **$P(V_1, V_2, \ldots\ldots V_{1024})$** ?
- Apart from classifying and generating (as discussed for previous language modelling in prev. slides), we can also **correct noisy inputs** (here, images) or help in **completing incomplete images.**

Generative Models: The concept of latent variables

- Latent variables : Blue sky, Green grass, White clouds

- There are certain underlying hidden (latent) characteristics which are determining the pixels and their interactions.
- We could think of these as additional (latent) random variables in our distribution
- And the interactions between the visible pixels are captured through the latent variables.

Generative Models: Introduction

Latent Variable = daytime

Latent Variable = night

Latent Variable = cloudy

- Based on this notion, we would now like learn **the joint distribution P(V,H)** where, $V = \{V_1, V_2 .., V_{\#pixels}\}$ is observed variable and $H = \{H_1, H_2 .., H_{\#latent\ features}\}$ is hidden variables.

- Using this we can find: $P(H|V) = \frac{P(V,H)}{\sum_H P(V,H)}$

- That is, given an image, we can find the most likely latent configuration (H = h) that generated this image, where h captures a latent (abstract) representation (imp. properties of the image)

Generative Models: Introduction

- Under this abstraction, all these images would look very similar (i.e., they would have very similar latent configurations h)
- Even though in the original feature space (pixels) there is a significant difference between these images, in the latent space they would be very close to each other
- Once again, assume that we are able to learn the joint distribution P(V,H)
- Using this distribution, we can find $P(V|H) = \dfrac{P(V,H)}{\sum_V P(V,H)}$
- By using this, we can now say "**Create an image which is cloudy, has a beach and depicts daytime**"
- In other words, I can now generate images given certain latent variables

Variational Autoencoders (VAE)

yt video - > https://www.youtube.com/watch?v=2m9E-aSXtl8

IMP

Figure: Abstraction

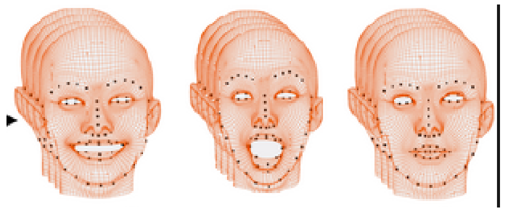
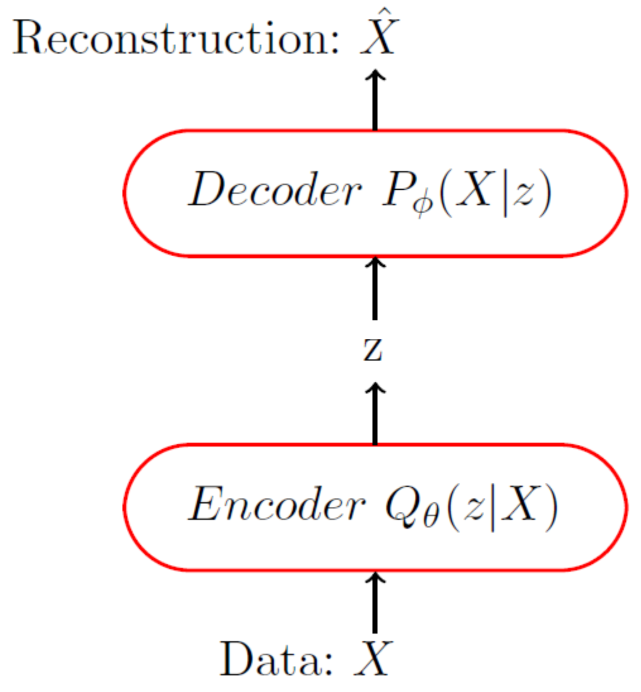


Figure: Generation

- Let $\{X = x_i\}_{i=1}^{N}$ be the training data
- We can think of X as a random variable in R^n
- For example, X could be an image and the dimensions of X correspond to pixels of the image
- We are interested in learning an abstraction (i.e., given an X find the hidden representation z)
- We are also interested in generation (*i.e.*, given a hidden representation generate an X)
- In probabilistic terms we are interested in $P(z|X)$ and $P(X|z)$ (to be consistent with the literation on VAEs we will use z instead of H and X instead of V)

Variational Autoencoders (VAE)



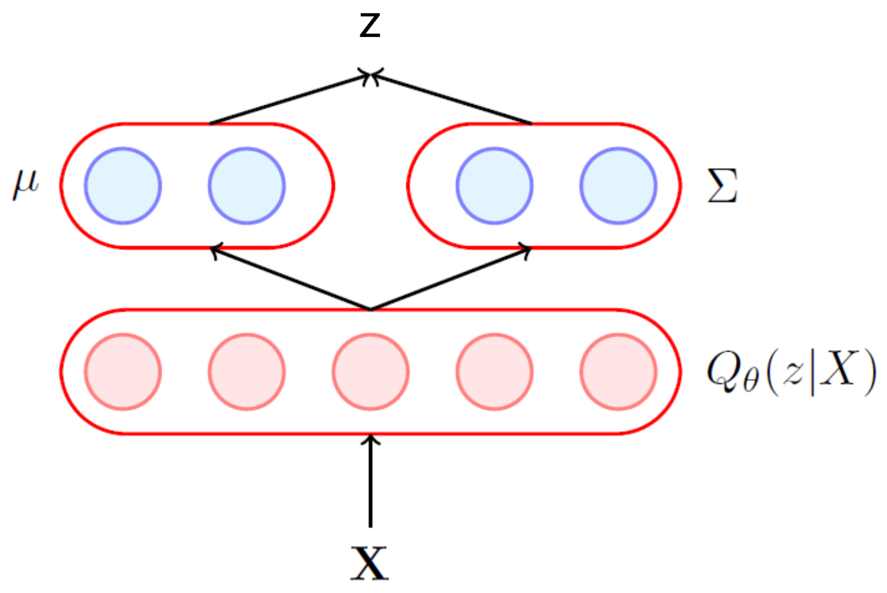
θ: the parameters of the encoder neural network
ϕ: the parameters of the decoder neural network

- **Encoder Goal:** Learn a distribution over the latent variables (Q(z | X))
- **Decoder Goal:** Learn a distribution over the visible variables (P(X | z))

- The training data is mapped to a latent space using a Neural network where the latent space posterior distribution (Q(z|X)) and prior distribution (Q(z)) are modelled as Gaussian and the output of this network is two parameters – mean and covariance (parameters of the posterior distribution)

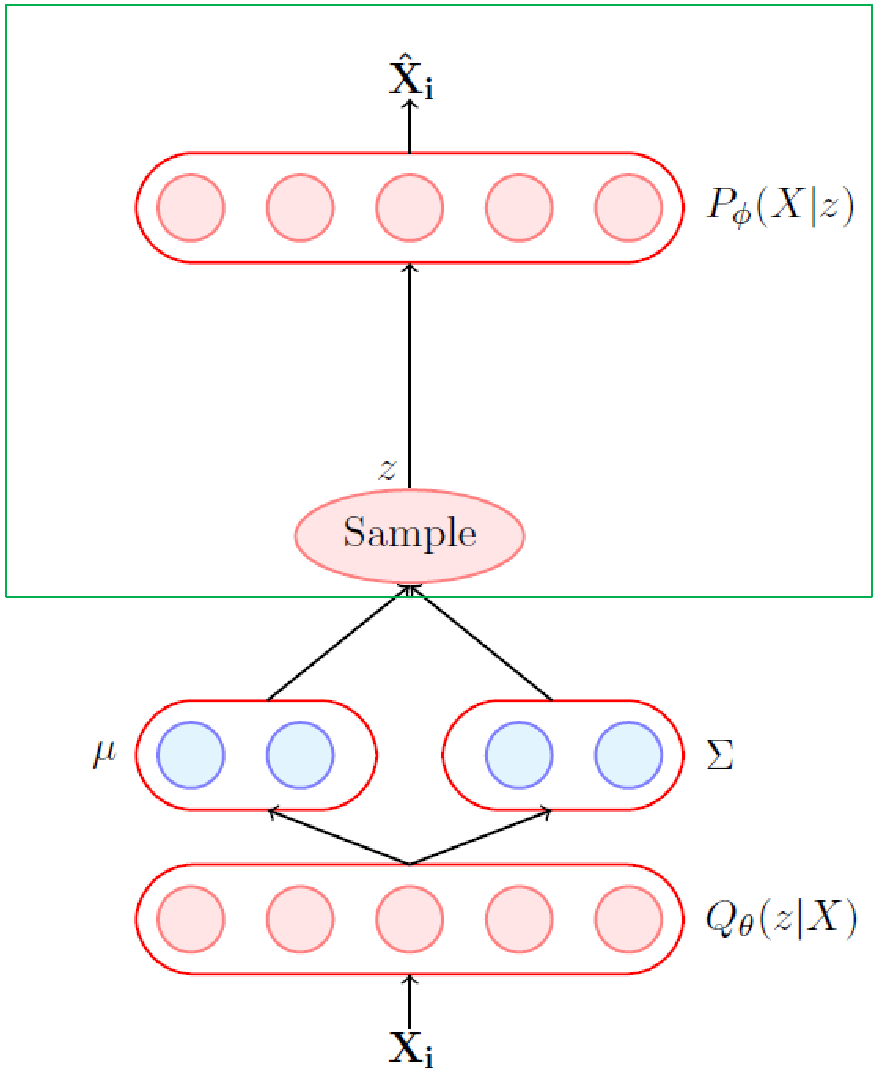
- The latent space vector is mapped to input image using another neural network.

Variational Autoencoders: Encoder



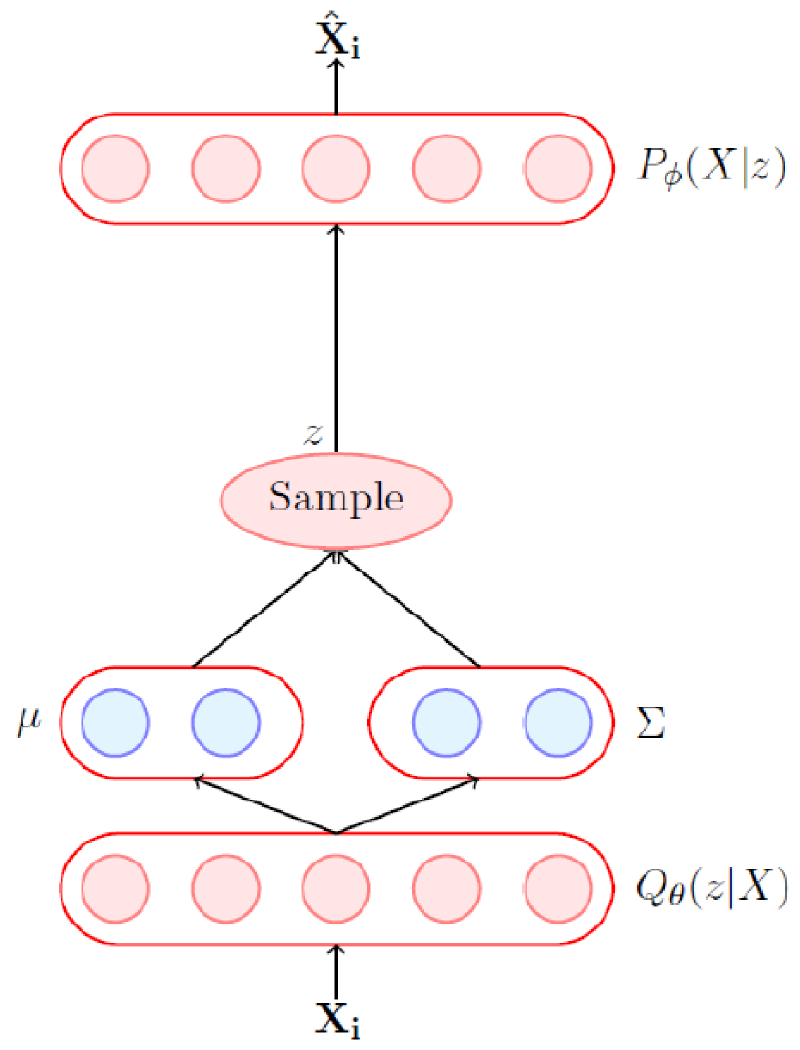
In VAEs we assume that the latent variables come from a standard normal distribution $\mathcal{N}(0, I)$ and the job of the encoder is to then predict the parameters of this distribution

Variational Autoencoders: Decoder



- The job of the decoder is to predict a probability distribution over X : $P(X|z)$
- Once again we will assume a certain form for this distribution
- For example, if we want to predict 28 x 28 pixels and each pixel belongs to $\mathbb{R}$ (*i.e.*, $X \in \mathbb{R}^{784}$) then what would be a suitable family for $P(X|z)$?
- We could assume that $P(X|z)$ is a Gaussian distribution with unit variance
- The job of the decoder f would then be to predict the mean of this distribution as $f_\phi(z)$

Variational Autoencoders



- The following is the total loss function (summed over all the data points):

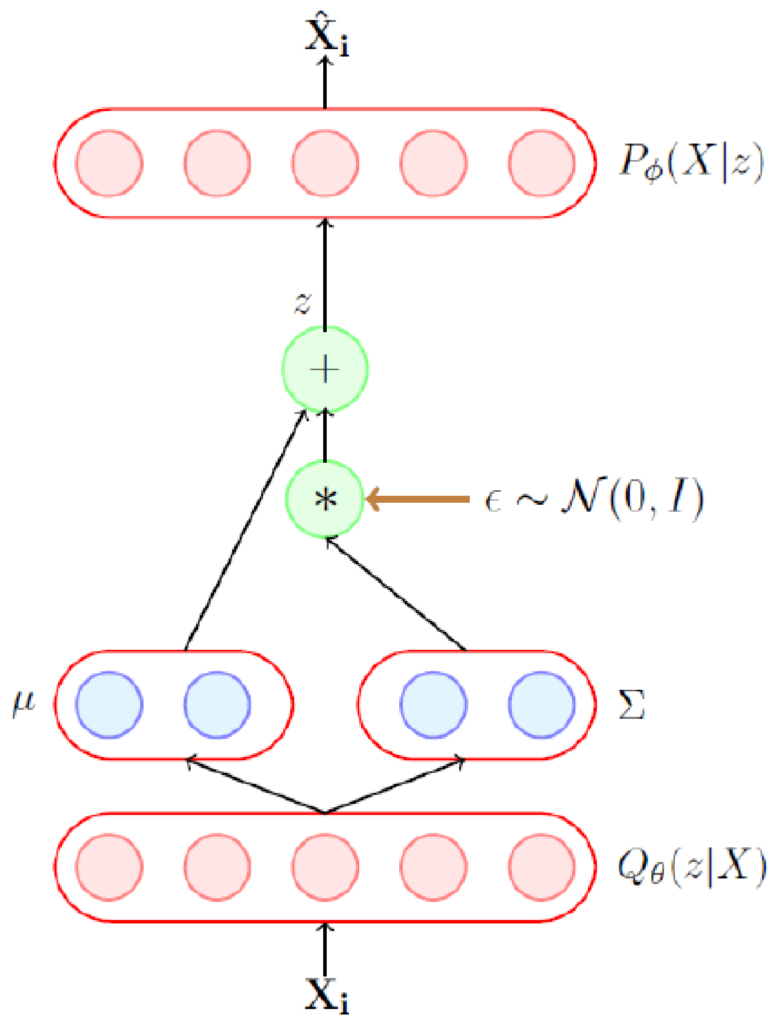
$$\mathscr{L}(\theta) = \sum_{i=1}^{m} l_i(\theta,\phi) = -\mathbb{E}_{z\sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

KL-divergence → captures the difference between the two distributions

This term acts as a regularizer by forcing the encoder to produce latent representations that look like samples from a standard Gaussian distribution.

Evidence Lower Bound

Variational Autoencoders



- The latent space in VAEs is typically modeled as a Gaussian distribution, and sampling from this distribution is a non-differentiable operation. This makes it difficult to compute gradients with respect to the model parameters.
- VAEs use a **reparameterization trick** to tackle the problem where we move the process of sampling to the input layer.

For 1 dimensional case, given μ and σ we can sample from $\mathcal{N}(\mu, \sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0, 1)$, and then computing

$$z = \mu + \sigma * \epsilon$$