Autoencoders

Yulia Newton, Ph.D.

CS156, Introduction to Artificial Intelligence

San Jose State University

Spring 2021

What are autoencoders?

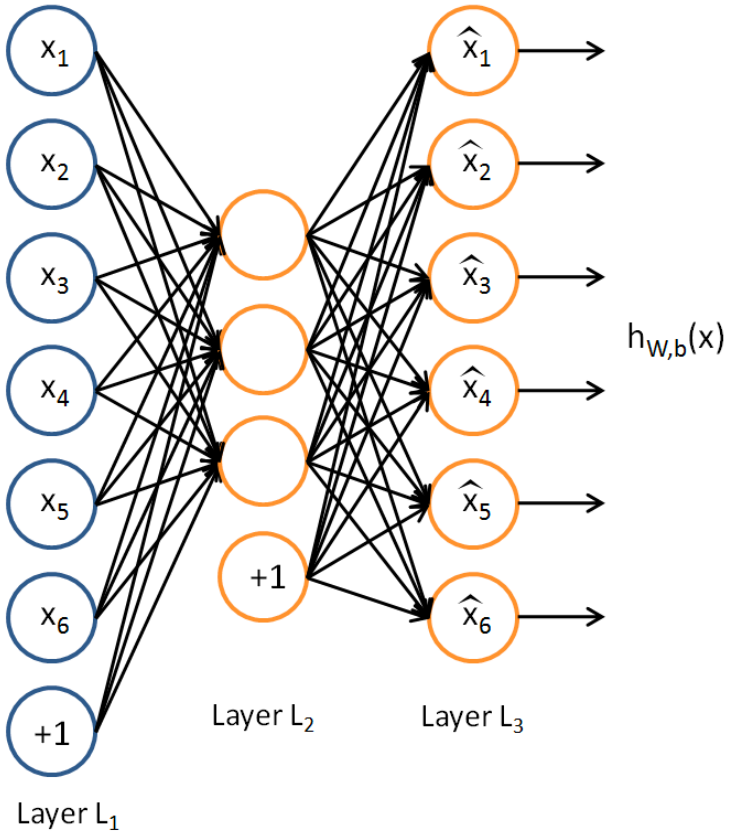
- Unsupervised learning technique for representation learning
- Autoencoder is a special type of artificial neural network
- Automatically learns the features that best represent the data by introducing a “bottleneck” layer
  - “Representation” is often called “encoding”
  - Reduction layer(s) followed by reconstruction layer(s)
- Have been shown to “learn” how to ignore the noise in the data

How do autoencoders work?

- Let’s say we have some unlabeled input data and we want to learn the transformation of the data best approximating the identity function
- We can do this by building an artificial neural network in which the input and the output layers have the same dimensions and the output label is the input value
  - Find the transformation, applicable to all training data, which best approximates the input
  - Best estimator of the data approximation to itself

Simple autoencoder

h(x) is the function we learn by training this ANN

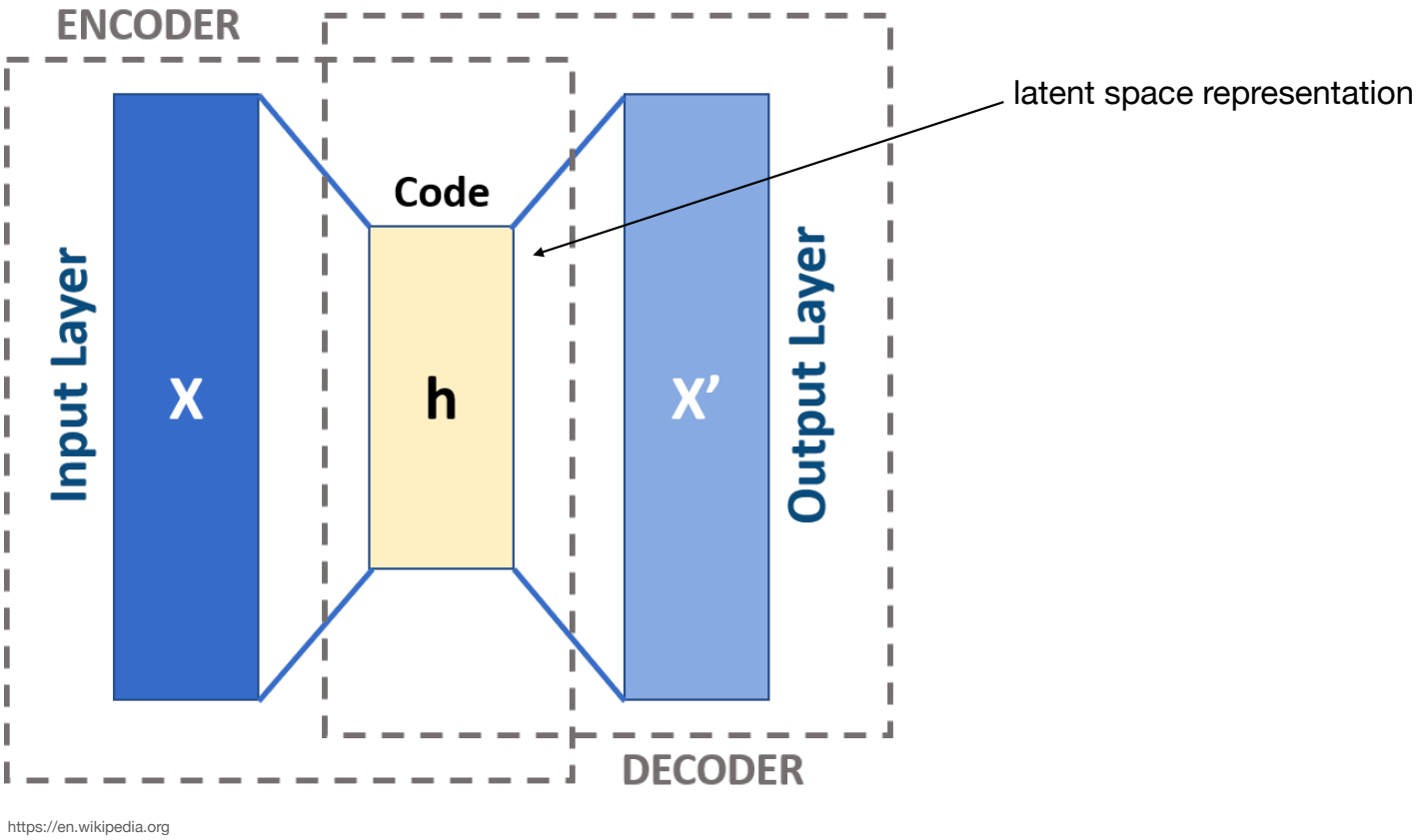


http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders

Why do we need autoencoders?

- Detecting irregularities/noise/outliers in the data
- Learn interesting patterns about our data
- Learning compressed representation of the data (latent data representation)
  - The compression/bottleneck/latent layer is usually smaller than the input and the output layers
    - Although sometimes the hidden layer is larger and that can also help finding interesting patterns in the data
  - Compressed representation often finds highly correlated features and reduces feature space to independent components
    - Similar to PCA and other dimensionality reduction techniques
- Dimensionality reduction
  - We are interested in the hidden layer of the autoencoder
- Detect amount of randomness in the input data
  - An autoencoder will have hard time learning on a completely random data, so when we compare the input and the output they will look nothing alike
- Data denoising
- Anomaly detection problems

Interplay between encoding-decoding



Reconstruction layer

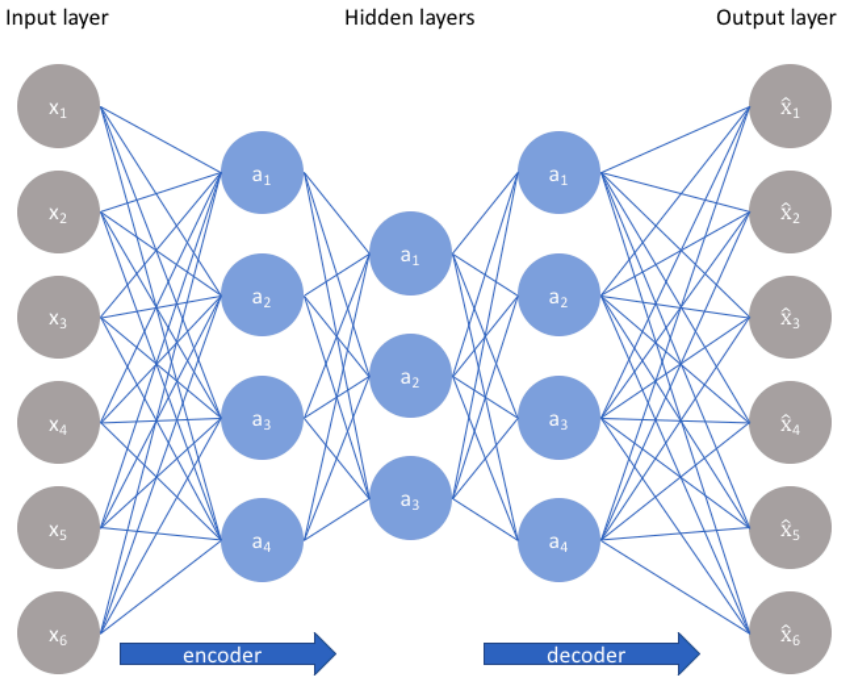
- Often the error/loss function in autoencoders is referred to as reconstruction error
  - Difference between the values of the input layer and the output layer

Regularization in autoencoders

- We want the autoencoder to be sensitive to the input data to replicate it as accurately as possible but insensitive enough to it to avoid memorizing the input
- Regularization is often used with the reconstruction loss function to avoid this problem

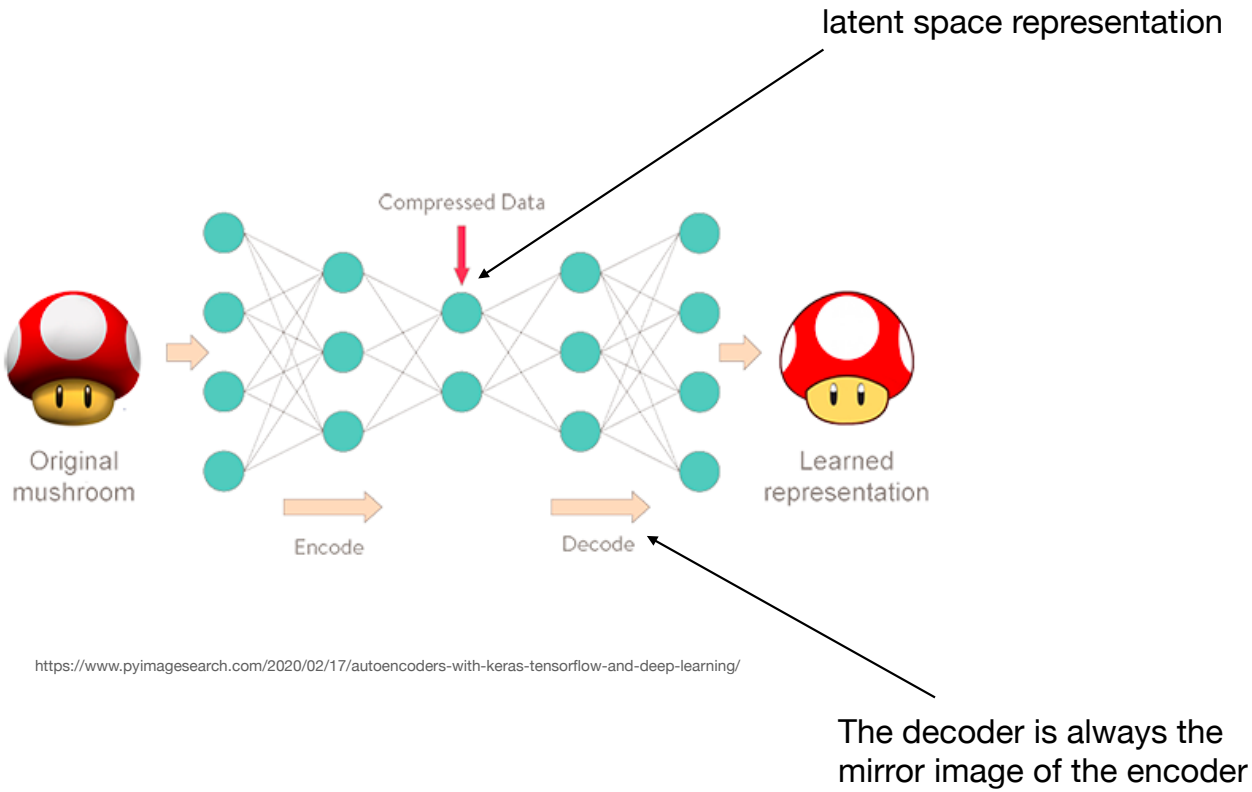
$$\mathcal{L}(x, \hat{x}) + regularizer$$

Autoencoders can have multiple hidden layers



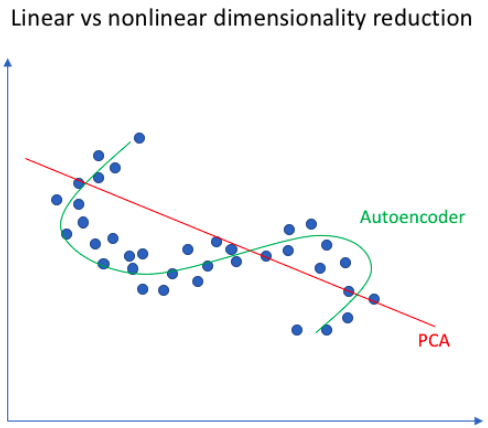
https://www.jeremyjordan.me/autoencoders

Latent space representation is the end goal of the autoencoders



Autoencoders vs. PCA

- Autoencoders learn non-linear function representing the data while PCA and similar methods learn a linear projection function
  - Latent space representation



https://www.jeremyjordan.me/autoencoders

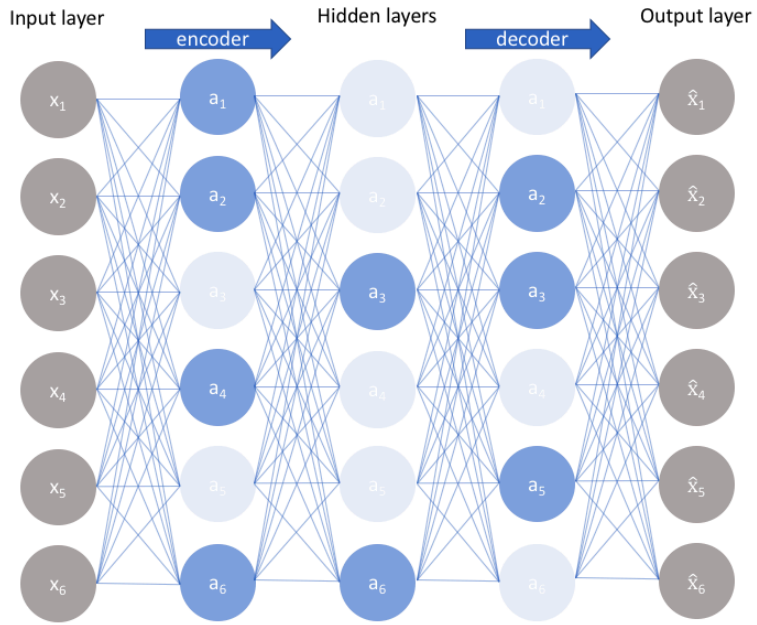
Flavors of autoencoders

- There are many variations on autoencoders
  - Remember that ANNs are very good at memorizing the input
  - We try to avoid overfitting, reducing model bias and model variance

- Sparse autoencoders
  - Imposing sparsity on how many neurons fire up by penalizing activations in the hidden layer
    - Introduces regularization
- Denoising autoencoders
  - Add noise to the input data
- Variational autoencoders (VAE)
  - Takes a probabilistic approach to latent variables
  - VAE are generative models

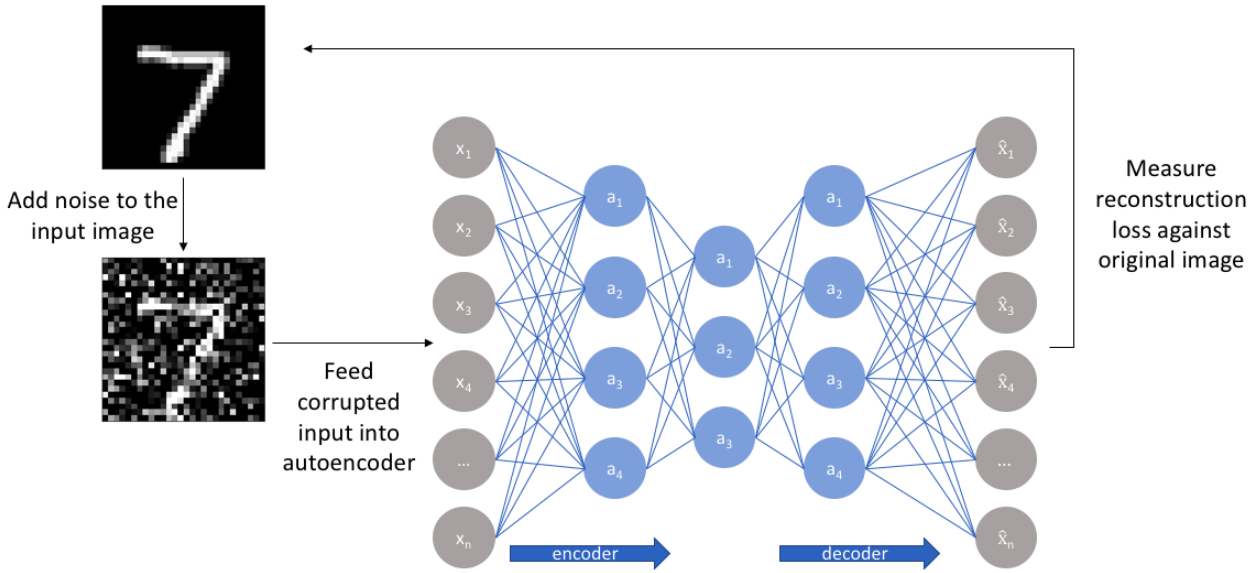
Sparse autoencoder

Imposing sparsity on how many neurons fire up by penalizing activations in the hidden layer



https://www.jeremyjordan.me/autoencoders

Denoising autoencoder



https://www.jeremyjordan.me/autoencoders

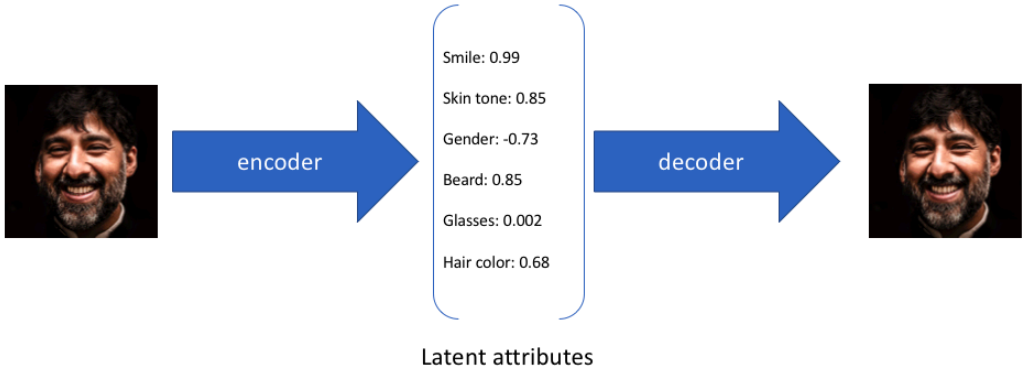
Variational autoencoder

- Latent variables describe the encoding for the input data
- Instead of each latent variable to be a single value attribute we can create probability distributions for each latent variable and estimate probabilities for each observation

- A really good introduction can be found here: https://www.jeremyjordan.me/variational-autoencoders/

Variational autoencoder (cont'd)

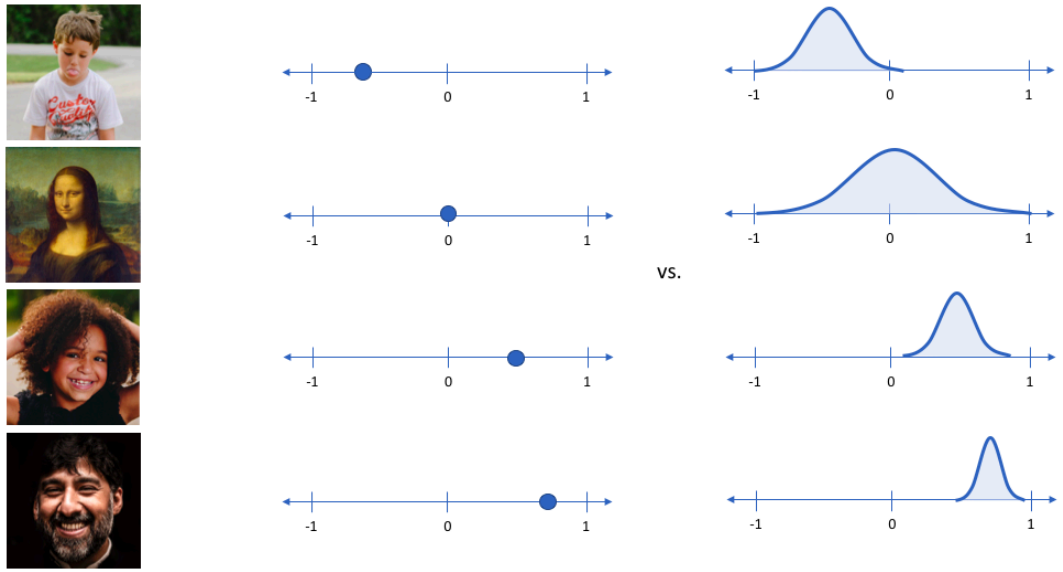
This is how we can visualize conceptual representation of a standard autoencoder:



https://www.jeremyjordan.me/variational-autoencoders/

Variational autoencoder (cont'd)

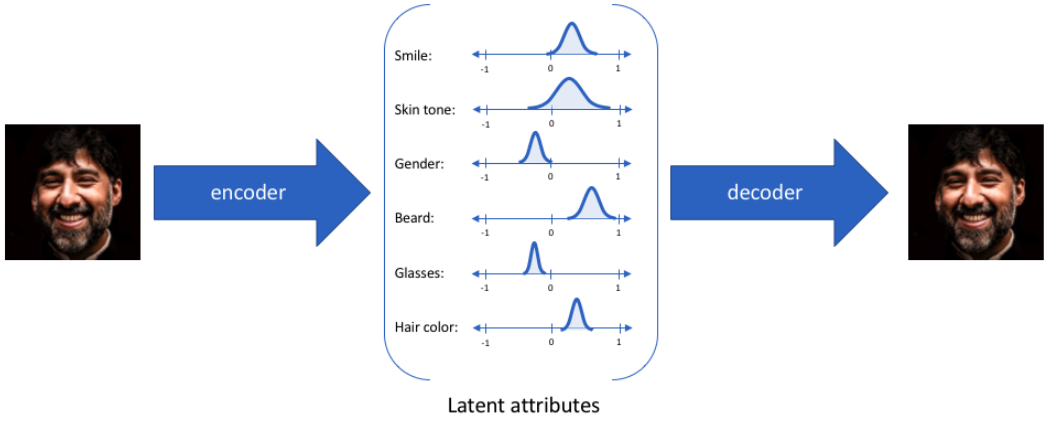
Instead of using single values for each encoded feature, we can use a probability distribution that describes possible values for that particular latent variable for each observation:



https://www.jeremyjordan.me/variational-autoencoders/

Variational autoencoder (cont'd)

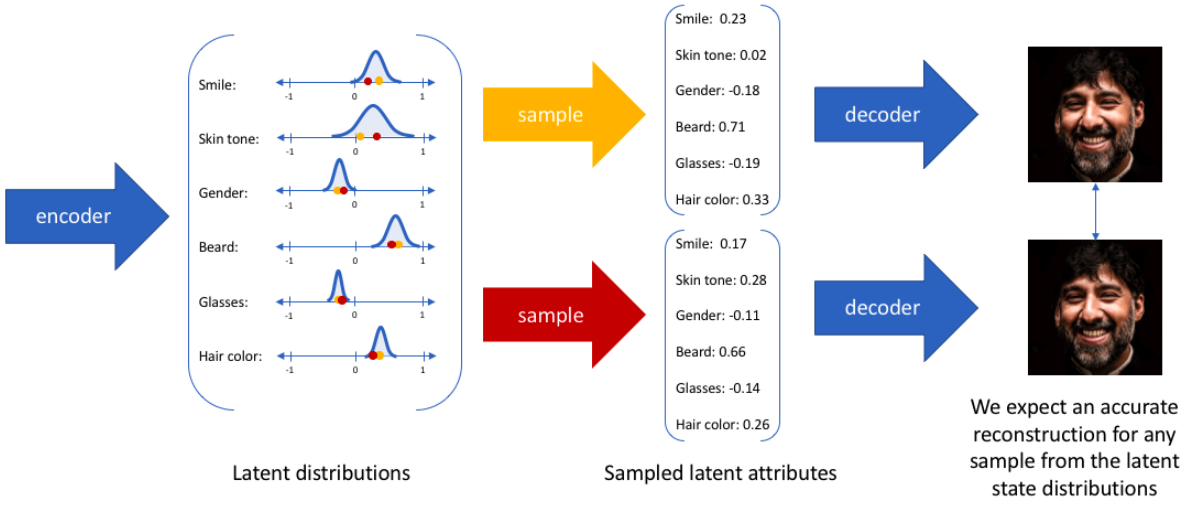
Therefore, each encoded feature is a probability distribution rather than a discrete value:



https://www.jeremyjordan.me/variational-autoencoders/

Variational autoencoder (cont'd)

- Generative model
  - We can use these latent variable distributions to generate new data
- Similar observations produced similar latent variable distributions
- Encoded values that are close to each other the latent space produce very similar decodings

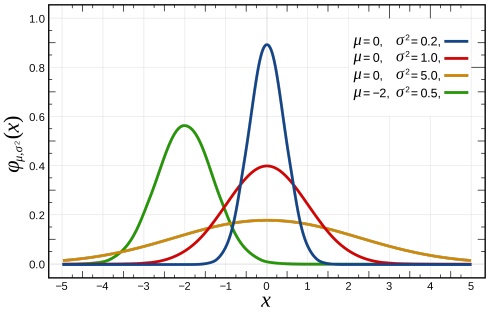


https://www.jeremyjordan.me/variational-autoencoders/

Variational autoencoder (cont'd)

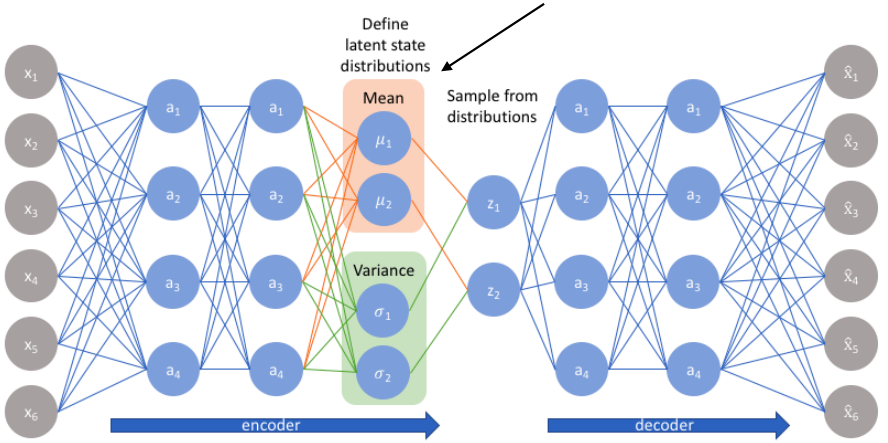
- Latent variables can be modeled by different distributions
  - Remember from statistics that each distribution can be described by a set of parameters
- Multivariate Gaussian VAEs are commonly used models

Gaussian (normal) distribution:



https://en.wikipedia.org

There are 2 latent variables here



https://www.jeremyjordan.me/variational-autoencoders/

Let’s look at some code examples

- Standard autoencoders for MNIST dataset
  - *Autoencoders.MNIST.ipynb*
- Variational autoencoder for generating new digit images
  - *Variational_autoencoder.MNIST.ipynb*