Autoencoders

Hossein Javidnia

DEFINITION

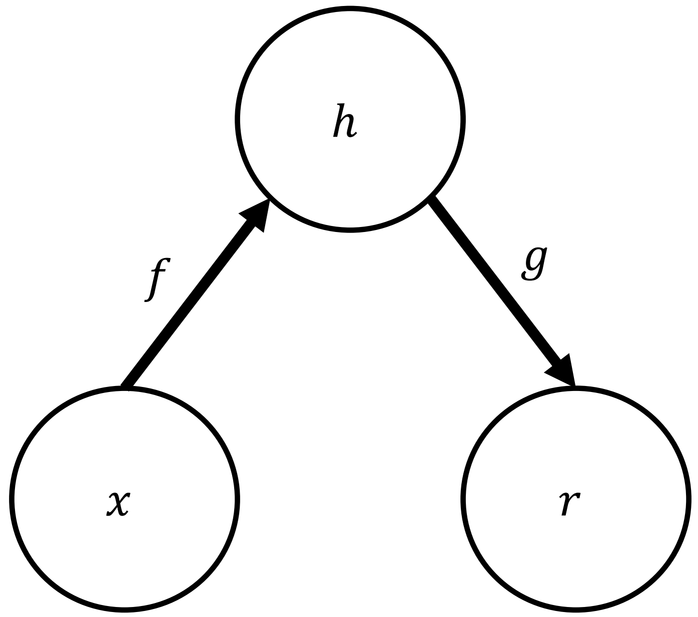
An Autoencoder is a neural network that is trained to attempt to copy its input to its output.

IDEA

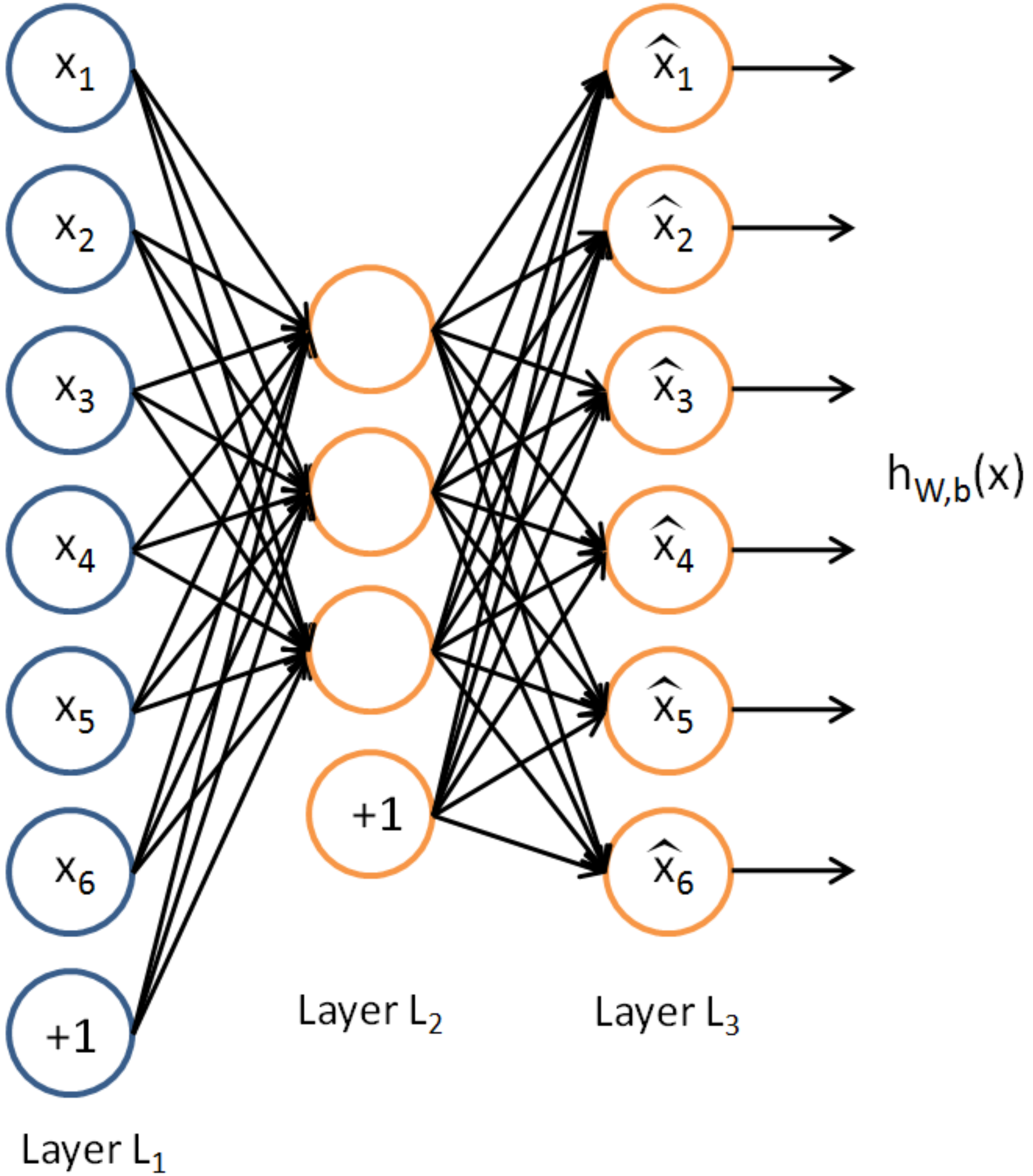
- So far, we have described the application of neural networks to supervised learning, in which we have labelled training examples.

- Now suppose we have only a set of unlabelled training examples $\{x^1, x^2, x^3, \dots\}$, where $x^i \in \Re^n$.

- An **autoencoder** neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. i.e., it uses $y^i = x^i$.



The general structure of an autoencoder, mapping an input x to an output (called reconstruction) r through an internal representation. The autoencoder has two components: the encoder f (mapping x to h) and the decoder g (mapping h to r)


x_1
x_2
x_3
x_4
x_5
x_6
+1
Layer L_1
+1
Layer L_2
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
Layer L_3
$h_{W,b}(x)$

IDEA

- Internally, it has a hidden layer that describes a code used to represent the input.

- The autoencoder tries to learn a function $h_{W,b}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x.

- The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data

EXAMPLE

- As a concrete example, suppose the inputs x are the pixel intensity values from a $10{\times}10$ image (100 pixels) so $n = 100$, and there are $s_2 = 50$ hidden units in layer L_2. Note that, we also have $y \in \Re^{100}$. Since there are only 50 hidden units, the network is forced to learn a ”compressed” representation of the input. i.e., given only the vector of hidden unit activations $a^2 \in \Re^{50}$, it must try to ”reconstruct” the 100-pixel input x.

UNDERCOMPLETE AUTOENCODERS

- Copying the input to the output may sound useless, but we are typically not interested in the output of the decoder. Instead, we hope that training the autoencoder to perform the input copying task will result in h taking on useful properties.

- One way to obtain useful features from the autoencoder is to constrain h to have a smaller dimension than x.

- An autoencoder whose code dimension is less than the input dimension is called **undercomplete**.

UNDERCOMPLETE AUTOENCODERS

- Learning an undercomplete representation, forces the autoencoder to capture the most salient features of the training data.

- The learning process is described simply as minimizing a loss function:

$$L(x, g(f(x)))$$

- Where L is a loss function penalizing $g(f(x))$ for being dissimilar from x, such as the mean squared error.

OVERCOMPLETE AUTOENCODERS

- An autoencoder whose code dimension is larger than the input dimension is called **overcomplete**.

- In this cases, even a linear encoder and a linear decoder can learn to copy the input to the output without learning anything useful about the data distribution.

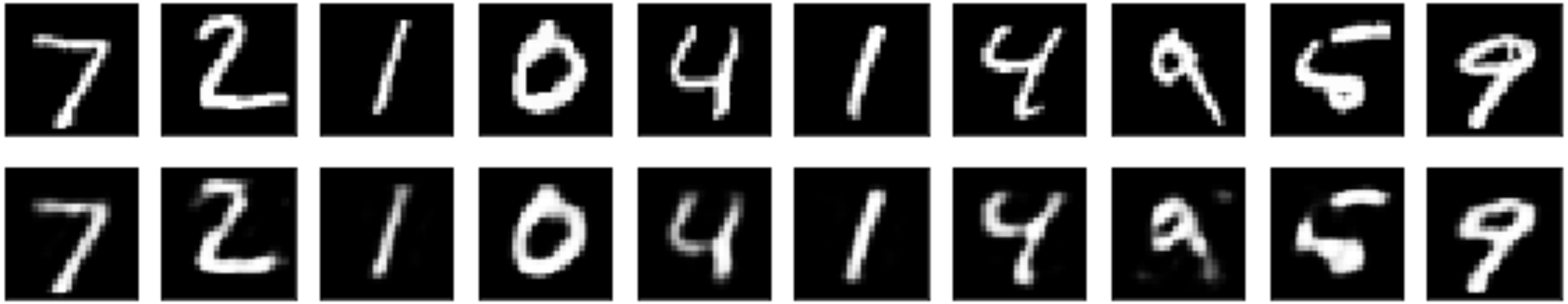
APPLICATIONS

- Dimensionality Reduction
- Image Compression
- Image Denoising
- Feature Extraction
- Image generation
- Sequence to sequence prediction
- Recommendation system

IMAGE COMPRESSION

```
Input shape (28, 28)

inputs = Input(shape=(784,)) #28*28 flatten

encoded = Dense(32, activation='relu’)(inputs) #to 32 data points

decoded = Dense(784, activation='sigmoid’)(encoded) #to 784 data points

model = Model(inputs, decoded)
```

- This whole processing becomes the trainable autoencoder model.



The first row has the original images. The second row has the restored images.

IMAGE DENOISING

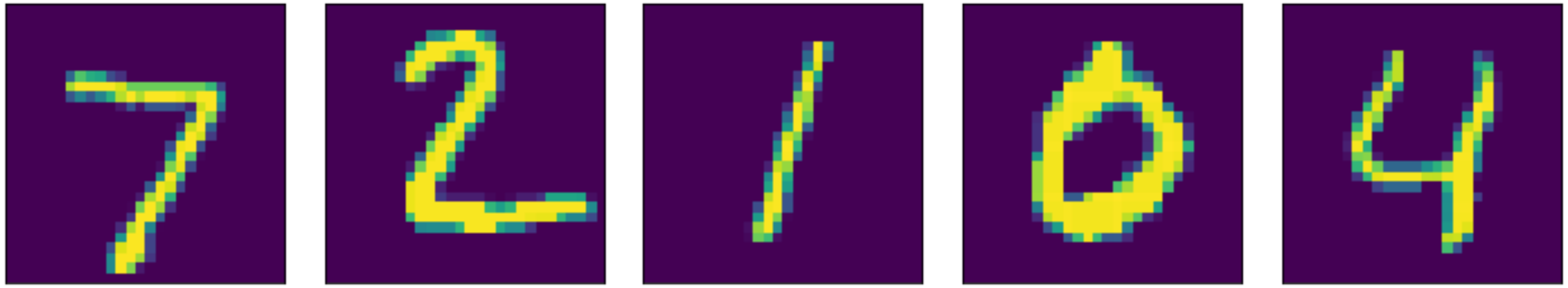
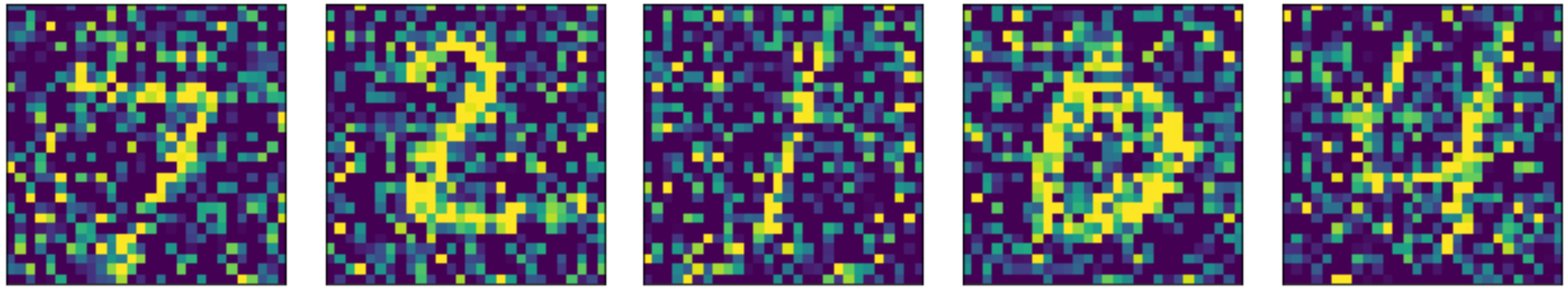


IMAGE DENOISING



```
Input_img = Input(shape=(28, 28, 1))

# encoder architecture
x1 = Conv2D(64, (3, 3), activation='relu', padding='same')(Input_img)
x1 = MaxPool2D( (2, 2), padding='same')(x1)
x2 = Conv2D(32, (3, 3), activation='relu', padding='same')(x1)
x2 = MaxPool2D( (2, 2), padding='same')(x2)
x3 = Conv2D(16, (3, 3), activation='relu', padding='same')(x2)
encoded = MaxPool2D( (2, 2), padding='same')(x3)

# decoding architecture
x3 = Conv2D(16, (3, 3), activation='relu', padding='same')(encoded)
x3 = UpSampling2D((2, 2))(x3)
x2 = Conv2D(32, (3, 3), activation='relu', padding='same')(x3)
x2 = UpSampling2D((2, 2))(x2)
x1 = Conv2D(64, (3, 3), activation='relu')(x2)
x1 = UpSampling2D((2, 2))(x1)
decoded = Conv2D(1, (3, 3), padding='same')(x1)


autoencoder = Model(Input_img, decoded)
```

IMAGE DENOISING