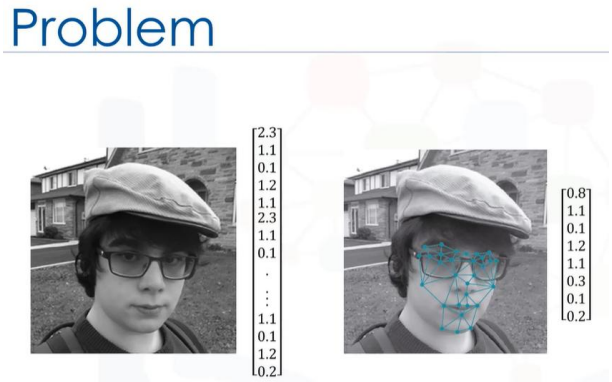
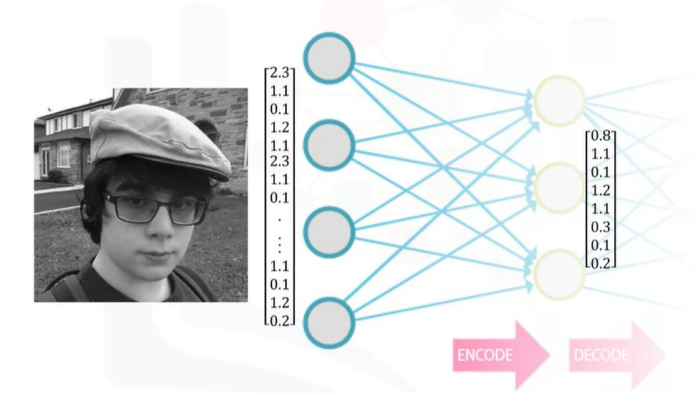
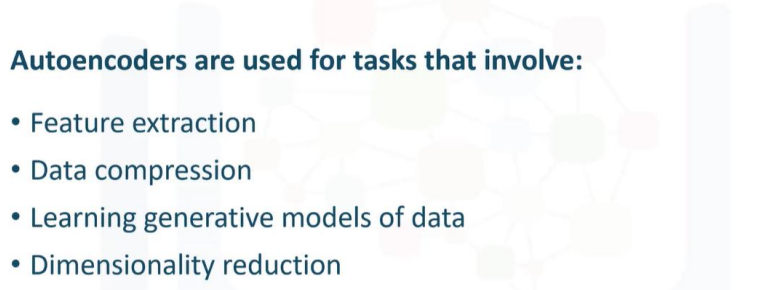
Module 5 - Autoecoders

5.1 Intro to Autoencoders

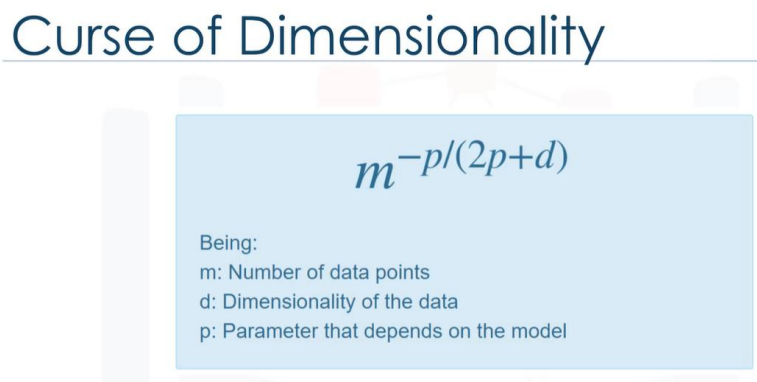






Hello, and welcome! In this video, we'll be covering the basicconcepts and the motivation behind autoencoders, a type of neural network used in unsupervised machine learning. Let's say that you want to extract the feeling or emotion of a person in a photograph. The photograph that you see here is a small image that's just 256 by 256 pixels… But, this means that there are over 65 thousand pixels in play in defining the dimension of the input! As we increase the dimensionality, the time to deal with data increases exponentially, in order to train and fit the raw data into a neural network that can detect the emotion. So, we need a way to extract the most important features of a face, and represent each image with those features which are of lower dimensions. An autoencoder works well for this type of problem.

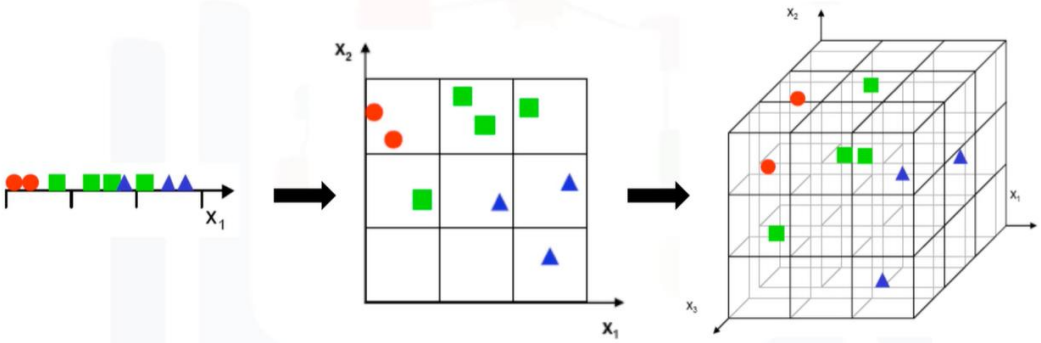
An autoencoder is a type of unsupervised learning algorithm that will find patterns in a datasetby detecting key features. It is a type of neural net that analyzes all of the images in your dataset and extracts some useful features automatically in such a way that it can distinguish images using those features. Generally speaking, autoencoders excel in tasks that involve feature learning or extraction, data compression, and learning generative models of data and dimensionality reduction.



Let me talk more about Dimension Reduction.High-dimensional data is a significant problem for machine learning tasks. In fact, data scientists commonly refer to it as the "CURSE OF DIMENSIONALITY". Many common problems that we want to target have a large number of dimensions. Even that 256 x 256 pixel image that we saw earlier, would have over 65 thousand dimensions; or in other words, one dimension for each pixel. A high-resolution image from a smart phone would be even larger. According to a study, the time to fit a model is, at best, the function you see here, which is a function of the number of points, dimensionality, and parameters. So, as we increase the dimensionality, our time to fit our model will increase exponentially.

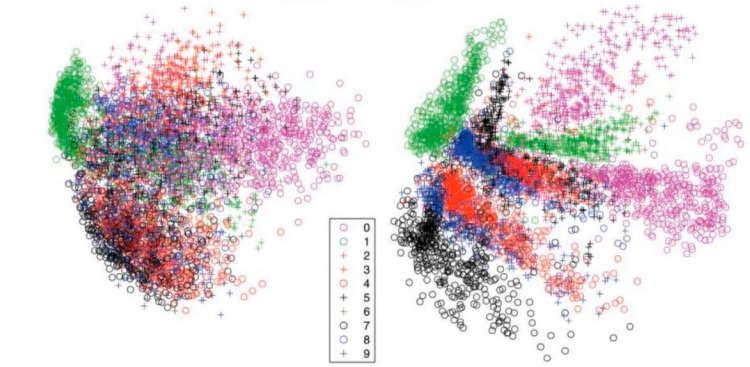
总结:

Sparsity



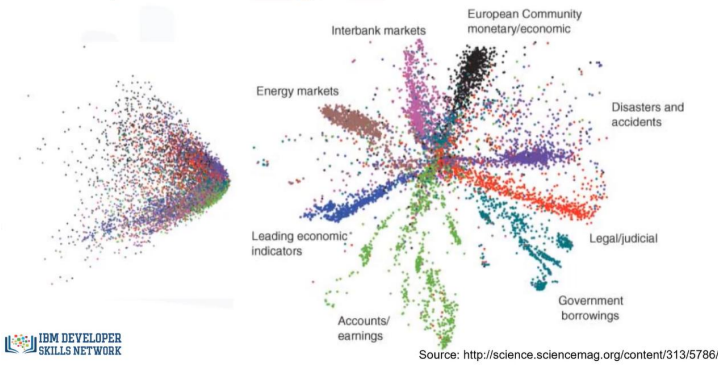
So, what happens when we increase or reduce the dimension of data?Well, if we have a huge number of dimensions, our data will start to get sparse, which results in an over-allocation of memory and slow training time. We run into additional problems when we try to reduce the dimensions. If we have a small number of dimensions, our data could overlap, resulting in a loss of data characteristics. You can see how that looks in one dimension and three dimensions. **Overlap** and **sparsity** make it difficult to determine the underlying patterns. However, with the proper number of dimensions, the patterns become much clearer.

Comparison against PCA



Source: http://science.sciencemag.org/content/313/5786/504.full

Comparison against PCA



Source: http://science.sciencemag.org/content/313/5786/504.full

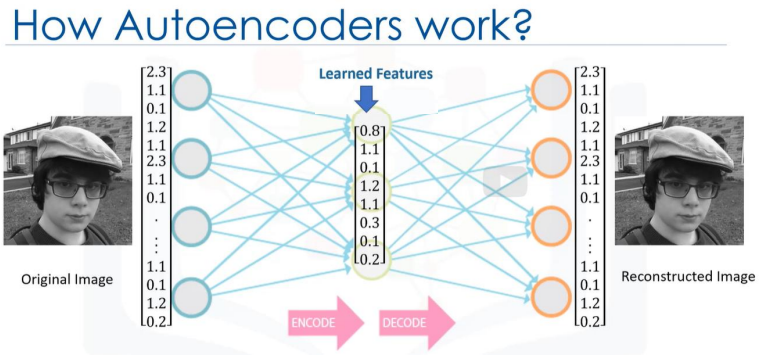
It's important to know that an Autoencoder is not the only dimension reduction methodin Machine Leaning. Principal Component Analysis (or PCA) has been around for a long time, and is a classic algorithm for dimensionality reduction. Take a look at the data separation outputs for the MNIST dataset of handwritten digits. These charts show the image data after reducing their dimensions to 2 dimensions. The left output is from PCA. The output on the right is from an autoencoder. As you can see, it's easier to discern the data with the autoencoder's output.

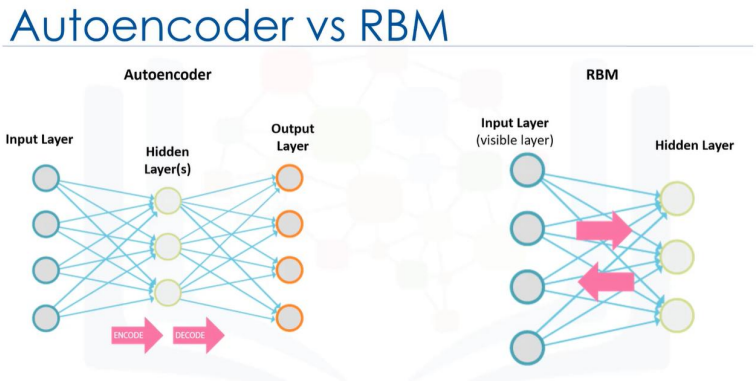
Here is another comparison between PCA and an autoencoder, now applied to news stories.The separability of the autoencoder is far better than the PCA, in this case. Since separability is important for applying clustering algorithms, this difference in quality is significant.

By now, you should understand that an autoencoder can extract key image features, improve training times of other networks, and improve the separability of reduced datasets when compared to other methods. For these reasons, the autoencoder was a breakthrough in the unsupervised learning research field.

总结:

5.2 Autoencoder Structure



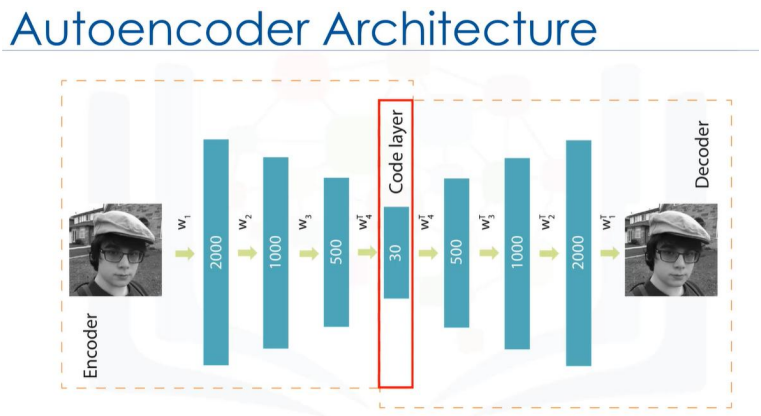


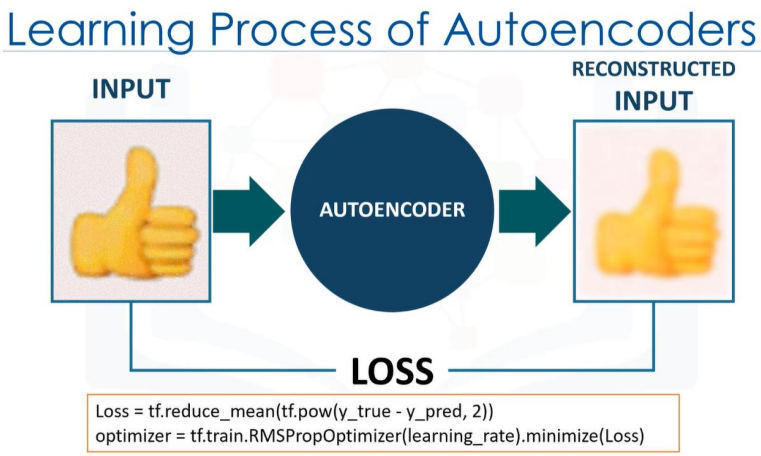
Hello, and welcome! In this video, we'll be examining the architectureof autoencoders and show you how autoencoders work. An autoencoder neural network is supposed to represent the images in a dataset with a low dimension feature set. For example, it extracts the most important features of faces, for an arbitrary task such as face recognition. Also, it is supposed to do it in an unsupervised manner, that is, "feature extraction" without provided labels for images. Now, let's see how Authoencoders actually work. An autoencoder is an artificial neural network that's designed to find important features by recreating the given input. Generally speaking, the main goal of Autoencoders is to take unlabeled inputs, encode them, and then try to reconstruct them afterwards, based on the most valuable features identified in the data. In fact, Autoencoders are based on Restricted Boltzmann Machines, or in other words, Restricted Boltzmann Machines are a type of Autoencoder.

So, what is the difference between Autoencoders and Restricted Boltzmann Machines -- or RBMs,for short?

- Well, most autoencoders are shallow networks with an input layer, a few hidden layers, and an output layer. RBMs, on the other hand, are autoencoders with only two layers.
- Also, Autoencoders differ from Restricted Boltzmann Machines because they use a deterministic approach, rather than a stochastic approach.

Ok, let's take a closer look at the architecture of autoencoders.





An autoencoder can be divided into two parts, the encoder and the decoder.The encoder needs to compress the representation of an input. In this case, we're going to compress the face of our actor, moving from 2000-dimensions of data to only 30 dimensions. The decoder is a reflection of the encoder's network. It works to recreate the input as accurately as it can. It has an important role during training, and that is to force the autoencoder to select the most important features in the compressed representation. As you can see, we really don't care about the reconstructed image in this network. What we do care about, though, is the code layer values, which represents the input layer. That is, if the network has learned enough that it can generate the replica of input images only based on the code-layer value, then, this vector of size 30, is good enough as the feature set to represent the input image. Therefore, after the training is complete, you can use the encoded data that has been dimensionally-reduced for the application of your choosing. This can include clustering, classification, or visualization of your data. So, as you can see, the Autoencoder can be considered as an unsupervised feature extraction technique for different machine learning algorithms.

So, how do Autoencoders learn? Autoencoders use back-propagation in theirlearning process. The metric used to assess the quality of the network is loss, which is the amount of information lost in the reconstruction of the input. The squared error of reconstruction for a single sample is the average of the squared error of the reconstructed image for all dimensions. The goal is to minimize the loss, so that we have an output that's as close to the input as possible. At this point, you should have a better understanding of the structure and applications of autoencoders.

总结:

what is the difference between Autoencoders and RBMs?

- ○ Autoencoders are used for supervised learning, but RBMs are used for unsupervised learning.
- ◉ Autoencoders use a deterministic approach, but RBMs use a stochastic approach.
- ○ Autoencoders have less layeres than RBMS.
- ○ All of the above

✔

Which of the following problems cannot be solved by Autoencoders:

- ○ Dimensionality Reduction
- ◉ Time series prediction
- ○ Image Reconstruction
- ○ Emotion Detection
- ○ All of the above

✔

What is TRUE about Autoencoders:

- ○ Help to Reduce the Curse of Dimensionality
- ○ Used to Learn the Most important Features in Data
- ○ Used for Unsupervised Learning
- ◉ All of the Above

✔

总结：

What are Autoencoders:

- ○ A Neural Network that is designed to replace Non-Linear Regression
- ◉ A Neural Network that is trained to attempt to copy its input to its output
- ○ A Neural Network that learns all the weights by using labeled data
- ○ A Neural Network where different layer inputs are controlled by gates
- ○ All of the Above

✔

What is a Deep Autoencoder:

- ◉ An Autoencoder with Multiple Hidden Layers
- ○ An Autoencoder with multiple input and output layers
- ○ An Autoencoder stacked with Multiple Visible Layers
- ○ An Autoencoder stacked with over 1000 layers
- ○ None of the Above

✔

总结:

Module 6 - Deployment

Learning Objectives

In this course you will learn about:

- How to train your machine learning model using **Watson Machine Learning**.
- Saved your machine learning model on **Cloud Object Storage**.
- Deploy your machine learning model using **Python Cloud Foundry**.

Lab: Model Deployment

Bookmark this page

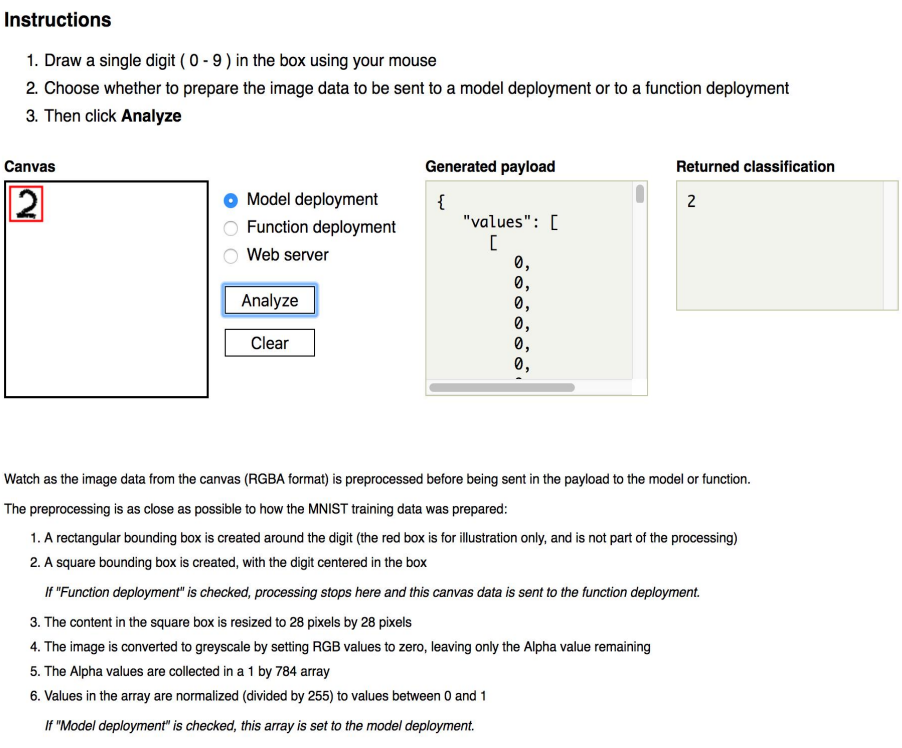
In the previous sections you learned how to train various deep learn model. This section you will learn how to save your machine learning model so that you can use your saved model for retraining, another machine learning task, or deploy your model on the cloud.

There are various advantages to deploying your machine learning model on the cloud. First, you can access your model anywhere in the world by entering the URL of your model into any web browser. Second, you can display your machine learning model on your portfolio, anyone can view your machine learning model by clicking the link to your model.

This lab consists of two Jupyter Notebooks.

- The first notebook Model-Training-Saving.ipynb, contains the instructions to train and save your machine learning model on IBM Cloud
- The second notebook Model-Deployment.ipynb, contains the instructions to deploy your machine learning model model on IBM Cloud

Upon finishing this lab you will be able to deploy your model as a web app as shown in the image below.



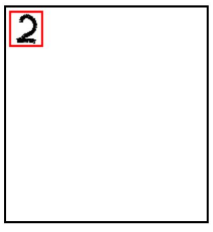
总结：

Final Exam

Instructions

1. Draw a single digit (0 - 9) in the box using your mouse
2. Choose whether to prepare the image data to be sent to a model deployment or to a function deployment
3. Then click **Analyze**

Canvas



- (•) Model deployment
- ( ) Function deployment
- ( ) Web server

Analyze

Clear

Generated payload

```
{
   "values": [
      [
         0,
         0,
         0,
         0,
         0,
         0,
```

Returned classification

```
2
```

Watch as the image data from the canvas (RGBA format) is preprocessed before being sent in the payload to the model or function.

The preprocessing is as close as possible to how the MNIST training data was prepared:

1. A rectangular bounding box is created around the digit (the red box is for illustration only, and is not part of the processing)
2. A square bounding box is created, with the digit centered in the box

   If "Function deployment" is checked, processing stops here and this canvas data is sent to the function deployment.

3. The content in the square box is resized to 28 pixels by 28 pixels
4. The image is converted to greyscale by setting RGB values to zero, leaving only the Alpha value remaining
5. The Alpha values are collected in a 1 by 784 array
6. Values in the array are normalized (divided by 255) to values between 0 and 1

   If "Model deployment" is checked, this array is set to the model deployment.

总结: