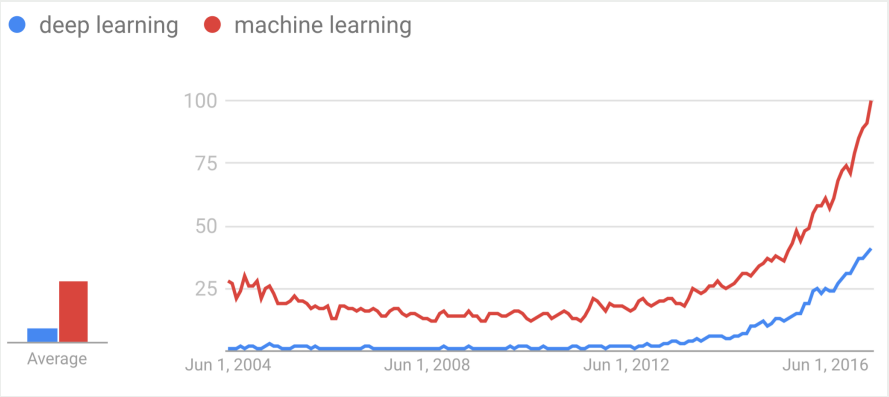
Deep Learning: Auto-Encoder

Rensheng Wang,
https://sit.instructure.com/courses/22680
March 22, 2018

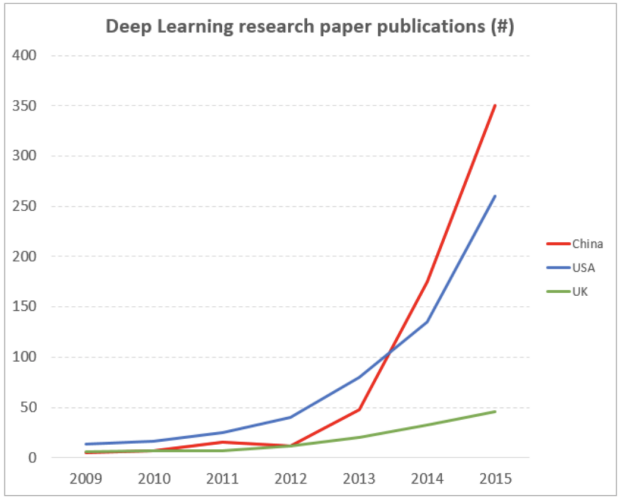
Machine Learning vs. Deep Learning

- Interest in google trends



from Machine Learning to Deep Learning

- Academic publications about deep learning



The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.

this lecture from supervised → unsupervised methods

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.
- For example, if you pick a random point in a unit square (a 1×1 square), it will have only about a 0.4% chance of being located less than 0.001 from a border.

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.
- For example, if you pick a random point in a unit square (a 1×1 square), it will have only about a 0.4% chance of being located less than 0.001 from a border.
- But in a 10,000-dimensional unit hypercube (a $1 \times 1 \times \cdots \times 1$ cube, with ten thousand 1s), this probability is greater than 99.999999%.

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.
- For example, if you pick a random point in a unit square (a 1×1 square), it will have only about a 0.4% chance of being located less than 0.001 from a border.
- But in a 10,000-dimensional unit hypercube (a $1 \times 1 \times \cdots \times 1$ cube, with ten thousand 1s), this probability is greater than 99.999999%.
- This fact implies that high- dimensional datasets are at risk of being very sparse: most training instances are likely to be far away from each other.

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance.
- Not only does this make computational complex, it can also make it much harder to find a good solution. This problem is often referred to as the curse of dimensionality.
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.
- For example, if you pick a random point in a unit square (a 1×1 square), it will have only about a 0.4% chance of being located less than 0.001 from a border.
- But in a 10,000-dimensional unit hypercube (a $1 \times 1 \times \cdots \times 1$ cube, with ten thousand 1s), this probability is greater than 99.999999%.
- This fact implies that high- dimensional datasets are at risk of being very sparse: most training instances are likely to be far away from each other.
- The more dimensions the training set has, the greater the risk of overfitting it.

What is Auto-Encoder?

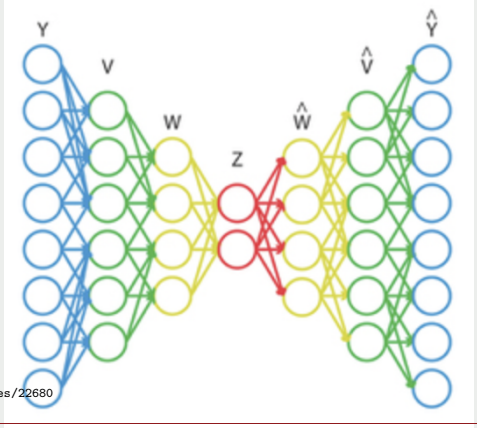
- Autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.

What is Auto-Encoder?

- Autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.
- Unsupervised Learning : Work by simply learning to copy their inputs to their outputs

What is Auto-Encoder?

- Autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.
- Unsupervised Learning : Work by simply learning to copy their inputs to their outputs
- Find the deterministic function f : $\boldsymbol{z} = f(\boldsymbol{y})$, $\boldsymbol{y}$: data input, $\boldsymbol{z}$: coding
- Find the generation function g : $\hat{\boldsymbol{y}} = g(\boldsymbol{z})$, $\hat{\boldsymbol{y}}$: reconstructed input, $\boldsymbol{z}$: coding



Auto-Encoder vs. PCA

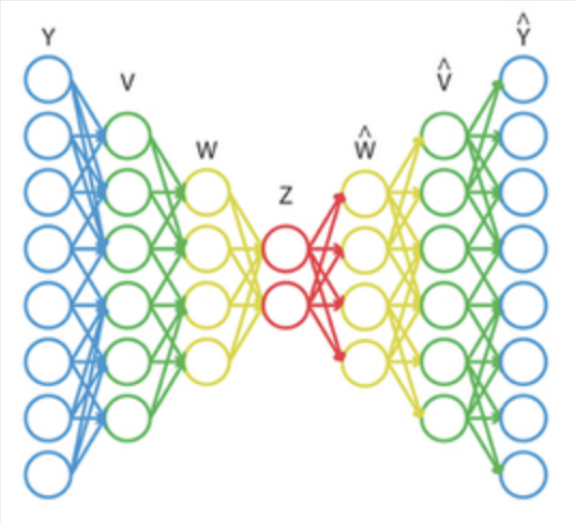
- auto-encoder vs. principal components analysis (PCA)
  - It is an unsupervised learning algorithm (like PCA)
  - It minimizes the same objective function as PCA
  - It is a neural network
  - The neural networks target output is its input

Auto-Encoder vs. PCA

- auto-encoder vs. principal components analysis (PCA)
  - It is an unsupervised learning algorithm (like PCA)
  - It minimizes the same objective function as PCA
  - It is a neural network
  - The neural networks target output is its input
- Why not PCA?
  - Auto-encoders are much more flexible than PCA
  - Neural networks have an activation function which brings nonlinearities in our encoding, whereas PCA can only represent linear transformations
  - The network representation also means you can stack auto-encoders to form a deep network

Auto-Encoder

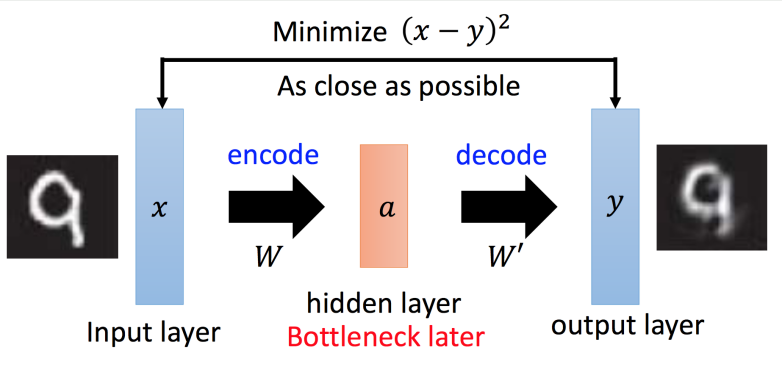
- Use 28×28 matrix to represent a digit



- It is possible to represent the digit image in a more compact way

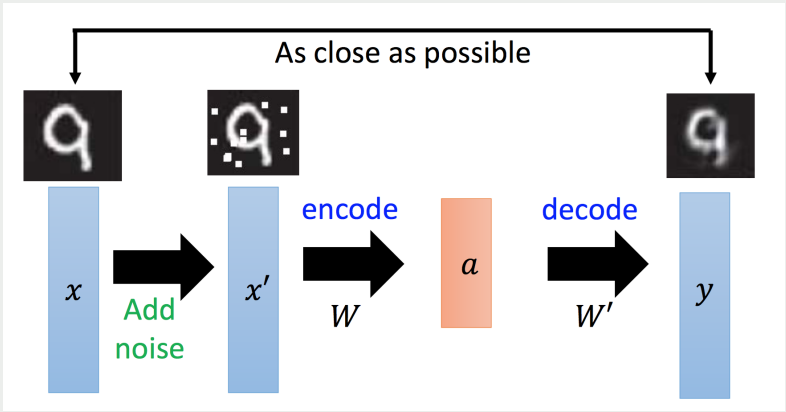
Auto-Encoder

- The cost function is the distance between original image and the decoded image.
- It is an optimization problem.



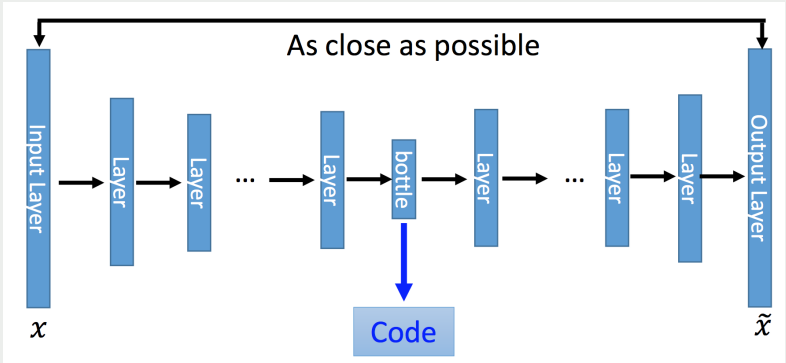
De-Noising Auto-Encoder

- Adding noise to make the learning more robust
- De-noising training force NN implicitly learn the structure



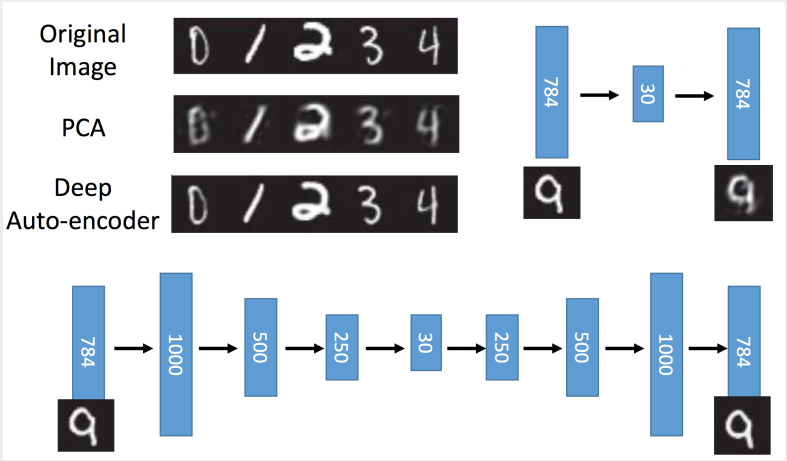
Deep Auto-Encoder

- Of course, the auto-encoder can be **deep**.
- The motivation is to reduce the dimensionality of data using neural network.



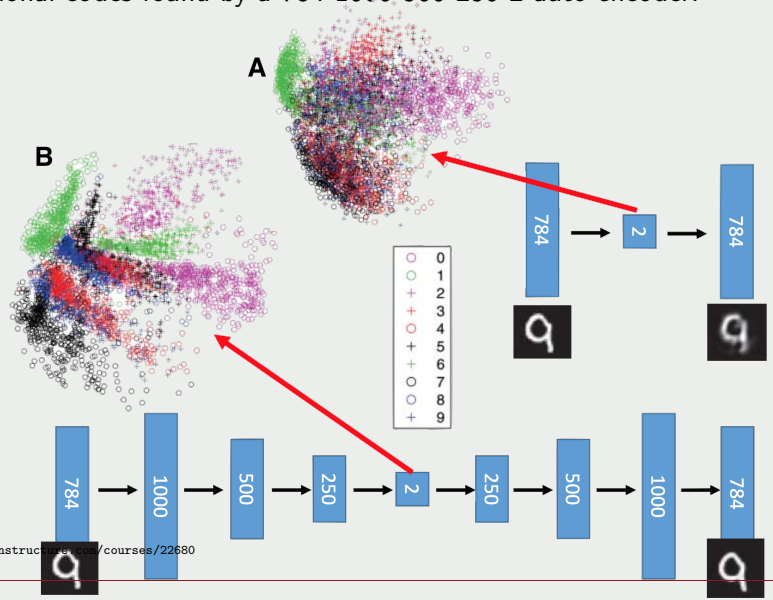
Deep Auto-Encoder: Deeper is Better

- High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors.



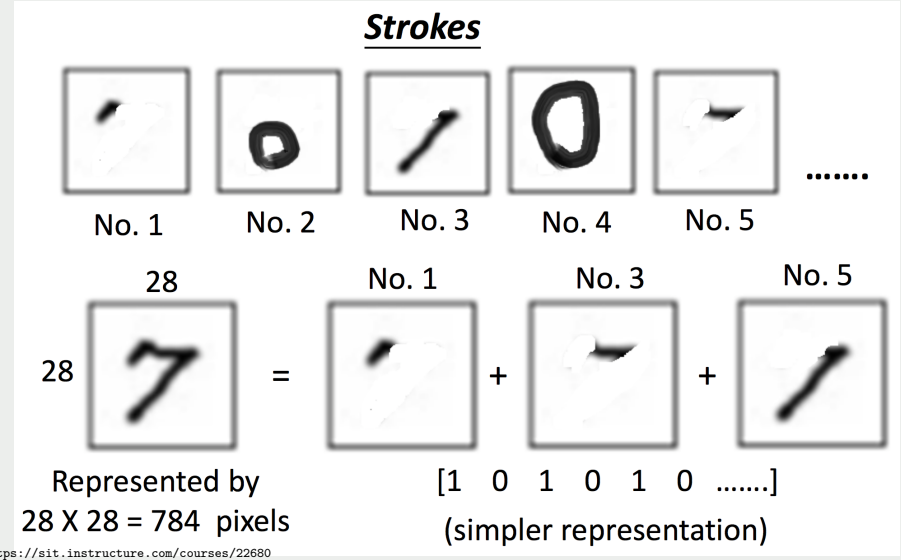
Deep Auto-Encoder: Deeper is Better

- (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (B) The two-dimensional codes found by a 784-1000-500-250-2 auto-encoder.



Why Self-Taught Learning Can Work?

- Deep learning can extract the common basic components and reform the combinations in a simpler way



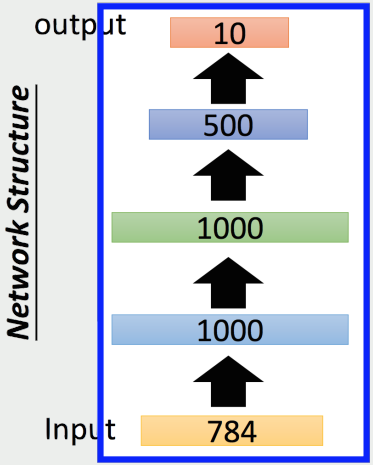
Auto-Encoders as an Initialization Method

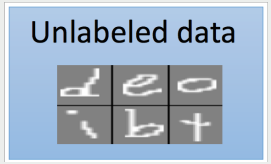
- The magnitudes of gradients in the lower layers and in higher layers are different
- The landscape or curvature of the objective function is difficult for stochastic gradient descent to find a good local optimum
- Deep networks have many parameters, which can remember training data and do not generalize well.
- Researchers have shown that using auto-encoder for pretraining improves deep neural networks;
- Perhaps because pretraining is done one layer at a time which means it does not suffer from the difficulty of full supervised learning.

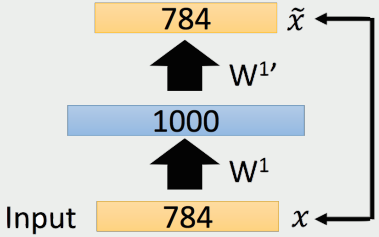
Semi-Supervised Learning

- Handwriting digit recognition

  Better initialization by auto-encoder trained with unlabeled data instead of random initial parameters

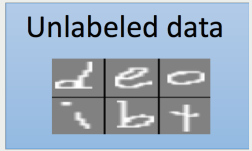


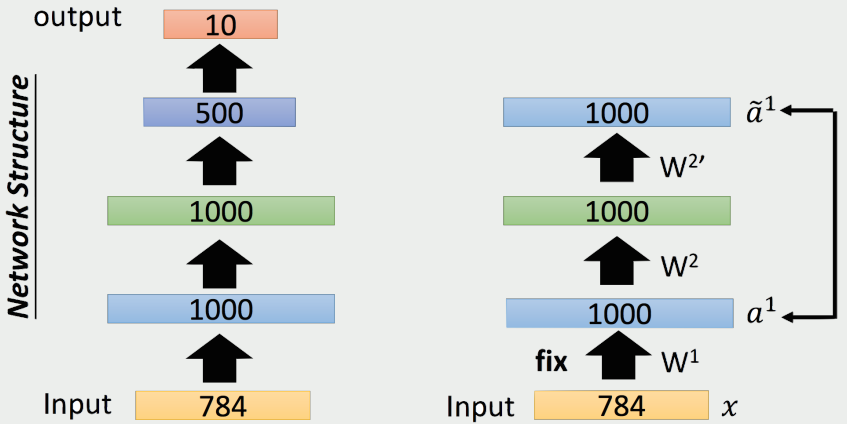




Semi-Supervised Learning

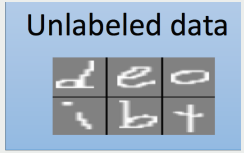
- After first-level auto-encoder training with unlabeled data, fix the parameters $\mathbf{W}_1$, and continue to auto-encode the next level

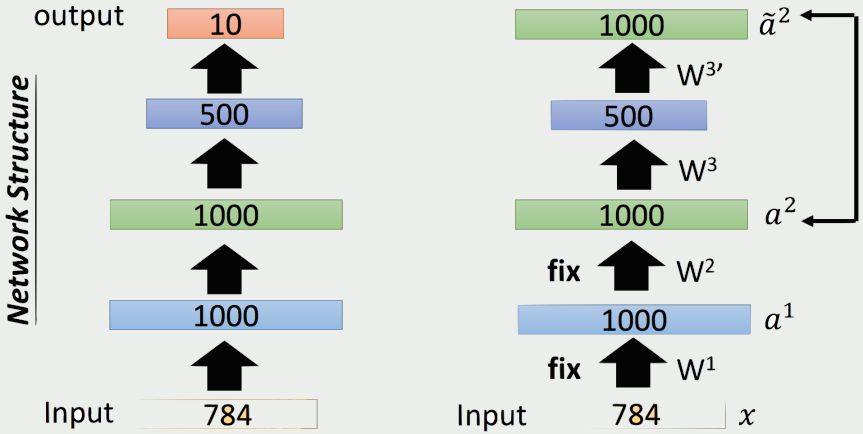




Semi-Supervised Learning

- Keep the trained parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ and continue to auto-encode the next level of unlabeled data





Semi-Supervised Learning

- Copy the training parameters to the first 3 levels, and continue the conventional supervised learning of labeled digits by initialization with random
- Initialized by unlabeled data is better than random initializatio