Introduction to Deep Learning

STAT5241 Section 2

Statistical Machine Learning

Xiaofei Shi

Overview

Images & Video

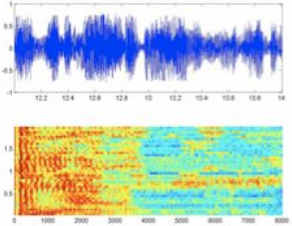
flickr
Google
YouTube

Text & Language

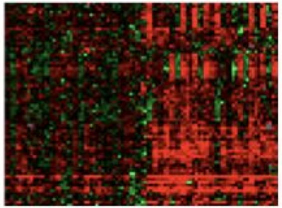
Wikipedia
The Free Encyclopedia

REUTERS
AP Associated Press

Speech & Audio



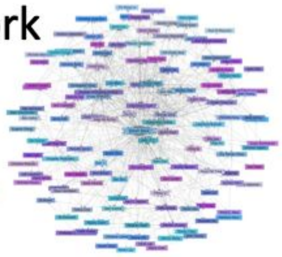
Gene Expression



Product
Recommendation

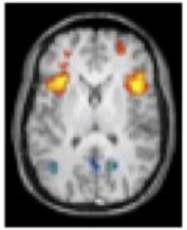
amazon
NETFLIX
ebay

Relational Data/
Social Network

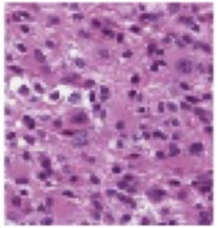


facebook
twitter

fMRI



Tumor region



Mining for Structure

- Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video

flickr Google YouTube

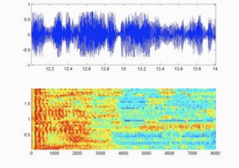
Text & Language

Wikipedia The Free Encyclopedia

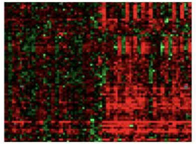
REUTERS

AP Associated Press

Speech & Audio



Gene Expression

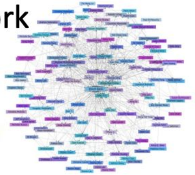


Product Recommendation

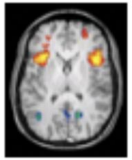
amazon NETFLIX ebay

Relational Data/ Social Network

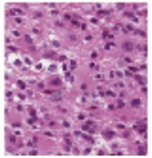
facebook twitter



fMRI



Tumor region



- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

Columbia University

Impact of Deep Learning

- Speech Recognition
- Computer Vision
- Recommender Systems
- Language Understanding
- Drug Discovery and Medical Image Analysis

Understanding pictures

TAGS:

strangers, coworkers, conventioneers, attendants, patrons

Nearest Neighbor Sentence:

people taking pictures of a crazy person

Model Samples

- a group of people in a crowded area .
- a group of people are walking and talking .
- a group of people, standing around and talking .

Caption generation

a car is parked in
the middle of nowhere .

a wooden table and chairs
arranged in a room .

there is a cat sitting on a shelf .

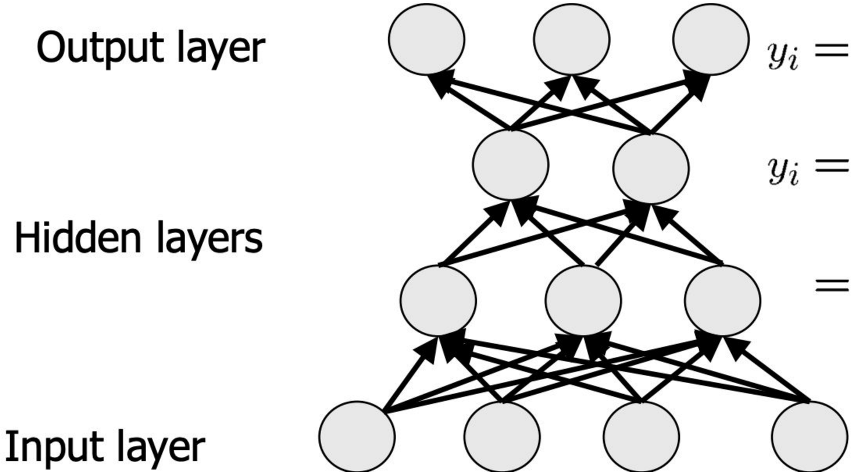
a ferry boat on a marina
with a group of people .

a little boy with a bunch
of friends on the street .

Columbia University

Definition of Deep Learning

Definition: Deep architectures are composed of *multiple levels* of non-linear operations, such as neural nets with many hidden layers.



$$y_i = \tanh\left(\sum_j W_{ij} x_j + W_{i0}\right)$$

$$y_i = \sigma\left(\sum_j W_{ij} x_j + W_{i0}\right)$$

$$= \frac{1}{1 + \exp[-(\sum_j W_{ij} x_j + W_{i0})]}$$

Important breakthrough

Deep Belief Networks (DBN)

Hinton, G. E, Osindero, S., and Teh, Y. W. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527-1554.

Autoencoders

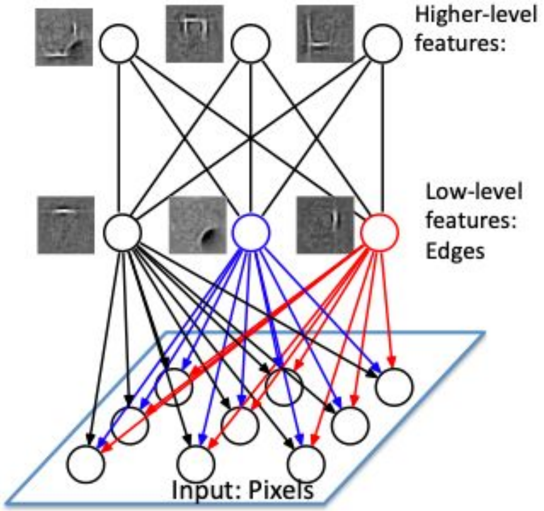
Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007).
Greedy Layer-Wise Training of Deep Networks,
Advances in Neural Information Processing Systems 19

Convolutional neural networks running on GPUs (2012)
Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, Advances in Neural
Information Processing Systems 2012

Deep neural network: important breakthrough

- Deep Belief Networks, 2006 (Unsupervised)

  Hinton, G. E., Osindero, S. and Teh, Y., A fast learning algorithm for deep belief nets, Neural Computation, 2006.



Theoretical Breakthrough:

- Adding additional layers improves variational lower-bound.

Efficient Learning and Inference with multiple layers:

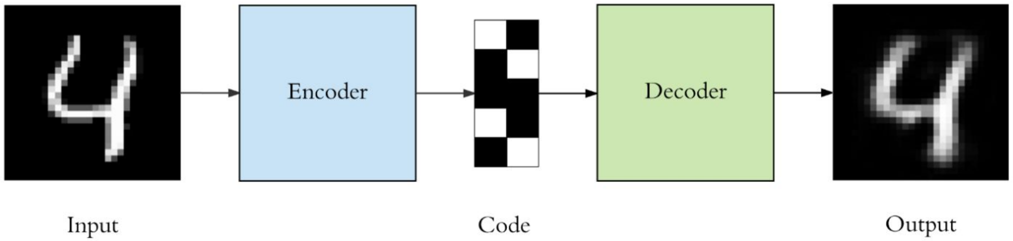
- Efficient greedy layer-by-layer learning learning algorithm.

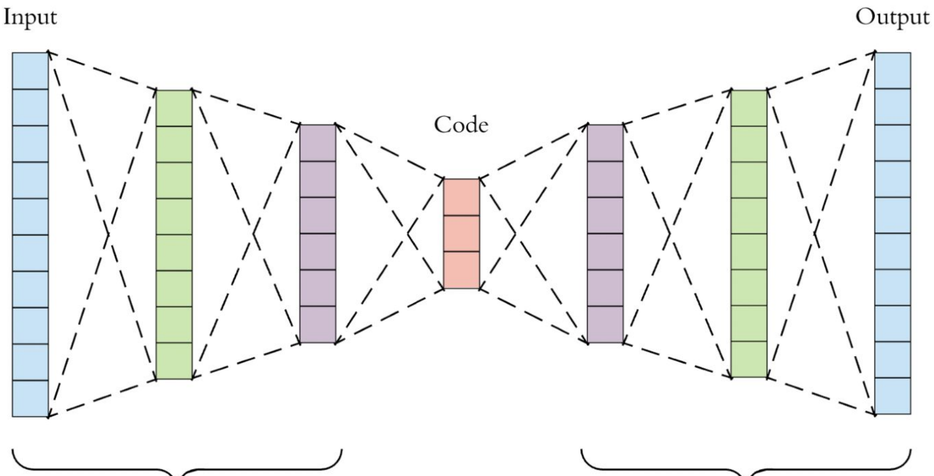
- Inferring the states of the hidden variables in the top most layer is easy.

Columbia University

Deep neural network: important breakthrough

Autoencoders





Key idea:
nonlinear activation + sparse representation

Columbia University

Deep neural network: important breakthrough

- Conditional generative model P(zebra images| horse images)

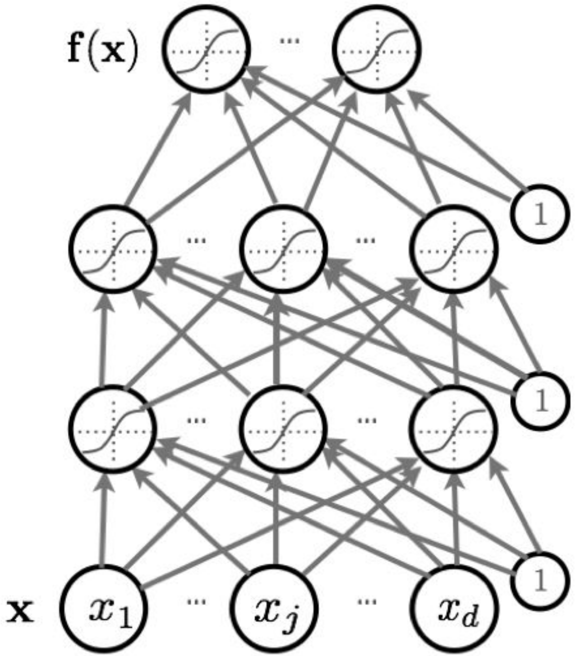
- Style Transfer

Input Image   Monet   Van Gogh

Zhou el al., Cycle GAN 2017

Deep neural network: architecture

- How neural networks predict f(x) given an input x:
  - Feed forward
  - Types of activations
  - Capacity of neural networks
- How to train neural networks:
  - Loss function
  - Backward propagation with gradient descent
- More recent techniques:
  - Architecture
  - Dropout
  - SGD
  - Batch normalization



Deep neural network: architecture

- Consider a network with L hidden layers.
  - layer pre-activation for k>0

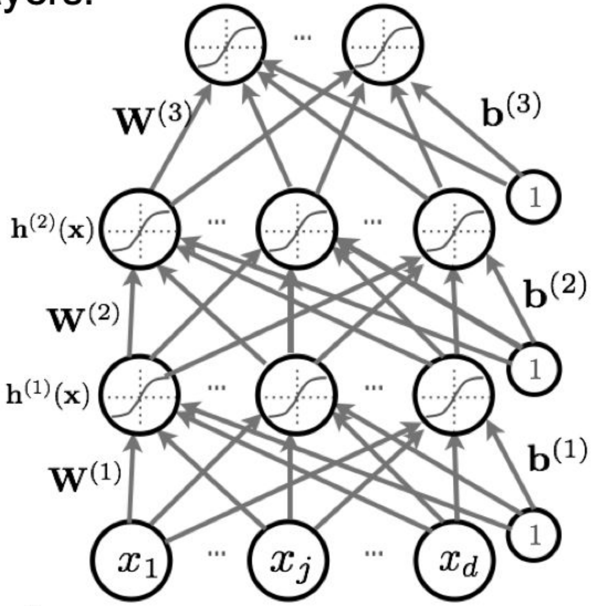
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation from 1 to L:

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation (k=L+1):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



$$(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$$

COLUMBIA UNIVERSITY

Neural network structure

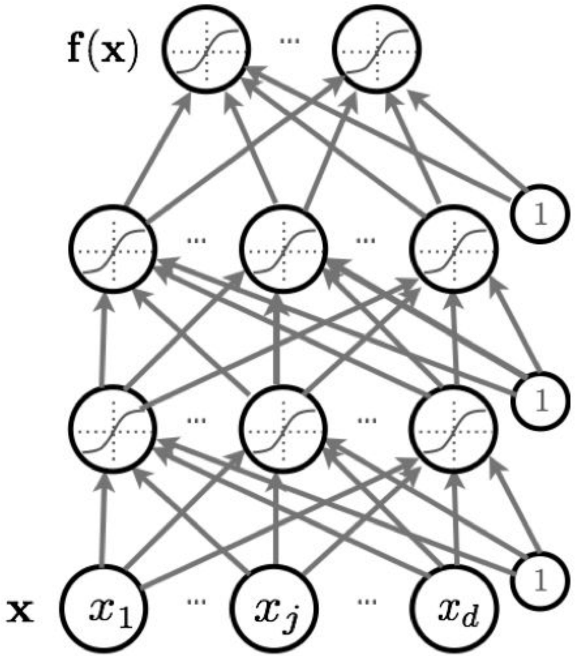
- Fully connected layer
- Deep architecture

  - LeNet5

  - AlexNet

  - ResNet

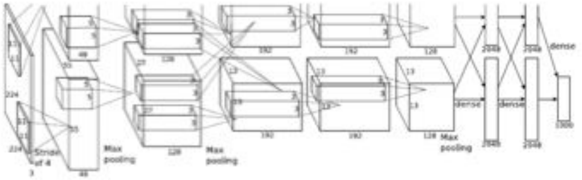
  - ......
- Semi-supervised learning



Deep neural network: important breakthrough

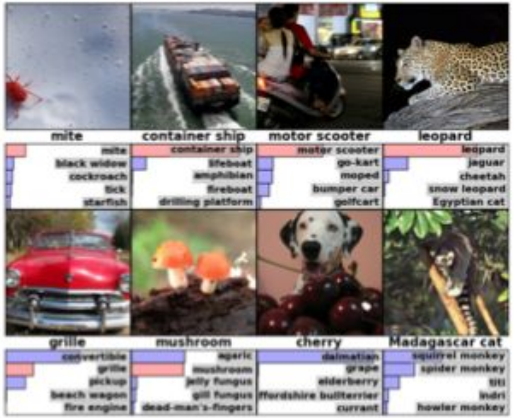
- Deep Convolutional Nets for Vision (Supervised)

  Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.



IMAGENET

1.2 million training images
1000 classes

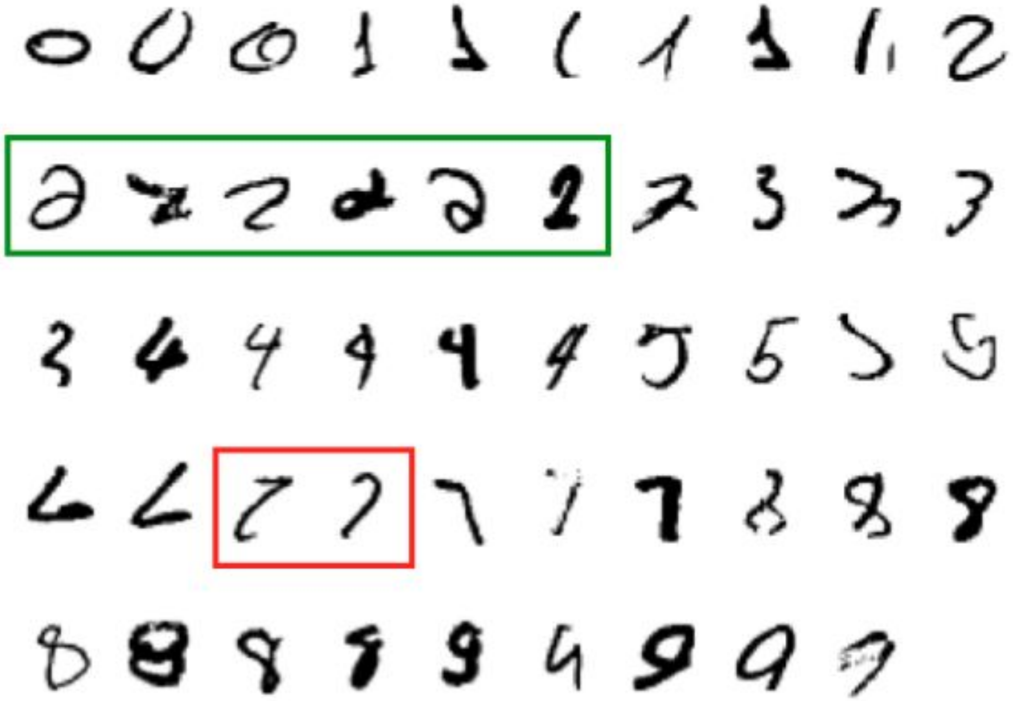


- Deep Nets for Speech (Supervised)

  Hinton et. al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Processing Magazine. 2012.

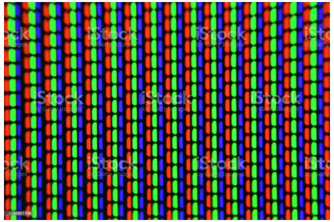
COLUMBIA UNIVERSITY

MNIST

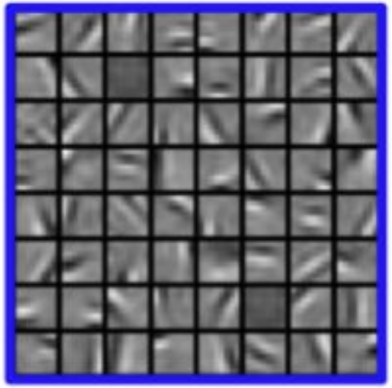


Convolutional neural network

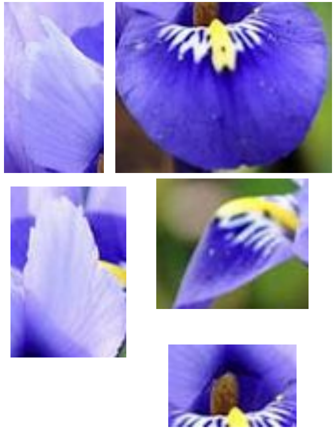
pixels



features



object parts/
combination of features

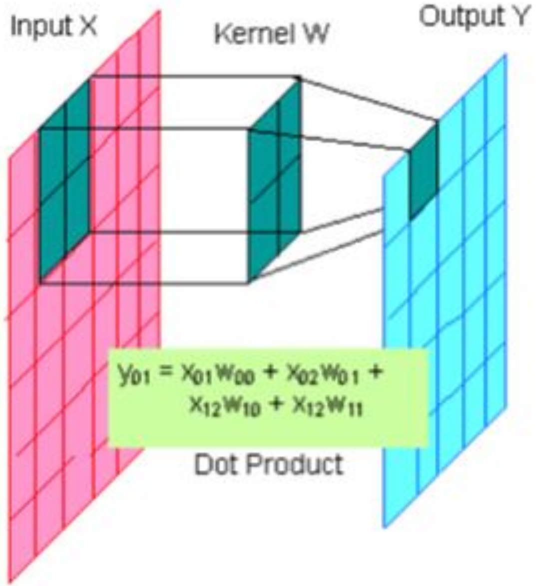


objects

COLUMBIA
UNIVERSITY

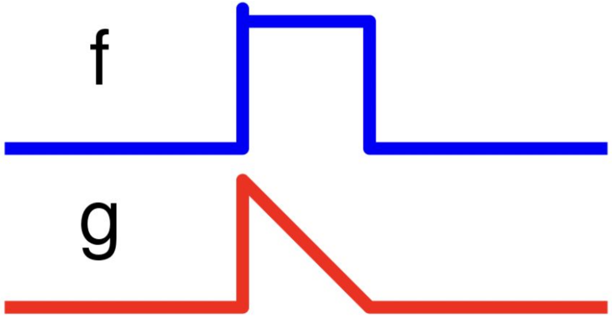
Convolutional neural network

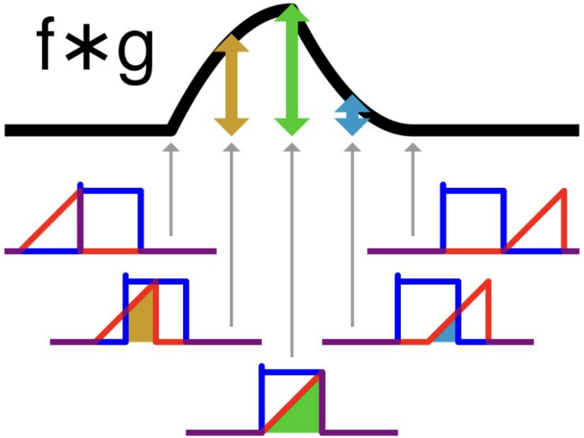
- Instead of focusing on individual, CNN provides a automatic algorithm to study groups of nearby pixels.
- Very successful in
  - computer vision (CV)
  - natural language processing (NLP)
- Compared to standard feedforward neural networks with similarly-sized layers,
  - CNNs have much fewer connections and parameters



COLUMBIA UNIVERSITY

Convolution





Continuous functions:

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau - = \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$

Discrete functions:

$$(f*g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m] = \sum_{m=-\infty}^{\infty} f[n-m]\, g[m]$$

2-dim Convolution

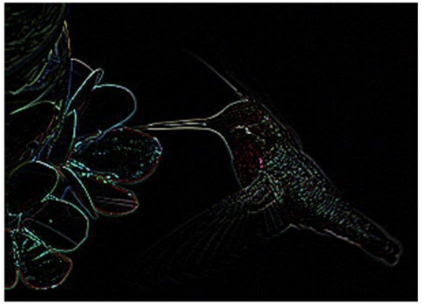
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty} \sum_{n_2=-\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

https://graphics.stanford.edu/courses/cs178/applets/convolution.html

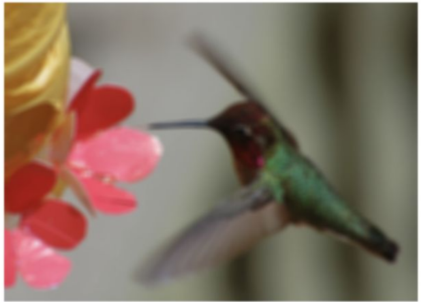
Original

Filter (=kernel)

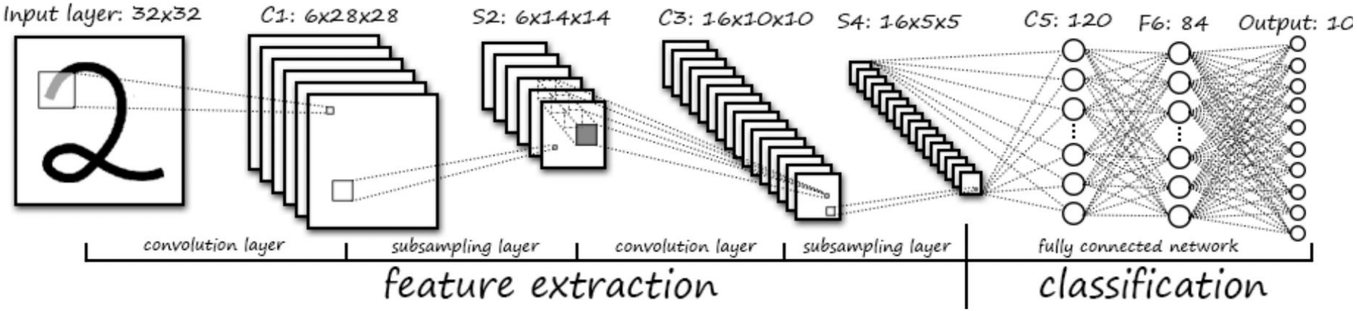
0.00	0.00	0.00	0.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	-2.00	8.00	-2.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00



0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04



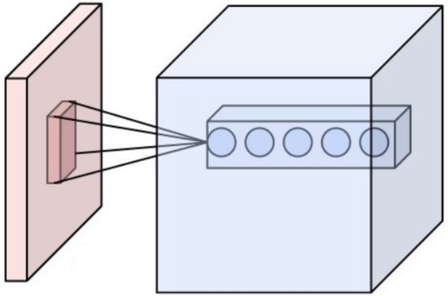
LeNet 5, LeCun et al. 1998



- **Input:** 32x32 pixel image. Largest character is 20x20
  (All important info should be in the center of the receptive fields of the highest level feature detectors)
- **Cx:** Convolutional layer (C1, C3, C5) tanh nonlinear units
- **Sx:** Subsample layer (S2, S4)
- **Fx:** Fully connected layer (F6) logistic/sigmoid units
- Black and White pixel values are normalized:
  E.g. White = -0.1, Black =1.175 (Mean of pixels = 0, Std of pixels =1)

Columbia University

Convolutional neural network

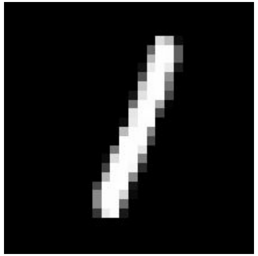


- Hyperparameters in convolutional layer:

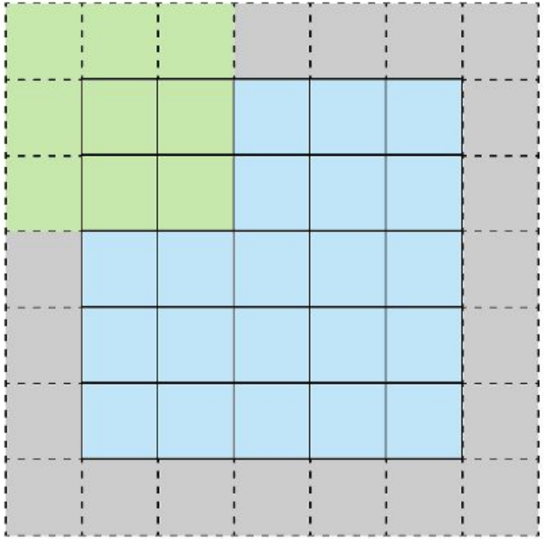
  (pytorch, MNIST data)

- The input sample size is (1,28,28)

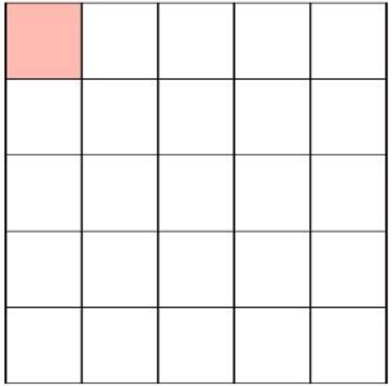
```
torch.nn.Conv2d(
in_channels    = 1,  # we only have grayscale figure
out_channels = 16, # number of filters
kernel_size    = 5,   # filter size
stride             = 1,   # filter movement
padding          = 2   # when stride = 1,
                              padding = (kernel_size -1)/2
)
```



- Output size is (16,28,28)



Stride 1 with Padding



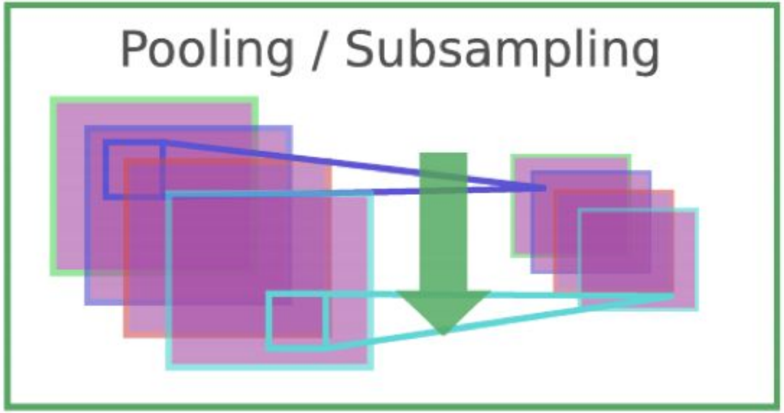
Feature Map

$$N + 2p - F + 1 = N \Rightarrow p = (F-1)/2$$

COLUMBIA UNIVERSITY

Convolutional neural network

- Pooling/subsampling hidden units in same neighborhood
  - Introduces invariance to local translations
  - Reduces the number of hidden units in hidden layer
- Hyperparameters in pooling layers
  torch.nn.MaxPool2d(2)



Jarret et al. 2009

COLUMBIA UNIVERSITY

Convolutional neural network

(1,28,28)

convolutional layer

(16, 28, 28)

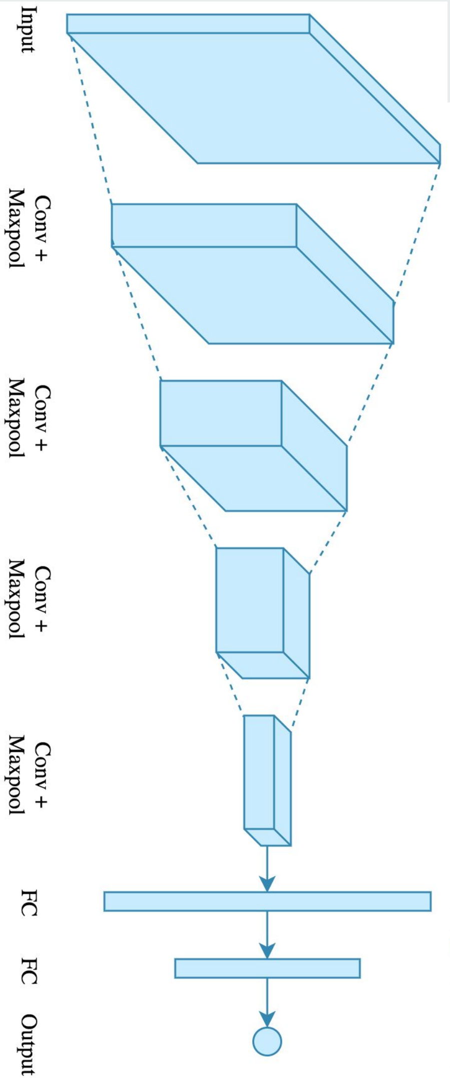
nonlinear activation

layer (usually ReLU)

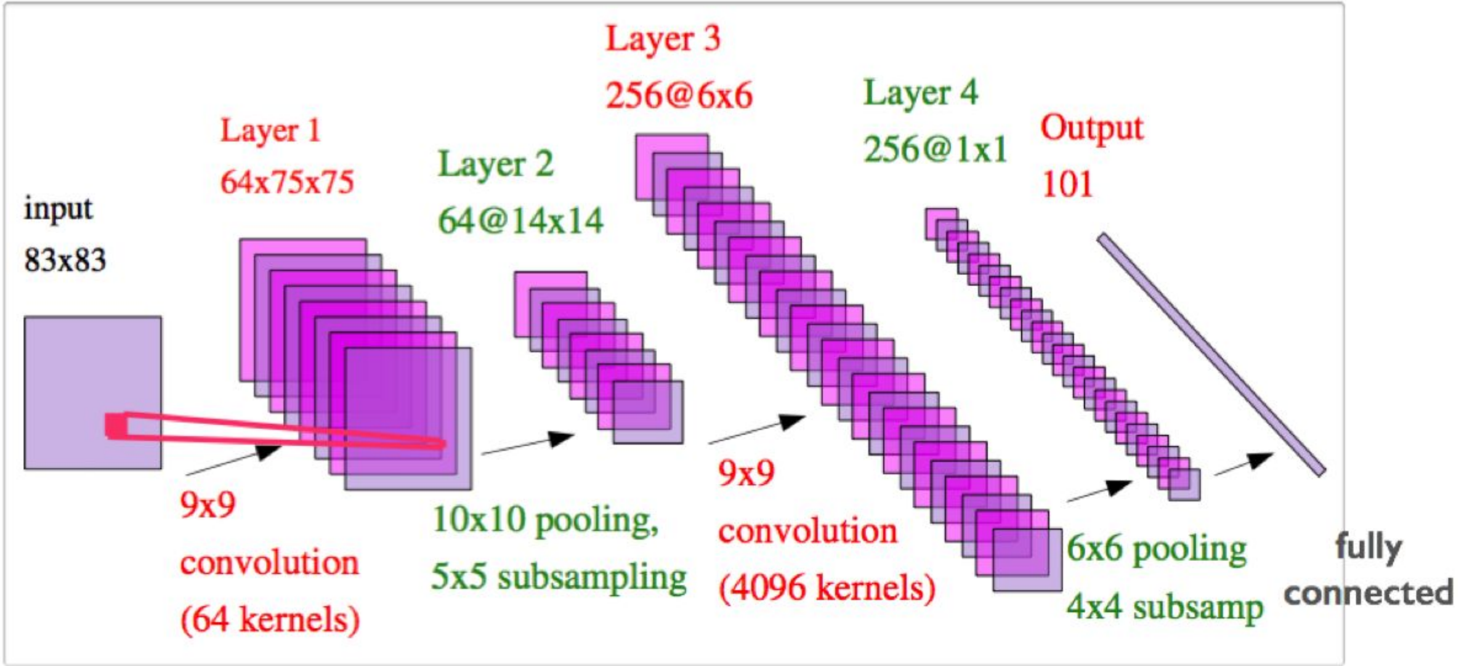
(16, 28, 28)

max pooling

(16, 14, 14)



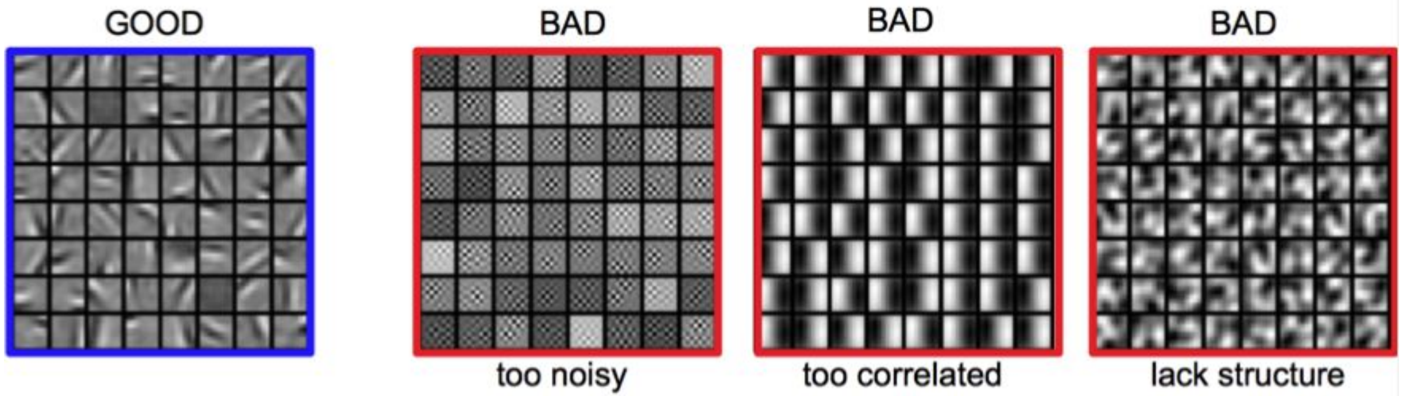
Convolutional neural network



From Yann LeCun's slides

Convolutional neural network

- Visualize parameters:
  learned features should exhibit structure and should be uncorrelated and are uncorrelated

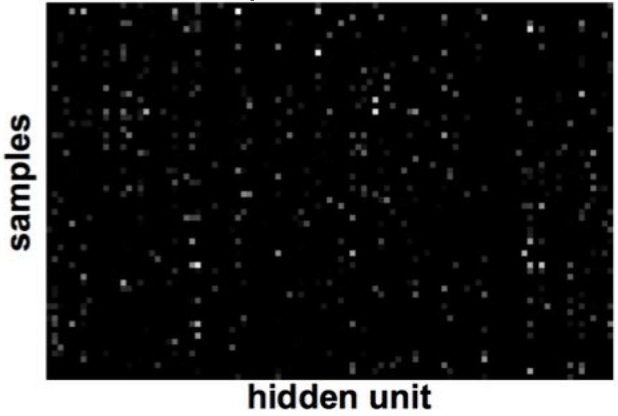


[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Convolutional neural network

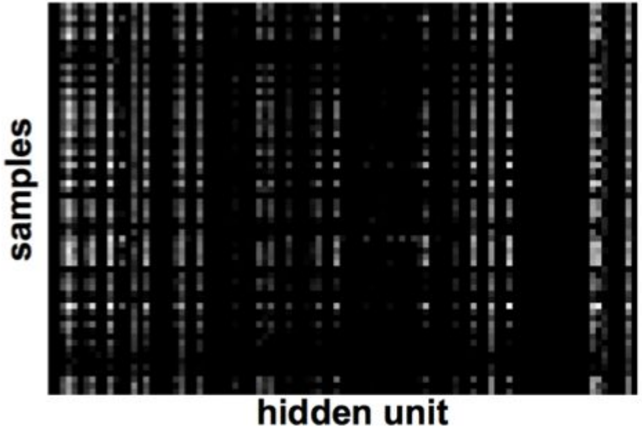
- Visualize features (feature maps need to be uncorrelated)

Good training:
hidden units are sparse
across samples.



[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Bad training:
hidden units are highly
correlated.



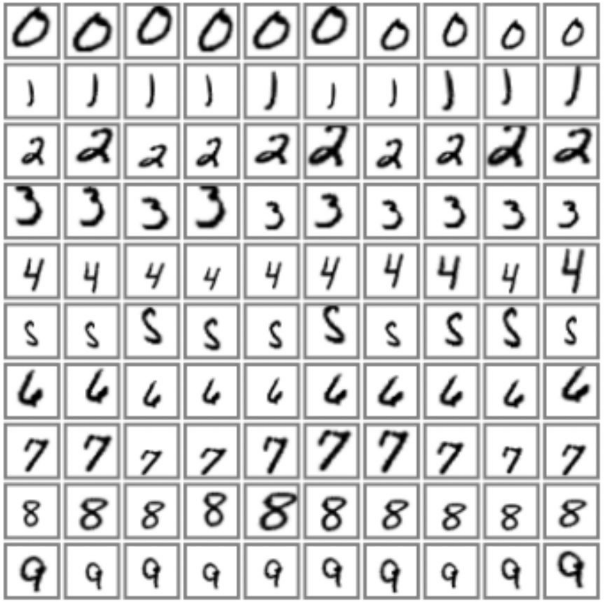
[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

COLUMBIA UNIVERSITY

MNIST data

60,000 original dataset

Test error: 0.95%



540,000 artificial distortions

+ 60,000 original

Test error: 0.8%

MNIST data misclassification cases

True label -> Predicted label

Columbia University

Summary: pipeline for deep neural net

forward passing to get predictions

(preprocessed) data → nonlinear activation → hidden layers → nonlinear activation → hidden layers → nonlinear activation → hidden layers → nonlinear activation → ...... → hidden layers → nonlinear activation → predicted values

backward propagation to update parameters

Why training is hard

- Underfitting: use better optimization:
  - use better optimization tools (e.g. batch-normalization, 2nd-order methods).
  - use GPUs, distributed computing.
- Overfitting: use better regularization:
  - unsupervised pre-training
  - stochastic drop-out training
- For many large-scale practical problems, have to scale up:
  - ReLu nonlinearity
  - initialization (e.g. Kaiming He's initialization)
  - stochastic gradient descent
  - momentum, batch-normalization, and drop-out

COLUMBIA UNIVERSITY

Preprocessing

- One-hot representation: class 0 or class 1 → (1,0) or (0,1)
- Normalizing the inputs will speed up training (Lecun et al. 1998)

  - could normalization be useful at the level of the hidden layers?
- Batch normalization is an attempt to do that

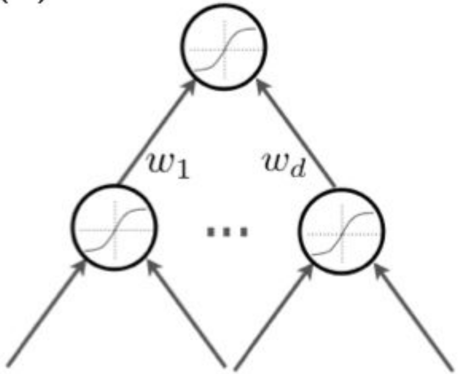
  - each unit's pre-activation is normalized (mean subtraction, stddev division)

  - during training, mean and stddev is computed for each minibatch

  - backpropagation takes into account the normalization

  - at test time, the global mean / stddev is used

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$



Initialization of parameters

- Initialize biases to 0
- For weights
  - Can not initialize weights to 0 with tanh activation
    - ➢ All gradients would be zero (saddle point)
  - Can not initialize all weights to the same value
    - ➢ All hidden units in a layer will always behave the same
    - ➢ Need to break symmetry
  - Sample $\mathbf{W}_{i,j}^{(k)}$ from $U\left[-b, b\right]$, where

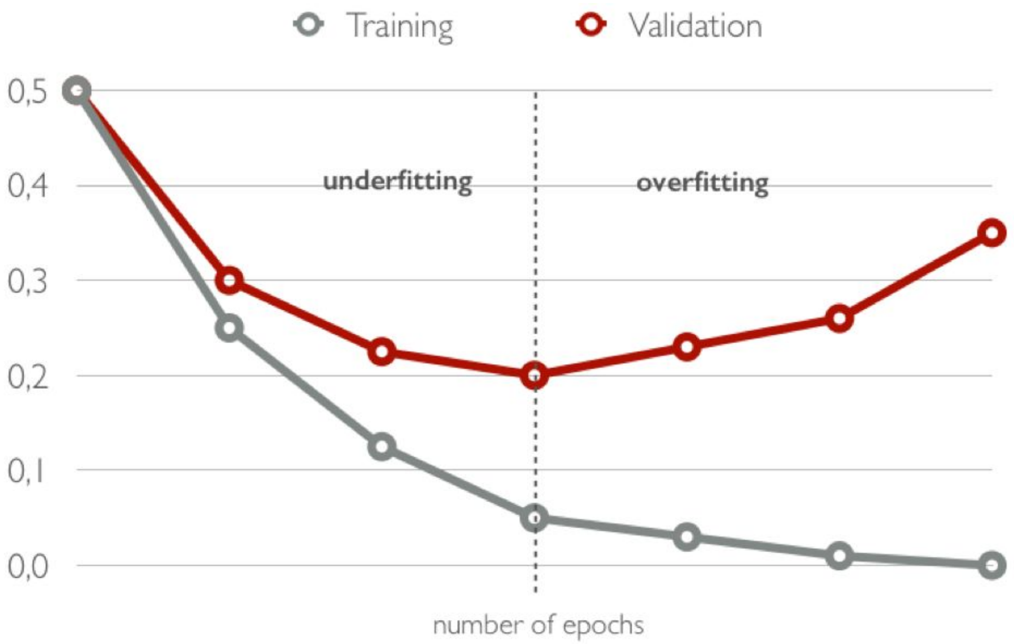
$$b = \frac{\sqrt{6}}{\sqrt{H_k + H_{k-1}}}$$

Sample around 0 and break symmetry

Size of $\mathbf{h}^{(k)}(\mathbf{x})$

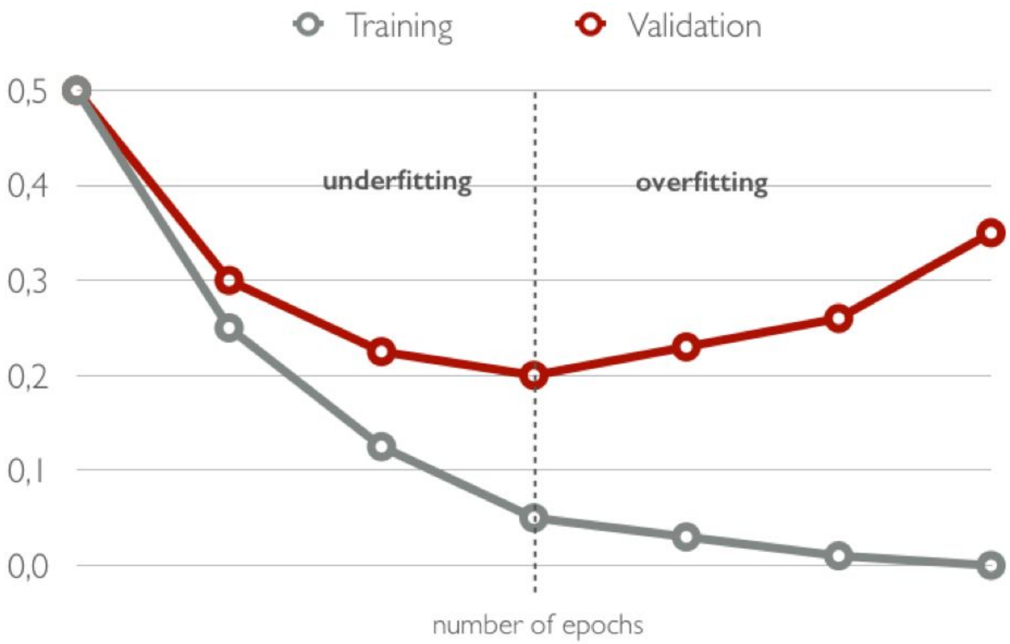
Deep neural network: overfitting

- Overfitting often occurs in applications of neural networks.
- Ways to overcome:
  - Early stopping:
    Stop training process early.
  - Dropout:
    Use random binary masks.



Columbia University

Early stopping



COLUMBIA UNIVERSITY

Dropouts

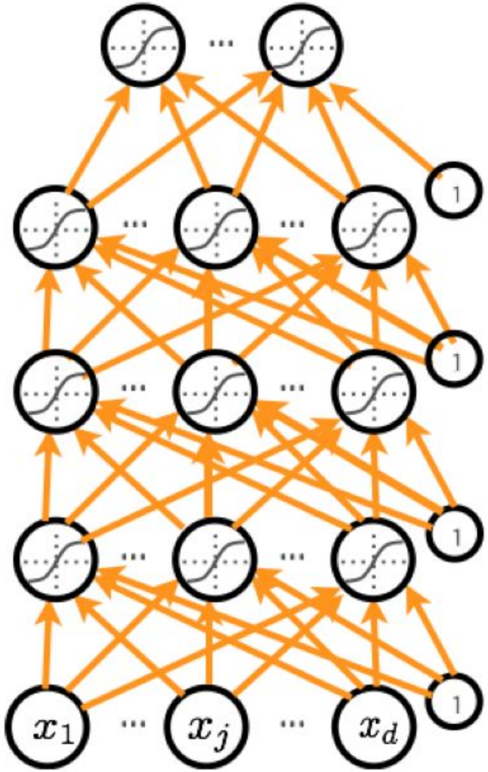
- Cripple neural network by removing hidden units stochastically

  - each hidden unit is set to 0 with probability 0.5

  - hidden units cannot co-adapt to other units

  - hidden units must be more generally useful

- Could use a different dropout probability, but 0.5 usually works well



Columbia University

Model selection

- Training Protocol:
  - Train your model on the Training Set $\mathcal{D}^{\text{train}}$
  - For model selection, use Validation Set $\mathcal{D}^{\text{valid}}$
    - Hyper-parameter search: hidden layer size, learning rate, number of iterations/epochs, etc.
  - Estimate generalization performance using the Test Set $\mathcal{D}^{\text{test}}$
- Remember: Generalization is the behavior of the model on **unseen examples**.

COLUMBIA UNIVERSITY

Optimization

- SGD with momentum, batch-normalization, and dropout usually works very well
- Pick learning rate by running on a subset of the data

  - Start with large learning rate & divide by 2 until loss does not diverge

  - Decay learning rate by a factor of ~100 or more by the end of training

  - Use ReLU nonlinearity

  - Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.
- Use adapted learning rate

Optimization

- SGD with momentum, batch-normalization, and dropout usually works very well
- Pick learning rate by running on a subset of the data

  - Start with large learning rate & divide by 2 until loss does not diverge

  - Decay learning rate by a factor of ~100 or more by the end of training

  - Use ReLU nonlinearity

  - Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.
- Use adapted learning rate

Columbia University

Summary:

- Actively used to model distributed computation in brain
- Highly non-linear regression/classification
- Vector-valued inputs and outputs
- Potentially millions of parameters to estimate - overfitting
- Hidden layers learn intermediate representations – how many to use?

- Prediction – Forward propagation
- Gradient descent (Back-propagation), local minima problems

- Coming back in new form as deep networks

  - Try different/deeper architecture

Columbia University

References

- Christopher Bishop: Pattern Recognition and Machine Learning, Chapter 5
- Ziv Bar-Joseph, Tom Mitchell, Pradeep Ravikumar and Aarti Singh: CMU 10-701
- Ryan Tibshirani: CMU 10-725
- Ruslan Salakhutdinov: CMU 10-703
- https://towardsdatascience.com/neural-network-architectures-156e5bad51ba
- https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Columbia University