Lecture 13:
– Introduction to Deep Learning
A reminder about course projects

- From now on, regular (weekly) blog posts about your progress on the course projects!
- We will use medium.com
"local gradient" $f = Wx$

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Computational Graph:

- $f = Wx$
- $s = \text{scores}$
- $L = \text{hinge loss}$

```
class ComputationalGraph(object):
    def forward(self, inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward(self):
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```
Mini-batch SGD

Loop:
1. **Sample** a batch of data
2. **Forward** prop it through the graph, get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient
This week

- Introduction to Deep Learning
- Deep Convolutional Neural Networks
What is deep learning?


“Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.”

– Yann LeCun, Yoshua Bengio and Geoff Hinton
1943 – 2006:
A Prehistory of Deep Learning
1943: Warren McCulloch and Walter Pitts

- First computational model
- Neurons as logic gates (AND, OR, NOT)
- A neuron model that sums binary inputs and outputs 1 if the sum exceeds a certain threshold value, and otherwise outputs 0

![Image of Warren McCulloch and Walter Pitts](image-url)
1958: Frank Rosenblatt’s Perceptron

- A computational model of a single neuron
- Solves a binary classification problem
- Simple training algorithm
- Built using specialized hardware

1969: Marvin Minsky and Seymour Papert

“No machine can learn to recognize X unless it possesses, at least potentially, some scheme for representing X.” (p. xiii)

- Perceptrons can only represent linearly separable functions.
  - such as XOR Problem

- Wrongly attributed as the reason behind the AI winter, a period of reduced funding and interest in AI research
1990s

- Multi-layer perceptrons can theoretically learn any function (Cybenko, 1989; Hornik, 1991)

- Training multi-layer perceptrons
  - Back-propagation (Rumelhart, Hinton, Williams, 1986)
  - Back-propagation through time (BPTT) (Werbos, 1988)

- New neural architectures
  - Convolutional neural nets (LeCun et al., 1989)
  - Long-short term memory networks (LSTM) (Schmidhuber, 1997)
Why it failed then

• Too many parameters to learn from few labeled examples.
• “I know my features are better for this task”.
• Non-convex optimization? No, thanks.
• Black-box model, no interpretability.

• Very slow and inefficient
• Overshadowed by the success of SVMs (Cortes and Vapnik, 1995)
A major breakthrough in 2006
2006 Breakthrough: Hinton and Salakhutdinov

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

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. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

- The first solution to the vanishing gradient problem.
- Build the model in a layer-by-layer fashion using unsupervised learning
  - The features in early layers are already initialized or “pretrained” with some suitable features (weights).
  - Pretrained features in early layers only need to be adjusted slightly during supervised learning to achieve good results.

The 2012 revolution
ImageNet Challenge

• **ImageNet** Large Scale Visual Recognition Challenge (ILSVRC)
  - **1.2M** training images with **1K** categories
  - Measure top-5 classification error

### ILSVRC 2012 Competition

<table>
<thead>
<tr>
<th>2012 Teams</th>
<th>%Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervision (Toronto)</td>
<td>15.3</td>
</tr>
<tr>
<td>ISI (Tokyo)</td>
<td>26.1</td>
</tr>
<tr>
<td>VGG (Oxford)</td>
<td>26.9</td>
</tr>
<tr>
<td>XRCE/INRIA</td>
<td>27.0</td>
</tr>
<tr>
<td>UvA (Amsterdam)</td>
<td>29.6</td>
</tr>
<tr>
<td>INRIA/LEAR</td>
<td>33.4</td>
</tr>
</tbody>
</table>

- **CNN based**, non-CNN based

#### The success of AlexNet, a deep convolutional network
- 7 hidden layers (not counting some max pooling layers)
- 60M parameters

#### Combined several tricks
- ReLU activation function, data augmentation, dropout
2012 – now
Deep Learning Era
Speech recognition


M.-T. Luong et al., "Effective Approaches to Attention-based Neural Machine Translation", EMNLP 2015

M. Bojarski et al., “End to End Learning for Self-Driving Cars”, In CoRR 2016

D. Silver et al., "Mastering the game of Go with deep neural networks and tree search", Nature 529, 2016

Game Playing

L. Pinto and A. Gupta, “Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours” ICRA 2015

Audio Generation


Self-Driving Cars

M. Ramona et al., "Capturing a Musician's Groove: Generation of Realistic Accompaniments from Single Song Recordings", In IJCAI 2015

And many more...
Why now?
GLOBAL INFORMATION STORAGE CAPACITY
IN OPTIMALLY COMPRESSED BYTES

1986 ANALOG
2.6 EXABYTES

DIGITAL
0.02 EXABYTES

ConvNets Developed

SVMs dominate NIPS

2002 "BEFINTING OF
THE DIGITAL AGE"

2007

ANALOG
19 EXABYTES
- Paper, film, audiotaape and vinyl: 6%
- Analog videotapes (VHS, etc): 94%

DIGITAL

- Portable media, flash drives: 2%
- Portable hard disks: 2.4%
- CDs & Minidisks: 6.8%
- Computer Servers and Mainframes: 8.9%
- Digital Tape: 11.8%
- DVD/Blu-Ray: 22.8%
- PC Hard Disks: 44.5%
- Others: < 1% (incl. Chip Cards, Memory Cards, Floppy Disks, Mobile Phones, PDAs, Cameras/Camcorders, Video Games)

DIGITAL
280 EXABYTES


Slide credit: Neil Lawrence
# Datasets vs. Algorithms

<table>
<thead>
<tr>
<th>Year</th>
<th>Breakthroughs in AI</th>
<th>Datasets (First Available)</th>
<th>Algorithms (First Proposed)</th>
</tr>
</thead>
</table>

**Average No. of Years to Breakthrough:**
- **Datasets:** 3 years
- **Algorithms:** 18 years

Table credit: Quant Quanto
Powerful Hardware

**NVIDIA DGX-1**

WORLD’S FIRST DEEP LEARNING SUPERCOMPUTER

- 170 TFLOPS FP16
- 8x Tesla P100 16GB
- NVLink Hybrid Cube Mesh
- Accelerates Major AI Frameworks
- Dual Xeon
- 7 TB SSD Deep Learning Cache
- Dual 10GbE, Quad IB 100Gb
- 3RU - 3200W

**TITAN X**

THE WORLD’S FASTEST GPU

- 8 Billion Transistors
- 3,072 CUDA Cores
- 7 TFLOPS SP / 0.2 TFLOPS DP
- 12GB Memory

**GOOGLE DATACENTER**

- 1,000 CPU Servers
- 2,000 CPUs * 16,000 cores
- 600 kWatts
- $5,000,000

**STANFORD AI LAB**

- 3 GPU-Accelerated Servers
- 12 GPUs * 14,432 cores
- 4 kWatts
- $33,000

**CPU**

Optimized for Serial Tasks

**GPU Accelerator**

Optimized for Parallel Tasks

Slide credit: NVIDIA
10X GROWTH IN GPU COMPUTING

2008
- 150,000 CUDA Downloads
- 27 CUDA Apps
- 60 Universities Teaching
- 4,000 Academic Papers
- 6,000 Tesla GPUs
- 77 Supercomputing Teraflops

2015
- 3 Million CUDA Downloads
- 319 CUDA Apps
- 800 Universities Teaching
- 60,000 Academic Papers
- 450,000 Tesla GPUs
- 54,000 Supercomputing Teraflops
Working ideas on how to train deep architectures

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different “thinned” networks. At test time, all thinned networks are used in parallel, with each network being used with probability 1/m, where m is the number of thinned networks.

Better Learning Regularization (e.g. Dropout)

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., sioffe@google.com

Christian Szegedy
Google Inc., szegedy@google.com

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as internal covariate shift, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases accelerating the search for optimizers. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 10 times lower training error, and halves the original model size by a significant margin. Using an ensemble of batch sizes, we show that Batch Normalization reduces the expected test error on ImageNet classification, reducing 4.5% top-5 validation error (and 6.6% test error), increasing the accuracy of human raters.

1 Introduction

Deep learning has dramatically advanced the state of the art in vision, speech, and many other areas. Stochastic gradient descent (SGD) has proven to be an effective way of training deep networks, and SGD variants, such as momentum (Robbins and Monro, 1951) and Nesterov momentum (Nesterov, 2009), have been used to achieve state-of-the-art performance. SGD optimizes the parameters \( \theta \) of the network, as we try to minimize the loss

\[
L = \sum_n L(x_n, \theta)
\]

where \( x_n \) is the training data set. With SGD, the training process is prone to get stuck in bad minima, and each step requires us to consider training a whole mini-batch \( x_n \) of size \( m \). The extra cost is not as bad as it sounds: by computing

\[
\hat{E}(x_n, \theta) = \frac{1}{m} \sum_{i=1}^{m} L(x_i, \theta)
\]

we can compute the gradient of the loss function with respect to the parameters, by computing

\[
\nabla \hat{E}(x_n, \theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla L(x_i, \theta)
\]

\[
\text{(for batch size \( m \) and learning rate \( \alpha \) is exactly equivalent to that for a mini-batch size \( m \) with learning rate \( \alpha/m \). Therefore, the input distributions properties that made training more efficient - such as having the mini-batch distributions between the training and test data - apply to training the sub-network as well.)}
\]

Better Optimization Conditioning (e.g. Batch Normalization)
Working ideas on how to train deep architectures

Deep Residual Learning for Image Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unformulated functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error.

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Better neural architectures (e.g. Residual Nets)

So what is deep learning?
Three key ideas

• (Hierarchical) Compositionality

• End-to-End Learning

• Distributed Representations
Three key ideas

• (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

• End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extract

• Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together
Traditional Machine Learning

VISION

This burrito place is yummy and fun!

SPEECH

NLP

This burrito place is yummy and fun!
It’s an old paradigm

- The first learning machine: the Perceptron
  - Built at Cornell in 1960
- The Perceptron was a linear classifier on top of a simple feature extractor
- The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.
- Designing a feature extractor requires considerable efforts by experts.

\[ y = \text{sign} \left( \sum_{i=1}^{N} W_i F_i(X) + b \right) \]
Hierarchical Compositionality

VISION

pixels $\rightarrow$ edge $\rightarrow$ texton $\rightarrow$ motif $\rightarrow$ part $\rightarrow$ object

SPEECH

sample $\rightarrow$ spectral band $\rightarrow$ formant $\rightarrow$ motif $\rightarrow$ phone $\rightarrow$ word

NLP

character $\rightarrow$ word $\rightarrow$ NP/VP/.. $\rightarrow$ clause $\rightarrow$ sentence $\rightarrow$ story
Building A Complicated Function

Given a library of simple functions

\[ \sin(x) \quad \log(x) \]
\[ \cos(x) \quad x^3 \]
\[ \exp(x) \]

Compose into a complicate function
Building A Complicated Function

Given a library of simple functions

\[ f(x) = \sum_{i} \alpha_i g_i(x) \]

Idea 1: Linear Combinations

- Boosting
- Kernels
- …
Building A Complicated Function

Given a library of simple functions

\[ f(x) = g_1(g_2(\ldots (g_n(x) \ldots)) \]

Idea 2: Compositions

- Deep Learning
- Grammar models
- Scattering transforms…
Building A Complicated Function

Given a library of simple functions

\[ \begin{align*}
\sin(x) & \\
\log(x) & \\
\cos(x) & \\
x^3 & \\
\exp(x) &
\end{align*} \]

Idea 2: Compositions

- Deep Learning
- Grammar models
- Scattering transforms…

Compose into a complicate function

\[ f(x) = \log(\cos(\exp(\sin^3(x)))) \]

\[
\begin{array}{c}
\sin(x) \rightarrow x^3 \rightarrow \exp(x) \rightarrow \cos(x) \rightarrow \log(x)
\end{array}
\]
Deep Learning = Hierarchical Compositionality

“car"
Deep Learning = Hierarchical Compositionality

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Sparse DBNs

[Lee et al. ICML '09]

Figure courtesy: Quoc Le
Three key ideas

• (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

• End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extract

• Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together
Traditional Machine Learning

VISION

This burrito place is yummy and fun!

SPEECH

NLP

slide by Marc'Aurelio Ranzato, Yann LeCun
Traditional Machine Learning (more accurately)

VISION

SIFT/HOG → K-Means/pooling → classifier

fixed  unsupervised supervised

“Learned”

“car”

SPEECH

MFCC → Mixture of Gaussians → classifier

fixed  unsupervised supervised

This burrito place is yummy and fun!

NLP

Parse Tree Syntactic → n-grams → classifier

fixed  unsupervised supervised

“+”
Deep Learning = End-to-End Learning

VISION

<table>
<thead>
<tr>
<th>SIFT/HOG</th>
<th>K-Means/pooling</th>
<th>classifier</th>
<th>“car”</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>unsupervised</td>
<td>supervised</td>
<td></td>
</tr>
</tbody>
</table>

SPEECH

<table>
<thead>
<tr>
<th>MFCC</th>
<th>Mixture of Gaussians</th>
<th>classifier</th>
<th>“d ē p”</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>unsupervised</td>
<td>supervised</td>
<td></td>
</tr>
</tbody>
</table>

NLP

This burrito place is yummy and fun!

<table>
<thead>
<tr>
<th>Parse Tree</th>
<th>n-grams</th>
<th>classifier</th>
<th>“+”</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>unsupervised</td>
<td>supervised</td>
<td></td>
</tr>
</tbody>
</table>
Deep Learning = End-to-End Learning

• A hierarchy of trainable feature transforms
  – Each module transforms its input representation into a higher-level one.
  – High-level features are more global and more invariant
  – Low-level features are shared among categories
“Shallow” vs Deep Learning

• “Shallow” models

  hand-crafted Feature Extractor
  fixed

  “Simple” Trainable Classifier
  learned

• Deep models

  Trainable Feature-Transform / Classifier

  Trainable Feature-Transform / Classifier

  Trainable Feature-Transform / Classifier

  Learned Internal Representations

slide by Marc'Aurelio Ranzato, Yann LeCun
Three key ideas

• (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

• End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extract

• Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together
Localist representations

• The simplest way to represent things with neural networks is to dedicate one neuron to each thing.
  - Easy to understand.
  - Easy to code by hand
    • Often used to represent inputs to a net
  - Easy to learn
    • This is what mixture models do.
    • Each cluster corresponds to one neuron
  - Easy to associate with other representations or responses.

• But localist models are very inefficient whenever the data has componential structure.
Distributed Representations

- Each neuron must represent something, so this must be a local representation.

- **Distributed representation** means a many-to-many relationship between two types of representation (such as concepts and neurons).
  - Each concept is represented by many neurons
  - Each neuron participates in the representation of many concepts

Local  \[ \begin{array}{c}
\bullet \quad \bullet \quad \bigcirc \quad \bullet \end{array} \Rightarrow \text{VR} + \text{HR} + \text{HE} = ? \]

Distributed \[ \begin{array}{c}
\bullet \quad \bullet \quad \bigcirc \quad \bullet \end{array} \Rightarrow \text{V} + \text{H} + \text{E} \approx \bigcirc \]
Power of distributed representations!

Scene Classification

- Objects
- Scene attributes
- Object parts
- Textures

• Possible internal representations:

Next Lecture:
Convolutional Neural Networks