I Robot
Isaac Asimov
1950
Future Vision

Eye Robot
CSCI 1430
2017 MWF 1PM
Computer Vision
Wow so misclassified false positives

what class no good filtr

so misclassified

cool kernel
Goals

Build a classifier which is more powerful at representing complex functions \textit{and} more suited to the learning problem.

What does this mean?

1. Assume that the \textit{underlying data generating function} relies on a composition of factors.
2. Learn a feature representation that is specific to the dataset.
Neural Networks

• Basic building block for composition is a *perceptron* (Rosenblatt c.1960)

• Linear classifier – vector of weights $w$ and a ‘bias’ $b$

\[
\begin{align*}
\mathbf{w} &= (w_1, w_2, w_3) \\
b &= 0.3
\end{align*}
\]

\[
\text{output} = \begin{cases} 
0 & \text{if } w \cdot x + b \leq 0 \\
1 & \text{if } w \cdot x + b > 0
\end{cases}
\]

\[w \cdot x \equiv \sum_j w_j x_j\]
Mark 1 Perceptron
c.1960

20x20 pixel camera feed
Universality

A single-layer network can learn any function:
  • So long as it is differentiable
  • To some approximation;
    More perceptrons = a better approximation

Visual proof (Michael Nielson):
If a single-layer network can learn any function... 
...given enough parameters...

...then why do we go deeper?

Intuitively, composition is efficient because it allows reuse.

Empirically, deep networks do a better job than shallow networks at learning such hierarchies of knowledge.
Composition

Layers that are in between the input and the output are called *hidden layers*, because we are going to *learn* their weights via an optimization process.
Interpretation of many layers

[0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 …] truck feature

Exponentially more efficient than a 1-of-N representation (à la k-means)
Interpretation

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \ldots
\end{bmatrix}
\]

motorbike

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & \ldots
\end{bmatrix}
\]

truck
Interpretation

- prediction of class
- distributed representations
- feature sharing
- compositionality

NOTE: Not actually the weights; a demonstrative visualization!

Lee et al. “Convolutional DBN's …” ICML 2009
Activation functions:
Rectified Linear Unit

- ReLU \( f(x) = \max(0, x) \)
Neural Networks: example

\[x\rightarrow max(0, W^1 x) \rightarrow h^1 \rightarrow max(0, W^2 h^1) \rightarrow h^2 \rightarrow W^3 h^2 \rightarrow o\]

- **x** input
- **h^1** 1-st layer hidden units
- **h^2** 2-nd layer hidden units
- **o** output

Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).
Does anyone pass along the weight without an activation function?

No – this is linear chaining.
"Does anyone pass along the weight without an activation function?"

No – this is linear chaining.
What is the relationship between SVMs and perceptrons?

SVMs attempt to learn the support vectors which maximize the margin between classes.
What is the relationship between SVMs and perceptrons?

SVMs attempt to learn the support vectors which maximize the margin between classes.

A perceptron does not.
Both of these perceptron classifiers are equivalent.

‘Perceptron of optimal stability’ is used in SVM:

Perceptron  
+ optimal stability  
+ kernel trick  
= foundations of SVM
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
Images as input to neural networks
Images as input to neural networks

Example: 200x200 image
40K hidden units

~2B parameters!!!
Images as input to neural networks

Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
Motivation

• Sparse interactions – *receptive fields*
  • Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  • Composition of layers will expand local -> global.
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).
STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).
Motivation

• Sparse interactions – *receptive fields*
  • Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  • Composition of layers will expand local -> global.

• Parameter sharing
  • ‘Tied weights’ – use same weights for more than one perceptron in the neural network.
  • Leads to *equivariant representation*
    • If input changes (e.g., translates), then output changes similarly
Share the same parameters across different locations (assuming input is stationary):
Filtering reminder:
Correlation (rotated convolution)

\[ h[m, n] = \sum_{k,l} f[k, l] I[m + k, n + l] \]

Credit: S. Seitz
Convolutional Layer

Perceptron: \( \text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \)

\[ w \cdot x \equiv \sum_j w_j x_j \]

This is convolution!

Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

\[
\begin{pmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1
\end{pmatrix}
\]

Shared weights
Convolutional Layer

Learn multiple filters.
Filter = ‘local’ perceptron. Also called kernel.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters
Interpretation

- distributed representations
- feature sharing
- compositionality

high-level parts

mid-level parts

low-level parts

Input image

Lee et al. “Convolutional DBN's ...” ICML 2009
Convolutional Layer

\[ h_j^n = \max(0, \sum_{k=1}^{K} h_{k}^{n-1} \ast w_{kj}^n) \]

- \( n \) = layer number
- \( K \) = kernel size
- \( j \) = # channels (input) or # filters (depth)
$h_j^n = \max (0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n)$
Convolutional Layer

\[ h^n_j = \max \left( 0, \sum_{k=1}^{K} h^{n-1}_k \ast w^n_{kj} \right) \]

output feature map

input feature map

kernel
Stride = 1
Stride = 1
Stride = 3
Stride = 3
Stride = 3
Stride = 3
Pooling Layer

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
By *pooling* responses at different locations, we gain robustness to the exact spatial location of image features.
Pooling is similar to downsampling...except sometimes we don’t want to blur, as other functions might be better for classification.
Pooling Layer: Receptive Field Size

$h^{n-1}$ \rightarrow \text{Conv. layer} \rightarrow h^n \rightarrow \text{Pool. layer} \rightarrow h^{n+1}$
Pooling Layer: Examples

Max-pooling:

\[ h^n_j(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]

Average-pooling:

\[ h^n_j(x, y) = \frac{1}{K} \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]
Max pooling

Single depth slice

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

X

Y

6

8

3

4

Wikipedia
Pooling Layer: Examples

Max-pooling:

\[ h^n_j(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]

Average-pooling:

\[ h^n_j(x, y) = \frac{1}{K} \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]

L2-pooling:

\[ h^n_j(x, y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y})^2} \]

L2-pooling over features:

\[ h^n_j(x, y) = \sqrt{\sum_{k \in N(j)} h^{n-1}_k(x, y)^2} \]
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)
Pooling Layer: Receptive Field Size

If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$
Local Contrast Normalization
Local Contrast Normalization

We want the same response.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

N(x,y) = model pixel values in window as a normal distribution

m = mean
\( \sigma = \) variance

**Note:** computational cost is negligible w.r.t. conv. layer.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

Performed also across features and in the higher layers.

Effects:

- improves invariance
- improves optimization
- increases sparsity

Note: computational cost is negligible w.r.t. conv. layer.
ConvNets: Typical Architecture

One stage (zoom)

Whole system

Input Image → 1st stage → 2nd stage → 3rd stage → Fully Conn. Layers → Class Labels
ConvNets: Typical Architecture

Whole system

Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM
Lazebnik et al. “...Spatial Pyramid Matching...” CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM
Yann LeCun’s MNIST CNN architecture
Convolutions: More detail

32x32x3 image

32 height
32 width
3 depth
Convolutions: More detail

32x32x3 image

5x5x3 filter
Convolutions: More detail

Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolutions: More detail

For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolutions: More detail

- CONV, ReLU
  - e.g. 6 5x5x3 filters
- CONV, ReLU
  - e.g. 10 5x5x6 filters
- CONV, ReLU

Andrej Karpathy
Convolutions: More detail

Output size: \((N - F) / \text{stride} + 1\)
Our connectomics diagram

Auto-generated from network declaration by nolearn (for Lasagne / Theano)

Input
75x75x4

Conv 1
3x3x4
64 filters
Max pooling
2x2 per filter

Conv 2
3x3x64
48 filters
Max pooling
2x2 per filter

Conv 3
3x3x48
48 filters
Max pooling
2x2 per filter

Conv 4
3x3x48
48 filters
Max pooling
2x2 per filter
Reading architecture diagrams

Layers
- Kernel sizes
- Strides
- # channels
- # kernels
- Max pooling
AlexNet diagram (simplified)

Input size
227 x 227 x 3

Conv 1
11 x 11 x 3
Stride 4
96 filters

Conv 2
5 x 5 x 48
Stride 1
256 filters

Conv 3
3 x 3 x 256
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
256 filters

Krizhevsky et al. 2012
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

Ciresan et al. “MCDNN for image classification” CVPR 2012
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014
CONV NETS: EXAMPLES

- Scene Parsing

Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013
Pinheiro et al. “Recurrent CNN for scene parsing” arxiv 2013
CONV NETS: EXAMPLES

- Segmentation 3D volumetric images

Ciresan et al. “DNN segment neuronal membranes...” NIPS 2012
Turaga et al. “Maximin learning of image segmentation” NIPS 2009
CONV NETS: EXAMPLES

- Object detection

Szegedy et al. “DNN for object detection” NIPS 2013
CONV NETS: EXAMPLES

- Face Verification & Identification

Dataset: ImageNet 2012

Deng et al. “Imagenet: a large scale hierarchical image database” CVPR 2009
Architecture for Classification

Total nr. params: 60M

- 4M \(\text{LINEAR}\)
- 16M \(\text{FULLY CONNECTED}\)
- 37M \(\text{FULLY CONNECTED}\)
- 442K \(\text{MAX POOLING}\)
- 1.3M \(\text{CONV}\)
- 1.3M \(\text{CONV}\)
- 884K \(\text{CONV}\)
- 307K \(\text{MAX POOLING}\)
- 307K \(\text{LOCAL CONTRAST NORM}\)
- 35K \(\text{CONV}\)
- 35K \(\text{LOCAL CONTRAST NORM}\)
- input

Total nr. flops: 832M

- 4M
- 16M
- 37M
- 74M
- 224M
- 149M
- 223M
- 105M

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Results: ILSVRC 2012

**TASK 1 - CLASSIFICATION**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>15</td>
</tr>
<tr>
<td>SIFT+FV</td>
<td>25</td>
</tr>
<tr>
<td>SVM1</td>
<td>27</td>
</tr>
<tr>
<td>SVM2</td>
<td>30</td>
</tr>
<tr>
<td>NCM</td>
<td>33</td>
</tr>
</tbody>
</table>

**TASK 2 - DETECTION**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>30</td>
</tr>
<tr>
<td>DPM-SVM1</td>
<td>50</td>
</tr>
<tr>
<td>DPM-SVM2</td>
<td>55</td>
</tr>
</tbody>
</table>

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Phew!

- Friday:
  - Network training
More ConvNet explanations

- https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/