Topological Data Analysis to understand Convolutional Neural Networks

Aleksei Prokopev, SeoulAI, 2018
Shape
Topology
A solid 2-dimensional blob

\[ \beta_0 = 1 \]
\[ \beta_{i>0} = 0 \]

A 2D blob with three holes

\[ \beta_0 = 1 \]
\[ \beta_1 = 3 \]
\[ \beta_{i>1} = 0 \]

A sphere

\[ \beta_0 = 1 \]
\[ \beta_1 = 0 \]
\[ \beta_2 = 1 \]
\[ \beta_{i>2} = 0 \]

A torus

\[ \beta_0 = 1 \]
\[ \beta_1 = 2 \]
\[ \beta_2 = 1 \]
\[ \beta_{i>2} = 0 \]
Topological Data Analysis
Metric data set → Build geometric filtered complex on top of data → Filtered simplicial complex → Compute persistent homology of the complex.

Signature: persistence diagram

- 0-dimensional homology
- 1-dimensional homology
Refine your data into knowledge with Topological Data Analysis

AYASDI
Examples
Nodes are groups of similar data points

Edges connect similar nodes

Colors let you see values of interest

Position of a node on the screen doesn’t matter
Case study: Yelp Dataset Challenge

Result comparison: TDA with other techniques

Topological Data Analysis (275 sec)

PCA (0.19 sec)

Spectral Embedding (806 sec)

Modified LLE (1206 sec)

LLE (366 sec)
Case study: Netflix competition

Result comparison: TDA with other techniques
Convolutional Neural Networks
Convolution
<table>
<thead>
<tr>
<th>35</th>
<th>40</th>
<th>41</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>40</td>
<td>42</td>
<td>46</td>
<td>52</td>
</tr>
<tr>
<td>42</td>
<td>46</td>
<td>50</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>48</td>
<td>52</td>
<td>56</td>
<td>58</td>
<td>60</td>
</tr>
<tr>
<td>56</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
</tr>
</tbody>
</table>

\[ \times \]

\[ \begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \]

\[ = \]

\[ \begin{array}{ccc} & & \text{42} \\ & & \end{array} \]
<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Gaussian Blur</th>
<th>Sharpen</th>
<th>Edge Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix</strong></td>
<td><img src="image1.png" alt="Matrix Image" /></td>
<td><img src="image2.png" alt="Matrix Image" /></td>
<td><img src="image3.png" alt="Matrix Image" /></td>
<td><img src="image4.png" alt="Matrix Image" /></td>
</tr>
<tr>
<td><strong>Images</strong></td>
<td><img src="image5.png" alt="Image 1" /></td>
<td><img src="image6.png" alt="Image 2" /></td>
<td><img src="image7.png" alt="Image 3" /></td>
<td><img src="image8.png" alt="Image 4" /></td>
</tr>
</tbody>
</table>
Weights
Problems
TDA for CNN
On the Local Behavior of Spaces of Natural Images

Gunnar Carlsson · Tigran Ishkhanov · Vin de Silva · Afra Zomorodian

Received: 19 May 2006 / Accepted: 27 March 2007 / Published online: 30 June 2007
© Springer Science+Business Media, LLC 2007
**Fig. 4** Klein bottle representation as a rectangle with opposite edges identified

**Fig. 5** The 'three circle' space

**Fig. 6** $3 \times 3$ patches parametrized by the Klein bottle

**Fig. 7** PLEX results for $X(15, 30)$
Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks

By Gunnar Carlsson
June 21, 2018

ARTIFICIAL INTELLIGENCE, MACHINE INTELLIGENCE, MACHINE LEARNING, TOPOLOGY

TLDR: Neural Networks are powerful but complex and opaque tools. Using Topological Data Analysis, we can describe the functioning and learning of a convolutional neural network in a compact and understandable way. The implications of the findings are profound and will accelerate the development of a wide range of applications from self-driving cars and drones to complying with things like GDPR.
MAPPER IV

B Coloring by filter value

C Binning by filter value

D Clustering and network construction

Filter Range: [0-4.2]
Interval Length: 1
Overlap: 20%
Going Deeper: Understanding How Convolutional Neural Networks Learn Using TDA

By Gunnar Carlsson
August 9, 2018

ARTIFICIAL INTELLIGENCE, MACHINE INTELLIGENCE, MACHINE LEARNING, TOPOLOGY

In my earlier post I discussed how performing topological data analysis on the weights learned by convolutional neural nets (CNN's) can give insight into what is being learned and how it is being learned.
Mathematical Acceleration: Incorporating Prior Information to Make Neural Nets Learn 3.5X Faster

By GunnarCarlsson

August 30, 2018

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, TOPOLOGY
<table>
<thead>
<tr>
<th>Validation Accuracy</th>
<th># Batch iterations Boosted</th>
<th># Batch iterations standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>.8</td>
<td>187</td>
<td>293</td>
</tr>
<tr>
<td>.9</td>
<td>378</td>
<td>731</td>
</tr>
<tr>
<td>.95</td>
<td>1046</td>
<td>2052</td>
</tr>
<tr>
<td>.96</td>
<td>1499</td>
<td>2974</td>
</tr>
<tr>
<td>.97</td>
<td>2398</td>
<td>4528</td>
</tr>
<tr>
<td>.98</td>
<td>5516</td>
<td>8802</td>
</tr>
<tr>
<td>.985</td>
<td>9584</td>
<td>16722</td>
</tr>
<tr>
<td>.25</td>
<td>303</td>
<td>1148</td>
</tr>
<tr>
<td>.5</td>
<td>745</td>
<td>2464</td>
</tr>
<tr>
<td>.75</td>
<td>1655</td>
<td>5866</td>
</tr>
<tr>
<td>.8</td>
<td>2534</td>
<td>8727</td>
</tr>
<tr>
<td>.83</td>
<td>4782</td>
<td>13067</td>
</tr>
<tr>
<td>.84</td>
<td>6312</td>
<td>15624</td>
</tr>
<tr>
<td>.85</td>
<td>8426</td>
<td>21009</td>
</tr>
</tbody>
</table>
Thank you!
Weapon of choice
The GUDHI library is a generic open source C++ library, with a Python interface, for Topological Data Analysis (TDA) and Higher Dimensional Geometry Understanding. The library offers state-of-the-art data structures and algorithms to construct simplicial complexes and compute persistent homology.

The library comes with data sets, demos, examples and test suites.

The GUDHI library is developed as part of the GUDHI project supported by the European Research Council.

**NEW RELEASE**

**GUDHI version 2.2.0**

As a major new feature, the GUDHI library now offers a Čech complex module, a sparse version of the Rips complex and a utility to build the Rips complex from a correlation matrix (no Python interface yet).

**More Articles**

- New release - GUDHI version 2.1.0 Debian package
- New release - GUDHI version 2.1.0
- New release - GUDHI version 2.0.1
GUDHI
Geometric Understanding in Higher Dimensions

Point cloud

Geometric filtered complex ➔ Data structure ➔ Persistent homology ➔ Persistence intervals

Distance matrix

Bar code ➔ Diagram ➔ Betti numbers ➔ Bottleneck distance ➔ Statistic tools ➔ ...

Inria
Filtered simplicial complexes – Simplex tree

- Memory and time-efficient data structure to store simplicial complexes.
- Every simplex is a word stored in the tree.
- The nodes corresponding to simplices of the same dimension having the same maximal vertex are stored in a cyclic list.
- It is a base of all algorithms to compute persistence of weighted simplicial complexes in GUDHI.

by Clément Maria
GUDHI

Geometric Understanding in Higher Dimensions

Geometric filtered complex – Rips from a point cloud

by Clément Maria
Project description

See the project home page http://danifold.net/mapper for a detailed description and documentation.

This package features both a GUI and a Python package for custom scripts. The Python package itself works with Python 2 and 3. The GUI, however, depends on wxPython, which is available for Python 2 only. Therefore, the setup script will install the GUI only if it is executed by Python 2.

See also https://pypi.python.org/pypi/cmapptools for the companion package with fast C++ algorithms.

The authors of Python mapper are Daniel Müllner and Aravindakshan Babu. (PyPI apparently suppresses everything but the first name in the “author” field, hence only one author is displayed below.)
MAPPER IV

B Coloring by filter value

C Binning by filter value

D Clustering and network construction

Filter Range: [0-4.2]
Interval Length: 1
Overlap: 20%
```python
model.summary()

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d_5 (Conv2D)</td>
<td>(None, 64, 26, 26)</td>
<td>640</td>
</tr>
<tr>
<td>max_pooling2d_5 (MaxPooling2)</td>
<td>(None, 32, 13, 26)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_6 (Conv2D)</td>
<td>(None, 16, 12, 25)</td>
<td>2064</td>
</tr>
<tr>
<td>max_pooling2d_6 (MaxPooling2)</td>
<td>(None, 8, 6, 25)</td>
<td>0</td>
</tr>
<tr>
<td>flatten_3 (Flatten)</td>
<td>(None, 1200)</td>
<td>0</td>
</tr>
<tr>
<td>dense_3 (Dense)</td>
<td>(None, 10)</td>
<td>12010</td>
</tr>
</tbody>
</table>

Total params: 14,714
Trainable params: 14,714
Non-trainable params: 0
```
Train on 60000 samples, validate on 10000 samples

Epoch 1/8
60000/60000 [==============================] - 3s 53us/step - loss: 0.4136 - acc: 0.8756 - val_loss: 0.1806 - val_acc: 0.9449
Epoch 2/8
60000/60000 [==============================] - 3s 47us/step - loss: 0.1528 - acc: 0.9548 - val_loss: 0.1083 - val_acc: 0.9674
Epoch 3/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.1086 - acc: 0.9673 - val_loss: 0.0814 - val_acc: 0.9741
Epoch 4/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0862 - acc: 0.9739 - val_loss: 0.0671 - val_acc: 0.9795
Epoch 5/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0702 - acc: 0.9788 - val_loss: 0.0622 - val_acc: 0.9797
Epoch 6/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0605 - acc: 0.9822 - val_loss: 0.0599 - val_acc: 0.9828
Epoch 7/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0540 - acc: 0.9837 - val_loss: 0.0562 - val_acc: 0.9826
Epoch 8/8
60000/60000 [==============================] - 3s 47us/step - loss: 0.0486 - acc: 0.9850 - val_loss: 0.0528 - val_acc: 0.9829

Test loss: 0.05276332234479487
Test accuracy: 0.9829
HOW MANY NODES?
Ok
<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d_1</td>
<td>(None, 4096, 26, 26)</td>
<td>40960</td>
</tr>
<tr>
<td>max_pooling2d_1</td>
<td>(None, 2048, 13, 26)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_2</td>
<td>(None, 16, 12, 25)</td>
<td>131088</td>
</tr>
<tr>
<td>max_pooling2d_2</td>
<td>(None, 8, 6, 25)</td>
<td>0</td>
</tr>
<tr>
<td>flatten_1</td>
<td>(None, 1200)</td>
<td>0</td>
</tr>
<tr>
<td>dense_1</td>
<td>(None, 10)</td>
<td>12010</td>
</tr>
</tbody>
</table>

Total params: 184,058
Trainable params: 184,058
Non-trainable params: 0
Train on 60000 samples, validate on 10000 samples
Epoch 1/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.2617 - acc: 0.9183 - val_loss: 0.0865
val_acc: 0.9728
Epoch 2/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0858 - acc: 0.9737 - val_loss: 0.0637
val_acc: 0.9806
Epoch 3/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.6658 - acc: 0.9864 - val_loss: 0.0517
val_acc: 0.9843
Epoch 4/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0547 - acc: 0.9827 - val_loss: 0.0522
val_acc: 0.9826
Epoch 5/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0481 - acc: 0.9846 - val_loss: 0.0650
val_acc: 0.9891
Epoch 6/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0436 - acc: 0.9868 - val_loss: 0.0460
val_acc: 0.9845
Epoch 7/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.6376 - acc: 0.9882 - val_loss: 0.0539
val_acc: 0.9841
Epoch 8/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0347 - acc: 0.9891 - val_loss: 0.0480
val_acc: 0.9854
Epoch 9/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0310 - acc: 0.9900 - val_loss: 0.0512
val_acc: 0.9850
Epoch 10/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.6296 - acc: 0.9903 - val_loss: 0.0621
val_acc: 0.9814
Epoch 11/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0264 - acc: 0.9913 - val_loss: 0.0484
val_acc: 0.9857
Epoch 12/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0247 - acc: 0.9916 - val_loss: 0.0486
val_acc: 0.9863
Epoch 13/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0229 - acc: 0.9923 - val_loss: 0.0623
val_acc: 0.9821
Epoch 14/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0209 - acc: 0.9935 - val_loss: 0.0592
val_acc: 0.9846
Epoch 15/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0200 - acc: 0.9933 - val_loss: 0.0719
val_acc: 0.9824
Epoch 16/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0176 - acc: 0.9944 - val_loss: 0.0634
val_acc: 0.9839
Test loss: 0.9034
Test accuracy: 0.9839
MEMORY ERROR!!!
Does it make sense?
Challenges

- Mapper is confusing, too many parameters to tune
- Computations are very memory extensive
- Requires sophisticated preprocessing
- Toolkits are not perfect
Still very promising!
Why TDA?

- No good understanding what is happening inside Neural Networks, despite of abundance of good research done by very smart people
- Intellectually satisfying and intuitive
- Terra incognita
Questions to ask

- How topology changes over layers?
- How topology changes over training?
- Do different nets have the same underlying structures?
- What do this structures mean?
- ...
Further research

- CNNs:
  - Do they have the same structure?
  - What happens when overfit?
  - How topology of learned weights depends on topology of training data
  - ...

- RNNs:
  - What do the cycles mean?
Thank you!
Aleksei Prokopev

aleksei@akaintelligence.com

+82 10 3742 3945