Capsule Network with Routing Mechanism
Part 2: Matrix Capsule & EM Routing

- Ou Changkun
- Email: hi@changkun.us
- Homepage: https://changkun.de
Agenda

1. Recap & Matrix Capsule Network
   ○ (Vector) Capsules
   ○ Dynamic Routing by Agreement
   ○ (Recap: Capsule blueprint & Routing by Agreement
   ○ Matrix) Capsules Blueprint

2. Routing Mechanism and Unsupervised Clustering
   ○ Dynamic Routing & k-Mean
   ○ GMM & EM Routing & Gaussian Mixture Model

3. Experiments
   ○ smallNORB classification task
   ○ Adversarial examples
(Vector) Capsules blueprint

- "A capsule is a group of neurons whose output represents different properties of the same entity."

- General ideas differ from [Sabour et al. 2017]:
  - Vector $\rightarrow$ Matrix
  - Activity Vector $\rightarrow$ Pose Matrix + Activity Probability

Dynamic **Routing** *(by Agreement)*

\[ \| \mathbf{v} \| \text{ is confidence} \]

Routing Algorithm

Initialize \( b_{11}, b_{21} = 0 \)

\[
\text{for } r \text{ in range}(1 \ldots T) \\
c_{1r}, c_{2r} = \text{softmax}(b_{1r}, b_{2r}) \\
a_r = \text{squashing}(c_{1r} u_1 + c_{2r} u_2) \\
b_{1(r+1)} = b_{1r} + a_r \cdot u_1 \\
b_{2(r+1)} = b_{2r} + a_r \cdot u_2
\]

Here: \( T=2 \)
Matrix Capsule Network Blueprint

Layer L

Vote matrix

Non-linear Routing Procedure

Pose matrix $M_{4 \times 4}^{(i)}$

Activation $a^{(i)}$

Vote matrix $M_{4 \times 4}^{(k)}$

Activation $a^{(k)}$

Layer (L+1)

Vote matrix $V_{4 \times 4}^{(ij)}$

Pose matrix $V_{4 \times 4}^{(kj)}$

Activation $a^{(i)}$

Activation $a^{(k)}$

Non-linear Routing Procedure

[ Ou Changkun © 2018 ]
Routing by EM Clustering (GMM)

Layer L

- Pose matrix $M^{(i)}_{4 \times 4}$
- Activation $a^{(i)}$

Layer (L+1)

- Vote matrix $V^{(i,j)}_{4 \times 4}$
- Activation $a^{(i)}$
- Non-linear Routing Procedure
- Pose matrix
- Activation

- Pose matrix $M^{(k,j)}_{4 \times 4}$
- Activation $a^{(k)}$

Activation $a^{(k)}$

Image source from wikipedia "EM Algorithm"

[ Ou Changkun © 2018 ]
Architecture: Matrix Capsule

Tensor Version:

Experiments: smallNORB

<table>
<thead>
<tr>
<th>Routing iterations</th>
<th>Pose structure</th>
<th>Loss</th>
<th>Coordinate Addition</th>
<th>Test error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Matrix</td>
<td>Spread</td>
<td>Yes</td>
<td>9.7%</td>
</tr>
<tr>
<td>2</td>
<td>Matrix</td>
<td>Spread</td>
<td>Yes</td>
<td>2.2%</td>
</tr>
<tr>
<td>3</td>
<td>Matrix</td>
<td>Spread</td>
<td>Yes</td>
<td><strong>1.8%</strong></td>
</tr>
<tr>
<td>5</td>
<td>Matrix</td>
<td>Spread</td>
<td>Yes</td>
<td>3.9%</td>
</tr>
<tr>
<td>3</td>
<td>Vector</td>
<td>Spread</td>
<td>Yes</td>
<td>2.9%</td>
</tr>
<tr>
<td>3</td>
<td>Matrix</td>
<td>Spread</td>
<td>No</td>
<td>2.6%</td>
</tr>
<tr>
<td>3</td>
<td>Vector</td>
<td>Spread</td>
<td>No</td>
<td>3.2%</td>
</tr>
<tr>
<td>3</td>
<td>Matrix</td>
<td>Margin¹</td>
<td>Yes</td>
<td>3.2%</td>
</tr>
<tr>
<td>3</td>
<td>Matrix</td>
<td>CrossEnt</td>
<td>Yes</td>
<td>5.8%</td>
</tr>
</tbody>
</table>

Baseline CNN with 4.2M parameters: 5.2%

CNN of Cireșan et al. (2011) with extra input images & deformations: 2.56%

Our Best model (third row), with multiple crops during testing: **1.4%**

**Open Source Implementation:**
- CNN baseline (4.2M): 88.7%(best)/94.8%(paper)
- Matrix Cap with EM routing (310K, 2 iteration): 91.8%(best)/98.6%(paper)
- [https://github.com/www0wwwjs1/Matrix-Capsules-EM-Tensorflow](https://github.com/www0wwwjs1/Matrix-Capsules-EM-Tensorflow)

Experiments: smallNORB

Experiments: Adversarial Robustness

*BIM & FGSM are methods for creating adversarial examples

https://arxiv.org/abs/1412.6572

Summary of Matrix CapsNet

- Key Points of Matrix Capsule:
  - (Matrix, Activation) → (Matrix, Activation)
  - Encapsulate entity or its pattern
  - Routing by agreement Mechanism
  - ...

- Pros:
  - Equivariance
  - Built-in interpretability
  - Adversarial robustness

- Cons:
  - Reproducibility
  - Computational Performance
  - Routing process
  - ...

References of this Section