Stochastic Neighbor Compression

Motivation: Nearest Neighbor Rule
+ easy to implement
+ natural multiclass
+ trivial to train
- expensive to test
- expensive to store

Optimization Details
Define auxiliary matrices,
\[
\begin{align*}
Q &= \{q_{ij} \} = \begin{bmatrix} q_{11} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots \\ q_{m1} & \cdots & q_{mn} \end{bmatrix} \\
\end{align*}
\]
Gradient with respect to \( z \)
\[
\frac{\partial C(Z, A)}{\partial z} = 4A^T (Q - P) - \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} (x_i - z)(x_j - z)
\]

1. reduce dimensionality
   - Tenenbaum et al., 2000
   - Hinton & Roweis, 2002
   - Weinberger et al., 2004
   - van der Maaten & Hinton, 2008
   - Weinberger & Saul, 2009
2. reduce distance computations
   - Hart, 1968
   - Bempeji & Cenobresky, 1999
   - Byegzimer et al., 2006
   - Garcia et al., 1999
   - Mulicka et al., 2002
   - Angluin, 2005
3. reduce instances
   - Probably useful in large dataset reduction
   - Byegzimer et al., 2006
   - Garcia et al., 1999
   - Mulicka et al., 2002
   - Angluin, 2005

Main Idea
• Select to preserve class balance
• Sample compressed training inputs:
\( x_1 \ldots x_n \)

Figure 1 Dimensionality reduction before and after applying LNN. A distance metric \( A \in \mathbb{R}^{n \times n} \) is optimized, where \( r < c \) is the reduced dimensionality and \( A_{rc} = (x_r - x_c) \) is the new distance, so that neighbors of the same class lie closer than those of other classes.

Step 1. Dimensionality Reduction (Optional)

Step 2. Subsampling
- Sample \( m \) compressed training inputs: \( \{ x_1 \ldots x_m \} \)
- Select to preserve class balance
- For each compressed input \( x_i \), such that \( x_i = x_j \) fix its label as such, \( y_i = y_j \)
- Compression results are stable across different random samples

Step 3. Learning Compressed Inputs
Goal: Learn a compressed set \( \{ x_1 \ldots x_m \} \) that predicts training set \( \{ x_1 \ldots x_n \} \) correctly

Training Complexity
\[
\begin{align*}
\text{computation} & \quad \text{computation} \\
\text{complextiy} & \quad \text{complextiy} \\
\text{total complexity} & \quad \text{total complexity} \\
\end{align*}
\]

Datasets and Training Time

Results

Parameter Sensitivity

Label Noise Sensitivity

Figure 4 KNN test error rates with various data set reduction methods on the letters dataset under artificial label noise. The figure shows clearly that the KNN error increases approximately linearly with label noise. SNC with 2%, 4%, 8% compression seems to smooth out missclassified inputs and yields a significantly more robust KNN classifier. In contrast, CKN, FCNN and also subsampling (not shown in the figure to reduce clutter) do not mitigate the effect of label noise and at times tend to even amplify the test error.

References


Acknowledgements

@Washington University in St. Louis

Machine Learning

Stochastic Neighbor Compression

Matt J. Kusner, Stephen Tyree, Kilian Q. Weinberger, Kunal Agrawal
Department of Computer Science & Engineering, Washington University in St. Louis, USA

Figure 2 YaleFaces before and after compression

Figure 3 The decision rule and SNC data set (white circles) learned from 2D USPS digits under varying \( A = 1^{-r} \)

Test-time Speed-up

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
<td>264.8 s</td>
</tr>
<tr>
<td>SVHN</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
</tr>
<tr>
<td>SVHN</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
</tr>
<tr>
<td>SVHN</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
<td>166.6 s</td>
</tr>
</tbody>
</table>

Table 3: Left: Speed-up of KNN testing through SNC compression without a data structure in black (on top of bag-trains) and LSH (in red). Results where SNC matches or exceeds the accuracy of full KNN (up to statistical significance) are in bold. Right: Speed-up of SNC at 4% compression versus full training and - 1% on the full dataset. Bold text indicates matched or exceeded accuracy.

Compressed Faces

initial faces

learned synthetic faces

Figure 2 YaleFaces before and after compression

Parameter Sensitivity

initial subsampling

after convergence

Comparison

Compression

Results

Optimization Details
Objective
\[
L(Z, A) = - \sum_{z} \log(p(z)) \quad \text{where} \quad Z = [z_1 \ldots z_m]
\]