Stochastic Neighbor Compression

Matt J. Kusner, Stephen Tyree, Kilian Q. Weinberger, Kunal Agrawal

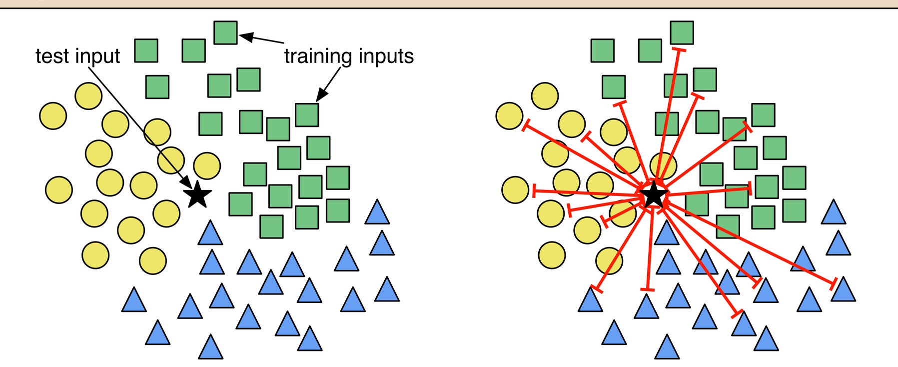
Department of Computer Science & Engineering, Washington University in St. Louis, USA

Machine Learning @Washington University in St. Louis

Stochastic Neighbor Compression

Motivation: Nearest Neighbor Rule

- \+ easy to implement
- \+ naturally multiclass
- \+ trivial to train
- **- expensive to test**
- **- expensive to store**



test complexity: $O(nd)$
n training inputs
d features

prior solutions

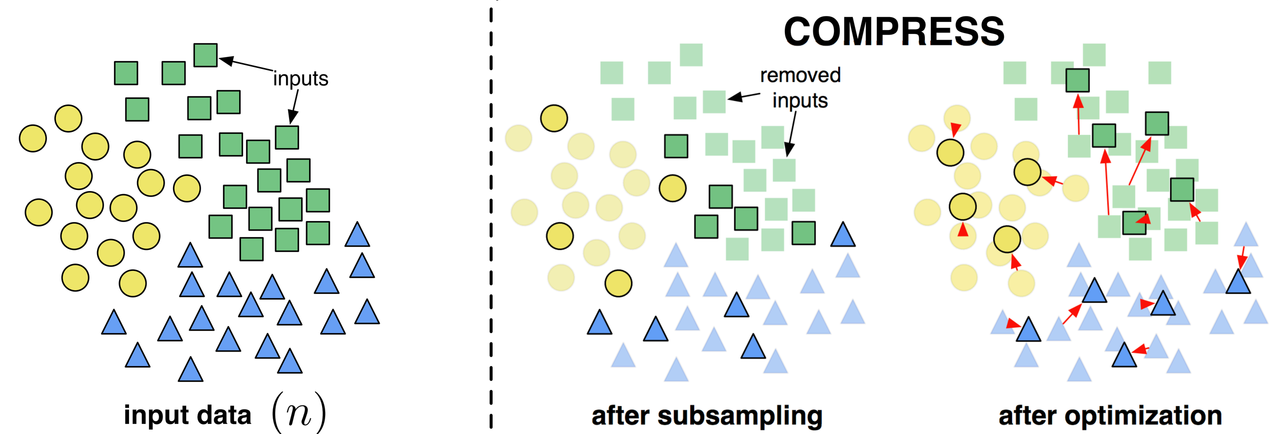
1. reduce dimensionality
- Tenenbaum et al., 2000
- Hinton & Roweis, 2002
- Weinberger et al, 2004
- van der Maaten & Hinton, 2008
- Weinberger & Saul, 2009

2. reduce distance computations
- Omohundro, 1989
- Beygelzimer et al., 2006
- Gionis et al., 1999
- Andoni & Indyk, 2006

3. reduce instances
- Hart, 1968
- Bermejo & Cabestany, 1999
- Toussaint, 2002
- Mollineda et al., 2002
- Anguilli, 2005

new idea: **dataset compression**

Main Idea



input data (n)
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n\}$
$\{y_1, y_2, \ldots, y_i, \ldots, y_{n-1}, y_n\}$

$\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ (m)
$\{\hat{y}_1, \ldots, \hat{y}_m\}$

test complexity: $O(md)$
m training inputs
d features

$m << n$

Step 1. Dimensionality Reduction (Optional)

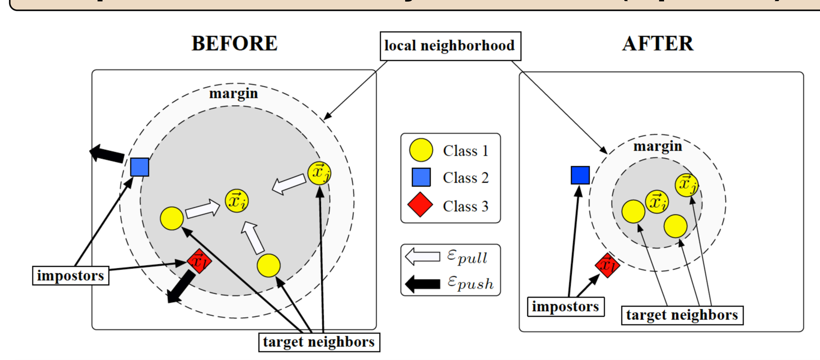
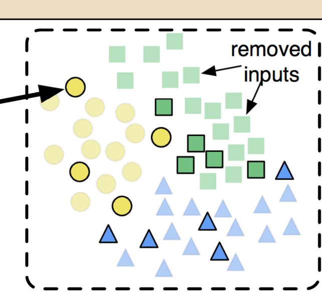


Figure 1 Dimensionality reduction before and after applying LMNN. A distance metric $\mathbf{A} \in \mathcal{R}^{r \times d}$ is optimized, where $r << d$ is the reduced dimensionality and $\|\mathbf{A}(\mathbf{x}_i - \mathbf{x}_k)\|_2^2$ is the new distance, so that neighbors of the same class lie closer than those of other classes.

Step 2. Subsampling

- Sample m compressed training inputs: $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$
- Select to preserve class balance
- For each compressed input $\mathbf{z}_j$, such that $\mathbf{z}_j = \mathbf{x}_j$, fix its label as such, $\hat{y}_j = y_j$
- Compression results are stable across different random samples



Step 3. Learning Compressed Inputs

Goal: Learn a compressed set $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ that predicts training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ correctly

normalized over all $\mathbf{z}_k$

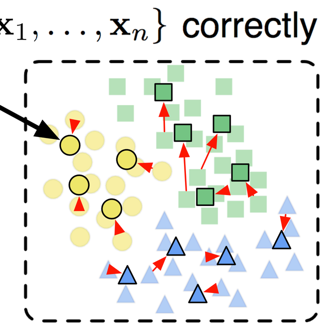
$$p_{ij} = \frac{\exp(-\|\mathbf{A}(\mathbf{x}_i - \mathbf{z}_j)\|_2^2)}{\sum_{k=1}^{m} \exp(-\|\mathbf{A}(\mathbf{x}_i - \mathbf{z}_k)\|_2^2)}$$

probability $\mathbf{z}_j$ is nearest neighbor of $\mathbf{x}_i$ [Hinton & Roweis, 2002]

$$p_i = \sum_{j:\hat{y}_j = y_i} p_{ij}$$

probability $\mathbf{x}_i$ is correctly classified [Goldberger et al., 2004]

sum over compressed inputs with same label as $\mathbf{x}_i$



- Ideally, $p_i = 1$ for all $i = 1, \ldots, n$
- KL-Divergence between p_i and 1: $-\log(p_i)$

Solution: Solve the optimization, $\min_{\mathbf{z}_1, \ldots, \mathbf{z}_m} -\sum_{i=1}^{n} \log(p_i)$

solve using conjugate gradient descent!

Optimization Details

Optimization

Objective

$$\mathcal{L}(\mathbf{Z}, \mathbf{A}) = -\sum_{i=1}^{n} \log(p_i) \quad \text{where} \quad \mathbf{Z} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{z}_1 & \mathbf{z}_2 & \ldots & \mathbf{z}_m \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Optimization Details

Define auxiliary matrices, $\delta_{y_i, \hat{y}_j} = \begin{cases} 1 & y_i = \hat{y}_j \\ 0 & \text{otherwise} \end{cases}$

$$\mathbf{Q} = \begin{bmatrix} \delta_{y_1,\hat{y}_1} - p_1 & \cdots & \delta_{y_1,\hat{y}_m} - p_1 \\ \vdots & \vdots & \vdots \\ \delta_{y_n,\hat{y}_1} - p_n & \cdots & \delta_{y_n,\hat{y}_m} - p_n \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_{11}/p_1 & \cdots & p_{1m}/p_1 \\ \vdots & \vdots & \vdots \\ p_{n1}/p_n & \cdots & p_{nm}/p_n \end{bmatrix}$$

Gradient with respect to $\mathbf{Z}$

place vector on diagonal of 0-matrix

size n vector of all 1s

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{A})}{\partial \mathbf{Z}} = 4\mathbf{A}^\top \mathbf{A}\left(\mathbf{X}(\mathbf{Q} \circ \mathbf{P}) - \mathbf{Z}\Delta\left((\mathbf{Q} \circ \mathbf{P})^\top \mathbf{1}_n\right)\right)$$

can relearn metric $\mathbf{A}$!

Gradient with respect to $\mathbf{A}$

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{A})}{\partial \mathbf{A}} = -2\mathbf{A} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{p_{ij}}{p_i} q_{ij} (\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^\top \quad \text{where} \quad q_{ij} = [\mathbf{Q}]_{ij}$$

Training Complexity

recall $\mathbf{A} \in \mathcal{R}^{r \times d}$

with respect to $\mathbf{Z}$

computation	complexity
$\mathbf{A}^\top \mathbf{A}$	$O(rd^2)$
$\mathbf{X}(\mathbf{Q} \circ \mathbf{P}) - \mathbf{Z}\Delta\left((\mathbf{Q} \circ \mathbf{P})^\top \mathbf{1}_n\right)$	$O(dmn)$
$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{A})}{\partial \mathbf{Z}}$	$O(d^2 m)$

total complexity: $O(rd^2 + dmn + d^2 m)$

with respect to $\mathbf{A}$

computation	complexity
$\sum_{i=1}^{n} \sum_{j=1}^{m} \frac{p_{ij}}{p_i} q_{ij} (\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^\top$	$O(d^2 nm)$
$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{A})}{\partial \mathbf{A}}$	$O(rd^2)$

total complexity: $O(rd^2 + d^2 nm)$

Results

Datasets and Training Time

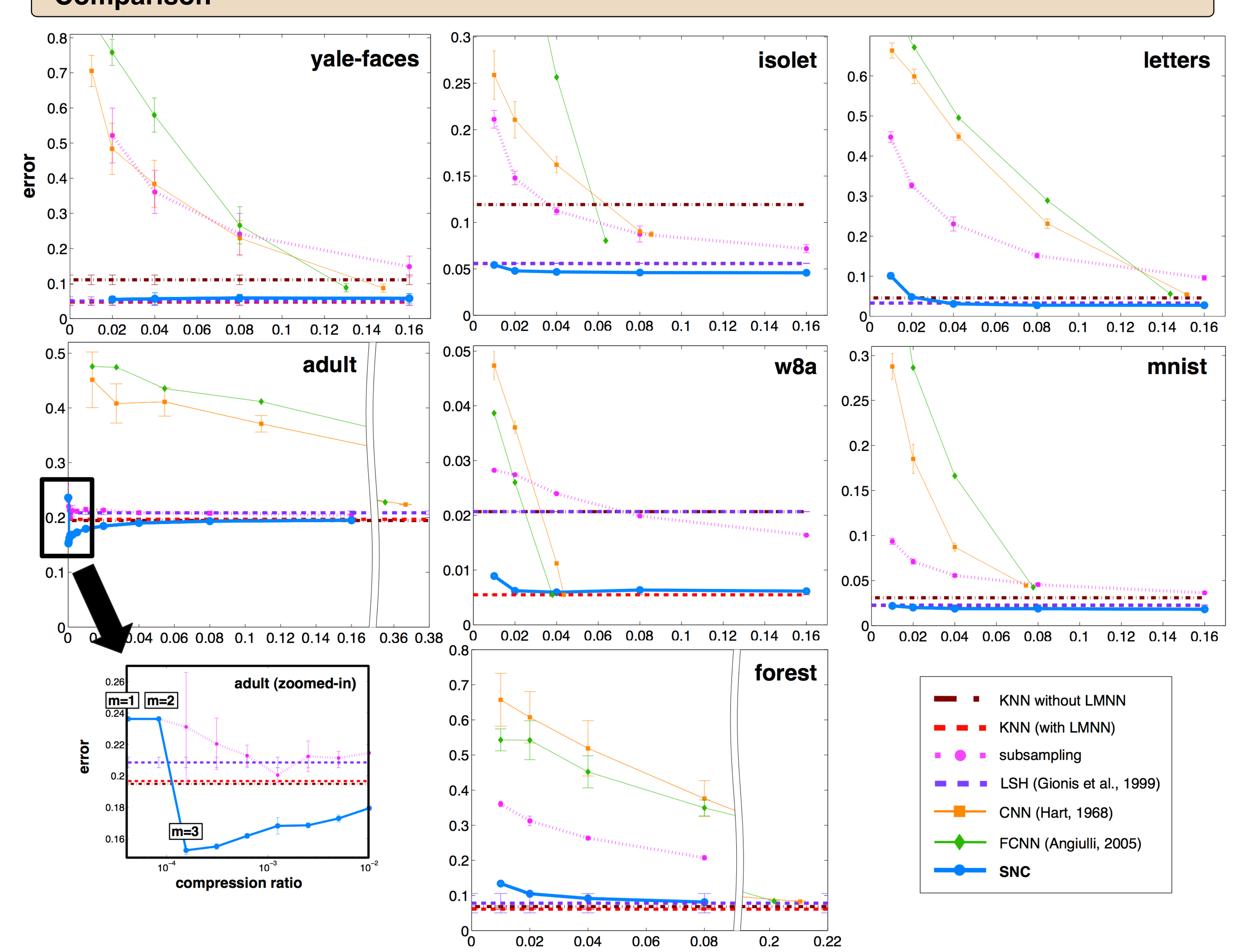
Dataset Statistics

Name	n	$\lvert\mathcal{Y}\rvert$	d (d_L)
YALE-FACES	1961	38	8064 (100)
ISOLET	3898	26	617 (172)
LETTERS	16000	26	16 (16)
ADULT	32562	2	123 (50)
W8A	49749	2	300 (100)
MNIST	60000	10	784 (164)
FOREST	100000	7	54 (54)

Training Times

Dataset	Compression Ratio 1%	2%	4%	8%	16%
YALE-FACES	–	4s	6s	9s	15s
ISOLET	11s	17s	28s	50s	1m 26s
LETTERS	41s	1m 18s	2m 44s	4m 34s	8m 13s
ADULT	2m 27s	4m 1s	7m 39s	12m 51s	23m 18s
W8A	6m 5s	10m 19s	19m 26s	39m 12s	1h 12m
MNIST	17m 18s	36m 43s	1h 13m	2h 17m	4h 57m
FOREST	17m 38s	33m	55m 44s	1h 45m	–

Comparison



Results

Test-time Speed-up

Dataset	Speed-up, Compression Ratio 1%			2%			4%			8%			16%			SNC 4% Comparison, Distance Comps. Ball-Trees	LSH
YALE-FACES	–	–	–	28	17	3.6	19	11	3.5	12	7.3	3.2	6.5	4.2	2.8	7.1	21
ISOLET	76	23	13	47	13	13	26	6.8	13	14	3.7	13	7.0	2.0	13	13	14
LETTERS	143	9.3	100	73	6.3	61	34	3.6	34	16	2.0	17	7.6	1.1	8.4	3.3	23
ADULT	156	56	3.5	75	28	3.4	36	15	3.3	17	7.3	3.1	7.8	3.8	3.0	17	0.7
W8A	146	68	39	71	36	35	33	19	26	15	10	18	7.3	5.5	11	13	2.1
MNIST	136	54	84	66	29	75	32	16	57	15	8.4	37	7.1	3.6	17	11	8.5
FOREST	146	3.1	12	70	1.6	11	32	0.90	10	15	1.1	7.0	–	–	–	0.15	0.35

Table 3 *Left:* Speed-up of kNN testing through SNC compression without a data structure (in black) *on top* of ball-trees (in teal) and LSH (in purple). 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 ball-trees and LSH on the full dataset. **Bold** text indicates matched or exceeded accuracy.

Compressed Faces

Figure 2 YaleFaces before and after compression

Parameter Sensitivity

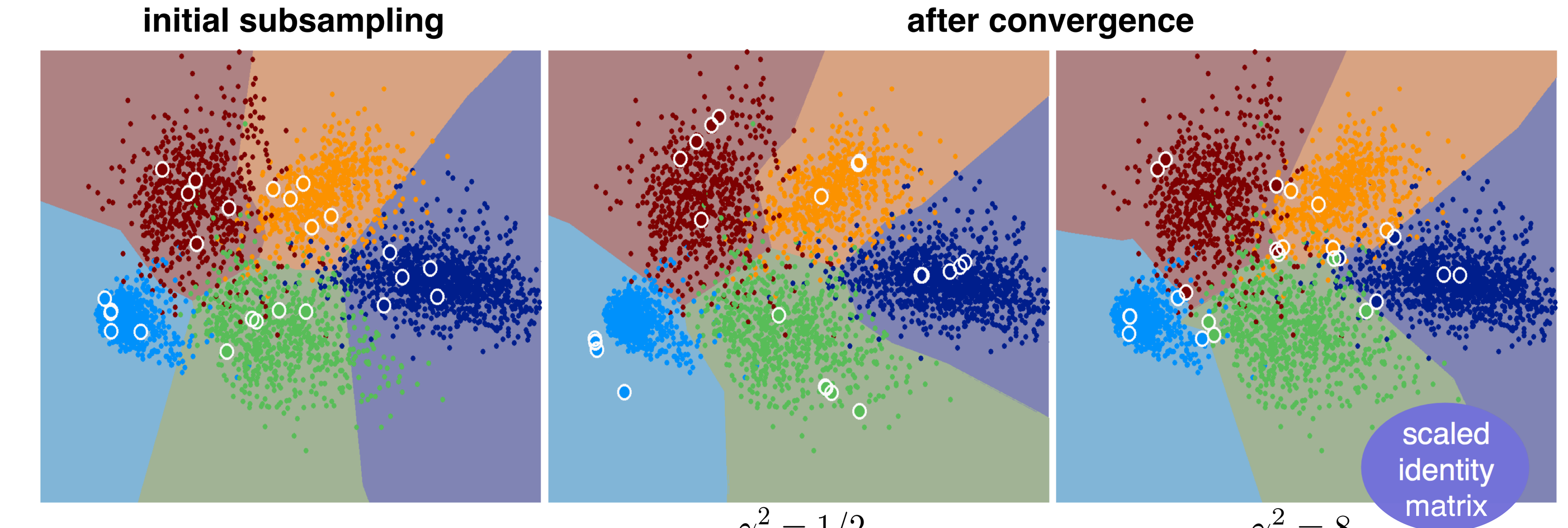
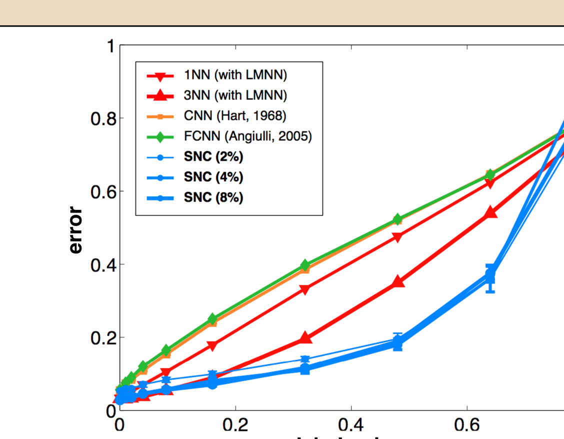


Figure 3 The decision rule and SNC data set (white circles) learned from 2d USPS digits under varying $\mathbf{A} = \gamma^2 \mathbf{I}$

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 mislabeled inputs and yields a significantly more robust kNN classifier. In contrast, CNN, 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

[1] Hinton, G.E., Roweis, S.T. Stochastic neighbor embedding. NIPS, 2002.
[2] Goldberger, J., Hinton, G.E., Roweis, S.T., Salakhutdinov, R. Neighbourhood components analysis. NIPS, 2004.
[3] Tenenbaum, J.B., de Silva, V., Langford, J.C. A global geometric framework for nonlinear dimensionality reduction, Science, 2000.
[4] Weinberger, K.Q., Sha, F., Saul, L.K. Learning a kernel matrix for nonlinear dimensionality reduction, ICML, 2004.
[5] Van der Maaten, L., Hinton, G. Visualizing data using t-sne. JMLR, 2008.
[6] Weinberger, K.Q., Saul, L.K. Distance metric learning for large margin nearest neighbor classification. JMLR, 2009.
[7] Omohundro, S.M. Five balltree construction algorithms. International Computer Science Institute, Berkeley, 1989.
[8] Beygelzimer, A., Kakade, S., Langford, J. Cover trees for nearest neighbor. ICML, 2006.
[9] Gionis, A., Indyk, P., Motwani, R., et al. Similarity search in high dimensions via hashing.VLDB, 1999.
[10] Andoni, A., Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. FOCS, 2006.
[11] Hart, P.E. The condensed nearest neighbor rule. IEEE Transactions on Information Theory, 1968.
[12] Bermejo, S., Cabestany, J. Adaptive soft k-nearest-neighbor classifiers. Pattern Recognition,1999.
[13] Toussaint, G.T. Proximity graphs for nearest neighbor decision rules: recent progress. Interface, 2002.
[14] Mollineda, et al. An efficient prototype merging strategy for the condensed 1-nn rule through class conditional hierarchical clustering. Pattern Recognition, 2002.
[15] Angiulli, F. Fast condensed nearest neighbor rule. ICML, 2005.

Acknowledgements

KQW, MK, ST are supported by NSF grants 1149882, 1137211. ST and KA are supported by NSF grants 1150036, 1218017. ST is supported by an NVIDIA Graduate Fellowship. The authors thank Laurens van der Maaten for helpful discussions. Computations were performed via the Washington University Center for High Performance Computing, partially provided through grant NCRR 1S10RR022984-01A1.