Project 2: KNN Classification with Different Distance Metrics

Group:
Yang Zhou
Yiming Liu
Yakai Wang
Hongkai Zheng

Abstract

In this project, we compared different distance measures including Chebyshev distance, Euclidean distance, Manhattan distance, and Cosine distance for k-NN classification. We then investigated four metric learning methods including Large Margin Nearest Neighbors(LMNN), Neighborhood Component Analysis(NCA), MMC, and Information-Theoretic Metric Learning(ITML), to learn a good metric. Experimental results shows that these algorithms do help improve the performance of k-NN classification over AwA2 dataset. Besides, we adopted 2-D visualization to visualize how metric learning algorithms help the k-NN classification. And we have further studied the LMNN with series of experiments.

1 Introduction to k-NN

Suppose we have pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ taking values in $\mathbb{R}^d \times \{1, 2, \ldots, m\}$ (totally m cluters), where Y is the class label of X. Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, let $\left(X_{(1)}, Y_{(1)}\right), \ldots, \left(X_{(n)}, Y_{(n)}\right)$ be a reordering of the training data such that $\left\|X_{(1)} - x\right\| \leq \cdots \leq \left\|X_{(n)} - x\right\|$. The class label of x is more likely to be decided by the nearest k neighbors.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. Another important parameter is the distance metric used (commonly Euclidean distance). In this experiment, we tried Manhattan, chebyshev, Mahalanobis, and Cosine distance metric to compare their performance. Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor (LMNN) or Neighbourhood components analysis (NCA).

1.1 Some distance metrics

The k-NN classifier fundamentally relies on a distance metric. The better that metric reflects label similarity, the better the classified will be. In this section, we will give some common distance metrics, which would be used in our experiment. Many of them are based on Minkowski distance:

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= \left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}} \\ &= \left(\sum_i |d_i|^p \right)^{\frac{1}{p}} \end{aligned}$$

This distance definition is pretty general and contains many well-known distances as special cases:

- When $p = 1$, it means Manhattan distance:

$$d(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i| \tag{2}$$

- When $p = 2$, it means Euclidean distance, which is most common in our daily life:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2} \tag{3}$$

- When $p = \infty$, it means Chebyshev distance:

$$d(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i| \tag{4}$$

Additionally, we used Cosine distance as another distance metric, it is defined as follows:

$$d(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}_i^T \mathbf{y}_i}{\|\mathbf{x}_i\| \cdot \|\mathbf{y}_i\|} \tag{5}$$

2 Metric Learning

In this section we give a brief introduction about the distance metric learning problem and metric learning algorithms we investigated in our experiments to improve K-Nearest Neighbors Classification. Table 1 shows the notions we will use in our report.

Table 1: Notations

Notation	Concept
$\mathcal{M}_d(\mathbb{R})$	Square matrices of orden d.
$S_d(\mathbb{R})_0^+$	Positive semidefinite matrices of order d
$j \rightsquigarrow i$	x_j is a target neighbor of x_i
$[\cdot]_+$	$[z]_+ = \max\{z, 0\}, \mathbb{R} \to \mathbb{R}_0^+$
$\mathrm{KL}(P\|Q)$	Kullback–Leibler divergence: $\mathrm{KL}(P\|Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$

2.1 Mahalanobis distance

Let $d \in \mathbb{N}$ and $M \in S_d(\mathbb{R})_0^+$. The Mahalanobis distance corresponding to the matrix M is the map $d_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by

$$d_M(x, y) = \sqrt{(x - y)^T M (x - y)}, \quad x, y \in \mathbb{R}^d \tag{6}$$

If $M \in S_d(\mathbb{R})_0^+$, then there exists a matrix $L \in \mathcal{M}_d(\mathbb{R})$ so that $M = L^T L$, and this matrix is unique except for an isometry. So then we get

$$d_M^2(x, y) = (x - y)^T M (x - y) = (x - y)^T L^T L (x - y) = (L(x - y))^T (L(x - y)) = \|L(x - y)\|_2^2 \tag{7}$$

2.2 LMNN

Large Margin Nearest Neighbors [4] is a distance metric learning algorithm aimed specially at improving the accuracy of the k-nearest neighbors classifier. It is based on the premise that classifier will label a sample more reliably if k neighbors share the same label. To do so it tries to learn a distance that maximizes the number of samples that share its label with as many neighbors as possible.

In this way, the LMNN algorithm tries to minimize an error function that penalizes, on the one hand, the large distance between each sample and those considered its ideal neighbors, and on the other hand, the small distances between examples of different classes.

Suppose we have a dataset $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ with corresponding labels $y_1, \ldots, y_N$. Given a sample $x_i \in \mathcal{X}$, the k nearest samples of the same class as x_i are called x_i's *target neighbors*. Once the target neighbors have been established, we define the object function, which will be composed of two terms. The first one will penalize distant target neighbors and the second one will penalize nearby impostors. The first term is defined as

$$\varepsilon_{\text{pull}}(L) = \sum_{i=1}^{N} \sum_{j \sim i} \|L(x_i - x_j)\|^2 \tag{8}$$

The minimization of this error causes a pulling force between the data samples. The second term is defined as

$$\varepsilon_{push}(L) = \sum_{i=1}^{N} \sum_{j \sim i} \sum_{l=1}^{N} (1 - y_{il}) \left[1 + \|L(x_i - x_j)\|^2 - \|L(x_i - x_l)\|^2\right]_+ \tag{9}$$

where y_{il} is a binary variable which takes the value 1 if $y_i = y_l$, and 0 if $y_i \neq y_l$, and the operator $[\cdot]_+ : \mathbb{R} \to \mathbb{R}_0^+$ is defined as $[z]_+ = \max\{z, 0\}$ Finally, the objective function results from combining these two terms. After fixing $\mu \in [0, 1]$, we define

$$\varepsilon(L) = (1 - \mu)\varepsilon_{pull}(L) + \mu\varepsilon_{push}(L) \tag{10}$$

2.3 NCA

Neighborhood Component Analysis [2] is another distance metric learning algorithm aimed specially at improving the accuracy of the nearest neighbors classifiers. Its aim is to learn a linear transformation with the goal of minimizing the leave-one-out error expected by the nearest neighbor classification.

We consider the training set $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$, labeled by $y_1, \ldots, y_N$. We want to learn a distance, determined by linear transformation $L \in \mathcal{M}_d(\mathbb{R})$, that optimizes the accuracy of the nearest neighbors classifier. Given two samples $x_i, x_j \in \mathcal{X}$, we define the probability that x_i has x_j as its nearest neighbor, for the distance determined by the mapping L, as follows:

$$p_{ij}^L = \frac{\exp\left(-\|Lx_i - Lx_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|Lx_i - Lx_k\|^2\right)} (j \neq i), \quad p_{ii}^L = 0 \tag{11}$$

Under this probability law, we can define the probability that the sample is correctly classified as the sum of the probabilities that x_i has as its nearest neighbor each sample of its same class, that is

$$p_i^L = \sum_{j \in C_i} p_{ij}^L, \text{ where } C_i = \{j \in \{1, \ldots, N\} : y_j = y_i\} \tag{12}$$

Finally, the expected number of correctly classified samples, and the function we try to maximize, is obtained as

$$f(L) = \sum_{i=1}^{N} p_i^L = \sum_{i=1}^{N} \sum_{j \in C_i} p_{ij}^L = \sum_{i=1}^{N} \sum_{\substack{j \in C_i \\ j \neq i}} \frac{\exp\left(-\|Lx_i - Lx_j\|^2\right)}{\sum_{j \neq i} \exp\left(-\|Lx_i - Lx_k\|^2\right)} \tag{13}$$

2.4 MMC

MMC [5] adapted a simple way of defining a criterion for the desired metric. It tries to minimize the squared distance between points with the same label. But this is trivially solved with $M = 0$, which is not useful, and we add a constraint to ensure that M does not collapse the dataset into a single point.

$$\min_{M} \sum_{y_i = y_j} d_M^2(x_i, x_j) \tag{14}$$

$$\text{s.t.} \sum_{y_i \neq y_j} d_M^2(x_i, x_j) \geq 1 \tag{15}$$

$$M \in S_d(\mathbb{R})_0^+ \tag{16}$$

2.5 ITML

Information-Theoretic Metric Learning [1] considers the settings that prior information about the Mahalanobis distance is known. It regularizes the M to be as close as possible to a given Mahalanobis distance function, parameterized by M_0. The measure of "closeness" between M and M_0 is quantified via a natural information-theoretic approach. There exists a simple bijection between the set of Mahalanobis distances and the set of equalmean multivariate Gaussian distributions (without loss of generality, we can assume the Gaussians have mean $\boldsymbol{\mu}$). Given a Mahalanobis distance parameterized by M, we express its corresponding multivariate Gaussian as $p(\boldsymbol{x}; M) = \frac{1}{Z} \exp(-\frac{1}{2} d_M(\boldsymbol{x}, \boldsymbol{\mu}))$, where Z is a normalizing constant and A^{-1} is the covariance of the distribution. Using this bijection, ITML gets the following metric learning problem:

$$\min_{M} \quad \mathrm{KL}\left(p\left(\boldsymbol{x}; M_0\right) \| p(\boldsymbol{x}; A)\right) \tag{17}$$

$$\text{subject to } d_M\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \leq u \quad y_i = y_j \tag{18}$$

$$d_M\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \geq \ell \quad y_i \neq y_j \tag{19}$$

3 Experiments and Results

In this section, we first evaluated different distance measures for KNN image classification, including Chebyshev distance, Euclidean distance, Manhattan distance, and Cosine distance. Then we investigated three metric learning methods including LMNN, NCA, MMC, and ITML, which help improve the performance of KNN classifier.

3.1 Dataset and Implementation

In our experiments, we use Animals with Attributes dataset(AwA2) [3], which contains 37322 images of 50 animal classes with pre-extracted deep learning features of 2048 dimensions for each image. All of the experimental results are based on randomly 60/40 splits of data, where 60% for training 40% for testing.

3.2 Explore Different Distance Measures

We compared different distance measures for k-NN classification via 5-fold cross-validation. Figure 1 reports the results of 5-fold cross-validation, where scores are averaged over 5 loops. We then evaluated them over the test set, setting k by cross-validation mentioned above.

Distance Metric	K neighbors	Test Score
Manhattan	5	88.4
Euclidean	5	89.5
Chebyshev	7	79.0
Cosine	7	**90.7**

Table 2: K set by cross-validation and test scores for different distance metrics

It is evident that Cosine distance got the first place in all the validations over training set, showing its ability fitting to the data, but it has also obtained clear victories over test set, thus also demonstrating a great capacity for generalization.

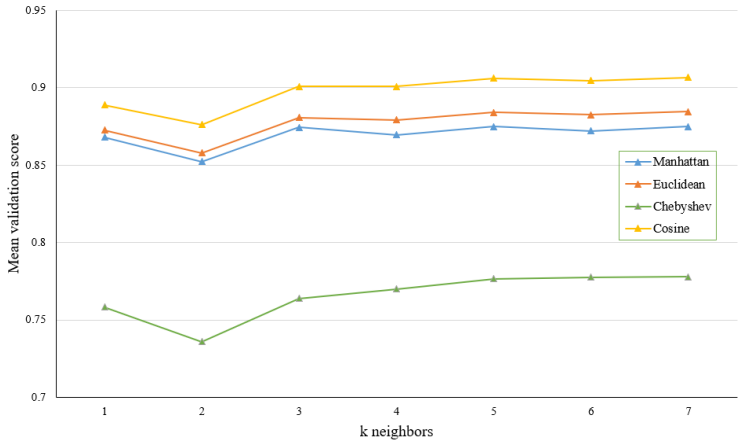


Figure 1: Average cross-validation scores of different distance measures with varying numbers of neighbors(Ks)

We can also see that Euclidean and Manhattan distance stand out, although at a considerable distance from Cosine. Results shows that data can be well separated by both angle and magnitude. But it is more helpful for knn classifier to look at the angle between data points than take magnitude into consideration.

3.3 Metric Learning

We evaluated the metric learning algorithms including LMNN, NCA, MMC, and ITML, applied to the k-nn classification, for different values of k. Figure 2 depicts the knn classification test score of different metrics learned by algorithms. As for MMC and ITML, we used Principal components analysis (PCA) to reduce the dimensionality, to both speed up training and avoid overfitting. In LMNN, we set its number of *target neighbors*(explained in section2.2) to the same as the number of nearest neighbors(K).

By comparing learned metrics versus the baseline Euclidean distance, we can clearly see that NCA is the one that obtains the best results. This is partly due to the fact that the algorithms have been evaluated with kNN classifier, and NCA is specially designed to improve this classifier. NCA got highest performance on test set, demonstrating a great capacity of generalization.

LMNN and ITML also help improved kNN classification a lot. ITML has better performance than LMNN in the case of small number of neighbors. But they both performed as good as NCA when K reached 7.

As for MMC, we found that in case of learning full matrix M, we got very low test score. But if we confined M to be diagonal, it can achieve the same performance as Euclidean with only 64 features, which is $1/32$ of original 2048 features. To find out the difference between diagonal M and full M, we adopted 2-D visualization by t-SNE to visualize their transformed data distributions. Figure 3 shows the results of plotting transformed data $A^{\frac{1}{2}}X$. As we see, diagonal M is much more successful than full M in bringing together the similar points while keep dissimilar ones apart.

To have a intuitive view into how metric learning algorithms works, we also visualize the original data and the data that was transformed by learned metrics of LMNN and NCA, which is shown in figure 4. Figure 5 demonstrated the 128-dimension data that generated by PCA and ITML transformation on that. As shown in table 3, we choose the best experimental parameters for each algorithms to generate transformation matrices.

As is vividly demonstrated in figure 5 3 4, metric learning algorithms are able to learn a transformation adapt to our data that bring the samples with the same labels closer while keep the different ones apart.

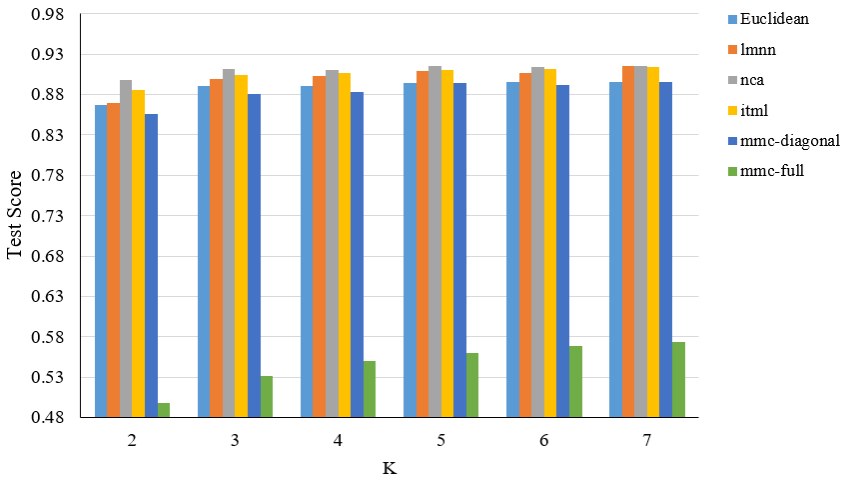
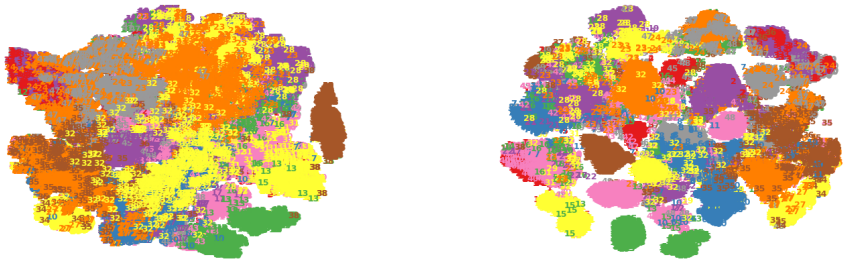
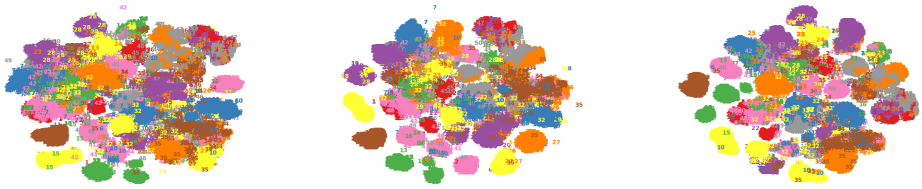


Figure 2: Classification test score for knn via different learned metrics



(a) full M (b) diagonal M

Figure 3: 2-D visualization of MMC transformed data $A^{\frac{1}{2}}X$

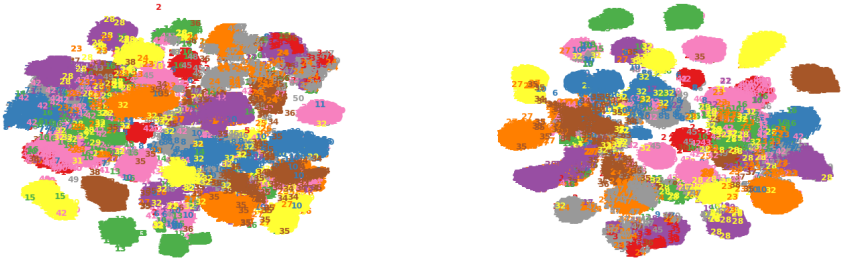


(a) Original data (b) LMNN transformation (c) NCA transformation

Figure 4: 2-D visualization of original data and LMNN&NCA transformed data($A^{\frac{1}{2}}X$)

Algorithms	LMNN	NCA	MMC-diagonal	MMC-full	ITML
K neighbors	7	7	7	7	7
features	2048	2048	64	64	128

Table 3: Experimental parameters of each algorithm for 2-D visualization



(a) 128-dimension data (b) ITML transformation

Figure 5: 2-D visualization of ITML transformed data $A^{\frac{1}{2}}X$

μ	2-nn	3-nn	4-nn	5-nn	6-nn	7-nn
0.1	0.8505	0.8699	0.8776	0.8806	0.8863	0.8851
0.2	0.8415	0.8681	0.8802	0.8792	0.8881	0.8901
0.3	0.8343	0.8734	0.8794	0.8839	0.8816	0.8919
0.4	0.8429	0.8765	0.8705	0.8869	0.8783	0.8877
0.5	0.8489	0.8751	0.8739	0.8875	0.8826	0.8769
0.6	0.8482	0.8685	0.8782	0.8769	0.8869	0.8832
0.7	0.8388	0.8718	0.8731	0.8803	0.8742	0.8905
0.8	0.8436	0.8538	0.8729	0.8658	0.8788	0.8749
0.9	0.8313	0.8657	0.8424	0.7883	0.8664	0.8523

Table 4: Results of series of experiments on μ

3.4 Further Study into LMNN

The object function of LMNN consists of two terms 2.2. The first term is to pull the samples with the same labels together and the second one is to force the different-class samples to stay apart. Regularization hyperparameter μ in $\varepsilon(L)10$ creates a trade-off between these two forces. We conducted series of experiments of LMNN with μ varying from $[0, 1]$ to investigate the impact of μ. To speed up the training, PCA was used to reduce dimensionality to 32.

The experiments were conducted on 2-nn, 3-nn, 5-nn, 6-nn, 7-nn classifiers. Figure 6 and table 4 reports the results. Except for a extreme point of 5-nn, there is no obvious relation between μ and the performance of k-nn classifiers. The choice of μ dose not cause great differences in results, so we can make it 0.5 or set it by cross-validation.

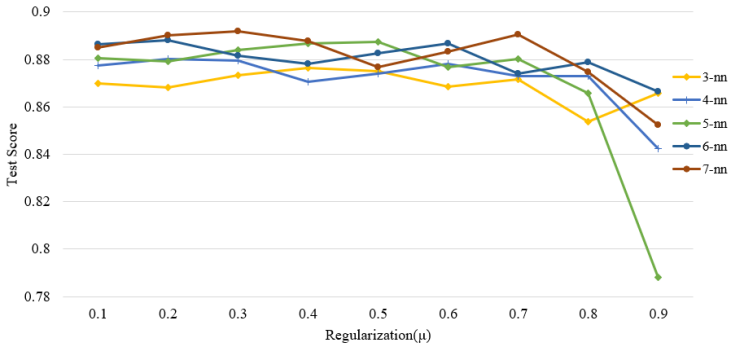


Figure 6: Results of k-nn LMNN with μ varying from $[0, 1]$

3.5 Summary of Metric learning

Table 5 summarizes the optimal experimental parameters and properties of each metric learning algorithms and their best performance on test set. It is evident that NCA and LMNN is the most effective and efficient metric learning algorithms for k-NN classification. ITML can achieve the top performance with only $1/16$ of original features. It is effective but time consuming. MMC is fast but it cannot handle high dimension data and got weaker capacity of generalization.

Except for MMC-full, other algorithms including LMNN, NCA, MMC-diagonal, and ITML, shows a strong ability to bring together the similar samples while keep dissimilar ones apart and successfully improved k-NN classification. Further study about LMNN shows that the choice of μ does not cause great differences in results, which implies that empirically μ can be set by cross-validation.

Algorithms	LMNN	NCA	MMC-diagonal	MMC-full	ITML
optimal K neighbors	7	7	7	7	7
features	2048	2048	64	64	128
CPU time(s)	3853	9107	46.4	37.4	15078
Test Score	91.49	**91.52**	89.51	57.32	91.43

Table 5: Experimental parameters and properties of different algorithms and their performance

4 Conclusion

In conclusion, we compared different distance measures including Chebyshev distance, Euclidean distance, Manhattan distance, and Cosine distance for k-NN classification over AwA2 dataset. Cosine is empirically the best measure that achieves the optimal performance.

We then investigated four metric learning methods including LMNN,NCA, MMC, and ITML to learn a good metric. Experimental results shows that NCA and LMNN is the most effective and efficient algorithms for k-NN classification. MMC is fastest but hard to deal with high dimension data. ITML is most effective but most time consuming. Besides, we adopted 2-D visualization to visualize how good metric help the k-NN classification. And we have further studied the impact of hyperparameter μ in LMNN with series of experiments.

References

[1] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 209–216, 2007.

[2] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 513–520, 2004.

[3] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):453–465, 2014.

[4] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 1473–1480, 2005.

[5] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 505–512, 2002.