From Instance-level Constraints to Space-level Constraints: Making the Most of Prior Knowledge in Data Clustering

Dan Klein KLEIN@CS.STANFORD.EDU
Sepandar D. Kamvar KAMVAR@SCCM.STANFORD.EDU
Christopher D. Manning MANNING@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA 94305-9040 USA

Abstract

We present an improved method for clustering in the presence of very limited supervisory information, given as pairwise instance constraints. By allowing instance-level constraints to have space-level inductive implications, we are able to successfully incorporate constraints for a wide range of data set types. Our method greatly improves on the previously studied constrained k-means algorithm, generally requiring less than half as many constraints to achieve a given accuracy on a range of real-world data, while also being more robust when over-constrained. We additionally discuss an active learning algorithm which increases the value of constraints even further.

1. Introduction

For many of the large datasets now available online, extensive hand-labeling would be costly and time-consuming enough to make standard supervised learning algorithms infeasible. Beyond that, part of the goal might be pattern discovery: a good labeling of the instances may not be known. In many such cases, gathering a large amount of unlabeled data is cheap and easy, and we may well be able to get a small amount of prior knowledge, such as some instance-level constraints indicating whether particular items are similar or dissimilar.

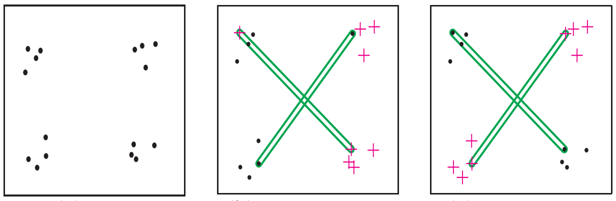
Here, we consider the two types of constraints introduced by Wagstaff and Cardie (2000): either two instances are known to be in the same class (in which case we say that they are *must-linked*), or they are known to be in different classes (in which case we say that they are *cannot-linked*). These types of constraints are intuitively appealing for the task of data clustering, where the goal is to group similar instances. They are a natural way to encode background knowledge even when class labels are not known a priori. For instance, for the task of protein function prediction, genome sequence data can be augmented by knowledge about functional links between proteins (Eisenberg et al., 2000). Here, functional links can be found by experimental means, such as the phylogenetic profile method or the gene neighbor method, and complement similarity information that can be automatically computed from sequence data. In collaborative filtering, the user may wish to modify his recommendations if he knows a priori that two books are alike (or not alike). Depending on the nature of the problem and source of the background knowledge, either or both types of constraints may be present.

The task of constrained clustering is closely related to the problem of semi-supervised learning, where the goal is to induce class labels for data given a very small training set. However, it is important to note that the information given by the pairwise constraints we explore is weaker than the information given by labeled data. While class labels can be used to generate pairwise constraints, pairwise constraints only give information about pairs of instances, and cannot be used to partially label the data sets.

The idea of using background knowledge to constrain clustering has been widely explored (Gordon, 1996; Wagstaff et al., 2001). However, the present work is novel in both the consideration of a spatial inductive interpretation of the constraints and in the presentation of an active constraint selection strategy.

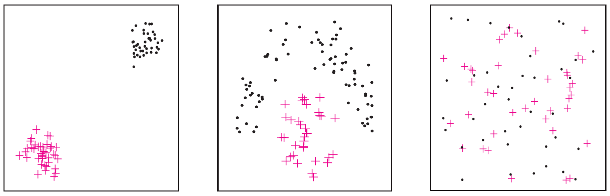
2. Instance vs. Space Level Constraints

While it is important for a clustering algorithm to satisfy known constraints, it is equally important for the algorithm to satisfy the *implications* of those constraints. For example, in figure 1, both sets of clusters (1b and 1c) satisfy the diagonal must-link constraints, but figure 1c is clearly a more intuitive partitioning. This is because the constraints suggest space-level generalizations beyond their explicit instance-level assertions; not only should points that are must-linked be in the same cluster, but the points that are near these points should probably also be in the same cluster. Cannot-links have similar spatial generalizations.



(a) Data (b) Instance Bias (c) Spatial Bias

Figure 1. The effects of adding two diagonal must-link constraints to the data in (a): an instance-level inductive bias results in single outliers (b) while a stronger space-level bias results in qualitative changes to the clusters (c).



(a) (b) (c)

Figure 2. The scenarios for clustering. (a) Distant, tight clusters are easily detected without constraints. (b) Odd-shaped or non-contiguous clusters can be detected more easily with constraints. (c) Clusters which are not at all separated in the feature space will not likely be detected, even with constraints.

Previous algorithms (COP-COBWEB (Wagstaff & Cardie, 2000) and COP-k-means (Wagstaff et al., 2001)) designed for the task of clustering with constraints have failed to show marked improvement over their unsupervised counterparts with the addition of very few constraints. COP-COBWEB is a constrained variant of COBWEB (Fisher, 1987), an incremental partitioning algorithm; and COP-k-means (CKM) is a constrained incremental-assignment variant of standard k-means (KM) clustering (McQueen, 1967). In these algorithms, a check is made at each assignment to see if the instance being assigned is must-linked or cannot-linked to a previously assigned instance, and the assignment is made accordingly. A major flaw with these algorithms is that they fail to utilize the space-level implications suggested by the constraints; in other words, they have no mechanism for propagating the constraints. Therefore, they will often exhibit the "outlier" behavior seen in figure 1b. While the clusters they produce may be consistent with the constraints themselves, they are often not consistent with the natural implications of those constraints.

It is important to stress that constrained clustering is a problem of induction, and therefore subject to differing induction principles. The principle we propose is that elements involved in pairwise constraints are generally representative of their local neighborhoods. However, if one were supplying constraints with the express purpose of reclassifying known outliers, then our induction principle would not apply, and in these cases, it might well be better to use an algorithm like COP-k-means which exhibits this behavior. But this latter bias is unnatural as a starting point for clustering; it would more naturally apply to fixing up mistakes in an existing clustering.

We propose here an algorithm that addresses this problem by distorting pairwise proximities between instances to reflect the spatial information given by the constraints. Proximity-based clustering is then done with the distorted proximities. This entire algorithm, called Constrained Complete-Link (CCL), performs substantially better than previously proposed algorithms in empirical studies.

3. Constraint Applicability

Before presenting the details of the algorithm, it is helpful to identify the cases in which adding a few constraints will be useful in pattern discovery or classification.

If the data naturally form tight clusters that are well-separated (as in figure 2a), there is no need for background knowledge at all – any reasonable clustering algorithm will detect the desired clusters. Likewise, if no distinction can be made between classes in feature space (as in figure 2c), then little useful information can be found in the data itself, and constraints will again be of little use.

Background knowledge will therefore be most useful when patterns are at least partially present in the data, but a clustering algorithm will not detect them correctly without assistance. This situation can arise in many ways. We focus on the case where the fine proximity structure in feature space is strongly correlated with the underlying similarity, but the coarse proximity structure may be misleading. Figure 2b and figure 7 show extreme examples of such cases. While these examples may seem contrived, real data often has such characteristics to a lesser degree, and the method we present works well for real data (figure 10) as well as for the examples in figure 7.[1]

Our goal is to take feature-space proximities, along with a sparse collection of pairwise constraints which indicate ways in which the feature space is unlike the target similarity space, and to cluster in a space which is generally like the feature space, but altered to accommodate the constraints. This alteration may involve a radical change in the topology of the original space which allows entirely new clusters to be detected, or it might involve only small deformations which improve the boundaries of mostly correct clusters.

[1] In the case where a good similarity space can be constructed from the feature space by a global linear transformation, feature weighting may be a more appropriate way to improve performance. Zhu et al. (2001) describe a heuristic method for selecting feature weights from pairwise constraints.

4. Imposing and Propagating Constraints

The outline of our general algorithm is as follows. We are given a proximity matrix for the instances in our data set, as well as a set of constraints given as pairwise cluster decision assertions. We create a new proximity matrix on the basis of the constraints and their implications. We then supply this new matrix to a proximity-based clustering algorithm.

Our distortion algorithm has two goals. First, in our distorted proximities, we would like specific items known to be in the same class to be very close together, while two items in different classes should be very far apart. Adjusting the feature space in this manner, for example increasing the distance between two cannot-linked points, is called *imposing constraints*, and, by itself, corresponds to an instance-level view of the constraints.

Secondly, we would like to further distort other entries in the proximity matrix to reflect the following two intuitions.

- If points $\mathbf{x}_i$ and $\mathbf{x}_j$ are very close together, then points that are very close to $\mathbf{x}_i$ are close to $\mathbf{x}_j$.
- If points $\mathbf{x}_i$ and $\mathbf{x}_j$ are very far apart, then points that are very close to $\mathbf{x}_i$ are far from $\mathbf{x}_j$.

Further adjusting the proximities to satisfy these intuitions corresponds to a space-level view of the constraints. This adjustment is called *propagating constraints*, and is illustrated in figure 3.

Notice that the two intuitions above are trivially satisfied by the triangle inequality if we have a metric input proximity matrix and no constraints. However, in imposing constraints, we may lose metricity. One way of propagating constraints is therefore to find a way of restoring the metricity of proximities while maintaining the constraints that were imposed.

One could imagine various methods for proximity distortion and various clustering algorithms; we give a concrete method for each in the following sections (and pseudocode in figures 5 and 6).

4.1. Must-links

In the case where the only constraints are must-link pairs, imposing the constraints will only involve shortening certain entries in the proximity matrix. Concretely, we interpret the proximity matrix as weights for a complete graph over the data points, and we impose must-link constraints by lowering the distance between the must-linked pair to zero. If the original proximities are metric, then the arc directly connecting two points is a shortest path between those points. By imposing the constraints, we will likely have violated the triangle inequality and therefore this shortest path property, but only in a very specific way: points which were previously some distance d apart may now be closer along some path which skips through the constrained pairs. We can therefore find a new metric which respects these new constrained entries by running an all-pairs-shortest-paths algorithm on the imposed matrix. The resulting path-length matrix will be metric, and will still be faithful to the original distances in some sense.

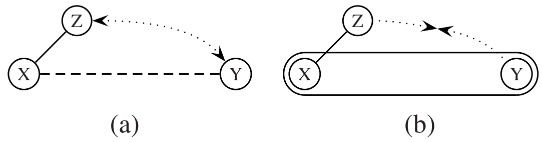


Figure 3. Constrained pairs have implications for nearby points. If X and Z are very close, then (a) constraining X away from Y should push Z from Y and (b) constraining X towards Y should pull Z towards Y.

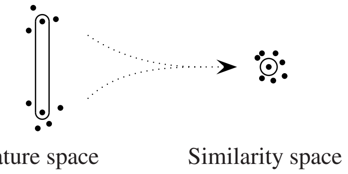


Figure 4. Clusters which are distant in feature space can be brought together in similarity space with a propagated must-link constraint.

Our method of imposing constraints allows us to speed up the computation of the all-pairs-shortest-path lengths. In particular, for every source s and goal g there is a shortest path p where all points along p are either s, g, or some point involved in a must-link constraint. A trivial modification to the Floyd-Warshall algorithm (Cormen et al., 1990) allows us to do the all-pairs-shortest-paths computation in time $O(N^2C)$ where C is the number of points involved in some must-link constraint, rather than $O(N^3)$. If we assume that $C \ll N$, then this phase is no more expensive than proximity-based clustering algorithms alone.[2]

4.2. Cannot-links

The addition of cannot-links complicates matters substantially. First of all, while we can find *some* satisfying clustering for pure must-links in only slightly superlinear time, it is NP-complete to even determine whether a satisfying assignment exists when cannot-links are present. Even in practice, it is much harder to devise a satisfactory procedure when cannot-link constraints are included.

An example of a well-founded but ineffective procedure would be to take the input proximity matrix D, constrain all must-linked entries to zero, constrain all cannot-linked entries to some large number (perhaps $\max_{i,j} D_{ij} + 1$), and allow all other entries to vary. Then we could search

[2]Complete-link clustering standardly runs in $O(N^2 \log N)$ time with a priority queue implementation.

for a matrix D' such that D' defines a metric space and $D' = \arg\min_{D' \text{ metric}} |D - D'|$ for some norm. This constrained optimization problem is too large to solve for more than a dozen points with general-purpose solvers, since each permutation of three data points (a, b, c) corresponds to some triangle inequality $D_{ac} + D_{cb} \geq D_{ab}$.

What we use for mixed constraints is less satisfying conceptually, though it works well in practice. We first add the must-link constraints using the all-pairs-shortest-paths algorithm from section 4.1. This gives us a metric matrix. Then, we only impose instance-level cannot-links, setting those entries to $\max_{i,j} D_{ij} + 1$. Then, rather than explicitly restore the metricity, we choose a proximity-based clustering algorithm that will indirectly propagate the cannot-link constraints, implicitly restoring some metricity each time it performs a merge. We will discuss this further in section 4.3, but we mention here that the clustering phase is effective at propagating cannot-links. One way in which this division between propagation methods is appropriate in our context is that, as mentioned before, satisfying cannot-links is NP-complete, as is the clustering problem, while must-links can be satisfied very efficiently. Therefore, since one hard problem is being approximated with heuristic clustering, it is convenient to approximate both at once.

4.3. Clustering

For the present work, we use complete-link hierarchical agglomerative clustering (see Jain & Dubes, 1988) as our clustering algorithm. We assume basic familiarity with complete-link (CL) clustering. CL merges clusters in order of proximity; the closest clusters will be merged first, and the furthest clusters will be merged last. By setting the must-link entries in the proximity matrix to 0, and the cannot-link entries to $\max_{i,j} D_{ij} + 1$, we can achieve a direct operational (instance-level) interpretation of the constraints without any modification to the clustering algorithm. The propagation of the cannot-link constraints occurs through the merges. At each merge, CL creates a *reduced proximity matrix*, with one less row and column. Because CL defines the distance between clusters as the maximum distance between points in each cluster, if A is cannot-linked to B, merging A and C will cause C to also be cannot-linked to B. In this way, CL achieves implicit propagation of cannot-link constraints.

constrainProximities(Matrix D, Constraints C)
  imposeMustLinks(D,C)
  propagateMustLinks(D,C)
  imposeCannotLinks(D,C)
  propagateCannotLinks(D,C)

imposeMustLinks(Matrix D, Constraints C)
  for $(i, j) \in C_{\text{must}}$
    $D_{ij}, D_{ji} = 0$

imposeCannotLinks(Matrix D, Constraints C)
  for $(i, j) \in C_{\text{cannot}}$ and $(j, k) \in C_{\text{must}}$
    $D_{ik}, D_{jk} = 0$

propagateMustLinks(Matrix D, Constraints C)
  D = fastAllPairShortestPaths(D,C)
  for (i, j) s.t. $D_{ij} = 0$
    $C_{\text{must}} = C_{\text{must}} \cup \{(i, j)\}$

propagateCannotLinks(Matrix D, Constraints C)
  (done implicitly by completeLink)

fastAllPairsShortestPaths(Matrix D, Constraints C)
  % find valid intermediates
  $I = i : \exists j \neq i, (i, j) \in C_{\text{must}}$
  % modified Floyd-Warshall
  for $k \in I$, for $i \in \{1 : n\}$, for $j \in \{1 : n\}$
    $D_{ij} = \min\{D_{ij}, D_{ik} + D_{kj}\}$

Figure 5. Pseudocode for constraining an input proximity matrix.

constrainedCL(Matrix D, Constraints C)
  constrainProximities(D, C)
  completeLink(D)

completeLink(Matrix D)
  $Clusters = \{c_i$ for each point $i\}$
  $Linkage$ starts empty
  distances $\delta(c_i, c_j) = D_{ij}$
  while $|Clusters| > 1$
    choose closest $(c_1, c_2) = \arg\min_{c_1, c_2 \in Clusters} \delta(c_1, c_2)$
    add (c_1, c_2) to $Linkage$
    merge c_1 and c_2 into c_{new} in $Clusters$
    for $c_i \in Clusters$
      $\delta(c_i, c_{new}) = \max\{\delta(c_i, c_1), \delta(c_i, c_2)\}$

Figure 6. Pseudocode for constrained complete-link clustering.

5. Results and Discussion

5.1. Evaluation Criteria

Several methods exist for cluster evaluation (Siegel & Castellan, 1988). When a target classification is known, a commonly used index is the Rand Index (Rand, 1971). The Rand index views a clustering of the data as a linkage decision for each pair of data points. A pair is considered correct if the proposed clustering agrees with the target clustering. The Rand index is then:

$$RI = \# \textit{ correct decisions} / \# \textit{ total decisions}$$

Its value lies between 0 and 1, 1 being perfect agreement.

Following Wagstaff and Cardie (2000), we use a modification of the Rand index, suitable for constrained clustering. Adding constraints ensures the correctness of pairs fixed by the constraints or their closure. Therefore, we confine our evaluation to decisions which are underdetermined by the constraints. We have:

$$CRI = \# \textit{ correct free decisions} / \# \textit{ total free decisions}$$

We use this not only because it is a natural evaluation criteria for clustering with pairwise constraints, but also to

Figure 7. Synthetic data sets: target clusterings. (a) CIRCLES, (b) TWOCIRCLES, (c) XOR, and (d) STORMCLOUDS

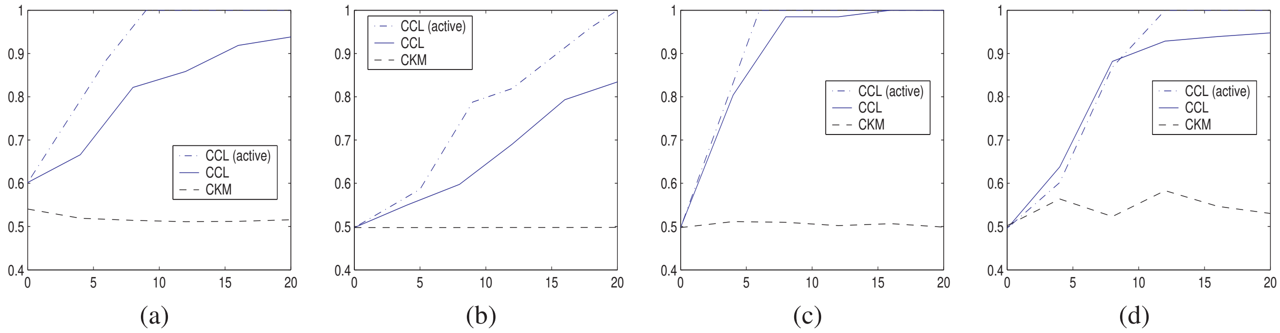


Figure 8. Synthetic data sets: number of constraints vs. accuracy. CCL is constrained CL with random constraint selection, CCL (active) is constrained CL with active selection, and CKM is COP k-means. (a) CIRCLES, (b) TWOCIRCLES, (c) XOR, and (d) STORMCLOUDS

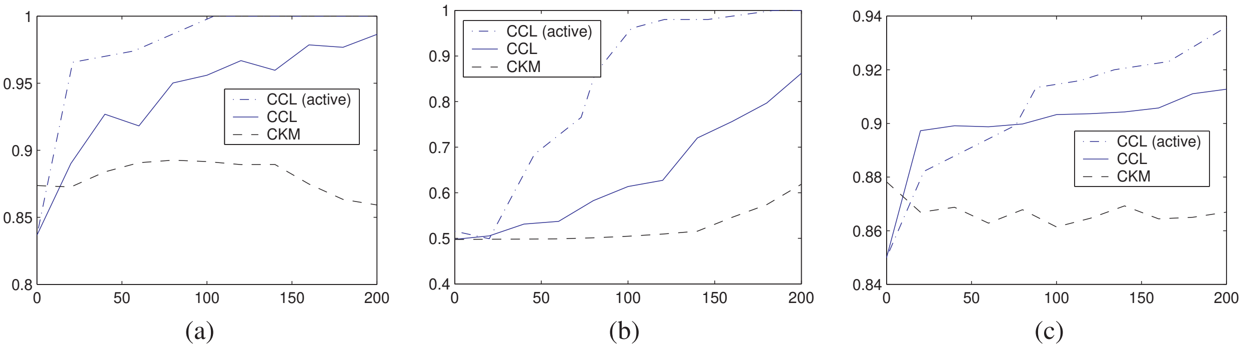


Figure 9. Real data sets: number of constraints vs. accuracy. (a) IRIS, (b) CRABS, (c) SOYBEAN.

facilitate comparison with previous work in this area. The term "accuracy" will be used to refer to CRI values.

In what follows, CCL is the constrained complete-link algorithm presented above and CKM is our re-implementation of COP-k-means (Wagstaff et al., 2001).

5.2. Synthetic data

We evaluate our system using both synthetic and real-world data. The synthetic data is designed to highlight problems which can be solved effectively with CCL but not with either unconstrained CL or other constrained algorithms. Figure 7 shows the target clusterings of the synthetic sets:

(a) CIRCLES is a difficult case for spherical clustering methods (like CL and KM).

(b) TWOCIRCLES is difficult for any common clustering algorithm because the centers' equality is not proximal in the feature space.

(c) XOR is difficult because the solution is not linearly separable (and so not solvable by two-class KM) and prior knowledge is required to distinguish the target labeling from alternate ones.

(d) STORMCLOUDS is a difficult case for spherical clustering methods because of the high ellipticality.

Figure 8 shows that CCL does very well on all these sets. Constraints were added by randomly choosing data pairs and constraining that pair to be whatever it actually is in the target clustering (we also examine active selection in section 5.5). In every case, there is sizable improvement over the unconstrained accuracy, even with very few constraints. Moreover, CCL's spatial propagation allows it to substantially outperform CKM. To investigate the qualitative behavior of both algorithms, figure 12 shows example results for varying numbers of randomly chosen constraints.

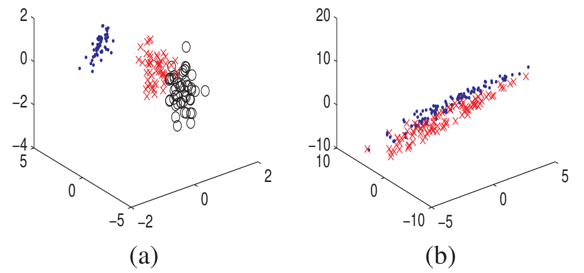


Figure 10. The (a) IRIS and (b) CRABS data sets projected into their first three principal components.

Consider figure 12a, which is representative. With no constraints, both CCL and CKM simply divide the data roughly linearly in half. Constraints cause CKM to slightly alter its chosen centers, but, as suggested earlier, CKM can satisfy instance-level constraints by assigning points to a different cluster from their close neighbors, essentially creating outliers in the middle of qualitatively unchanged clusters: for unconstrained data points, assignment boundaries will still be Voronoi partitions of the feature space. This behavior persists even with large numbers of constraints. CCL, however, deforms the feature space in such a way that the circles lie in disjoint spheres in proximity space.

For all four of these data sets, either must-links or cannot-links are able to shape the proximities in such a way that the desired clusters are easily found. It is worth pointing out that a non-spherical clustering method, for example single-link clustering, can detect some of these non-spherical synthetic patterns. However, the next section demonstrates that our algorithm is effective for real data sets where single-link is completely ineffective.

5.3. Real-World Data

We also give results for several real-world data sets, two of which are shown in figure 10.

- SOYBEAN is the SOYBEAN-LARGE data set from the UCI repository. It has 562 instances, 35 features, and 15 different classes. It is nominal, and Hamming distance is the default metric. The instances represent different soybeans, the features represent qualitative measurements, and the classes are plant diseases.[3]
- IRIS is the classic iris data from (Fisher, 1936). It has 150 instances and 4 features. There are three classes which are relatively separated but non-spherical. The instances represent different irises, the features are structural dimensions, and the classes are iris species.
- CRABS is crabs data from (Campbell & Mahon, 1974). There are 200 instances, 5 features, and 2 classes. The instances represent different crabs, the features represent structural dimensions, and the classes are crab species. This data set is difficult because the the first principal component (essentially crab size) is mostly irrelevant to the target classification.

[3]Wagstaff et al. (2001) reports results on the simpler, smaller SOYBEAN-SMALL set, but we omit this set because unconstrained CL alone yielded a perfect clustering.

Figure 9 shows the accuracy of CCL as constraints are added. Constraints improve performance substantially in every case. CKM is also shown; CCL again outperforms it substantially, supporting the hypothesis that a spatial induction principle is appropriate for real data sets. Note that in the SOYBEAN example, the unconstrained CL algorithm performs worse than unconstrained KM. However, CCL exploits constraints so well that it quickly overtakes CKM in accuracy, whereas a limited number of constraints appears to be ineffective in helping the CKM algorithm.

5.4. Constraint Types

In the results above, constraints were selected by randomly choosing pairs and constraining that pair to have its target equality. In practice, most pairs are cannot linked. However, we argued in section 1 that some applications may have must-links only, cannot-links only, or other mixes of constraints available. This issue is especially important in the present context as Wagstaff et al. (2001) suggest that CKM best exploits cannot-link constraints.

To test the dependence on the mix, figure 11 shows the behaviour of CCL for several different constraint mixes for the SOYBEAN, IRIS, and CRABS data sets. In all cases, CCL's accuracy improves quickly (and faster than CKM) as constraints are added.

5.5. Active Learning

In a real-world domain, one might have control over which pairs to assay. In this case, we would like to choose pairs which we believe will have maximum impact on our accuracy. We claimed that our constraint propagation was intended for the case where the local proximity structure of the feature space was reliable, but the global structure was not. In this case, we might then want to perform CL in an unconstrained fashion until we had some moderate number of clusters remaining. We might then have the scientist supplying the constraints simply make the harder top-level decisions. We believe that this is a valuable intuition, but doing so requires the scientist to supply a quadratic number of constraints. Instead, we propose gradually feeding constraints to the algorithm as it requires them, requiring only a small (hopefully linear) number of constraints to complete the clustering once the algorithm begins to request constraints.

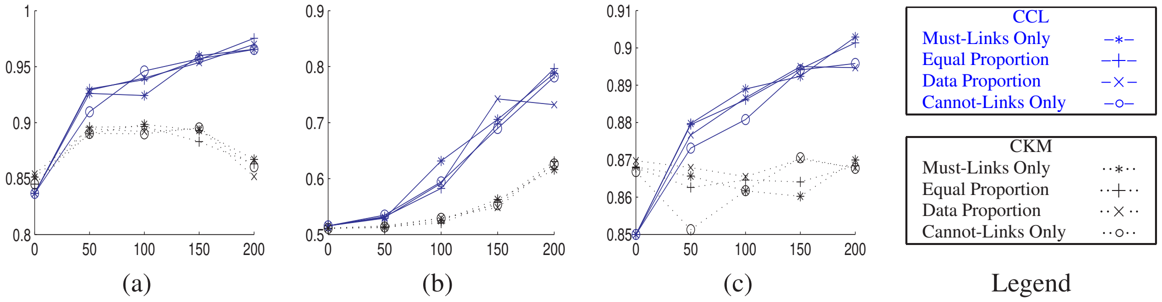


Figure 11. Constraints are effective for CCL over a wide range of mix types, including 100% must-links and 100% cannot-links, as well as mixes in equal proportion and in proportion to the relative number of pair types in the data. (a) IRIS, (b) CRABS, (c) SOYBEAN

More precisely, we implemented the following active learning scheme. The algorithm is told that it will be allowed m pairwise questions. Recall that the merged-cluster distance is always increasing in the CL algorithm. The learner clusters the data without constraints, and determines at what distance cutoff α it can begin asking questions without expecting to need more than m questions. It then clusters until it must make a merge of distance α and asks whether the roots of the next proposed merge belong together. Based on the response, it imposes the constraint accordingly and propagates it on the reduced proximity matrix. It then selects a new merge if needed, and continues. If it keeps proposing bad merges, it might exhaust its questions on a single stage. On the other hand, spatial contraction can cause later merges to be closer than α and several merges may occur before another question is asked. If at any point, it has no questions left, it continues onward, unsupervised.

Figures 8 and 9 show results for active constraint selection. In all cases, actively chosen constraints are much more effective than passively chosen ones.[4] Figure 12 shows the actual constraints chosen on the synthetic sets. The active selection converges to the correct structures very quickly.

6. Conclusions

Previously proposed algorithms for constrained clustering treat pairwise constraints as assertions about individual instances, but fail to exploit spatial implications of those constraints. We have given a method for inducing spatial effects of pairwise constraints and have demonstrated that it substantially outperforms previous approaches, exhibiting behavior which is both quantitatively superior and qualitatively more natural. We have also presented an active learning scheme which dramatically decreases the number of constraints required to achieve a given accuracy.

[4]For some data sets and very small numbers of constraints, passively selected constraints sometimes outperformed the active selection. The appears to be because the active strategy tends to focus questions on hard regions of the data, which can cause unequal contraction across the space. This is an obstacle for CL, which has a bias toward equal-radius clusters.

Acknowledgements

We would like to thank Kiri Wagstaff for helpful discussion and criticism, as well as the anonymous reviewers for suggestions and corrections. This paper is based on work supported in part by the National Science Foundation (under Grant No. IIS-0085896 and by an NSF Graduate Fellowship), and by the research collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University.

References

Campbell, N. A., & Mahon, R. J. (1974). A multivariate study of variation in two species of rock crab of genus *Leptograpsus*. *Australian Journal of Zoology*, *22*, 417–425.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.

Eisenberg, D., Marcotte, E., Xenarios, I., & Yeates, T. O. (2000). Protein function in the post-genomic era. *Nature*, *405*, 823–6.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Fisher, R. A. (1936). The use of multiple measurements in axonomic problems. *Annals of Eugenics*, *7*, 179–188.

Gordon, A. D. (1996). A survey of constrained classification. *Computational Statistics & Data Analysis*, *21*, 17–29.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall.

McQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Symposium on Math, Statistics, and Probability* (pp. 281–297).

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*, 846–850.

Siegel, S., & Castellan, Jr., N. (1988). *Nonparametric statistics for the behavioral sciences*. New York: McGraw-Hill.

Wagstaff, K., & Cardie, C. (2000). Clustering with instance-level constraints. *The Seventeenth International Conference on Machine Learning* (pp. 1103–1110).

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. *The Eighteenth International Conference on Machine Learning*.

Zhu, X., Chu, T., Zhu, J., & Caruana, R. (2001). Heuristically inducing a distance metric from user preferences for clustering. Ms., Carnegie Mellon University, http://www.cs.cmu.edu/~zhuxj/academics/ml/clusterfinal.ps.gz.

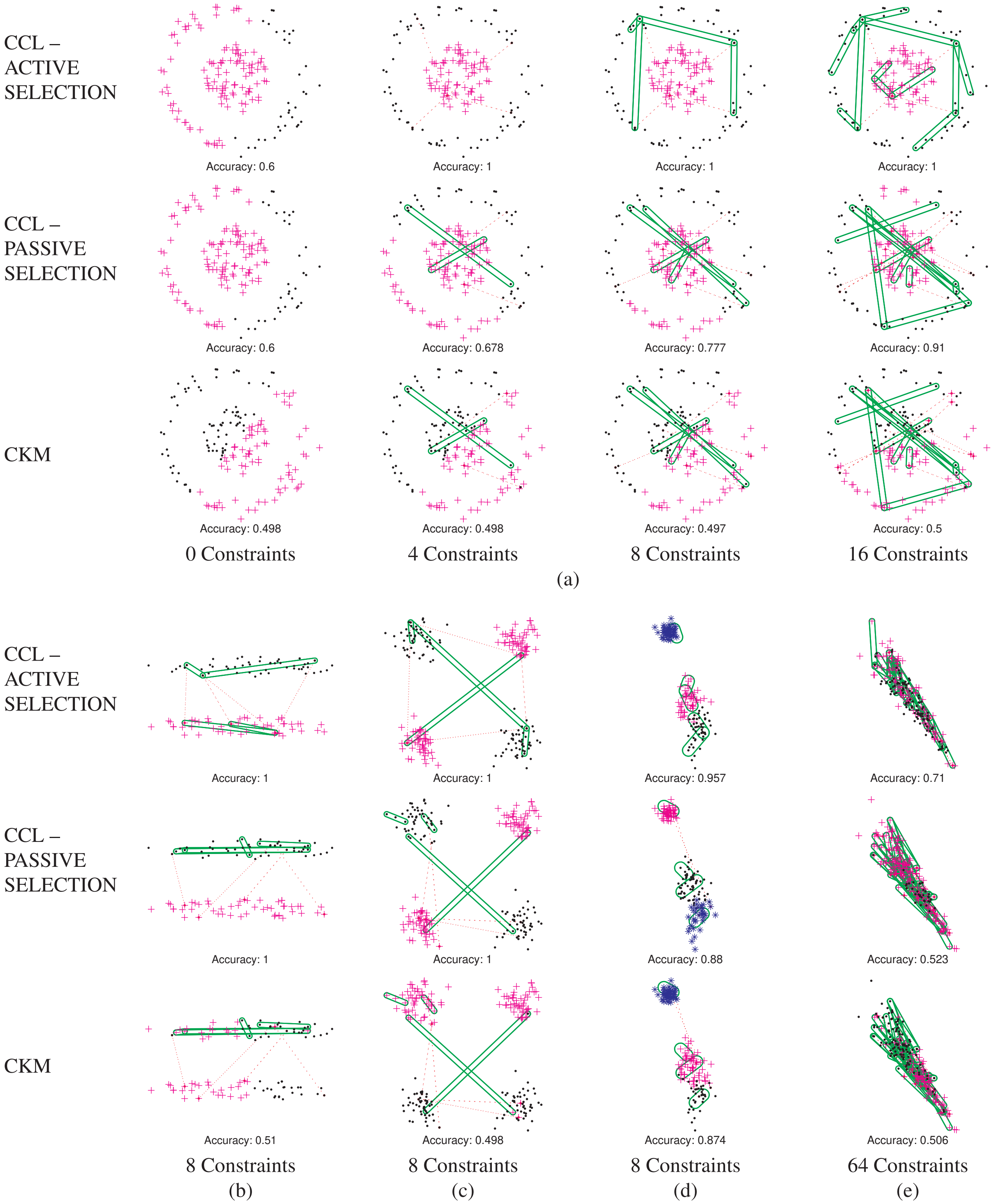


Figure 12. Examples of clustering behavior: (a) CIRCLES, (b) STORMCLOUDS, (c) XOR, (d) IRIS, (e) CRABS. Loops indicate must-link pairs, dashed lines indicate cannot-link pairs.