Clustering with modified cosine distance learned from constraints

Leonid Rachevsky, Dimitri Kanevsky, Ruhi Sarikaya, Bhuvana Ramabhadran

IBM T.J. Watson Research Center, Yorktown, NY 10598, USA
{lrachevs, kanevsky, sarikaya, bhuvana}@watson.ibm.com

Abstract

In this paper we present a modified cosine similarity metric that helps to make features more discriminative. The new metric is defined via various linear transformations of the original feature space to a space in which these samples are better separated. These transformations are learned from a set of constraints representing available domain knowledge by solving related optimization problems. We present results on two natural language call routing datasets that show significant improvements ranging from 3% to 5% absolute in the purity of clusters obtained in an unsupervised fashion.

Index Terms: constrained clustering, cosine metric, SVM, TF-IDF

10.21437/Interspeech.2011-437

1. Introduction

Statistically based solutions to speech and language processing problems often require manual annotation to learn the structures in the labeled data and to make generalizations to unseen data. As such, data labeling is a key step impacting the subsequent steps in building statistical speech and language applications. Call-routing is one such application where the user utterances are labeled with one of N classes. Minimizing the human labor needed for labeling the data is crucial for rapid deployment of these speech and language applications [9]. Effective clustering algorithms play an important role towards reducing the human labor component, as they organize numerous sentences into a small number of meaningful clusters and simplify the process of assigning appropriate labels to unlabeled data. The performance of clustering algorithms such as K-Means strongly depends on the distance metric defined in the feature space, since samples can be better separated in one metric than in others. For example, in all our experiments with unsupervised K-Means clustering of textual data, the results were better by 3-6% by using a cosine metric instead of Euclidian. Let us assume the data set has some additional information found by a domain expert. This knowledge can be presented in two ways: pair-wise constraints (*must-link* and *cannot-link*) and labeled data. The pair-wise constraints contain a set of pairs of samples of two types: those that must be in the same cluster and those that cannot reside together. The labeled data is usually very limited and does not exceed 10-15% of the whole data set. Obviously, the pair-wise constraints can always be obtained from the labeled data. The labeled data in turn can be generated from the pair-wise constraints only if the pairs do not contradict with each other. But in practice a set of pairwise constraints can be contradictive. For example, let A denote a sample phrase "open new account", B denote "account information" and C denote "cancel account". Samples A and B as well as B and C can be easily viewed by a domain expert defining pair-wise constraints as *must-link* pairs, especially if they are not presented in the data one after the other. If later on the A,C pair is marked as *cannot-link* the set of constraints becomes contradictive.

There are two major directions in clustering with side information: one is a constrained clustering invented by K. Wagstaff et al ([1]), the other is based on learning distance metrics using side information ([2]-[5], [9]-[10]). We observed that the constrained K-Means [1] method quality gets worse as more and more samples are added to the clusters. The impact of seeded samples decreases on every iteration when new centroids are recalculated. The centroids drift away from their initial locations, as a result the desired domain knowledge encoded in the set of constraints disappears. Cosine similarity based distance metrics were more efficient in text classification than the Euclidean metric. The purpose of this work is to learn the cosine distance metric from constraints, apply the modified cosine distance to K-Means algorithm, and find out how it affects the clustering quality.

The rest of the paper is structured as follows. In Section 2 we introduce a general concept of clustering using modified metrics. In the section Section 3 we introduce three clustering methods that are based on a modified cosine distance metric, a matrix composed of support vectors, and a Term Frequency-Inverse Document frequency (TF–IDF) matrix. In Section 4 and 5 we describe experimental results and provide conclusions.

2. Clustering with modified distance metrics

Consider a set of observations (training set) $X = \{x_t\} \in \mathbb{R}^n, t = 1, .., n$. Assume that we have a set of constraints in a form

$$\delta : X \times X \to \{0, 1\} \tag{1}$$

where x, y belong to one group if $\delta(x, y) = 1$ and different groups if $\delta(x, y) = 0$. Here "groups" could mean anything such as classes (labels), clusters, *must-link* and *cannot-link*, or other structures.

Let us introduce a transformation $\Lambda = \Lambda_\delta : X \to \mathbb{R}^n$ that maps X into $\mathbb{R}^m$, and approximates the constrains (1) in some sense. For example, if $\delta(x, y) = 1$ then $\Lambda(x)$ becomes closer to $\Lambda(y)$ in some metric and, opposite, if $\delta(x, y) = 0$ then $\Lambda(x)$ becomes further away from $\Lambda(y)$ in this metric. The process of finding transformations approximating constraints can be formulated using a discrimination objective function. This transformation Λ allows modificaion of a scalar product on a vector space by the following rule:

$$< x, y >_\Lambda = < \Lambda(x), \Lambda(y) > \tag{2}$$

One can then consider K-Means and constrained K-Means clustering on some test set Y using the modified metric , i.e. the modified scalar product $<, >_\Lambda$.

Copyright © 2011 ISCA

3. K-Means with modified cosine distance

Here we provide three different ways for constructing transformations that approximate *must-link* and *cannot-link* constraints: diagonal, based on a SVM map and based on TF-IDF weight map. Experimental results for these methods are described in this next section.

3.1. Diagonal transformations

Let $X = \{x_t\} \in \mathbb{R}^n, t = 1, 2, ...T$ be a training sample. Let us represent *must-link* and *cannot-link* conditions as two sets of pairs $(i, j) \in I_1, (i, j) \in I_2$ where I_1, I_2 are sets of pairs of integers (note that I_1 and I_2 can have non-empty intersection). Assume also that we want to transform a feature space in such a way that *must-link* pairs become closer and *cannot-link* pairs become further in some metrics. We can then hope that this transformation will improve similarity for test features belonging to the same classes and increase dissimilarity for features belonging to different classes.

In order to make *must-link* and *cannot-link* representation more discriminative one can look for a transformation Λ of $\mathbb{R}^n$ that ideally satisfies the following set of equalities:

$$\cos(\Lambda x_i, \Lambda x_j) = (\Lambda x_i, \Lambda x_j)/|\Lambda x_i||\Lambda x_j| = 1 \text{ if } (i, j) \in I_1 \tag{3}$$

and

$$\cos(\Lambda x_i, \Lambda x_j) = (\Lambda x_i, \Lambda x_j)/|\Lambda x_i||\Lambda y_j| = 0 \text{ if } (i, j) \in I_2 \tag{4}$$

In other words, features that are similar are mapped into proportional vectors and dissimilar features are mapped into orthogonal vectors. In general for sufficiently large sets of constraints I_1, I_2 it is impossible practically to construct the map Λ that satisfies (3, 4). Approximation to (3) are the following equations that move similar features closer and dissimilar features further in the cosine metric.

$$\cos(\Lambda x_i, \Lambda x_j) = (\Lambda x_i, \Lambda x_j)/|\Lambda x_i||\Lambda x_j| < \cos(x_i, x_j) \text{ if } (i, j) \in I_1 \tag{5}$$

and

$$\cos(\Lambda x_i, \Lambda x_j) = (\Lambda x_i, \Lambda x_j)/|\Lambda x_i||\Lambda y_j| > \cos(x_i, y_j) \text{ if } (i, j) \in I_2 \tag{6}$$

For practical purposes of clustering textual data, one can consider the following transformations as diagonal $n \times n$-dimensional matrices with all entries that are positive:

$$\Lambda = (\lambda(1), ..., \lambda(n)) \in \mathbb{R}^n_+ : x = (x(1), ...x(n)) \rightarrow \Lambda x = (\lambda(1)x_1, ...\lambda(n)x_n) \tag{7}$$

that acts on $\mathbb{R}^n$ (here $\mathbb{R}^n_+$ means positive entries in $\mathbb{R}^n$).
The conditions (6) can be approximated by the following "average" function:

$$F(\Lambda) = F(X, Y, I; \Lambda) = \sum_{(i,j)\in I_1} \cos(\Lambda x_i, \Lambda y_j) - \sum_{(i,j)\in I_2} \cos(\Lambda x_i, \Lambda y_j) \tag{8}$$

Here we use the fact that feature vectors that represent textual data consist of counts of words in a set of sentences, i.e. they are nonnegative. They remain nonnegative also after application of a transformation Λ consisting of positive entries. Therefore all cosine expressions in (8) are nonnegative and this allows us to look for a transformation Λ that optimizes the objective function $F(\Lambda)$ in (8) subject conditions $\lambda(s) > 0, s = 1, ..n$

3.2. Support Vector Machine map

Consider a set of observations (training set) $X = \{x_t\} \in \mathbb{R}^n, t = 1, .., T$. Assume that we have a set of m labels that split X in m classes and let us use SVM associated with these classes that produces m support vectors $s_i \in \mathbb{R}^{n+1}, i = 1, ...m$ (the dimension $n + 1$ represents the dimension of the feature space and a bias). Let a matrix $S = [s_1, ...s_m] \in \mathbb{R}^{m,n}$ is obtained by conjugating m support vectors as m rows. The matrix S defines a projection $x \in \mathbb{R}^n \rightarrow Sx \in \mathbb{R}^m$ from n-dimensional space into m dimensional space. We call this matrix Support Vector Map (SVMap). In practical textual applications, number of classes is less than dimension of the feature space X that coincides with a vocabulary size for the bag of words (features) and therefore S defines the projection to a lower dimensional space. The matrix S maps a test set Y into a new feature set $Z = \{S(y), y \in Y\}$. We then apply to this set Z K-Means. This creates clusters on Z that induce clusters on Y by the rule: $y_1 \sim y_2 \in Y$ if $Sy_1 \sim Sy_2 \in Z$ where $z_1 \sim z_2$ in Z if z_i belongs to the same cluster. The matrix S also maps constraints in the feature space into new constraints in Z. For example, *must-link* (cor. *cannot-link*) features are mapped into *must-link* (cor. *cannot-link*) features in Z . We call map S composed of support vectors as Support Vector Map (SVMap). SVMap is applied to original feature vectors to get their images that are better separated.

3.3. Term frequency-inverse document frequency map

The SVM map of feature vectors that was described in the previous section is not the only map defined on the feature space. Another map can be constructed via term-frequency inverse document frequency (TF–IDF). Given training data X and a set of m labels on X an (m, n) matrix T is constructed using TF–IDF. The $T_{i,j}$ is defined as the product of the term frequency (i.e. a count of the ith feature in utterances labeled by jth label divided by the sum of all feature counts of this label) and the inverse document frequency (i.e. the logarithm of the total number of labels divided by the number of labels by which utterances containing this feature were labeled). The matrix T defines a projection $x \in \mathbb{R}^n \rightarrow Tx \in \mathbb{R}^m$. The image of this map can be clustered using K-Means.

4. Experiments

4.1. Dataset: Product delivery company

Training set consists of 15 labels, 600 samples; testing set consists of 25 labels, 4783 samples (10 labels of the testing set are not presented in the training data). In Table 5 each cell shows purity [%] per # of computed clusters. Purity is computed as:

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \bigcap c_j| \tag{9}$$

where N is a total number of utterances that divide the count of all labels c_j that are most frequently assigned to all clusters ω_k. The following types of metrics were used for rows in this table: "Cosine" for K-Means using cosine metrics, "DT" for modified cosine (diagonal transformation), "TF–IDF" for modified cosine with the TF–IDF map, "SVM" for modified cosine with the SVM map, "TFIDF+W" for modified cosine the TF–IDF map combined with the Wagstaff method, "SVM+W" for modified cosine with the SVM map combined with the Wagstaff method. By Wagstaff method we refer to the approximation of constrained clustering invented by Wagstaff ([1]). In the

context of our work this method means the following. If the number of requested clusters ($N_{cluster}$) exceeds the number of labels in the training set (N_{labels}), the initial N_{labels} clusters consist of labeled samples available in the training data. These samples cannot ever be reassigned to other clusters and centroids for these N_{labels} clusters are computed by finding a mean of initial samples that share the same label. Remaining $N_{clusters} - N_{labels}$ clusters are initialized randomly as in a regular K-Means algorithm.
In Table 5 each column represents results for requested clusters (16, 40, 60 and 100). The number of clusters that are actually computed are obtained using the following procedure: in each found cluster the dominating label is found. All clusters sharing the same dominating label are merged.

Table 1: *Product delivery company*

TYPE	16 clust	40 clust	60 clust	100 clust
Cosine	35.31/9	46.29/11	47.04/13	54.42/17
DT	38.47/9	50.70/13	50.86/13	58.37/17
TFIDF	65.84/14	68.20/17	70.48/18	72.01/20
SVMap	68.72/15	71.13/19	72.19/20	72.59/20
TFIDF+W	68.86/14	72.17/17	73.53/18	75.21/20
SVMap+W	**71.67**/15	**74.34**/19	**75.37**/19	**75.77**/21

The TF–IDF classification gave 64.39% accuracy and the SVM classification gave 67.17% (for 15 labels).

Table 2: *Product delivery company, range 400-1000 clusters*

TYPE	400 clust	600 clust	800 clust	1000 clust
TFIDF	76.46/22	78.28/22	79.47/22	80.53/23
SVMap	75.43/22	76.85/23	77.59/23	78.57/22
TFIDF+W	**78.97**/22	**80.60**/22	**81.70**/22	**82.76**/23
SVMap+W	78.10/22	79.27/23	80.14/22	80.98/22

Our training set is smaller than our test set. This was done intentionally to speed up the tagging process,when building a new application such as call–routing to help the domain expert to assign labels to the unlabeled data. We presume that 10–15% of labeled data is what the tagger might have, and 85–90% of the data has no labels. We would like to show that our method is capable of discovering new clusters (labels) unseen in the training set.

As we can see from table 1 six new clusters were discovered in SVMap+W case (75.77/21) in addition to the 15 classes available in the training set. The clustering quality was improved as well, i.e. 75.77% comparing to SVM 67.17% classification accuracy.

Even though clustering using SVMap+W outperforms other ways of supervised clustering we explored, there are situations when no classifier can be used to learn constraints. Specifically, if only *cannot-link* pairs are given, it is not clear how to present this knowledge in a traditional phrase-label format expected by SVM, TF–IDF, and other types of classifiers. As a result, no transformation matrix associated with a classifier can be constructed. In such cases, we may still use diagonal transformation matrices to improve clustering quality.

Following examples illustrate how TF–IDF map makes features more discriminative.

We placed six pairs of phrases from each of two just discovered clusters missing in the training set into Table 3. Cells in these tables show a distance between phrases composing a pair residing in the first column. As one can observe, the learned relationships between features encapsulated in the transformation matrix, help to minimize distance between related phrases and thus increase clustering quality.

Table 3: *Discriminative example for a TF–IDF matrix A*

Pair of phrases	$1 - cos(x, y)$	$1 - cos(Ax, Ay)$
x_1 vs y_1	0.36363643	0.038611114
x_2 vs y_2	0.5454546	0.047801137
x_3 vs y_3	0.5441577	0.047618926
x_4 vs y_4	0.4545455	0.011376977
x_5 vs y_5	0.65811825	0.0086689
x_6 vs y_6	0.65811825	0.014387965

In this table x_i and y_i denotes the following phrases:

x_1 **:** "*I need to change my order for this coming delivery*"
y_1 **:** "*I wanna change my order for for tomorrow*"
x_2 **:** "*I need to change my order for this coming delivery*"
y_2 **:** "*that's not satisfactory I wanna change that order*"
x_3 **:** "*I need to change my order for this coming delivery*"
y_3 **:** "*I'd like to request an order change*"
x_4 **:** "*I wanna change my order for for tomorrow*"
y_4 **:** "*that's not satisfactory i wanna change that order* "
x_5 **:** "*I wanna change my order for for tomorrow*"
y_5 **:** "*I'd like to request an order change*"
x_6 **:** "*that's not satisfactory i wanna change that order*"
y_6 **:** "*I'd like to request an order change*"

4.2. Dataset: Package shipment company

Training set: 22 labels, 683 samples; testing set: 34 labels, 16720 samples (12 labels of the testing set are not presented in the training data, 1 label in the training set is not presented in the testing data).

Table 4: *Package shipment company*

TYPE	25 clust	40 clust	60 clust	100 clust
Cosine	25.72/11	29.22/14	33.81/16	35.64/17
DT	27.33/11	34.88/15	39.17/17	40.22/17
TFIDF	52.64/17	53.99/19	58.33/19	61.12/20
SVMap	65.57/20	65.58/19	65.84/20	66.73/19
TFIDF+W	54.64/17	55.93/20	60.25/19	62.10/20
SVMap+W	**66.86**/20	**66.91**/19	**67.08**/20	**67.67**/21

The TFIDF classification gave 59.04% accuracy and the SVM classification gave 65.45% (for 22 labels)

For this Package shipment company, it was observed that for 200 -1000 requested clusters one new cluster was discovered, while purity was improved by 5-7% (absolute) compared to SVM classification accuracy.

Table 5: *Package shipment company, range 400-1000 clusters*

TYPE	400 clust	600 clust	800 clust	1000 clust
TFIDF	66.25/21	67.46/22	68.21/22	69.22/22
SVMap	68.71/20	69.80/22	70.22/23	70.89/23
TFIDF+W	67.86/22	68.67/22	69.21/22	70.68/22
SVMap+W	**69.93**/22	**70.91**/23	**71.31**/23	**72.00**/23

The following are examples how TF–IDF maps make features more discriminative for this dataset. As before we placed six pairs of phrases from each of two just discovered clusters missing in the training set into Table 6.

Table 6: *Discriminative example for a TF–IDF matrix A*

Pair of phrases	$1 - cos(x, y)$	$1 - cos(Ax, Ay)$
x_1 vs y_1	0.5917517	0.12574834
x_2 vs y_2	0.903775	0.06915957
x_3 vs y_3	0.68377227	0.12003714
x_4 vs y_4	0.7642977	0.10059303
x_5 vs y_5	0.61270165	0.0936116
x_6 vs y_6	0.81742585	0.11637181

In this table x_i and y_i denotes the following phrases:

x_1 **:** "*do we have a credit on our account*"

y_1 **:** "*have a question regarding my bill*"

x_2 **:** "*do we have a credit on our account*"

y_2 **:** "*I wanna find out the amount of the bill*"

x_3 **:** "*do we have a credit on our account*"

y_3 **:** "*I'm checking on some charges on my bill*"

x_4 **:** "*have a question regarding my bill*"

y_4 **:** "*I wanna find out the amount of the bill* "

x_5 **:** "*have a question regarding my bill*"

y_5 **:** "*I'm checking on some charges on my bill*"

x_6 **:** "*I wanna find out the amount of the bill*"

y_6 **:** "*I'm checking on some charges on my bil*"

4.3. Number of clusters for human annotation

There is a tradeoff for the number of requested clusters and the amount of effort required to find-and-merge clusters. Namely, requesting too many clusters is undesirable, even though the purity measurement approaches 100%. The extreme case is when every cluster has only one sample: in other words, no tag/label suggestions were made by the algorithm. Requesting too few clusters may lead to the discovery of clusters consisting of large amounts of irrelevant data. Building a call routing application requires labeling the data according to call-types.

The human annotators typically do not label sentences one by one, but rather look at the clusters that are automatically generated. Then they clean up these clusters by removing impurities (i.e. sentences belonging to those clusters) There are pros/cons of having too many and too few clusters. For example, if the human annotators would have too few (e.g. 20-30) then the clusters would be to large to clean up. And if there would be too many (e.g.more than 1000) then the labor/effort would shift towards merging these purer but smaller clusters. As empirical observation shows, the annotators usually found that 300-500 clusters were a good target to strive for across multiple call routing tasks.

5. Conclusions

We described a promising clustering technique: modified cosine metrics where transformation is defined by constraints. We gave three examples of this technique: diagonal transformations, SVMap and TF–IDF maps and a combination of these maps with the Wagstaff constrained method. Experiments on real data sets demonstrated that our technique improved clustering of textual data. The best results were obtained using the combination of SVMap and Wagstaff methods. We plan to investigate what impact a chain of transformations (SVMap + DT or TF–IDF + DT, etc.) may have on the clustering quality.

6. References

[1] Kiri Wagstaff, Claire Cardie, Seth Rogers and Stefan Schroedl, "Constrained K-means Clustering with Background Knowledge", Proceedings of the Eighteenth International Conference on Machine Learning, 2001, p. 577584.

[2] E. P. Xing, A. Y. Ng, M. I. Jordan and S. J. Russell, "Distance metric learning with application to clustering with side-information", Proc. NIPS., 2002.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, Learning distance functions using equivalence relations, In The Twentieth International Conference on Machine Learning (ICML), 2003

[4] B. Alipanahi, M. Biggs, and A. Ghodsi. Distance metric learning versus Fisher discriminant analysis, Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI), 2008

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. ACM Computing Surveys, 31(3):264323, 1999.

[6] Asa Ben-Hur, David Horn, Hava T. Siegelmann and Vladimir Vapnik"Vector Clustering", The Journal of Machine Learning Research, 125-137, 2001.

[7] Tao Q, Wu GW, Wang FY and Wang J., "Posterior probability support vector machines for unbalanced data", IEEE Trans Neural Netw, v. 16(6), 1561-73, 2005

[8] Vladimir Vapnik, "Statistical Learning Theory", 1998, John Willey & Sons

[9] R. Sarikaya, "Rapid Bootstrapping of Statistical Spoken Dialog Systems", Speech Communications, vol. 50, no. 7, pp: 580-593, July 2008.

[10] Kilian Q.Weinberger, Lawrence K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification", Journal of Machine Learning Research 10 (2009) 207-244.