# Iterative Double Clustering for Unsupervised and Semi-supervised Learning

Ran El-Yaniv and Oren Souroujon

Computer Science Department, Technion - Israel Institute of Technology
{rani, orenso}@cs.technion.ac.il

**Abstract.** This paper studies the Iterative Double Clustering (IDC) meta-clustering algorithm, a new extension of the recent Double Clustering (DC) method of Slonim and Tishby that exhibited impressive performance on text categorization tasks [1]. Using synthetically generated data we empirically demonstrate that whenever the DC procedure is successful in recovering some of the structure hidden in the data, the extended IDC procedure can incrementally compute a dramatically better classification, with minor additional computational resources. We demonstrate that the IDC algorithm is especially advantageous when the data exhibits high attribute noise. Our simulation results also show the effectiveness of IDC in text categorization problems. Surprisingly, this unsupervised procedure can be competitive with a (supervised) SVM trained with a small training set. Finally, we propose a natural extension of IDC for (semi-supervised) transductive learning where we are given both labeled and unlabeled examples, and present preliminary empirical results showing the plausibility of the extended method in a semi-supervised setting.

## 1 Introduction

Data clustering is a fundamental and challenging routine in information processing and pattern recognition. Informally, when we cluster a set of elements we attempt to partition it into subsets such that points in the same subset are more "similar" to each other than to points in other subsets. Typical clustering algorithms depend on a choice of a similarity measure between data points [2], and a "correct" clustering result can be dependent on an appropriate choice of a similarity measure. However, the choice of a "correct" measure is an ill-defined task without a particular application at hand. For instance, consider a hypothetical data set containing articles by each of two authors, so that half of the articles authored by each author discusses one topic, and the other half discusses another topic. There are two possible dichotomies of the data which could yield two different bi-partitions: one according to topic, and another, according to writing style. When asked to cluster this set into two sub-clusters, one cannot successfully achieve the task without knowing the goal: Are we interested in clusters that reflect writing style or semantics? Therefore, without a suitable target at hand and a principled method for choosing a similarity measure suitable for the target, it can be meaningless to interpret clustering results.

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 121–132, 2001.

The *information bottleneck (IB)* method of Tishby, Pereira and Bialek [3] is a new framework that can sometimes provide an elegant solution to this problematic aspect of data clustering. Let the data be a set of observed values of some random variable $S$. The main idea of the IB method is to use some "target information" that can guide the clustering process towards the desired goal. Let this target information be the observed values of the random variable $T$. The goal is to extract the essence of $S$ which can still predict $T$. Thus, we aim to compute a compressed representation $\tilde{S}$ of $S$ (e.g., a clustering of $S$) so as to preserve $I(\tilde{S}, T)$, the mutual information between $\tilde{S}$ and $T$, as much as possible. Tishby et al. [3] also present an algorithm for computing the desired clustering $\tilde{S}$, which is a "soft" assignment of data points into cluster centroids.

In [4], Slonim and Tishby developed a simplified "hard" variant of this IB clustering, where there is a hard assignment of points to their clusters. Employing this hard IB clustering, the same authors introduced an effective two-stage clustering procedure called *Double Clustering (DC)* [1]. Roughly, the idea is as follows. Let $X$ represent the data where each data point is a vector over real valued features. The goal is to cluster $X$ based on the joint empirical distribution of data points and their features. Informally, during the first stage, a clustering of the features $F$ is performed using the above $(S, T)$-IB procedure with $S$ representing the features and $T$ representing the data points. This results in a compressed representation $\tilde{F}$ of the features $F$ that preserves information about the points. During the second stage, the data $X$ is compressed with respect to the feature clusters $\tilde{F}$ (i.e. with $S = X$ and $T = \tilde{F}$), thus generating a clustering $\tilde{X}$ of the data $X$. An experimental study of DC on text categorization tasks [1] showed a consistent advantage of IB clustering over other clustering methods. A striking finding in [1] is that DC sometimes even attained results close to those of supervised learning.[1]

In this paper we present a powerful extension of the DC procedure which we term *Iterative Double Clustering (IDC)*. As its name suggests, IDC performs iterations of DC so that the input data variable of the next iteration is the clustered data of the previous DC iteration (and the first iteration is a standard DC). Whenever the first DC iteration succeeds in extracting a meaningful structure of the data, a number of the next consecutive iterations can continually improve the clustering quality. Intuitively, this continual improvement achieved by IDC is due to generation of progressively less noisy target variables that serve better the IB clustering of the features (see details in Section 3). Using synthetically generated data, we study some properties of the IDC method. Not only that IDC can dramatically outperform DC whenever the data is noisy, our experiments indicate that IDC attains impressive categorization results on text categorization tasks. In particular, we show that our unsupervised IDC procedure can outperform an SVM trained over a small sized training set.

These findings and the studies of Baker and McCallum [5] and of Slonim and Tishby [4,6] on *supervised* applications of the information bottleneck for

[1] Specifically, the DC method obtained in some cases accuracy close to that obtained by a naive Bayes classifier trained over a small sized sample [1].

word clustering provided us with the motivation for extending the IDC method into a semi-supervised setting, and we propose a natural extension of IDC for transductive and semi-supervised learning. Our preliminary empirical results indicate that our transductive IDC can yield effective text categorization.

## 2 Problem Setup and Preliminaries

We consider a data set $X$ of *elements*, each of which is a $d$-dimensional vector over a set $F$ of *features*. In this paper we focus on the case where feature values are non-negative real numbers. For every element $x = (f_1, \ldots, f_d) \in X$ we consider the empirical conditional distribution $\{p(f_i|x)\}$ of features given $x$, where $p(f_i|x) = f_i / \sum_{i=1}^{d} f_i$. For instance, $X$ can be a set of documents, each of which is represented as a vector of word-features where $f_i$ is the frequency of the $i$th word (in some fixed word enumeration). Thus, we represent each element as a distribution over its features, and are interested in a partition of the data based on these feature conditional distributions. Given a predetermined number of clusters, a straightforward approach to cluster the data using the above "distributional representation" would be to choose some (dis)similarity measure for distributions (e.g. based on some $L_p$ norm or some statistical measure such as the KL-divergence) and employ some "plug-in" clustering algorithm based on this measure (e.g. agglomerative algorithms). Perhaps due to feature noise, this simplistic approach can result in mediocre results (see e.g. [1]).

### 2.1 The Information Bottleneck Method

Let $S$ and $T$ be two random variables. The *information bottleneck (IB)* method [3] aims to extract a compressed representation $\tilde{S}$ of $S$ with minimum compromise of information content with respect to the variable $T$. Let $I(S,T) = \sum_{s \in S, t \in T} p(s,t) \log \frac{p(s,t)}{p(s)p(t)}$, the *mutual information* between $S$ and $T$ [7]. The IB method attempts to compute $p(\tilde{s}|s)$, a "soft" assignment of a data point $s$ to clusters $\tilde{s}$, so as to minimize $I(S,\tilde{S}) - \beta I(\tilde{S},T)$, given the Markov condition $T \to S \to \tilde{S}$ (i.e., $T$ and $\tilde{S}$ are conditionally independent given $S$). Here, $\beta$ is a Lagrange multiplier that controls a constraint on $I(\tilde{S},T)$. As shown in [3], this minimization yields a system of coupled equations for the clustering mapping $p(\tilde{s}|s)$ in terms of the cluster representations $p(t|\tilde{s})$ and the cluster weights $p(\tilde{s})$. The paper [3] also presents an algorithm similar to deterministic annealing [8] for recovering a solution for the coupled equations.

Slonim and Tishby [4] proposed a simplified IB approach for the computation of "hard" cluster assignments. Specifically, for any cardinality $m = |\tilde{S}|$ (i.e. $m$ is the desired number of clusters), the coupled equations in [3] induce, in the limit $\beta \to \infty$, the following coupled equations.

$$p(\tilde{s}|s) = \begin{cases} 1, & \text{if } s \in \tilde{s} \\ 0, & \text{otherwise} \end{cases} \qquad \forall s \in S$$

$$p(t|\tilde{s}) = \frac{1}{p(\tilde{s})} \sum_{s \in \tilde{s}} p(s,t) \qquad \forall t \in T$$

$$p(\tilde{s}) = \sum_{s \in \tilde{s}} p(s)$$

Using the above equations and the identity $I(S,T) = \sum_{s \in S, t \in T} p(s)\, p(t|s) \log \frac{p(t|s)}{p(t)}$, a greedy agglomerative clustering algorithm was proposed in [4]. This algorithm initializes with a trivial clustering, where each data point $s$ is a single cluster. Then, at each step, the algorithm merges the two clusters that minimize the loss of mutual information $I(\tilde{S},T)$. As shown in [4], the reduction in $I(\tilde{S},T)$ due to a merge of two clusters $\tilde{s}_i$ and $\tilde{s}_j$ is

$$(p(\tilde{s}_i) + p(\tilde{s}_j)) D_{JS}[p(t|\tilde{s}_i), p(t|\tilde{s}_j)], \tag{1}$$

where, for any two distributions $p(x)$ and $q(x)$, with priors $\lambda_p$ and $\lambda_q$, $\lambda_p + \lambda_q = 1$, $D_{JS}[p(x), q(x)]$ is the *Jensen-Shannon divergence* (see [9,10]),

$$D_{JS}[p(x), q(x)] = \lambda_p D_{KL}(p||\frac{p+q}{2}) + \lambda_q D_{KL}(q||\frac{p+q}{2}).$$

Here, $\frac{p+q}{2}$ denotes the distribution $(p(x)+q(x))/2$ and $D_{KL}(\cdot||\cdot)$ is the Kullbak-Leibler divergence [7]. The Jensen-Shannon divergence is non-negative, symmetric and bounded but does not satisfy the triangle inequality (and therefore is not a metric). Note that the priors used in the Jensen-Shannon divergence (1) for $p(t|\tilde{s}_i)$ and $p(t|\tilde{s}_j)$ are proportional to $p(\tilde{s}_i)$ and $p(\tilde{s}_j)$, respectively. For a detailed description of the (soft and hard) IB method the reader is referred to [3,4].

As all agglomerative clustering algorithms, the agglomerative IB clustering algorithm derived using the dissimilarity measure of Equation (1) is only *locally* optimal, since at each step it greedily merges the two most similar clusters. Another disadvantage of this algorithm is its time complexity, which can be accomplished in $O(n^2)$ for a data set of $n$ elements (see [1] for details).

The IB method can be viewed as a meta-clustering procedure that, given observations of the variables $S$ and $T$ (via empirical co-occurrence samples of $p(s,t)$), attempts to cluster $s$-elements represented as distributions over $t$-elements. Clearly, one can attempt to approximate IB clustering using any vectorial clustering algorithm that can be applied within the simplex containing the distributional representations of the $s$-elements.

### 2.2 Double Clustering (DC)

Employed with the IB clustering method we now return to our problem setup (see Section 2) and describe the double clustering (DC) method of [1]. Notice that in the IB clustering method the roles of the random variables $S$ and $T$ can be switched. In our problem setup, this means that we can cluster elements as distributions over features, but can also cluster features as distributions over

elements. For instance, we can cluster documents as distributions over words but also cluster words as distributions over documents.

DC is a two-stage procedure where during the first stage we cluster features represented as distributions over elements, thus generating *feature clusters*. During the second stage we cluster elements represented as distributions over the feature *clusters* (a more formal description follows). For instance, considering the document clustering domain, in the first stage we cluster words as distributions over documents to obtain word clusters. Then in the second stage we cluster documents as distributions over word clusters, to obtain document clusters.

Intuitively, the first stage in DC generates more coarse *pseudo* features (i.e. feature centroids), which can reduce noise and sparseness that might be exhibited in the original feature values. Then, in the second stage, elements are clustered as distributions over the "distilled" pseudo features, and therefore can generate more accurate element clusters. As reported in [1], this DC two-stage procedure outperforms various other clustering approaches as well as DC variants applied with other dissimilarity measures (such as the variational distance) different from the optimal JS-divergence of Equation (1). It is most striking that in some cases, the accuracy achieved by DC was close to that achieved by a supervised Naive Bayes classifier.

**Input:**
$X$ (input data)
$N_{\tilde{X}}$ (number of element clusters)
$N_{\tilde{F}}$ (number of feature clusters to use)
$k$ (number of iterations)
**Initialize:** $S \leftarrow F$, $T \leftarrow X$,
**loop** $\{k$ times$\}$
  $N \leftarrow N_{\tilde{F}}$
  $\tilde{F} \leftarrow IB_N(T|S)$
  $N \leftarrow N_{\tilde{X}}$, $S \leftarrow X$, $T \leftarrow \tilde{F}$
  $\tilde{X} \leftarrow IB_N(T|S)$
  $S \leftarrow F$, $T \leftarrow \tilde{X}$
**end loop**
**Output** $\tilde{X}$

**Fig. 1.** Pseudo-code for IDC

## 3 Iterative Double Clustering (IDC)

Our Iterative Double Clustering (IDC) is a novel extension of the DC algorithm, which performs multiple iterations of the original DC. It works by feeding the element clusters output of each DC iteration as input to the first stage of the following DC iteration. As we shall later see, IDC consistently improves (in a sense to be defined) over DC whenever the data is noisy.

Denote by $IB_N(T|S)$ the clustering result, into $N$ clusters, of the information bottleneck hard clustering procedure when the data is $S$ and the target variable is $T$ (see Section 2.1). For instance, if $T$ represents documents and $S$ represents words, the application of $IB_N$(documents|words) will cluster the words, represented as distributions over the documents, into $N$ clusters. Using the notation of our problem setup, with $X$ denoting the data, and $F$ denoting the features, suppose we are interested in clustering $X$ into $N_{\tilde{X}}$ clusters. In Figure 1 we pro-

vide a pseudo-code of the IDC meta-clustering algorithm. Note that the DC procedure is simply an application of IDC with $k = 1$.

The code of Figure 1 requires to specify $k$, the number of IDC iterations to run. It also requires as input both $N_{\tilde{X}}$, the number of element clusters (e.g. the desired number of of document clusters), and $N_{\tilde{F}}$, the number of feature clusters to use during each iteration. The question of how to choose these parameters requires study and some of the results we present in this paper provide partial answers. In general, a correct choice of $N_{\tilde{X}}$ is a model selection problem and is beyond the scope of this paper. In the experiments reported in the paper we always assumed that we know the correct $N_{\tilde{X}}$. A correct choice of $N_{\tilde{F}}$ is also related to model selection, but we cannot avoid it. Fortunately, as we discuss later, the algorithm is not too sensitive to an overestimate of $N_{\tilde{F}}$. We also studied the behavior of IDC as a function of the parameter $k$ (number of iterations). Perhaps the first question to ask is whether or not IDC *converges* to a steady state (e.g. where two consecutive iterations generate identical partitions). Unfortunately, a theoretical understanding of this convergence issue is left open in this paper. Nevertheless, in all our experiments IDC converged. Our empirical experience with IDC is that convergence takes between a few iterations and a few dozens of iterations. We discovered that in most cases IDC achieved its best performance after 8-10 iterations. Unless otherwise is specified, in the experiments reported below we used a fixed $k = 15$.

The first DC iteration is the most computationally intensive. After this iteration the co-occurrence (joint) distribution maintained by IDC reduces in size from $|X||F|$ to $|\tilde{X}||F|$ where $|X|$ is the sample size, and $|\tilde{X}|$ is the number of clusters (which corresponds to the number of classes) and typically $|\tilde{X}| \ll |X|$.

As noted earlier, the "hard" implementation of IB-clustering (and of DC) originally presented by [1] uses an agglomerative procedure as the basic clustering algorithm (see Section 2.1). The "soft" implementation of IB [3] applies a deterministic annealing clustering [8] as its underlying clustering procedure. As already discussed, the IB method can be viewed as a meta-clustering algorithm. Therefore, we can employ many vectorial clustering algorithms in the underlying IB procedure used by DC and IDC. We implemented IDC using several clustering algorithms including agglomerative clustering and deterministic annealing. Since both

**Input:**
  a sample $x_1, \ldots, x_n$,
  a distance measure $d(\cdot, \cdot)$,
  the desired number of clusters $k$
**Initialize:**
  $c := 0$ (current number of clusters)
**for** each point $x_i$ in the sample **do**
  **if** $c > 0$ **then**
    Let $c_j$ be the $d$-closest centroid to $x_i$
    Assign $c_j$ to be the center of gravity
    of the cluster $C_j$ together with $x_i$
  **end if**
  **if** $c = k$ **then**
    merge the two $d$-closest clusters
  **end if**
  create a new cluster containing $x_i$
**end for**

**Fig. 2.** Pseudo-code for Add-C

these algorithms are computationally intensive, we also implemented IDC using a fast algorithm called *Add-C* proposed by Guedalia et al. [11]. Add-C is an online greedy clustering algorithm with linear running time.

A pseudo-code of Add-C appears in Figure 2. In the code we left the choice of the distance measure open. In order to better approximate the original hard information bottleneck procedure we applied Add-C with the Jensen-Shannon divergence of Equation (1). For the exact details of this algorithm, the reader is referred to [11]. Our experience with information bottleneck clustering with both Add-C and the agglomerative algorithm indicates that the IDC implementation with Add-C yields inferior results to those produced by IDC with the agglomerative clustering routine. Specifically, when we apply the first iteration of IDC (= DC) using the agglomerative routine we usually get better results than a first iteration of IDC applied with the Add-C routine. Nevertheless, we gain a significant runtime speed-up since Add-C maintains a linear time complexity in the number of points. Moreover, as we later discuss, several iterations of IDC applied with Add-C are always superior to one iteration (DC) applied with agglomerative clustering (or Add-C). Due to its computational intensity we have not experimented much with a (multi-round) IDC applied with the agglomerative routine.

Following [1] we chose to evaluate the performance of IDC with respect to a *labeled* data set. Specifically, we count the number of classification errors made by IDC as obtained from labeled data.

In order to better understand the properties of IDC, we first examined it within a controlled setup of synthetically generated data points whose feature values were generated by $d$-dimensional Gaussian distributions (for $d$ features) of the form $N(\mu, \Sigma)$, where $\Sigma = \sigma^2 I$, with $\sigma$ constant. In order to simulate different sources, we assigned different $\mu$ values (from a given constant range) to each combination of source and feature. Specifically, for data simulating $m$ classes and $|F|$ features, $|F| \times m$ different distributions were selected.

Not surprisingly, when the range for selecting $\mu$ values was large in proportion to $\sigma$ (i.e. the sources were far apart), both DC and IDC performed well and gave results close to the Bayes error. Our next step was to introduce feature noise and see how IDC reacts. We used the following two noise models.

**Constant feature noise:** Distorts each of the features by adding random values sampled from $N(0, \sigma^2)$, where $\sigma$ is a constant "noise amplitude". Resulting negative values are rounded to zero.

**Proportional feature noise:** Distorts each entry with value $v$ by adding a random sample from $N(0, (\alpha \cdot v)^2)$, where $\alpha$ is the "noise amplitude". Resulting negative values are again rounded to zero.

In figure 3(a), we plot the average accuracy of 10 runs of IDC. The solid line plots (average) accuracies achieved in the last (15th) round of IDC vs. proportional feature noise amplitude ($\alpha$). The dashed line plots first rounds of IDC (equivalent to DC) vs. $\sigma$. Each error bar specifies one standard deviation. As can be seen, at low level noise amplitudes IDC attains perfect accuracy. When the noise amplitude increases, both IDC and DC deteriorate but the multiple

rounds of IDC can better resist the extra noise. A similar effect is observed when applying constant feature noise.

After observing the large accuracy gain between DC and IDC at a specific interval of noise amplitude within the proportional feature noise setup, we set the noise amplitude to a value in that interval and examined the behavior of the IDC run in more detail. Figure 3(b) shows a trace of the accuracy obtained at each of the 20 iterations of an IDC run over noisy data. This learning curve shows a quick improvement in accuracy during the first few rounds, and then reaches a plateau.

## 4 Empirical Results for Text Categorization

Following [1] we used the *20 Newsgroups (NG20)* [12] data set to evaluate IDC on real, labeled data. NG20 consists of 20,000 newsgroup articles from 20 different newsgroups, each group containing 1,000 documents. In this section we describe experiments with various subsets of NG20. We chose subsets with various degrees of difficulty. In the first set of experiments we used the following four newsgroups (denoted as NG4), two of which deal with sports subjects: 'rec.sport.baseball', 'rec.sport.hockey', 'alt.atheism' and 'sci.med'. In these experiments we tested some basic properties of IDC. In all the experiments reported in this section we performed the following preprocessing: We lowered the case of all letters, filtered out low frequency words which appeared only 3 times in the entire set and filtered out numerical and non-alphabetical characters. We also stripped off newsgroup headers which contain the class labels.

In Figure 3(c) we display accuracy vs. number of feature clusters ($N_{\tilde{F}}$). The accuracy deteriorates when $N_{\tilde{F}}$ is too small and we see a slight negative trend when it increases. A possible interpretation is that when the number of clusters is too small, there are not enough clusters to represent the inherent structure (in our case, word clusters with contextually similar meanings) of the data. It may also suggest that when the number of clusters is too large, either not enough features are grouped together to form 'semantic' units, or not enough noise is filtered by the clustering procedure. We performed an additional experiment which tested the performance using very large numbers of feature clusters. Indeed, these results indicate that after a plateau in the range of 10-20 there is a minor negative trend in the accuracy level. Thus, with respect to this data set, the IDC algorithm is not too sensitive to an overestimation of the number $N_{\tilde{F}}$ of feature clusters.

Other experiments over the NG4 data set confirmed the results of [1] that the JS-divergence dissimilarity measure of Equation (1) outperforms other measures, such as the variational distance ($L_1$ norm), the KL-divergence, the square-Euclidean distance and the 'cosine' distance. Details of all these experiments will be presented in the full version of the paper.

In the next set of experiments we tested IDC's performance on the same newsgroup subsets used in [1]. Table 1(a) compares the accuracy achieved by DC to the the last (15th) round of IDC with respect to all data sets described

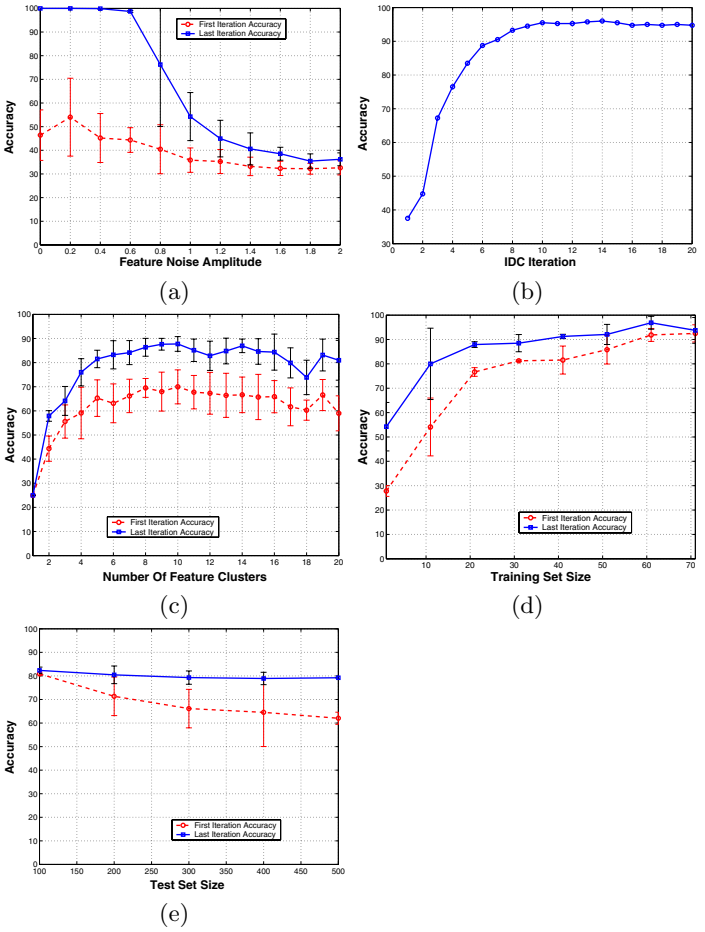


**Fig. 3.** (a) Average accuracy over 10 trials for different amplitudes of proportional feature noise. Data set: A synthetically generated sample of 200 500-dimensional elements in 4 classes (max $\mu = 50$, $\sigma = 50$). (b) A trace of a single IDC run. The $x$-axis is the number of IDC iterations and the $y$-axis is accuracy achieved in each iteration. Data set: Synthetically generated sample of 500, 400-dimensional elements in 5 classes (max $\mu = 50$, $\sigma = 45$); Noise: Proportional feature noise with $\alpha = 1.0$; (c) Average accuracy (10 trials) for different numbers of feature clusters. Data set: NG4. (d) Average accuracy of (10 trials of) transductive categorization of 5 newsgroups. Sample size: 80 documents per class, X-axis is training set size. Upper curve shows trans. IDC-15 and lower curve is trans. IDC-1. (e) Average accuracy of (10 trials of) transductive categorization of 5 newsgroups. Sample size: constant training set size of 50 documents from each class. The $x$-axis counts the number of unlabeled samples to be categorized. Upper curve is trans. IDC-15 and lower curve is trans. IDC-1. Each error bar (in all graphs) specifies one std.

in [1]. Results of DC were taken from [1] where DC is implemented using the agglomerative routine.

Table 1(b) displays a preliminary comparison of IDC with the results of a Naive Bayes (NB) classifier (reported in [6]) and a support vector machine (SVM) classifier. In each of the 5 experiments the supervised classifiers were trained using 25 documents per class and tested on 475 documents per class. The input for the unsupervised IDC was 500 unlabeled documents per class. As can be seen, IDC outperforms in this setting both the naive Bayes learner and the SVM. It should be emphasized that all the IDC results in this section were achieved by an unsupervised algorithm.

**Table 1.** (a) Accuracy of DC vs. IDC on most of the data sets described in [1]. DC results are taken from [1]; (b) Accuracy of Naive Bayes (NB) and SVM classifiers vs. IDC on some of the data sets described in [6]. The IDC-15 column shows final accuracy achieved at iteration 15 of IDC; the IDC-1 column shows first iteration accuracy. The NB results are taken from [6]. The SVM results were produced using the LibSVM package [13] with its default parameters. In all cases the SVM was trained and tested using the same training/test set sizes as described in [6] (25 documents per newsgroup for training and 475 for testing; the number of unlabeled documents fed to IDC was 500 per newsgroup). The number of newsgroups in each hyper-category is specified in parenthesis (e.g. COMP contains 5 newsgroups).

| **Newsgroup** | **DC** | **IDC-15** |
|---|---|---|
| $Binary1$ | 0.70 | 0.85 |
| $Binary2$ | 0.68 | 0.83 |
| $Binary3$ | 0.75 | 0.80 |
| $Multi5_1$ | 0.59 | 0.86 |
| $Multi5_2$ | 0.58 | 0.88 |
| $Multi5_3$ | 0.53 | 0.86 |
| $Multi10_1$ | 0.35 | 0.56 |
| $Multi10_2$ | 0.35 | 0.49 |
| $Multi10_3$ | 0.35 | 0.55 |
| **Average** | **0.54** | **0.74** |

(a)

| **Data Set** | **NB** | **SVM** | **IDC-15** | **IDC-1** |
|---|---|---|---|---|
| COMP (5) | 0.50 | 0.51 | 0.50 | 0.34 |
| SCIENCE (4) | 0.73 | 0.68 | 0.79 | 0.44 |
| POLITICS (3) | 0.67 | 0.76 | 0.78 | 0.42 |
| RELIGION (3) | 0.55 | 0.78 | 0.60 | 0.38 |
| SPORT (2) | 0.75 | 0.78 | 0.89 | 0.76 |
| **Average** | **0.64** | **0.70** | **0.71** | **0.47** |

(b)

## 5 Learning from Labeled and Unlabeled Examples

The impressive performance of IDC in unsupervised classification of data raises the question of whether the IDC procedure can serve for (semi) supervised learning or transductive learning. In this section, we present a natural extension of

IDC for transductive and semi-supervised learning that can utilize *both* labeled and unlabeled data. In transductive learning, the testing is done on the unlabeled examples in the training data, while in semi-supervised learning it is done on previously unseen data.

For motivating the transductive IDC, consider a data set $X$ that has emerged from a statistical mixture which includes several sources (classes). Let $C$ be a random variable indicating the class of a random point. During the first iteration of a standard IDC we cluster the features $F$ so as to preserve $I(F, X)$. Typically, $X$ contains predictive information about the classes $C$. In cases where $I(X, C)$ is sufficiently large, we expect that the feature clusters $\tilde{F}$ will preserve some information about $C$ as well. Having available some *labeled* data points, we may attempt to generate feature clusters $\tilde{F}$ which preserve more information about class labels.

This leads to the following straightforward idea. During the first IB-stage of the IDC first iteration, we cluster the features $F$ as distributions over *class labels* (given by the labeled data). This phase results in feature clusters $\tilde{F}$. Then we continue as usual; that is, in the second IB-phase of the first IDC iteration we cluster $X$, represented as distributions over $\tilde{F}$. Subsequent IDC iterations use all the unlabeled data.

In Figure 3(d) we show the accuracy obtained by DC and IDC in categorizing 5 newsgroups as a function of the training (labeled) set size. For instance, we see that when the algorithm has 10 documents available from each class it can categorize the entire unlabeled set, containing 90 unlabeled documents in each of the classes, with accuracy of about 80%. The benchmark accuracy of IDC with no labeled examples obtained about 73%.

In Figure 3(e) we see the accuracy obtained by DC and transductive IDC trained with a constant set of 50 labeled documents, on different unlabeled (test) sample sizes. The graph shows that the accuracy of DC significantly degrades, while IDC manages to sustain an almost constant high accuracy.

## 6 Concluding Remarks

Our contribution is threefold. First, we present a natural extension of the successful double clustering algorithm of [1]. Empirical evidence indicates that our new iterative DC algorithm has distinct advantages over DC, especially in noisy settings. Second, we applied the *unsupervised* IDC on text categorization problems which are typically dealt with by supervised learning algorithms. Our results indicate that it is possible to achieve performance competitive to supervised classifiers that were trained over small samples. Finally, we present a natural extension of IDC that allows for transductive learning. Our preliminary empirical evaluation of this scheme over text categorization is very promising.

A number of interesting questions are left for future research. First, it would be of interest to gain better theoretical understanding of several issues: the generalization properties of DC and IDC, the convergence of IDC to a steady state and precise conditions on attribute noise settings within which IDC is advanta-

geous. Second, it would be important to test the empirical performance of IDC with respect to different problem domains. Finally, we believe it would be of great interest to better understand and characterize the performance of transductive IDC in settings having both labeled and unlabeled data.

**Acknowledgements.** We thank Naftali Tishby and Noam Slonim for helpful discussions and for providing us with the detailed descriptions of the NG20 data sets used in [1]. We also thank Ron Meir for useful comments and discussions. This research was supported by the Israeli Ministry of Science. R. El-Yaniv is a Marcella S. Geltman Memorial academic lecturer.

## References

1. Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *ACM SIGIR 2000*, 2000.
2. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, New Jersey, 1988.
3. F.C. Pereira N. Tishby and W. Bialek. Information bottleneck method. In *37-th Allerton Conference on Communication and Computation*, 1999.
4. N. Slonim and N. Tishby. Agglomerative information bottleneck. In *NIPS99*, 1999.
5. L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proceedings of SIGIR'98*, 1998.
6. N. Slonim and N. Tishby. The power of word clustering for text classification. To appear in the European Colloquium on IR Research, ECIR, 2001.
7. T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.
8. K. Rose. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2238, 1998.
9. J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.
10. R. El-Yaniv, S. Fine, and N. Tishby. Agnostic classification of markovian sequences. In *NIPS97*, 1997.
11. I.D. Guedalia, M. London, and M. Werman. A method for on-line clustering of non-stationary data. *Neural Computation*, 11:521–540, 1999.
12. 20 newsgroup data set. http://www.ai.mit.edu/jrennie/20_newsgroups/.
13. Libsvm. http://www.csie.ntu.edu.tw/cjlin/libsvm.