Combining Active Learning and Dynamic Dimensionality Reduction

Mustafa Bilgic*

Abstract

To date, many active learning techniques have been developed for acquiring labels when training data is limited. However, an important aspect of the problem has often been neglected or just mentioned in passing: the curse of dimensionality. Yet, the curse of dimensionality poses even greater challenges in the case of limited data, which is precisely the setup for active learning. Reducing the dimensions is not a trivial task, however, as the correct number of dimensions depends on a number of factors including the training data size, the number of classes, the discriminative power of the features, and the underlying classification model. Moreover, active learning is typically applied in an iterative manner where the number of labels is smaller in the earlier iterations compared to the later ones. We propose an adaptive dimensionality reduction technique that determines the appropriate number of dimensions for each active learning iteration, utilizing the labeled and unlabeled data effectively to learn more accurate models. Extensive experiments comparing various approaches and parameter settings show that the proposed method improves performance drastically on three real-world text classification tasks.

Keywords: active learning; dimensionality reduction; regularization; classification.

1 Introduction

In many domains of interest, we often have access to ample amount of unlabeled data whereas the labeled data is either limited or non-existent. Such domains include text classification, speech recognition, person identification in video, and image classification on the web. We can ask domain experts to label some of the instances to build better predictive models, but annotating text, transcribing speech, and identifying persons take time and effort.

Active learning carefully chooses which instances to label in order to build powerful predictive models with minimal supervision [19]. To date, many query strategies (techniques that determine which instances' labels should be acquired) have been proposed, such as uncertainty sampling [12], query-by-committee [20], and empirical risk minimization [16]. These techniques and many others are typically applied iteratively, where a model is learned with the existing labels and new instances are chosen to be labeled to refine and improve that model.

An important aspect of the problem, however, has often been largely ignored. The query strategies are often used with common classifiers such as Naive Bayes, logistic regression, and SVM. These classifiers are not specifically designed for active learning; they often require ample labeled data. When the number of features is large and the training data is limited, it is difficult to get reliable estimates on the model parameters, which is referred as the curse of dimensionality [2]. This is especially problematic for active learning where limited supervision is not the exception but the norm. This problem is exacerbated by the fact that new instances are chosen to be labeled based on the current model that was trained with limited supervision. When the learned model is far from accurate (and it can potentially be worse than random when the parameters are estimated incorrectly), then the whole active learning process can be adversely affected.

We can build more accurate, stable, and reliable models at each iteration of the labeling process if we can intelligently reduce the dimensions, i.e., pick the correct features and the correct number of them. However, this is not a trivial task, as the correct number of features depends on a number of factors including the discriminative power of the features, the number of classes, the size of the training data, and the underlying classification model. Additionally, the number of dimensions has to be determined dynamically as new labels arrive at each iteration of the active learning process.

We propose a novel and dynamic dimensionality reduction (`DDR`) technique that determines which features and how many of them to include at each iteration. When applied, `DDR` improves the underlying model drastically, as presented in Figure 1. Without changing the querying strategy, which is random for this figure, the application of `DDR` improves the AUC from 0.66 to 0.81; an absolute increase of 0.15 (or a relative improvement of 23%). Given that various active querying strategies

*Computer Science Department, Illinois Institute of Technology, Chicago, IL. E-mail: mbilgic@iit.edu

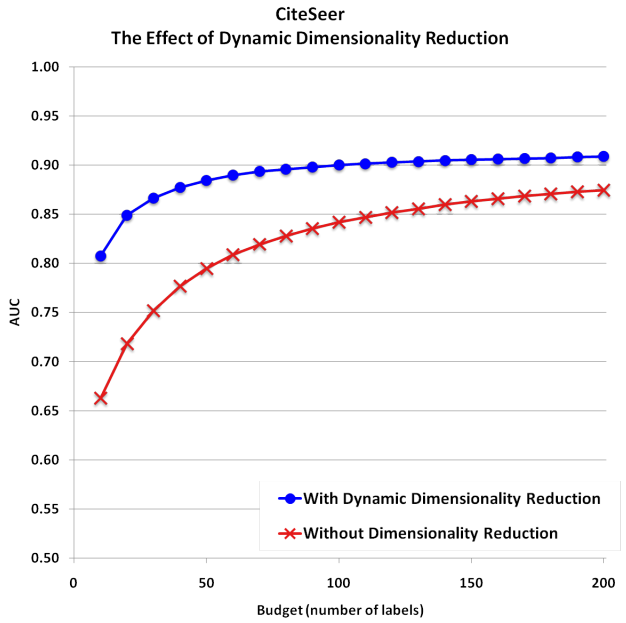


Figure 1: The effect of dynamic dimensionality reduction (DDR) on active learning. DDR improves performance significantly.

can improve only a few AUC points, this improvement obtained even without an active querying strategy is quite significant.

The rest of the paper is organized as follows. We provide background information on active learning and regularization for feature selection in Section 2. Section 3 introduces and provides details on DDR. We discuss several baselines in Section 4 and evaluate DDR and the baselines on three real-world datasets in Section 5. We then discuss related work in Section 6 and conclude in Section 7.

2 Background

In this section, we first discuss active learning and then discuss regularization as a means for feature selection.

2.1 Active Learning In many practical applications, it is easy to collect unlabeled data but labeling them costs time and money. Examples include annotating scientific papers with their topics, transcribing recorded speech, and recognizing hand written text. Active learning [19] carefully chooses which instances to label to build powerful predictive models with minimal supervision.

Active learning is typically performed iteratively where a model is learned with the existing labeled data and new instances are carefully chosen to refine and improve that model. The generic active learning procedure is given in Algorithm 1. In the pool-based setup [19], the active learner is given a "pool" of unlabeled examples ($\mathcal{U}$) and a budget (B) to spend on labeling instances. The active learner iteratively picks instances to label from the pool, an oracle provides the labels for them, they are added to the labeled set ($\mathcal{L}$), and the underlying model (M) is updated with the new information.

Algorithm 1: The generic active learning algorithm.

Input:
B – Budget, $\mathcal{U}$ – Pool of unlabeled instances, n – Batch size for labels, M – Base learner
Output:
$\mathcal{L}$ – Labeled examples

1 **while** $|\mathcal{L}| < B$
2 $\quad \mathcal{S} \leftarrow pickInstances(\mathcal{U}, \mathcal{L}, M, n)$
3 $\quad$ Label instances in $\mathcal{S}$
4 $\quad \mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}$
5 $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{S}$
6 $\quad$ Update model M by utilizing the new labels

The heart of the algorithm lies in step 2 where the choice of which instances to pick is made. Uncertainty sampling [12] picks instances on which the model M is most uncertain, where uncertainty can be measured using the predicted probability distribution and how close the instances are to the decision boundary. In query-by-committee [20], the model M consists of a committee of classifiers and the instances on which the committee members disagree the most are chosen to be labeled. In expected risk minimization [16], the instances which, if labeled, would reduce the expected empirical risk on a hold-out unlabeled set are chosen to be labeled.

In this paper, we do not propose a new query strategy; rather, we address the problem that the size of the labeled set $\mathcal{L}$ is so small, especially in the early iterations, that learning an accurate model at step 6 is rather challenging. The high dimensionality of the data only exacerbates this problem. This problem has important consequences for at least two reasons. First, in a real-world setting where the model is currently deployed and used in practice while still being constantly refined through active learning, the model correctness matters. Second, most active learning strategies choose their queries based on the current model; if the current model is far from accurate, the initial steps of active learning are used to make the model more reasonably accurate. When the model is trained through a dynamic

dimensionality reduction, however, it is possible to train reasonably accurate models with even severely limited training data. We modify the main active learning algorithm (Algorithm 1) so that the dimensionality of the instances can be reduced dynamically in a way that the model M can be learned more effectively at step 6. The reason that the dimensionality reduction needs to be dynamic is that the size of the labeled set $\mathcal{L}$ increases at each iteration; thus, the number of dimensions also needs to be adjusted accordingly.

2.2 Regularization for Feature Selection With the increased number of dimensions and the limited training data, the parameters for the underlying model cannot be estimated reliably, a problem referred as the curse of dimensionality [2]. In the active learning setup, the scarcity of the labels is not an exception but the norm, and thus the curse of dimensionality is a natural problem. A closely related problem is overfitting, where, instead of the general trends in the data and the true probability distribution, the details and the noise are captured by the model. Though high dimensionality is not the only cause, it is a big contributor of overfitting.

There are numerous ways to deal with the curse of dimensionality and overfitting. Feature selection techniques aim directly at reducing the dimensions by choosing a subset of the features. For example, filter based methods employ a criteria such as information gain to select the most informative features [5]. Wrapper methods search for the best subset of features for a given classifier and dataset [10]. An important question that needs to be answered in feature selection is how many features and which ones to choose. The correct number of dimensions depends on many factors including the discriminative powers of the features, the number of classes, the size of the training data, and the underlying learning algorithm.

A promising approach to taking these factors jointly into account is to use learners that directly incorporate feature selection into their optimization criteria. These learners balance how much the model fits the data ($\mathcal{X}$) and how complex the model is. Model complexity is measured as the model size, which is closely related to the number of features. Optimization for such models is typically formulated as follows:

(2.1)
$$\underset{w}{\operatorname{argmax}} \left(modelFit(\mathcal{X}; w) - C \times modelComplexity(w) \right)$$

where w is the parameter vector being optimized and C is a parameter that balances the fit and the complexity.

An alternative but equivalent formulation is to minimize the sum of the loss and model complexity:

(2.2)
$$\underset{w}{\operatorname{argmin}} \left(C \times loss(\mathcal{X}; w) + modelComplexity(w) \right)$$

where the loss can be the log-loss, 0/1 loss, etc.

Examples of such models include L_2-regularization where L_2-norm, $\|w\|_2^2$, L_1-regularization where L_1-norm, $\|w\|_1$, and decision trees where the size of the tree is used as the measure of model complexity. L_2-regularized logistic regression with logistic loss, for example, solves the following problem:

(2.3)
$$\underset{w}{\operatorname{argmin}} \left(C \times \sum_{i=1}^{N} \log(1 + e^{-y^{(i)} w^T x^{(i)}}) + \sum_{j=1}^{K} w_j^2 \right)$$

where, $x^{(i)}$ is the i^{th} instance and $y^{(i)}$ is its label, N is the number of instances in the data, and K is the number of features. L_1-regularization is defined similarly:

(2.4)
$$\underset{w}{\operatorname{argmin}} \left(C \times \sum_{i=1}^{N} \log(1 + e^{-y^{(i)} w^T x^{(i)}}) + \sum_{j=1}^{K} |w_j| \right)$$

Because L_2 penalizes large weights more, whereas L_1 penalizes all weights equally, L_1 tends to lead to sparser solutions, where a number of the weights is zero, essentially performing an implicit feature selection [21]. It has been shown that, in fact, L_1-regularization is more robust to irrelevant features than L_2 is and thus it is an effective feature selector [14]. This property makes L_1-regularization a promising candidate for dealing with the curse of the dimensionality. More importantly, it jointly optimizes feature selection and model learning, and thus is able to take the discriminative power of the features and the difficulty of the learning problem into account and it selects features as long as they improve the model fit without adding too much into the model complexity.

However, there are a few problems with this approach. The biggest problem is that L_1-regularization essentially performs a supervised feature selection: which features to include is decided based on the labeled data. For example, L_1-regularized logistic regression chooses features that optimize Equation (2.4), which is computed over the labeled instances. To achieve the optimal solution, it suffices to find a handful of features that can minimize the logistic loss. This would be OK and in fact desirable *only if* we had enough labels, which is precisely the problem for active learning.

The second problem, which is related to the first, is that the total loss, a sum over individual loses over the training instances, can be decreased only at the expense of model complexity and when the size of the training data is small, the model complexity can grow only so much. This forces L_1-regularization to choose a handful of features. The curse of dimensionality can play a detrimental role here: with a high number of features and only limited number of labels, it is quite possible for otherwise useless features to look useful just by chance. When the L_1-regularization chooses those seemingly useful features, the learned model cannot be expected to generalize well to unseen data.

Finally, and this is a problem for most methods, is that the complexity parameter C needs to be tuned to find the correct balance between the model fit and complexity. This is typically done through a separate validation data. However, because the labels are scarce in active learning, having a separate validation data is not very practical. Fortunately though, as we show later in the experiments, the complexity parameter does not play a huge role on the final results.

Next, we propose a technique that can utilize both L_2 and L_1-regularization simultaneously, that can make use of both labeled and unlabeled data, and that does not need a separate validation data to determine the appropriate number of features.

3 Dynamic Dimensionality Reduction (DDR)

We propose a dynamic dimensionality reduction technique that addresses these three issues and more. In a nutshell, we first pre-process the data (both labeled and unlabeled data together) using principal component analysis (PCA). Second, instead of using L_1-regularization on all the features, we use L_2-regularization on only a carefully selected small subset of the features. Finally, we determine the number of dimensions by analyzing the objective value of L_1-regularization. We next explain these in detail.

In pool-based active learning, we have access to a large set of unlabeled instances $\mathcal{U}$ in addition to the limited training data $\mathcal{L}$. If we perform dimensionality reduction using only $\mathcal{L}$, we are in essence ignoring the feature distribution in the unlabeled instances. If we can utilize the information from the unlabeled instances, we can hope to eliminate some of the noisy features if they are not well represented in the unlabeled data.

To this end, we first pre-process both the labeled and unlabeled data together using PCA. Pre-processing the data with PCA does not only help to deal with noisy features, but it also creates "super" features that are linear combinations of the original features. The "super" features that correspond to large eigenvalues are expected to be more informative than any of their single constituents. With the rare features downgraded, and the introduction of the super features, now L_1 has a better chance on selecting the correct features that will help generalize to unseen data. Moreover, a handful of features now can be enough to learn the correct model, because the super features can capture a big portion of the variance in the data. As we later show in the experiment section, L_1-regularized logistic regression on a dataset pre-processed through PCA performs significantly better than using L_1 on the original representation of the data.

To address the second problem (i.e., that L_1-regularization is forced to choose a handful of features and due to high dimensionality and limited labels, otherwise useless features can seem useful just by chance), we utilize the fact that the features with the largest eigenvalues capture the most variance in the data that has been preprocessed using PCA. Rather than leaving feature selection to L_1-regularization which has the chance element in it due to limited number of labels, we pick the features with the largest eigenvalues and use L_2-regularization on this subset. L_2-regularization, unlike L_1-regularization, does not over-invest in any feature; rather, it distributes the weights on all features, where the (seemingly) useful features receive high weight values but not extreme values, and (seemingly) useless features receive low weight values but not zero.

Finally, to determine the correct number of dimensions in the absence of validation data, we analyze the objective values achieved using Equations 2.3 and 2.4. We can potentially find the appropriate number of dimensions by starting with an empty set of features and adding features as long as the benefit of adding them (model fit) outweighs the cost (model complexity). That is, we can select the number of features that leads to the best objective value. However, adding more features will never cause an inferior objective value for a convex optimization problem that can be solved optimally; with more features, the optimization procedure has more freedom to achieve a better objective value.

To assess the benefit versus cost of adding a feature, we take a similar but slightly different approach. We start with an empty set of features and iteratively add k features to our set and train an L_2-regularized model (Equation (2.3)). We stop adding features when the increase in the model complexity measured using L_1-norm does not justify the decrease in the logistic loss. Learning the weights using L_2-regularization and using L_1-objective value to determine when to stop adding new features has the benefit of determining when L_2-regularization starts achieving a better objective value by simply distributing the weights across various

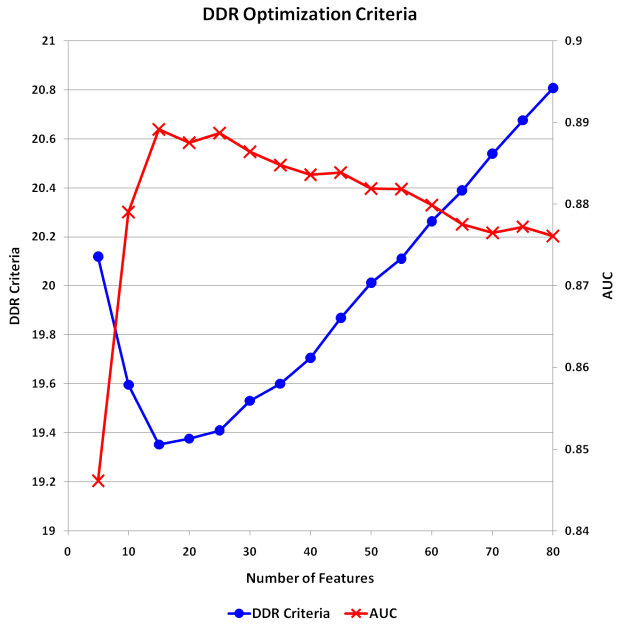


Figure 2: The L_1-objective value (Equation (2.4)) and AUC of an L_2-regularized logistic regression correlate highly.

features without improving the model fit as much.

To illustrate this phenomenon on a simple toy example, consider the following example. We currently have one single feature in our domain and its weight, learned using L_2-regularization, is w. Now, consider adding an identical feature into the domain. Simply setting the weights to $w/2$ for both features does not change the model fit but the L_2-norm is now $w^2/4 + w^2/4 = w^2/2$ compared to the initial value w^2; L_2-regularization was able to improve the L_2-norm simply by distributing the weights equally, without improving the model fit at all. The L_1-norm, on the other hand, does not change.

To illustrate this empirically, we show in Figure 2 how AUC and the L_1-objective value of an L_2-regularized logistic regression (we call this the DDR criteria) change as we add more features. As can be seen from this figure, both the DDR criteria and AUC improve first and then deteriorate again as we add more features. More importantly, they start deteriorating at the same number of features. As we show later in the experiment section, training an L_2-logistic regression but determining the appropriate number of features by inspecting the L_1-objective value provides drastic improvements over several baselines.

The Dynamic Dimensionality Reduction (DDR) algorithm works as follows. We first pre-process $\mathcal{L} \cup \mathcal{U}$ using PCA before the step 1 of the Algorithm 1 and sort the constructed features in decreasing order of how much variance they capture. Then, we search for the best number of dimensions using the DDR criteria (the L_1 objective value computed using the weights learned by an L_2-regularized model). To make the search more practical, rather than starting with an empty set and adding one feature at a time, we search in increments of k features. Additionally, rather than starting with an empty set of features at each iteration of active learning, we start from where we left off in the previous iteration; this works because the labeled set grows at each iteration and more labels can benefit from more dimensions. The querying strategy (step 2) and model updating (step 6) are done using the reduced dimensions.

4 Baselines

The most obvious baselines are the L_1-regularized and the L_2-regularized models on the original representation of the data (ORI-L_1, ORI-L_2) and on the PCA representation of the data (PCA-L_1, PCA-L_2). These baselines do not utilize any dimensionality reduction beyond what PCA and regularization offer. In addition to these baselines, we define two more baselines that utilize PCA and work on a subset of the features.

4.1 Expected Error Reduction (EE) This technique is inspired by the active learning method of Roy and McCallum [16]. They proposed estimating the utility of labeling another instance by estimating how much it is expected to reduce the error on unseen data. Similar to that method, we also define an expected error reduction (EE) technique that calculates the utility of adding a set of features as how much the added features are expected to reduce the error on the unseen data. First, we define the expected error:

(4.5)
$$EE(\mathcal{U}; M) = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} \left(1 - \max_{y_j} P(Y^i = y_j \mid x^i; M)\right)$$

Very much like DDR, EE also first pre-processes the data using PCA and sorts the features in the decreasing order of how much variance they capture. EE then searches for the subset of features that leads to the minimum expected error on the unlabeled data (Equation (4.5)). Similar to DDR, it searches for the best subset in increments of k and starts from where it left off in the previous active learning iteration.

Even though this technique seems promising, because we are directly optimizing the expected error on unseen data, it has a serious limitation for feature selection. The problem is that adding more features tends to make the underlying model more and more confident

in its predictions (incorrectly so); thus, this technique ends up adding most of the features. In the experiments section, we present results on both how well EE does as well as how many features it picks compared to the other techniques.

The next dimensionality reduction technique is a simple approach that will serve as a baseline as well as a sanity check.

4.2 Incremental Method (INCR) In this technique, we first pre-process the data using PCA and sort the features in the decreasing order of how much variance they capture. Then, starting with where it left off in the previous active learning iteration (0 in the beginning of the first iteration), it adds exactly and only k features. That is, the number of features at the i^{th} iteration of active learning is $i \times k$. We call this the incremental method (INCR). The reason why this is a reasonable method and a sanity check is that it adjusts the dimensionality based on the training data size and it adds the features in the order of their eigenvalues.

5 Experimental Evaluation

In this section, we first describe the datasets (the number of classes, features, etc.) we used in our experimental evaluation. Then, we describe the methodology we used to evaluate different techniques, followed by i) an analysis of the effect of the complexity parameter C in Equation (2.4), ii) comparison of L_1 and L_2-regularized logistic regressions on the original data (ORI-L_1, ORI-L_2), iii) how PCA effects the results of L_1 (PCA-L_1) and L_2 (PCA-L_2), iv) how well DDR performs compared to PCA-L_1, EE and INCR, and finally v) how the parameter k affects the results of DDR, EE, and INCR.

The dimensionality reduction techniques we discussed are largely orthogonal to the active learning techniques; they can be combined with many. In this paper, we present results on combining dimensionality reduction with two query strategies: random sampling, a common baseline for active learning, and uncertainty sampling [12], a simple yet popular query strategy that selects which instances to query based on how much the underlying model is uncertain on them. We next describe the datasets.

5.1 Datasets We experimented with three datasets that had relatively high dimensionality. The first two datasets, CiteSeer and Cora, are available at http://www.cs.umd.edu/projects/linqs/projects/lbc/. The CiteSeer dataset consists of 3312 scientific publications classified into one of six classes. Each publication in the dataset is described by a binary valued feature vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words. The Cora dataset consists of 2708 scientific publications classified into one of seven classes. Each publication in the dataset is represented by 1433 unique words. The third dataset is the Nova dataset that was used for the active learning workshop and challenge co-located with AISTATS 2010. The dataset, available at http://www.causality.inf.ethz.ch/al_data/NOVA.html. It consists of 19,466 documents extracted from 20-Newsgroup dataset. It is a binary classification problem where each document is classified as politics and religion or other. The documents are again represented as 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 16,969 unique words.

5.2 Methodology We present results on both random sampling and uncertainty sampling. We labeled 10 instances at each iteration (i.e., $n = 10$ in Algorithm 1) and searched for the best feature set in increments of 10 (i.e., $k = 10$). We also experimented with $k = 10, 20$, and 30, and we present those results as well. We present the learning curves, where the x-axis represents the number of labeled instances, and y-axis represents the performance. We used Area Under the ROC curve (AUC) as our performance measure.

We stopped labeling instances when one of the methods reached within 2% of the maximum achievable when trained using all the data. This criteria corresponded to having a budget, B, of 200 for Nova and CiteSeer, and 400 for the Cora dataset.

For determining which n instances to label in the case of uncertainty sampling, we followed [17] and used the uncertainty measures as weights and sampled the instances probabilistically in proportion to their weights.

We performed 10-fold cross validation, each time nine folds were used as the pool, $\mathcal{U}$, and the remaining fold was used as the test set. For each fold, we repeated the experiments 10 times. Thus, we report the averages over 100 AUC scores for each point on the learning curve. We used Weka's [9] implementation of PCA for our experiments. We performed PCA only on the pool, not looking at the test data at all. This required running PCA 10 times for each dataset. Because PCA was fairly slow for the Nova dataset (it had 17K features), we removed any words that appeared in fewer than 100 documents (i.e. 0.5% of all documents) before applying PCA. For L_1 and L_2-regularized logistic regression, we used the LibLinear package [7]. We used the default parameter settings for LibLinear.

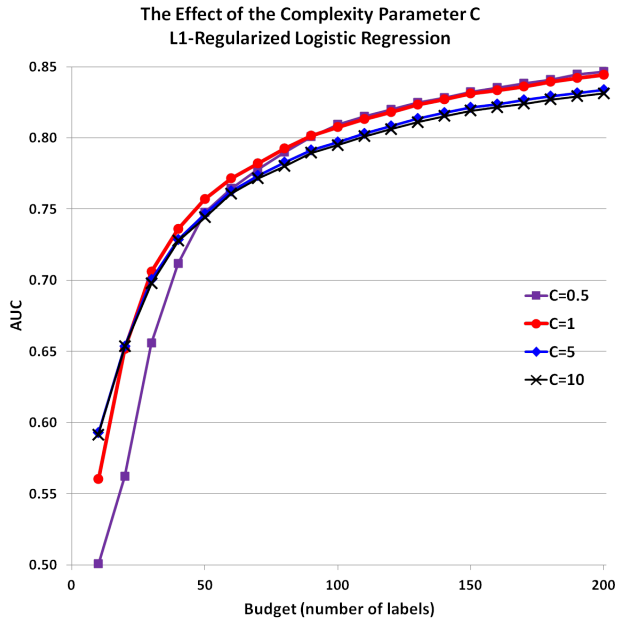


Figure 3: The effect of the complexity parameter C on the performance of the L_1-regularized logistic regression. The default parameter in the LibLinear package, $C = 1$, works fairly well.

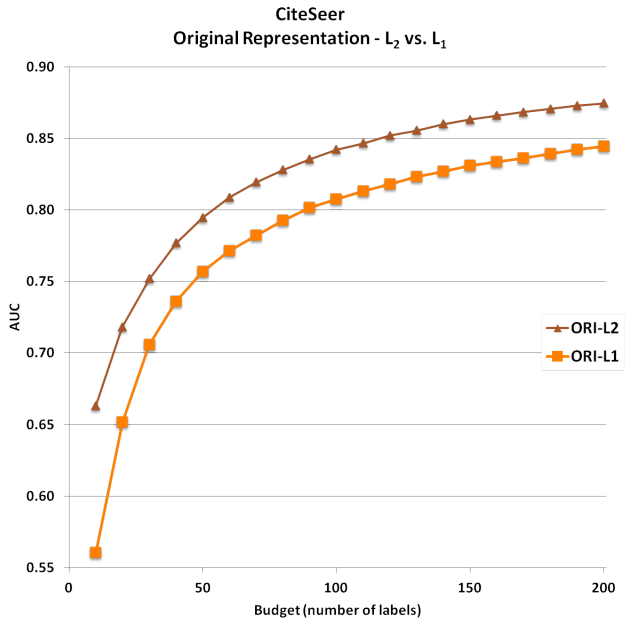


Figure 4: Comparison of ORI-L_1 and ORI-L_2 on the original representation of the data on the CiteSeer dataset. ORI-L_1 performs considerably worse than ORI-L_2.

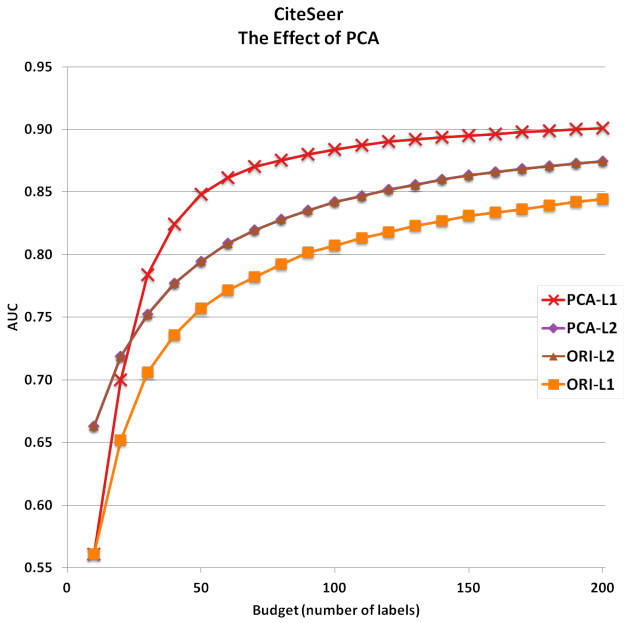


Figure 5: The effect of pre-processing both the unlabeled and labeled data through PCA. PCA-L_1 performed significantly better than other techniques. ORI-L_2 and PCA-L_2 has exactly the same performance as L_2-regularized logistic regression is rotation invariant.

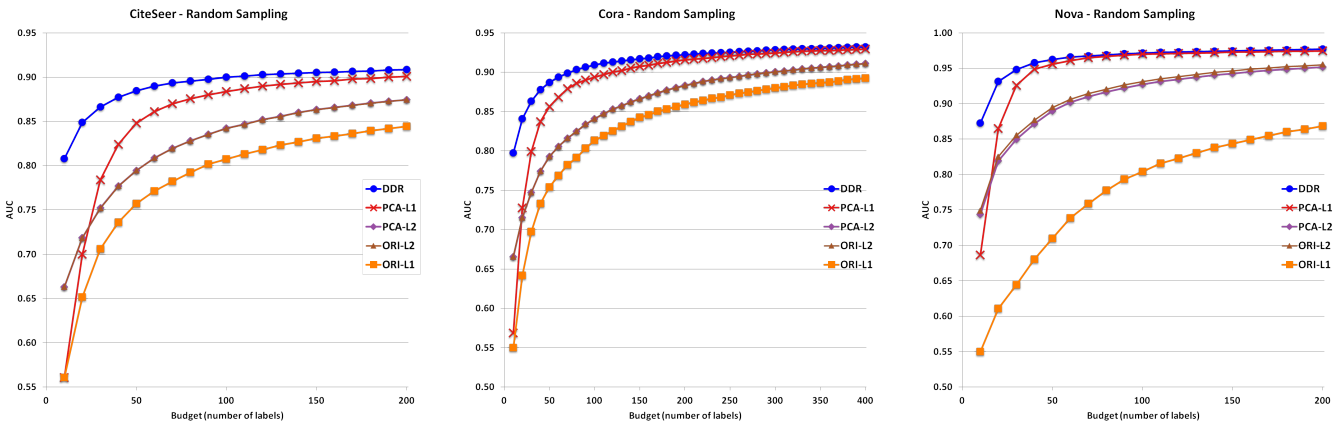
5.3 Results We first present the effect of the complexity parameter C on the performance of L_1-regularized logistic regression. The default parameter in the LibLinear implementation is $C = 1$. We experimented with various values and present the results in Figure 3.[1]

As these results suggest and as expected, the complexity parameter C affects the performance of L_1-regularized logistic regression. An obvious strategy is to use a high C value in the earlier iterations and decrease it as the number of labels increases. However, which value to use in the first iteration and how much to decrease it at each iteration is not easy to answer and it requires access to a separate validation data. Nonetheless, the default parameter $C = 1$ works fairly well. In the remainder of the experiments, we use this value.

We next present how well L_1-regularized logistic regression performs compared to the L_2 counterpart on the original representation of the data; we denote these as ORI-L_1 and ORI-L_2 respectively. The results for the CiteSeer dataset are presented in Figure 4.

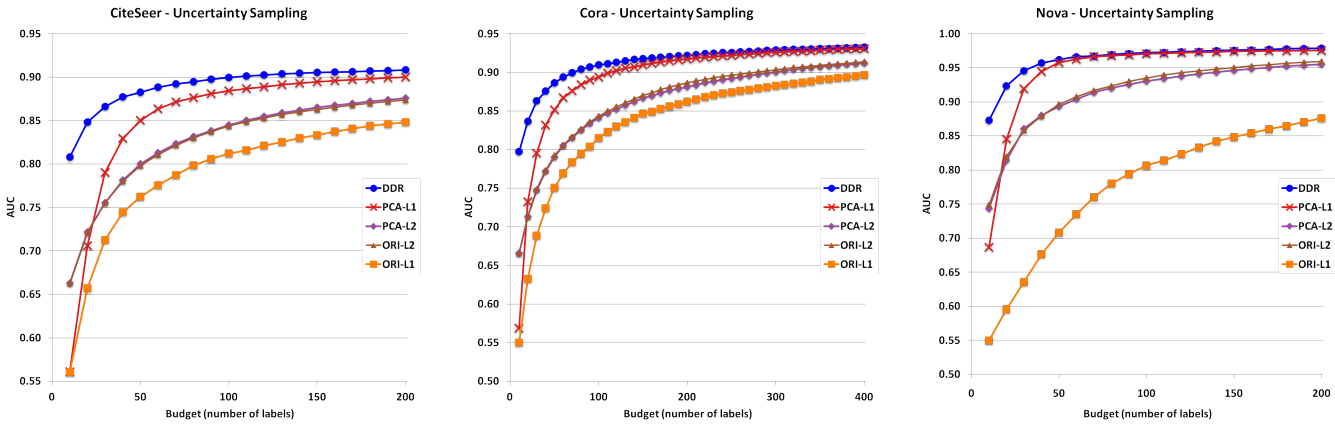
We see that, even though ORI-L_1 performs feature selection implicitly, it performs considerably worse than

[1] In this figure and the remaining figures, we focus on the region of interest, by zooming in and scaling the axes accordingly.



(a) (b) (c)

Figure 6: Comparing various dimensionality reduction techniques combined with random sampling. DDR has the best performance, followed by PCA-L_1.



(a) (b) (c)

Figure 7: Comparing various dimensionality reduction techniques combined with uncertainty sampling. DDR has the best performance, followed by PCA-L_1.

ORI-L_2. Possible reasons include that i) there are not enough labels for L_1 to determine the correct features to choose, ii) L_1 is forced to select only a handful of features to optimize Equation (2.4), and iii) small number of features (i.e., words) are not discriminative enough to learn the correct model.

We next show how applying PCA affects the results of L_1 and L_2-regularization. We apply PCA to the data but do not perform any feature selection in this case; that is, we include all the features that have been created by PCA. We add two more learning curves to our plot, corresponding to using L_1 and L_2 regularization on the PCA transformed data (PCA-L_1 and PCA-L_2 respectively). Figure 5 shows the results.

We see that pre-processing both the labeled and unlabeled data through PCA increased the performance of L_1-regularized logistic regression significantly. We believe that the unlabeled data provided essential information to the learning process through PCA. First, it reduced the possibility of choosing a noisy feature, as rare features are downgraded by PCA. Second, PCA constructed "super" features that are linear combinations of multiple features; such features are likely to be more powerful than any of its constituents. The performance of L_2-regularized logistic regression did not change at all because it is rotational invariant [14].

We finally discuss how DDR compares to the above methods. The results for both CiteSeer and the other two datasets, Cora and Nova, are shown in Figure 6. DDR outperforms all other methods including PCA-L_1. The

Table 1: Significance tests comparing DDR and PCA-L_1.

	Random			Uncertainty		
	W	T	L	W	T	L
Cora	40	0	0	37	3	0
CiteSeer	20	0	0	20	0	0
Nova	20	0	0	17	3	0

Table 2: Significance tests comparing DDR and INCR.

	Random			Uncertainty		
	W	T	L	W	T	L
Cora	39	1	0	39	1	0
CiteSeer	17	3	0	15	4	1
Nova	19	1	0	10	10	0

Table 3: Significance tests comparing DDR and EE.

	Random			Uncertainty		
	W	T	L	W	T	L
Cora	40	0	0	35	5	0
CiteSeer	20	0	0	20	0	0
Nova	20	0	0	13	7	0

performance differences are especially pronounced in the earlier iterations when the training data is severely limited. This result shows the power and promise of combining the benefits of L_1 and L_2-regularization.

In summary, pre-processing the data through PCA boosted the performance of L_1-regularized logistic regression significantly. Applying DDR provided the best results. With these improvements, random sampling has become a much more competitive baseline for active learning.

These dimensionality reduction and feature selection techniques are not limited to random sampling; they are largely orthogonal to the specific active learning technique used. As a proof of concept, we show the results of applying dynamic dimensionality reduction to uncertainty sampling in Figure 7. We observe similar trends in Figure 7, where DDR performs the best, and it is followed by PCA-L_1.

We performed statistical significance tests using t-test at each iteration of the learning, with a significance threshold of 0.05. We summarize the results using Win, Tie, Loss tables. If DDR is statistically significantly better, then it is counted as a win, if it is statistically significantly worse, it is counted as a loss, and otherwise, it is counted as a tie. We present the results comparing DDR and PCA-L_1 in Table 1. DDR significantly wins an overwhelming majority of the time and never loses to PCA-L_1.

Finally, perhaps less obvious but an important observation is that the set of labeled instances at a given iteration is same for all methods (ORI-L_2, ORI-L_1, PCA-L_2, PCA-L_1, and DDR) in random sampling. Thus, the AUC differences at any iteration are not due to differences in the training data. Rather, the differences occur based on how the same labeled instances are utilized by different techniques. We cannot, however, make the same claims for uncertainty sampling, where the next batch of labels are dependent on the underlying model.

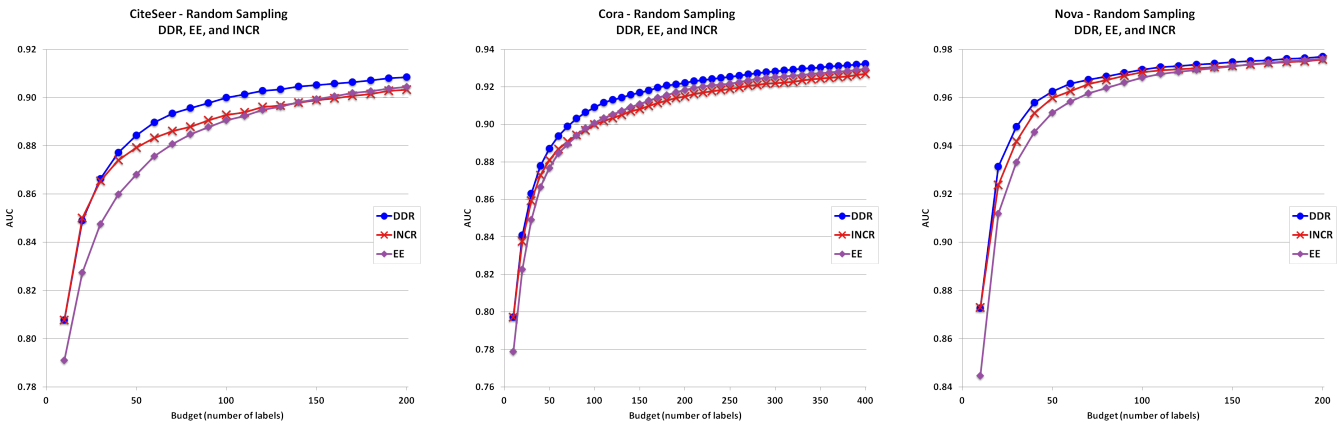
5.4 Comparisons of DDR, EE, and INCR In this section, we present how DDR performs compared to the two alternative techniques we described earlier, EE and INCR. We present results for both random sampling (Figure 8) and uncertainty sampling (Figure 9).

These figures show that DDR outperforms both EE and INCR. The differences are statistically significant for the most of the iterations as the significance tests presented in Tables 2 and 3 show. For $k = 10$, the simple INCR method outperforms the EE method for all cases, with the exception of uncertainty sampling on the Cora dataset.

5.4.1 Sensitivity to k Like DDR, the INCR and EE methods also utilize the parameter k; INCR adds k features at each iteration, whereas EE searches for the best number of features in increments of k. We experimented with three different values of k, $k = 10, 20$, and 30. We presented the results for $k = 10$ in the previous section and here we present the AUC results for $k = 30$. The purpose of these experiments is to shed some light on how many features DDR and EE use as well as how robust DDR and EE are to the choice of parameter k.

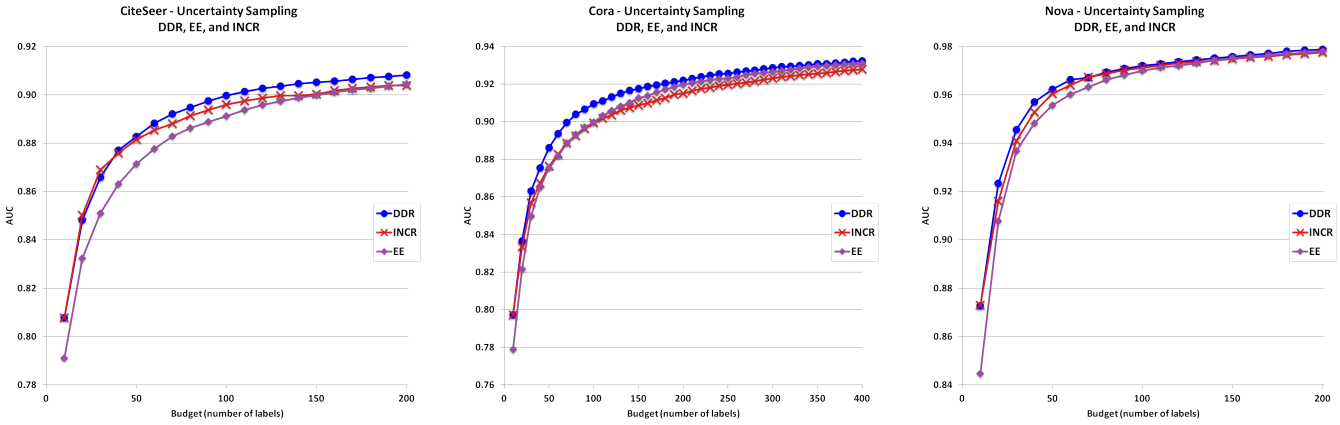
Figure 10 show that the gap between DDR and the other two methods increased with a higher value of k. In other words, comparing Figure 10 with Figure 8, we see that DDR's performance did not change much, whereas the performance of EE and INCR dropped significantly. Moreover, INCR's performance dropped more than EE's performance; when $k = 10$, INCR was a better method in general, and when $k = 30$, EE outperforms INCR on both CiteSeer and Cora. We do not show the results for uncertainty sampling due to space limitations but the trends are very similar.

We finally investigate how many features each method uses at each iteration. We do not show the



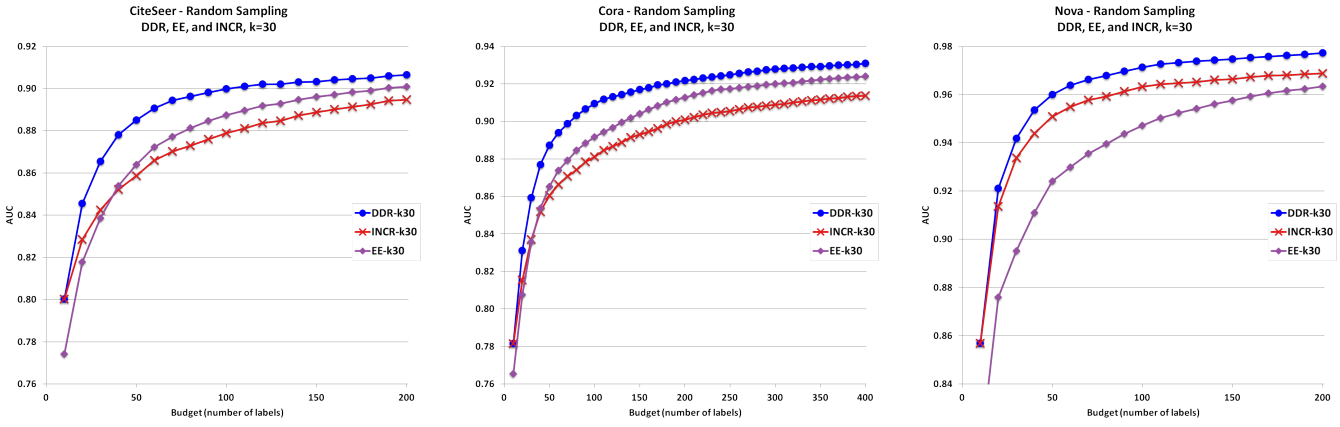
(a) (b) (c)

Figure 8: Comparing DDR, INCR, and EE on random sampling using $k = 10$.



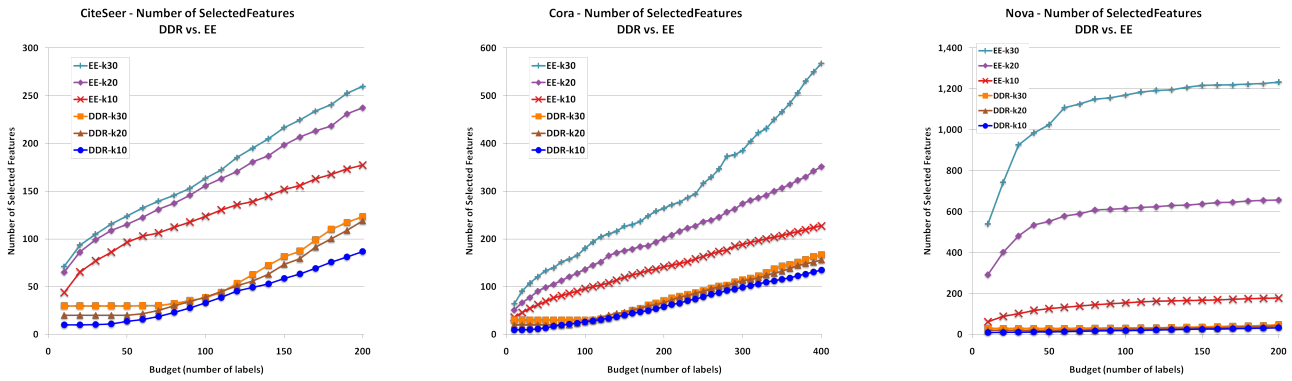
(a) (b) (c)

Figure 9: Comparing DDR, INCR, and EE on uncertainty sampling using $k = 10$.



(a) (b) (c)

Figure 10: Comparing DDR, INCR, and EE on random sampling using $k = 30$.



(a) (b) (c)

Figure 11: The number of features selected by DDR vs. EE. EE tends to over select features, whereas DDR is largely independent of k.

results for INCR to simplify the graphs and because it is easy to calculate how many features INCR uses (it uses $i * k$ features at the i^{th} iteration.) As the results in Figure 11 show, EE tends to pick a lot more features than DDR does. DDR on the other hand is largely independent of k.

5.5 Summary of Results

- DDR is the best performing dimensionality reduction technique compared to several baselines.
- EE is unreliable; it tends to over select features as more features tend to make the underlying model overly confident.
- PCA-L_1 is a fairly reasonable and simple alternative to DDR.
- L_1-regularization should be avoided when the training data is severely limited.

6 Related Work

Lewis and Gale [12] mention the importance of feature selection for robust probability estimation but they do not discuss how to determine the right number of features to use. Bilgic et al. [4] use PCA to reduce dimensionality as a pre-processing step; however, they do not optimize the number of features to use; rather, they use a fixed number of features throughout all iterations. We are not aware of any other work that discusses feature selection in the context of active learning.

Ng [14] discusses using L_1 and L_2 regularizations in the case of many irrelevant features and proves that sample complexity of L_1-regularized logistic regression grows only logarithmically in the number of irrelevant features whereas the sample complexity of L_2-regularized logistic regression grows linearly. L_1 regularization has also been used increasingly in learning the model structure for relational graphical models, for e.g. [11, 18].

Most feature selection techniques such as filtering based on information theoretic measures [5] or wrapper methods [10] can potentially be used for dimensionality reduction for active learning. However, these are supervised techniques and often there is not enough supervision in the active learning setup. Instead, we use an unsupervised technique, Principal Component Analysis, first to project the data into a new dimension and then select the top components as guided by L_1-regularization.

Active feature selection [3, 13, 22] is also a related area, but there the focus is on determining which feature values to acquire in cases where feature values are missing and acquiring their values has an associated cost (such as running costly laboratory experiments).

Finally, an alternative (and possibly complementary) approach to automated feature selection is to ask the user for feedback on features [1, 6, 8, 15]. In this line of work, users are asked about how discriminative features are, or asked to provide, possibly imprecise, constraints between the features and labels.

7 Conclusion

We presented an effective and dynamic dimensionality reduction technique, DDR, that combined the benefits of L_1, and L_2-regularization. The experimental validation showed that application of dynamic dimensionality reduction drastically improved the performance of active learning techniques. The techniques we described in this paper are largely orthogonal to the underlying active learning algorithms and can be combined with many.

Yet, they improved random sampling so much that it is now a fairly competitive baseline for active learning. We hope that our work raises awareness of the curse of the dimensionality problem in active learning and sheds some light on how to deal with it effectively.

References

[1] Josh Attenberg, Prem Melville, and Foster Provost. A unified approach to active dual supervision for labeling features and examples. In *European conference on Machine learning and knowledge discovery in databases*, pages 40–55, 2010.

[2] Richard Bellman and Stuart E Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 1962.

[3] Mustafa Bilgic and Lise Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research (JAIR)*, 41:69–95, 2011.

[4] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, 1991.

[6] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602, 2008.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[8] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 320–327, 2006.

[9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[10] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[11] Su I. Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of markov networks using L1-Regularization. In *Advances in Neural Information Processing Systems*, pages 817–824, 2007.

[12] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[13] Prem Melville, Maytal Saar-Tsechansky, Foster Provost, and Raymond Mooney. Active feature-value acquisition for classifier induction. In *IEEE International Conference on Data Mining*, pages 483–486, 2004.

[14] Andrew Y. Ng. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *International Conference on Machine Learning*, 2004.

[15] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

[16] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

[17] Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.

[18] Mark Schmidt, Alexandru Niculescu-Mizil, and Kevin Murphy. Learning graphical model structure using l1-regularization paths. In *AAAI Conference on Artificial Intelligence*, pages 1278–1283, 2007.

[19] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[20] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *ACM Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

[21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1996.

[22] Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.