# Unsupervised Feature Selection by Preserving Stochastic Neighbors

**Xiaokai Wei**
Department of Computer Science
University of Illinois at Chicago
xwei2@uic.edu

**Philip S. Yu**
Department of Computer Science
University of Illinois at Chicago
psyu@uic.edu

## Abstract

Feature selection is an important technique for alleviating the curse of dimensionality. Unsupervised feature selection is more challenging than its supervised counterpart due to the lack of labels. In this paper, we present an effective method, Stochastic Neighbor-preserving Feature Selection (SNFS), for selecting discriminative features in unsupervised setting. We employ the concept of stochastic neighbors and select the features that can best preserve such stochastic neighbors by minimizing the Kullback-Leibler (KL) Divergence between neighborhood distributions. The proposed approach measures feature utility jointly in a non-linear way and discriminative features can be selected due to its 'push-pull' property. We develop an efficient algorithm for optimizing the objective function based on projected quasi-Newton method. Moreover, few existing methods provide ways for determining the optimal number of selected features and this hampers their utility in practice. Our approach is equipped with a guideline for choosing the number of features, which provides nearly optimal performance in our experiments. Experimental results show that the proposed method outperforms state-of-the-art methods significantly on several real-world datasets.

## 1 Introduction

In the era of big data, datasets are often characterized by high dimensionality in many machine learning or data mining tasks. To alleviate the curse of dimensionality, feature selection [8] [12] [19] has become an important technique. By selecting a subset of high-quality features, feature selection can speed up the learning process and provide easier interpretation of the problem.

Depending on the availability of supervision information, feature selection methods can be categorized into two classes: supervised feature selection and unsupervised feature selection. Since class labels are usually expensive to obtain, our work focuses on unsupervised scenario. It is usually more difficult to evaluate the discriminativeness of features without guidance from class labels. Different heuristics (e.g., frequency based, variance based) have been proposed to perform unsupervised feature selection. Similarity-preserving approaches [8] [22] have gained much popularity among others. In such similarity preserving methods, a feature is considered to be good if it can preserve the local manifold structure well.

Recently, pseudo label based algorithms with $L_{2,1}$ norm [20] [10] have become increasingly popular. Since class labels are not available, such methods attempt to generate cluster labels (i.e., pseudo labels) or subspace representations through linear transformation/regression regularized by $L_{2,1}$ norm. They rank features by their usefulness on predicting pseudo label/constructing the subspace. One major drawback of such approach is that the cluster labels are usually far from accurate and such inaccurate pseudo labels can mislead feature selection.

The central issue in unsupervised feature selection is how to effectively uncover the discriminative information embedded in the data. Inspired by the popular visualization technique Stochastic Neighbor Embedding (SNE) [9], we employ the concept of stochastic neighbors for the purpose of unsupervised feature selection. We develop a novel unsupervised feature selection method, Stochastic Neighbor-preserving Feature Selection (SNFS), to select a set of high-quality features. Specifically, for each data point, other data

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

points are its neighbors with certain probability. The goal is to select a set of features that best preserve such stochastic probability. With this criterion, the derived gradient update formula is very simple, and it has a desirable *pull-push* property that the selected features can pull similar data points close and push dissimilar data points far apart. As a result, data points from different classes could be better separated with the set of selected features. The advantages of SNFS can be summarized as follows:

- The aim of unsupervised feature selection is usually to improve subsequent clustering tasks. Popular clustering methods such as KMeans and Spectral Clustering [11] are distance/similarity-based methods: KMeans needs to measure the similarity/distance to centroids when assigning data points and Spectral Clustering needs to build a similarity graph for clustering. State-of-the-art $L_{2,1}$ norm based approaches [20] [10] [14] select features based on how well they can linearly explain the variance of cluster labels (i.e., by their linear regression coefficients). By contrast, SNFS is not based on linear regression and is able to evaluate features jointly in a more similarity-friendly manner.
- The proposed criterion aims to keep similar data points closer than dissimilar data points. Such a criterion can select discriminative features to make the clusters more separable.
- For supervised feature selection, one can choose the number of selected features based on cross-validation performance. But it is very challenging to choose the optimal number of features in unsupervised setting. The inability of existing approaches [10] [14] to choose optimal feature size limits their practical utility. We provide a guideline for deciding feature sizes and experimental results indicate that this proposed guideline can usually achieve decent performance.

We develop an efficient optimization algorithm for the proposed method based on projected quasi-Newton method. Experimental results on six real-world datasets illustrate the superiority of SNFS.

## 2 Related Work

In this section, we review related work on feature selection.

### 2.1 Supervised Feature Selection

The goal of feature selection is to alleviate the curse of dimensionality, enabling machine learning models to achieve comparable, if not better, performance. Traditional feature selection methods generally fall into three categories: filter model [21] [13], wrapper model [5] and embedded model [3] [17]. In supervised feature selection, the criterion for feature quality is usually straightforward: high-quality features should be highly correlated with class labels. Different methods are proposed to capture the correlation between label and feature, such as Mutual Information, Fisher Score [4] and HSIC [16]. For example, Song et al. (2007) introduces Hilbert-Schmidt Independence Criterion (HSIC) as a measure of dependence between the features and the labels [16]. LASSO [17], as an embedded model, performs feature selection during regression/classification by using $L_1$ regularization.

### 2.2 Unsupervised Feature Selection

In the unsupervised setting, various heuristics are proposed to guide the feature selection process. One popular guiding principle is to preserve the local manifold structure or similarity [8] [21] [22]. But features useful for preserving similarity are not necessarily discriminative. Also, these earlier unsupervised feature selection algorithms tend to evaluate the importance of features individually [8] [21], which neglects correlation among features and may introduce redundancy in the selected features. Recent methods [20] [10] [15] [18] using sparsity-inducing norms overcome this issue by evaluating the features as a whole. For example, Unsupervised Discriminative Feature Selection (UDFS) [20] introduces pseudo label-based regression to better capture discriminative information. Sparsity-inducing $L_{2,1}$ norm is used to select the feature jointly. Robust Unsupervised Feature Selection (RUFS) [14] further employs robust $L_{2,1}$ loss on the regression objective to alleviate the effect of outlier instances. Robust Spectral Feature Selection (RSFS) [15] uses robust learning framework with local kernel regression for generating pseudo-labels.

Essentially, all the pseudo label based methods evaluate the utility of features based on how well in linear projection they can explain the variance of the cluster labels. As a result, they have similar drawbacks: first, they only evaluate features on their linear ability and overlook their non-linear usefulness. Second, the pseudo labels derived from clustering are usually not accurate enough. The noisy information contained in the pseudo labels can further mislead feature selection. Moreover, state-of-the-art pseudo-label approaches [14] [15] usually have $3 \sim 5$ free parameters (e.g., neighborhood size, number of latent dimensions and hyperparameters controlling regularization terms), which are difficult, if not impossible to tune without supervision.

## 3 Formulations

### 3.1 Notations

Suppose we have $n$ data samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ and the total number of features is $D$. So $\mathbf{x}_i \in \mathbb{R}^D$ and $x_{it}$ denotes the value of $t$-th ($t = 1, \ldots, D$) feature of $\mathbf{x}_i$. Our goal is to select $d$ ($d \ll D$) discriminative features. We use $\mathbf{w} \in \{0,1\}^D$ as the selection indicator vector: $w_t = 1$ indicates the $t$-th feature is selected and $w_t = 0$ otherwise.

### 3.2 Stochastic Neighbors-preserving Feature Selection

We assume each data sample has all the other data samples as stochastic neighbors with certain probability, rather than having a fixed set of neighbors. Let us denote the probability of $\mathbf{x}_i$ having $\mathbf{x}_j$ ($j \neq i$) as its neighbors as $p_{ij}$ and assume $p_{ij}$ depends on their similarity $S_{ij}$. The larger $S_{ij}$ is, the more likely $\mathbf{x}_j$ is $\mathbf{x}_i$'s neighbor. For convenience, we also define $p_{ii} = 0$ for $i = 1, \ldots, n$.

To make $\sum_{j=1}^{n} p_{ij} = 1$, we use the softmax function to define this probability.

$$p_{ij} = \frac{\exp(S_{ij})}{\sum_{k\neq i} \exp(S_{ik})} \tag{1}$$

In principle, $S_{ij}$ could be any affinity measure, such as cosine similarity and negative euclidean distance. We use inner product to measure the similarity of two data points in this paper and therefore $S_{ij} = \mathbf{x}_i^T \mathbf{x}_j$.

To add more flexibility to the model, we also include a scale (bandwidth) parameter $\sigma^2$ in the softmax function as follows. We will discuss how to set this parameter later in this paper.

$$p_{ij} = \frac{\exp(S_{ij}/\sigma^2)}{\sum_{k\neq i} \exp(S_{ik}/\sigma^2)} \tag{2}$$

After feature selection, we denote similarity calculated on the selected features as $s_{ij} = \mathbf{x}_i^T \mathrm{diag}(\mathbf{w}) \mathbf{x}_j$, where $\mathrm{diag}(\mathbf{w})$ is the diagonal matrix using $\mathbf{w}$ as diagonal elements. So, the probability of $\mathbf{x}_j$ being the neighbor of $\mathbf{x}_i$ after feature selection is $q_{ij}$.

$$q_{ij} = \frac{\exp\left(\frac{\mathbf{x}_i^T \mathrm{diag}(\mathbf{w}) \mathbf{x}_j}{\sigma^2}\right)}{\sum_{k\neq i} \exp\left(\frac{\mathbf{x}_i^T \mathrm{diag}(\mathbf{w}) \mathbf{x}_k}{\sigma^2}\right)} \tag{3}$$

Note that $q_{ij}$ (or $p_{ij}$) is not only influenced by $s_{ij}$ (or $S_{ij}$), but also affected by $s_{ik}$ (or $S_{ij}$, $k = 1, \ldots, j-1, j+1, \ldots, n$) via the normalization term. Therefore, $q_{ij}$ (or $p_{ij}$) is determined by the relative value of $s_{ij}$ (or $S_{ij}$) compared with other $s_{ik}$ (or $S_{ik}$).

To preserve the stochastic neighbors, we try to make two distributions $\mathbf{q}_i = [q_{i1}, \ldots, q_{in}]^T$ and $\mathbf{p}_i = [p_{i1}, \ldots, p_{in}]^T$ similar by minimizing their KL divergence for each $\mathbf{x}_i$.

$$KL(\mathbf{p}_i \| \mathbf{q}_i) = \sum_{j\neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4}$$

We propose the following feature selection criterion: selecting the set of features to minimize the sum of KL divergence between $\mathbf{p}_i$ and $\mathbf{q}_i$ on all the data points.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sum_{i=1}^{n} \sum_{j\neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ s.t. \quad & \sum_{t=1}^{D} w_t = d \\ & w_t \in \{0,1\}, \forall t = 1, \ldots, D \end{aligned} \tag{5}$$

The goal is that, for similar data points, we still want them to be similar after feature selection. For dissimilar data points, it is desirable to keep them dissimilar with selected features. So, by minimizing KL-divergence between $\mathbf{p}_i$ and $\mathbf{q}_i$ for $i = 1, \ldots, n$, we select the features that make similar data samples still closer than dissimilar samples.

### 3.3 Setting Scale Parameter

In this subsection, we discuss how to set the scale/bandwidth parameter $\sigma^2$. $\mathbf{p}_i$ is influenced by the value of $\sigma^2$: the higher $\sigma^2$ is, the higher entropy $\mathbf{p}_i$ has. For data sample $\mathbf{x}_i$, when $\sigma^2$ is relatively large, other data samples tend to have similar probability of being $\mathbf{x}_i$'s neighbors. In the extreme case, when $\sigma^2$ goes to infinity, all other data samples have equal probability of being $\mathbf{x}_i$'s neighbor. When $\sigma^2$ is small, the probability tends to be concentrated on a small number of most similar neighbors. We define the average perplexity as follows.

$$Perplexity(P) = 2^{\frac{1}{n}\sum_{i=1}^{n} H(\mathbf{p}_i)} \tag{6}$$

where $H(\mathbf{p}_i) = -\sum_{j\neq i} p_{ij} \log p_{ij}$ is the entropy of $\mathbf{p}_i$. The perplexity has a more intuitive interpretation than $\sigma^2$: it can be interpreted as a smooth measure of the effective number of neighbors. The perplexity is a monotonically increasing function of $\sigma^2$ and larger perplexity corresponds to larger $\sigma^2$. After we specify the value of perplexity, the value of $\sigma^2$ can be found by line search (e.g., binary search). So we do not need to directly set $\sigma^2$. Rather, we use perplexity as a proxy since it has more intuitive explanation. As we will show in the experimental results, SNFS can usually achieve good performance for a reasonably large range of perplexity (e.g., $5 \sim 50$).

## 4 Optimization

### 4.1 Gradient Derivation

The formulation in Eq (5) is a '0/1' integer programming problem, which is time-consuming to optimize. To make the optimization more efficient, we relax the '0/1' constraint on $w_t$ ($\forall t = 1, \ldots, D$) to real values in the range of $[0, 1]$. Also, we re-write the summation constraint $\sum_{t=1}^{D} w_t = d$ using Lagrangian multiplier.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sum_{i=1}^{n} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} + \lambda ||\mathbf{w}||_1 \\ s.t. \quad & 0 \le w_t \le 1, \forall t = 1, \ldots, D \end{aligned} \tag{7}$$

where $|| \cdot ||_1$ is the $L_1$ norm and $\lambda$ is the parameter to control the $L_1$ regularization. Note that in general $L_1$ norm is not differentiable due to the non-smoothness at value 0, but in our case, $|w_t| = w_t$ since $w_t$ ($\forall t = 1, \ldots, D$) is always non-negative.

Let us denote the objective in Eq (7) as $\mathcal{L}$. It takes several steps to calculate the gradient $\frac{\partial \mathcal{L}}{\partial w_t}$ (derivation details are in supplemental material), but the final result is simple.

$$\frac{\partial \mathcal{L}}{\partial w_t} = -\sum_{i=1}^{n} \sum_{j \neq i} (p_{ij} - q_{ij}) x_{it} x_{jt} / \sigma^2 + \lambda \tag{8}$$

If we use negative euclidean distance as the affinity measure (i.e., $s_{ij} = -(\mathbf{x}_i - \mathbf{x}_j)^T \text{diag}(\mathbf{w})(\mathbf{x}_i - \mathbf{x}_j)$), one can derive the following gradient formula in a similar manner:

$$\frac{\partial \mathcal{L}}{\partial w_t} = \sum_{i=1}^{n} \sum_{j \neq i} (p_{ij} - q_{ij})(x_{it} - x_{jt})^2 / \sigma^2 + \lambda \tag{9}$$

Such a gradient update formula in Eq (8) (or Eq (9)) has an intuitive *push-pull* interpretation: when $\mathbf{x}_j$ is more likely to be $\mathbf{x}_i$'s neighbor than desired (e.i., $p_{ij} < q_{ij}$), $w_t$ is updated in the direction of $x_{it} x_{jt} / \sigma^2$ (or $-(x_{it} - x_{jt})^2 / \sigma^2$) to push them away; when $\mathbf{x}_j$ is less likely to be $\mathbf{x}_i$'s neighbor than desired (e.i., $p_{ij} > q_{ij}$), $w_t$ is updated to pull them closer. If a feature has little contribution in preserving the distribution of stochastic neighbors, its weight tends to shrink to zero under the effect of $L_1$ regularization.

### 4.2 Projected Quasi-Newton Method

To make the optimization more efficient, we incorporate second order information by using projected quasi-Newton method [1]. At each iteration, we partition $w_t$ ($t = 1, \ldots, D$) into two groups: restricted variables $\mathcal{R}_w$ and free variables $\mathcal{F}_w$.

$$\begin{aligned} \mathcal{R}_w = & \{w_t | (w_t \le \epsilon \wedge \frac{\partial \mathcal{L}}{\partial w_t} > 0) \text{ or} \\ & (w_t \ge 1 - \epsilon \wedge \frac{\partial \mathcal{L}}{\partial w_t} < 0)\} \end{aligned} \tag{10}$$

$$\mathcal{F}_w = \{w_1, w_2, \ldots, w_D\} - \mathcal{R}_w \tag{11}$$

where $\epsilon$ is a small positive value. The restricted variables are those close to the lower or upper bound in their gradient direction. In Newton's Method, the scaling matrix $\bar{\mathbf{S}}^k$ for the free variables at iteration $k$ is the inverse Hessian matrix.

$$\bar{\mathbf{S}}^k = [\nabla^2 \mathcal{L}(\mathbf{w}^k)]^{-1}_{\mathcal{F}_w^k}, \tag{12}$$

For both free and restricted variables, the scaling matrix can be defined as follows.

$$\mathbf{S}^k = \begin{bmatrix} \bar{\mathbf{S}}^k & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \tag{13}$$

The scaling matrix $\mathbf{D}$ for restricted variables can be identity matrix. In each iteration, we find appropriate step size $\eta^k$ using backtracking line search to satisfy Armijo rule:

$$f(\mathbf{w}^k) - f(\mathbf{w}^k + \eta^k \mathbf{d}^k) \le c_1 \eta^k \mathbf{d}^k \tag{14}$$

where $c_1$ is a constant in the range of $0 \le c_1 \le 1$ and $\mathbf{d}$ is the descent direction ($\mathbf{d} = \mathbf{S}^k \nabla \mathcal{L}(\mathbf{w}^k)$ in our case). Note that computing the step size does not increase the computational complexity of the method, since computing the orthogonal projection after each backtracking step is trivial.

The final projected-Newton update formula is as follows.

$$\mathbf{w}^{k+1} \leftarrow \mathcal{P}(\mathbf{w}^k - \eta^k \mathbf{S}^k \nabla \mathcal{L}(\mathbf{w}^k)) \tag{15}$$

where the projection operator $\mathcal{P}(\cdot)$ projects the value $(\cdot)$ to $[0, 1]$.

$$[\mathcal{P}(\mathbf{w})]_t = \min(1, \max(0, w_t)), \forall t = 1, \ldots, D \tag{16}$$

For restricted variables $w_t^k \in \mathcal{R}_w^k$, we can directly set them to 0 or 1 if $\epsilon$ is sufficiently small.

$$[\mathcal{P}(\mathbf{w}^k - \eta^k \mathbf{S}^k \nabla \mathcal{L}(\mathbf{w}^k))]_t = \left\{ \begin{array}{ll} 0, & \text{if } w_t \le \epsilon \wedge \frac{\partial \mathcal{L}}{\partial w_t} > 0 \\ 1, & \text{if } w_t \ge 1 - \epsilon \wedge \frac{\partial \mathcal{L}}{\partial w_t} < 0 \end{array} \right. \tag{17}$$

So, we only need to compute the scaling matrix $\bar{\mathbf{S}}$ for free variables. This can save considerable computation time if the number of free variables is small (i.e., $|\mathcal{F}_w| \ll |\mathcal{R}_w|$), which is usually the case in feature selection scenario. It has been shown [1] [6] this projected-Newton method is convergent under mild conditions.

**Theorem 4.1** *For a loss function $\mathcal{L}$, assume that $\nabla\mathcal{L}$ is Lipschitz continuous and $\nabla^2\mathcal{L}$ has bounded eigenvalues. Then every limit point of $\mathbf{w}^k$ generated by Eq (15) is a stationary point of Eq (7).*

However, the Newton-step is often computation-intensive and requires $D^2$ storage. To save computation time and storage space, we approximate the Hessian with L-BFGS, as shown in Algorithm 1. L-BFGS only requires $O(mD)$ storage if the gradients in that last $m$ iterations are used. Though the convergence rate of the method has been shown for $\mathbf{S}^k$ derived from the Hessian, the convergence itself only requires a positive-definite gradient scaling $\mathbf{S}^k$ with bounded eigenvalues for all $k$ [1]. Thus, quasi-Newton approximations (e.g., L-BFGS) can also be employed to derive convergent methods. In our experiments, the optimization algorithm usually converges in less than 20 iterations.

**Algorithm 1** Projected L-BFGS Algorithm for SNFS

```
1: Initialize $\mathbf{w} \leftarrow [1, 1, \ldots, 1]$
2: while not converge do
3:     Identify restricted and free variables by Eq (10) and Eq (11).
4:     Set the restricted variables to the corresponding lower or upper bound (i.e., 0 or 1)
5:     Calculate the gradient using Eq (8) and $\bar{\mathbf{S}}$ for free variables using the gradient information of last $m$ iterations.
6:     Use backtracking line search to find the step size $\eta$ that satisfies Armijo condition Eq (14)
7:     Update $\mathbf{w}$ using formula Eq (15)
8: end while
9: Select features with $w_t$ $(t = 1, \ldots, D)$ greater than $1 - \alpha$
```

### 4.3 Determining the number of selected features

It is worth noting that $\mathbf{w}$ can be intuitively interpreted as features' importance scores in preserving stochastic neighbors. The relaxed $w_t$ $(t = 1, \ldots, D)$ has the maximum value of 1 and the minimum of 0. For unrelaxed $\mathbf{w}$, we simply retain the features with $w_t = 1$. Similarly, to select the high-quality features from relaxed version of $\mathbf{w}$, we can select the features with $w_t$ equal or close to 1. For example, we can keep the features with $w_t$ greater than $(1 - \alpha)$ for a small $\alpha$ (e.g., $\alpha = 0.05$ or $\alpha = 0.1$. We denote the number of features with scores larger than $(1 - \alpha)$ as $N_{1-\alpha}$. For example, $N_{0.9}$ is the number of features that have scores greater than 0.9. As we will show in the experimental results, such a strategy can usually achieve near-optimal performance.

Since $N_{0.9}$ is influenced by the regularization parameter $\lambda$ (i.e., larger $\lambda$ leads to smaller $N_{0.9}$), one can also do a line search for appropriate $\lambda$ (e.g., via binary search) if he wants to retain a specific number of features.

## 5 Discussion

Similarity-based approaches are a popular thread of unsupervised feature selection methods. In this section, we discuss how SNFS is different and superior to other similarity-based methods.

Laplacian Score [8] and SPEC [21] are based on the eigenvalues of similarity matrix. They assign a score to each feature and select the features with higher scores. Features are evaluated individually and redundancy can have negative impact on the performance of selected features. Besides, while the selected features will make similar data points still similar, they make little effort to make dissimilar data points far apart.

SPFS and MCFS [2] perform sparse linear regression towards the spectral decomposition of similarity matrix and choose the features with large coefficients. NDFS [10], RUFS [14] and RSFS [15] generate cluster labels and perform linear regression with $L_{2,1}$ norm. The drawback is that the inaccurate cluster labels can provide misleading information for feature selection. In all these regression-based methods, the selection criterion depends on how well the features can linearly explain the variance of cluster labels/subspace representation. This limits their effectiveness in clustering tasks, since most popular clustering algorithms are based on similarity/distance, such as KMeans and Spectral Clustering [11].

Moreover, a common shortcoming of all these methods is that they do not provide any guideline for choosing the number of selected features.

In contrast, SNFS evaluates features jointly and non-linearly. Rather than preserving the similarity itself, SNFS focuses on preserving the relative value of similarity in each neighborhood. Table 1 summarizes the difference between several popular similarity-based feature selection methods. We can see that SNFS has several desirable properties, which enable it to identify a set of more discriminative features.

## 6 Experiment

### 6.1 Baselines

We compare our approach to using all features and five unsupervised feature selection methods as baselines. LS and MCFS are manifold-preserving/similarity-

Table 1: Comparison of different similarity-based unsupervised feature selection methods

| Methods | LS[8], SPEC[21] | SPFS[23], MCFS[2] | NDFS[10], RUFS[14], RSFS[15] | SNFS |
|---|---|---|---|---|
| Evaluate features jointly? | × | ✓ | ✓ | ✓ |
| Evaluate features non-linearly? | ✓ | × | × | ✓ |
| Do not rely on clustering? | ✓ | ✓ | × | ✓ |
| Guideline for setting number of selected features? | × | × | × | ✓ |

Table 2: Statistics of datasets

| Statistics | BBC | BBC Sport | BlogCatalog | TDT | Guardian | Newsgroup |
|---|---|---|---|---|---|---|
| # of instances | 2225 | 737 | 500 | 1500 | 302 | 1575 |
| # of features | 9636 | 4612 | 4547 | 6458 | 3631 | 2849 |
| # of classes | 5 | 5 | 5 | 15 | 6 | 4 |

preserving approaches. UDFS, RUFS and RSFS are pseudo-label based methods which also consider the similarity information.

- All Features: It uses all the features for evaluation.
- Laplacian Score (LS): Laplacian score [8] selects the features which can best preserve the local manifold structure.
- MCFS: Multi-cluster Feature Selection [2] performs spectral analysis and sparse regression to select features.
- UDFS: Unsupervised Discriminative Feature Selection [20] is a psuedo-label based approach which performs $L_{2,1}$-norm regularized subspace learning.
- RUFS: Robust Unsupervised Feature Selection [14] generates psuedo labels by NMF (Non-negative Matrix Factorization) and local learning-based regularization [7].
- RSFS: Robust Spectral Feature Selection [15] selects features by robust spectral analysis framework and $L_{2,1}$-norm regularized regression.

## 6.2 Datasets

We use six publicly available datasets: BBC and BBC-Sport news dataset[1], Guardian news dataset[2], BlogCatalog[3] blog-posts dataset, Newsgroup [4] and TDT2[5]. More details of the datasets can be found in the supplemental material. The statistics of six datasets are summarized in Table 2.

[1]`http://mlg.ucd.ie/datasets/bbc.html`

[2]`http://mlg.ucd.ie/datasets/3sources.html`

[3]`http://dmml.asu.edu/users/xufei/datasets.html`

[4]`http://www.cs.umb.edu/~smimarog/textmining/datasets/`

[5]`http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

## 6.3 Experimental Setting

In this section, we evaluate the quality of selected features by their clustering performance. We use Accuracy and Normalized Mutual Information (NMI) to evaluate the result of clustering, following the typical setting of evaluation for unsupervised feature selection [20] [10]. These two metrics evaluate the cluster quality by matching and comparing the cluster labels with ground-truth labels (more detailed definition of the two metrics is presented in supplemental material). Higher values of Accuracy and NMI indicate better quality of clustering.

We set $k = 5$ for the kNN neighbor size in the baseline methods following previous convention [10]. For the number of pseudo classes in UDFS, RUFS and RSFS, we use the ground-truth number of classes. Besides, UDFS, RUFS and RSFS also require specifying the values of several other regularization parameters. In the original papers of UDFS, RUFS and RSFS, they use class labels to find the best parameters by grid search. However, this violates the assumption of no supervision and could be unfair to approaches with less or no free parameters. Nonetheless, we perform grid search in the range of $\{0.1, 1, 10\}$ for the regularization parameters in UDFS, RUFS and RSFS. Besides the best performance, we also report the median performance for them, which is a more realistic reflection of these methods' practical power. For SNFS proposed in this paper, we fix $perplexity = 15$ and $\lambda = 10^{-3} \times n$ on all datasets and we will discuss the sensitivity of these two parameters in the following subsection. [6].

[6]For the projected quasi-Newton method in the opti-

Following the convention in previous work [2] [20], we use KMeans [7] for clustering evaluation. Since Kmeans is affected by the initial seeds, we repeat the experiment for 20 times and report the average performance. We vary the number of features in the range of {100, 200, 400, 600}. For SNFS, we report the clustering performance using the features with scores greater than 0.9.

### 6.4 Clustering Results

The clustering accuracy on six datasets is shown in Table 3 (NMI results can be found in supplemental material). The experimental results show that feature selection is a very effective technique for enhancing clustering. With much less features, SNFS ($N_{0.9}$) can obtain better accuracy and NMI than using all the features. For instance, compared with using all 4547 features, SNFS with only 230 features improves the clustering accuracy by 36.4% on BlogCatalog dataset. Besides the improved accuracy and NMI, using selected features rather than all features can also lead to better interpretability.

We can observe that for SNFS, using $N_{0.9}$ features usually performs the best (or nearly the best) among different number of features. The top $N_{0.9}$ features all have scores equal to 1 or very close to 1. So it is not wise to use only a subset of them. Also, using more than $N_{0.9}$ features may lead to redundancy since less important features are included.

When comparing SNFS with the baseline methods, we observe that SNFS has very competitive performance. The accuracy and NMI of SNFS ($N_{0.9}$) is better than or comparable to the best performance of two strong baselines (RUFS and RSFS) and outperforms their median performance significantly. Since in practice one cannot know the optimal parameters of RUFS and RSFS in unsupervised scenario, the median performance is more representative of their practical utility. Also, all the baseline methods do not provide guidelines for determining the number of selected features. For example, RUFS achieves its top median performance with 400, 200 and 600 features on BBCSport, BlogCatalog and Guardian datasets, respectively. This makes these baseline methods less favorable in practice.

In summary, although these baseline methods also attempt to exploit similarity information in certain ways, they do not perform as well as SNFS. The experimental results illustrate that SNFS is a more effective method for selecting discriminative features.

### 6.5 Sensitivity Analysis

For unsupervised feature selection, it is important that the feature selection algorithm is not very sensitive to its parameters. SNFS has two free parameters: the perplexity and the regularization parameter $\lambda$ for $L_1$ norm. In this section, we investigate how the performance of SNFS ($N_{0.9}$) varies w.r.t different parameter values [8].

Figure 1 shows the clustering accuracy of SNFS ($N_{0.9}$) over different values of perplexity. We can observe that SNFS has consistently good performance with different perplexity values ranging from $5 \sim 50$. In most cases, the performance is better than using all the features.

For $\lambda$, we vary its value in the range of $[0.0001, 0.0002, 0.0004, 0.0006, 0.0008, 0.001, 0.002, 0.004] \times n$. The clustering performance of SNFS ($N_{0.9}$) over different $\lambda$ is shown in Figure 2. On most datasets, SNFS has decent performance and can outperform using all features, if $\lambda$ is not too small ($< 2 \times 10^{-4} \times n$) or too large ($> 2 \times 10^{-3} \times n$).

## 7 Conclusion

In this paper, we propose a new method, SNFS, for unsupervised feature selection by preserving stochastic neighbors. For each data point, other data points can be its potential neighbors with certain probability. We select the features that can approximate the original distribution by minimizing the KL-divergence. This criterion can select discriminative features that makes similar data points close and push dissimilar data points far apart. The objective function has less parameters than the state-of-the-art pseudo-label methods and it has a simple gradient update formula. We develop an efficient optimization algorithm for SNFS based on projected L-BFGS. Empirical results show that the proposed method outperforms state-of-the-art approaches on several real-world datasets.

**Acknowledgements**

This work is supported in part by NSF through grants III-1526499, CNS-1115234, and OISE-1129076, and Google Research Award.

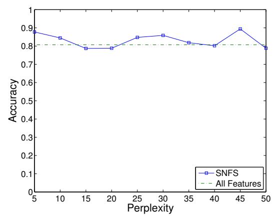
mization of SNFS, we use the implementation at `http://www.cs.ubc.ca/~schmidtm/Software/minConf.html`

[7]We use the code at `http://www.cad.zju.edu.cn/home/dengcai/Data/Clustering.html`

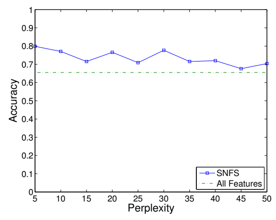
[8]Due to space constraint, we only present the results on the first four datasets here and put the full results in the supplemental material

Table 3: Clustering accuracy on six datasets. For UDFS, RUFS, RSFS, median/best performance is reported. SNFS($N_{0.9}$) denotes the performance of SNFS with top $N_{0.9}$ features.

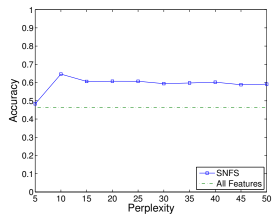
| Method | **BBC** | | | | **BBC Sport** | | | |
|---|---|---|---|---|---|---|---|---|
| # features | 100 | 200 | 400 | 600 | 100 | 200 | 400 | 600 |
| All Features | 0.8071 | | | | 0.6551 | | | |
| LS | 0.2360 | 0.2655 | 0.4322 | 0.4718 | 0.4185 | 0.4561 | 0.5110 | 0.6751 |
| MCFS | 0.6223 | 0.7489 | 0.7793 | 0.8217 | 0.6082 | 0.7075 | 0.7027 | 0.7248 |
| UDFS | 0.4246/0.4811 | 0.6174/0.6681 | 0.7766/0.7805 | 0.7599/0.7763 | 0.4661/0.4875 | 0.4770/0.5601 | 0.5390/0.605 | 0.5770/0.6139 |
| RUFS | 0.4744/0.7548 | 0.6708/0.8584 | 0.7976/0.8991 | 0.8263/0.8836 | 0.6018/0.7487 | 0.6598/0.7683 | 0.7009/0.7518 | 0.6812/0.7187 |
| RSFS | 0.5677/0.7660 | 0.7523/0.8118 | 0.8068/0.8863 | 0.8326/0.8693 | 0.6158/0.6658 | 0.6546/0.714 | 0.6648/0.6961 | 0.6494/0.7030 |
| SNFS | 0.6040 | 0.7729 | 0.8165 | 0.8102 | 0.5847 | 0.6881 | 0.7455 | 0.6964 |
| SNFS($N_{0.9}$) | 0.8414(550) | | | | 0.7195(440) | | | |
| Method | **BlogCatalog** | | | | **Guardian** | | | |
| # features | 100 | 200 | 400 | 600 | 100 | 200 | 400 | 600 |
| All Features | 0.4627 | | | | 0.5477 | | | |
| LS | 0.2998 | 0.4084 | 0.4203 | 0.4003 | 0.3364 | 0.4083 | 0.6573 | 0.6237 |
| MCFS | 0.3704 | 0.4428 | 0.4143 | 0.4161 | 0.5093 | 0.5053 | 0.5361 | 0.5348 |
| UDFS | 0.3901/0.3917 | 0.4069/0.4691 | 0.4383/0.4749 | 0.4876/0.5321 | 0.3682/0.4998 | 0.4525/0.5144 | 0.5127/0.5394 | 0.5247/0.5411 |
| RUFS | 0.4877/0.5307 | 0.5508/0.5756 | 0.5476/0.5889 | 0.5375/0.5648 | 0.4369/0.5608 | 0.5329/0.5659 | 0.5490/0.5661 | 0.5563/0.5791 |
| RSFS | 0.3847/0.4969 | 0.4371/0.5346 | 0.4709/0.5464 | 0.5031/0.5412 | 0.5320/0.5553 | 0.5296/0.5816 | 0.5550/0.5907 | 0.5541/0.5921 |
| SNFS | 0.5842 | 0.6350 | 0.5924 | 0.5821 | 0.5288 | 0.6063 | 0.6290 | 0.6071 |
| SNFS($N_{0.9}$) | 0.6313(230) | | | | 0.6270(440) | | | |
| Method | **Newsgroup** | | | | **TDT** | | | |
| # features | 100 | 200 | 400 | 600 | 100 | 200 | 400 | 600 |
| All Features | 0.7184 | | | | 0.7711 | | | |
| LS | 0.2808 | 0.3863 | 0.6420 | 0.7063 | 0.6548 | 0.7472 | 0.7870 | 0.7816 |
| MCFS | 0.3374 | 0.4368 | 0.4883 | 0.5059 | 0.6128 | 0.6656 | 0.7250 | 0.7367 |
| UDFS | 0.3516/0.4145 | 0.3954/0.4173 | 0.4403/0.4653 | 0.4604/0.6335 | 0.4863/0.4979 | 0.6102/0.6110 | 0.7231/0.7247 | 0.7499/0.7520 |
| RUFS | 0.4687/0.6073 | 0.4595/0.6435 | 0.4915/0.6476 | 0.5295/0.6477 | 0.4381/0.6865 | 0.5423/0.8112 | 0.6731/0.7967 | 0.7614/0.8198 |
| RSFS | 0.3969/0.6045 | 0.4776/0.6516 | 0.6069/0.6765 | 0.6225/0.6923 | 0.6589/0.7806 | 0.7730/0.8153 | 0.7695/0.8173 | 0.7854/0.8261 |
| SNFS | 0.4518 | 0.5075 | 0.6833 | 0.7039 | 0.7502 | 0.7902 | 0.7835 | 0.7890 |
| SNFS($N_{0.9}$) | 0.8007(495) | | | | 0.8161(163) | | | |



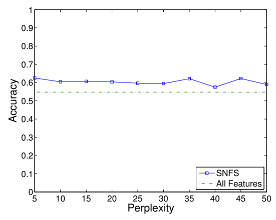
(a) BBC



(b) BBCSport

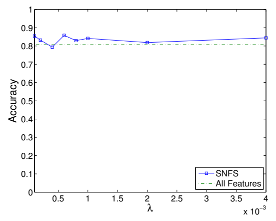


(c) BlogCatalog

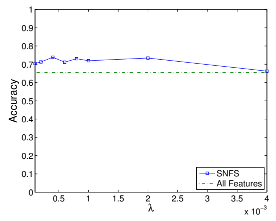


(d) Guardian

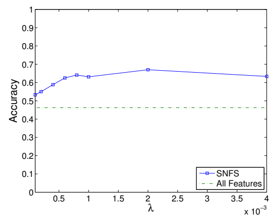
Figure 1: Clustering accuracy with different perplexity values



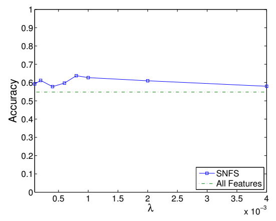
(a) BBC



(b) BBCSport



(c) BlogCatalog



(d) Guardian

Figure 2: Clustering accuracy with different $\lambda$

## References

[1] D. P. Bertsekas. Projected newton methods for optimization problems with simple constraints. In *SIAM Jounal on Control and Optimization*, 1982.

[2] D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. In *KDD*, pages 333–342, 2010.

[3] G. C. Cawley, N. L. C. Talbot, and M. Girolami. Sparse multinomial logistic regression via bayesian l1 regularisation. In *NIPS*, pages 209–216, 2006.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, 2 edition, 2001.

[5] J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

[6] E. M. Gafni and D. P. Bertsekas. Two-metric projection methods for constrained optimization. In *SIAM Journal on Control and Optimization*, 1984.

[7] Q. Gu and J. Zhou. Local learning regularized nonnegative matrix factorization. In *IJCAI*, pages 1046–1051, 2009.

[8] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *NIPS*, 2005.

[9] G. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:833–840, 2003.

[10] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*, 2012.

[11] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[12] F. Nie, H. Huang, X. Cai, and C. H. Q. Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *NIPS*, pages 1813–1821, 2010.

[13] H. Peng, F. Long, and C. H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

[14] M. Qian and C. Zhai. Robust unsupervised feature selection. In *IJCAI*, 2013.

[15] L. Shi, L. Du, and Y.-D. Shen. Robust spectral learning for unsupervised feature selection. In *ICDM*, pages 977–982, 2014.

[16] L. Song, A. J. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *ICML*, volume 227, pages 823–830. ACM, 2007.

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

[18] X. Wei, B. Cao, and P. S. Yu. Unsupervised feature selection on networks: A generative view. In *AAAI*, 2016.

[19] X. Wei, S. Xie, and P. S. Yu. Efficient partial order preserving unsupervised feature selection on networks. In *SDM*, pages 82–90, 2015.

[20] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou. l2, 1-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI*, pages 1589–1594, 2011.

[21] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML*, volume 227, pages 1151–1157, 2007.

[22] Z. Zhao, L. Wang, and H. Liu. Efficient spectral feature selection with minimum redundancy. In *AAAI*, 2010.

[23] Z. Zhao, L. Wang, H. Liu, and J. Ye. On similarity preserving feature selection. *IEEE Trans. Knowl. Data Eng.*, 25(3):619–632, 2013.