Unsupervised Feature Learning for Object Classification

Laxman Dhulipala, Harry Gifford, Wangzi He
Department of Computer Science
Carnegie Mellon University University
Pittsburgh, PA 15213
{ldhulipa, hgifford, wangzih}@andrew.cmu.edu

Abstract

A major theme in modern object recognition research has been the use of features to boost classification techniques. Stemming from the seminal work of Viola and Jones, features and feature learning techniques have been used year after year to improve existing object recognition algorithms, and increase classification rates. In this work, we implement and evaluate an object classification framework that first learns a dictionary of features, and subsequently uses this feature basis to represent and classify images into categories. We evaluate our feature learning framework by analyzing its performance on the CIFAR-10 dataset, and also consider several optimizations and heuristics to help boost performance.

1 Introduction

Feature learning as a task has its roots firmly embedded in a long history of research in Neuroscience. Stemming from Hubel and Wiesel's discovery in the late 1950's of complex and simple cells, researchers have been captivated by using techniques and ideas from neuroscience as building blocks for robust object classification in a machine[7]. To this end, ideas from neuroscience such as the Perceptron algorithm, convolution neural networks and most recently, deep belief networks have all had tremendous success in object recognition and machine learning.

A fundamental step for many feature learning algorithms is to learn hierarchies of features from images. Learnt features from a dataset afford the algorithm designer to then represent images in the dataset as linear combinations of the learnt features. In some sense, this is the most natural reason to learn features, as they allow one to transform the space of images into the space of images as represented by their features. This transformed space typically produces better results when running typical supervised learning tasks, such as classification. Often, learned features represent the space well enough that even simple classification techniques such as k-nearest neighbors or linear SVMs will perform well.

Feature heirarchies can be learnt in a variety of ways, from semi-supervised learning algorithms to fully unsupervised learning algorithms. Our approach to feature learning relies on an elegant and simple algorithm that was recently published by Adam Coates and Andrew Ng [4]. They propose the use of k-means clustering for feature learning as it is a simple and easily paralellizable algorithm that can be deployed at scale. In an earlier work, they show how k-means in practice is highly competitive against the (then) state-of-the-art feature learning algorithms, often beating them. In particular, they show a surprising result that k-means based feature learning with a single layer neural network achieves state-of-the-art error rate on the CIFAR-10 dataset.

In this paper we evaluate the performance of different (heirarchical) feature learning algorithms based on k-means. Our contribution is that we show that sparse k-means can be used in higher

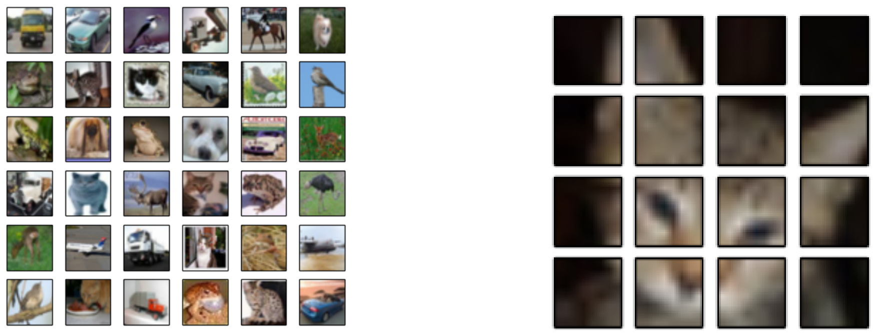


Figure 1: Example images from the CIFAR-10 dataset, and a dicing of one particular image into 16 8×8 non-overlapping patches. Normally, we will infact allow patches to overlap in order to support translation invariance.

level features to help reduce the dimensionality of middle layers and increase the quality of learned features. This can be seen in our accuracy on the CIFAR-10 dataset.

2 Related Work

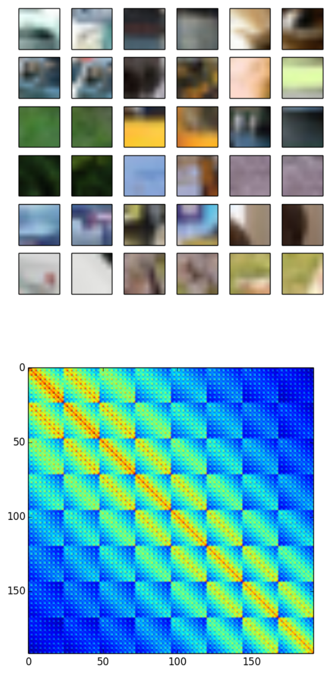
Object classification has had a long history of research and experimental results. In particular, many researchers in the past decade have made attempts at understanding the CIFAR-10 dataset with varying degrees of success. In this section we first describe several other methods for performing feature learning, consider several algorithms which use features to boost object classification performance, and finally describe the success of some of these methods on the CIFAR-10 dataset.

Feature learning can be split into three categories - supervised approaches semi-supervised approaches, and lastly unsupervised feature learning. In terms of semi-supervised approaches, perhaps the most classic approach is that of Nigam et al. who describes how to learn text classifiers when there are limited number of labeled examples [12]. They in a sense overcome the dearth of labeled examples by using unlabeled documents to support the data. This is a somewhat counterintuitive result, but can be rationalized by understanding that unlabeled data, based on inferences about which class an unlabeled example belongs to can be used to boost the joint probability distribution of features in the document. The authors ultimately used EM (Expectation Maximization) with the assumption that the underlying data arises from a mixture model to classify text.

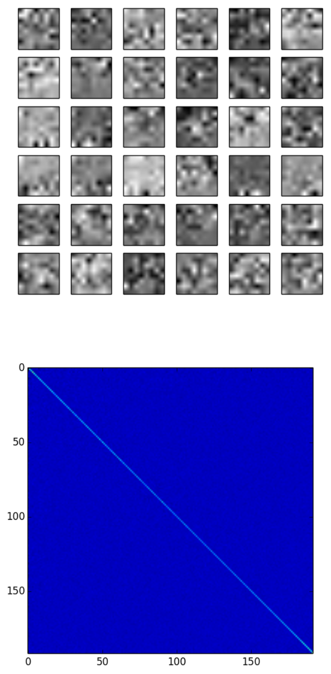
Another recent feature learning approach is pioneered by Raina et al. and runs under the moniker of Self-Taught Learning [13]. Self-Taught Learning works similarly to the EM-based approach described above, but makes no assumptions about the underlying distribution or classes of the unsupervised data (there is no assumption that the unlabeled data even contains objects that are provided in the training and test datasets). Instead, they exploit the fact that the unsupervised data is a collection of natural images. This makes their algorithm significantly easier to apply in practice, as one can simply take a set of labeled examples, and augment it with an enormous unlabeled data-set of support examples.

Both approaches are interesting due to their real world practicality - which we are deeply concerned with. Both approaches augment their small set of labeled examples with a large number of unlabeled examples, and use the unlabeled examples to improve performance on the test dataset. In the world, labeled examples are prohibitively costly - therefore it is imperative to have techniques that will be almost fully unsupervised.

For the past decade neural networks have been making a comeback [5], and a large part of that is due to the application of neural networks to feature learning. Many of the approaches we include here originated as components of neural networks, such as the convolution and pooling of features [10] [9]. One of the major problems with neural networks is that they are notouriously difficult to train, with many different tunable hyper-parameters. This is where clustering based approaches can



(a) Before whitening



(b) After whitening

Figure 2: Top row: a sample of 64 random patches from the unsupervised data. Bottom row: covariance matrix of all image patches. (red means large value, blue means small value). The n^{th} row and column represents the $i * \text{width} + j$ pixel of the patch. As can be seen from 2a there is a large amount of correlation between neighboring pixels, which we seek to remove.

help signficantly. Specifically, k-means has just a single tunable parameter, which makes it very attractive for feature learning.

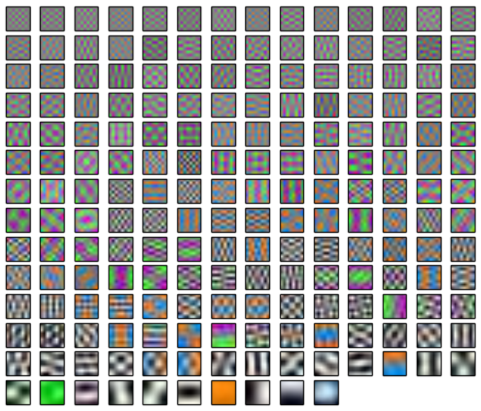
3 Datasets

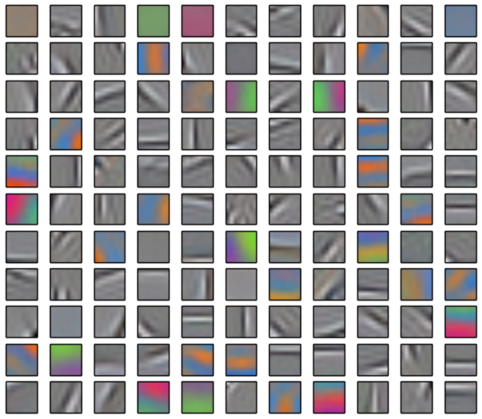
We choose to evaluate our feature learning algorithm on the CIFAR-10 dataset. CIFAR-10 is a standard dataset of 32x32 color images of a single object in one of ten categories. It contains 60,000 labeled images, with 50,000 for training and 10,000 for testing. We use the dataset both for both feature learning as well as classification. We extracted a large number of random patches from the training examples, which we then used in the k-means phase. Then, for each image, we extracted all possible patches and built the new features from these patches, which we subsequently pooled.

4 Algorithm

We now describe the image classification algorithm, based on [4]. Assume that we have a dataset with large amounts of unlabeled data from a similar domain to the classification algorithm. Notice that this assumption differs from that in semi-supervised learning in that we are not garanteed that the unlabeled data contains only examples from what we want to classify. For example, if we are classifying birds vs. planes we may have images of buildings or plants in the unlabeled data.

1. Extract random patches from the unsupervised data. Standardize the patches, so that each patch has mean 0 and standard deviation 1. Whiten the input patches to make the covariance matrix of the each patch approximately the identity matrix. This can be done via PCA or ZCA, as described in [8]. The effect of whitening on the can be seen in Figure 2.


(a) All 192 eigenvectors of a selection of 8x8 color patches from the CIFAR dataset.


(b) 100 centroids learned from applying K-means to 8x8 patches extracted from natural images. Notice the similarity to Gabor wavelets.

This means that we do not waste learning resources learning unneccessary imformation, such as knowing that neighboring pixels are more likely to have similar colors. In Figure 3a we show the eigenvectors we extract when performing PCA/ZCA. We see that each eigenvector corresponds to a different frequency basis. These eigenvectors are very similar to those from the discrete cosine transform (DCT), which is often used in image compression.

We can also optionally use PCA to reduce the dimensionality of the data. This doesnt help with accuracy of the quality of features learned, but it does speed up the pipeline significantly. In fact, we show in the analysis section that performing PCA is not a good idea in general.

2. Cluster these patches using K-means with the euclidean or cosine distance metric. Generally it is best to pick as large a k as the amount of data will allow, however metrics such as silhouette scoring etc. can also be useful if data is severely limited. Figure 3b shows the centroids learned when performing euclidean K-means with 100 centroids. These features are similar to those learned by blind source seperation, such as ICA [6].

3. For each training example, extract all patches from each image. Normalize and whiten these patches as in 1. Get some measure of distance from the centroids learned by K-means. There are many approaches that can be taken here. For example, we can find the euclidean distance to each patch or convolve each centroid with the whitened patches from the image. We can also use the distance metric suggested by Coates which has the benefit of adding some sparsity: $f_{ij} = \max(0, \mu_i - x_{ij})$, where μ_i is the average distance from the patch to each centroid and x_{ij} is the distance from patch i to centroid j.

4. Pool these feature responses in order to reduce the dimensionality. One approach is to pool the responses into a $n \times n$ grid. Pooling is beneficial since it acts as a form of translation invariance.

5. Pass these features into your favorite linear classifier (e.g. SVM or logistic regression) or pass these features back into the K-means pipeline described above. If we reuse the k means pipeline, it is important to develop some way to reduce the dimensionality of the features in higher levels of the pipeline. One way we can do this is to measure the correlation in terms of activation between different features, and then combine the highly correlated features. For example, this can happen with edges that share very similar angles, but perhaps have a slighly different translations.

   We also experimented with using sparse k means as described in [14] in the higher layers, which helps to filter out those features that do not significantly help representation.

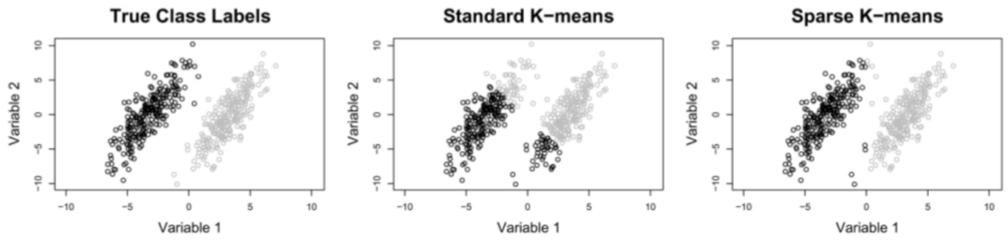


Figure 4: Sparse K-means helps us perform feature selection (right). Notice that without it (middle), k-means gets distracted by the irrelevant features.

4.1 Extensions

One of the benefits of feature learning is that we can extend our learner to take the output features from one layer of the k-means learner and use them as input to the next layer. We have to be careful when extending the feature learner to multiple layers, since we can end up with very high dimensional features. Therefore we have to perform some kind of feature selection. There are two main ways to do this, construct many local receptive fields based on some kind of similarity or correlation metric, or to perform feature selection within a regularized form of k-means.

4.1.1 Fusing Features

We can use the idea of the receptive field [1] to select a small subset of highly related features with which we will pass into the usual feature learning pipeline described above. The reason we do this is to reduce the dimensionality of the data, by only consider small subsets of the feature at a time and then constructing independent banks of features, which we then use later as the basis for our higher level layers.

[3] have suggested using the squared correlation between two features as a measure of their dependency. The dependency metric is

$$d(z_j, z_k) = \mathrm{corr}(z_j^2, z_k^2) = \mathbb{E}[z_j^2 z_k^2 - 1] / \sqrt{\mathbb{E}[z_j^4 - 1]\mathbb{E}[z_k^4 - 1]} \tag{1}$$

We then select the top R features (100 or so) from one random seed feature. We then solely use these top R features to learn a higher level set of features. We then combine this with the features learned from other seeds and these become our new features.

4.1.2 Regularized k-means

K-means is not by default robust to non-discriminative features. Specifically, if we have a lot of features that will not help us to seperate different classes, then those features will actually hurt the clustering algorithm. This can happen often when we have high dimensional data, i.e. $p \gg n$. This problem is shown in Figure 4. Here, we notice that the second feature does not help us to determine which class a point belongs to. Worse, in this case it actually changes the assignment of the clusters. We want to remove these features from consideration.

Here is where we can use a regularized form of k-means, which assigns a weight to each feature as it clusters.

Specifically, we wish to maximize the weighted between cluster sum of squares, which is simply framed as

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{j=1}^{p} w_j \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right) \\ \text{subject to} \quad & ||w||^2 \leq 1, \ ||w||_1 \leq s, \ w \geq \mathbf{0} \end{aligned} \tag{2}$$

Method	Accuracy
Random	10.0%
Raw pixels	37.3%
K-means + all PCs	77.9%
K-means + 0.99 variance	50.0%
3 layers	82.8%
Sparse 2 layers	77.1%
Sparse 3 layers	**83.5%**
Deep Neural Network [2]	80.49%
Network in Network [11]	**91.2%**

Table 1: Performance of the different methods we tried.

The $||w||_1 \leq s$ constraint ensures sparsity when s is small. The $||w||^2 \leq 1$ constraint ensures that we have more than one non-zero feature. Obviously we do not want any negative weights, which is why we enforce $w \geq \mathbf{0}$.

The details of how we actually optimize this function are not important for our purposes, but the details are discussed at length in [14], along with good methods for selecting s. We did not explore how s changes the clustering results.

We can combine this method with the correlation idea above, which helps us to automatically throw out certain dimensions, which further reduces dimensionality.

5 Experiments

We analysed the above algorithm by implementing the full pipeline, and then testing the effects of changing certain parts. We will discuss the effect of depth, the effect of PCA for dimensionality reduction and the effect of sparse k-means.

Table 1 shows the accuracy of different algorithms with all 50000 training examples from CIFAR-10. The algorithms were

1. randomly choosing a class.
2. Taking the raw pixels from each image and feeding them into an SVM
3. Single layer K-means we have described above.
4. K-means with PCA run on the patches, keeping 99% of the variance.
5. Stacking the K-means feature learner into a deep network.
6. 2-layer network, with sparse second layer.
7. 3-layer network, but we use sparse K-means at the deeper levels.

8/9. Comparison to methods using neural networks. State of the art is $> 90\%$, but this is with difficult to train neural networks.

Since we have 10 classes in the dataset we tested, we want to do better than 10% in classification. In fact, we really have a stricter baseline, since we have no interest in the classifier that the learned features are output. Therefore, we hope that our learned features will do better than raw pixels when fed into an off the shelf classifier, such as a linear SVM. As can be seen from Table 1, running an SVM against raw pixels performs poorly, suggesting that raw pixels do not make for a good feature representation for image classification. In fact, we suspect that most of the accuracy for this method is explained by the color of different images, since different classes have different distributions of color.

Running PCA on image patches also sacrifices a large amount of accuracy. If we were to run PCA on the data in Figure 4 we would not be able to get good clusters on the new space. This is reflected in the accuracy drop from using PCA based on the total variance of the data.

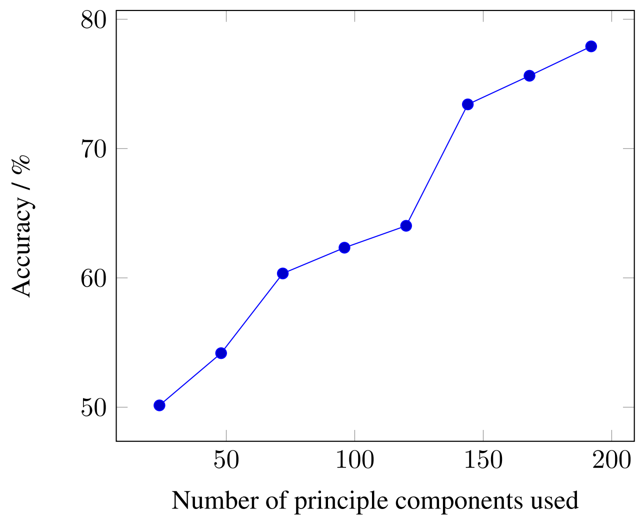


Figure 5: Using PCA to perform dimension reduction is not always a good idea. Here we show the accuracy on a single layer k-means, when only keeping n principle components per patch.

5.1 K-means

The k-means pipeline, described in [4], which we implemented gets 77.9% accuracy. If we extend this to multiple layers as described in [3] we get 82.8%. Finally, if we combine the 3 layer k-means pipeline with sparse k-means, we further increase the accuracy to 83.5%. This makes sense, since in higher layers, we start to get both 'lazy features', that is features that do not discriminate between images, as well as bad features which might get through the earlier stages for some reason - for example, features from clusters that were singletons or had very few members. Therefore it is very reasonable to see a slight improvement from using a sparse k-means.

5.2 Deeper Networks

We experimented with adding depth to the pipeline, by stacking the learned features, along with the methods described above to reduce dimensionality. We were able to build meaningful features up to a depth of three, although we were suprised at how well a single layer performed when compared to the comparitively weak improvement by adding depth.

In fact, two layers did not improve upon a single layer. The reason we use three layers, and not two, is because two layers does not give enough expressive power to adequetly explain the data. Additionally, with two layers we significantly increase the dimensionality of the data, whilest not getting enough additional information from the higher level features.

For reference, the k-means approach is competitive with deep neural networks, although recent regularized neural networks have outperformed us. The fact that clustering based approaches are competitive with neural networks is suprising, considering the expressiveness of neural networks.

5.3 PCA

We experimented with running principle component analysis (PCA) on the original image patches in order to reduce the dimensionality of the data before passing them into k-means. There are two benefits to reducing dimensionality before running k-means. The first is that with lower dimensions significantly less data is required for PCA to perform well. Secondly clustering in a low dimensional space takes up much less time and memory.

When we run PCA on the input image patches, we get a significant reduction in accuracy, as can be seen in Figure 5. In fact, this reduction in accuracy makes a lot of sense, since we expect that most of the pixels in each patch will have a reasonably similar importance in representing each image, since the patches are extracted at random from the image.

We therefore suggest not running PCA (or any linear dimensionality reduction tool) on the data as a preprocessing step.

6 Future Work

Recent results in the deep learning community have begun to investigate how to regularize neural networks. This has led to a major increase in performance of almost all neural networks. It would be interesting to see if some of those approaches could be modified to work in our pipeline. For example, it might well be possible to perform micro-clustering over features in order to combine them.

It would also be interesting to see if we could add dependencies between neighboring patches in order to enforce some global consistency, without the need for the deeper level features. This could help clear up some of the wasted resources in extending to a deep network.

It would also be nice to see if we could add rotation invariance to the pipeline, which at the moment is not taken care of at all. Rotation invariance would have the added benefit of allowing our pipeline to recognize that different orientations of gabor wavelets are really the same thing. At the moment, a large number of filter banks are wasted representing the same filter at different orientations, which prevents less dominant, but equally important high frequency wavelets from being expressed. This can in fact be seen in Figure 3b, where there are very few high frequency Gabor filters represented.

7 Conclusion

We have described a general pipeline to use k-means to learn features, which can then be fed back to form a deep heirarchy of features. We show that this method is competitive with neural networks in terms of learned features and indeed learns similar features.

It is also possible to use the k-means pipeline recursively in order to learn higher level features, which allows for a much more expressive array of features and indeed helps to improve the quality of features, as shown by the increase in accuracy.

We also showed that sparse k-means can be used to perform feature selection in higher layers of the pipeline and that this improves on simple correlation based feature receptive fields as well as beating simple PCA based dimension reduction.

References

[1] BARROW, H. G. Learning receptive fields. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA) (1987), vol. IV, Piscataway, NJ: IEEE, pp. 115–121.

[2] CIRESAN, D. C., MEIER, U., MASCI, J., GAMBARDELLA, L. M., AND SCHMIDHUBER, J. Flexible, high performance convolutional neural networks for image classification. In *IJCAI* (2011), T. Walsh, Ed., IJCAI/AAAI, pp. 1237–1242.

[3] COATES, A., AND NG, A. Y. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 561–580.

[4] COATES, A., NG, A. Y., AND LEE, H. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics* (2011), pp. 215–223.

[5] HINTON, G. E., OSINDERO, S., AND TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput. 18*, 7 (July 2006), 1527–1554.

[6] HOYER, P. O., AND HYVRINEN, A. Independent component analysis applied to feature extraction from colour and stereo images. In *Network (Bristol, England)* (2000), pp. 11(3):191–210.

[7] HUBEL, D. H., AND WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology 160* (Jan. 1962), 106–154.

[8] KRIZHEVSKY, A. Learning multiple layers of features from tiny images. In *Technical report* (2009).

[9] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[10] LE, Q. V., NGIAM, J., CHEN, Z., CHIA, D., KOH, P. W., AND NG, A. Y. Tiled convolutional neural networks. In *In NIPS, in press* (2010).

[11] LIN, M., CHEN, Q., AND YAN, S. Network in network. *CoRR abs/1312.4400* (2013).

[12] NIGAM, K., MCCALLUM, A. K., THRUN, S., AND MITCHELL, T. Text classification from labeled and unlabeled documents using em. *Machine learning 39*, 2-3 (2000), 103–134.

[13] RAINA, R., BATTLE, A., LEE, H., PACKER, B., AND NG, A. Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 759–766.

[14] WITTEN, D. M., AND TIBSHIRANI, R. A framework for feature selection in clustering. *Journal of the American Statistical Association 105*, 490 (2010), 713–726.