Active Learning with Clustering and Unsupervised Feature Learning

Abstract. Active learning is a type of semi-supervised learning in which the training algorithm is able to obtain the labels of a small portion of the unlabeled dataset by interacting with an external source (e.g. a human annotator). One strategy employed in active learning is based on the exploration of the cluster structure in the data, by using the labels of a few representative samples in the classification of the remaining points. In this paper we show that unsupervised feature learning can improve the "purity" of clusters found, and how this can be combined with a simple but effective active learning strategy. The proposed method shows state-of-the art performance in MNIST digit recognition in the semi-supervised setting.

Keywords: Active Learning, Clustering, Unsupervised Feature Learning.

1 Introduction

Active learning algorithms can exploit labeled and unlabeled data, as in the general semi-supervised setting, but instead of using a predefined set of labeled examples, the training algorithms is allowed to query an oracle during training (e.g. a human annotator) to obtain the labels of the training examples considered the most informative. This approach is particularly useful in problems in which unlabeled data is abundant, but obtaining labels is expensive, such as in protein classification [18].

There are two main strategies employed in active learning. The most commonly explored in previous works is based on the use of labeled examples near the decision boundary of a discriminative classifier to direct the search in the hypothesis space in a efficient way. This strategy assumes that points near the decision boundary are more informative and gives little importance to points farther away [5]. The second strategy is based on the manifold hypothesis, according to which points of real data concentrate in the neighborhood of a low-dimensional manifold embedded in a high-dimensional space [12,4]. In an ideal scenario, a clustering algorithm would isolate points of different classes in their own clusters and one labeled sample for each would be enough to correctly classify the remaining examples [6]. In practice finding good clusters in complex and high-dimensional data such as images is not easy.

It has been long suggested that images could be recognized by multiple layers of feature detectors [15], but it has not been until recently that effective

methods for training models with multiple layers have been devised [8,3]. The first successful techniques relied on a greedy layer-wise unsupervised procedure (pre-training) for initializing the network before applying backpropagation. One perspective of pre-training procedure is that each layer learns to extract good features from the layer below. In image recognition, for example, a linear classifier in the output layer can more easily separate different classes in terms of higher level features, such as object parts, than in terms of the original input, such as pixels.

In this work we explore the intuition that the same layer-wise unsupervised training procedure used in deep learning can also be used to transform the input from the original space to a feature space in which clusters are more easily identifiable. We start by applying an unsupervised feature learning technique to extract features from unlabeled data. We then convert the raw input into features and use the k-means algorithm to find clusters in the feature space. Then we query the labels of the training examples closest to cluster centroids and classify all points in the same clusters with the same labels. We also propose a probabilistic model to determine the quality of the classification and remove the "excess" of points classified with low probabilities. Finally, we use the training examples with the labels found as the training set of a neural network. We demonstrate experimentally that unsupervised feature learning can improve clustering purity and we show how the learned features can be combined in a simple but effective active learning strategy.

2 Unsupervised Feature Learning

Every clustering algorithm needs a measure of dissimilarity (or similarity), such as the euclidean distance used in k-means. In certain datasets it is easy to notice that the euclidean distance does not represent well our intuitive notion of what similar points are. In digit images for example a shifted version of a digit will stop sharing most pixels in common with the original version and thus their distance will be large, although they still belong to the same class. Unsupervised feature learning techniques can be used to convert the raw input to a feature space where the distance metric can more easily represent the actual notion of dissimilarity between data points. If we have features which represent objects parts, for example, it is easier to tell whether two images are similar or not by checking if they share the same object parts, rather than by comparing their raw pixels.

2.1 Denoising Auto-Encoder

Unsupervised feature learning has been one of the main factors behind the recent breakthroughs in machine learning [2]. Denoising auto-encoder (DAE) is a technique that has been proven to yield useful representations despite its simplicity. A DAE is a neural network trained to reconstruct a clean input from a noisy version [16]. More formally, let $\tilde{x} \in [0,1]^d$ be a corrupted version of the

input $x \in [0,1]^d$ and $\theta = \{W, b, W', b'\}$ the encoder and decoder parameters, respectively. The network computes the reconstruction z, given by

$$z = g(W' f(W\tilde{x} + b) + b')$$

where f and g are the activation functions. Usual choices are the logistic function $s(x) = (1 + exp(-x))^{-1}$ for both encoder and decoder, and the linear activation function for the decoder when the data is real valued. The network parameters are optimized such that

$$\theta^* = \underset{\theta}{arg\,min} \frac{1}{n} \sum_{i=1}^{n} L\left(x^{(i)}, z^{(i)}\right)$$

where L is a loss function, usually the squared error $L(x, z) = \|x - z\|^2$ for real data, and the reconstruction cross-entropy

$$L_H(x, z) = -\sum_{i=1}^{d} x_i \log z_i + (1 - x_i) \log (1 - z_i)$$

for binary or near binary inputs. To generate $\tilde{x}$, one common option is to add masking noise to the input (i. e. to set each element of x to zero with probability p).

2.2 Stacking Denoising Auto-Encoders

A DAE can be seen as a mapping from a lower to a higher level representation. Using the output from a previously trained DAE to train a new DAE, we can build a Stacked Denoising Auto-encoder (SDAE), which is capable of learning still higher levels representations [17]. Although this greedy unsupervised layer-wise procedure has been shown to be useful for initializing deep networks for subsequent fine-tuning, it has fallen out of favor since the publication of better optimization techniques, such as Hessian-Free Optimization, which can achieve comparable (or even better) results [11].

3 Active Learning by Clustering

Finding one single big cluster per class in the data in a purely unsupervised way is intrinsically hard or even impossible. Different criteria used to judge similarity (e.g. weights given to different features) can lead to different clusters, which do not necessarily correspond to the classes we want to find. We can though assume that in general nearby examples belong to the same classes independently of the distance metric used, forming locally small homogeneous clusters. We can explore this local structure for active learning by querying the labels of representative samples from each cluster and assuming their neighbors share the same classes.

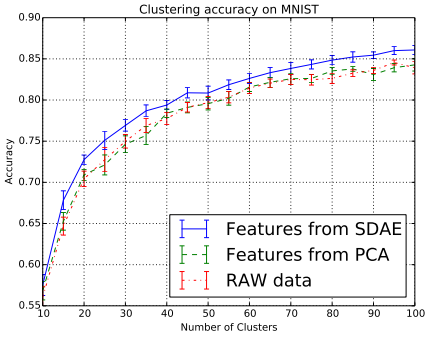

Fig. 1. Classification accuracy with 10000 training examples from MNIST. Each image is labeled with the most frequent class in the cluster. Applying PCA is not better than using the original input, while features extracted by a SDAE lead consistently to higher accuracies. Each point is the average of 10 runs of k-means with random initialization. The error bar shows the 95% confidence interval.

3.1 Clustering Evaluation on MNIST

The clustering evaluation measure which best reflects our goal of obtaining clusters containing elements from a single class is the *Purity*. It is a external criterion (i. e. compares cluster assignments against a gold standard) which is applicable even when the number of clusters is different from the number of known classes. It is given by

$$Purity = \frac{1}{N} \sum_{k} \max_{j} n_j^k$$

where N is the total number of data points and n_j^k is the number of points in cluster k from class j. If we treat the most frequent class of each cluster as the label given to every element of the cluster by a predictive model, this measure is equivalent to the common notion of classification accuracy. Whereas we are most interested in the final prediction, we will therefore use the term accuracy throughout the paper as a synonym of purity.

To assess the improvements of clustering after feature extraction, we compared the accuracy found by using different features as input. Fig. 1 shows the increase in classification accuracy in function of the number of clusters used (i.e. the number of labeled training examples) on a portion of MNIST training set. While Principal Component Analysis (PCA) did not improve the accuracy, SDAE consistently led to lower error rates. We tested PCA with number of components ranging from 50 to 500 and did not find any significant difference in the results. A major concern in semi-supervised learning is the lack of validation data for hyperparameter optimization. We assume the premise that labeled data is scarce, thus any additional labeled data that could be used for validation would be in most real scenarios better spent as part of the training set. We have therefore tested a reduced set of hyperparameters to prevent overfitting (one

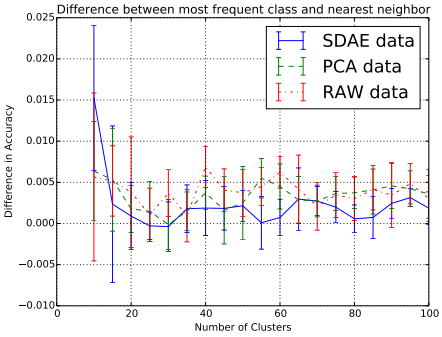

Fig. 2. Difference in accuracy by classifying points in each cluster with the most frequent class within it and with the label of the points closest to cluster centroids. For a number of clusters slightly higher than the number of classes (10) the difference is close to zero. The difference tend to be slightly lower when using SDAE to extract features.

and two layers, corruption level 0.1 and 0.2, and layer size 300 and 500) and used the best performing model in all our experiments (SDAE with 2 layers, 0.2 corruption level, and 500 units per layer).

3.2 Finding Representative Samples

By using k-means as the clustering algorithm, we can use as the representative sample from each cluster the data points closest to their centroids. We then assign to all points in the same cluster the same label. This is equivalent to classifying the dataset with a k-nearest neighbors algorithm using the cluster centroids as the training examples (with k equals to one). This procedure leads to results almost identical to assigning to each cluster the most frequent label within it, as shown in Fig. 2. On MNIST dataset using just 10 clusters the difference between the two procedures is in average less than 1%. Using more than 15 clusters, this difference falls quickly to less than 0.005%.

3.3 Determining Classification Probability

Since the most central training example in a cluster obtained by k-means is a good representative of the most frequent class in the cluster, we can presume that the most central points have higher probabilities of being correctly classified, while points farther from cluster centroids have lower probabilities. However points which lay in-between clusters of the same class are likely to be correctly classified, despite of being peripheral, because two neighbor clusters of the same class suggest the existence of a bigger cluster in which both are contained.

We tested two models to evaluate the conditional probability of a point belonging to a class. Let $d(x, p)$ be the euclidean distance from x to p and C_k the

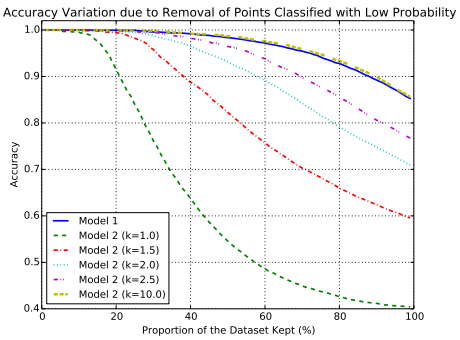

Fig. 3. After classification, points were sorted accordingly to the difference between the two highest probabilities. With both models we can obtain near 100% accuracy by keeping about 40% of points classified with highest margin between first and second guesses. Model 2 is worse than Model 1 for low values of K. For K = 10 both model are almost equivalent.

set of cluster centroids from class k, we define the probability of point x belonging to class j as the normalized inverse distance from x to the nearest neighbor from class k, given by

$$p(j \mid x) = \frac{\left(\min_{p \in C_j} d(x,p)\right)^{-1}}{\sum_i \left(\min_{p \in C_i} d(x,p)\right)^{-1}}$$

or more simply

$$p(j \mid x) = \frac{\sum_i \min_{p \in C_i} d(x,p)}{\min_{p \in C_j} d(x,p)}$$

which accounts as evidence for a class just the distance of the single closest point from the class. Our second model takes into consideration the sum of the inverse distances to every point in the class. The probability is given by

$$p(j \mid x) = \frac{\sum_{p \in C_j} d(x,p)^{-K}}{\sum_i \sum_{p \in C_i} d(x,p)^{-K}}$$

where K is a constant which regulates how quickly the influence of a point decays with distance. For large values of K the influence of the nearest points tend to dominate over the influence of more distant points, and this model when used for ranking the best classifications gives similar results as the first, but with less meaningful probabilities.

We can use these models to identify points classified with low probabilities. This can be useful in the labeling process of a large dataset. It makes more sense to spend the most resources labeling the "hard" cases instead of the easier ones. Removing the excess of wrongly labeled points can also be useful in the next

Table 1. Error rate comparison (on MNIST test set)

Method	100	600	1000
Supervised Neural Network	25.13	11.21	9.84
Manifold Tangent Classifier [14]	12.6	5.13	3.64
Semi-Supervised Embedding [19]	**7.75**	3.82	**2.73**
Pseudo-label [9]	10.49	4.01	3.46
AL + MLP	8.17	4.57	3.86
AL + ConvNet	7.98	**3.73**	3.32

step, when training a neural network. Wrong labels can degrade the performance of a classifier. Fig. 3 shows how the accuracy varies accordingly to the proportion of points removed. With high values of K the results of both models are almost identical, for low values of k the first model is better. This difference occurs because the cluster structures are not confined in convex regions in space. The information of far points is not as meaningful as the information of closer points, thus in general considering just the nearest neighbors is better.

3.4 K-means Initialization

The qualities of clusters found by k-means can vary significantly with the initialization scheme used. Initialization with k-means++, which chooses as initial centroids data points more evenly spread than by random initialization [1], results in average in higher accuracies ($t(58) = 3.66$, $p <= 0.001$). There is a modest negative correlation ($r = -0.26$, $p < 0.001$) between clustering accuracy and the k-means distortion function (sum of squared distances from points to cluster centroids to which they are assigned), given by

$$\sum_{i=0}^{N} \|x_i - \mu_{x_i}\|^2$$

where N is the number of points and μ_{x_i} is the centroid of the cluster to which x_i belongs. Repeating the clustering step many times with k-means++ initialization and selecting the run with the lowest distortion results in a significant improvement in final accuracy.

4 Neural Network Training

We can improve further our results by using the labeled examples obtained so far as train data of a supervised model. We trained two architectures of neural networks: Multilayer Perceptron (MLP) and Convolutional Neural Network (ConvNet). In Table 1 we compare the error rate of our active learning strategy (AL) with other semi-supervised models on MNIST test set when using 100, 600

and 1000 labeled examples. To represent the expected performance in a more realistic setting, we ran the clustering step with 30 random initializations and selected the model with the median accuracy to provide the labeled examples used in the neural network training. We chose not to use an additional validation data, as before, for hyperparameters optimization and used instead 10000 samples from the training examples with the labels found in the previous step. As a negative effect of adopting this validation set "contaminated" with wrong labels, early stopping had its efficacy reduced. Even though, we achieved the lowest error rates reported for 600 labeled examples. For the other values, our results were just slightly worse than the results obtained by Semi-Supervised Embedding.

We have observed that removing the excess of incorrectly labeled data, excluding the training examples classified with low probabilities has somewhat unpredictable effects. Although the training data created this way has fewer wrong labels, it also has less diversity. In general this procedure does not seem to improve the final accuracy and in some cases the results are even worse, showing that the diversity of the dataset can be as important as its size.

5 Related Work

One of the first attempts to explore the clustering structure in data for active learning was made by Xu et al. [20], who employed k-means clustering to draw representative samples from unlabeled examples which were used to speed up the convergence of Support Vector Machines. Zhu et al [21] explored the manifold structure in the data by modeling the probability distribution of points belonging to classes as a Gaussian Random Field built on a graph whose nodes are the training examples and the edges encode the distances between them. Nguyen et al. [12] addressed some shortcomings of the two previous works by taking measures to avoid repeatedly labeling samples in the same clusters. They used the representative samples from clusters to train a discriminative classifier to obtain the labels from the remaining unlabeled points (differently from our work which uses a distance based classifier to propagate labels). The idea of using stacked auto-encoders before clustering was explored by Lefakis et al. [10], but this work focused in the use of under-complete representations and "regular" auto-encoders (without further regularization, such as noise), which usually learns poorer features.

6 Conclusion and Future Work

In this work, we explored the use of unsupervised feature learning together with clustering to build a simple and effective active learning strategy which shows state-of-the-art performance on MNIST in the semi-supervised setting. Our results raises a series of questions to orient future investigations. Unsupervised feature learning using only MNIST data is too limited. Humans learn from a lot of unlabeled data before being able to recognize handwritten digits and more

complex objects. This idea of learning features from one kind o data and using them in a different context has already been successful explored in the "self-taught learning" setting [13] and we expect it to be useful in active learning as well. It is also worth to explore other metrics and clustering algorithms. K-means is intimately related to the euclidean distance, which could be possibly replaced by better metrics for comparing objects in a feature space. We briefly experimented some k-means variations, such as spherical k-means, [7] which is based on the cosine similarity, but it seems to suffer from the same k-means and euclidean space problems and did not lead to improvements.

References

1. Arthur, D., Vassilvitskii, S.: k-means ++ : The Advantages of Careful Seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. vol. 8, pp. 1027–1035 (2007)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(8), 1798–1828 (2013)
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. Advances in neural information processing systems 19(1), 153–160 (2007)
4. Cayton, L.: Algorithms for manifold learning. Univ. of California at San Diego Tech. Rep pp. 1–17 (2005), `http://www.vis.lbl.gov/~romano/mlgroup/papers/manifold-learning.pdf`
5. Dasgupta, S.: Two faces of active learning. Theoretical computer science 412(19), 1767–1781 (2011)
6. Dasgupta, S., Hsu, D.: Hierarchical sampling for active learning. In: Proceedings of the 25th international conference on Machine learning. pp. 208–215 (2008)
7. Dhillon, I.S.: Concept Decompositions for Large Sparse Text Data using Clustering. Machine Learning 42(1-2), 143–175 (2004)
8. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation 18(7), 1527–1554 (2006)
9. Lee, D.H.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on Challenges in Representation Learning, ICML (2013)
10. Lefakis, L., Wiering, M.: Semi-Supervised Methods for Handwritten Character Recognition using Active Learning. Proceedings of the BelgiumNetherlands Conference on Artificial Intelligence pp. 205–212 (2007)
11. Martens, J.: Deep learning via Hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 735–742 (2010)
12. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedings of the twenty-first international conference on Machine learning. p. 79. ACM (2004)
13. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th international conference on Machine learning. pp. 759–766. ACM (2007)
14. Rifai, S., Dauphin, Y.N., Vincent, P., Bengio, Y., Muller, X.: The manifold tangent classifier. In: Advances in Neural Information Processing Systems. pp. 2294–2302 (2011)

15. Selfridge, O.G.: Pandemonium: A paradigm for learning. In: Proceedings of the Symposium on Mechanisation of Thought Processes. vol. 1, pp. 511–529. HMSO (1959)
16. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. pp. 1096–1103. ACM (2008)
17. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research 11, 3371–3408 (2010)
18. Weston, J., Leslie, C., Ie, E., Zhou, D., Elisseeff, A., Noble, W.S.: Semi-supervised protein classification using cluster kernels. Bioinformatics (Oxford, England) 21(15), 3241–7 (Aug 2005)
19. Weston, J., Ratle, F., Mobahi, H., Collobert, R.: Deep learning via semi-supervised embedding. In: Neural Networks: Tricks of the Trade, pp. 639–655. Springer (2012)
20. Xu, Z., Yu, K., Tresp, V., Xu, X.W., Wang, J.: Representative Sampling for Text Classification Using Support Vector Machines. Springer Berlin Heidelberg (2003)
21. Zhu, X., Ghahramani, Z., Lafferty, J., et al.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML. vol. 3, pp. 912–919 (2003)