LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-644296

Features in Deep Learning Architectures with Unsupervised Kernel k-Means

K. S. Ni, R. J. Prenger

September 27, 2013

GlobalSIP
Austin, TX, United States
December 3, 2013 through December 5, 2013

Disclaimer

Learning Features in Deep Architectures with Unsupervised Kernel k-Means

Karl Ni, Ryan Prenger
Video Laboratory Directed Research and Development
Lawrence Livermore National Laboratory
ni4@llnl.gov, prenger1@llnl.gov

***Abstract*—Deep learning technology and related algorithms have dramatically broken landmark records for a broad range of learning problems in vision, speech, audio, and text processing. Meanwhile, kernel methods have found common-place usage due to their nonlinear expressive power and elegant optimization formulation. Based on recent progress in learning high-level, class-specific features in unlabeled data, we improve upon the result by combining nonlinear kernels and multi-layer (deep) architecture, which we apply at scale. In particular, our experimentation is based on k-means with an RBF kernel, though it is a straightforward extension to other unsupervised clustering techniques and other reproducing kernel Hilbert spaces. With the proposed method, we discover features distilled from unorganized images. We augment high-level feature invariance by pooling techniques.**

I. Introduction

Tasks in computer vision, audio and multimedia research, and natural language and text processing require features that saliently describe a semantic concept. One approach is to directly learn low-level features to hierarchically construct classifiers with Haar wavelets [4] and deformable parts [5], but generalization has suffered and labels are costly to produce. Meanwhile, theoretical features, such as the oft-cited SIFT [1] and GIST [2] in vision and standard MFCC's [3] for speech, speaker recognition, and diarization, have been shown to be somewhat successful, though are often not as salient for recognition tasks as more class-specific high-level features. Instead, we have seen a recent push to *learn* high-level features in a completely unsupervised fashion given large enough data sets with deep learning architectures.

The success of high-level features in deep learning architectures [6], [7] has been demonstrated in audio [10], [8], [9] and vision [11], working especially well at scale [12] with sparse autoencoders. While previous computer vision techniques focused on labeled training data sets, deep learning methodology features have shown potential in building class-specifity in an *unsupervised* setting. Meanwhile, the expressive power of nonlinear kernels, particularly in kernel machines like SVMs [13], have been used regularly with much success. In addition to constructing high-dimensional discrimination boundaries, kernels can easily be used to generatively approximate densities (kernel density estimates), parameterize nonlinear regression algorithms (kernel ridge regression and support vector regression), dimensionality reduction (KPCA), and in general, construct spaces that offer more flexibility.

The proposed algorithm seeks to unify the expressiveness of nonlinear kernels with the learning ability of deep networks. Specifically, we explore an architecture that can best be described as **deep kernel k-means**. The nomenclature deliberately denotes the methodology: iteratively determine multiple layers of centroids using clustering, where comparisons are made based on a similarity function. In our case, the similarity function is the radial basis function (RBF) kernel, and the clustering algorithm is k-means.

Our choice of the k-means algorithm, and by extension, kernel k-means [14], was primarily due to its simple implementation, but the generalized deep learning paradigm extends to any unsupervised clustering method. Moreover, although recent successful deep architectures have been implemented with sparse auto-encoders, it has been shown [15] that, at least on a *per layer* basis, certain brands of k-means, i.e., soft k-means or "triangle" k-means, are at least commensurate in quality and often outperform sparse auto-encoders and sparse RBM. The trend has, in fact, inspired some deep implementations with variations on a modified k-means optimization algorithm [16], [17].

The remainder of this paper is devoted to explaining the deep kernel k-means algorithm and its application to learning high-level features. Discussion of the algorithm in the next section includes the construction and application of kernel space, description of the deep architecture, and methods for data augmentation and aggregation. Because the proposed algorithm is derived from several related works, we also explore their relationship to our technique, and compare recognition performance.

II. Algorithm

Deep kernel k-means consists of (1) alternating processes of kernel construction and kernel application, (2) layering with care to patch size, and (3) a good set of training dimenions. We discuss the three aspects of this problem presently:

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

A. Constructing and Applying the Kernel Space

At each level of the deep hierarchy, we determine parameters of a kernel space through kernel k-means. Seminal work in kernel k-means [14] constructs the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ without explicitly calculating $\phi(\mathbf{x})$, the high dimensional mapping, where $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. In practice, however, the kernel *matrix*, one which is massive and grows as $\mathcal{O}(n^2)$, can seldomly be calculated or transductively inferred, and it is generally much simpler to calculate $\phi(\mathbf{x})$ directly.

To construct the kernel space at layer j, we first perform k-means given a set of vectors, $\Omega^{(j-1)} = \{\mathbf{x}_1^{(j-1)}, \mathbf{x}_2^{(j-1)}, \cdots \mathbf{x}_n^{(j-1)}\}$,which is based on transformations on the original training set Ω. At layer one, we use the unadulterated Ω, itself. The j^{th} kernel space is thus entirely determined by the distribution of all the transformed training points via *their* kernel spaces at every, previously calculated layer, i.e., from layers $1, 2, \cdots, (j-1)$.

Each dimension of the new kernel space is based on its RBF similarity to a centroid. That is to say, the i^{th} dimension of the j^{th} transformation layer can be given by $\phi^{(j)}(\mathbf{x})[i] = K(\mathbf{x}, \mathbf{x}_i)$, provided we have previously learned $\phi^{(j)}$, correctly. Here, we train on features $\mathbf{x} \in \{\mathbf{x}^{(j-1)}\}$, the set of features transformed from the previous layer $j-1$, and $\phi^{(j)}(\mathbf{x}) : \mathbb{R}^{k_{j-1}} \rightarrow \mathbb{R}^{k_j}$, where k_j is the number of clusters chosen at layer j.

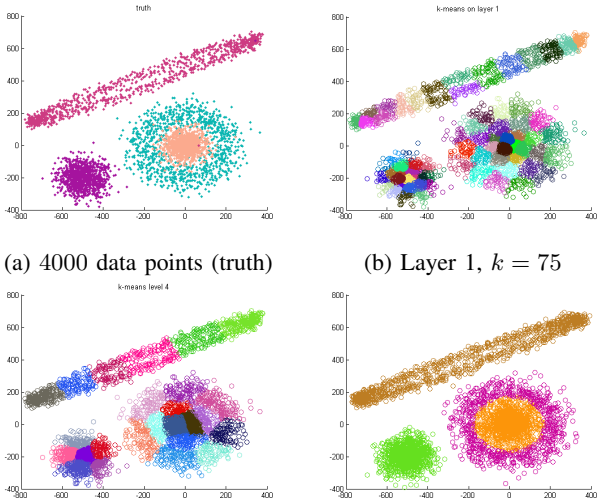
Concretely, the i^{th} dimension of $\phi(\mathbf{x})$ can be defined by $\phi(\mathbf{x})[i] = K(\mathbf{x}, \hat{\mathbf{m}}_i)$, where $\phi(\hat{\mathbf{m}}_i) = \mathbf{m}_i$, and we have:

$$\phi(\mathbf{x}) = \frac{1}{\sum_{i=1}^{k} e^{\frac{\|\mathbf{x}-\mathbf{m}_i\|^2}{2\sigma^2}}} \begin{bmatrix} e^{-\frac{\|\mathbf{x}-\mathbf{m}_1\|^2}{2\sigma_1^2}} \\ e^{-\frac{\|\mathbf{x}-\mathbf{m}_2\|^2}{2\sigma_2^2}} \\ \vdots \\ e^{-\frac{\|\mathbf{x}-\mathbf{m}_k\|^2}{2\sigma_k^2}} \end{bmatrix} \tag{1}$$

Fig.1 is a simple and yet effective application of deep kernel k-means (six layers) to synthetically generated random data. Because the original training set is nonlinearly separable, we start with a large k and successively shrink it to the actual k. The oversampled k (almost effectively a kernel density estimate) in the initial layers not only effectively encompass much of the structure of data, but has also enabled the overcoming poor initialization cases.

Upon further examination, it is apparent that (1) satisfies conditions of being a complete vector space. Indeed, we can (and do) endow it with a dot product, eminently important when we consider k-means for the next level/stage of the deep learner. The dot product is, again, the RBF, and the space is thus, unsurprisingly, a kernel Hilbert space.

The new feature space is not unlike one generated by softmax regression [18] with k classes, which is often used as the final stage of deep learners (as opposed to the autoencoder). The difference is that instead of the softmax correlation exponent, $\mathbf{x}^T\mathbf{m}_i$, which expects normalized inputs, we use the ℓ_2 distance: $\|\cdot\|^2$. Furthermore, in our optimization, we define $\phi(\mathbf{x})$ with the Euclidean distance as a vector space on a GMM



(a) 4000 data points (truth) (b) Layer 1, $k = 75$

(c) Layer 4, $k = 25$ (d) (Final) Layer 6, $k = 4$

Fig. 1: Six layers of unsupervised deep kernel k-means applied to non-linearly separable 2D data.

manifold with independent dimensions, where we, again, use ℓ_2 distance to define a RKHS. We use this terminology because although the RBF is implemented in our rendition of deep kernel k-means, we could certainly generalize to other positive definite functions as long as every dimension is defined by a valid dot product.

Not coincidentally, (1) observes the softmax property that multiplying through e_i^{α} will not bias clustering results. Also note the bandwidth parameter σ can be replaced with a relevant weighting matrix, depending on available computational resources. The normalization constant applied to the entire vector ensures that a single dimension is a posterior probability value, $P(i|\mathbf{x})$, where $\mathbf{m}_i$ is the i^{th} mean.

Having tried several options, the bandwidth parameter σ_i in (1) is actually implemented as a diagonal covariance matrix, Σ_i, dependent on the spread of points within cluster i. We have heuristically chosen

$$\Sigma_i = \frac{\alpha_i}{M} \sqrt{\sum_{m \in \text{cluster } i} \mathbf{x}_m \mathbf{x}_m^T} \tag{2}$$

where M is the number of points in cluster i. In our experience $\alpha_i \approx 6$ yields the best result in the first layer, i.e. the one that distributes the data across an "appropriate" number of centroids. (It is different, though, for higher dimensional vectors.) Due to computational constraints,the full matrix was unnecessary, though interestingly, a diagonal Σ_i matrix did not outperform the spherical case with scalar σ_i by much, especially since we have pre-whitened Ω.

B. Feature Hierarchy

The dimensionality of each layer is determined by the number of clusters k chosen. We use the hierarchy defined in Fig. 2 to train layers of selective and invariant features.

We will train 4 layers, beginning with 32×32 patches from images of size 256×256.

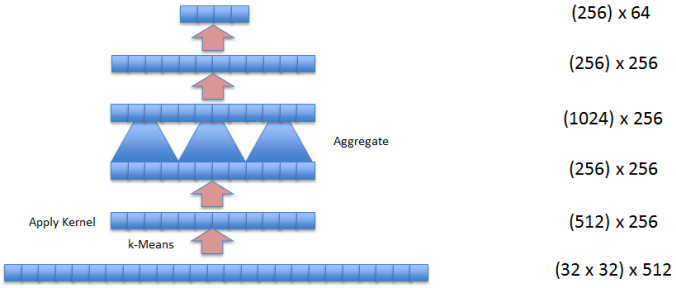


Fig. 2: Deep learning architecture

Fig. 2 depicts the learning architecture for the application to high-level image features. We begin with the original training set of 1.4 million patches, whiten the data, and learn 512 centroids at the first layer. Therefore, the second layer has vectors of dimension 256. From the third to the fourth layer, we conduct a merge of four adjacent patches, meaning that the original 32x32 patches have been joined together to form 64 x 64 patches, and the effective number of patches is now 1.6m/4 = 400k image patches. This is done again to effect 100k image patches, and the final decision is made between 64 clusters defining 64 high-level features.

C. Aggregation and Pooling

Determining the amount of overlap between blocks in training as well as the indexing for data aggregation (e.g., Four 4×4 patches form a single 8×8 patch) have been an obstacle in our implementation. To some extent, using k-means effects a quantizing phenomena, introducing a limited amount of invariance. A significant body of work has been to devoted to the topic of pooling [19] at the lower levels. For the smaller shifts, we typically apply $\max$ pooling, similar to [16].

III. Relationship to Related Work

There is a close relationship that can be observed between kernel k-means to sparse autoencoders, RBM's, and softmax regression. Upon close examination, there are certain circumstances under which the proposed algorithm is equivalent. To first order, it is worth nothing that each dimension, i, of (1) can be thought of as a "neuron", which fires only when $\mathbf{x}$ belongs to cluster i by being close to $\mathbf{m}_i$. Recall the objective function of at each layer

$$\min_W \frac{1}{2N} \sum_N \|h_W(\mathbf{x}) - y_i\|^2 + \lambda_1 T_1(W) + \lambda_2 T_2(W) \quad (3)$$

where the two terms T_1 and T_2 relate to regularization and sparsity. With k-means, there is no such sparsity term, though it is analogous to the $\alpha\hat{\Sigma}_i$ term in (2), regulating the influence of neighboring "neurons". Rather, in the proposed algorithm, sparsity is, to an extent, "enforced", albeit manually, by the value of k. Small values of k limit the number of data points that neurons layer ℓ_i can be similar to in layer ℓ_{i+1}. There is a subtle different, however. For the most part, while neuron dimensionality per a given layer discards similarity information with respect to un-related dimensions in the previous layer (those which the sparsity parameter λ_2 has deemed irrelevant), a single layer in kernel k-means retains it by keeping a high-dimensional representation. That is to say, layer ℓ_{i+1} is also of dimensionality k_i, the number of clusters used to encode the previous layer, retains some amount of information on its similarity to other neurons in layer $\ell i + 1$.

IV. Data and Experimentation

Depending on the training process, we implemented on two architectures. For larger data sets (e.g., TrecVid and Media Eval) a 40-node cluster, each of which has 4 cores and 6GB of memory, and total effective memory of 240GB for resilient distributed dataset objects used with Spark [20], an open source cluster package originating from UC Berkeley's AMPLab. The architecture takes advantage of map/reduce concepts but is 10x faster than traditional HDFS and Hadoop solutions. To prototype, for the CiFAR 10 data set and Faces in the Wild, we found that local MATLAB on Intel Xeon(R) 8-Core 3.3GHz + 64GB Mem, was sufficient (and faster).

A. Lower level image feature representation

The averaged centroids in the lowest feature layer of those that activated the highest feature layer in the deep kernel k-means structure is provided in Fig. 3. At the highest layer, α_N was selected to be very small to produce a type of "selector" vector, from which we took the argument that maximized that cluster's posterior probability.

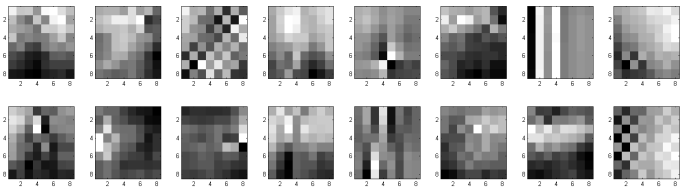
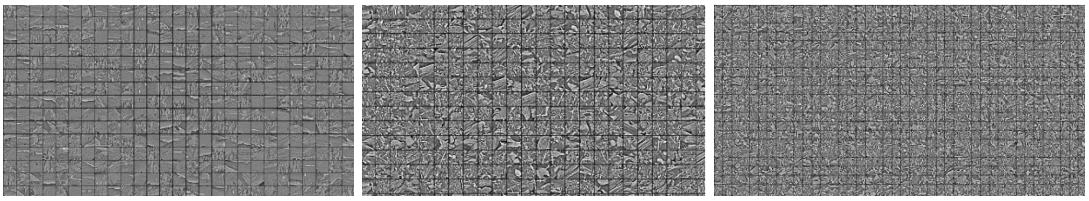


Fig. 3: Low level patches (16 of 64) that "excite" the final layer of k-means from the CiFAR 10 database.

Mid-layer clusters (without aggregation) already show some content discrimination in Fig. 4. As on can tell, there is no consistent orientation of patches, but interestingly, the *type* of texture appears to be preserved. The groupings are consistent with some higher level phenomena. Using the nonlinearity provided by the RBF, it appears that some rudimentary class-based segregation has occurred.

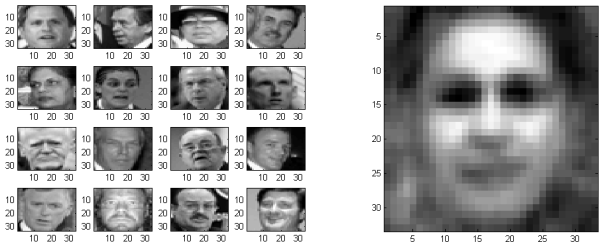
B. High level image feature representation

One of the higher level features depicted in Fig. 5 from the UMass "Faces in the Wild" database produces an interesting result. The input that best activate it are, of course, faces.



(a) Mid-layer kernel cluster on "mountains" (b) Mid-layer kernel cluster on "trees" (c) Mid-layer kernel cluster on "leaves"

Fig. 4: TREC-VID mid-layer data clustering.



(a) Input that activated centroid 55 of $64 > 0.80$ (b) Aggregate over highest activated faces

Fig. 5: High-level feature activation of faces in LFW data set.

C. Detection and Classification

We found that initial resolution of the patch-size altered the performance, depending on the data set size. For the CiFAR10 database, the bottom layer was 8×8, whereas for the TRECVid and Media Eval retrieval challenges, we used 32×32, twice aggregating. The percentages on CiFAR are not quite state of the art, but this is likely due to the training data in this case as we did not pool. However, the concept demonstrates the considerable potential. Kernel GMM's took an incredibly long time, and interestingly did not perform well. This is, perhaps, an artifact of redundancy in constructing the high dimensional space.

TABLE I: Deep Learner Comparisons

	Accuracy
Sparse Autoencoder	80.2%
Sparse RBM	79.0 %
Kernel GMM's	75.6 %
Kernel k-Means	80.6%

V. Conclusions

We have proposed a deep learning architecture using kernel k-means. It has been shown to work on large data sets, and it performs at least as well as comparative methods with considerable ease of implementation. There are a number of future directions that we wish to pursue, mainly experimentation on the types of kernels. We also wish to theoretically prove properties that relate the algorithm to traditional deep learning methodologies.

Acknowledgments

David Buttler has been instrumental to administering our distributed computing efforts. Doug Poland has discussed and directed our work. Grace Vesom and Carmen Carrano have offered valuable input.

References

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2 2004, pp. 91-110

[2] A. Oliva, A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope, " International Journal of Computer Vision, Vol. 42(3): 145-175, 2001. PDF

[3] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," in Pattern Recognition and Artificial Intelligence, C. H. Chen, Ed., pp. 374388. Academic, New York.

[4] P. Viola, M. Jones, "Robust real-time face detection," International journal of computer vision, 2004 - Springer

[5] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2008

[6] Y. Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009

[7] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, 2006

[8] G. Hinton, L. Deng, D. Yu, G. Dahl, A.Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," IEEE Signal Processing Magazine, 29, November 2012

[9] L. Deng, "An Overview of Deep-Structured learning for Information Processing," Microsoft Research Technical Report, 2011

[10] A. Mohamed, G. Dahl, G. Hinton, "Acoustic Modeling using Deep Belief Networks," IEEE Transactions on Audio, Speech, and Language Processing

[11] G. Hinton, "To recognize shapes, first learn to generate images," Technical Report 2006

[12] Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, A.Y. Ng., "Building high-level features using large scale unsupervised learning," ICML, 2012

[13] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK, 2000

[14] B. Scholkopf, A. Smola, K.-R. Muller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," Technical Report 44, Max-Plank Institute

[15] A. Coates, H. Lee, and A. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," In AISTATS 14, 2011

[16] A. Coates, A. Karpathy, and A Ng, "Emergence of Object-Selective Features in Unsupervised Feature Learning," In *NIPS*, 2012

[17] A. Coates and A Ng, "Learning Feaure Representations with K-means," In *Neural Networks: Tricks of the Trade, Reloaded, Springer LNCS*, 2012

[18] D. Hosmer and S. Lemeshow, "Applied Logistic Regression," Wiley, 2013

[19] A. Agarwal, B. Triggs, "Hyperfeatures: Multilevel local coding for visual recognition. In: 9th European Conference on Computer Vision," Vol. 1, pp. 30-43, 2006

[20] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, "Spark: Cluster Computing with Working Sets," HotCloud 2010