Learning Multiple Non-Linear Sub-Spaces using K-RBMs

Siddhartha Chandra[1] Shailesh Kumar[2] C. V. Jawahar[1]
[1]CVIT, IIIT Hyderabad, [2]Google, Hyderabad
{siddhartha.chandra@research.,jawahar@}iiit.ac.in,shkumar@google.com

Abstract

*Understanding the nature of data is the key to building good representations. In domains such as natural images, the data comes from very complex distributions which are hard to capture. Feature learning intends to discover or best approximate these underlying distributions and use their knowledge to weed out irrelevant information, preserving most of the relevant information. Feature learning can thus be seen as a form of dimensionality reduction. In this paper, we describe a feature learning scheme for natural images. We hypothesize that image patches do not all come from the same distribution, they lie in multiple non-linear subspaces. We propose a framework that uses K Restricted Boltzmann Machines (*K-RBMS*) to learn multiple non-linear subspaces in the raw image space. Projections of the image patches into these subspaces gives us features, which we use to build image representations. Our algorithm solves the coupled problem of finding the right non-linear subspaces in the input space and associating image patches with those subspaces in an iterative EM like algorithm to minimize the overall reconstruction error. Extensive empirical results over several popular image classification datasets show that representations based on our framework outperform the traditional feature representations such as the SIFT based Bag-of-Words (BoW) and convolutional deep belief networks.*

1. Introduction

Feature extraction and modelling together dictate the overall complexity of any computer vision system. Rich features that capture most of the complexity in the input space require simpler models while simpler features require more complex models. This "law-of-conservation of complexity" in modelling has driven many efforts in feature engineering, especially, in complex domains such as computer vision where the raw input is not easily tamed by simple features. Finding semantically rich features, that capture the inherent complexity of the input data, is a challenging and necessary pre-processing step in many machine learning applications.

We propose a feature learning framework motivated by the hypothesis: data really lies in *multiple* non-linear subspaces (as opposed to a single subspace). Finding these subspaces and clustering the right data points into the right subspaces will result in the kind of features we are looking for. Our approach requires that we solve the coupled problem of non-linear projection and clustering of data points into those projections simultaneously. Clustering cannot be done in the raw input space because the data really lies in certain non-linear subspaces and the right subspaces cannot be discovered without proper groupings of the data. While most of the work in clustering and projection methods is done independently, attempts have been made to combine them [1, 17]. In this paper, we take this coupling a step forward by learning clusters and projections simultaneously. This is fundamentally different from an approach like Sparse Subspace Clustering (SSC) [5] that first learns a sparse representation (SR) of the data and then applies spectral clustering to a similarity matrix built from this SR.

We further hypothesize that a mere non-linear clustering is not the best way to understand the nature of data. Further simple clusters (concepts) might be present in each of the non-linear subspaces. An overall solution should first find multiple non-linear sub-spaces within the data and then further cluster the data within each sub-space if necessary. Once we discover the subspaces the data points (image patches) lie in, projections into these subspaces will give us the features that best represent the patches. We propose a systematic framework for a two-level clustering of input data into meaningful clusters – first level being clustering coupled with non-linear projection by Restricted Boltzmann Machines (RBMs), and the second level being simple K-means clustering in each non-linear subspace. In other words, we use K-RBMS for the first level clustering and K-means on the RBM projections for the second level clustering. We apply our framework to clustering, improving BoW and feature learning from raw image patches. We demonstrate empirically that our clustering method is comparable to the state of the art methods in terms of accuracy, and much faster. Representations based on K-RBM features

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.358

outperform traditional deep learning and SIFT based BoW representations on image classification tasks.

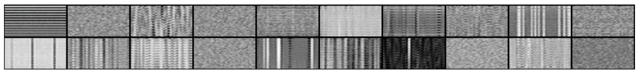


Figure 1: RBM weights (learnt by the model) representing 20 non-linear subspaces in the Pascal 2007 data. Local K-RBM features are computed by projecting image patches to the subspace they belong to, and adding the biases.

Restricted Boltzmann Machines (RBMS) [22] are undirected, energy-based graphical models that learn a non-linear subspace that the data fits to. RBMs have been used successfully to learn features for image understanding and classification [12], speech representation [18], analyze user rating of movies [21] , and better bag-of-word representation of text data [20]. Moreover, RBMS have been stacked together to learn hierarchical representations such as deep belief networks [12, 3] and convolutional deep belief networks [16] for finding semantically deeper features in complex domains such as images. Most nonlinear subspace learning algorithms [6, 2] make various assumptions about the nature of the subspaces they intend to discover. RBMS are a generic framework for learning non-linear subspaces, make no assumptions about the sub-spaces other than the size of the subspace, use a standard energy based learning algorithm, and can model subspaces of any degree of complexity via the number of hidden units making them most suitable as general purpose sub-space learning machines.

Our model learns K RBMs simultaneously. Each RBM represents a subspace in the data. Figure 1 shows 20 non-linear subspaces in VOC PASCAL 2007 data. Note the significant variation in the appearance of the subspaces. It is evident from the figure that the huge diversity in the image patches can not be captured by a single subspace. The association of a data point to an RBM depends on the reconstruction error of each RBM for that data point. Each RBM updates its weights based on all the data points associated with it. Through various learning tasks on synthetic and real data, we show the convergence properties, quality of subspaces learnt, and improvement in the accuracies of both descriptive and predictive tasks.

Note that [19] also uses RBMs for data partitioning. However, their approach is different from ours in several ways. Firstly, while we employ traditional second order (2-layer) RBMs, [19] describes an implicit mixture of RBMs which is formulated using third order RBMs. Authors in [19] introduce the cluster label (explicitly) as a hidden discrete variable in the RBM formulation describing an energy function that captures 3-way interactions among visible units, hidden units, and the cluster label variable. In our solution, the cluster label is implied by the RBM id, and the model parameters capture the usual 2-way interactions. One reason for our choice of traditional RBMs as building blocks was the availability of a great deal of research on properly training RBMs [11]. Secondly, the partition function of an RBM is intractable. By introducing the third layer [19] manages to fit the mixture of boltzmann machines without explicitly computing the partition function. We tackle the partition problem by associating samples with the RBMs that reconstruct them best (minimizing the reconstruction errors) in an EM algorithm. Since the reconstruction error is not an inherent part of the traditional RBM formulation, our framework is not a mixture model.

2. Training RBMs

RBMS are two layered, fully connected networks that have a layer of input/visible variables and a layer of hidden random variables. RBMS model a distribution over visible variables by introducing a set of stochastic features. In applications where RBMS are used for image analysis, the visible units correspond to the pixel values and the hidden units correspond to visual features.

There are three kinds of design choices in building an RBM: the objective function used, the frequency of parameter updates, and the type of visible and hidden units. RBMS are usually trained by minimizing the contrastive divergence objective (CD-1)[10] which approximates the actual RBM objective. For an RBM with I visible units $v_i, i = 1, \ldots, I$ ($v_0 = 1$ is the bias terms), J hidden units $h_j, j = 1, \ldots, J$ ($h_0 = 1$ is the bias term) and symmetric weighted connections between the visible and hidden layers denoted by $\mathbf{w} \in \mathbb{R}^{(I+1)\times(J+1)}$ (these include asymmetric forward and backward bias terms), the activation probabilities of units in one layer are computed based on the states of the opposite layer:

$$Pr(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i=0}^{I} \mathbf{w}_{ij} v_i\right) \tag{1}$$

$$Pr(v_i = 1|\mathbf{h}) = \sigma\left(\sum_{j=0}^{J} \mathbf{w}_{ij} h_j\right) \tag{2}$$

$\sigma(\cdot)$ is the sigmoid activation function. In the CD-1 forward pass (visible to hidden), we activate the hidden units h_j^+ from visible (input) unit activations v_i^+ (Eq.1). In the backward pass (hidden to visible), we recompute visible unit activations v_i^- from h_j^+ (Eq.2). Finally we compute the hidden unit activations h_j^- again from v_i^-. The weights are updated using the following rule: $\Delta w_{ij} = \eta(< v_i^+ h_j^+ > - < v_i^- h_j^- >)$ where η is the learning rate and $<\cdot>$ is defined as the mean over N examples. The reconstruction error for any sample is computed as:

$$\epsilon = \sum_{i=1}^{I} \left(v_i^+ - v_i^-\right)^2 \tag{3}$$

RBM weights are usually updated once per mini-batch. Other options are once per sample update (fully online) and corpus level update (fully batch). We found doing a full batch update gives a more reliable gradient and slightly better reconstruction compared to other strategies.

An RBM can have binary or non-binary visible and hidden units. Most RBM implementations use binary visible units. In our applications, we have used Gaussian visible units to model distributions of real valued data. The stochastic output of hidden unit (Eq.1) is always a probability which is thresholded against a random value between 0 and 1 to give a binary activation h_j. In CD-1, it is customary to use binary hidden states when the hidden units are driven by data (h_j^+) and the probabilities without sampling when the hidden units are driven by reconstructions (h_j^-). Thresholding introduces sparsity by creating an information bottleneck. We however always use the activation probabilities in place of their binary states for parameter updates. This decision was based on the desire to eliminate unnecessary randomness from our approach[1] and was supported by extensive experimentation.

3. Learning Multiple Non-Linear Subspaces using K-RBMs

Our framework uses K component RBMS. Each component RBM learns one non-linear subspace. The visible units $v_i, i = 1, \ldots, I$ correspond to an I dimensionsional visible (input) space and the hidden units $h_j, j = 1, \ldots, J$ correspond to a learnt non-linear J-dimensional subspace. For the sake of simplicity, we experiment with RBMS of the same size; all the subspaces our model learns have the same assumed dimensionality J. However, this restriction is unnecessary and we are free to learn subspaces with different assumed dimensions.

3.1. K-RBMs

The K-RBM model has K component RBMS. Each of these maps a set of sample points $\mathbf{x_n} \in \mathbb{R}^I$ to a projection in $\mathbb{R}^J$. Each component RBM has a set of symmetric weights (and asymmetric biases) $\mathbf{w}^k \in \mathbb{R}^{(I+1)\times(J+1)}$ that learns a non-linear subspace. Note that these weights include the forward and backward *bias* terms. The error of reconstuction for a sample $\mathbf{x}_n$ given by the k^{th} RBM is simply the squared Euclidean distance between the data point $\mathbf{x}_n$ and its reconstruction by the k^{th} RBM, computed using (Eq.3). We denote this error by ϵ_{kn}. The total reconstruction error ϵ_t in any iteration t is given by $\sum_{n=1}^{N} \min_k \{\epsilon_{kn}\}$

The K RBMS are trained simultaneously. During the RBM training, we associate data points with RBMs based on how well each component RBM is able to reconstruct the data points. A component RBM is trained only on the training data points associated with it. The component RBMS are given random initial weights $\mathbf{w}^k, k = 1, \ldots, K$.

[1]We use the reconstruction error as a cost function in our clustering; random thresholding introduces randomness in the projections, hence affecting the reconstruction errors.

3.2. Clustering using K-RBMs

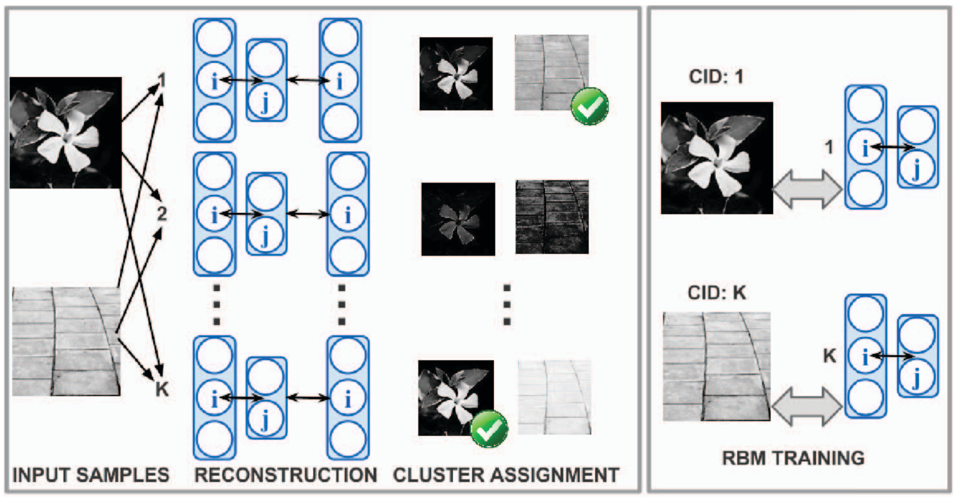
As in traditional K-means clustering, the algorithm alternates between two steps: (1) Computing association of a data point with a cluster and (2) updating the cluster parameters. In K-RBMS n^{th} data point is associated with k^{th} RBM (cluster) if its reconstruction error from that RBM is lowest compared to other RBMS, i.e. if $\epsilon_{kn} < \epsilon_{k'n} \forall k \neq k', k, k' \in \{1, \ldots, K\}$.

Once all the points are associated with one of the RBMS the weights of the RBMS are learnt in a batch update. In hard clustering the data points are partitioned into the clusters exhaustively (i.e. each data point must be associated with some cluster) and disjointly (i.e. each data point is associated with only one cluster). In contrast with K-means where the update of the cluster center is a closed form solution given the data association with clusters, in K-RBMS the weights are learnt iteratively.

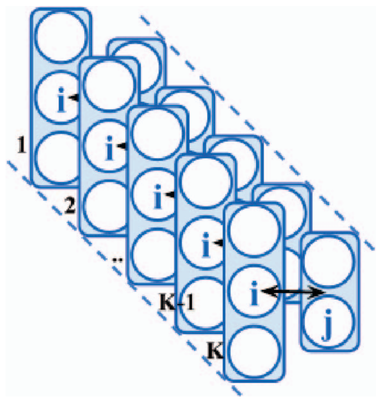
We can extend our model to incorporate soft clustering where instead of assigning a data point to only one RBM cluster, it can be assigned softly to multiple RBM clusters. The soft association of the n^{th} data point with the k^{th} cluster is computed in terms of the reconstruction error of this data point with the RBM:

$$\alpha_{nk} = \frac{\exp(-\epsilon_{kn}/T)}{\sum\limits_{k'=1}^{K} exp(-\epsilon_{k'n}/T)} \tag{4}$$

where T is the temperature parameter that is reduced over time as in simulated annealing [13]. Each sample $\mathbf{x}_n$ contributes to the training of all RBMS in proportion to its association with the RBMS. While updating weights, the association factor is also multiplied with the learning rate. A K-RBM trained using the soft approach can be seen as a set of RBMS, each of which learns a distribution of all the data but using more information from those it can represent most accurately. Each RBM can reconstruct all the points, some more accurately than the others. This is fundamentally different from the hard clustering where each component RBM learns the distribution of a subset of the data and tries to distort samples from other clusters to look like the samples that it has learnt from.



(a)



(b)

Figure 2: (a) Schematic Diagram of K-RBM training: Each input sample is fed to all component RBMs, and is assigned to the one which reconstructs it best. Each RBM is then trained using the samples assigned to it. (b) Block Diagram of K-RBMs.

3.3. Convergence and Initialization

K-RBM training seeks to learn both the associations (clusters) and the parameters (non-linear subspaces) simultaneously. There are two kinds of convergences associated with the model: the clustering convergence and the RBM learning (subspace learning) convergence. In our experiments the clustering process is said to have converged when more than 99% of the samples stop changing cluster associations. In case we require only the cluster associations, we can stop the algorithm once the clustering converges. However, the convergence of clustering just means that the points in each cluster belong to the same non-linear subspace, it does not guarantee the accuracy of the learnt subspaces. For feature learning, we require data projections in the non-linear subspaces, therefore we continue training the RBMs until the total reconstruction error stabilizes. Our experiments indicate that clustering converges far before the RBM training converges. We empirically decide the number of epochs our algorithm iterates for and we call this number $maxepoch$.

Figure 3 shows that K-RBMs significantly outperform the single RBM in terms of the final mean reconstruction error per data point. This supports our hypothesis that the input data lies in multiple simpler non-linear sub-spaces (multiple K-RBMs) and not in a single complex non-linear subspace (single RBM).

Like most EM methods, our model is sensitive to initialization. However, following the standard best RBM implementation practices (small initial weights, small learning rates, weight decay, momentum and so on) [11] ensures that this sensitivity is minimal. Further, the reconstruction errors typically converge around the same value over $maxepoch$ iterations. All our experiments were conducted once with random initialization.

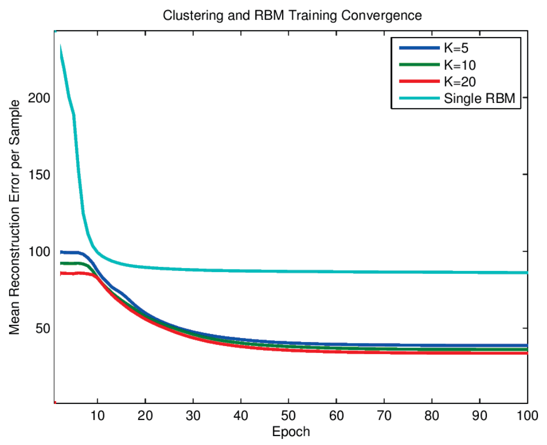


Figure 3: A plot of reconstruction errors vs epochs of training process for our experiments on the Pascal dataset (section 4.2). Reconstructions are significantly better with a K-RBM over a single RBM. For the Single RBM, we divide the mean error by 10 to bring it to scale with the others.

3.4. K-RBMs for Image Feature Learning

Traditionally, hand-crafted features like SIFT and HoG have been employed for building image representations. Such hand crafted features are often not semantically meaningful representations of images. Also they are not “learnt” but just “computed” from raw data. Recent times have seen the introduction of features that are learnt from the data.

Deep belief networks [16, 18] and convolutional networks [15] have been employed for feature learning to solve a variety of tasks. These methods are based on the hypothesis that good data representations are hierarchical and can be learnt directly from the data; these methods usually have hierarchical layered feature extractors. Although deep learning methods yield robust features, training deep networks involves making many design choices, tuning many parameters, and are often computationally challenging. We propose a feature learning scheme using K-RBMs that learns from the data like the deep networks but is simpler in terms of the overall model complexity and parameters. By doing so, we intend to take a step forward towards promoting feature extraction schemes that "learn" semantically meaningful representations of the data from the data, while keeping a check on the model complexity.

In image domains, we typically compute local features over patches in an image and then pool the local features to get global image representations (e.g. BoW). In this paper, we describe dense local K-RBM features. K-RBM features are computed by *hard* clustering patches from dense grids in images. K-RBM features are the projections of these patches in the corresponding learnt subspaces. Unlike the $128-$dimensional SIFT descriptors, the size of the K-RBM features is dictated by the number of hidden units in the component RBMs. In our experiments, we work with patches of size 12×12 pixels. Each patch can thus be represented as a $144-$dimensional sample vector. Our component RBMs have 144 visible units and 36 hidden units. Each local K-RBM feature is thus $36-$dimensional. Unlike SIFT BoW representations where we can perform K-Means clustering of all the SIFT features directly, we can't cluster K-RBM features coming from different component RBMs since they lie in different subspaces. All SIFT features lie in the same $128-$dimensional space. However each K-RBM feature lies in one of K different subspaces. Thus, we cluster the K-RBM features from each component RBM separately, get a different BoW representation for each non-linear subspace and concatenate these BoW representations to get the final BoW representation.

RBMs are generative models that learn a non-linear subspace the data lies in. RBM features are merely projections of the data onto the learnt subspace. Our K-RBM objective minimizes the error of reconstruction of the data from these projections, hence the projections are good "learnt" representations of the data. RBM feature extraction can semantically be understood as non-linear dimensionality reduction of the data. K-RBM feature extraction partitions the data across several RBMs (or subspaces). This has a two-fold advantage: (a) it gives more reliable similarity measures among data in the same subspace, (b) much of the discriminative information is encoded into the data partitions. Figure 4 shows image patches corresponding to different BoW/K-RBM clusters for SIFT and K-RBM features. SIFT space is discrete in some sense because it counts the types of edge directions. K-RBMs use a knowledge of the underlying non-linear subspaces to partition the data. In line with our second hypothesis, K-Means followed by K-RBM clustering helps achieve better partitioning of the data and consequently better vector quantization.

Both SIFT and K-RBM project image patches into non-linear sub-spaces. While SIFT introduces non-linearity by using non-linear filters followed by counting the number of directions the edges take, K-RBMs "learn" features from the data without assuming a specific class of low level features (e.g. edges assumed by SIFT). Thus while SIFT "computes" the features, K-RBMs are more adaptable to the image corpus they are applied to. While SIFT itself is a histogram of very simple artefacts (edges), K-RBMs treat each patch as an artefact.

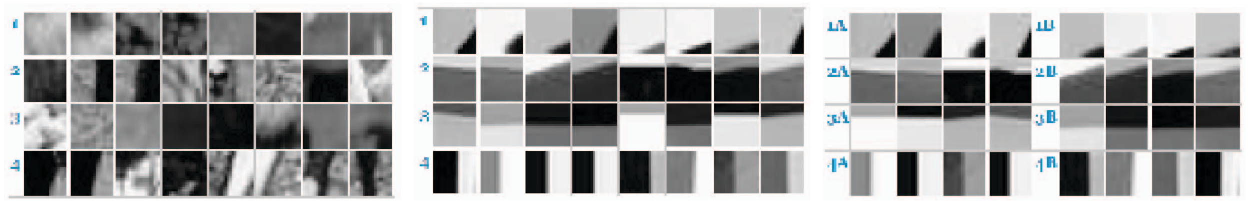
4. Applications

4.1. Application to Clustering

In this section, we demonstrate the use of K-RBMs for clustering. We compare the accuracy and speed of K-RBM clustering with the state of the art subspace clustering methods, Random Sample Consensus (RANSAC)[9] and Sparse Subspace Clustering (SSC)[5] in addition to PCA + K-means, t-SNE [23] + K-means and RBM + K-means on two synthetic datasets where we can control the nature of the sub-spaces in the data. t-SNE is a non-linear dimensionality reduction method which minimizes the divergence between distributions over pairs of points. RANSAC works by iteratively sampling a number of points randomly from the data, fitting a model to those points and rejecting outliers. SSC computes a sparse representation (SR) of the data and applies spectral clustering to a matrix obtained from the SR. These algorithms represent *decoupled* learning of projection and clustering.

The goal of these experiments is to investigate our first hypothesis i.e. *clustering* and *projection* are better done in a coupled manner than in a sequential manner. In these experiments, we compare the performance of a K-RBM with that of KMeans over data processed by a single RBM. In these comparisions, we could either (a) fix the complexity (size) of the latent non-linear subspaces by fixing the number of hidden units in each RBM or (b) fix the number of total RBM parameters in the two models (i.e. if we have a K-RBM with K components having J hidden units each, we allow the single RBM to have KJ hidden units). Here, we use the latter scheme: therefore the subspaces learnt by the two models have different dimensionalities. This was done to ensure our model had no undue advantage over the single RBM model in terms of complexity.

The synthetic datasets in table 1 were generated using



(a) K-Means on SIFT (b) K-RBM (c) K-RBM followed by K-Means

Figure 4: Sample patches corresponding to the different clusters (experiments in section 4.3). Each row in (a) and (b) represents a cluster. A row in (c) represents 2 clusters: the concatenation of these 2 clusters gives the cluster in corresponding row in (b). Patches in (a) are independent of (b) and (c). Total number of SIFT clusters in (a) was 1000, K_1 for (b) was 40, K_2 in (c) was 50.

the RANSAC demo code at *www.vision.jhu.edu/downloads*. Dataset $D1$ comprises of 500 points drawn from 5 randomly generated subspaces having orthogonal basis vectors, 100 points from each subspace. For all the points, the dimension of the raw feature space is 144 while the assumed intrinsic dimensionality is 36. $D1$ also contains added Gaussian noise. Dataset $D2$ consists of 1000 points drawn from 5 randomly generated subspaces with non-orthogonal basis vectors, 200 points are drawn from each subspace. $D2$ is thus harder, and bigger than $D1$.

Method	Dataset D1		Dataset D2	
	Time(s)	Error	Time(s)	Error
K-means	0.68	27.4%	3.92	26.2%
PCA	0.37	27.4%	1.58	25.3%
t-SNE	11.68	11.3%	18.20	9.8%
RBM	3.29	26.6%	4.56	19.4%
RANSAC	134.80	66.6%	612.43	38.4%
SSC	365.29	0%	760.48	0%
K-RBM	0.46	0%	4.58	0%

Table 1: Running Time and Misclassification Errors of various methods on synthetic $D1$ and $D2$ datasets. K-RBMs are comparable to SSC in terms of accuracy, but practically much faster.

The clustering results are reported in table 1 in terms of misclassification error and the running time of these algorithms. We chose 36 principal components for PCA. All the RBMs had 144 Gaussian visible units. Each RBM in the K-RBM had 36 binary hidden units while the single RBM had 180. It can be seen that K-RBM is comparable to SSC in terms of quality metrics, but orders of magnitude faster as well. Due to the time complexity of RANSAC and SSC it is impractical to train these models on huge datasets without serious sampling. Kindly note that SSC uses three kinds of spectral clusterings, and thus gives three error rates. In table 1 we report the least of the three errors. Typically SSC uses top K' connections (where K' is set to the size of the assumed subspace) in the similarity graph to build the adjacency matrix. We observed that using all the connections in the similarity graph gives better performance.

4.2. K-RBMs for Visual Bag-of-Words

These experiments investigate the second hypothesis: multi-variate real-valued data generally lies in multiple non-linear subspaces (e.g. as learnt by K-RBMS) and that there are further potential clusters within each of the sub-spaces. This points to a two stage clustering of data: first clustering coupled with non-linear projection (e.g. K-RBM) followed by further sub-clustering within each first level cluster. The second goal of these experiments is to propose an alternative to the traditional bag-of-words representations used ubiquitously in computer vision applications.

We experiment with 3 datasets here: PASCAL VOC 2007 [7], 15 Scene Categories [14] and Caltech 101 [8]. PASCAL VOC 2007 data has a total of 5011 training images and 2944 testing images in 20 classes. The 15 Scene Categories dataset has 4485 images in all split over 15 different scene categories. As in [14], we choose 100 random images per category for training and the rest for testing. We repeated the experiments 5 times and report the average accuracy. Caltech 101 has 9146 images, split among 101 distinct object categories. In these experiments, we sampled 30 random images for training from each of the 101 categories, getting a total of 3030 training images; the rest of the images were treated as testing images; however, as in [14], we limited the number of testing images per category to 50. These experiments were repeated 5 times with random subsampling and the mean classification accuracies over the five experiments are reported.

$128-$ dimensional SIFT features on all datasets are computed using a scale of 12 and a shift of 6. For the baseline BoW representation, we cluster SIFT features coming from 10 random images per class into 1000 visual words using standard K-means. We use a 2nd level spatial pyramid [14] to get the BoW image representations. For Scene 15 and

Caltech 101 datasets, we trained a 1-vs-rest classifier for each class and the test image was assigned the label of the classifier with the highest score. For PASCAL data, we train a 1-vs-rest classifier per class and report the mean Average Precision per class.

In our approach, we create the 1000 clusters in a different way. We train a K-RBM with K_1 components over SIFT points. The RBMs use $128-$dimensional Gaussan visible units. These are reduced to $20-$dimensional real valued hidden units. The model here is that the feature points in the original 128-dimensional SIFT space reside in K_1 non-linear 20-dimensional subspaces. Once trained, the K-RBM partitions the SIFT data points into K_1 exhaustive and non-overlapping (we used hard clustering) subsets. We further clustered each of the K_1 subsets *in the trasnformed 20-dimensional space* into K_2 clusters using simple K-means clustering. This is in-line with our hypothesis that within each sub-space there might be multiple clusters. To keep the total number of clusters compatible with the baseline $K = 1000$, we chose K_1 and K_2 such that their product is 1000. The K_1 and K_2 we report in table 3 for different datasets were learnt by using a validation set. Hence, each SIFT descriptor is first mapped to one of the K_1 RBM clusters and then its transformed representation is further mapped to one of the K_2 clusters giving $K = 1000$ final cluster BoW representation for the images. Here too, we use the 2nd level spatial pyramid for the BoW image representation. The same SVM classifier and evaluation methodology was used for this new image representation.

Method	K1	K2	Mean AP
Baseline BoW (K-Means)	-	1000	52.84%
K-RBM BoW	5	200	55.10%
K-RBM BoW	8	125	56.40%
K-RBM BoW	10	100	55.35%
K-RBM BoW	20	50	54.85%

Table 2: Learning Bow by two level clustering: mean classification AP on VOC Pascal 2007

Overall mean classification average precision (AP) on various code-books on Pascal 2007 is shown in Table 2. For K_1 = 8, K_2 = 125, mean AP is highest, significantly higher than traditional BoW. Thus learning clusters in a two-stage process: non-linear subspaces followed by clustering within each subspace improves the quality of the clustering. Also, the right balance has to be struck on how the complexity is distributed between the two stages. The size of projected RBM spaces (in our case 20-dimensional) is also a factor in the overall complexity of the representation. These need to be empirically determined for any dataset.

Results on the 3 datasets are listed in table 3. A 2 level clustering of SIFT features yields better BoW representation. This is indicated by better classification performance, and low mean quantization error on the three datasets. The mean quantization error is the mean euclidean distance between the SIFT/K-RBM features and the correspoding cluster centers, divided by the length of the feature vector. Note that we normalize the SIFT vectors to contain all values between 0 and 1 (as for K-RBM features) to ensure fair comparision. Smaller quantization errors indicate better understanding of the feature space.

4.3. Feature learning using K-RBMs

In this section, we compare the classification performance of K-RBM features with that of SIFT and Convolutional Deep Belief Networks (CDBN) [16] on Caltech 101 and VOC Pascal 2007 datasets. Note that CDBN classification results are unavailable on VOC 2007. Hierarchical methods such as CDBN work well on Caltech 101 which has object-centered and cropped images, conducive to hierarchical learning of artefacts. Pascal data has huge variation in the scale, position and orientation of objects, even has multiple objects per image. Dense local K-RBM features work well even on Pascal because they exploit the invariance of BoW representations.

SIFT and K-RBM features are computed over a dense grid of 12×12 patches with a shift of 6. The component RBMs have 144 Gaussian visible units and 36 real hidden units. We also use a 2nd level spatial pyramid [14] to get the BoW Image representations. We fix the BoW vocabulary size to 1000 as in section 3. We use a linear pegasos SVM classifier with the χ^2 kernel map for classification [24]. For Caltech 101, as in section 4.2, we used 30 random images per class for training and use the rest for testing, limiting the test images to 50 per category. We repeat the experiments 5 times and report the mean classification accuracy. The classification schemes for the two datasets remain the same as in section 4.2. K_1, K_2 are learnt using a validation set. The results are reported in tables 5 and 4 along with State of the Art results based on SIFT-Fisher vectors as in [4]. Features learnt using K-RBMs significantly outperform the SIFT and CDBN features. Low level hand-crafted features work well because of scale, distortion invariant pooling schemes like BoW and powerful SVM classifiers. Deep learning methods work because of semantically meaningful features. Our approach combines rich features with powerful BoW representation and SVM classifiers and thus outperforms the two competing classes of methods.

5. Conclusions

We developed a framework that uses K RBMs to learn rich, complex, and more meaningful features. K-RBM features are projections of the input image patches onto the non-linear subspaces they lie in. Compared to clustering methods like SSC and RANSAC, K-RBMs is faster and

DATASET	BASELINE BOW		K-RBM BOW	
	PERFORMANCE	MEAN Q.E.	PERFORMANCE	MEAN Q.E.
VOC PASCAL 2007	52.84%	0.7678	$\mathbf{56.40}\%$ ($K_1 = 8, K_2 = 125$)	0.1620
15 Scene	$80.50 \pm 0.5\%$	0.5635	$\mathbf{85.75 \pm 0.6}\%$ ($K_1 = 20, K_2 = 50$)	0.0840
Caltech 101	$68.34 \pm 1.3\%$	0.6420	$\mathbf{72.80 \pm 1.1}\%$ ($K_1 = 8, K_2 = 125$)	0.1365

Table 3: Classification Performance on VOC Pascal 2007, 15 Scene Categories and Caltech 101

Method	Accuracy
SIFT Features	$68.3 \pm 1.3\%$
CDBN (layers 1+2)	$65.4 \pm 0.5\%$
K-RBM Features ($K_1 = 20$)	$\mathbf{74.2 \pm 1.7}\%$
STATE OF ART [4]	$\mathbf{77.8 \pm 0.6}\%$

Table 4: Caltech 101

Method	Mean AP
SIFT Features	52.84%
K-RBM Features ($K_1 = 20$)	$\mathbf{58.40}\%$
STATE OF ART [4]	$\mathbf{61.69}\%$

Table 5: VOC Pascal 2007

Classification Performance of K-RBM Features on Caltech 101 and VOC Pascal 2007 Datasets.

more accurate. The two stage feature learning where first stage uses K-RBMs followed by K-Means for BoW helps improve the overall image representation. K-RBM+K-means features outperform SIFT+Kmeans and CDBN features for image classification. Complex input domains such as images where input lies in multiple non-linear subspaces, the K-RBM approach provides a general, robust, and fast feature learning framework compared to other methods that are either too computationally intensive or make lots of assumptions about the nature of the data or need a lot of parameter tuning. So far we have worked with an unsupervised version of K-RBM but this can be extended to supervised version where a separate K-RBM can be learnt for each class.

References

[1] M. S. Baghshah and S. B. Shouraki. Semi-supervised metric learning using pairwise constraints. In *IJCAI*, 2009.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. Montral, and M. Qubec. Greedy layer-wise training of deep networks. In *NIPS*, 2007.

[4] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[5] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.

[6] E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In *NIPS*, 2011.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *WGMBV*, 2004.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus. *Commun. ACM*, 1981.

[10] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2000.

[11] G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 2006.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 1983.

[14] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[16] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. ICML, 2009.

[17] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, 2010.

[18] A. Mohamed, G. Dahl, and G. Hinton. Deep belief networks for phone recognition. In *ICASSP*, 2011.

[19] V. Nair and G. E. Hinton. Implicit mixtures of restricted boltzmann machines. In *NIPS*, 2008.

[20] R. Salakhutdinov and G. Hinton. Replicated softmax: an undirected topic model. In *In NIPS*, 2010.

[21] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2004.

[22] P. Smolensky. In *Parallel Distributed Processing: Volume 1: Foundations*. 1987.

[23] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. In *JMLR*, 2008.

[24] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.