On Compact Codes for Spatially Pooled Features

Yangqing Jia JIAYQ@EECS.BERKELEY.EDU
Oriol Vinyals VINYALS@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
UC Berkeley EECS, Berkeley, CA 94704 USA

Abstract

Feature encoding with an overcomplete dictionary has demonstrated good performance in many applications, especially computer vision. In this paper we analyze the classification accuracy with respect to dictionary size by linking the encoding stage to kernel methods and Nyström sampling, and obtain useful bounds on accuracy as a function of size. The Nyström method also inspires us to revisit dictionary learning from local patches, and we propose to learn the dictionary in an end-to-end fashion taking into account pooling, a common computational layer in vision. We validate our contribution by showing how the derived bounds are able to explain the observed behavior of multiple datasets, and show that the pooling aware method efficiently reduces the dictionary size by a factor of two for a given accuracy.

1. Introduction

In the recent decade, overcompletely encoded features have been shown to provide state-of-the-art performance on various applications. In computer vision, locally encoded and spatially pooled feature extraction pipelines work particularly well for image classification. Such pipelines usually start from densely extracted local image patches (either normalized raw pixel values or hand-crafted descriptors such as SIFT or HOG), and perform dictionary learning to obtain a dictionary of codes (filters). The patches are then encoded into an over-complete representation using various algorithms such as sparse coding (Olshausen & Field, 1997; Wang et al., 2010) or simple inner product with a non-linear post-processing (Coates & Ng, 2011; Krizhevsky et al., 2012). After encoding, spatial pooling with average or max operations are carried out to form a global image representation (Yang et al., 2009; Boureau et al., 2010). The encoding and pooling pipeline may be stacked in a deep structure to produce a final feature vector, which is then used to predict the labels for the images usually via a linear classifier.

There is an abundance of literature on single-layered networks for unsupervised feature encoding. Various dictionary learning methods have been proposed to find a set of basis that reconstructs local image patches or descriptors well (Mairal et al., 2010; Coates & Ng, 2011), and encoding methods have been proposed to map the original data to a high-dimensional space that emphasizes certain properties, such as sparsity (Olshausen & Field, 1997; Yang et al., 2009; 2010) or locality (Wang et al., 2010). Among these, a particularly interesting finding in the literature (Coates et al., 2011; Rigamonti et al., 2011; Coates & Ng, 2011; Saxe et al., 2011) is that very simple patch-based algorithms like K-means or even random selection, combined with feed-forward encoding methods with a naive nonlinearity, produces state-of-the-art performance on various datasets. Explanation of such phenomenon often focuses on the local image patch statistics, such as the frequency selectivity of random samples (Saxe et al., 2011), but does not offer an asymptotic theory on the dictionary learning behavior.

In addition, a potential problem with local patch-based dictionary learning methods for vision is that they may learn redundant features when we consider the pooling stage, as two codes that are uncorrelated may become highly correlated after pooling due to the introduction of spatial invariance. While using a larger dictionary almost always alleviates this problem, in practice we often want the dictionary to have a limited number of codes, as feature computation has become the dominant factor in the state-of-the-art image classification pipelines, even with purely feed-forward

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

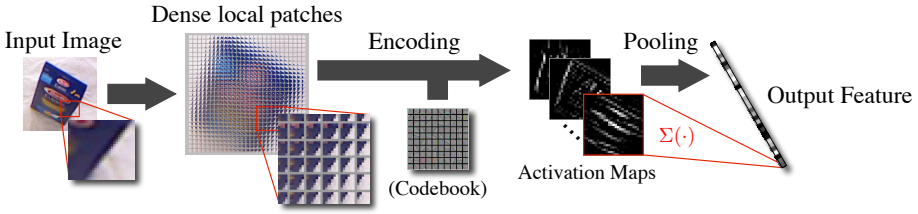


Figure 1. The feature extraction pipeline, composed of dense local patch extraction, encoding, and pooling. Illustrated is the average pooling over the whole image for simplicity, and in practice the pooling can be carried out over finer grids of the image as well as with different operations (such as max).

methods (Coates & Ng, 2011) or speedup algorithms (Wang et al., 2010). A reasonably sized dictionary also helps to more easily learn further tasks that depends on the encoded features; this is especially true when we have more than one coding-pooling stage such as stacked deep networks, or when one applies more complex pooling stages such as second-order pooling (Carreira et al., 2012), as a large encoding output would immediately drive up the number of parameters in the next layer. Thus, it would be beneficial to design a dictionary learning algorithm that takes pooling into consideration and learns a compact dictionary.

In this paper we address the above questions by providing a novel view of the single-layer feature encoding based on kernel methods and Nyström sampling. In particular, we view the coding of a data point with a local representation based on a dictionary with fewer elements than the number of total data points as a proxy to approximate the actual function that would compute pair-wise similarity to all data points (often too many to compute in practice), where the approximation is done by a random or K-means based selection of data points.

Furthermore, since bounds are known on the approximation power of Nyström sampling as a function of how many samples we consider (i.e. the dictionary size), we derive bounds on the approximation of the exact (but expensive to compute) kernel matrix, and use it as a proxy to predict accuracy as a function of the dictionary size, which has been observed to increase but also to saturate as we increase its size. The Nyström view helps explain the behavior of feature learning with increasing dictionary sizes, and justifies the use of simple algorithms such as K-means in dictionary learning (Kumar et al., 2012). It may also help justify the need to stack more layers (often referred to as deep learning), as flat models are guaranteed to saturate as we add more complexity.

We then show that the empirical findings in Nyström sampling view lead to a particularly simple yet effective algorithm that is analogous to the patch-based K-means method for dictionary learning, but that takes into account the additional redundancy introduced in the pooling stage. Specifically, we present a two-stage clustering method to learn a dictionary that identifies post-pooling invariant features. The resulting dictionary yields a higher classification accuracy while introducing no additional computation overhead during classifier training and testing time.

We believe that the Nyström sampling based explanation provides a new view on the unsupervised visual feature extraction pipeline, and may inspire new algorithms to perform efficient unsupervised feature learning in a deeper structure, as we have demonstrated for learning a pooling invariant dictionary.

2. Background

Figure 1 illustrates the feature extraction pipeline that is composed of encoding dense local patches and pooling encoded features. This pipeline is architecturally very similar to the basic structure of convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012), but instead relies on unsupervised learning (rather than fine tuning) of the network parameters. Specifically, starting with an input image $\mathbf{I}$, we formally define the encoding and pooling stages as follows.

(1) Coding. In the coding step, we extract local image patches[1], and encode each patch to c activation values based on a dictionary of size c (learned via a separate dictionary learning step). These activations are typically binary (in the case of vector quantization) or continuous (in the case of e.g. sparse coding), and it is generally believed that having an over-complete ($c >$ the dimension of patches) dictionary while keeping the activations sparse helps classification, especially when linear classifiers are used in the later steps.

We will mainly focus on what we call the decoupled

[1]Although we use the term "patches", the pipeline works with local descriptors such as SIFT as well.

encoding methods, in which the activation of one code does not rely on other codes, such as threshold encoding (Coates & Ng, 2011), which computes the inner product between a local patch $\mathbf{x}$ and each code, with a fixed threshold parameter α: $\mathbf{c}(\mathbf{x}) = \max\{0, \mathbf{x}^\top \mathbf{D} - \alpha\}$ where $\mathbf{D} \in \mathbb{R}^{d \times c}$ is the dictionary. Such methods have been increasingly popular mainly for their efficiency over coupled encoding methods such as sparse coding, for which a joint optimization needs to be carried out. Their use in several deep models, e.g. (Krizhevsky et al., 2012), also suggests that such simple non-linearity may suffice to learn a good classifier in the later stages.

(2) Learning the dictionary: Recently, it has been found that relatively simple dictionary learning and encoding approaches lead to surprisingly good performance (Coates et al., 2011; Saxe et al., 2011). For example, to learn a dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_c]$ of size K from a set of local patches $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ each reshaped as a vector of pixel values, one could simply adopt the K-means algorithm, which aims to minimize the squared distance between each patch and its nearest code: $\min_{\mathbf{D}} \sum_{i=1}^{N} \min_j \|\mathbf{x}_i - \mathbf{d}_j\|_2^2$. We refer to (Coates et al., 2011) for a detailed comparison about different dictionary learning and encoding algorithms.

(3) Pooling. Since the coding results are highly over-complete and highly redundant, the pooling layer aggregates the activations over a spatial region of the image to obtain a c dimensional vector, where each dimension of the pooled feature is obtained by taking the output of the corresponding code in the given spatial region (also called receptive field in the literature) and performing a predefined operator (usually average or max). Figure 1 shows an example when average pooling is carried out over the whole image. In practice we may define multiple spatial regions per image (such as a regular grid or a spatial pyramid), and the global representation for the image will then be a vector of size c times the number of spatial regions.

3. Relationship Between Random Dictionaries and Nyström Sampling

An important empirical observation was made in (Coates & Ng, 2011) regarding the importance of the dictionary learned from the data and the encoding technique. The authors observed that, with a rather simple coding scheme and dictionary learning, results were in most cases comparable to the widely used but more computationally expensive sparse coding technique. It was particularly noteworthy that selecting random dictionaries yielded close to state-of-the-art results. Further work on this domain (Denil & de Freitas, 2012) suggests that the encoding technique used is a proxy to solving sparse coding (but in a simple and faster fashion).

The fact that random dictionaries perform well when operating with large codebook sizes poses interesting questions such as how the dictionary size affects performance, and why sufficiently large random dictionaries match learned dictionaries. In addition, even though the size of the dictionary (or codebook) is important, the accuracy seems to saturate, which is a phenomenon that was empirically verified in many tasks, and for which we now give a theoretical interpretation by linking random dictionaries with Nyström sampling.

3.1. The Nyström Sampling View

Nyström sampling has been proposed as an efficient way to approximate large PSD matrices (such as kernel matrices) by sampling columns of the matrix. Specifically, let $\mathbf{K}$ be an $N \times N$ matrix, the Nyström method defines an approximation as $\mathbf{K}' = \mathbf{E}\mathbf{W}^{+}\mathbf{E}^\top$, where $\mathbf{E}$ is a $N \times c$ matrix with the c columns randomly sampled from those of $\mathbf{K}$, and $\mathbf{W}$ is the square $c \times c$ matrix formed by picking the same c columns and rows from $\mathbf{K}$. Such a sampling perspective have been shown to be very effective in kernel machines (Zhang et al., 2008; Cortes et al., 2010; Kumar et al., 2012).

We consider forming a dictionary by sampling our training set (although, as discussed below, better techniques exist that lead to further gains in performance). To encode a new data point $\mathbf{x} \in \mathbb{R}^d$, we apply a (generally non-linear) coding function $\mathbf{c}$ so that $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^c$. The standard classification pipeline considers $\mathbf{c}(\mathbf{x})$ as the new feature space, and typically uses a linear classifier on this space. In this section, we consider the threshold encoding function as in (Coates & Ng, 2011), $\mathbf{c}(\mathbf{x}) = \max(0, \mathbf{x}^\top \mathbf{D} - \alpha)$, but the derivations are valid for other different coding schemes.

In the ideal case (infinite computation and memory), we encode each sample $\mathbf{x}$ using the whole training set $\mathbf{X} \in \mathbb{R}^{d \times N}$, which can be seen as the best local coding of the training set $\mathbf{X}$, to the extent that overfitting is handled by the classification algorithm. In fact, larger dictionary sizes yield better performance assuming the linear classifier is well regularized, as it can be seen as a way to do manifold learning (Wang et al., 2010). We define the new features in this high-dimensional coded space as $\mathbf{C} = \max(0, \mathbf{X}^\top \mathbf{X} - \alpha)$, where the i-th row of $\mathbf{C}$ corresponds to coding the i-th sample $\mathbf{c}(\mathbf{x}_i)$. The linear kernel function between samples i and j is $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{c}(\mathbf{x}_i)^\top \mathbf{c}(\mathbf{x}_j)$. Thus, performing linear

classification on the coded features effectively uses the kernel matrix $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$.

In the conventional context of Nyström sampling for kernels, one randomly samples a subset of the columns of $\mathbf{K}$ and then replaces the original matrix $\mathbf{K}$ with a low-rank approximation $\hat{\mathbf{K}}$. However, in our problem, naively applying Nyström sampling to the matrix $\mathbf{K}$ does not save any computation, as every column of $\mathbf{K}$ requires to encode the corresponding feature with the large dictionary of all N samples. However, if we approximate the matrix $\mathbf{C}$ with Nyström sampling to obtain $\mathbf{C}' \approx \mathbf{C}$, we would get an efficient approximation of the kernel matrix as $\mathbf{K}' \approx \mathbf{K}$:

$$\mathbf{C}' = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top, \text{ and} \quad (1)$$

$$\mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^\top, \quad (2)$$

where the first equation comes from applying Nyström sampling to $\mathbf{C}$, $\mathbf{E}$ is a random subsample of the columns of $\mathbf{C}$, and $\mathbf{W}$ the corresponding square matrix with the same random subsample of both columns and rows of $\mathbf{C}$.

We note that in the traditional coding scheme proposed in (Coates & Ng, 2011), if the dictionary is taken randomly then $\mathbf{K}_{coding} = \mathbf{E}\mathbf{E}^\top$, and by applying Nyström sampling to $\mathbf{C}$ we obtain almost the same kernel, where the matrix $\mathbf{\Lambda}$ acts as an additional Mahalanobis metric on the coded space. Adding the term $\mathbf{\Lambda}$ seemed to help in some cases, when the dictionary size is small (for example, in the CIFAR10 dataset, classification performance was improved by about 0.5% when $c < 500$.). We refer to the supplementary material to discuss the effect of $\mathbf{\Lambda}$ and how to efficiently find it without explicitly computing the original $N \times N$ matrix.

3.2. Error Bounds on the Approximation

Many existing analyses have computed bounds on the error made in estimating $\mathbf{C}$ by $\mathbf{C}'$ by sampling c columns, such as (Talwalkar & Rostamizadeh, 2010; Kumar et al., 2012), but not between $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$ and $\mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top$, which we aim to analyze in this section. The bound we start with is (Kumar et al., 2012):

$$||\mathbf{C} - \mathbf{C}'||_F \leq ||\mathbf{C} - \mathbf{C}_k||_F + \epsilon \max(n\mathbf{C}_{ii}), \quad (3)$$

valid if $c \geq 64k/\epsilon^4$ (c is the number of columns that we sample from $\mathbf{C}$ to form $\mathbf{E}$, i.e. the codebook size), where k is the sufficient rank to estimate the structure of $\mathbf{C}$, and $\mathbf{C}_k$ is the optimal rank k approximation (given by Singular Value Decomposition (SVD), which we cannot compute in practice).

Fixing k to the value that retains enough energy from $\mathbf{C}$, we get a bound that gives a minimum ϵ to plug in Eqn. 3 for every c (sample dictionary size). This gives us a useful bound of the form $\epsilon \geq \hat{M}\,(1/c)^{1/4}$ for some constant $\hat{M}$ (that depends on k). Hence:

$$||\mathbf{C} - \mathbf{C}'||_F \leq O + M\,(1/c)^{1/4}, \quad (4)$$

where O and M constants that are dataset specific.

However, having bounded the error $\mathbf{C}$ is not yet sufficient to establish how the code size will affect the classifier performance. In particular, it is not clear how the error on $\mathbf{C}$ affect the error on the kernel matrix $\mathbf{K}$. Similarly, having a kernel matrix of different quality will affect classification performance. Recent work (Cortes et al., 2010) proves a linear relationship between kernel matrix degradation and classification accuracy. Furthermore, in the supplementary material, we provide a proof that shows the degradation of $\mathbf{K}$ is also proportional to the degradation of $\mathbf{C}$. Hence, the error bound on $\mathbf{K}'$ is of the same form as the one we obtained for $\mathbf{C}$:

$$||\mathbf{K} - \mathbf{K}'||_F \leq O' + M'\,(1/c)^{1/4}. \quad (5)$$

We note that the bound above also applies to the case when further steps, such as pooling, is carried out after coding, provided that such steps produce output feature dimensions that have a one-to-one correspondence with the dictionary entries. Pooling over multiple spatial regions does not change the analysis as it could be deemed as concatenating multiple kernel matrices for the data.

3.3. Evaluating Bounds

We empirically evaluate the bound on the kernel matrix, used as a proxy to model classification accuracy, which is the measure of interest. To estimate the constants in the bounds, we do interpolation of the observed accuracy using the first three samples of accuracy versus codebook size, which is of practical interest: one may want to quickly run a new dataset through the pipeline with small dictionary sizes, and then quickly estimate what the accuracy would be when running a full experiment with a much larger dictionary (which would take much longer to run) with our formulation. We always performed Nyström sampling schemes by doing K-means instead of random selection (although the accuracy between both methods does not change too much when c is sufficiently large).

In Figure 2 we plot the accuracy (on both train and test sets) on four datasets: CIFAR-10 and STL from vision, and WSJ and TIMIT from speech. For each dataset we used the first three samples to determine

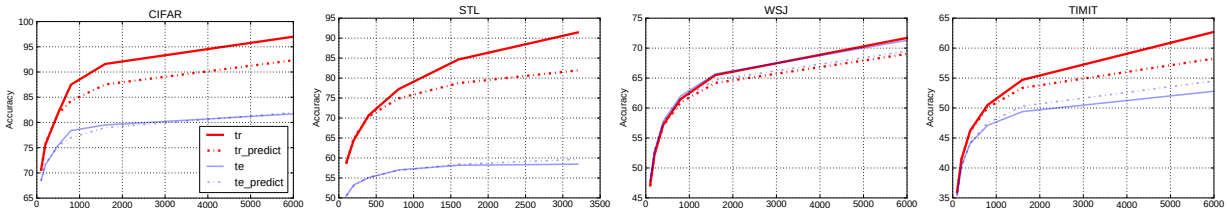


Figure 2. The actual training and testing accuracy (solid) and the predicted accuracy using our bound (dashed), on four datasets: CIFAR, STL, WSJ and TIMIT from left to right.

the constants given in the bound. One may practically favor this approach to evaluate performance, as small dictionary sizes are fast to try while large dictionary sizes are of interest. The bound is designed to predict training accuracy (Cortes et al., 2010), but we also do regression on testing accuracy for completeness. We note that testing accuracy will in general also be affected by the generalization gap, which is not captured by the bound analysis.

The results show that in all cases, the red dashed line is a lower bound of the training actual accuracy, and follows the shape of the empirical accuracy, predicting its saturation. In the testing case, our model is slightly optimistic when overfitting exists (e.g. STL and TIMIT), but correctly predicts the trend with respect to the number of dictionary entries.

The implication of linking Nyström sampling theory to current learning pipelines has several immediate consequences: first, it clarifies why random sampling or K-means produce very reasonable dictionaries that are able to perform well in terms of classification accuracy (Zhang et al., 2008; Coates et al., 2011; Kumar et al., 2012); more importantly, due to known bounds such as the one derived in this section, we can model how the codebook size will affect performance by running a few experiments with smaller codebook sizes, and extrapolating to larger (and more computationally expensive to compute) codebook sizes by means of Eq. 5, thus predicting accuracies before running potentially long jobs.

4. Pooling Aware Dictionary Learning

The Nyström sampling view suggests that one could find a better subset of a large (potentially infinite) dictionary to obtain more informative features. In addition, existing work suggests that this could be often done in an efficient way with methods such as clustering. However, current clustering algorithms for dictionary learning (Coates et al., 2011; Coates & Ng, 2011) only apply to the local coding step, and do not consider the pooling effect. In this section, we show that by explicitly taking into account the whole pipeline

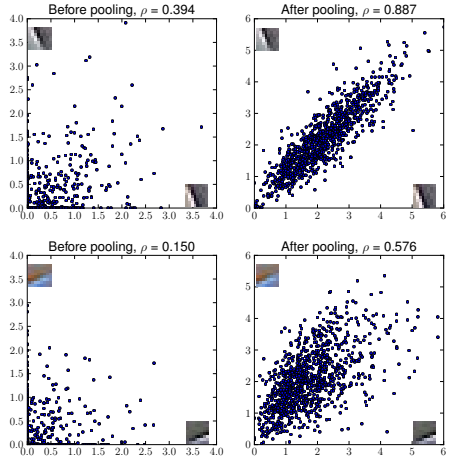


Figure 3. Two codes learned from a patch-based K-means algorithm that produce lowly correlated patch-based responses (left), but highly correlated responses after pooling (right). Such phenomenon may root from various causes, such as codes with translational difference (above) and color difference (below).

shown in Figure 1 to include both local coding and pooling when learning the dictionary, one gets a much more compact feature representation.

Figure 3 shows two examples why pooling-aware dictionary learning may be necessary, as local patch-based dictionary learning algorithms often yield similar filters with small translations. Such filters, even when uncorrelated on the patch level, produce highly correlated responses when pooled over a certain spatial region, leading to redundancy in the feature representation.

Observing the effectiveness of clustering methods in patch-based dictionary learning, we propose to learn a final dictionary of size K in two stages: first, we adopt the K-means algorithm to learn a more over-complete starting dictionary of size M ($M >> c$) on patches, effectively "overshooting" the dictionary we aim to obtain. We then perform encoding and pooling using the dictionary, and learn the final smaller dictionary of size c from the statistics of the M-dimensional pooled features.

4.1. Post-Pooling Feature Selection

The first step of our algorithm is identical to the patch-based K-means algorithm with a dictionary size M. After this, we can sample a set of image super-patches of the same size as the pooling regions, and obtain the M dimensional pooled features from them. Randomly sampling a large number of pooled features in this way allows us to analyze the pairwise similarities between the codes in the starting dictionary in a post-pooling fashion. We would then like to find a c-dimensional, lower dimensional subspace that best represents the M pooled features.

If we simply would like to find a low-dimensional representation from the M-dimensional pooled features, one would naturally choose SVD to find the K most significant projections of the covariance matrix. With a little abuse of terminology and denoting the matrix of randomly selected pooled feature as $\mathbf{X}$ where each column is a feature vector, the SVD is carried out as

$$\mathbf{X} \approx \mathbf{U}_c \mathbf{\Lambda}_c \mathbf{V}_c^\top, \tag{6}$$

where $\mathbf{R}$ is the covariance matrix computed using the random sample of pooled features, the $M \times c$ matrix U_c contains the left singular vectors, and the $c \times c$ diagonal matrix $\mathbf{\Lambda}_c$ contains the corresponding singular values. The low-dimensional features are then computed as $\mathbf{x}_c = \mathbf{U}_c^\top \mathbf{x}$.

While the "oracle" low-dimensional representation by SVD guarantees the best c-dimensional approximation, it does not meet our goal since the dictionary size is not reduced, as SVD almost always yields non-zero coefficients for all the dimensions. Linearly combining the dictionary entries does not work either due to the nonlinear nature of the encoding algorithm. In our case, we would need the coefficients of only a subset of the features to be non-zero, so that a minimum number of filters need to be applied during testing time. Various machine learning algorithms aim to solve this, most notably structured sparse PCA (Jenatton et al., 2010). However, these methods often requires a structured sparsity term to be applied during learning, making the training time-consuming and difficult to scale up.

Based on the analysis of the last section, the problem above could again be viewed as a Nyström sampling problem by subsampling the rows of the matrix $\mathbf{X}$ (corresponding to selecting codes from the large dictionary). Empirical results from the Nyström sampling then suggests the use of clustering algorithms to solve this. Thus, we resort to a simpler K-centroids method.

Specifically, we use affinity propagation (Frey & Dueck, 2007), which is a version of the K-centroids algorithm, to select exemplars from the existing dictionary. Intuitively, codes that produce redundant pooled output (such as translated versions of the same code) would have high similarity between them, and only one exemplar would be chosen by the algorithm. We briefly explain the affinity propagation procedure here: it finds exemplars from a set of candidates where pairwise similarity $s(i,j)$ $(1 \le i,j \le M)$ can be computed. It iteratively updates two terms, the "responsibility" $r(i,j)$ and the "availability" $a(i,j)$ via a message passing method following such rules (Frey & Dueck, 2007):

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\} \tag{7}$$

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum\nolimits_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\} \quad (\text{if } i \neq k) \tag{8}$$

$$a(k,k) \leftarrow \sum\nolimits_{i' \neq k} \max\{0, r(i',k)\} \tag{9}$$

Upon convergence, the centroid that represents any candidate i is given by $\arg\max_k (a(i,k) + r(i,k))$, and the set of centroids $\mathcal{S}$ is obtained by

$$\mathcal{S} = \{k | \exists i, k \text{ s.t. } k = \arg\max_{k'} (a(i,k') + r(i,k'))\} \tag{10}$$

And we refer to (Frey & Dueck, 2007) for details about the nature of such message passing algorithms. The similarity between two pooled dimensions (which correspond to two codes in the starting dictionary) i and code j, as in Eqn. (7)-(9), is computed as

$$s(i,j) = \frac{2R_{ij}}{\sqrt{R_{ii}R_{jj}}} - 2. \tag{11}$$

Note that this is equivalent to the negative Euclidean distance between the coded output i and the coded output j when the outputs are normalized to have zero mean and standard deviation 1. We note that related work such as (Coates et al., 2012) adopt a similar approach by max-pooling the outputs of similar codes to generate next-layer features in a deep fashion. Our method shares the same merit while focusing on model compression by bounding the computation time in a single layer.

Clustering algorithms has shown to be very effective in the context of Nyström sampling (Kumar et al., 2012), and are often highly parallelizeable, easily being scaled up by simply distributing the data over multiple machines. This allows us to maintain the efficiency of dictionary learning. Using a large, overshooting starting dictionary allows us to preserve most information from the patch-level, and the second step prunes away the redundancy due to pooling. Note that the large dictionary is only used during the feature learning time -

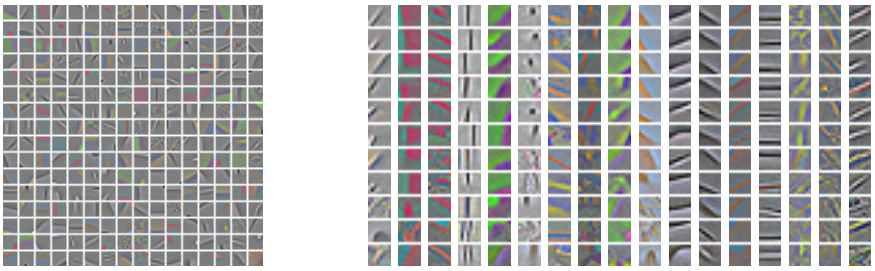
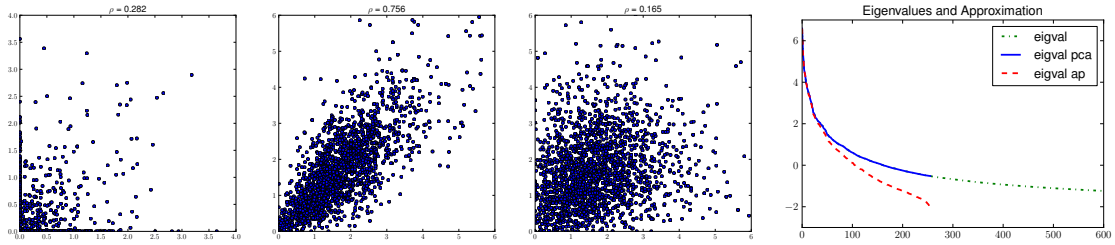


Figure 4. Visualization of the learned codes. Left: the selected subset of 256 centroids from an original set of 3200 codes. Right: The similarity between each centroid and the other codes in its cluster. For each column, the first code is the selected centroid, and the remaining codes are in the same cluster represented by it. Notice that while translational invariance is the most dominant factor, our algorithm does find invariances beyond that (e.g., notice the different colors on the last column). Best viewed in color.



(a) (b) (c) (d)

Figure 5. (a)-(c): The filter responses before and after pooling: (a) before pooling, between codes in the same cluster (correlation $\rho = 0.282$), (b) after pooling, between codes in the same cluster ($\rho = 0.756$), and (c) after pooling, between the selected centroids ($\rho = 0.165$), (d): the eigenvalues of the approximated matrix (in log scale).

after this, for each input image, we only need to encode local patches with the selected, relatively smaller dictionary of size c, not any more expensive than existing feature extraction methods.

5. Experiments

In this section we empirically evaluate two sets of experiments: using the bound to approximate the classification accuracy, and using the two-staged clustering algorithm to find better pooling invariant dictionaries.

5.1. Analysis of Selected Filters

To visually show what codes are selected by affinity propagation, we applied our approach to the CIFAR-10 dataset by first training an over-complete dictionary of 3200 codes following (Coates & Ng, 2011), and then performing affinity propagation on the 3200-dimensional pooled features to obtain 256 centroids, which we visualize in Figure 4. Translational invariance appears to be the most dominant factor, as many clusters contain translated versions of the same Gabor like code, especially for gray scale codes. On the other hand, clusters capture more than translation: clusters such as column 5 focus on finding the contrasting colors more than finding edges of exactly the same angle, and clusters such as the last column finds invariant edges of varied color. We note that the selected codes are not necessarily centered, as the centroids are selected solely from the pooled response covariance statistics, which does not explicitly favor centered patches.

We could also verify whether the second clustering stage captures the pooling invariance by checking the statistics of three types of filter responses: (a) pairwise filter responses *before pooling* between codes in the same cluster, (b) pairwise filter responses *after pooling* between codes in the same cluster, and (c) pairwise filter responses after pooling between the selected centroids. The distribution of such responses shown in Figure 5 verifies our argument: first, codes that produce uncorrelated responses before pooling may become correlated after the pooling stage (Figure 5(a,b)); second, by explicitly considering the pooled feature statistics, we are able to select a subset of the dictionary whose responses are lowly correlated (Figure 5(b,c)), preserving more information with a fixed number of codes. In addition, Figure 5(d) shows the eigenvalues of the original covariance matrix and those of the approximated matrix, showing that the approximation captures the largest eigenvalues of the original covariance matrix well.

Table 1. Classification Accuracy on the CIFAR-10 and STL datasets under different budgets.

Task	Learning Method	Accuracy
CIFAR-10 200 codes	K-means	69.02
	2x PADL	70.54 (+1.52)
	4x PADL	71.18 (+2.16)
	8x PADL	71.49 (+2.47)
CIFAR-10 1600 codes	K-means	77.97
	2x PADL	78.71 (+0.74)

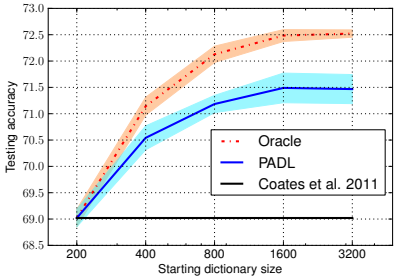


Figure 6. Performance improvement on CIFAR when using different starting dictionary sizes and a final dictionary of size 200. Shaded areas denote the standard deviation over different runs. Note that the x-axis is in log scale.

5.2. Pooling Invariant Dictionary Learning

To evaluate the improvement introduced by learning a pooling invariant dictionary as in Section 4, we show in Figure 6 the relative improvement obtained on CIFAR-10 when we use a fixed dictionary size 200, but perform feature selection from a larger overshooting dictionary as indicated by the X axis. The SVD performance is also included in the figure as an "oracle" for the feature selection performance. Learning the dictionary with our feature selection method consistently increases the performance as the size of the original dictionary increases, and is able to get about two thirds the performance gain as obtained by the oracle performance. We note again that SVD still requires the large dictionary to be used and does not save any testing time.

The detailed performance gain of our algorithm on the two datasets, using different overshooting and final dictionary sizes, is visualized in Figure 7. Table 1 summarizes the accuracy values of two particular cases - final dictionary sizes of 200 and 1600 respectively, on CIFAR. Note that our goal is not to get the best overall performance - as performance always goes up when we use more codes. Rather, we focus on two evaluations: (1) how much gain we get given a fixed dictionary size as the budget, and (2) how much computation time we save to achieve the same accuracy.

Overall, considering the pooled feature statistics always help us to find better dictionaries, especially when relatively small dictionaries are used. During

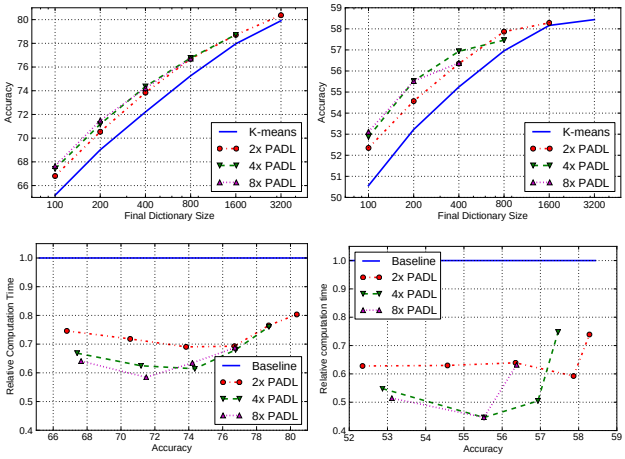


Figure 7. Above: accuracy values on the CIFAR-10 (left) and STL (right) datasets under different dictionary size budgets. "nx PADL" means learning the dictionary from a starting dictionary that is n times larger. Below: Relative computation time to achieve the same accuracy using dictionary obtained from PADL.

testing time, it costs only about 60% computation time with PADL to achieve the same accuracy as K-means does. For the STL dataset, an overly large starting dictionary may lessen the performance gain (Figure 7(b)) possibly due to feature selection being more prone to local optimum and the small number of training data (thus more overfitting). However, in general the codebook learned by PADL is consistently better than its patch-based counterpart, suggesting the applicability of the Nyström sampling view in feature learning with a multi-layer structure including spatial pooling.

6. Conclusion

Feature encoding with an overcomplete dictionary has demonstrated good performance in many applications, especially computer vision. In this paper, we proposed a novel perspective on recent approaches to object recognition by linking the feature extraction pipeline to kernel methods and Nyström sampling. As a result, we obtained useful bounds on the overall accuracy as a function of the dictionary size. We validated our contribution by showing how the derived bounds are able to explain the observed asymptotic behavior of dictionary learning across several datasets. We further extended the Nyström paradigm to take into account a pooling layer, which is a common computational layer in vision. The algorithm we propose is an efficient, end-to-end clustering method based on affinity propagation. Empirical evidence on image datasets demonstrates that significant improvement is obtained in learning compact, non-redundant dictionaries.

References

Boureau, Y, Bach, F, LeCun, Y, and Ponce, J. Learning mid-level features for recognition. In *CVPR*, 2010.

Carreira, J, Caseiro, R, Batista, J, and Sminchisescu, C. Semantic segmentation with second-order pooling. In *ECCV*, 2012.

Coates, A and Ng, A. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011.

Coates, A, Lee, H, and Ng, A. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

Coates, Adam, Karpathy, Andrej, and Ng, Andrew. Emergence of object-selective features in unsupervised feature learning. In *NIPS*, 2012.

Cortes, C, Mohri, M, and Talwalkar, A. On the impact of kernel approximation on learning accuracy. In *AISTATS*, 2010.

Denil, M and de Freitas, N. Recklessly approximate sparse coding. *arXiv preprint arXiv:1208.0959*, 2012.

Frey, BJ and Dueck, D. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

Jenatton, R, Obozinski, G, and Bach, F. Structured sparse principal component analysis. In *AISTATS*, 2010.

Krizhevsky, A, Sutskever, I, and Hinton, GE. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Kumar, S, Mohri, M, and Talwalkar, A. Sampling methods for the nyström method. *JMLR*, 13(Apr): 981–1006, 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11): 2278–2324, 1998.

Mairal, J, Bach, F, Ponce, J, and Sapiro, G. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010.

Olshausen, B and Field, DJ. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

Rigamonti, R, Brown, MA, and Lepetit, V. Are sparse representations really relevant for image classification? In *CVPR*, 2011.

Saxe, A, Koh, PW, Chen, Z, Bhand, M, Suresh, B, and Ng, A. On random weights and unsupervised feature learning. In *ICML*, 2011.

Talwalkar, Ameet and Rostamizadeh, Afshin. Matrix coherence and the nystrom method. *arXiv preprint arXiv:1004.2008*, 2010.

Wang, J, Yang, J, Yu, K, Lv, F, Huang, T, and Gong, Y. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

Yang, J, Yu, K, and Gong, Y. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

Yang, J, Yu, K, and Huang, T. Efficient highly overcomplete sparse coding using a mixture model. In *ECCV*, 2010.

Zhang, K, Tsang, IW, and Kwok, JT. Improved nyström low-rank approximation and error analysis. In *ICML*, 2008.