# Deep dynamic routing using hierarchical unsupervised learning


## Abstract

Generic vector-quantization (K-means clustering) is one of the most widely used unsupervised feature learning algorithms. Despite efforts to find its effective application to find hierarchically complex, invariant recognizers, it remains unclear how deep recognition systems could be constructed with vector-quantization in the one-learning-algorithm paradigm, as how single perceptrons build up to a forward neural network. We propose an algorithm to apply K-means on smaller decompositions, then aggregate features across increasingly larger spatio-temporal domain. This construct is iterated to form a hierarchy of invariant recognizers. We utilize the learning capacity of each basic K-mean unit and also use them as dynamic routing mechanisms for choosing pathways in a neural network. In the meantime, we are able to numerically optimize unsupervised learning objectives in the high-level, global network by euclidean metric objectives on the feature representations projected by a hierarchical network of K-means and dynamically routed modules. To verify the effectiveness of our approach, we perform unsupervised learning experiments on image and text categorization.


## 1. Introduction

Deep neural networks have been recently shown to be a competitive practical architectures for supervised learning. The effectiveness of this architectures is attributable to their hierarchical representations of the input while exploring structures in the data, such as local receptive fields and convolution. Despite this success, the element of unsupervised learning whereby data forms groupings or clusters in the deep semantic embedding, is still missing. Although unsupervised learning algorithms are widely applicable in many fields, and have shown promising results to account for low-level and mid-level representations in images (Coates et al., 2011; 2012; Le et al., 2012), it seems yet unclear whether these generic unsupervised learning algorithms, without labeled information, could help discover large collections of highly invariant object-level representations such as those discovered by large-scale supervised training of deep convolutional networks (A. Krizhevsky & Hinton, 2012).

In this work, we make two different angles of attack compared to prior works. First, we decompose the learning algorithm into one layer of unsupervised algorithm, and a 'model-based' projection with neural networks. Specifically, in our formulation for images, the unsupervised algorithm is vector-quantization, and the projection model is a multi-layer convolutional network. Using this set-up, we illustrate a novel algorithm that learns each component iteratively. This iterative algorthim is based on the K-Nearest-Neighbour principle in the Euclidean metric space. This principle not only governs the vector-quantization layer, but also, through an EM-like iterative process, the tuning of the weights in the representation model, implemented by a CNN. This perspective focuses on how a deep network could optimize globally through K-means or vector-quantization.

The previous point-of-view is limited since the learning principle of vector quantization does not apply locally, nor does the algorithm builds hierarchical more complex understanding of data. Its complementary perspective is the 'one-learning-algorithm' principle where a vector-quantizer is applied in a hierarchical manner. Similar topics are discussed in prior work such as (Coates et al., 2011; 2012). However, the we make an important observation that cross-layer feature aggregation mechanism is critical in a hierarchical vector-quantization algorithm. This is the key difference of our proposal with prior work. We propose an entirely new way of building a deep network of vector-quantizers by aggregating smaller domain data-instances through (to think: soft) histograms while enlarging the spatio domain in each layer of the hierarchy. This perspective illustrates how a larger system could learn locally and iteratively using the same learning principle.

Finally, we both perspectives and illustrate learning an entire network in an iterative fashion. The network learning combines local and iterative learning, with global optimization. We illustrate how this method could learn invariant


Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

and hierarchically more complex features through unsupervised learning.

## 2. Algorithm and Method

### 2.1. vector-quantization

Here we quickly revisit the well-known vector-quantization, or K-means clustering algorithm. Given data points $x^{(1)}, ..., x^{(m)}$, the algorithm tries to find the optimal cluster assignment, across $C$ clusters. The parameter $C$ is pre-defined. The iterative algorithm starts with a set of random assignments of the datapoints to clusters. In each iteration, the mean across each cluster is found, then another optimal assignment is computed using the reasignment. This is the formalized vector-quantization algorithm:

**Algorithm 1** K-means algorithm

Initialize cluster centroids $\mu_1, \mu_2, \mu_3, ..., \mu_k \in \mathbb{R}^n$
**while** convergence criteria not met **do**
  **for** every $i$ **do**
    $c^{(i)} := \arg\min_i \|x^{(i)} - \mu_j\|_2^2$
  **end for**
  **for** every $j$ **do**
    $\mu_j = \frac{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}$
  **end for**
**end while**

### 2.2. Vector-quantization unit processors for deep learning

Previous methods use stacking layers for building deep networks. The next layer builds on the output activations of previous layers. For K-means algorithm, the hard and soft activation layer-wise formulations are the following:

$$f_k(x) = \begin{cases} 1 \text{ if } k = \arg\min_j ||x - c^{(j)}||_2^2 \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

$$f_k(x) = \max\{0, \mu(z) - z_k\} \quad (2)$$

in above equation, $z_k = ||x - c^{(k)}||_2^2$ and $\mu(z)$ is the mean of the elements of $z$.

The benefit of stacking vector-quantization layers is simplicity, and multi-layer construction is similar to multi-layer perceptrons which is a natural construct for deep learning. The direct stacking of vector-quantization layers implies, the upper layers must operate on the bottom layer's manifold. Bottom layer clustering results or centroids clearly characterizes the data, and such characterization can be inferred easily from the feature vector of distances from each centroid. This means discovering the patterns on top of those feature vectors may lead to trivial solutions similar to the characterization of bottom layer.

In light of this, we propose an alternative method to build a deep network out of vector quantization layers. Instead of stacking, we perform domain expansion per each layer of the hierarchy, and use histograming as the feature aggregation method. This is described in the following section.

#### 2.2.1. Spatio-temporal domain expansion and histograming

Stacking of K-means modules can utilize feature embeddings of previous layer. This seemingly effective method incurs problems for unsupervised learning since upper layers can find trivial solutions of previous layer, illustrated in Figure x.

However, illustrated in Figure x, if we attempt to expand the spatial-temporal domain, and find multiple data points categorized to different centroids in the same domain, we are able to find meaningful representations which clearly adds new feature groupings of the data. This perspective looks at not only featurizing data using the K-means transformation layer, but look at building a more diverse, varied and meaningful feature collection through aggregating those featurizations on a larger spatio domain.

To illustrate and formalize this featurization method. We can abstract the K-means transformation layer as $K(\hat{x}, \{c^{(k)}\})$, which transforms the smaller data decomposition $\hat{x}$ with K-means $K$ into its one-hot vector representation obtained by allocating $\hat{x}$ to its centroid. Then, we can explain another operator $H$, which looks at a larger scope of spatio domain $x$. For data decompositions $\hat{x} \in x$, the operator $H$ aggregates one-hot vectors by $K(\hat{x}, \{c^{(k)}\})$ by summation. The overall operator which represents above-mentioned process is denoted as $\sigma(x)$:

$$\sigma(x, \{c^{(k)}\}) = H\{K(\hat{x}, \{c^{(k)}\}) \text{ where } \hat{x} \in x\} \quad (3)$$

Here $x$ is the parent data segment, and $\hat{x}$'s are smaller child segments inside $x$, $K$ is the K-means hard assignment operator, and $H$ is the histogram aggregation operator.

#### 2.2.2. Convolutional feature aggregation

This feature aggregation can operate in a convolutional manner. This is to say, the same set of k-means centroids and aggregation region can be used to apply to different segments or patches of data. Application of the previously shown operator $\sigma$ in a convolution. On each coordinate index pair $(i, j)$, the output map of the convolution layer can be computed with:

$$y_k(i, j) = \sigma_k(x(i, j), \{c^{(k)}\}) \quad (4)$$

For convolutional neural networks, we can add intermediate pooling layers such as average or max pooling.

#### 2.2.3. Neighborhood Component Analysis

The Neighborhood Component Analysis algorithm was proposed by (**?**), and later applied in a deep learning context in (Salakhutdinov & Hinton, 2007). More recently this algorithm could be seen as a form of 'metric learning' technique for cases where not only class labeled are used in the supervision task, but also a certain metric is defined, or learned at the same time in the projected non-linear embedding space (Kedem et al., 2012).

To prepare for the formulation of the deep convolutional K-means algorithm, we now re-iterate the formulation of NCA from (Roweis et al.).

We are given a set of $N$ labeled training cases $(x^a, c^a)$, where $a = 1, 2, ..., N$. For each training vector $x^a$, define the probability that point $a$ selects one of its neighbours $b$ in the transformed feature space as:

$$p_{ab} = \frac{\exp(-d_{ab})}{\sum_{z \neg a} \exp(-d_{az})}, p_{aa} = 0 \tag{5}$$

Take $d_{ab}$ to be the Euclidean distance metric:

$$d_{ab} = \|f(x^a|W) - f(x^b|W)\|^2 \tag{6}$$

and $f(|W)$ is the projection function defined by the multi-layer deep convolutional neural network parameterized by weights $W$. The probability that point $a$ belongs to class $k$ depends on the relative proximity of all other data points that belong to class $k$:

$$p(c^a = k) = \sum_{b:c^b=k} p_{ab} \tag{7}$$

The NCA objective is to maximize the expected number of correctly classified points on the training data:

$$O_{NCA} = \sum_{a=1}^{N} \sum_{b:c^a=c^b} p_{ab} \tag{8}$$

or the log probabilities of the correct classification:

$$O_{ML} = \sum_{a=1}^{N} \log( \sum_{b:c^a=c^b} p_{ab}) \tag{9}$$

#### 2.2.4. Vector-quantization on Network Projection

In the revised deep vector-quantization algorithm, we iterate between training the vector-quantization layer on the learned representations, and the non-linear projection using the multi-layer convolutional neural network. At the network training stage, we use non-linear NCA and use the current cluster asignment as class labels. At the VC stage, we keep the representations fixed and perform a number of iterations of the K-means algorithm. Essentially, instead of performing K-means clustering on data points $\mathbf{x}$, we substitute $\mathbf{x}$ with a non-linear projection using the deep neural network from data $\mathbf{x}$ to higher layer deep embedding $f(\mathbf{x}|W)$. Thus the algorithm becomes **Algorithm 2**.

**Algorithm 2** Vector-quantization on Network Projection algorithm

Initialize cluster centroids $\mu_1, \mu_2, \mu_3, ..., \mu_k \in \mathbb{R}^n$
**while** Global convergence criterion not met **do**
  *# Obtain non-linear projections using deep network*
  **for** every $i$ **do**
    $y^{(i)} = f(x^{(i)}|W)$
  **end for**
  *# run m iterations of K-means*
  **for** $m_1$ iterations **do**
    **for** every $i$ **do**
      $c^{(i)} := \arg\min_i \|y^{(i)} - \mu_j\|_2^2$
    **end for**
    **for** every $j$ **do**
      $\mu_j = \frac{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}$
    **end for**
  **end for**
  *# Fine-tune deep network with NCA*
  **for** $m_2$ iterations **do**
    $\underset{W}{\text{minimize}} O_{NCA}(x^{(i)}, c^{(i)})$
  **end for**
**end while**

We also present a variant of the algorithm, by replacing NCA with directly minimizing the K-means objective. In otherwise, the objective for fine-tuning the neural network is the sum of all $L2$ norms of differences between datapoints and its cluster mean. This alternative is formalized in **Algorithm 3**.

### 2.3. Incorporation of k-means in semi-supervised deep learning

Consider the goal to obtain best estimate for a deep discriminative model $p(y|x)$, while we leverage a k-means unsupervised process with centroids $c$. The k-means algorithm is learned iteratively as in the previous section, where the original k-means algorithm can be applied on a projection neural network. This neural network not only offers more model capacity, but also constructs an a feature representation function $\hat{x} = f_W(x)$ which serve the intermediate basis for semi-supervised learning. The network weights $W$ in the projection function $\hat{x} = f_W(x)$ represents model capacity dedicated to unsupervised learn-

**Algorithm 3** Alternative deep vector-quantization algorithm

Initialize cluster centroids $\mu_1, \mu_2, \mu_3, ..., \mu_k \in \mathbb{R}^n$
**while** Global convergence criterion not met **do**
  *# Obtain non-linear projections using deep network*
  **for** every $i$ **do**
    $y^{(i)} = f(x^{(i)}|W)$
  **end for**
  *# run m iterations of K-means*
  **for** $m_1$ iterations **do**
    **for** every $i$ **do**
      $c^{(i)} := \arg\min_i \|y^{(i)} - \mu_j\|_2^2$
    **end for**
    **for** every $j$ **do**
      $\mu_j = \frac{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}{\sum_{i=1}^{m} \mathbb{1}\{c^{(i)}=j\}}$
    **end for**
  **end for**
  *# Fine-tune deep network with K-means objective*
  **for** $m_2$ iterations **do**
    $\underset{W}{\text{minimize}} \sum_i \|x^{(i)} - \mu^{(i)}\|_2^2$
  **end for**
**end while**

ing, and upper layers are dedicated to supervised. Instead of $p(c|x)$, the intermediate $\hat{x}$ is used to learning and assign centroids , with$p(c|\hat{x}) = p(c|f_W(x))$.

If the two random variables $y$ and $c$ were to be independent given $x$, the benefit of semi-supervised learning would have been unclear. Consider transfer learning or self-taught learning frameworks, learning feature sets represented by $c$ will correlate the decision on $y$, thus it is natural to assume non-independence of $y$ and $c$ given $x$. Further part of the goal to develop semi-supervised algorithms, is to have unsupervised process help the supervised learning algorithm, or vice versa.

Motivated by this goal, we can define a dependency relationship between the two parts by marginalizing out the posterior $p(y|x)$.

$$p(y|x) = \sum_c p(y, c|\hat{x}) \tag{10}$$

$$= \sum_c p(y|c, \hat{x}) p(c|\hat{x}) \tag{11}$$

$$= \sum_c p(y|c, f_W(x)) p(c|f_W(x)) \tag{12}$$

Rather then only modeling discriminative posterior, $p(y|c, x)$ represents another discriminative model conditional of knowing results of unsupervised learning. This term is formulated as a neural network, with simplest form a softmax regression. In practice we apply this framework in the context of language modeling and text classification, where $p(c|f_W(x))$ is specified by a softmax regression with cluster dependent weightings on the vocabulary words.

Intuitively, we create more flexibility or specialization in the label classification model by making it dependent on the clustering algorithm. We can see unsupervised clustering as being used to dynamically route the problem, in the context of language models. to different topics or word collections. This algorithm is loosely related to the mixture-of-experts paradigm, where we rely on the clustering algorithm, and the dependency between label classification and cluster assignment to route data-points to separate labeling experts.

In more detail, the routing of mixture-of-experts is done using the $p(c|f_W(x))$. This term can be specified using a softmax function weighted by euclidean distances to the centroids, similar to equation 5:

$$p(y|c^{(k)}, f_W(x)) = \frac{\exp(-d_{\{c^{(k)}, f_W(x)\}})}{\sum_{j \in \{\mathbf{c}\}} \exp(-d_{\{c^{(j)}, f_W(x)\}})} \tag{13}$$

The counter part of this algorithm will be to marginalize out $y$:

$$p(c|x) = \sum_y p(y, c|\hat{x}) \tag{14}$$

$$= \sum_y p(c|y, \hat{x}) p(y|\hat{x}) \tag{15}$$

$$= \sum_y p(c|y, f_W(x)) p(y|f_W(x)) \tag{16}$$

How does supervised learning in return help unsupervised so that the algorithm is balanced? one way is have a neural network to model class-specific assignment functions $p(c|y, f_W(x))$, so that the $p(c|x)$ quantity can written as a linear combination of classifier probability $p(y|f_W(x))$ and the cluster assignment function. The class-dependent assignment function is learned the same way as k-means projection layers.

Consider the cost of if two data points belong to the same class (centroid) in one problem, but may or may not belong to the same class for the other problem. Can we utilize or incorporate this cost function for semi-supervised or multi-task learning problems?

All in all, the loop-EM algorithm can be used to optimize for $p(y|x)$ and $p(c|x)$ iteratively. Intuitively, the supervised discriminative model and unsupervised algorithms can be improved in a alternate, iterative fashion.

## 3. Experiments

We perform experiments on the IMDB dataset for sentiment classification. The algorithm we adopt for the base model as $f_W(x)$ is multi-layer LSTMs with word embedding representations. On the LSTMs, unsupervised learning to determine $p(c|f_W(x))$ in Equation 10 is performed using k-means clustering described in Section 2.2.4. The learning algorithm performs initially random assignments to $c$, then iteratively optimize for the best solution for projection network using the NCA objective function.

In this sub-section of our algorithm, we differentiate ourselves to prior art for using deep learning clustering algorithm, by first, we adopt a completely different model and data modality, with multi-layer LSTMs to learn on text inputs, and second, applying a different algorithm NCA as opposed to label classification. Finally, we adopt this unsupervised learning step as part of our algorithm to explicitly define the dependency framework elaborated in Equation 10, and learn the framework as a complete model.

The $p(y|c, f_W(x))$ term in Equation 10 is specified in two ways. We first try to define the term with a separately parameterized softmax regression function, and use other auxiliary parameters $b$ for topic word probabilities and parameters in the variational auto-encoder in (**?**).

## References


A. Krizhevsky, I. Sutskever and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Coates, Adam, Ng, Andrew Y, and Lee, Honglak. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

Coates, Adam, Karpathy, Andrej, and Ng, Andrew Y. Emergence of object-selective features in unsupervised feature learning. In *Advances in Neural Information Processing Systems*, pp. 2681–2689, 2012.

Kedem, Dor, Tyree, Stephen, Sha, Fei, Lanckriet, Gert R, and Weinberger, Kilian Q. Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pp. 2573–2581, 2012.

Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G.S., and Ng, A. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

Roweis, Sam, Hinton, Gf, and Salakhutdinov, R. Neighbourhood component analysis. In *Neural Information Processing Systems*, volume 17, pp. 513–520.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 412–419, 2007.