
Why Size Matters: Feature Coding as Nyström Sampling

Oriol Vinyals
UC Berkeley
Berkeley, CA

Yangqing Jia
UC Berkeley
Berkeley, CA

Trevor Darrell
UC Berkeley
Berkeley, CA

1 Introduction

Recently, the computer vision and machine learning community has been in favor of feature extraction pipelines that rely on a coding step followed by a linear classifier, due to their overall simplicity, well understood properties of linear classifiers, and their computational efficiency. In this paper we propose a novel view of this pipeline based on kernel methods and Nyström sampling. In particular, we focus on the coding of a data point with a local representation based on a dictionary with fewer elements than the number of data points, and view it as an approximation to the actual function that would compute pair-wise similarity to all data points (often too many to compute in practice), followed by a Nyström sampling step to select a subset of all data points.

Furthermore, since bounds are known on the approximation power of Nyström sampling as a function of how many samples (i.e. dictionary size) we consider, we can 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. This model may help explaining the positive effect of the codebook size [2, 7] and justifying the need to stack more layers (often referred to as deep learning), as flat models empirically saturate as we add more complexity.

2 The Nyström View

We specifically consider forming a dictionary by sampling our training set. To encode a new sample $\mathbf{x} \in \mathbb{R}^d$, we apply a (generally non-linear) coding function \mathbf{c} so that $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^c$. Note that d is the dimensionality of the original feature space, while c is the dictionary size. The standard classification pipeline considers $\mathbf{c}(\mathbf{x})$ as the new feature space, and typically uses a linear classifier on this space. For example, one may use the threshold encoding function [2] as an example: $\mathbf{c}(\mathbf{x}) = \max(0, \mathbf{x}^\top \mathbf{D} - \alpha)$ where $\mathbf{D} \in \mathbb{R}^{d \times c}$ is the dictionary. Note that our discussion on coding is valid for many different feed-forward 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} (as long as over-fitting is handled by the classification algorithm). In general, 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 [6]. We define the new coded feature 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 $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{c}(\mathbf{x}_i)^\top \mathbf{c}(\mathbf{x}_j)$. The kernel matrix is then $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$. Naively applying Nyström sampling to the matrix \mathbf{K} does not save any computation, as every column of \mathbf{K} requires computing an inner product with N samples. However, if we decompose the matrix \mathbf{C} with Nyström sampling (i.e., with a subsampled dictionary) we obtain $\mathbf{C}' \approx \mathbf{C}$, and as a consequence $\mathbf{K}' \approx \mathbf{K}$:

$$\mathbf{C}' = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top, \quad \mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top = \mathbf{E}\mathbf{A}\mathbf{E}^\top$$

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} .

3 Main Results on Approximation Bounds

More interestingly, many bounds on the error made in estimating \mathbf{C} by \mathbf{C}' exist, and finding better sampling schemes that improve such bounds is an active topic in the machine learning community (see e.g. [4]). The bound we start with is [4]:

$$\|\mathbf{C} - \mathbf{C}'\|_F \leq \|\mathbf{C} - \mathbf{C}_k\|_F + \epsilon \max(n\mathbf{C}_{ii}) \quad (1)$$

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). Note that, if we assume that our training set can be explained by a manifold of dimension k (i.e. the first term in the right hand side of eq. 1 vanishes), then the error is proportional to ϵ times a constant (that is dataset dependent).

Thus, if we fix k to the value that retains enough energy from \mathbf{C} , we get a bound that for every c (dimension of code), gives a minimum ϵ to plug in equation 1. This gives us a useful bound of the form $\epsilon \geq Mc^{-\frac{1}{4}}$ for some constant M (that depends on k). Putting it all together, we get:

$$\|\mathbf{C} - \mathbf{C}'\|_F \leq O + Mc^{-\frac{1}{4}}$$

with O and M constants that are dataset specific.

Having bounded the error \mathbf{C} is not 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} . However, we are able to prove that the error bound on \mathbf{K}' is in the same format as that on \mathbf{C} :

$$\|\mathbf{K} - \mathbf{K}'\|_F \leq O + Mc^{-\frac{1}{4}} \quad (2)$$

Even though we are not aware of an easy way to formally link degradation in Frobenius norm of our approximation \mathbf{K}' to \mathbf{K} to classification accuracy, the bound above is informative as one may reasonably expect kernel matrices of different quality to have classification performances in the same trend.

4 Experiments

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 bound, we do interpolation of the observed accuracy in the first two 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 codebook sizes, and then quickly estimate what the accuracy would be when running a full experiment with a much larger dictionary size.

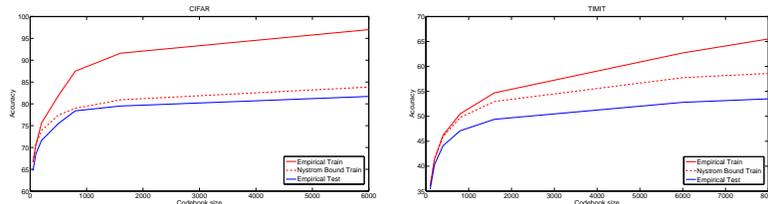


Figure 1: Empirical accuracy (solid line) and Nyström model accuracy (dashed line) on the training (red) and testing (blue) sets versus dictionary size, on CIFAR-10 (left) and TIMIT (right).

Figure 1 shows the results on on the CIFAR-10 image classification and TIMIT speech recognition datasets respectively. It is observed that the derived model closely follow our own empirical observations, with red dashed line serving as a lower bound of the actual accuracy and following the

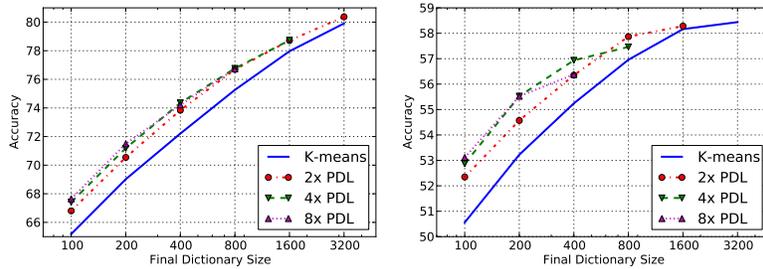


Figure 2: Accuracy values on the CIFAR-10 (left) and STL (right) datasets under different final dictionary size. “ n x PDL” means overshooting the dictionary from a starting dictionary that is n times larger than the final one. We refer to our tech report [3] for more details.

shape of the empirical accuracy, predicting its saturation. The model is never too tight though, due to various factors of our approximation, e.g., the analytical relationship between the approximation of \mathbf{K} and the classification accuracy is not clear.

The Nyström view of feature encoding and the approximation bounds we proposed helps understanding several key observations in the recent literature: (1) the linear classifier performance is always bounded when using a fixed codebook, and performance increases when the codebook grows [2], even with a huge codebook [7], and (2) simple dictionary learning techniques have been found efficient in some classification pipelines [1, 5], and K-means works particularly well as a dictionary learning algorithm albeit its simplicity, a phenomenon that is common in the Nyström sampling context [4].

In addition, in many image classification tasks the feature extraction pipeline is composed of more than feature encoding. For example, recent state-of-the-art methods pool locally encoded features spatially to form the final feature vector. The Nyström view presented in the paper inspires us to employ findings in the machine learning field to learn better, pooling-aware dictionaries. In one of our related work [3], we form a dictionary by first “overshooting” the coding stage with a larger dictionary, and then pruning it using K-centers with pooled features. Figure 2 shows an increase in the final classification accuracy compared with the baseline that only learns the dictionary on the patch-level, with no additional computation cost for either feature extraction or classification.

References

- [1] A Coates, H Lee, and AY Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.
- [2] A Coates and AY Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011.
- [3] Y Jia, O Vinyals, and T Darrell. Pooling-invariant image feature learning. <http://arxiv.org/abs/1302.5056>, 2013.
- [4] S Kumar, M Mohri, and A Talwalkar. Sampling methods for the nyström method. *JMLR*, 13(4):981–1006, 2012.
- [5] A Saxe, PW Koh, Z Chen, M Bhand, B Suresh, and AY Ng. On random weights and unsupervised feature learning. In *ICML*, 2011.
- [6] J Wang, J Yang, K Yu, F Lv, T Huang, and Y Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.
- [7] J Yang, K Yu, and T Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.