# Why Size Matters:

**Feature Coding as Nystro¨m Sampling**

**Oriol Vinyals** UC Berkeley Berkeley, CA

**Yangqing Jia** UC Berkeley Berkeley, CA

**Trevor Darrell** UC Berkeley Berkeley, CA

arXiv:1301.5348v2 [cs.LG] 16 Apr 2013

## Introduction

Recently, the computer vision and machine learning community has been in favor of feature extrac- tion 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 Nystro¨m 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 Nystro¨m sampling step to select a subset of all data points.

Furthermore, since bounds are known on the approximation power of Nystro¨m sampling as a func- tion of how many samples (i.e. dictionary size) we consider, we can derive bounds on the approx- imation 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 satu- rate 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.

## The Nystro¨m View

We specifically consider forming a dictionary by sampling our training set. To encode a new sample **x** R*d*, we apply a (generally non-linear) coding function **c** so that **c**(**x**) R*c*. Note that *d* is the dimensionality of the original feature space, while *c* is the dictionary size. The standard classification pipeline considers **c**(**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: **c**(**x**) = max(0*,* **x***T***D** *α*) where **D** R*d×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 **x** using the whole training set **X** R*d×N* , which can be seen as the best local coding of the training set **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 **C** = max(0*,* **X***T***X** *α*), where the *i*-th row of **C** corresponds to coding the *i*-th sample **c**(**x***i*). The linear kernel function between samples *i* and *j* is *k*(**x***i,* **x***j*) = **c**(**x***i*)*T***c**(**x***j*). The kernel matrix is then **K** = **CC***T*. Naively applying Nystro¨m sampling to the matrix **K** does not save any computation, as every column of **K** requires computing an inner product with *N* samples. However, if we decompose the matrix **C** with

*∈*

*−*

Nystro¨m sampling (i.e., with a subsampled dictionary) we obtain **C***j ≈* **C**, and as a consequence

**K***j ≈* **K**:

**C***j* = **EW***−*1**E***T,* **K***j* = **C***j***C***jT* = **EW***−*1**E***T***EW***−*1**E***T* = **EΛE***T*

where the first equation comes from applying Nystro¨m sampling to **C**, **E** is a random subsample of the columns of **C**, and **W** the corresponding square matrix with the same random subsample of both columns and rows of **C**.

## Main Results on Approximation Bounds

More interestingly, many bounds on the error made in estimating **C** by **C***j* 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]:

*||***C** *−* **C***j||F ≤ ||***C** *−* **C***k||F* + *s* max(*n***C***ii*) (1) valid if *c* 64*k/s*4 (*c* is the number of columns that we sample from **C** to form **E**, i.e. the codebook size), where *k* is the sufficient rank to estimate the structure of **C**, and **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 *s* times a constant (that is dataset dependent).

Thus, if we fix *k* to the value that retains enough energy from **C**, we get a bound that for every *c* (dimension of code), gives a minimum *s* to plug in equation 1. This gives us a useful bound of the form *s ≥ Mc−* 4 for some constant *M* (that depends on *k*). Putting it all together, we get:

1

1

*||***C** *−* **C***j||F ≤ O* + *Mc−* 4

with *O* and *M* constants that are dataset specific.

Having bounded the error **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 **C** affect the error on the kernel matrix **K**. However, we are able to prove that the error bound on **K***j* is in the same format as that on **C**:

*||***K** *−* **K***j||F ≤ O* + *Mc−* 4 (2)

1

Even though we are not aware of an easy way to formally link degradation in Frobenius norm of our approximation **K***j* to **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.

## Experiments

We empirically evaluate the bound on the kernel matrix, used as a proxy to model classification ac- curacy, 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 prac- tical 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.

100

CIFAR

TIMIT

70

Empirical Train Nystrom Bound Train Empirical Test

Empirical Train Nystrom Bound Train Empirical Test

95 65

90

60

85

55

Accuracy

Accuracy

80

50

75

45

70

65 40

60

0 1000 2000 3000 4000 5000 6000

Codebook size

35

0 1000 2000 3000 4000 5000 6000 7000 8000

Codebook size

Figure 1: Empirical accuracy (solid line) and Nystro¨m 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 ob- servations, with red dashed line serving as a lower bound of the actual accuracy and following the

59

K-means 2x PDL

4x PDL

8x PDL

K-means 2x PDL

4x PDL

8x PDL

80

58

78

57

76 56

Accuracy

Accuracy

74 55

72

70

68

66

100 200 400 800

Final Dictionary Size

1600 3200

54

53

52

51

50

100 200 400 800 1600 3200

Final Dictionary Size

Figure 2: Accuracy values on the CIFAR-10 (left) and STL (right) datasets under different final dictionary size. “nx 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 **K** and the classification accuracy is not clear.

The Nystro¨m view of feature encoding and the approximation bounds we proposed helps under- standing several key observations in the recent literature: (1) the linear classifier performance is al- ways 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 Nystro¨m 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 Nystro¨m 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*,](http://arxiv.org/abs/1302.5056) 2013.

1. S Kumar, M Mohri, and A Talwalkar. Sampling methods for the nystro¨m method. *JMLR*, 13(4):981–1006, 2012.
2. A Saxe, PW Koh, Z Chen, M Bhand, B Suresh, and AY Ng. On random weights and unsuper- vised feature learning. In *ICML*, 2011.
3. J Wang, J Yang, K Yu, F Lv, T Huang, and Y Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.
4. J Yang, K Yu, and T Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.