Compressive Classification (Machine Learning without learning)

Vincent Schellekens* and Laurent Jacques*

arXiv:1812.01410v1 [cs.LG] 4 Dec 2018

***Abstract*—Compressive learning is a framework where (so far unsupervised) learning tasks use not the entire dataset but a compressed summary (sketch) of it. We propose a compressive learning classification method, and a novel sketch function for images.**

1 Introduction and background

Machine Learning (ML)—inferring models from datasets of numerous learning examples—recently showed unparalleled success on a wide variety of problems. However, modern massive datasets necessitate a long training time and large memory storage. The recent *Compressive Learning* (CL) framework alleviates those drawbacks by computing a compressed summary of the dataset—its *sketch*—prior to any learning [1]. The sketch is easily computed in a single parallelizable pass, and its required size (to capture enough information for successful learning) does not grow with the number of examples: CLs time and memory requirements are thus unaffected by the dataset size.

So far, CL focused on *unsupervised* ML tasks, where learning examples don't belong to a (known) class [1, 2, 3]. We show that CL easily extends to *supervised* ML tasks by proposing (Sec. 2) and experimentally validating (Sec. 3) a first simple *compressive classification method* using only a sketch of the labeled dataset (Fig. 1). We also introduce a sketch feature function leveraging a random convolutional neural network to better capture information in images. While not as accurate as ML methods learning from the full dataset, this compressive classification scheme still attains remarkable accuracy considering its *unlearned nature*. Our method also enjoys from a nice geometric interpretation, *i.e.*, Maximum A Posteriori classification performed in the Reproducible Kernel Hilbert Space associated with the sketch.

(Unsupervised) Compressive Learning: Unsupervised ML usually amount to estimate parameters of a distribution $\mathcal{P}$, from a dataset $\mathcal{X} := \{\boldsymbol{x}_i \sim_{\text{iid}} \mathcal{P}\}_{i=1}^N \subset \mathbb{R}^n$ of examples—associated to an *empirical distribution* $\hat{\mathcal{P}}_{\mathcal{X}} := \frac{1}{N}\sum_{\boldsymbol{x}_i \in \mathcal{X}} \delta_{\boldsymbol{x}_i}$, with $\delta_{\boldsymbol{u}}$ the Dirac measure at $\boldsymbol{u}$. While most unsupervised ML algorithms require (often multiple times) access to the entire dataset $\mathcal{X}$, CL algorithms require only access to the sketch: a single vector $\boldsymbol{z}_{\mathcal{X}} \in \mathbb{C}^m$ summarizing $\mathcal{X}$. This *dataset sketch* $\boldsymbol{z}_{\mathcal{X}}$ actually serves as a proxy for the true *distribution sketch* $\mathcal{A}(\mathcal{P})$, *i.e.*, a linear embedding of the "infinite-dimensional" probability distribution $\mathcal{P}$ into $\mathbb{C}^m$, a space of lower dimension:

$$\mathcal{A}(\mathcal{P}) := \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} f(\boldsymbol{x}) \simeq \boldsymbol{z}_{\mathcal{X}} := \mathcal{A}(\hat{\mathcal{P}}_{\mathcal{X}}) = \frac{1}{N}\sum_{\boldsymbol{x}_i \in \mathcal{X}} f(\boldsymbol{x}_i), \quad (1)$$

where f is a random nonlinear feature map to $\mathbb{C}^m$. This map defines a positive definite kernel $\kappa(\boldsymbol{u}, \boldsymbol{v}) := \mathbb{E}\langle f(\boldsymbol{u}), f(\boldsymbol{v})\rangle$, and κ in turn provides a Reproducible Kernel Hilbert Space (RKHS) $\mathcal{H}_\kappa$ to embed distributions; $\mathcal{A}$ indirectly maps $\mathcal{P}$ to its *Mean Map* $\kappa(\cdot, \mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, \kappa(\cdot, \boldsymbol{x}) \in \mathcal{H}_\kappa$ [4, 5, 6]. Existing methods [2, 3] use Random Fourier Features [7] as map f:

$$f_{\text{RFF}}(\boldsymbol{x}) = \left[\exp(\mathrm{i}\,\boldsymbol{\omega}_j^T \boldsymbol{x})\right]_{j=1}^m \quad \text{with} \quad \boldsymbol{\omega}_j \sim_{\text{iid}} \Lambda, \quad (2)$$

and κ is then shift-invariant and the Fourier transform of the distribution Λ: $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \varpi(\boldsymbol{x} - \boldsymbol{x}') := (\mathcal{F}\Lambda)(\boldsymbol{x} - \boldsymbol{x}')$ [8]. CL is promising because the sketch $\boldsymbol{z}_{\mathcal{X}}$ retains sufficient information (to compete with traditional ML) whenever its size m exceeds some value independent on the number of examples N, yielding algorithms that scale well when N increases.

Random Convolutional Neural Networks (CNN): Shift-invariant kernels are not that relevant when dealing with images (they are sensitive to image translations for example). Recent studies have shown that the last layer of a *randomly weighted* (convolutional) neural network CNN (combining convolutions with random weights, nonlinear activations, and pooling operations) captures surprisingly meaningful image features [9, 10, 11, 12]. We thus propose the feature map $f_{\text{CNN}}(\boldsymbol{x}) = \text{CNN}(\boldsymbol{x}) \in \mathbb{R}^m$ as sketch map f for images: the associated kernel κ is (for a fully connected network) an *arc-cosine kernel*, that surpasses shift-invariant kernels for solving image classification tasks with kernel methods [12].

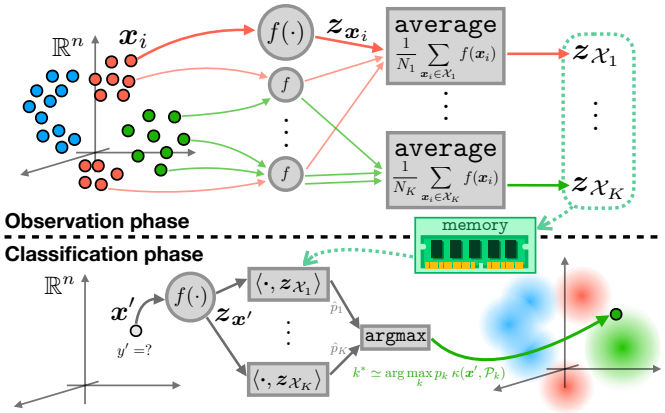

Figure 1: **Observation phase:** we record only a summary of the dataset $\mathcal{X}$ as the K class sketches $\boldsymbol{z}_{\mathcal{X}_k}$: the class average of non-linear maps $\boldsymbol{z}_{\boldsymbol{x}_i} = f(\boldsymbol{x}_i)$ of the examples $\boldsymbol{x}_i$. **Classification phase:** a new sample $\boldsymbol{x}'$ gets the class label k^* that maximizes the correlation between its sketch $\boldsymbol{z}_{\boldsymbol{x}'}$ and the stored class sketches; this can be interpreted as a MAP classifier in a RKHS $\mathcal{H}_\kappa$.

2 Compressive learning classification

Observation phase: *Supervised* ML infers a mathematical model from a labeled dataset $\mathcal{X} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where each signal $\boldsymbol{x}_i \in \mathbb{R}^n$ belongs to a class $\mathcal{C}_k$ as designated by its *class label* $y_i \in [K]$. Denoting $p_k := \mathbb{P}(\boldsymbol{x} \in \mathcal{C}_k) = \mathbb{P}(y = k)$, the signals are assumed drawn from an unknown density $\mathcal{P}$:

$$\boldsymbol{x_i} \sim_{\text{iid}} \mathcal{P} = \textstyle\sum_{k=1}^K p_k\, p(\boldsymbol{x} | \boldsymbol{x} \in \mathcal{C}_k) =: \sum_k p_k \mathcal{P}_k(\boldsymbol{x}). \quad (3)$$

As illustrated in Fig. 1(top), our *supervised compressive learning* framework considers that $\mathcal{X}$ is not explicitly available but compressed as a collection of K *class sketches* $\boldsymbol{z}_{\mathcal{X}_k}$ defined as:

$$\boldsymbol{z}_{\mathcal{X}_k} = \mathcal{A}(\hat{\mathcal{P}}_{\mathcal{X}_k}) \quad \text{where} \quad \mathcal{X}_k := \{\boldsymbol{x}_i \in \mathcal{C}_k\}. \quad (4)$$

*E-mail: *{vincent.schellekens, laurent.jacques}@uclouvain.be*. ISPGroup, ELEN/ICTEAM, UCLouvain (UCL), B1348 Louvain-la-Neuve, Belgium. VS and LJ are funded by Belgian National Science Foundation (F.R.S.-FNRS).

We can also require approximated *a priori* class probabilities $\hat{p}_k$, *e.g.*, $\hat{p}_k = \frac{N_k}{N}$ if we count the class occurrences $N_k = |\mathcal{X}_k|$, or setting an uniform prior $\hat{p}_k = \frac{1}{K}$ otherwise.

Classification phase: Under (3), the optimal classifier (minimal error probability) for a test example $\boldsymbol{x}'$ is the Maximum A Posteriori (MAP) estimator $k^{\text{MAP}} := \arg\max_k p_k \mathcal{P}_k(\boldsymbol{x}')$, where $\mathcal{P}_k$ is generally hard to estimate. In our CL framework, we classify $\boldsymbol{x}'$ from $\boldsymbol{z}_{\mathcal{X}_k}$ and $\hat{p}_k$ only (Fig. 1, bottom): we acquire its sketch $\boldsymbol{z}_{\boldsymbol{x}'} = f(\boldsymbol{x}')$ and maximize the correlation with the class sketch weighted by $\hat{p}_k$, *i.e.*, we assign to $\boldsymbol{x}'$ the label

$$k^* := \arg\max_k \ \hat{p}_k \langle \boldsymbol{z}_{\boldsymbol{x}'}, \boldsymbol{z}_{\mathcal{X}_k} \rangle \qquad \text{(CC)}$$

Note that this Compressive Classifier (CC) does not require parameter tuning. Interestingly, under a few approximations, this procedure can be seen as a MAP estimator in the RKHS $\mathcal{H}_\kappa$. Indeed, we first note that if m is large, the law of large numbers (LLN) provides the *kernel approximation* (KA)

$$\langle f(\boldsymbol{u}), f(\boldsymbol{v}) \rangle \simeq \kappa(\boldsymbol{u}, \boldsymbol{v}), \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n. \qquad \text{(KA)}$$

Assuming N_k is also large, another use of the LLN gives the *mean map approximation* (MMA): we have both $\hat{p}_k \simeq p_k$ and

$$\langle \boldsymbol{z}_{\boldsymbol{u}}, \boldsymbol{z}_{\mathcal{X}_k} \rangle = \frac{1}{N_k} \sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \langle f(\boldsymbol{u}), f(\boldsymbol{x}_i) \rangle \underset{\text{(KA)}}{\simeq} \frac{1}{N_k} \sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \kappa(\boldsymbol{u}, \boldsymbol{x}_i)$$
$$\simeq \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_k} \kappa(\boldsymbol{u}, \boldsymbol{x}) =: \kappa(\boldsymbol{u}, \mathcal{P}_k) \quad \forall \boldsymbol{u} \in \mathbb{R}^n. \qquad \text{(MMA)}$$

Consequently, under the KA and MMA approximations,

$$k^* \simeq \arg\max_k \ p_k \, \kappa(\boldsymbol{x}', \mathcal{P}_k), \qquad (5)$$

or in other words, we replace $\mathcal{P}_k$ in the MAP estimator by its Mean Map $\kappa(\cdot, \mathcal{P}_k)$—its embedding in $\mathcal{H}_\kappa$—such that CC computes a *MAP estimation inside the RKHS* $\mathcal{H}_\kappa$. In all generality $\kappa(\cdot, \mathcal{P}_k)$ is not a probability density function, but can be interpreted as a smoothing of $\mathcal{P}_k$ by convolution with $\varpi(\boldsymbol{u}) := \kappa(\boldsymbol{u}, 0)$ if κ is a properly scaled shift-invariant kernel. Alternatively, (5) can be seen as a Parzen-windows classifier—a nonparametric Support Vector Machine (without weights learning)—evaluated compressively thanks to the sketch [13, 14].

3 Experimental proof of concept

Synthetic datasets: We build two datasets that are not linearly separable (Fig. 2 left), and sketch them using $f = f_{\text{RFF}}$ with $\Lambda \sim \mathcal{N}(\boldsymbol{0}, \frac{I_n}{\sigma^{-2}})$: therefore $\kappa(\boldsymbol{u}, \boldsymbol{v}) \propto \exp(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2})$. As shown Fig. 2(right), the test accuracy of CC improves with m until reaching—when the KA is good enough—a constant floor depending on the compatibility between κ and $\mathcal{P}$. Accuracy is almost optimal when κ is close to the constituents of $\mathcal{P}$ (*e.g.*, 1st dataset, $\sigma = 0.1$), but degrades when the kernel scale and/or shape mismatches the data clusters (*e.g.*, 1st dataset, $\sigma = 10$; or 2nd dataset). CC thus reaches good accuracy provided m is large enough and κ is well adapted to the task.

Standard datasets: We also test CC on some well-known "real-life" datasets from the UCI ML Repository [15]. Table 1 compares the error rates of CC and SVM, a fully learned approach. Although worse than SVM, CC is surprisingly accurate considering its compressive nature, low computational cost (especially when $m = 50$), and that κ is a basic, non-tuned kernel.

Image classification: More challenging are image classification datasets: handwritten digit recognition (MNIST [16]) and vehicle/animal recognition (CIFAR-10 [17]). We use $f = f_{\text{CNN}}$ (the default architecture provided by [18]) because it yielded better accuracy than f_{RFF}, and compare CC to the same CNN architecture with a classification layer, with all weights learned in one pass over $\mathcal{X}$ for fairness. Again CC is outperformed by the learned approach, but still achieves reasonable, non-trivial accuracy. Surprisingly, CC performs here *better* on the test set than on the training set.

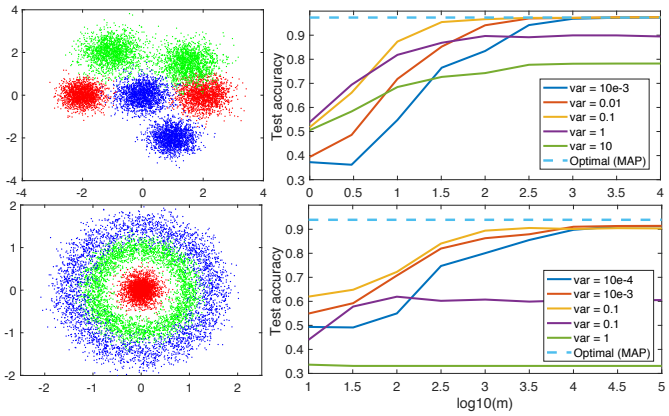

Figure 2: **Left**: synthetic 2-d datasets of $N = 10^4$ examples from $K = 3$ equiprobable classes, separated into $2/3$ for "training" (observation phase) and $1/3$ for testing (classification phase). **Right**: testing accuracy (average over 10 trials) of our compressive classification method for different values of σ (noted `var`) and increasing m (solid), compared to MAP classification (dashed).

	N	n	K	SVM	$m = 50$	$m = 1000$
Iris	150	4	3	2.00	6.51 ± 1.81	5.51 ± 1.23
				4.00	8.22 ± 3.25	6.18 ± 2.40
Wine	178	13	3	0.84	4.56 ± 2.34	2.43 ± 0.72
				1.69	13.75 ± 4.09	8.19 ± 1.29
Breast cancer	569	30	2	3.67	7.00 ± 1.40	3.93 ± 0.39
				2.13	9.22 ± 2.33	6.23 ± 0.69
Adult (3 attr.)	30718	3	2	21.03	23.88 ± 4.37	23.11 ± 1.05
				21.06	36.09 ± 6.67	35.04 ± 1.63

Table 1: Standard datasets: train set (white, 2/3 of data) and test set (gray) average error rates $\pm$ standard deviation (in %, 100 repetitions), for SVM and CC with $m \in \{50, 1000\}$, and with $\sigma = 2$ (data re-scaled inside $[-1, +1]^n$).

	N	n	CNN	m = 250	m = 5000
MNIST	60000	$28 \times 28 \times 1$	1.60 ± 0.12	17.73 ± 1.43	16.60 ± 1.54
	10000		1.63 ± 0.11	16.83 ± 1.39	15.80 ± 1.61
CIFAR10	50000	$32 \times 32 \times 3$	39.08 ± 1.48	71.76 ± 1.85	72.83 ± 2.00
	10000		40.28 ± 1.36	71.12 ± 1.72	72.02 ± 1.85

Table 2: Image datasets: train (white) and test (gray) average error rates $\pm$ standard deviation (in %, 10 repetitions), for SVM and CC with $m \in \{250, 5000\}$.

4 Discussion and conclusion

We proposed a very simple and flexible *compressive classification* method, relying only on class sketches: accumulated random nonlinear signatures $f(\cdot)$ of the learning examples. This classifier is cheap to evaluate (*e.g.*, in low-power hardware, following ideas from [19]), involves no parameter tuning, and has an interesting interpretation: a MAP estimator inside the RKHS $\mathcal{H}_\kappa$ associated with the kernel κ defined by f. Preliminary experimental results, relying on a basic Gaussian κ, are an encouraging proof of concept, but indicate room for improvement if the mapping f (and associated kernel κ) are optimized according to the true data distribution; for example, image classification accuracy improves when f is a random CNN (defining a shift-variant κ). Intuitively, κ should be such that the Mean Maps $\kappa(\cdot, \mathcal{P}_k) \in \mathcal{H}_\kappa$ of different classes k are "well separated" (ideally as much separated as the initial, unknown densities $\mathcal{P}_k$). This could be done by adding some a priori assumptions on the densities $\mathcal{P}_k$, or by first getting a rough estimation of them through a form of distilled sensing [20]. To be reliable, compressive classification also requires precise, non-asymptotic guarantees, *e.g.*, using results from [5] and [7].

References

[1] R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin, “Compressive Statistical Learning with Random Feature Moments,” *ArXiv e-prints*, Jun. 2017.

[2] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, “Compressive K-means,” *ICASSP 2017 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[3] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez, “Sketching for Large-Scale Learning of Mixture Models,” *Information and Inference: A Journal of the IMA*, 2017.

[4] N. Aronszajn, “Theory of reproducing kernels,” *Transactions of the Amererican Mathematical Sociecty*, no. 68, pp. 337–404, 1950.

[5] A. Smola, A. Gretton, L. Song, B. Scholkopf, “A Hilbert space embedding for distributions”, *International Conference on Algorithmic Learning Theory*, Springer, Berlin, Heidelberg, 2007.

[6] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet, “Hilbert Space Embeddings and Metrics on Probability Measures,” *Journal of Machine Learning Research*, vol. 11, pp. 1517–1561, Aug. 2010.

[7] A. Rahimi and B. Recht, “Random Features for Large-Scale Kernel Machines,” in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 1177–1184.

[8] W. Rudin, *Fourier Analysis on Groups*. Interscience Publishers, 1962.

[9] D. Ulyanov, A. Vedaldi, V. Lempitsky, "Deep Image Prior," *arXiv preprint*, 2017.

[10] R. Giryes, G. Sapiro, A.M. Bronstein "Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy?," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3444-3457, Jul. 2016.

[11] A. Rosenfeld, J.K. Tsotsos, "Intriguing Properties of Randomly Weighted Networks: Generalizing While Learning Next to Nothing," *arXiv preprint*, 2018.

[12] Y. Cho, L.K. Saul, "Kernel methods for deep learning," *Advances in neural information processing systems*, 2009.

[13] R. O. Duda, P. E. Hart “Pattern Classification and Scene Analysis”, Wiley Interscience, 1973.

[14] B. Scholkopf, A. J. Smola, “Learning with kernels: support vector machines, regularization, optimization, and beyond”, MIT press, 2001.

[15] A. Asuncion, D.J. Newman, “UC Irvine Machine Learning Repository,” `http://archive.ics.uci.edu/ml/index.php`, Accessed: 2018-05-15.

[16] Y. LeCun, C. Cortes, and C. J. Burges, “The MNIST database of handwritten digits,” `http://yann.lecun.com/exdb/mnist/`, Accessed: 2018-05-15.

[17] A. Krizhevsky, “The CIFAR-10 dataset,” `https://www.cs.toronto.edu/~kriz/cifar.html`, Accessed: 2018-05-15.

[18] The MatConvNet Team, “MatConvNet: CNNs for MATLAB,” `http://www.vlfeat.org/matconvnet/`, Accessed: 2018-06-13.

[19] V. Schellekens, and L. Jacques, "Quantized Compressive K-Means," *arXiv preprint*, arXiv:1804.10109, 2018.

[20] J. Haupt, R.M. Castro, and R. Nowak, "Distilled sensing: Adaptive sampling for sparse detection and estimation," *IEEE Transactions on Information Theory*, vol. 57, no. 9, 2011, pp. 6222-6235.