ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Learning with Few Bits on Small–Scale Devices: from Regularization to Energy Efficiency

Davide Anguita, Alessandro Ghio, Luca Oneto and Sandro Ridella

DITEN - University of Genova
Via Opera Pia 11A, I-16145 Genova - Italy

Abstract. The implementation of Machine Learning (ML) algorithms on stand-alone small-scale devices allows the incorporation of new services and advanced functionalities without the need of resorting to remote computing systems. Despite having undeniable advantages with respect to conventional general-purpose devices, e.g. in terms of cost/performance ratios, small-scale systems suffer of issues related to their resource-limited nature, like limited battery capacity and processing power. In order to deal with such limitations, we propose to merge local Rademacher Complexities and bit-based hypothesis spaces to build thrifty models, which can be effectively implemented on small-scale resource-limited devices. Experiments, carried out on a smartphone in a Human Activity Recognition application, show the benefits of the proposed approach in terms of model accuracy and battery duration.

1 Introduction

The learning process consists in aprioristically selecting an appropriate hypothesis space and, then, in choosing the most suitable model from it [1]. These two steps lead to: an *approximation* error, depending on the (non-optimal) choice of the hypothesis space; an *estimation* error, due to the finite number of available observations [2]. The approximation and estimation errors have been widely investigated throughout the last decades and effective theoretical procedures have been designed to deal with them. When resorting to real-world applications, however, further constraints arise, related to implementation restrictions. In small scale learning on resource-limited devices (e.g. smartphones, embedded systems, etc.), for example, power consumption and thermal dissipation lead to preferring fixed-point apps to floating-point ones to avoid battery draining or device overheating [3, 4]. In addition, new challenges in designing the learning process, such constraints introduce an *implementation* error, related to the restrictions of realizations on real-world devices. The investigation of these aspects from both a theoretical and a practical point of view is thus necessary to design procedures, which allow to best cope with these phenomena.

In this paper we deal with issues related to applying learning procedures in small scale learning on resource-limited devices where a limited number of bits can be exploited to describe and implement a model. While this problem has been usually tackled by adapting models to fit computational constraints *a-posteriori* (namely, after the learning process concluded), we reverse the perspective by acting on the whole learning process, starting from the definition of

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

the hypothesis space. For such purposes, we mostly rely on two main pillars, i.e. two ideas that emerged in recent literature. The first result [5, 6, 7] allows proving that enhancements can be obtained if the hypothesis space is *local*: namely, it consists of only those functions, that are most likely to be chosen by the learning procedure. The second pillar result [8] shows that exploiting a limited representation when defining an hypothesis space is equivalent to introducing regularization in the learning process: in fact, using few bits allows reducing both noise and the number of functions included in the hypothesis space. In this work we propose a novel learning procedure based on these pillars: by exploiting a representation relying on few bits, it allows to describe a limited number of functions, chosen so to foster locality. The derived advantages are threefold: the learning process is implicitly regularized by the use of a limited representation; model performance is improved thanks to the locality of the hypothesis space and physical/computational constraints hold, as only few bits are exploited.

2 Learning with Limited Resources

In this section, we recall the standard supervised learning framework [5] and, then, we properly modify it so to introduce the implementation constraints, typical of small scale learning with resource-limited devices.

Our goal is to approximate the relationship between inputs from a set $\mathcal{X}$ and outputs from a set $\mathcal{Y}$, which is encoded by a fixed, but unknown, probability distribution μ over $\mathcal{X} \times \mathcal{Y}$. The learning algorithm maps a set of labeled samples $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ to $h \in \mathcal{H}$ ($X \in \mathcal{X}$ and $Y \in \mathcal{Y}$) and the accuracy in representing the hidden relationship μ is measured with reference to a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$. The generalization error $L(h)$ is defined as $L(h) = \mathbb{E}_{(X,Y)}\ell(h(X), Y)$, where we assume that each labelled sample is independently generated according to μ.

Since μ is unknown, we can only compute its empirical estimate, i.e. the empirical error $\hat{L}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(X_i), Y_i)$. It is possible to prove that we can bound $L(h)$ with probability $(1 - \delta)$, $\forall h \in \mathcal{H}$, where $\mathcal{H}$ is the hypothesis space, by using empirical quantities only [5]:

$$\begin{aligned} L(h) \leq & c_1 \hat{L}_n(h) + c_2 r^* + \phi_1(\delta, n) \\ r^* : r = & c_3 \hat{\mathcal{R}}_n \left\{ \mathcal{H}^{(*,h_0)} : \ \hat{L}_n(h) \leq 2r \right\} + \phi_2(\delta, n) \end{aligned} \tag{1}$$

where c_1, c_2, c_3 are computable constants, while ϕ_1, ϕ_2 are functions that depend only on the level of confidence [5]. Moreover, $\hat{\mathcal{R}}_n \left\{ \mathcal{H}^{(*,h_0)} : \ \hat{L}_n(h) \leq 2r \right\}$ is the Local Rademacher Complexity (LRC) term, that can be computed as follows:

$$\hat{\mathcal{R}}_n \left\{ \mathcal{H}^{(*,h_0)} : \ \hat{L}_n(h) \leq 2r \right\} = \mathbb{E}_{\sigma_1,\ldots,\sigma_n} \sup_{\substack{h \in \mathcal{H}^{(*,h_0)} \\ \hat{L}_n(h) \leq 2r}} \frac{2}{n} \sum_{i=1}^{n} \sigma_i \ell(h(X_i), Y_i) \tag{2}$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

where $\sigma_1, \ldots, \sigma_n$ are independent uniform $\{\pm 1\}$-valued random variables. $\mathcal{H}^{(*,h_0)}$ is the hypothesis space, derived by star–shaping $\mathcal{H}$ around h_0:

$$\mathcal{H}^{(*,h_0)} = \{h_0 + \alpha(h - h_0) : \ h \in \mathcal{H}, \alpha \in [0,1]\} \tag{3}$$

$$\mathcal{H}^{*} = \mathcal{H}^{(*,0)} = \{\alpha h : \ h \in \mathcal{H}, \alpha \in [0,1]\}\,. \tag{4}$$

The bound of Eq. (2) contemplates only those functions in $\mathcal{H}$ that will be likely chosen by the learning process to assess the performance of a model. Consequently, according to the Structural Risk Minimization (SRM) principle [1], we can define a nested series of hypothesis spaces of increasing size $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots$ and select:

$$h^{\text{opt}} : \quad \arg \min_{h \in \mathcal{H}_i \in \{\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots\}} c_1 \hat{L}_n(h) + c_2 r^*(\mathcal{H}_i) + \phi_1(\delta, n). \tag{5}$$

As a last step, we devote the last part of this section to the introduction of implementation constraints of real-world applications, as discussed in the introduction. For this purpose, we exploit the results in [8], by limiting our analysis to the binary classification case: $\mathcal{X} \in \mathbb{R}^d$, so that $X = \boldsymbol{x}$, and $\mathcal{Y} \in \{\pm 1\}$, so that $Y = y$. $\mathcal{H}$ consists of functions, which are parametrized as follows: $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$, $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$ and $\boldsymbol{w} \in \mathbb{R}^D$. Moreover, the trimmed hinge loss $\ell_T(h(\boldsymbol{x}), y) = \min[1, \max[0, 1 - yh(\boldsymbol{x})]]$ is used as loss function, which allows to perform regularization in the learning process.

According to the SRM principle, we start by defining the nested series of hypothesis spaces as $\mathcal{H} = \{\boldsymbol{w} : \ \|\boldsymbol{w}\|_2^2 \leq w_{\text{MAX}}^2\}$, where w_{MAX}^2 is the hyperparameter that adjusts the size of the class of functions. It is worth underlining that $\mathcal{H}$ is star–shaped around zero by definition, i.e. $\mathcal{H} = \mathcal{H}^{(*,0)} = \mathcal{H}^*$. Given the restrictions of several real-world applications and implementations on resource-limited devices (e.g. embedded systems, smartphones, biomedical instruments, etc.), it is often convenient to resort to fixed-point arithmetics, in order to reduce the computational burden. A (possibly) limited number of bits should be then used in order to describe the hypothesis space and the functions it includes, for example by defining the following class of models:

$$\mathcal{H} = \left\{ \boldsymbol{w} : \left[\begin{array}{l} \|\boldsymbol{w}\|_2^2 \leq w_{\text{MAX}}^2 \\ w_j \in \frac{w_{\text{MAX}}}{2^\kappa - 1} \{-2^\kappa + 1, \ldots, 2^\kappa - 1\} \\ \sum_{i=1}^{D} [w_j \neq 0] \leq D\zeta \end{array} \right] \right\} \tag{6}$$

where κ is the number of bits. Moreover, a further constraint has been introduced to select a subset of features (ζ controls the maximum percentage of inputs to consider): in resource-limited applications, this is of importance to limit the computational burden by neglecting those features (e.g. sensors signals) that are of scarce influence on the overall classification process. It is clear that the size of the hypothesis space is controlled with the hyperparameters w_{MAX}, κ, and ζ.

As, unfortunately, the class of functions of Eq. (6) is not star–shaped, a further step is needed, i.e. we can impose star-shaping around zero:

$$\mathcal{H}^{*} = \{\alpha \boldsymbol{w} : \ \boldsymbol{w} \in \mathcal{H}, \alpha \in [0,1]\}\,. \tag{7}$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Thus, we can now reformulate the SRM optimization Problem (5) in order to exploit $\mathcal{H}^*$. We need to define some further quantities: $\Phi = [\boldsymbol{\phi}(\boldsymbol{x}_1)|\dots|\boldsymbol{\phi}(\boldsymbol{x}_n)]^T$, $Y = \text{diag}[y_1|\dots|y_n]$, $\boldsymbol{a}_n = [a_1|\dots|a_n]^T$, $\boldsymbol{y} = [y_1|\dots|y_n]^T$ and $\boldsymbol{\sigma} = [\sigma_1|\dots|\sigma_n]^T$. As we have to compute the two quantities of interest, namely $\min_{h\in\mathcal{H}} \hat{L}_n(h)$ and $\hat{\mathcal{R}}_n\left\{\mathcal{H}^* :\ \hat{L}_n(h) \leq 2r\right\}$, the corresponding minimization problem can be formulated as follows:

$$\min_{h\in\mathcal{S}, \frac{1}{n}\sum_{i=1}^{n} \ell_T(h(\boldsymbol{x}_i),y_i)\leq 2r} \sum_{i=1}^{n} -\sigma_i \ell(h(\boldsymbol{x}_i), y_i). \tag{8}$$

In order to compute $\hat{\mathcal{R}}_n\left\{\mathcal{H}^* :\ \hat{L}_n(h) \leq 2r\right\}$, we have $\mathcal{S} = \mathcal{H}^*$; instead, $\mathcal{S} = \mathcal{H}$ and $\sigma_i = -1,\ \forall i \in \{1,\dots,n\}$ when we are interested in deriving $\min_{h\in\mathcal{H}} \hat{L}_n(h)$. Consequently, the overall optimization problem can be formulated as:

$$\min_{\boldsymbol{w},\alpha,\boldsymbol{\xi}} \ -\boldsymbol{\sigma}^T \max[\mathbf{0}_n,\ \min[\mathbf{1}_n, \boldsymbol{\xi}]], \tag{9}$$

$$\text{s.t.} \begin{cases} \|\boldsymbol{w}\|_2^2 \leq w_{\text{MAX}}^2 \\ \boldsymbol{\xi} = \mathbf{1}_n - \alpha Y \Phi \boldsymbol{w} \\ \boldsymbol{w} \in \frac{w_{\text{MAX}}}{2^\kappa - 1}\{-\mathbf{2}_D^\kappa + \mathbf{1}_D, \dots, \mathbf{2}_D^\kappa - \mathbf{1}_D\} \\ \mathbf{1}_D^T [\boldsymbol{w} \neq \mathbf{0}_D] \leq D\zeta \\ \mathbf{1}_n^T \max[\mathbf{0}_n,\ \min[\mathbf{1}_n, \boldsymbol{\xi}]] \leq 2nr \end{cases} \tag{10}$$

where $\alpha = 1$ when searching for $\min_{h\in\mathcal{H}} \hat{L}_n(h)$.

Problem (9) is an NP-problem and, thus, we introduce a further error source, related to the impossibility of finding its global minimum. Nevertheless, the hypothesis space is designed so to consist of a limited number of functions, and thus it can be effectively explored [5]. Then, a good solution can be generally found by exploiting general purpose optimization tools, such as CPLEX [9], or the method proposed in [8]. It is worth underlining that Problem (9) is anyhow more complex to solve than other learning procedures (e.g. models training with the Support Vector Machine algorithm [1]): in other words, we move the computational burden to the learning phase, so to keep as light as possible the feed-forward running phase. This is not unusual where a trade-off between learning and model complexity is implemented and properly balanced.

3 Results & Discussion

We propose some preliminary results, obtained by applying the previously presented approach to the Human Activity Recognition on Smartphones (HARoS) [4, 10] dataset. The latter consists of six classes (1 – walking, 2 – walking upstairs, 3 – walking downstairs, 4 – sitting, 5 – standing, 6 – laying): in order to deal with two-class problems, we applied a One-vs-One (OvO) approach. Model selection was performed by varying w_{MAX} in the range $[10^{-3}, 10^2]$ among 10 values, equally spaced in a logarithmic scale; we also tested different values for $\kappa = \{8, 16, 32\}$ and $\zeta = \{0.1, 1\}$, for which we computed the class complexity r^*

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Table 1: Experimental results on the HARoS dataset.

	$\zeta = 0.1$								
	$\kappa = 8$			$\kappa = 16$			$\kappa = 32$		
Empirical error $\hat{L}(h)$, complexity r^*, Test set error $\hat{L}_T(h)$									
OvO	$\hat{L}$	r^*	$\hat{L}_T$	$\hat{L}$	r^*	$\hat{L}_T$	$\hat{L}$	r^*	$\hat{L}_T$
1 vs 2	4.06	1.41	9.32	4.06	1.57	9.32	3.90	2.39	9.14
1 vs 3	3.08	3.18	7.24	3.08	3.45	7.24	2.90	3.52	7.60
1 vs 4	0.00	2.22	0.29	0.00	2.99	0.29	0.00	2.99	0.35
1 vs 5	6.98	3.04	9.55	0.02	3.46	0.31	32.29	4.72	34.50
1 vs 6	0.00	4.13	0.00	0.00	5.56	0.00	0.00	8.36	0.00
2 vs 3	4.98	0.49	7.53	4.98	1.40	7.53	4.99	1.40	7.55
2 vs 4	0.01	0.49	0.35	0.01	1.94	0.35	0.00	4.75	0.32
2 vs 5	0.01	0.74	0.12	0.01	2.60	0.12	0.00	5.70	0.13
2 vs 6	0.00	2.29	0.00	0.00	3.50	0.00	0.00	5.69	0.00
3 vs 4	0.01	0.24	0.03	0.01	0.87	0.03	0.00	2.49	0.05
3 vs 5	0.00	0.44	0.00	0.00	0.97	0.00	0.00	3.63	0.00
3 vs 6	0.01	2.51	0.06	0.01	3.80	0.06	0.00	5.10	0.05
4 vs 5	13.40	0.71	17.42	13.40	0.71	17.42	12.81	0.80	16.37
4 vs 6	0.17	3.33	0.00	0.17	3.63	0.00	0.01	6.54	0.00
5 vs 6	0.00	1.63	0.00	0.00	1.83	0.00	0.00	3.60	0.00
Average prediction per seconds and battery life in hours.									
Predictions	4100			2700			330		
Battery Life	250			180			120		
	$\zeta = 1$								
	$\kappa = 8$			$\kappa = 16$			$\kappa = 32$		
Empirical error $\hat{L}(h)$, complexity r^*, Test set error $\hat{L}_T(h)$									
OvO	$\hat{L}$	r^*	$\hat{L}_T$	$\hat{L}$	r^*	$\hat{L}_T$	$\hat{L}$	r^*	$\hat{L}_T$
1 vs 2	3.90	1.58	9.14	3.90	1.75	9.18	3.90	2.75	9.18
1 vs 3	2.90	3.63	7.60	2.90	3.63	7.60	2.90	3.64	7.60
1 vs 4	0.00	2.75	0.35	0.00	3.16	0.35	0.00	3.75	0.35
1 vs 5	0.00	4.45	0.31	31.95	4.60	33.92	0.00	4.75	0.31
1 vs 6	0.00	7.50	0.00	0.00	8.35	0.00	0.00	8.50	0.00
2 vs 3	4.99	1.38	7.55	4.99	1.42	7.55	4.99	1.58	7.55
2 vs 4	0.00	1.94	0.32	0.00	4.55	0.32	0.00	4.57	0.32
2 vs 5	0.00	0.84	0.13	0.00	4.74	0.13	0.00	5.74	0.13
2 vs 6	0.00	2.29	0.00	0.00	5.62	0.00	0.00	5.64	0.00
3 vs 4	0.00	0.88	0.05	0.00	2.58	0.05	0.00	2.59	0.05
3 vs 5	0.00	1.03	0.00	0.00	4.25	0.00	0.00	4.27	0.00
3 vs 6	0.00	2.45	0.05	0.00	5.22	0.05	0.00	5.23	0.05
4 vs 5	12.81	0.80	16.37	12.80	0.80	16.35	12.80	0.80	16.35
4 vs 6	0.01	3.30	0.00	0.00	6.53	0.00	0.00	6.53	0.00
5 vs 6	0.00	1.48	0.00	0.00	3.82	0.00	0.00	3.82	0.00
Average prediction per seconds and battery life in hours.									
Predictions	715			442			67		
Battery Life	210			140			90		

accordingly. Table 1 reports the empirical error on the training set $\hat{L}(h)$ and on the test set $\hat{L}_T(h)$, in addition to r^*.

Table 1 also presents results showing how performance, in terms of predictions per second and battery life (in hours), changes as κ and ζ are varied: these results are averaged over the different OvO binary classification problems. These values were obtained by simulating the HARoS process on a Samsung Galaxy S II smartphone[1], where we implemented both fixed-point (8 and 16 bits) and floating-point (32 bits) procedures, in accordance with the explored values of κ. The time needed to perform a prediction was measured as the amount of milliseconds elapsed between data gathering from sensors and prediction computation. In order to estimate battery duration, instead, we fully charged the smartphone battery and, then, kept continuously running the application until a 10% level was reached.

[1]The smartphone is equipped with a Li-Ion 1650 mAh battery, with up to 610 hours of stand-by operation time, and mounts Android Gingerbread v2.3.4 operating system.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

By analyzing the results of Table 1, reducing the number of bits emerges as an appealing method to decrease the complexity of the hypothesis space while, at the same time, maintaining the capability of the trained models to generalize well on new and previously unseen data. The use of Local Rademacher Complexity plays a central role, as it allows to exclude from the hypothesis space those functions that will not be chosen by the learning procedure. The possibility, offered by the proposed approach, of more accurately shaping and designing the class of functions leads to remarkable positive outcomes when dealing with model accuracy, prediction rate, and battery lifespan in resource-limited devices.

References

[1] V. N. Vapnik. *Statistical learning theory*. Wiley–Interscience, 1998.

[2] E. De Vito, L. Rosasco, A. Caponnetto, U. D. Giovannini, and F. Odone. Learning from examples as an inverse problem. *Journal of Machine Learning Research*, 6:883–904, 2005.

[3] D. J. Cook and S. K. Das. Pervasive computing at scale: Transforming the state of the art. *Pervasive and Mobile Computing*, 8(1):22–35, 2012.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 437–442, 2013.

[5] P. L. Bartlett, O. Bousquet, and S. Mendelson. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

[6] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[7] D. Anguita, A. Ghio, L. Oneto, and S. Ridella. The impact of unlabeled patterns in rademacher complexity theory for kernel classifiers. In *Advances in Neural Information Processing Systems*, pages 585–593, 2011.

[8] Davide Anguita, Alessandro Ghio, Luca Oneto, and Sandro Ridella. A learning machine with a bit-based hypothesis space. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 467–472, 2013.

[9] User-manual cplex 12.6. IBM Software Group, 2014.

[10] K. Bache and M. Lichman. UCI machine learning repository, 2013.