3rd International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2020), 17-19 November 2021, Porto, Portugal

Oral

Topic: *Machine learning*

Binarization for Optical Processing Units via REINFORCE

B. Kozyrskiy [1], I. Poli[2], R. Ohana[2,3], L. Daudet[2], I. Carron[2], M. Filippone[1]
[1] Department of Data Science, EURECOM, 450 Route des Chappes, 06410 Biot, France
[2] LightOn, 2 rue de la Bourse, F-75002 Paris, France
[3]Laboratoire de Physique, Ecole Normale Supérieure, 24 rue Lhomond, 75005 Paris, France
E-mail: Bogdan.Kozyrskiy@eurecom.fr

Summary: Optical Processing Units (OPUs) are computing devices which perform random projections of input vectors by exploiting the physical phenomenon of scattering a light source through an opaque medium. OPUs have successfully been proposed to carry out approximate kernel ridge regression at scale and with low power consumption by the means of optical random features. OPUs require input vectors to be binary, and this work proposes a novel way to perform supervised data binarization. The main difficulty to develop a solution is that the OPU projection matrices are unknown which poses a challenge in deriving a binarization approach in an end-to-end fashion. Our approach is based on the REINFORCE gradient estimator, which allows us to estimate the gradient of the loss function with respect to binarization parameters by treating the OPU as a black-box. Through experiments on several UCI classification and regression problems, we show that our method outperforms alternative unsupervised and supervised binarization techniques.

Keywords: optimization, random features, linear regression, optical processing unit.

1. Motivation

Optical Processing Units (OPUs) are computing devices which perform random projections of input vectors by exploiting the physical phenomenon of scattering a light source through a diffusive medium [1]. The projection operation is particularly useful when approximating kernel functions via random features, a popular technique to implement these models for large-scale problems [2]. OPUs offer the possibility to obtain such approximations with a large number of random features at the speed of light and with low-power consumption, representing a very attractive line of work to further improve scalability of kernel machines. As an example, OPU-based random feature approximations have successfully been proposed to carry out approximate kernel ridge regression in [3].

The main limitations on the generality of this approach are that OPUs require input vectors to be binary and that OPU projection matrices are unknown and can only be retrieved through an expensive calibration procedure. Common approaches for optimization of binarized neural networks, like straight-through estimator or different kinds of a relaxation of the binarization procedure, can be found in the literature on neural networks, where existing methods rely on the possibility to propagate gradient through all operations of the network except binarization [4]. In the literature, there are approaches which address binarization by considering it as a pre-processing step, which happens independently of the regression/classification task [5]. In this case label information is omitted, and this might be suboptimal compared to strategies that take this information into account in the binarization phase.

In this paper, we propose a novel binarization method for OPUs which is learned along with the regression/classification task in an end-to-end manner. We overcome the main challenge to develop such an end-to-end solution, which is that OPU projection matrices are unknown, by employing the so-called REINFORCE gradient estimator. This allows us to estimate the loss function gradient with respect to binarization parameters by treating the OPU as a black-box. Through experiments on several UCI classification/regression problems, we show that our proposal outperforms alternative unsupervised and supervised binarization techniques.

2. Related work

In neural networks, binarization is generally targeting intermediate layer activations, and it may also stem from binarization of model parameters; in these cases, binarization is mostly introduced to reduce computational cost and memory consumption [6]. Neural networks with binary hidden layers find applications in binary autoencoders for hashing [7], data compression [5], and hard attention mechanism [8]. The binarization of layer activations is obtained by a suitable choice of activation functions; for instance, the sign or Heaviside functions for the deterministic case, or the sigmoid or tanh functions combined with the Bernoulli distribution for the stochastic case [4], [9]. The most popular technique to propagate gradients through such activation functions is the so called straight-through estimator (STE) [10]. More recently, there have been proposals to replace the STE with another estimator through a relaxation technique, also known as the Gumbel Softmax-trick [11]. Also,

different kinds of target propagation are used to learn suitable targets for each binary layer and then train the associated parameters with relaxation techniques or combinatorial optimization [12], [13], [14].

In this work, we aim to develop a supervised binarization model which is learned together with the supervised learning task. That is, we aim to provide a training procedure for the heterogeneous model consisting of the kernel ridge regression model approximated with random features and the binarization encoder before the OPU. In this context, a general-purpose framework called Method of Auxiliary Coordinates (MAC) was proposed in [14] with examples of application in [7] and [15]. The authors propose to introduce auxiliary variables into a deep neural network. These auxiliary variables are assigned the role of pre-activations for each layer, and they get replaced during the forward pass. The first step of the optimization targets the auxiliary variables, and, after this step, the parameters of each layer are optimized to regress on these variables, which take the role of layer-specific labels. This is very beneficial when some layers are discrete and vanilla backpropagation is not applicable. In [15], this approach is used to train a fully connected network with binary activation functions, using a STE to propagate a learning signal through the non-differentiable parts. Reference [7] is especially interesting because authors illustrate, how discrete binary layers can be optimized withing larger, non-binary model.

While splitting the optimization of the binarization and the model is a viable option, we still need a way to training each part individually. There is a wide variety of ways to obtain a solution for kernel ridge regression with the random feature approximation, so the most difficult point is how to optimize the part consisting of the binary encoder and the OPU, because it combines a non-differentiable function with an implicit random projection. These make the STE from [15] inapplicable. Also, we found that the combinatorial approach used in [7] and [12] is inapplicable for our case for two reasons. First, it is suitable only when the binary dimension is relatively small, which might be a limitation for a general solution. Second, the combinatorial approach combined with MAC converges in one iteration to poor local optima, and this happens because of the model setup which is different from the ones in [7] and [12].

From a different point of view, it is possible to view our problem through the lenses of reinforcement learning, where it is necessary to propagate binary codes through the OPU instead of discrete actions through the black-box environment. Instead of maximizing the reward from the environment, we are trying to minimize the loss function. The classical algorithm to solve this problem is REINFORCE [16]. This allows one to calculate gradients of the reward with respect to parameters of the policy that generates actions. The applicability of this method to other settings with black-box elements was shown in [17]. There are various versions of this algorithm intended to reduce variance of the gradient of the parameters. Very frequently they are based on relaxations of the non-differentiable sampling procedure [18], or approximation of the black-box part of the model [19]. It also worth noting that there exist competitive alternatives to REINFORCE, such as the one in [20], later extended with variance reduction [21] or relaxation [22].

3. Methods

In this paper we consider the kernel ridge regression model. Let $X = x_1, \dots, x_n$ a set of input vectors $x_i \in \mathbb{R}^d$ and let $Y = y_1, \dots y_n$ a set of corresponding binary labels. The aim of kernel ridge regression is to establish a mapping between the inputs and the labels by means of functions which belong to the so-called Reproducing Kernel Hilbert Space (RKHS) induced by the choice of a kernel function $k(\cdot,\cdot)$ [23].

Given a choice of kernel function, kernel ridge regression requires evaluating it among all possible pairs of inputs, yielding an nxn matrix K such that $K_{ij} = k(x_i, x_j)$. The solution of kernel ridge regression requires performing algebraic operations with K, and this is problematic when n is large.

A way to avoid these computations and scale kernel ridge regression to large data is to use an approximation based on random features [2]. In this work, we focus in particular on random features produced by OPUs.

OPU performs multiplication of a binary vector $x \in \mathbb{R}^d$ by a random matrix and applies the activation function $|\cdot|^2$.

$$\phi(x) = \frac{1}{\sqrt{D}}|Rx|^2 \qquad (1)$$

Where $R \in \mathbb{C}^{D\times d}$ is a complex Gaussian matrix with elements $R_{ij} \sim \mathcal{CN}(0,1)$. Performing regression on a linear model using these new random features in (1) gives equivalent results to the original kernel ridge regression problem when $D \to \infty$.

$$y^* = W^*\phi(x) \qquad (2)$$
$$W^* = \underset{W}{\operatorname{argmin}} ||\phi(X)W^T - Y||_2^2 + \frac{\lambda}{2}||W||_2^2$$

for the training set X, Y. Model (2) is equivalent to the ridge kernel regression with a kernel [3].

$$k(\mathbf{x},\mathbf{y}) \approx \phi(\mathbf{x})\phi(\mathbf{y}) \overset{D\to\infty}{=} ||\mathbf{x}||^2||\mathbf{y}||^2 + (\mathbf{x}^T\mathbf{y})^2 \qquad (3)$$

We propose to perform the binarization of the input to this model by means of an encoder with parameters W_{enc}. The output of the encoder parameterizes a multidimensional Bernoulli distribution from which

we sample binary vectors and use them as a binary representation of the input data. So, our regression model becomes:

$$\tilde{y} = \mathbb{E}_z[W_{\text{regr}}\phi(z)] \quad \text{where } z \sim \text{Bernoulli}(f(x, W_{\text{enc}})) \tag{4}$$

where W_{regr} are parameters of the linear regression, z is binary representation of the data, $\phi(z)$ are random features. Parameters of the Bernoulli distribution are generated from the input data x by the encoder function f with parameters W_{enc}.

The stochasticity is intentionally introduced to the encoder so that we can employ the so-called REINFORCE gradient estimator. The REINFORCE approach (also called log-derivative trick or score function estimator) aims to estimate the gradient of the expectation of some non-differentiable function f subject to parameters of the distribution of the random variable z:

$$\nabla_\theta \mathbb{E}_{p(z;\theta)} f(z) \approx \frac{1}{M}\sum_{i=1}^{M} \nabla_\theta log\ p(z;\theta) f(z) \tag{5}$$

where M is number of samples drown from $p(z, \theta)$. For our model, the optimization objective becomes:

$$\min_{W_{\text{regr}}, W_{\text{enc}}} \mathbb{E}_{z\sim \text{Bernoulli}(f(x,W_{\text{enc}}))}[\mathcal{L}(Y, W_{\text{regr}}\phi(z))] + \\ +\lambda_{\text{enc}}||W_{\text{enc}}||^2 + \lambda_{\text{regr}}||W_{\text{regr}}||^2 \tag{6}$$

where $\mathcal{L}\left(Y, \tilde{Y}\right)$ is the quadratic loss for regression problems and the cross-entropy loss for classification problems. The gradient of the first term with respect to W_{enc} becomes:

$$\nabla_{W_{\text{enc}}} \mathbb{E}_{z\sim q(z)}[\mathcal{L}(Y, W_{\text{regr}}\phi(z))] \approx \\ \approx \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\,(Y, W_{\text{regr}}\phi(z_i)) \nabla_{W_{\text{enc}}} \log\ q(z_i) \tag{7}$$

In order to reduce the variance of this estimator, we can use control variates as proposed in [21]:

$$\nabla_{W_{\text{enc}}} \mathbb{E}_{z\sim q(z)} \left[\mathcal{L}\left(Y, W_{\text{regr}}\phi(z)\right)\right] \approx \\ \approx \frac{1}{M}\sum_{i=1}^{M} \nabla_{W_{\text{enc}}} \log q\,(z_i) \left(\mathcal{L}\left(Y, W_{\text{regr}}\phi(z_i)\right) - v_i\right) \\ \text{where } v_i = \frac{1}{M-1}\sum_{i\neq j} \mathcal{L}\,(Y, W_{\text{regr}}\phi(z_j)) \tag{8}$$

Thanks to REINFORCE, we are able to optimize the encoder in an end-to-end fashion. In the remainder of this paper, we refer to this method as End-to-End SE.

In the End-to-End SE in order to estimate the gradient of the loss with respect to W_{enc} it is necessary to pass multiple samples from the encoder through the random projection and the approximate kernel ridge regression model. Depending on the number of random features used for the approximation, this operation can be expensive. To alleviate this computational burden, we propose a variation on the End-to-End SE where we propagate samples only through the random projections layer and then we average them before feeding them to the final linear operation.

$$\tilde{y} = W_{\text{regr}}\mathbb{E}_z[\phi(z)] \quad \text{where } z \sim \text{Bernoulli}(f(x, W_{\text{enc}})) \tag{9}$$

The optimization objective in this case becomes:

$$\min_{W_{\text{regr}}, W_{\text{enc}}} \mathcal{L}(Y, W_{\text{regr}}\mathbb{E}_{z\sim \text{Bernoulli}(f(x,W_{\text{enc}}))}[\phi(z)]) + \\ +\lambda_{\text{enc}}||W_{\text{enc}}||^2 + \lambda_{\text{regr}}||W_{\text{regr}}||^2 \tag{10}$$

So, the gradient of the first term with respect to encoder parameters becomes:

$$\nabla_{W_{\text{enc}}} \mathcal{L} = \frac{d\mathcal{L}}{d(\mathbb{E}\phi(z))} \nabla_{W_{\text{enc}}} \mathbb{E}(\phi(z)) \tag{11}$$

where $\nabla_{W_{\text{enc}}} \mathbb{E}(\phi(z))$ calculated with REINFORCE estimator. We will refer to this method as Isolated Supervised Encoder.

4. Results

We compared the performance of the proposed approaches (End-to-End SE and Isolated SE) against a model based on unsupervised autoencoder proposed in [5], encoder trained with direct feedback alignment (DFA) [23] and Gaussian process (GP) regression based on radial basis function (RBF) kernel over raw and binarized data. Results are reported in Fig. 1on several UCI regression and classification problems [24]. We want to emphasize that the main competitors of the proposed methods are the ones based on unsupervised autoencoder and encoder trained by DFA, because kernel ridge regression is unable to work with large datasets, and OPU-based regression just approximates this method and is intended to replace it on large datasets.

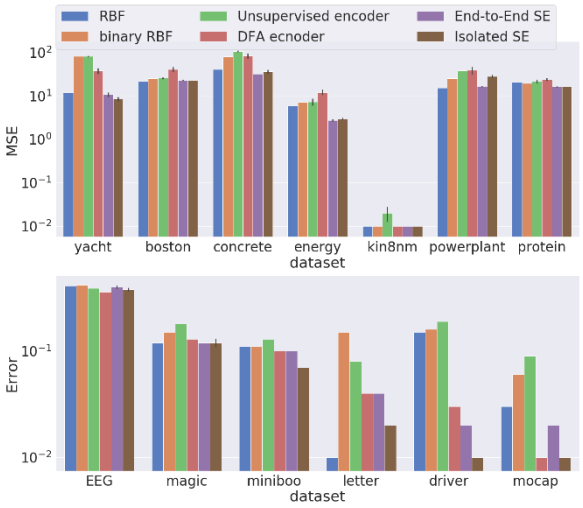

Fig. 1: Mean squared error (MSE) for regression (top) and negative error on classification (bottom) datasets comparison.

For Isolated SE and End-to-End SE as an encoding function $f(x, W_{\text{enc}})$ providing parameters for the Bernoulli, distribution we chose a single linear layer with a sigmoid activation:

$$f(x, W_{\text{enc}}) = \sigma(W_{\text{enc}} x) \tag{12}$$

All hyperparameters for the DFA encoder, End-to-End SE and Isolated SE models (size of binary embedding, learning rate, l2 regularization for the encoder and the regression layer, number of training epochs) were chosen with a random search during cross-validation.

In the comparison of binarization strategies we also include Gaussian processes on the original inputs and on the inputs binarized using unsupervised techniques, and we denote these two methods by RBF and binary RBF, respectively. To apply GP-based regression to the two-classes classification problems we represented class labels as -1, 1 and solved a classification problem as a regression one directly. In these cases, GP parameters were tuned by marginal likelihood maximization. This poses computational challenges for large datasets (MiniBoo, MoCap), so we resort to random feature approximations for these cases.

For the models involving random features (both Fourier and OPU-generated ones) we have tuned the variance of the distribution that generates these random features. Concretely, assuming that the elements of the R matrix generating the random projections are distributed through the standard Normal distribution, we can obtain a new random matrix R' by multiplying R by any variance, for instance:

$$\phi'(x) = c|R'x|^2 = c|\frac{R}{\sigma}x|^2 = c\frac{1}{\sigma^2}|Rx|^2 \tag{13}$$

with corresponding kernel:

$$k(\mathbf{x}, \mathbf{y}) = \frac{1}{\sigma^4}(||\mathbf{x}||^2||\mathbf{y}||^2 + (\mathbf{x}^T\mathbf{y})^2) \tag{14}$$

So, it is enough to multiply the output of the OPU by an additional set of parameters γ, such that $\gamma^2 = \frac{1}{\sigma^2}$, and optimize them with standard gradient descent. The parameter gamma is γ is not equivalent to the lengthscale parameter of the RBF kernel as it has simply a scaling effect on the kernel.

On the regression problems, both proposed methods outperformed their main competitors. On the classification problems, the DFA-based approach was better only on one dataset, and on all other datasets the proposed methods performed better or equally well. Considering the comparison between the proposed methods, we see that End-to-End SE is more stable and requires a significantly fewer number of samples from the encoder, although Isolated SE showed slightly better results on classification problems. We considered including results obtained by running these models on the real OPU (Fig. 2). Unfortunately, the regression problems required such a large number of epochs that we could not perform the experiments in a reasonable amount of time.

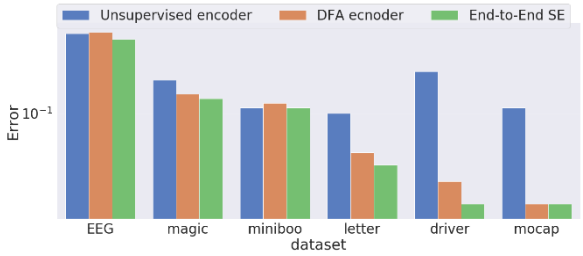

Fig. 2: Error comparison on classification (bottom) datasets for experiments on a real hardware.

We also tested the performance of our approach with respect to the number of samples required to employ REINFORCE. We found that End-to-End SE can achieve good results with a small number of samples from the encoder, and the increase of number of samples does not seem to improve performance. In Fig. 3 we plot the convergence of the loss for one classification and one regression problem. The convergence curves indicate that the convergence speed benefits from the gradient variance reduction.

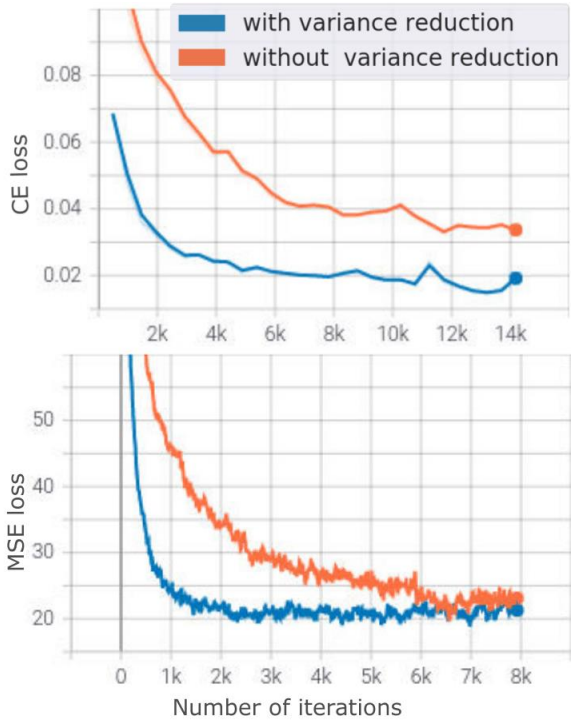

Fig. 3: Convergence of the training procedure on classification problem: mocap dataset (top) and regression problem: boston dataset (bottom).

5. Conclusion

We proposed a method inspired by reinforcement learning that allows us to use OPUs for approximately solving kernel ridge regression on real-valued data. We have empirically shown that gradient-based supervised optimization of the binarization part is beneficial compared to unsupervised binarization and strategies that do not employ gradient information.

References

[1] A. Saade, F. Caltagirone, I. Carron, L. Daudet, A. Drémeau, S. Gigan and F. Krzakala, "Random projections through multiple optical scattering: Approximating kernels at the speed of light," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[2] A. Rahimi and B. Recht, "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems*, 2009.

[3] R. Ohana, J. Wacker, J. Dong, S. Marmin, F. Krzakala, M. Filippone and L. Daudet, "Kernel computations from large-scale random features obtained by optical processing units," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[5] J. Tissier, C. Gravier and A. Habrard, "Near-lossless binarization of word embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[6] H. Qin, R. Gong, X. Liu, X. Bai, J. Song and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, 2020.

[7] M. A. Carreira-Perpinán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[8] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015.

[9] J. W. T. Peters and M. Welling, "Probabilistic binary neural networks," *arXiv preprint arXiv:1809.03368*, 2018.

[10] Y. Bengio, N. Léonard and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[11] E. Jang, S. Gu and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[12] A. L. Friesen and P. Domingos, "Deep learning as a mixed convex-combinatorial optimization problem," *arXiv preprint arXiv:1710.11573*, 2017.

[13] D.-H. Lee, S. Zhang, A. Fischer and Y. Bengio, "Difference target propagation," in *Joint european conference on machine learning and knowledge discovery in databases*, 2015.

[14] M. Carreira-Perpinan and W. Wang, "Distributed optimization of deeply nested systems," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Reykjavik, 2014.

[15] A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani and others, "Beyond backprop: Online alternating minimization with auxiliary variables," in *International Conference on Machine Learning*, 2019.

[16] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, p. 229–256, 1992.

[17] R. Ranganath, S. Gerrish and D. Blei, "Black box variational inference," in *Artificial intelligence and statistics*, 2014.

[18] G. Tucker, A. Mnih, C. J. Maddison, D. Lawson and J. Sohl-Dickstein, "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models," *arXiv preprint arXiv:1703.07370*, 2017.

[19] W. Grathwohl, D. Choi, Y. Wu, G. Roeder and D. Duvenaud, "Backpropagation through the void: Optimizing control variates for black-box gradient estimation," *arXiv preprint arXiv:1711.00123*, 2017.

[20] M. Yin and M. Zhou, "ARM: Augment-REINFORCE-merge gradient for stochastic binary networks," *arXiv preprint arXiv:1807.11143*, 2018.

[21] W. Kool, H. van Hoof and M. Welling, "Buy 4 REINFORCE Samples, Get a Baseline for Free!," *Deep Reinforcement Learning Meets Structured Prediction Workshop at the International Conference on Learning Representations*, 2019.

[22] Z. Dong, A. Mnih and G. Tucker, "DisARM: An antithetic gradient estimator for binary latent variables," *arXiv preprint arXiv:2006.10680*, 2020.

[23] K. P. Murphy, Machine learning: a probabilistic perspective, MIT press, 2012.

[24] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," *arXiv preprint arXiv:1609.01596*, 2016.

[25] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2017.