A supervised strategy for deep kernel machine

Florian Yger, Maxime Berar, Gilles Gasso and Alain Rakotomamonjy*

LITIS EA 4108 - Université de Rouen / INSA de Rouen,
76800 Saint Etienne du Rouvray - France

Abstract. This paper presents an alternative to the supervised KPCA based approach for learning a Multilayer Kernel Machine (MKM) [1]. In our proposed procedure, the hidden layers are learnt in a supervised fashion based on kernel partial least squares regression. The main interest resides in a simplified learning scheme as the obtained hidden features are automatically ranked according to their correlation with the target outputs. The approach is illustrated on small scale real world applications and shows compelling evidences.

1 Introduction

Deep architectures represent a prominent research area in machine learning. However, this approach was for a long time penalized by its inability to derive an efficient learning method leading to a model with good generalization skills. In that sense, shallow architecture such as kernel methods [2] were preferred. Recent attention devoted to deep architectures stems from the so-called *deep learning* strategy [3]. This strategy consists in layer-wise unsupervised pre-training followed by a final supervised tuning [4, 5]. Doing so allows the hidden layers to learn useful internal representations of data.

Another trend of deep learning consists in pre-training the hidden layers of a deep model in a supervised way. This idea pursued in the 90's was declined in [6] where internal features were learnt by minimizing a Fisher discriminant criterion. In the same streamline, Cho and Saul [1] has proposed the multilayer kernel machines (MKM) which represents one attempt to bridge the gap between deep architectures and kernel methods. First, these authors have proposed a new family of kernel functions which mimic the behavior of an infinite size deep architecture and turn a neural network into a shallow structure trainable under the convex optimization framework of SVMs. The so-designed kernel was also applied to train the hidden layers of a deep network.

In the procedure highlighted by Cho and Saul, the hidden layers are greedily trained by first applying a kernel PCA projection followed by features selection based on mutual information with class labels and supervised cross-validation. The obtained model proves competitive with usual deep architectures. However, a major drawback is the complication of features selection stage which involves a not easy setup of many hyper-parameters.

The main contribution of our paper it to alleviate this drawback by learning each hidden layer using kernel partial least squares regression (KPLS) [7]. The

*This work is funded in part by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886 and by the French ANR Project ASAP ANR-09-EMER-001.

interest of our proposal resides in a simplified learning scheme as the features provided by KPLS are automatically ranked according to their covariance with the target outputs. Therefore, we do not need to resort to any extra cross-validation. As a side effect, the learnt features are more informative as shown by the empirical evidences in Section 4.

2 Deep Kernel Machines

Assume a discriminative task with samples set $\mathcal{D} = \{(x_i, y_i) \in \mathbb{R}^q \times \mathbb{R}^p\}_{i=1}^N$. Deep kernel machines paradigm aims at learning useful features by stacking successive layers. Each layer is issued from a nonlinear projection of output of the previous layer using kernel method. Given a sample x the output of the ℓ-th layer is the size m_ℓ vector $\mathbf{h}^\ell(x) = g(\mathbf{h}^{\ell-1}(x))$ where g is a nonlinear function. Specifically, the j-th value of this vector is expressed as

$$\mathbf{h}_j^\ell(x) = \langle \mathbf{u}_j,\ \phi\left(\mathbf{h}^{\ell-1}(x)\right)\rangle_{\mathcal{F}} \tag{1}$$

with ϕ a mapping function which projects any input into the feature space $\mathcal{F}$ and $\mathbf{u}_j \in \mathcal{F}$. As usual in kernel methods, $\mathbf{u}_j$ is expanded over the set of basis functions $\{\phi\left(\mathbf{h}^{\ell-1}(x_i)\right)\}_{i=1}^N$ and reads

$$\mathbf{u}_j = \sum_{i=1}^N \alpha_i^j \phi\left(\mathbf{h}^{\ell-1}(x_i)\right), \quad \forall\ j = 1, \cdots, m_\ell \tag{2}$$

where coefficients α_i^j are to be sought at each stage of deep kernel machines learning. From equations (1) and (2), we obtain $\mathbf{h}_j^\ell(x) = \sum_{i=1}^N \alpha_i^j \mathbf{k}\left(\mathbf{h}^{\ell-1}(x), \mathbf{h}^{\ell-1}(x_i)\right)$ where $\mathbf{k}$ stands for the kernel function. Once the last layer is built, the resulting features are used to train any classifier.

Beyond the selection of the depth of the architecture, the issues of deep kernel machines concern how to learn the coefficients vector $\boldsymbol{\alpha}^j$ in (2) and to specify the size of each hidden unit. The next section is devoted to the presentation of existing solution and our proposed methodology.

3 Training of Multilayer Kernel Machines (MKM)

As mentioned previously, the architecture is learned layer by layer. The general framework is given in Algorithm 1.

Existing solution [1] is a compound method that combines an unsupervised learning followed by supervised model selection. In contrary, the approach we promote here is simple, efficient and relies on KPLS, a supervised method. Details about these methods are given below.

3.1 KPCA based MKM training

To learn each layer ℓ, the solution promoted by Cho and Saul [1] computes N principal directions $\mathbf{u}_j$ from which the best m_ℓ informative components are selected. According to [2, Chapter 14], $\mathbf{u}_j$ is an eigenvector and the corresponding

Algorithm 1 Learning of Multilayer Kernel Machine

```
[optional] Feature selection on raw data
for every layer $\ell$ do
    Compute the principal directions $\mathbf{u}_j$ (2) and corresponding $\mathbf{h}_j^\ell(x_i)$
    [optional] Rank and select the best principal directions
    Output the representation $\mathbf{h}^\ell(x_i)$ for each training sample $x_i$
end for
Apply any off-the-shelves classifier using features of the last layer
```

coefficients $\boldsymbol{\alpha}^j$ are solution of the eigenvalue problem $\lambda_j \boldsymbol{\alpha}^j = \mathbf{K}_\ell \boldsymbol{\alpha}^j$ where λ_j is the j-th eigenvalue of the centered kernel matrix $\mathbf{K}_\ell = (\mathbf{I} - \mathbf{M})\, \tilde{\mathbf{K}}_\ell\, (\mathbf{I} - \mathbf{M})$ with $(\tilde{\mathbf{K}}_\ell)_{ij} = \mathbf{k}\left(\mathbf{h}^{\ell-1}(x_i), \mathbf{h}^{\ell-1}(x_j)\right)$ and $\mathbf{M}$ a square matrix with entries all equal to $1/N$. Once the N principal components (features) are extracted, only the most interesting are retained. For this sake, the algorithm builds class-conditional and marginal histograms to estimate feature's mutual information with the class labels and re-ranks the features accordingly. Then a kNN classifier is applied on the first m_ℓ features and the validation error is recorded. This step is repeated for different values of k and m_ℓ in order to select the best m_ℓ features.

This procedure deems quite complicated as it involves at each layer first an unsupervised learning and then a trial and error approach to find the best features. The method can be simplified considerably in the sense that the cross-validation step can be rendered optional, depending on the used kernel projection. Our contribution is derived in that simplification sense and relies on KPLS strategy without validation step. This matter is developed hereafter.

3.2 Supervised MKM learning

We propose to get rid of the feature selection and embed it in the projection method. Therefore, the optional steps of algorithm 1 are no longer necessary. This is achieved through kernel partial least squares procedure.

Partial Least Squares method (PLS) computes latent variables based upon the eigenvectors of the empirical covariance matrix between the input and the output variables. Extension to kernel framework were proposed and has been used in the context of multiclass classification (see [7] for full details).

To train each layer of the MKM in a supervised way, KPLS searches for the j-th feature $\left[\mathbf{h}_j^\ell(x_1) \cdots \mathbf{h}_j^\ell(x_N)\right]^\top$ the most correlated with the class labels. Hence, to obtain $\mathbf{u}_j$ in Equation 2, one solves the eigenvalue problem $\lambda_j \boldsymbol{\alpha}^j = \mathbf{Y}\mathbf{Y}^\top \mathbf{K}_\ell^j \boldsymbol{\alpha}^j$ (for corresponding vector $\boldsymbol{\alpha}^j$) with $\mathbf{K}_\ell^j$ obtained by deflating $j-1$ times the matrix $\mathbf{K}_\ell$. In the previous relation, $\mathbf{Y} \in \mathbb{R}^{N \times p}$ stands for the class labels matrix. The deflation process aims at making the matrix $\mathbf{K}_{j+1}^\ell$ orthogonal to the subspaces generated by the previous features α_j, as the information modelled by the feature α_j should not be used in the successive features : $\mathbf{K}_\ell^{j+1} = \mathbf{H}_\ell^j \mathbf{K}_\ell^j \mathbf{H}_\ell^j$ where $\mathbf{H}_\ell^j = \mathbf{I} - \frac{\mathbf{K}_\ell^j \boldsymbol{\alpha}^j {\boldsymbol{\alpha}^j}^\top \mathbf{K}_\ell^j}{{\boldsymbol{\alpha}^j}^\top {\mathbf{K}_\ell^j}^2 \boldsymbol{\alpha}^j}$. The next feature is extracted by repeating the

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

procedure with the deflated matrix $\mathbf{K}_\ell^{j+1}$.

This scheme considerably simplifies informative features extraction. Indeed, as the features generated by KPLS are obtained iteratively and maximize their covariance with the class-labels, a good guess of the importance ranking of the learnt representations is directly obtained through the eigenvalue λ_j. Features number selection is achieved by thresholding these eigenvalues (in practice at a value of 10^{-15}). We coded $\mathbf{Y}$ in a *one-against-one* strategy. Finally, as in [1], we built on top of the last layer a Large Margin Nearest Neighbor classifier [8].

4 Experiments

In this section, we evaluate our learning approach on three multiclass datasets. The first two datasets were created from the MNIST handwritten digit dataset and were used as a benchmark by Cho and Saul [1] for their MKM. Finally, as a proof of concept, our approach is tested on a texture recognition task.

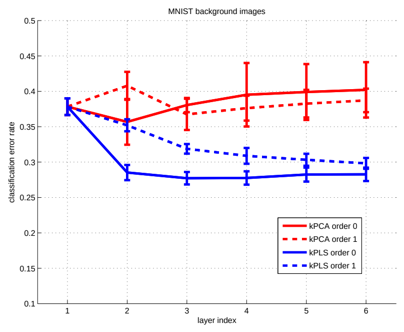
Although any non-linear kernel could be applied, we follow Cho and Saul and use arc-cosine kernel designed to mimic the behavior of an infinite size neural network. Its expression is given by $\mathbf{k}_n(x,z) = \frac{1}{\pi}\|x\|^n\|z\|^n J_n(\theta)$ with $J_n(\theta) = (-1)^n(\sin\theta)^{2n+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n\left(\frac{\pi-\theta}{\sin\theta}\right)$, the angular dependency $\theta = \cos^{-1}\left(\frac{\langle x,z\rangle}{\|x\|\|z\|}\right)$ and n the order of the kernel (see [1] for details). Note that due to ill-conditioning and stability issues, we did not explore order $n > 1$. Finally, let say that all presented results are averages over 10 runs.

4.1 Digit recognition benchmark

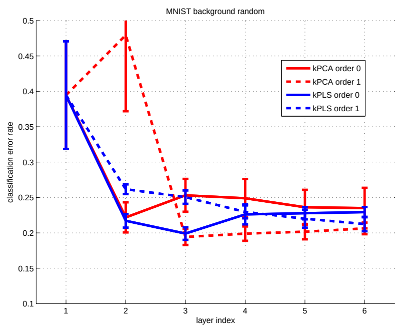
Datasets *mnist-back-image* and *mnist-back-random*[1] are composed of 28×28 grayscale handwritten digits which backgrounds were respectively filled with random image patches or random value pixels. These datasets have the particularity to show the largest error rate gap between shallow and deep architectures. Note that for scalability matters, we used subsets of those two MNIST datasets.

In MKM algorithm, the action performed in the first layer consists in a feature selection. The images being centered, any feature selection method would focus on the central pixels. Therefore, the number of features for the first layer was limited to 300. Even if the subsequent layers were also restricted to 300 features, we observed in practice that less features were selected. We compare the approach based on supervised selection of KPCA features [1] with our KPLS features selection using the same experimental protocol. The results were obtained using validation, training and test sets composed respectively of 500, 1000 and 2000 samples. Figure 1a depicts the error rates obtained for an increasing number of layers by 4 deep kernel approaches on *mnist-back-image* dataset. For this dataset, KPLS approaches clearly outperform the KPCA based approaches and showed a lower and stable variance. On the other MNIST dataset, results highlighted in figure 1b reveal that none of the methods clearly perform better than the others. Still, the variance of the KPLS approaches keeps on being low.

[1]Datasets and benchmark results available at `http://www.iro.umontreal.ca/~lisa/icml2007`



(a) *mnist-back-image*



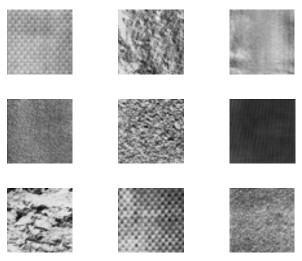
(b) *mnist-back-random*

Fig. 1: Mean classification error rate at each layer of the MKM .

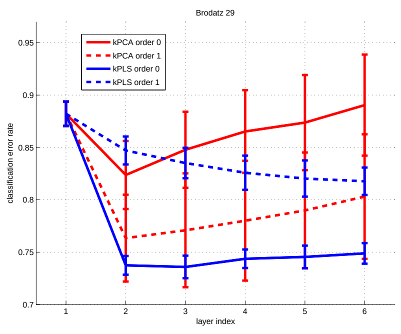
Notice that we have implemented the approach in [1] but we may have missed some heuristics to speed up the algorithm. Hence, due to computational issues, we shall only present results on small parts of the two MNIST datasets. As we were unable to run the KPCA based MKM on the whole MNIST datasets, we used subsets of those. This fact explains the difference in term of classification error rate between our results and those reported in [1]. However, we retrieve the global behavior of deep learning KPCA as exposed [1], even if our experiments are in small scale setting.

4.2 Application to texture classification

First of all, it should be stated that the MKM methods are not dedicated to texture recognition. They have been applied *out of the box* on the data in order to illustrate the gap of performance between the approaches. The Brodatz dataset consists of 112 textures of 640 × 640 pixels (as illustrated in figure 2a), among which we selected 29 images[2] to generate our dataset. As it contains 29 classes, a random classifier should achieve a mean error rate around 96.6%.



(a) Examples of Brodatz textures



(b) Experimental results

Fig. 2: Experimental results on Brodatz dataset.

[2]We used the textures from D1 to D30 (except D14 that was missing in our dataset).

Inspired by the protocol in [9], we split every texture into two non overlapping parts and randomly extracted 16×16 patches. Validation, training and test sets contained respectively 500, 1000 and 2000 samples.

On textures dataset, primary pixel selection could not be applied as for MNIST datasets where informations are spatially localized. Hence, the results for the first layer in figure 2b were obtained on raw textures. Here again, the size of layers $\ell > 1$ was limited to 300.

KPLS with order 0 kernel and KPCA with order 1 perform the best for this dataset with the trophy for KPLS. However, the impact of the kernel choice is completely different from the previous experiment and it may be explained by the numerical instability we observed as the number of layers increases. Here again, KPLS strategies have the lowest variance in error rate.

5 Conclusion

In this paper, we proposed a simplified and fast learning scheme for multilayer kernel machines based on KPLS. Experimental comparisons with the original KPCA approach on three datasets validate our method. Moreover the empirical computational cost for KPCA approach is higher (although not presented here). However, some behaviors are still unexplained for both schemes and further studies are still required to fully apprehend the impact of the chosen kernel on the model accuracy. Also, among forthcoming issues to be addressed are the application of other projection methods as for instance kernel CCA or combination of KPLS and KPCA and the large-scale extension of our approach.

References

[1] Y. Cho and L.K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350, 2009.

[2] B. Schölkopf and A.J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. the MIT Press, 2002.

[3] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160, 2007.

[5] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, 2008.

[6] Régis Lengellé and Thierry Denœux. Training mlps layer by layer using an objective function for internal representations. *Neural Networks*, 9(1):83 – 97, 1996.

[7] R. Rosipal, L.J. Trejo, and B. Matthews. Kernel PLS-SVC for linear and nonlinear classification. In *Proceedings of the 20th International conference on Machine learning*, pages 640–648, 2003.

[8] K. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18*, pages 1473–1480, 2006.

[9] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems 21*, pages 1033–1040, 2008.