Multilayer Radial Basis Function Kernel Machine

Mashail Alsalamah(✉) and Saad Amin

Faculty of Engineering and Computing, Coventry University, Priory Street, Coventry CV1 5FB, UK
alsalam2@coventry.ac.uk

Abstract. Radial Basis Function (RBF) Kernel Machines have become commonly used in Machine Learning tasks, but they contain certain flaws (e.g., some suffer from fast growth in the number of learning parameters while predicting data with large number of variations). Besides, Kernel Machines with single hidden layers lack mechanisms for feature selection in multidimensional data space, and machine learning tasks become intractable. This paper investigates "deep learning" architecture composed of multilayered adaptive non-linear components – Multilayer RBF Kernel Machine – to address RBF limitations. Three different approaches of features selection and dimensionality reduction to train RBF based on Multilayer Kernel Learning are explored, and comparisons made between them in terms of accuracy, performance and computational complexity. Results show that the multilayered system produces better results than single-layer architecture. In particular, developing decision support system in term of data mining.

Keywords: Machine learning · Classification · Unsupervised regression · Supervised regression · Principal component analysis

1 Introduction

In the recent decades, a number of techniques for different machine learning tasks, including classification, regression, function approximation clustering and feature transformation were developed with help of the class of non-linear functions – radial basis functions (rbf) [1, 2]. One of the interesting ideas is radial basis functions networks and their generalization kernel networks. In this work, special emphasis is given to the application of these networks to the problem of data classification.

Radial basis functions are a special kind of function which has a characteristic feature to monotonically decrease or increase with increase of the distance from the central point. The center, the distance scale and particular shape could vary for different models [1]. The most commonly used example is the Gaussian function:

$$f(x) = e^{-\frac{(x - c)^2}{r^2}}$$

and multi-quadratic function

P. Perego et al. (Eds.): MobiHealth 2016, LNICST 192, pp. 260–268, 2017.
DOI: 10.1007/978-3-319-58877-3_34

$$f(x) = \frac{\sqrt{r^{2+(x-c)^2}}}{r}$$

while, of course, a lot of variations possible. Another way to think about rbf networks is as kernel machines with specific type of kernel. Kernel machines are special machine learning methods which allow the use of regular machine learning techniques developed to learn linear functions in the problems with non-linear dependencies. This goal is achieved via transformation (mapping) of input feature space into the Hilbert space. The first kernel machines were a natural extension of the Support Vector Machine proposed by Vapnik for classification of the linearly separable data points. The goal of the algorithm was to find the hyperplane which will divide two datasets and will have maximum distance (margin) between itself and closest points from two classes. This hyperplane can be presented as the linear combination of the training samples lying on that margin (support vectors):

$$H(x) = \sum_i (\alpha_i y_i \langle x_i, x \rangle) + \alpha_0.$$

The algorithm finds the optimal values for the parameters α. The extension for the non-linear separable case exploits so called feature mapping function g with the hyperplane of the form:

$$H(x) = \sum_i (\alpha_i y_i \langle g(x_i), g(x) \rangle) + \alpha_0.$$

The function

$$K\left(x_i, x\right)$$

which satisfy the conditions of the Mercer's theorem can be presented in the form

$$K\left(x_i, x\right) = \langle g(x_i), g(x) \rangle$$

in the Hilbert space is called kernel. If the kernel function is selected appropriately the data points can become separable in the new feature space can become separable. This method is usually referred to in the literature as the "kernel trick". In this case the method for linear SVM training could be applied. The Gaussian radial basis function is one of this kernels, so the support vector learning could be applied as learning method for radial basis function network with support vectors being the centers of radial basis functions [7, 8].

The term of "deep" learning was coined in the contrast to the "shallow" learning algorithms which have fixed usually single layer architecture. The "deep" learning architectures are compositions of many layers of adaptive non-linear components [27].

Several attempts to combine the deep-learning approach with kernel based method were made in the past. For instance, method proposed in [29, 34] mimics the behavior of the multilayer neural network via single layer kernel network with specifically

generated kernel functions (arc-cosine kernels). Arc-cosine kernels are produced with recursive substitutions of the output of the kernel function. This action is equivalent to the non-linear transformation of the feature maps. In [35] the notion of the hierarchical feature invariance is proposed.

This paper provides three different approaches of features selection and dimensionality reduction to train RBF kernel based on the idea of MLK, and makes a comparison between them in terms of accuracy, the effect of changing the number of layers on the performance and computational complexity.

2 Deep RBF Kernel Machine

2.1 Multilayer RBF machine based on Kernel PCA

Below is a summary on how we implemented the Multilayer RBF machine based on Kernel PCA and shown in Fig. 2.

1. Let N be the number of layers we would like to use.
2. Prune the features by ranking method removing redundant features.
3. Select appropriate kernels and kernel parameters (cross-validation or otherwise) – not described in the Cho's work.
4. Apply kernel PCA algorithm and make the result be next layer set of features.
5. Determine number of features to extract (not described how), prune the redundant features from the resulting set – optional step.
6. If number of iterations exceeds N go to step 6, otherwise go to step 2.
7. Feed the feature representations to the classifier to make final decision.

2.2 Multilayer RBF machine based on supervised kernel regression

Below is a summary on how we implemented the Multilayer RBF machine based on supervised kernel regression and shown in Fig. 2.

1. Let N be the number of layers we would like to use.
2. Select appropriate kernels and kernel parameters (cross-validation or otherwise) – not described in the work.
3. Apply supervised regression to extract next feature value and corresponding eigen value.
4. If eigen value is greater than selected threshold go to step 3 otherwise use all the extracted features as input to the next layer.
5. If number of iterations exceeds N go to step 6, otherwise go to step 2.
6. Feed the feature representations to the classifier to make the final decision.

2.3 Multilayer RBF machine based on unsupervised kernel regression

Below is a summary of how we implement the Multilayer RBF machine based on unsupervised kernel regression and shown in Fig. 2.

1. Let N be the number of layers we would like to use.
2. Apply unsupervised regression to extract latent variables which better represent the input parameters. (Kernel parameters selection and dimensionality selection is embedded in this step) based on the ideas described in the Memisevic work: http://www.iro.umontreal.ca/~memisevr/pubs/ukr.pdf
   - Learning of optimal latent space representation with input data.
   - Learning of transformation from observable to latent space.
   - Selection of the kernel parameters and optimal dimensionality of the latent space.
3. Use extracted latent variables as input to the next step.
4. If number of iterations exceeds N go to step 6, otherwise go to step 2.
5. Feed the feature representations to the classifier to make the final decision (Fig. 1).

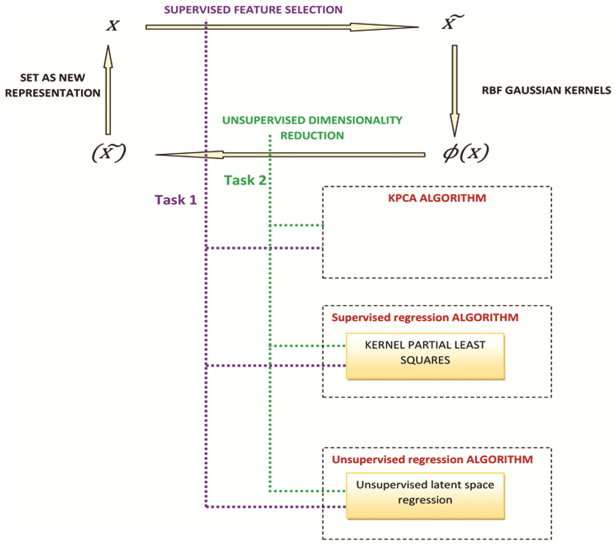

Fig. 1. Multilayer kernels machine (MKMs) for the three different transformation

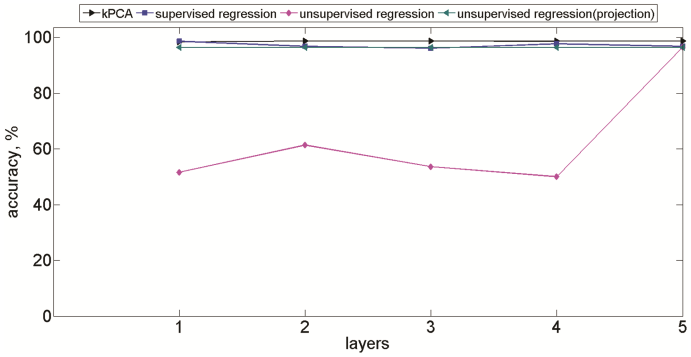

Fig. 2. Accuracy % with different methods

3 Results

In order to achieve the robustness of the results we used the 10-fold cross validation.

Accuracy is the ratio of correctly classified data points to the total number of data points. Mean squared error (MSE) in this case is the ratio of misclassified data to total number of data point. Training time is time in seconds which were solely used to train the model on specific amount of data with predefined number of layers.

Prediction time is the time in seconds spent solely on the prediction step with pre-trained model containing specific number of steps with fixed amount of training data.

The dataset is the MNIST digits dataset (http://yann.lecun.com/exdb/mnist/). The data there are presented as the grayscale images of the size 28 × 28 pixels. The values can vary from 0 to 255. Moreover, after trained and evaluated the three algorithms the unsupervised method gave not that good accuracy and in order to improve it the Unsupervised latent regression with projection method is suggested (Figs. 3 and 4).

(1) Dataset result.

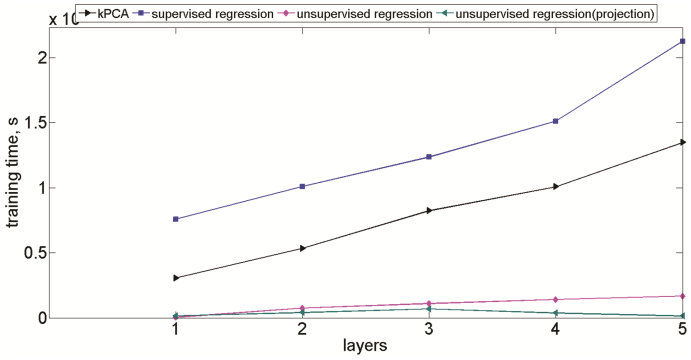

Fig. 3. Training time with different methods

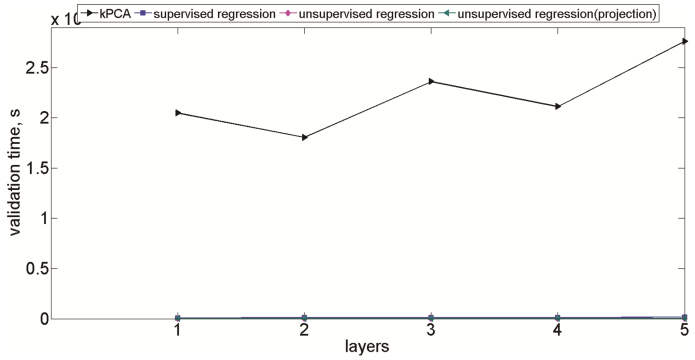

Fig. 4. Validation time with different methods

4 Discussion

The most stable results were shown by the supervised regression method for both balanced and unbalanced datasets. This is the only algorithm which shows continuous improvement in sensitivity and specificity with the increasing number of layers. Other algorithms show the tendency to stabilize results very quickly. That means that in practice it makes sense to test these algorithms only on small number of layers to understand whether they are sufficient for the current dataset or not. The worst time results were shown for the kPCA algorithms for both validation and training phases. It is caused by the large number of features selected at the projection step which are then transferred to the feature selection algorithm and knn-classifier. The time parameters can be improved by the limit on the number of features selected after projection. However, that might affect the accuracy in some cases.

5 Conclusion

In conclusion, the multilayered systems generally show relatively better results with large and highly varied data, as compared with "shallow learning" algorithms with usually single layer architecture. Moreover, amongst the multilayer algorithms, it show the ability of pattern and relationship discovery within large data sets. It holds promise in many area of the data application to quality assurance as found its way into data mining application and decision support system.

References

1. Orr, M.J.L.: Introduction to radial basis function network (1996). http://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf
2. Orr, M.J.L.: Recent advances in radial basis function networks (1999). http://www.anc.ed.ac.uk/rbf/papers/recad.ps

3. Kubat, M.: Decision trees can initialize radial basis function networks. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.6674&rep=rep1&type=pdf
4. Schwenker, F., Kestler, H.A., Palm, G.: Three learning phases for radial-basis-function networks (2001). http://sci2s.ugr.es/keel/pdf/specific/articulo/skg01.pdf
5. Kohonen, T.: Learning vector quantization. In: Arbib, M.A. (ed.) The Handbook of Brain Theory and Neural Networks, pp. 537–540. MIT Press, Cambridge (1995)
6. Schwenker F., Kestler H.A., Palm, G.: 3-D Visual Object Classification with Hierarchical Radial Basis Function Networks. http://www.uni-ulm.de/fileadmin/website_uni_ulm/iui/Ulmer_Informatik_Berichte/2001/UIB_2001-02.pdf
7. Cortes, C., Vapnik, V.: Support-vector networks (1995). http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf
8. Wettschereck, D., Dietterich, D.: Improving the performance of the radial basis function networks by learning center locations. http://papers.nips.cc/paper/544-improving-the-performance-of-radial-basis-function-networks-by-learning-center-locations.pdf
9. Chen, S., Grant, P.M., Cowan, C.F.N.: Orthogonal least-squares algorithm for training multi-output radial basis function networks (1992). https://cours.etsmtl.ca/sys828/REFS/B3/Chen_IEEE1992.pdf
10. Gomm, B.J., Yu, D.L.: Selecting radial basis function network centers with recursive orthogonal least-squares training (2000). http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=839002
11. De Castro, L.N., Von Zuben, F.J.: An immunological approach to initialize centers of radial basis function neural networks (2001). http://www.dca.fee.unicamp.br/~vonzuben/research/lnunes_dout/artigos/cbrn01.pdf
12. Yousef, R., el Hindi, K.: Training radial basis function networks using reduced sets as center points. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.4275&rep=rep1&type=pdf
13. Vachkov, G., Stoyanov, V., Christova, N.: Growing RBF network models for solving nonlinear approximation and classification problems (2015). http://www.scs-europe.net/dlib/2015/ecms2015acceptedpapers/0481-is_ECMS2015_0053.pdf
14. Billing, A.S., Zheng, G.L.: Radial basis function network configuration using genetic algorithms. http://www.sciencedirect.com/science/article/pii/089360809500029Y
15. Fasshauer, G.E., Zhang, J.G.: On choosing "optimal" shape parameters for RBF approximation. http://www.math.iit.edu/~fass/Dolomites.pdf
16. Hoffmann, G.A.: Adaptive transfer functions in radial basis function (RBF) networks. In: Bubak, M., Albada, G.D., Sloot, Peter M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3037, pp. 682–686. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24687-9_102. http://www.citemaster.net/get/a719201c-fb59-11e3-8a7f-00163e009cc7/hoffmann04adaptive.pdf
17. Duch, W., Jankowski, N.: Transfer functions: hidden probabilities for better neural networks. https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2001-400.pdf
18. Dorffner, G.: A unified framework for MLPs and RBFNs: introducing conic section functions networks. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.7264&rep=rep1&type=pdf
19. Mongillo, M.: Choosing basis functions and shape parameters for radial basis function methods. https://www.siam.org/students/siuro/vol4/S01084.pdf
20. Webb, R.A., Shennon, S.: Shape-adaptive radial basis functions. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=728359
21. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. http://www.cl.uni-heidelberg.de/courses/ws14/deepl/BengioETAL12.pdf

22. Wang, X.: The application of deep kernel machines to various types of data. https://uwaterloo.ca/computational-mathematics/sites/ca.computational-mathematics/files/uploads/files/shirly_project.pdf
23. Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. http://cseweb.ucsd.edu/~gary/cs200/s12/bengio-lecun-07.pdf
24. Bengio, Y.: Learning deep architecture for AI. http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf
25. Cho, Y.: Kernel methods for deep learning. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.387.4491&rep=rep1&type=pdf
26. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech and time-series. http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95a.pdf
27. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising auto-encoders: learning useful representations in a deep network using local denoising criteria. http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf
28. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep-belief nets. https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf
29. Cho, Y., Saul, K.: Large margin classification in infinite neural networks. http://cseweb.ucsd.edu/~yoc002/paper/neco_arccos.pdf
30. Bouvrie, J., Rosasco, L., Poggio, T.: On invariance in hierarchical models. http://papers.nips.cc/paper/3732-on-invariance-in-hierarchical-models.pdf
31. Bo, L., Ren, X., Fox, D.: Kernel descriptors for visual recognition. http://www.cs.washington.edu/robotics/postscripts/kdes-nips-10.pdf
32. Bo, L., Lai, K., Ren, X., Fox, D.: Object recognition with hierarchical kernel descriptors. http://www.cs.washington.edu/robotics/postscripts/hkdes-cvpr-11.pdf
33. Mairal, J., Koniusz, P., Harchaoui, Z., Schmid, C.: Convolutional kernel networks. http://arxiv.org/pdf/1406.3332v2.pdf
34. Zhuang, J., Tsang, I.W., Hoi, S.C.H.: Two-layer multiple kernel learning. http://jmlr.csail.mit.edu/proceedings/papers/v15/zhuang11a/zhuang11a.pdf, http://www.di.ens.fr/~fbach/skm_icml.pdf
35. Huang, P.-S., Avron, H., Sainath, T.N., Sindhvani, V., Ramabhadran, B.: Kernel methods match deep neural networks on TIMIT. http://www.ifp.illinois.edu/~huang146/papers/Kernel_DNN_ICASSP2014.pdf
36. Jose, C., Goyal, P., Aggrwal, P., Varma, M.: Local deep kernel learning for efficient non-linear SVM prediction. http://research.microsoft.com/en-us/um/people/manik/pubs%5Cjose13.pdf
37. Yger, F., Berar, M., Gasso, G., Rakotomamonjy, A.: A supervised strategy for deep kernel machine. https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2011-21.pdf
38. Schölkopf, B., Smola, A., Müller, K.-R.: Kernel principal component analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997). doi:10.1007/BFb0020217. http://www.ics.uci.edu/~welling/classnotes/papers_class/Kernel-PCA.pdf
39. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering analysis and an algorithm. http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf
40. Welling, M.: Kernel canonical correlation analysis. http://www.ics.uci.edu/~welling/classnotes/papers_class/kCCA.pdf
41. Wiering, M.A., Schutten, M., Millea, A., Meijster, A., Schomaker, L.R.B.: Deep support vector machines for regression problems. http://www.ai.rug.nl/~mwiering/GROUP/ARTICLES/DSVM_extended_abstract.pdf

42. Takeda, H., Farsiu, S., Milanfar, P.: Kernel regression for image processing and reconstruction. http://people.duke.edu/~sf59/KernelRegression_Final.pdf
43. Unsupervised kernel regression for non-linear dimensionality reduction. http://www.iro.umontreal.ca/~memisevr/pubs/ukr.pdf
44. Memisevic, R.: Unsupervised kernel dimension reduction. http://www.cs.berkeley.edu/~jordan/papers/wang-sha-jordan-nips11.pdf
45. http://deeplearning.net/datasets/
46. https://archive.ics.uci.edu/ml/datasets.html, https://physionet.org/physiobank/database/#multi