View kernel methods from the neural network perspective

Kaikai Zhao

Jan. 21, 2018

Outlines

- Introduction
- Priliminaries
- Neural network interpretation
- Adapting random kernel approximations
- Experiments
- Conclusions

Introduction

- **Deep neural networks** learn representations that are adapted to the data using **end-to-end training**.
- **Kernel methods** can only achieve this by selecting the **optimal kernels** to represent the data.
- A large number of random basis functions(features) are needed to obtain useful representations of the data.
- How much performance is lost due to the **kernel approximation error**(KAE) of the random basis?
- What is the possible gain of adapting the features to the **task** at hand?

Priliminaries

- Kernel methods

Kernels are **pairwise similarity** functions between two data points.

$$k(x, x') : R^d \times R^d \mapsto R$$

They are equivalent to the inner-products in an intermediate, potentially infinite-dimensional feature space produced by a function:

$$\phi : R^d \mapsto R^D, \; f(x) = W^T \hat{\phi}(x) + b$$

Non-linear kernel machines typically avoid using explicit functions by applying the kernel trick.

They work with kernel matrix which imposes a **quadratic** dependence on the number of samples *n* and prevents its applications in **large scale** settings.

- Approximate kernel methods

$$f(x) = W^T \hat{\phi}(x) + b$$

Priliminaries

- How to approximate kernel functions?

Gaussian Kernel

Given a smooth, shift-invariant kernel with Fourier transform *p(w)*, then:

$$k(z) = \int p(w) e^{jw^T z} dw \quad , \quad p(w) = N(0, \sigma^{-2})$$

$$k(z) = e^{-\frac{\|z\|_2^2}{2\sigma^2}}$$

$$\hat{\phi}(x) = \sqrt{\frac{1}{D}} [\sin(W_B x)^T \quad \cos(W_B x)^T]^T$$

$$W_B \in R^{D/2 \times d}$$

$$k(x, x') \approx \hat{\phi}(x)^T \hat{\phi}(x')$$

Reference

- Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. NIPS 2007

Priliminaries

ArcCos Kernel

$$k(x, x') = \frac{1}{\pi} \|x\| \|x'\| J(\theta)$$

$$J(\theta) = (\sin\theta + (\pi - \theta)\cos\theta)$$

$$\theta = \cos^{-1}\left(\frac{x \cdot x'}{\|x\| \|x'\|}\right)$$

$$\hat{\phi}(x) = \sqrt{\frac{1}{D}} \max(0, W_B x)$$

$$W_B \in R^{D \times d}$$

The feature map of the **ArcCos** kernel can be approximated by a one-layer neural network with the **ReLU** activation and a random weight matrix. Likewise the feature maps of the **ArcCos2** and **ArcCos3** kernels are then given by a **2- or 3-layer** neural network with the ReLU-activations.

Reference

- Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. NIPS 2009

Neural network interpretation

The approximated kernel features can be interpreted as the output of the hidden layer in a shallow neural network.

$$f(x) = W^T h(W_B^x) + b$$

$$W_B \in R^{D \times c}$$

$$z = W_B x$$

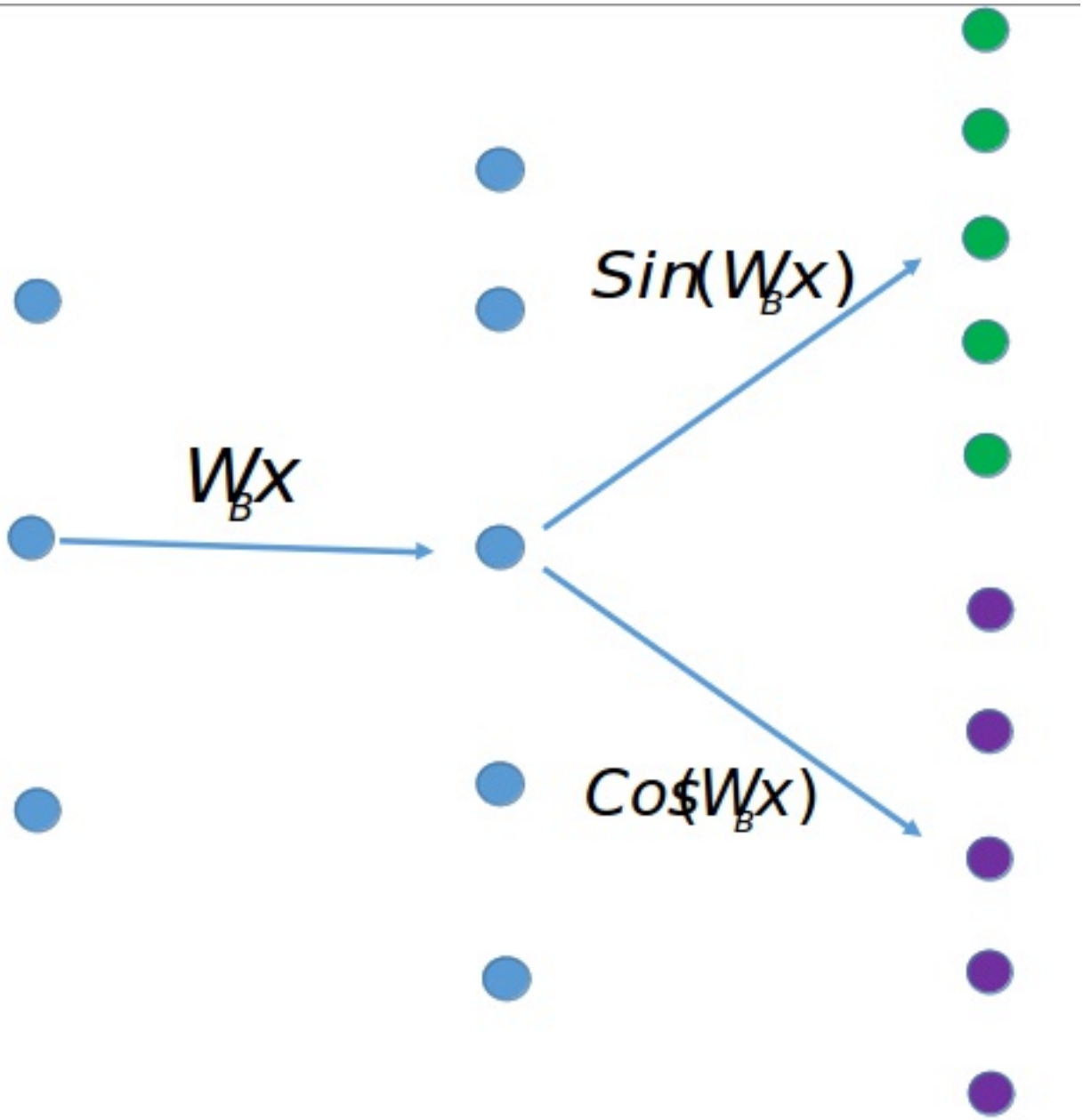
For Gaussian Kernels,

$$h(z) = \sqrt{\frac{1}{D}} [\sin(z)^T \quad \cos(z)^T]^T$$

For ArcCos Kernels,

$$h(z) = \sqrt{\frac{1}{D}} \max(0, z)$$

Neural network interpretation



Adapting random kernel approximations

The key difference between both methods is which parameters are trained.

- For the neural network, one optimizes the parameters in the bottom-layer and those in the upper layers jointly.
- For kernel machines, however, WB is fixed, i.e., the features are not adapted to the data.
- Hyper-parameters (such as σ defining the bandwidth of the Gaussian kernel) are selected with cross-validation or heuristics.

Reference

- Maximilian Alber, Pieter-Jan Kindermans, Kristof T. Schütt. An Empirical Study on The Properties of Random Bases for Kernel Methods. NIPS 2009

Adapting random kernel approximations

Random Basis - RB

For data-agnostic kernel approximation, we use the current state-of-the-art of random features. *Orthogonal* random features improve the **convergence** properties of the Gaussian kernel approximation over random Fourier features.

$$W_B \mapsto 1/\sigma G_B$$

sample GB from N(0,1) and orthogonalize the matrix

Unsupervised Adapted Basis - UAB

optimizing the sampled parameters WB w.r.t. the kernel approximation error (KAE):

$$\hat{L}(x, x') = \frac{1}{2}(k(x, x') - \hat{\phi}(x)^T \hat{\phi}(x'))^2$$

Reference

- X Yu Felix, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar.Orthogonal random features. NIPS 2016

Adapting random kernel approximations

Supervised Adapted Basis - SAB

Use kernel target alignment to **inject label information**. This is achieved by a target kernel function kY with kY (x; x') = +1 if x and x' belong to the same class and kY (x; x') = 0 otherwise. We maximize the alignment between the approximated kernel k and the target kernel kY for a given data set X:

$$\hat{A}(X, k, k_Y) = \frac{\langle K, K_Y \rangle}{\sqrt{\langle K, K \rangle \langle K_Y, K_Y \rangle}}$$

with $\langle K_a, K_b \rangle = \sum_{i,j}^{n} k_a(x_i, x_j) k_b(x_i, x_j)$.

Discriminatively Adapted Basis - DAB

A discriminatively adapted basis is trained jointly with the classifier to minimize the classification objective, i.e., WB, W, b are optimized at the same time.

Experiments

	Gaussian				*ArcCos*			
Dataset	**RB**	**UAB**	**SAB**	**DAB**	**RB**	**UAB**	**SAB**	**DAB**
Gisette	98.1	97.9	98.1	97.9	97.7	97.8	97.8	97.8
MNIST	98.2	98.2	98.3	98.3	97.2	97.4	97.7	97.9
CoverType	91.9	91.9	90.4	95.2	83.6	83.1	88.7	92.9
CIFAR10	76.4	76.8	79.0	77.3	74.9	76.3	79.4	75.3

Table 1: **Best accuracy** in % for different bases.

Experiments

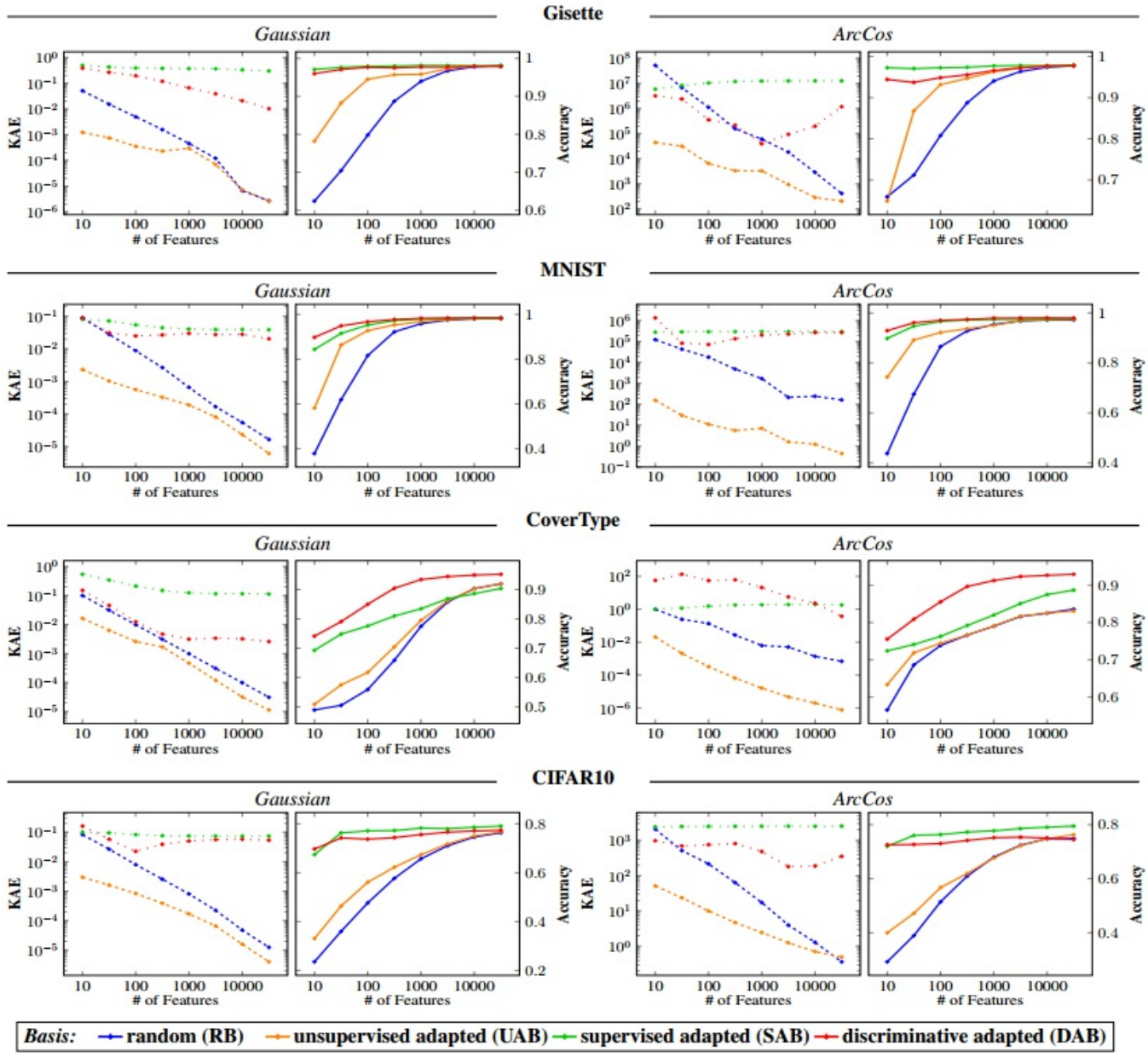


Figure 1: **Adapting bases.** The plots show the relationship between the number of features (X-Axis), the KAE in *logarithmic* spacing(**left, dashed lines**) and the classification error (**right, solid lines**). Typically, the KAE decreases with a higher number of features, while the accuracy increases. The KAE for SAB and DAB (orange and red dotted line) hints how much the adaptation deviates from its initialization (blue dashed line). Best viewed in digital and color.

A **typo**: **'orange'** is supposed to be **'green'** in the original paper.

Experiments

Transfer learning

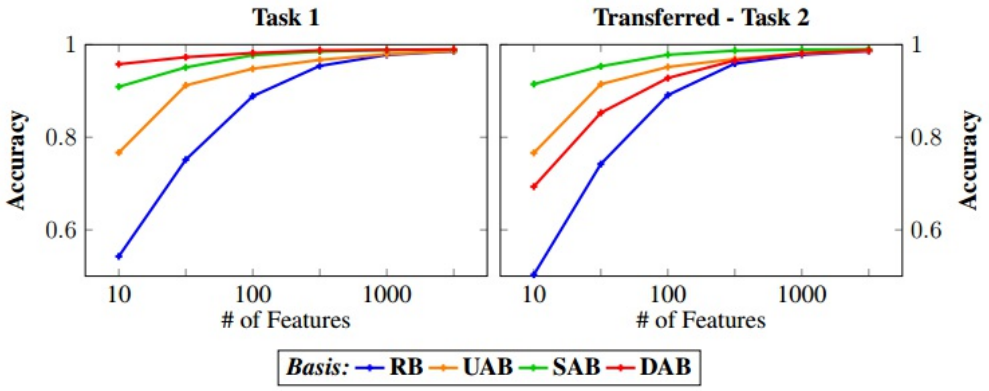


Figure 2: **Transfer learning.** We train to discriminate a random subset of 5 classes on the MNIST data set (left) and then transfer the basis function to a new task (right), i.e., train with the fixed basis from task 1 to classify between the remaining classes.

The performance of SAB for transfer learning is the best.

Experiments

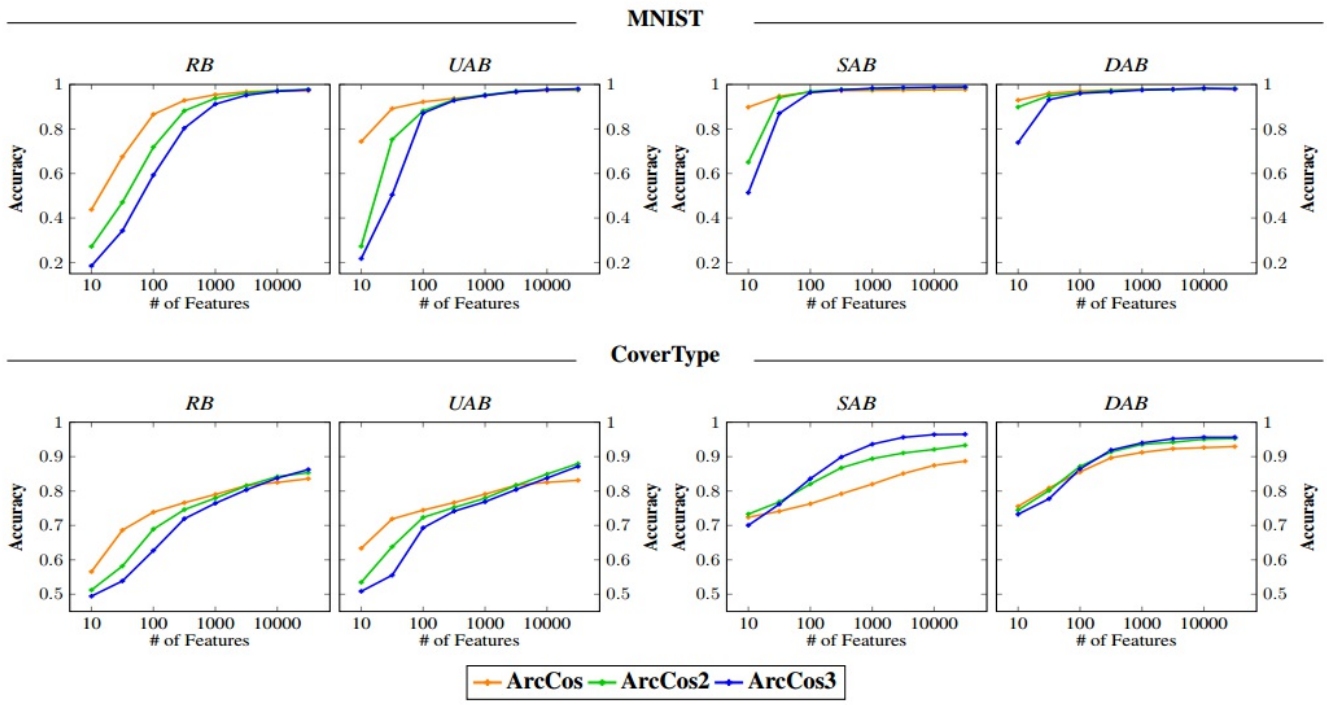


Figure 3: **Deep kernel machines.** The plots show the classification performance of the ArcCos-kernels with respect to the kernel (**first part**) and with respect to the number of layers (**second part**). Best viewed in digital and color.

For a limited number of features, i.e., less than 3000, the deeper kernels perform worse than the shallow ones. Only given enough capacity the deep kernels are able to perform as good as or better than the single-layer bases.

The deeper the net is, the more random features needed to train the parameters for the net.

Conclusions and Future work

- The main contribution of this paper is conducting an empirical study to understand the difference between approximated kernel methods and neural networks.
- Proposing an adaptive method to select the number of random features makes approximate kernel learning more economical.
- Scaling the existing single-kernel methods to multiple-kernel learning and applying it to large-scale machine learning applications may be a good idea.

Thanks for your time!