Paper Review on Kernel Analysis of Deep Networks

Yang Fu
Department of Statistics
University of Washington
Seattle, WA 98195
fuyang@uw.edu

1 Introduction

This is a paper review on *Kernel Analysis of Deep Networks*. In the original paper, depth and structure of deep neural networks are analysed via kernel principal component analysis. It verifies two hypotheses.

One is that as the network gets deeper, simpler and more accurate representations are obtained. Here simpleness is measured by the number of principal components used and accuracy is measurd by the error rate of the softmax classifier built on those principal components.

The other is that the structure of the network controls how fast the representation of the task is formed layer after layer. It means that the error rate for networks with different structure will decrease in different paths as the number of principal components grows although the networks will have the same error rate as we control them to.

In this review, section 2 will prove that the kernel used in the paper is positive semi-definite. Section 3 will introduce the kernel-based method, explain how it is solved with a numerical algorithm and check the convergence of the algorithm. The last section repeats some of the experiments on the artificial data, MLP and CNN.

2 Kernel

In the paper, the Gaussian RBF kernel is used. In this section, it will be shown that RBF kernel is a positive semi-definite kernel.

2.1 Definition

A Gaussian RBF kernel with hyper-parameter $\sigma > 0$ on two sample $x, x' \in \mathbb{R}^d$ is defined as

$$k(x, x') = \exp(-\frac{\|x - x'\|^2}{\sigma^2}).$$

2.2 Positive semi-definite

Lemma 1. *If $k_i, i = 1, 2, \cdots$ are positive semi-definite kernels, then $\lim_{i\to\infty} k_i$ is also a positive semi-definite kernel if the limit exists.*

Proof By the definition of positive semi-definite kernel, for all i,

$$g_i = \sum_{k,\ell} \alpha_k \alpha_\ell k_i(x_k, x_\ell) \geq 0,$$
$$\therefore \lim_{i\to\infty} g_i \geq 0.$$

■

Lemma 2. *If h is a positive semi-definite kernel, and $f : \mathbb{R}^d \to \mathbb{R}$ is a function. Then*

$$k(x, x') = f(x)h(x, x')f(x')$$

is also a positive semi-definite kernel.

Proof $\forall \alpha_1, \cdots, \alpha_n \in \mathbb{R}$,

$$\sum_{i,j} \alpha_i \alpha_j f(x_i) h(x_i, x_j) f(x_j) = \sum_{i,j} \beta_i \beta_j h(x_i, x_j) \geq 0,$$

where $\beta_i = \alpha_i f(x_i) \in \mathbb{R}$. ■

Theorem 3. *Gaussian RBF kernel $k(\boldsymbol{.}, \boldsymbol{.})$ is a positive semi-definite kernel.*

Proof Without loss of generality, assume $\sigma = \sqrt{2}$. $\forall x_1, \cdots, x_n \in \mathbb{R}^d$,

$$\begin{aligned}
&\because \|x_i - x_j\|^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j, \\
&\therefore k(x_i, x_j) = \exp(-\frac{x_i^T x_i}{2}) \exp(x_i^T x_j) \exp(-\frac{x_j^T x_j}{2}).
\end{aligned}$$

By Taylor expansion,

$$\exp(x_i^T x_j) = \sum_{k=0}^{\infty} \frac{(x_i^T x_j)^k}{k!}.$$

Let $h_\ell(x, y) = \sum_{k=0}^{\ell} \frac{(x^T y)^k}{k!}$. Since for each $\ell, h_\ell(x, y)$ is a finite sum of polynomial kernels, $h_\ell(x, y)$ is also positive semi-definite. By Lemma 1, $\exp(x_i^T x_j)$ is a positive semi-definite kernel and thus, by Lemma 2, RBF kernel is also positive semi-definite. ■

3 Kernel-based method

In the paper, Kernel PCA is used to analyse MLP and CNN. In this section, Kernel PCA will be introduced as well as the algorithm used to solve it.

3.1 Kernel PCA

As a kernel counterpart of regular PCA, Kernel PCA aims at solving the optimization problem

$$\arg\max_{f \in \mathcal{H}} \frac{\langle f, \hat{S} f \rangle_{\mathcal{H}}}{\|f\|_{\mathcal{H}}^2},$$

where $\mathcal{H}$ is RKHS, $\hat{S}$ is the covariance operator defined as

$$\begin{aligned}
\hat{S} &= \frac{1}{n} \sum_{i=i}^{n} [k(x_i, \boldsymbol{.}) - \hat{\mu}] \otimes [k(x_i, \boldsymbol{.}) - \hat{\mu}], \\
\langle f, \hat{S} g \rangle_{\mathcal{H}} &= \frac{1}{n} \sum_{i=1}^{n} \langle f, k(x_i, \boldsymbol{.}) - \hat{\mu} \rangle_{\mathcal{H}} \langle g, k(x_i, \boldsymbol{.}) - \hat{\mu} \rangle_{\mathcal{H}},
\end{aligned}$$

where $\hat{\mu} = \frac{1}{n} \sum k(x_i, \boldsymbol{.})$ is the mean element.

After centering the kernel matrix, the optimization problem becomes

$$\arg\max_{f \in \mathcal{H}} \frac{\sum_i f^2(x_i)}{\|f\|_{\mathcal{H}}^2}.$$

Since the objective function takes $x_1, \cdots, x_n, \|f\|_{\mathcal{H}}$ as inputs and is strictly decreasing in the $(i+1)$-th variable $\|f\|_{\mathcal{H}}$, the Representer Theorem can be invoked and the solution has the form $f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$, where $\alpha_i \in \mathbb{R}$.

Thus, the optimization problem becomes

$$\arg\max_{\alpha \in \mathbb{R}^n} \frac{\alpha^T \tilde{K}^2 \alpha}{\alpha^T \tilde{K} \alpha},$$

where $\tilde{K}$ is the centered kernel matrix.

Now we show how the problem is equivalent to finding the eigenvector correspond to the largest eigenvalue. By the Spectral Theorem,

$$\tilde{K} = \sum_{p=1} \lambda_p e_p e_p^T,$$

where e_p are unit-length eigenvectors and λ_p are eigenvalues in decreasing order.

Let $\alpha = \sum_{j=1}^{n} \beta_j e_j$, since e_i are orthogonal and unit-length,

$$\begin{aligned}
&\frac{\alpha^T \tilde{K}^2 \alpha}{\alpha^T \tilde{K} \alpha} = \frac{\sum_j \beta_j^2 \lambda_j^2}{\sum_j \beta_j^2 \lambda_j} \equiv Obj, \\
&\frac{\partial Obj}{\partial \beta_k} = \frac{2\beta_k \lambda_k^2 (\sum_j \beta_j^2 \lambda_j) - 2\beta_k \lambda_k (\sum_j \beta_j^2 \lambda_j^2)}{(\sum_j \beta_j^2 \lambda_j)^2} = 0, \\
\Leftrightarrow &\lambda_k = \frac{\sum_j \beta_j^2 \lambda_j^2}{\sum_j \beta_j^2 \lambda_j}, k = 1, \cdots, n.
\end{aligned}$$

That is to say, the local maximums of the objective function are the eigenvalues. Thus, the universal maximum is the largest eigenvalue λ_1, and

$$\begin{aligned}
&\frac{\alpha^T \tilde{K}^2 \alpha}{\alpha^T \tilde{K} \alpha} = \lambda_1, \\
\Leftrightarrow &\alpha^T \tilde{K} (\tilde{K} \alpha - \lambda_1 \alpha) = 0.
\end{aligned}$$

Thus, α is the eigenvector correspond to λ_1. Normalize α so that $\alpha^T \tilde{K} \alpha = 1$ and we have $\alpha = \frac{1}{\sqrt{\lambda_1}} e_1$.

The Kernel PCA algorithm can be summerized as follow,

Algorithm 1 Kernel PCA

Input: Kernel Matrix K;
Output: Principal component α;
1: Center kernel matrix $K = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)K(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$;
2: Compute eigenvectors e_j, eigenvalues λ_j of K;
3: The j-th principal component is the normalized eigenvector $\alpha_{(j)} = \frac{1}{\sqrt{\lambda_j}} e_j$.

3.2 Power Method

The algorithm used to solve eigen decomposition is called the Power Method.

Algorithm 2 Power Method

Input: Matrix A;
Output: Eigenvector v, eigenvalues λ;

1: Randomly initialize $v^{(0)}$;
2: **repeat**
3: $\quad z^{(t)} = Av^{(t-1)}$;
4: $\quad v^{(t)} = \frac{z^{(t)}}{\|z^{(t)}\|_2}$;
5: $\quad \lambda^{(t)} = (v^{(t)})^T Av^{(t)}$;
6: **until** Max iteration
7: Project A orthogonal to previous v: $A = (I - vv^T)A(I - vv^T)$;
8: Redo previous steps to get other eigenvectors and values.

The convergence property of the Power Method can be examined. Let A be a diagonalizable matrix and $u_1, \cdots, u_n$ be the eigenvetors. Since the eigenvetors form a basis of $\mathbb{R}^n$, the initialized vetor v_0 in the power method can be written as

$$v^{(0)} = \sum_{i=1}^{n} a_i u_i,$$

where a_i are scalars. Multiply both sides by A^t, we have

$$\begin{aligned} A^t v^{(0)} &= \sum_{i=1}^{n} a_i A^t u_i \\ &= \sum_{i=1}^{n} a_i \lambda_i^t u_i \\ &= a_1 \lambda_1^t \left(u_1 + \sum_{i=2}^{n} \frac{a_i}{a_1} \left(\frac{\lambda_i}{\lambda_1} \right)^t u_i \right), \end{aligned}$$

where λ_1 is the dominant eigenvalue. Thus, $\left(\frac{\lambda_i}{\lambda_1}\right)^t \to 0$ and $A^t v^{(0)} \to a_1 \lambda_1^t u_1$.

In order to scale things up, Simultaneous Power Method is used, which is the same as (non-simultaneous) Power Method in nature. Except that, instead of a vector, it multiplies A by multiple vectors put in a matrix Q, where $Q^T Q = I$. And each step the vectors are normalized with QR decomposition. The convergence propoty is the same as the non-simultaneous version.

4 Experiments

In the experiments on MNIST-10K, the deep networks are trained on the 10000 samples until a training error rate of 2.5% is reached, same as the orginal paper. Then, kernel principal components of different layers of the networks are esitmated and fed into a softmax classifier with no penalty to get the error rates. For each layer (0,1,2), each network (MLP, CNN) and each number of principal components (1-10), a grid of hyper-parameters of the RBF kernel is tested on and the hyper-parameter that gives the lowest error rate of the softmax classifier is chosen.

The parameters of the deep networks are exactly the same as the original paper except that ReLU is used instead of sigmoid to get a shorter training time.

4.1 Artificial data

The experiment on the artificial data coincides with the original paper's conclusion. Fig 1 shows that it does take more principal components for the more distorted data to be correctly classified.

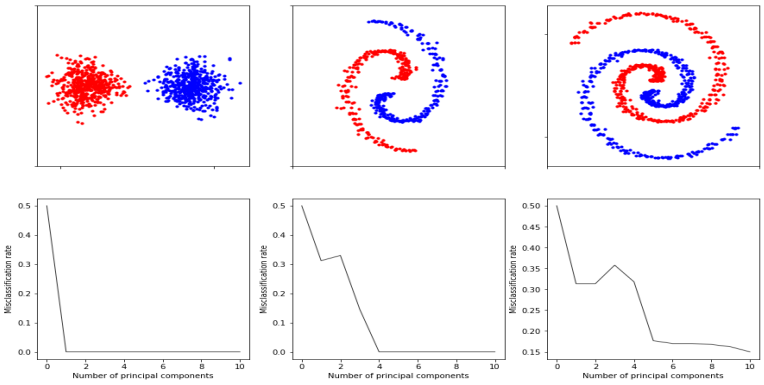

Figure 1: Results on artificial data

4.2 MNIST-10K

The experiment in the original paper to comfirm the first hypothesis is repeated on MNIST-10K. Fig 2 shows that the result is very similar to the one in the original paper and the hypothesis is comfirmed. Better representations are built layer after layer.

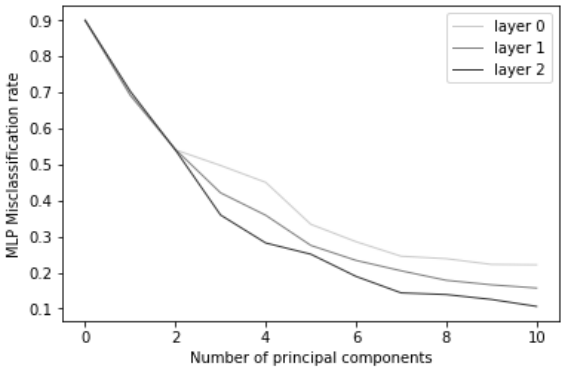

Figure 2: Layer-wise evolution of the error (MLP)

The experiment to comfirm the second hypothesis is also repeated except for the pretrained multilayer perceptron (PMLP) and transfer learning part. The result is less clear compared to the original paper. In the original paper, CNN and MLP have very different paths for layer-wise evolution of the representation. Fig 3 shows that the path is not that different.

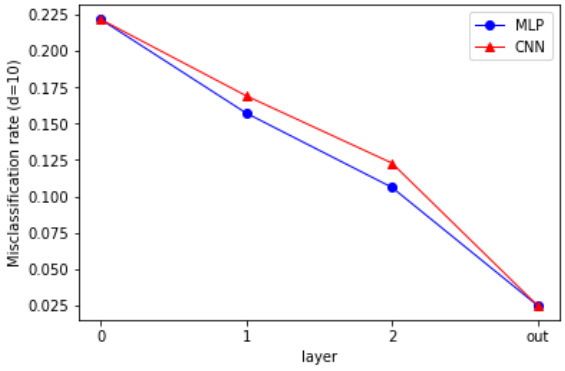

Figure 3: Layer-wise evolution of the error obtained for each training procedure for $d = 10$

5 Conclusion

The original paper proposes a new way to analyse deep networks via kernel PCA and certain experiments is carried out to test two hypothses.

The first hypothesis that as the network gets deeper, simpler and more accurate representations are obtained is comfired while the second hypothesis about the structure of the networks is less comfirmed to me in this paper review. Experiments on more different datasets and networks are needed in order to make that conclusion.

References

[1] Montavon, G., Braun, M.L. & Müller, K.R. (2011) Kernel analysis of deep networks. *Journal of Machine Learning Research* **12**(3):2563-2581.

[2] Convolutional Neural Networks (2018, January 27). Retrieved from `https://www.tensorflow.org/tutorials/deep_cnn`

[3] Power Iteration (2017, June 26). Retrieved from `http://mlwiki.org/index.php/Power_Iteration`