Multilayer Kernel Machines

Speaker: Siyu Gao

New York University Shanghai

MATH-SHU 236 Math Foundation for DS Final Presentation

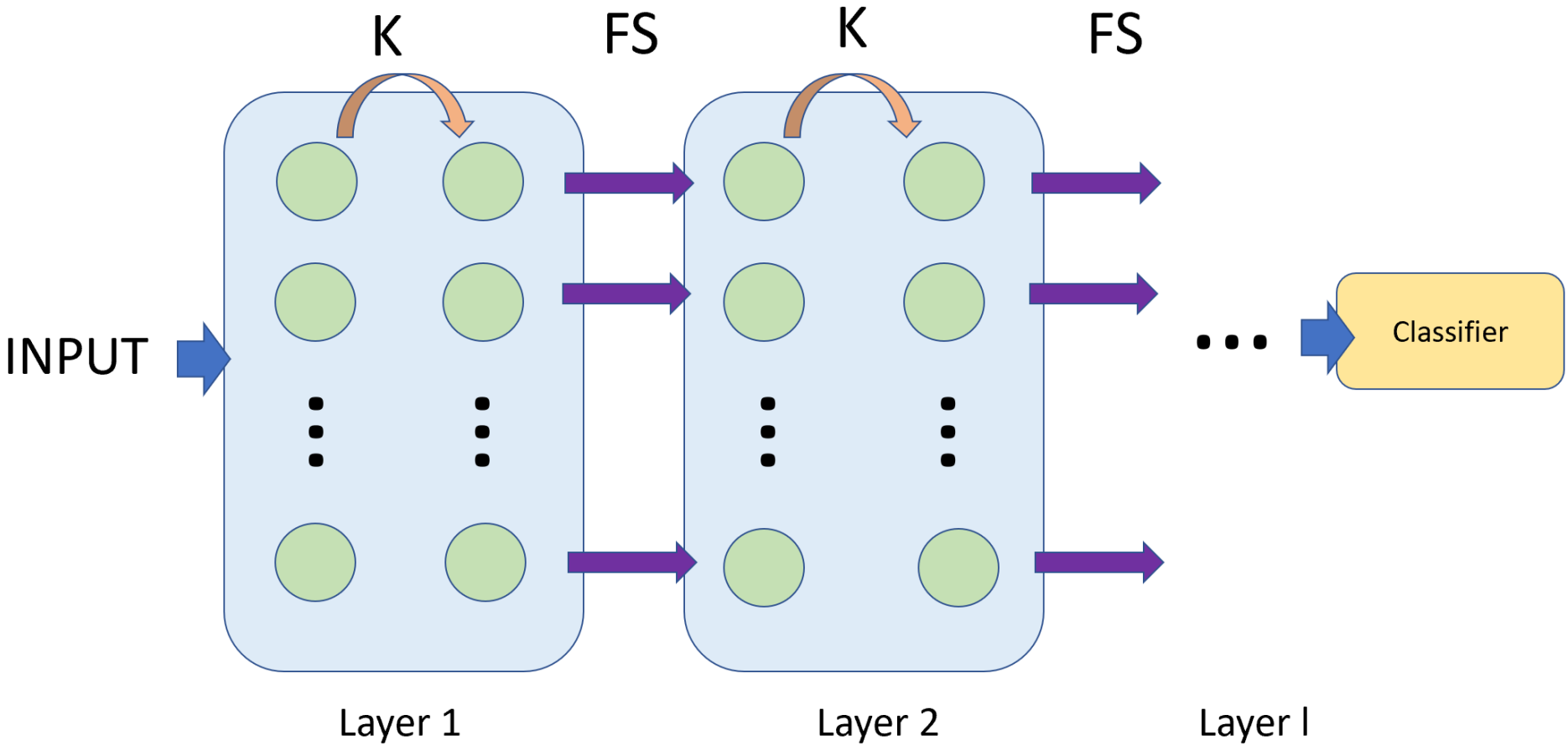
Outline

- Background
  - Project Motivation
  - Multilayer Kernel Machines (MKM)
  - Arc-cosine Kernels (ACK) Recap
- Empirical Tests
  - PCA + KNN
  - KPCA + KNN
  - KPCA + FS + KNN
- Time Complexity Analysis
- Summary and Questions
- Other Related Work and Future Discussions
- Q&A

Background-Motivation

1. Kernel Machines build a bridge between linearity and nonlinearity for many machine learning algorithms and essentially improve the performance
2. Deep kernel machines share the benefit of deep architecture.
3. MKM shows competitive performance in shape and digit recognition based on Cho's work.

Multilayer Kernel Machines (MKM)



Background - Arc-cosine Kernel

Definition: For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the **arc-cosine** function via the integral representation is :

$$k_n(\mathbf{x}, \mathbf{y}) = 2 \int \mathbf{dw} \frac{e^{-\frac{\|\mathbf{w}\|^2}{2}}}{(2\pi)^{d/2}} \Theta(\mathbf{w} \cdot \mathbf{x}) \Theta(\mathbf{w} \cdot \mathbf{y}) (\mathbf{w} \cdot \mathbf{x})^n (\mathbf{w} \cdot \mathbf{y})^n$$

To better understand the influence of inputs, it can be rewrite as

$$k_n(\mathbf{x}, \mathbf{y}) = \frac{1}{\pi} \|\mathbf{x}\|^n \|\mathbf{y}\|^n J_n(\theta)$$

where

$$J_n(\theta) = (-1)^n (\sin\theta)^{2n+1} \left(\frac{1}{\sin\theta} \frac{\partial}{\partial\theta} \right)^n \left(\frac{\pi - \theta}{\sin\theta} \right).$$

Background - Arc-cosine Kernel Properties

Observing that:

$$\begin{aligned} J_0(\theta) &= \pi - \theta \\ J_1(\theta) &= \sin\theta + (\pi - \theta)\cos\theta \\ J_2(\theta) &= 3\sin\theta\cos\theta + (\pi - \theta)(1 + 2\cos^2\theta) \end{aligned}$$

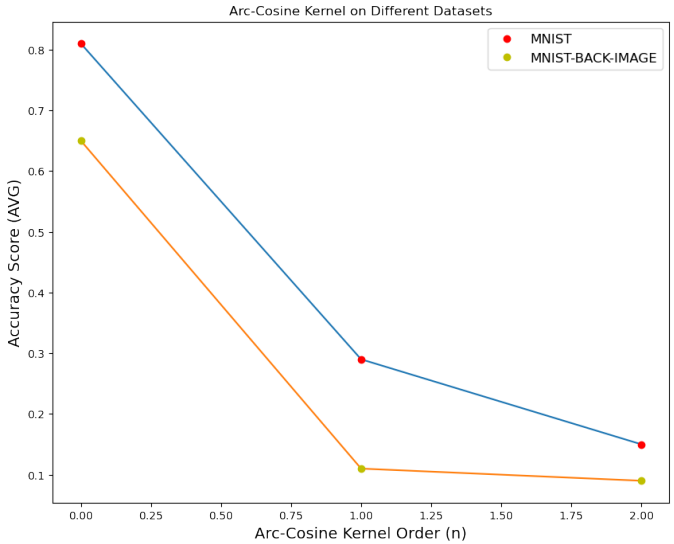
1. when $n = 0, k_0(\mathbf{x}, \mathbf{x}) = 1$
2. when $n = 1, k_1(\mathbf{x}, \mathbf{x}) = \|\mathbf{x}\|^2$
3. when $n > 1$, $k_n(\mathbf{x}, \mathbf{x}) \sim \|\mathbf{x}\|^{2n}$

Compared with other kernels:

- RBF, linear and polynomial kernels all share these three properties
- Without continuous tuning parameters.

ACK Combined with Shallow Model

Datasets:MNIST, MNIST-BACK-IMAGES
Kernel Setting: arc-cosine kernel with order = 0,1,2
SVM Setting: kernel = "precomputed"
Results:



Observations:

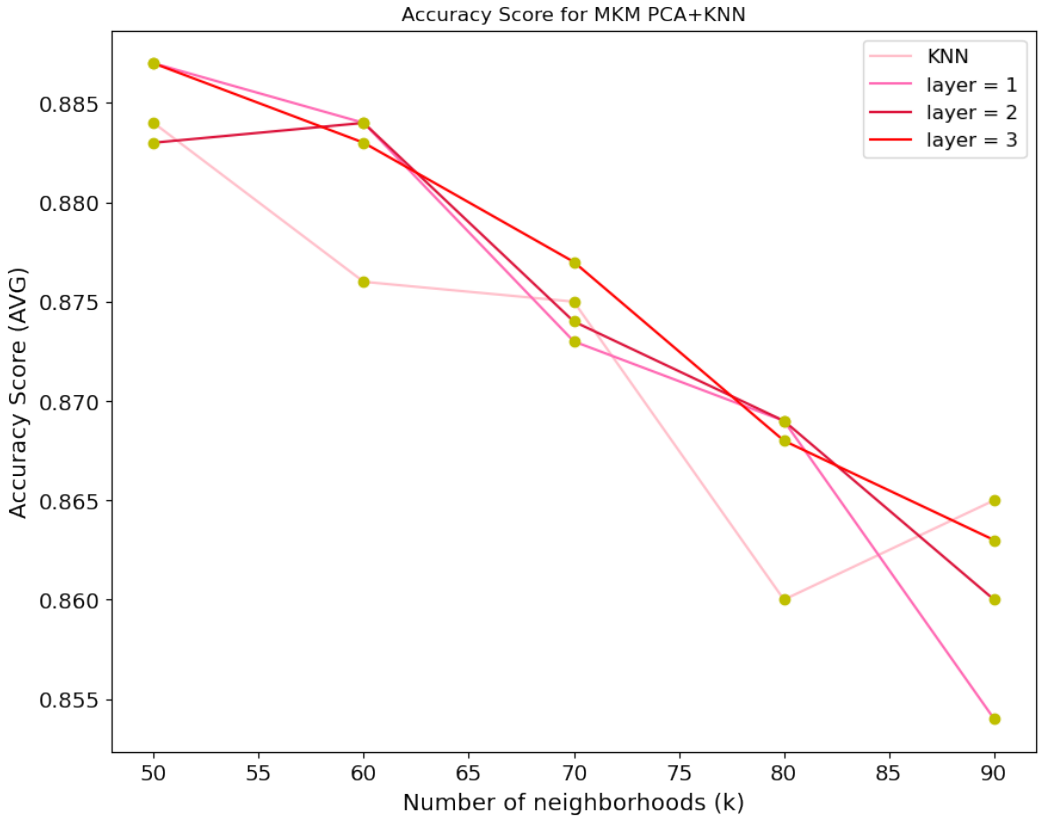
- Take very long time to fit dataset MNIST-BACK-RAND.
- Generally, lower order obtain better performance.

Result Consistency

- Though tested on the different datasets, we get the similar result compared with other's work.(lower layer obtain better performance)
- It outperforms the SVM-RBF on lower order.
- However, there are also inconsistency. In datasets to demonstrate benefit of deep architecture, high order performs better.

MKM: Start From PCA + classifier

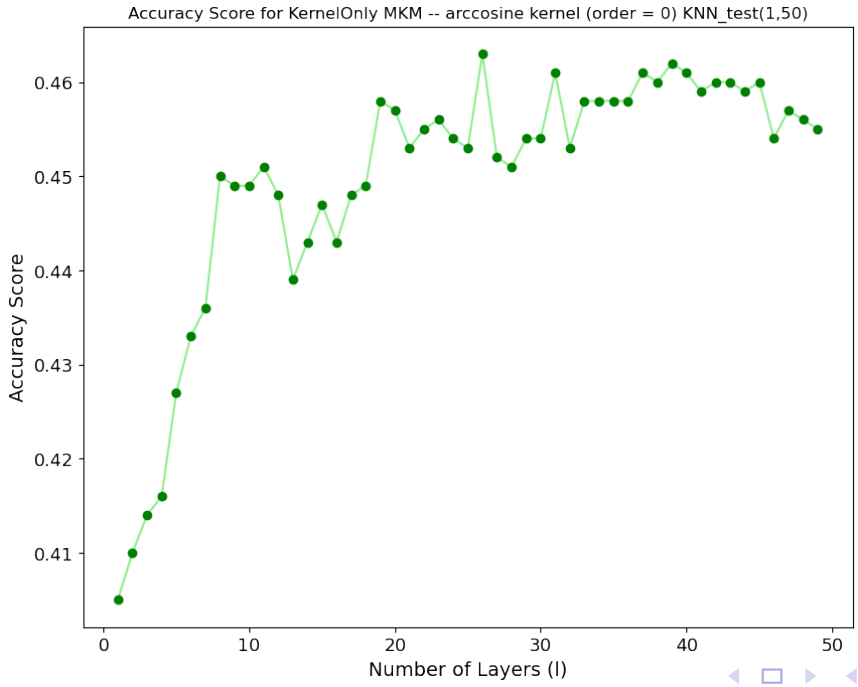
Datasets:MNIST
Kernel Setting: None
classifier Setting: KNN(k)
Results:



The Choice of KNN(k)

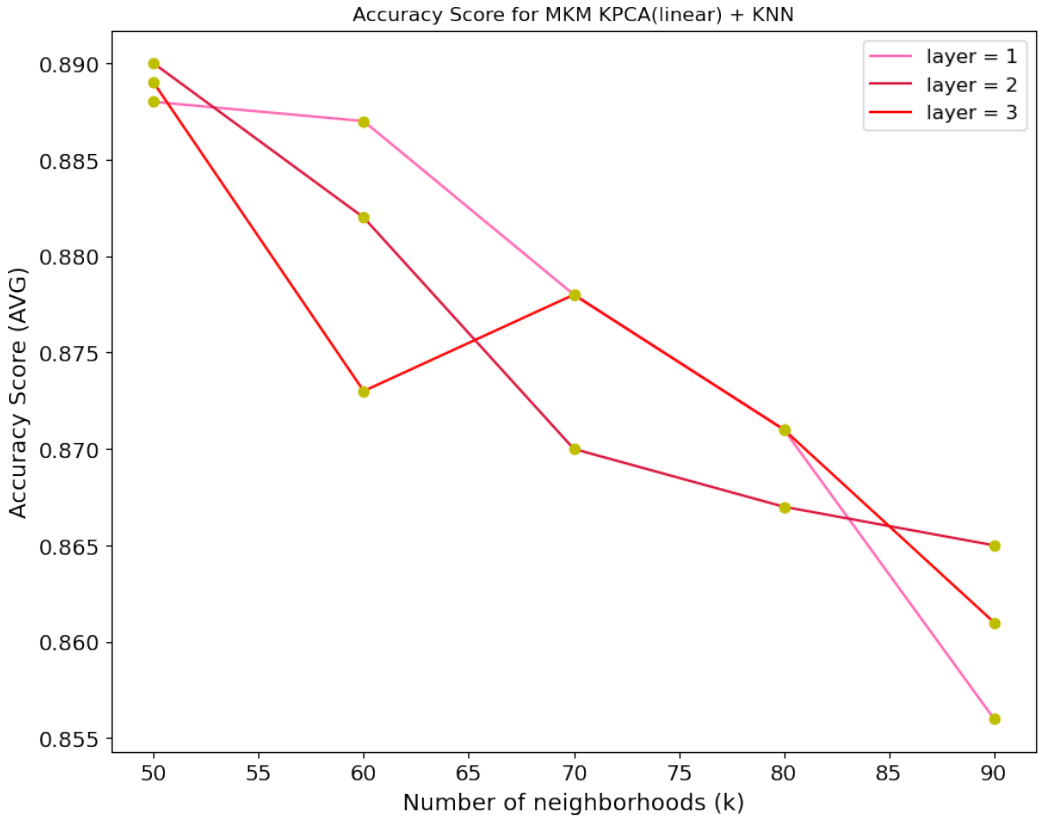
Datasets:MNIST
Kernel Setting: None
classifier Setting: KNN(k)
How to choose k?

- Cross validation to tune a parameter
- Here since data need to be transfered from all layers of kernels, simpler work are applied.



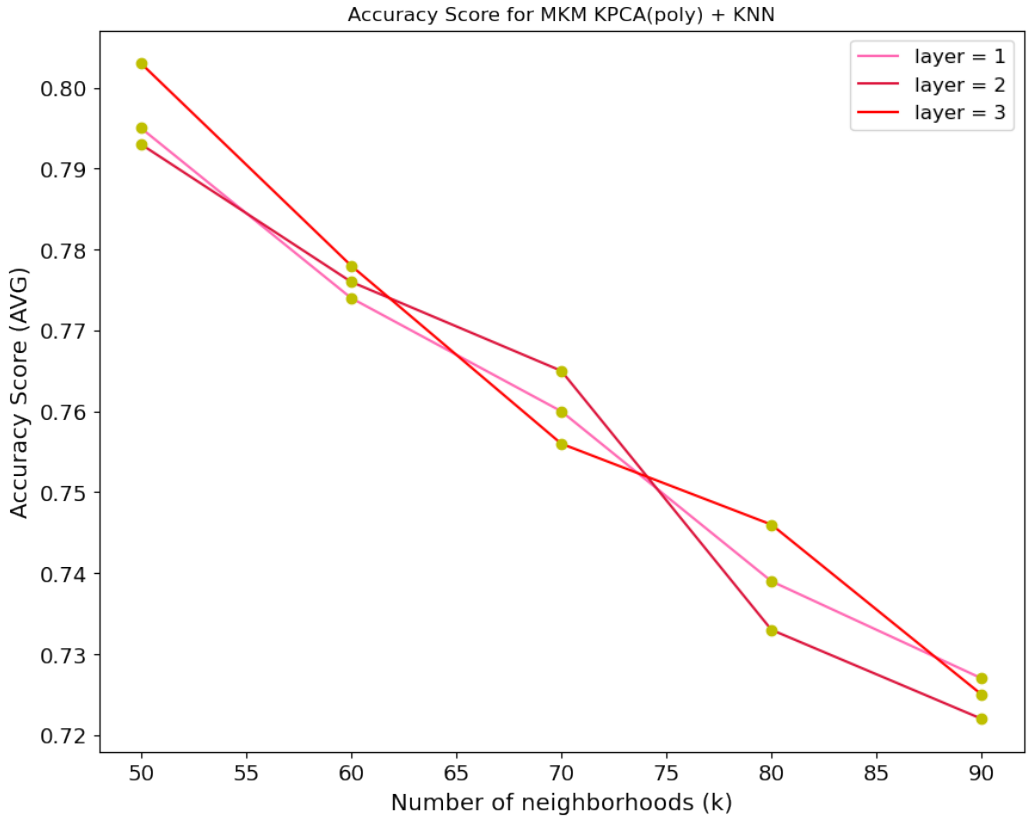
MKM: KPCA(linear) + KNN

Datasets: MNIST
Kernel Setting: linear (No hyper-parameter)
classifier Setting: KNN(k)
Results:



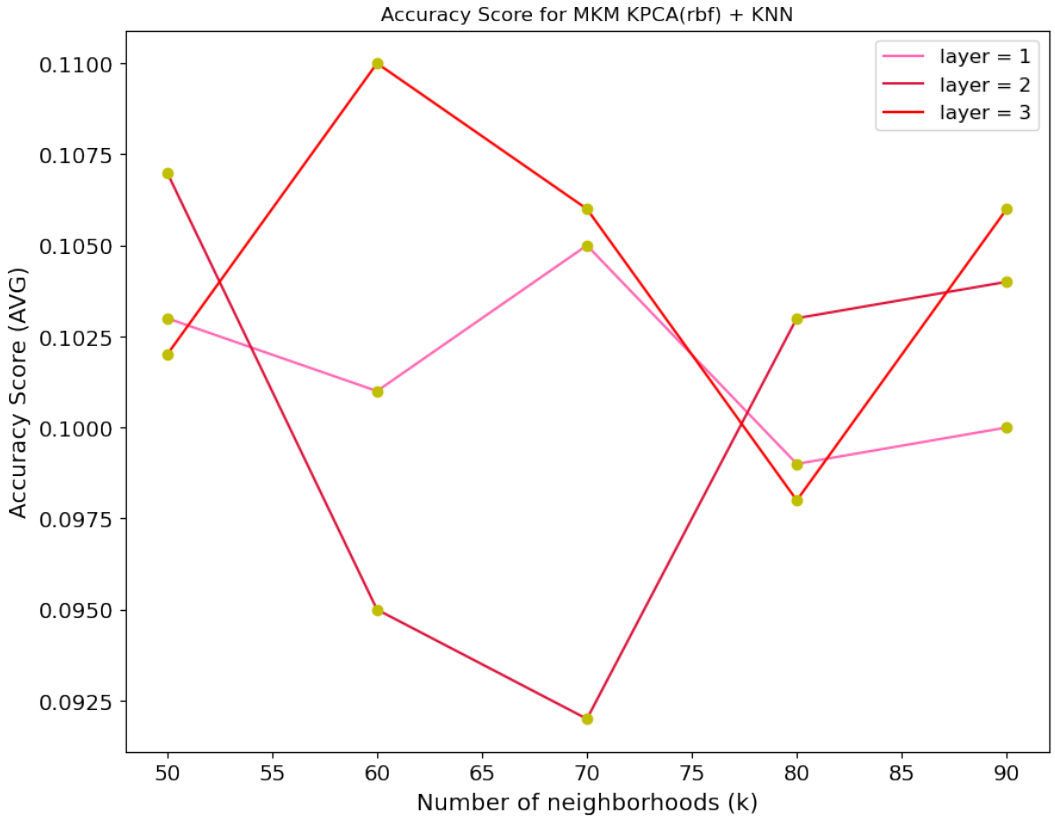
MKM: KPCA(poly) + KNN

Datasets: MNIST
Kernel Setting: polynomial (hyper-parameter set by default)
classifier Setting: KNN(k)
Results:



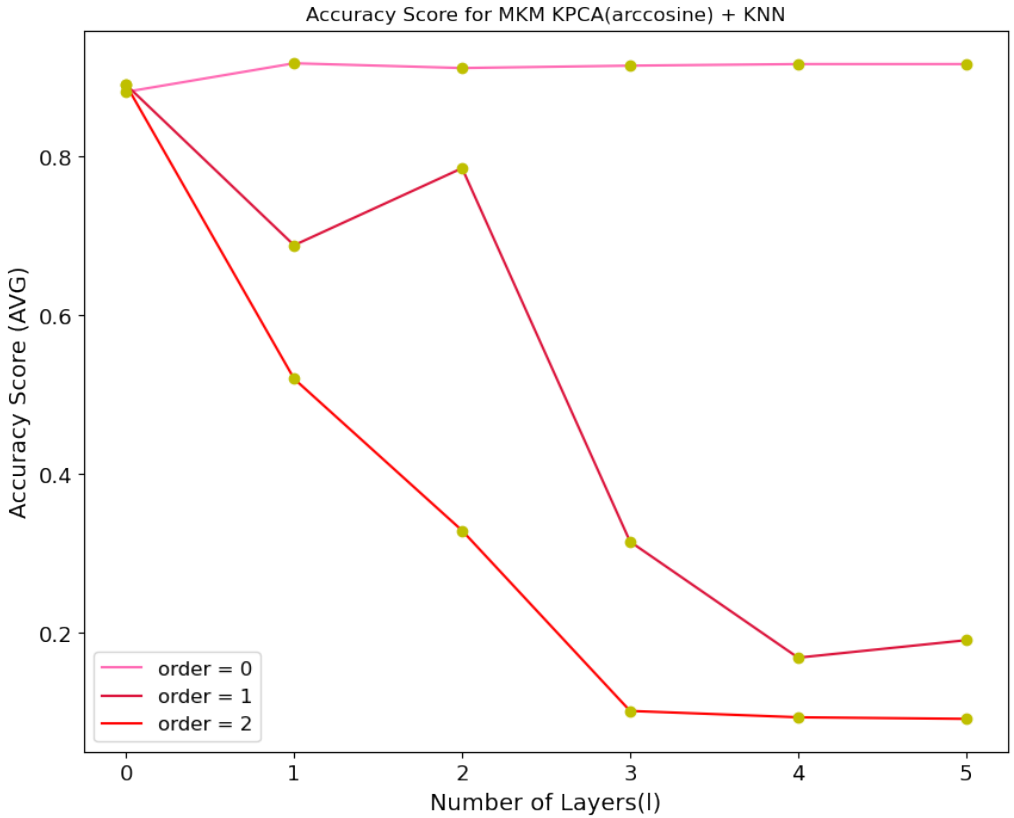
MKM: KPCA(rbf) + KNN

Datasets: MNIST
Kernel Setting: rbf (gamma set by default)
classifier Setting: KNN(k)
Results:



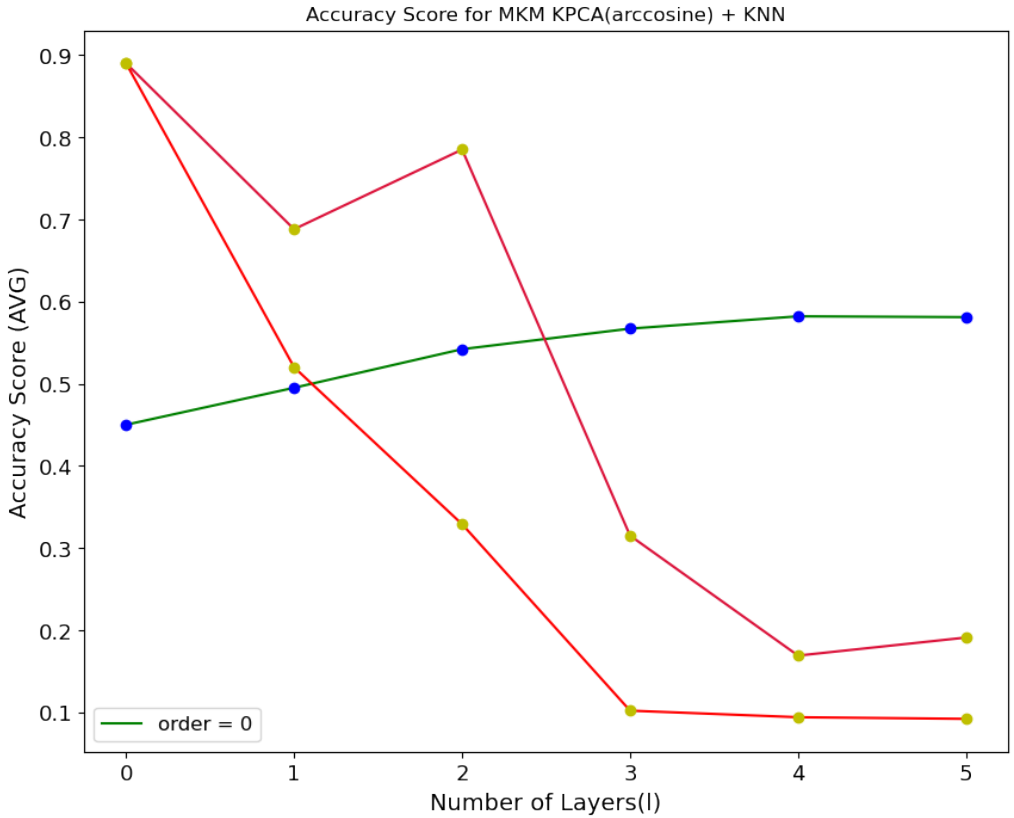
MKM: KPCA(arc-cosine) + KNN

Datasets: MNIST
Kernel Setting: Arc-cosines Kernel (order = 0,1,2)
classifier Setting: KNN(k)
Results:

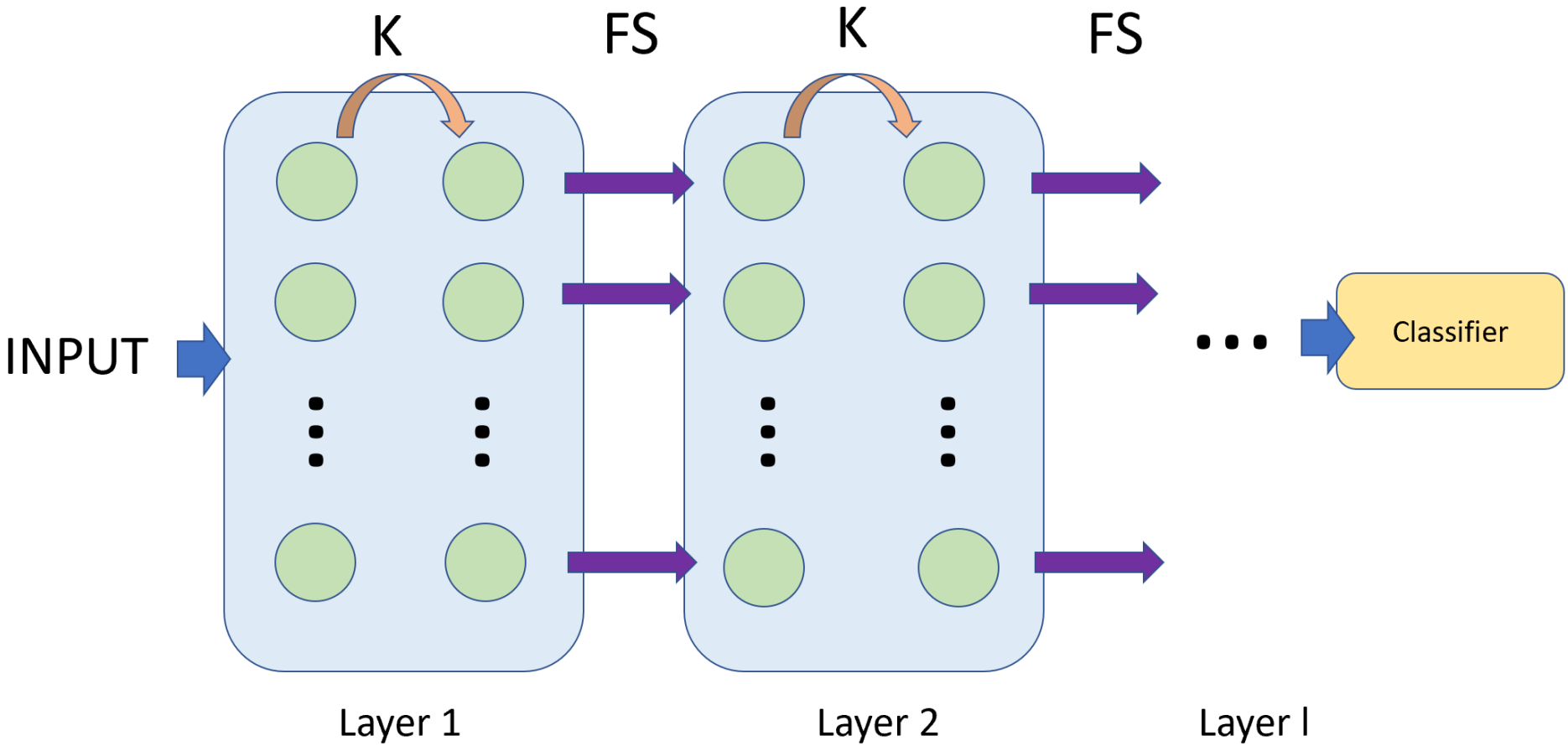


MKM: KPCA(arc-cosine) + KNN

Datasets: MNIST-BACK-IMAGES
Kernel Setting: Arc-cosines Kernel (order = 0,1,2)
classifier Setting: KNN(k)
Results:



Recall: Multilayer Kernel Machines (MKM)



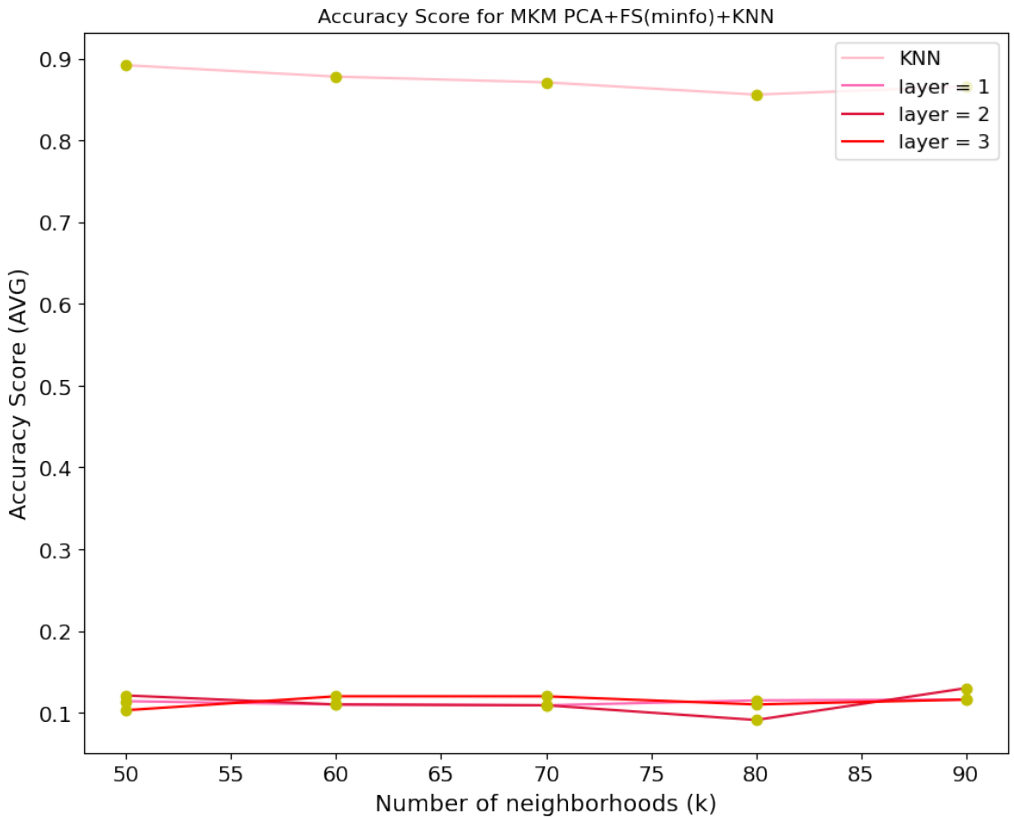
Feature Selection Implementation

- Calculating the mutual information score between each feature and the label.
- Ranking the feature based on descending order of mutual information score.
- Choose the top w feature and finish the selection.

Tricky Part: How to tune on w?
Methods:
Cross validation on basic KNN.

MKM: PCA + FS(minfo) + classifier

Datasets:MNIST
Kernel Setting: None
classifier Setting: KNN(k)
Feature Selection Methods: Mutual Info Ranking and cross validation
Results:



Time Complexity Analysis

Given datasets with n points and p features:

- For PCA: Covariance matrix computation is $O(p^2 n)$; its eigen-value decomposition is $O(p^3)$. So, the complexity of PCA is $O(p^{2n} + p^3)$. $O(\min(p^3, n^3))$ would imply that you could analyze a two-dimensional dataset of any size in fixed time, which is patently false.
- For Kernel PCA: Kernel computation takes $O(n^2)$. When applying PCA to the kernel matrix, we have dimension equals to the number of samples we have. Since here we need number of samples is greater than the number of features ($n > p$), we have higher time complexity than PCA.

Summary

- In the model of multilayer PCA with KNN, the result dosesn't vary a lot from layer changes.
- In the model of multilayer Kernel PCA with KNN, kernel plays an important roles. Arccosine kernel with 0 order outperforms all the kernel tested and it keepes the goood performance with multilayers. However, it doesn't work for higher order.
  (Theta is better preserved ?)
- In the model of multilayer PCA/Kernel PCA and festure selection and KNN, it performs really bad. (Some other FE methods tried are also not satisfying).
  (mutual info between feature is more important?)

Problems Raised in Experiments

- Technical problems:
  - Limited Computer memory caused value error.(when raise the layer of kernels to 2, problems will occur)
  - Long time for one result.(e.g: Kernel PCA and Cross Validation)
- Implementation Problems:
  - How to raise the order of kernels? (How to find the derivative?)
  - Tunning on the layer of multilayer kernel Machines. (Low layer performs better doesn't suit the deep learning intuition?)

Others Work on the same topic

- Cho's Study on LMNN
- A senior projrct is founded who implemented the same topic.
- Arc-cosine kernels: Acoustic modeling with infinite neural networks
- Save the time: Deep Core Kernel Machine

Future Works and Discussion

- Carefully go through the procedure of FS
- Review paper discussing the time complexity issue for kernel computation.
- Test on datasets used in the original paper. (if the time can be shorter).

Q & A

Welcome any related questions and we can learn together!

Thank you for listening!