Kernel Methods for Deep Learning

Akhil P M
SC14M044

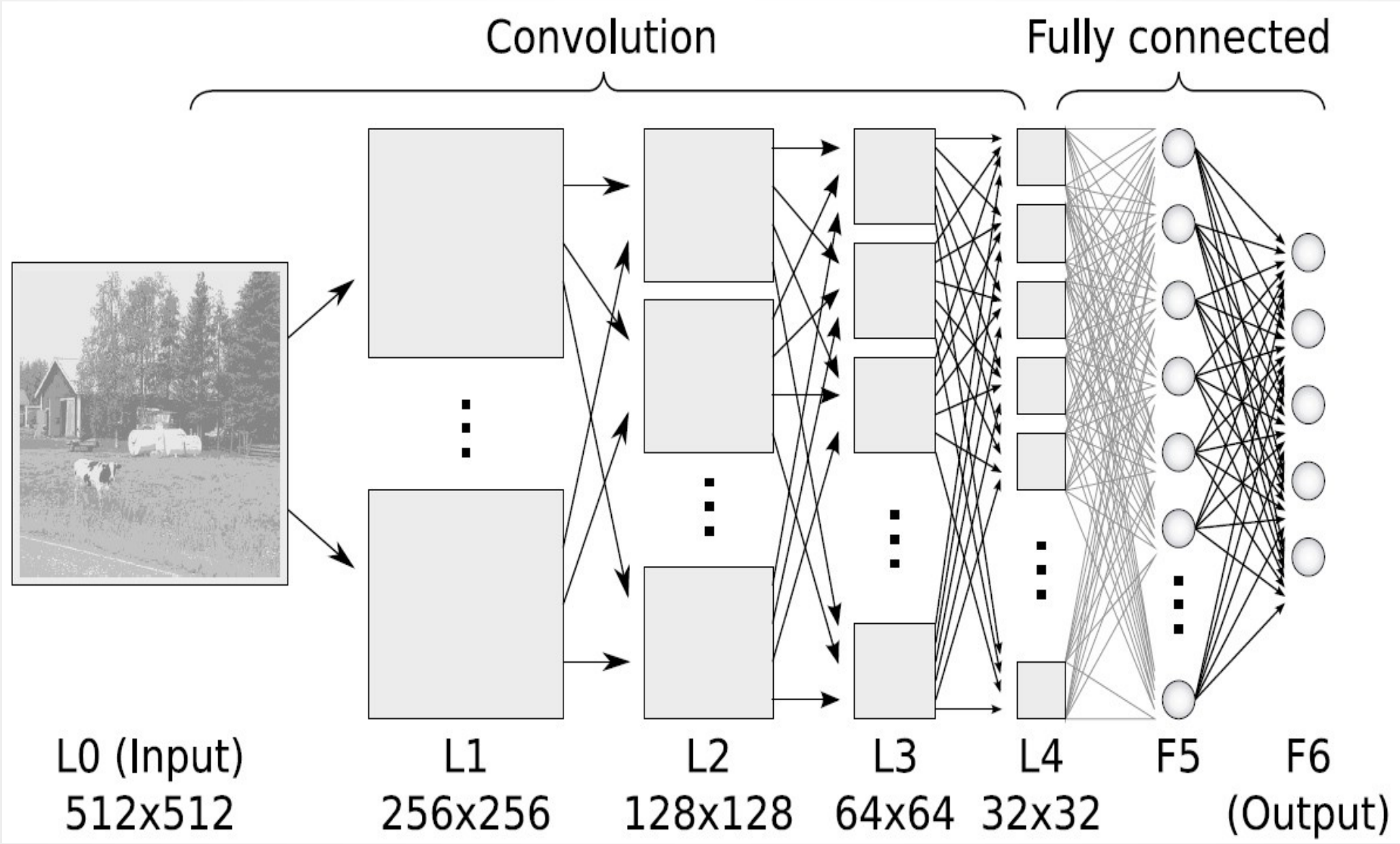
Deep Architectures

- Deep architectures learn complex mappings by transforming their inputs through multiple layers of non-linear processsing.
- These architectures are compositions of many layers of adaptive non-linear components.
- A ***wide range of functions*** can be parameterized by ***composing weakly nonlinear transformations***.
- The potential for ***combining*** unsupervised and supervised methods is an important feature of deep architectures.

Deep Architectures

- In deep architecures, the features produced by the lower layers represent ***lower-level abstractions***, that are combined to form high level features at the next layer, representing higher-level abstractions. So the fuctions at lower levels of abstractions should capture some simple aspects of data distribution, so that it is possible to learn first the simple functions and then ***compose*** them to learn more abstract concepts.
- The parameters at all levels of the model are trainable.

Deep Architectures



Shallow Architectures

- Shallow architecures are generally classfied into 3

  1. Type-1 architecutures - ***fixed preprocessing*** in the 1st layer

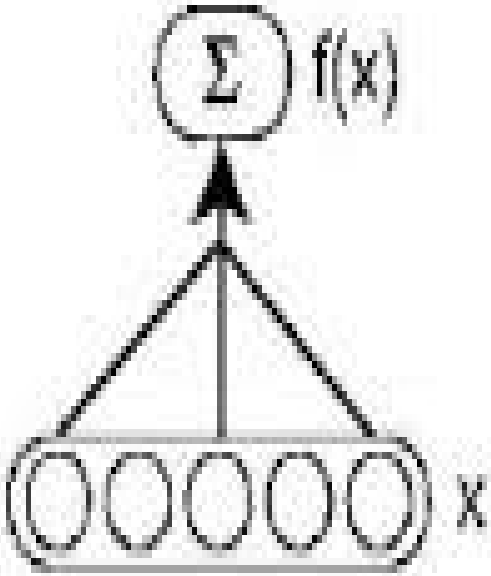
     Eg:- Perceptrons

  2. Type-2 architectures – ***template matchers*** in the 1st layer

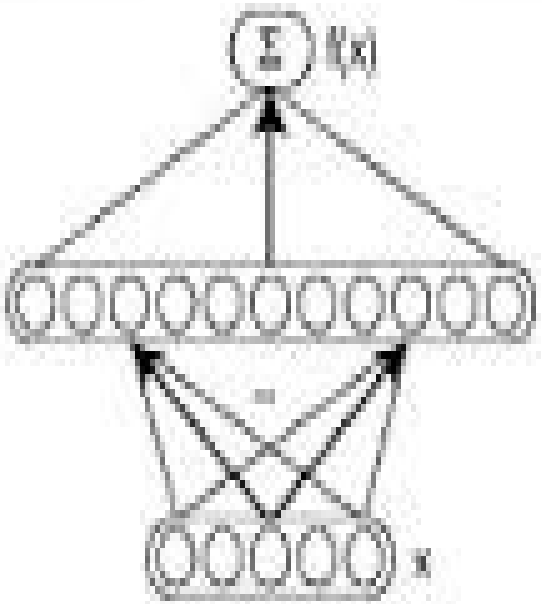
     Eg:- kernel machines

  3. Type-3 architectures – ***simple trainable basis functions*** in the 1st layer Eg:- Neural Network with single hidden layer.

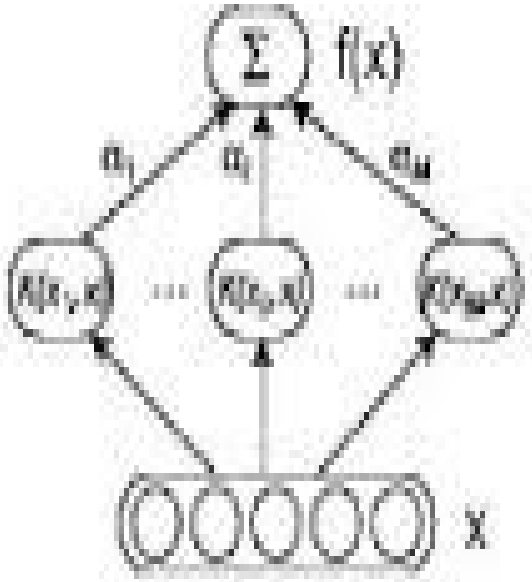
Shallow Architectures



(a) Linear model architecture



(b) Single layer neural network architecture



(c) Kernel SVM architecture

The tradeoff

- Deep architectures are genarally more ***difficult to train*** than shallow ones. They involve difficult nonlinear optimizations and many heuristics.
- Deep architectures ***support broad priors***, which can favour a wide set of functions. Whereas shallow architectures like SVMs favour a small set of functions.
- Optimization problem associated with Deep Neural Networks etc are ***non-convex***, but SVMs are solving convex optimization problem, hence global minimum can be attained.
- Generally the rule of thumb is to use a deep architecture, if the decision function involves ***many variations***. If the decision function if smooth, then shallow architectures like SVMs can be used.

Arc-cosine Kernels

These are a new family of kernel functions that mimic the computation in large neural networks. It is defined as

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

where x, y $\in \Re^d$ and $\Theta(z) = \frac{1}{2}(1 + sign(z))$
Here n is called the degree of the kernel and $k_n(x,y)$ is the nth order arc-cosine kernel. Let

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

be the angle between inputs x and y

Arc-cosine Kernels

This can be alternatively represented as

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

where

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

for n=0,1,2

$$J_0(\theta) = \pi - \theta$$

$$J_1(\theta) = sin\theta + (\pi - \theta)cos\theta$$

$$J_2(\theta) = 3sin\theta cos\theta + (\pi - \theta)(1 + 2cos^2\theta)$$

Arc-cosine Kernels

and

$$k_0(x, x) = 1$$

kernel maps imputs x to the unit hypersphere in feature space

$$k_1(x, x) = \|x\|^2$$

kernel preserves the norm of inputs

$$k_n(x, x) = \|x\|^{2n}$$

kernel expands the dynamic range of inputs

Relationship with Neural Networks

Consider the single layer neural network shown below whose weights W_{ij} connects the jth input unit to the ith output unit. The network maps input x to output f(x) by applying a non-linear map, thus

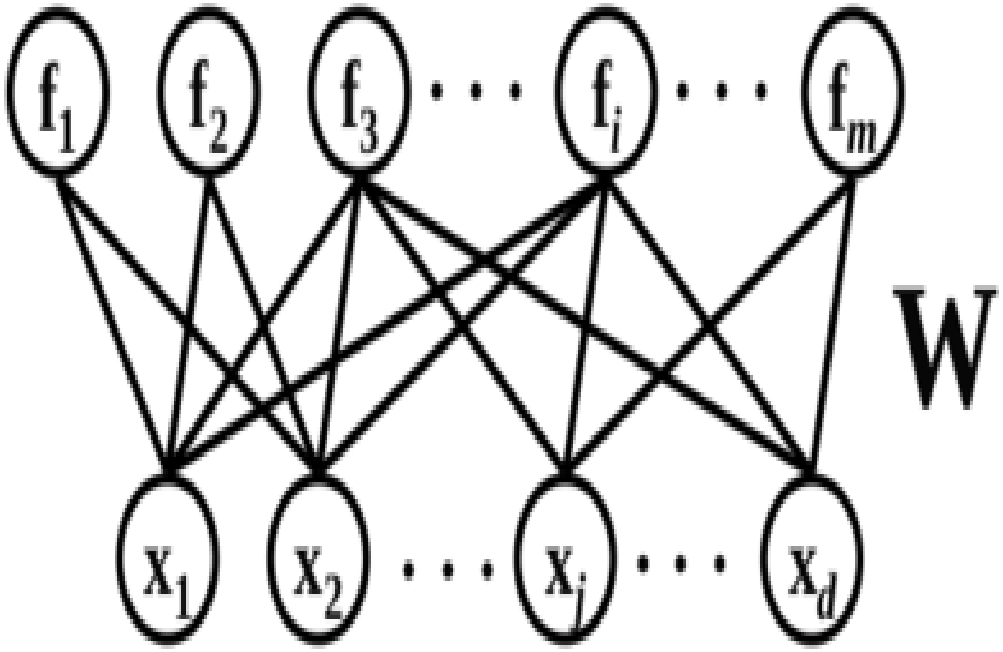
$f(x) = g(W \cdot x)$

where the non-linearity is described by the network's activation function

$$g_n(z) = \Theta(z)z^n$$

this activation function is called one-sided polynomial activation function, whose graph for different n values is shown below.

Relationship with Neural Networks



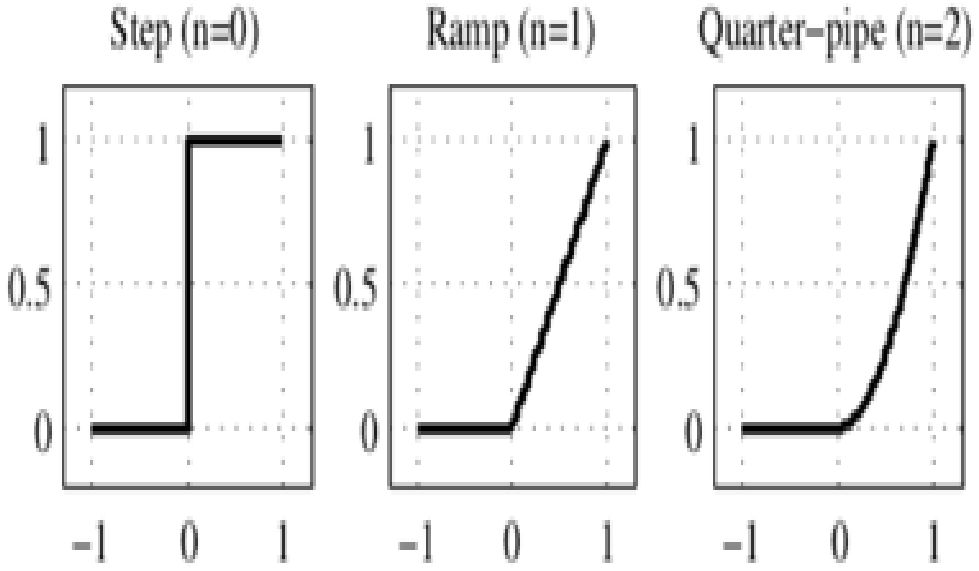


Figure 1: Single layer network and activation functions

Relationship with Neural Networks

let f(x) and f(y) be the outputs corresponding to input data points x and y. Then the inner product of f(x) and f(y) is given by

$$f(x) \cdot f(y) = \sum_{i=1}^{m} \Theta(w_i \cdot x)\Theta(w_i \cdot y)(w_i \cdot x)^n (w_i \cdot y)^n$$

where w_i denotes the ith row of weight matrix W and m is the no of output units. Assume W_{ij} are Gaussian distributed with zero mean and unit variance and the network has an infinite number of output units. Then

$$lim_{m \to \infty} \frac{2}{m} f(x) \cdot f(y) = k_n(x, y)$$

Relationship with Neural Networks

Arc-cosine kernels infeature spaces that mimic the sparse, nonegative, distributed representationsof single-layer threshold networks.

computation in multilayer networks

$$k^{(l)}(x,y) = \underbrace{\Phi(\Phi \dots \Phi(x)))}_{\text{l times}} \cdot \underbrace{\Phi(\Phi \dots \Phi(y)))}_{\text{l times}}$$

Motivation : Intuitively, if k(x,y) mimics the computation in a single-layer network, then the iterated mapping above should mimic the computation in a multilayer network.

Relationship with Neural Networks

Induction for iterative computation

$$k_n^{(l+1)}(x,y) = \frac{1}{\pi}\left[k_n^{(l)}(x,x) \times k_n^{(l)}(y,y)\right]^{\frac{n}{2}} J_n(\theta_n^{(l)})$$

where

$$\theta_n^{(l)} = cos^{-1}\left(k_n^{(l)}(x,y)\left[k_n^{(l)}(x,x) \times k_n^{(l)}(y,y)\right]^{\frac{-1}{2}}\right)$$

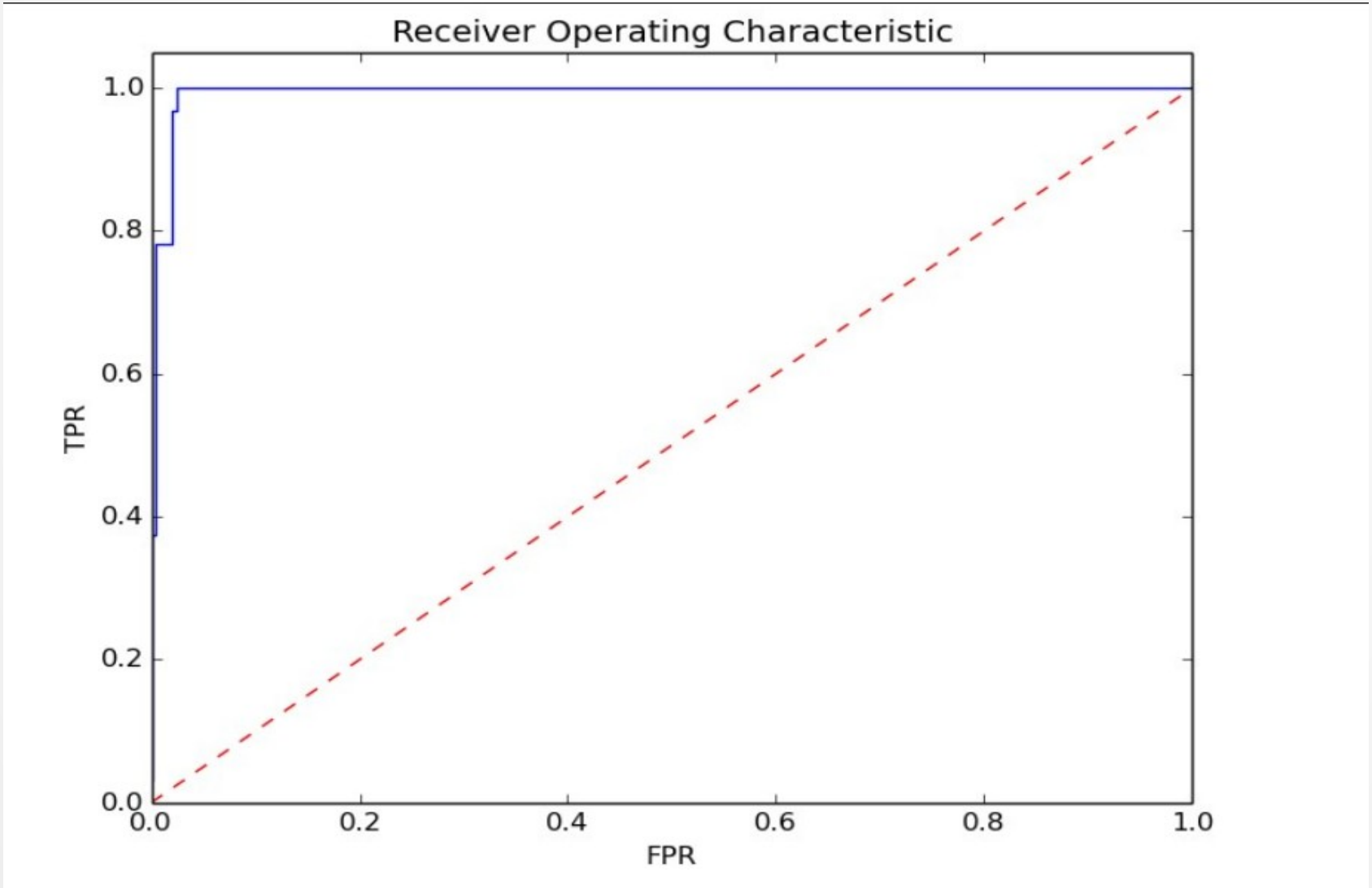
The results reveal that multilayer arc-cosine kernels often perform better than their single-layer counterparts.

Experimental Results

DataSet	*Arc Cosine Kernel*	*Othe Kernel(Best Result)*
Synthetic Dataset (Binary Classification)	0.99125 (3,3,1)	0.9975 (gaussian)
Wine Dataset (multiclass – 3 class)	0.97671 (0,3,3,1)	0.98219 (gaussian)
IRIS dataset (multiclass – 3 class)	0.98442 (1,1)	0.97329 (gaussian)
Breast Cancer Wiscosin (Binary classification)	0.9644 (0,1,1)	0.97293 (polynomial)
Digits Dataset (multiclass – 10 class)	0.98709 (3,1,1,1)	0.98559 (polynomial)

ROC Analysis of Breast Cancer Data

AUC = 0.99385



MNIST Dataset

Layer configuration	Accuracy
1	0.9512(1.0)
1 1	0.9581(1.0)
1 1 1	0.9607(1.0)
1 1 1 1	0.9619(1.0)
1 1 1 1 1	0.9631(1.0)

Running time Analysis

Analysis of Running Time(in svmlight library using iris dataset)

a.Using Arc-cosine kernel(two layer with n values 0, 1)

Operation	%time taken
QP	1.98%
Kernel	***18.05%***
Argmax	4.78%
Psi(combined feature space)	0.02%
Init	0.01%
Cache Update	58.70%
Cache Const	0.03%
Cache add	3.09%

Running time in CPU-seconds = 2.93


Thank
You