Combining Infinity Number Of Neural Networks Into One

Bo Tian
address: Reichenhallerstr 5A, 14199, Berlin, Germany
email: tmsfld@googlemail.com

Abstract

One of the important aspects of a neural network is its generalization property, which is measured by its ability to make correct prediction on unseen samples. One option to improve generalization is to combine results from multiple networks, which is unfortunately a time-consuming process. In this paper, a new approach is presented to combine infinity number of neural networks in analytic way to produce a small, fast and reliable neural network.

keywords— convolution, generalization, ensemble, transfer function

1 Introduction

Although it's proved that multi-layer feed forward neural network can be a universal approximator to any given function under mild conditions provided by [1, 5], the construction of neural network remains difficult to present day. It's often noticed that even when a solution with error close to zero is obtained, the network can still fail to make correct prediction on unseen samples (i.e. test samples). It does not help much by increasing the size or number of hidden layers and could even cause overfitting.

According to [13], this phenomenon can be illustrated as follows. The training samples can be actually represented by multiple functions. One neural network solution may well estimate one of those functions, but that function may not necessarily be precise on test set. This explanation implies that even the global minimum of the error function is obtained, the generalization ability is not guaranteed by the solution.

The term *generalization* is used to describe the ability of a neural network to make correct prediction on test samples. One common approach to improve generalization is referred as *neural network ensemble*, which is essentially to combine results from multiple networks that differ from each other [[4, 8, 9]]. The combining methods include taking average (or weighted average) for continuous interpolation, or majority voting for classification problems. Sharkey et al. suggested selecting those networks with great diversity to improve generalization, so that the error will hopefully cancel out with each other. Here diversity means that if the solutions partially fail on test set, they should fail differently. The networks with great diversity are thought to be those networks trained on different training samples, with different architectures, having different number of nodes or starting from different initial parameters.

Even though neural network ensemble generalizes quite well, both the construction of the ensemble and its usage requires much computation resource and thus limits its application.

It is notable that some researchers tried to replace popular transfer functions like sigmoid or RBF function with other nonlinear functions. [2] presented a list of those attempts . Some of those efforts are contributed to improve generalization. However, as mentioned in the paper, it's hard to evaluate their advantages and disadvantages. The most popular transfer functions remain as sigmoid and RBF function.

In the training of neural network, it's observed that when a solution with small error is obtained, one can also find multiple solutions with slightly bigger but still small error around the found solution. Each of those solutions shows different generalization property as diversity requires. This fact suggests the possibility to combine the results from networks with their parameters close to the existing solution to improve generalization.

Combining nearby solutions has a unique advantage as it can be done analytically, namely taking weighted average from nearby solutions can be done by means of convolutions with Gaussian function in parameter space. In this paper, this technique is refered as *convolutional tunneling*, or *CT* in short. A new type of transfer function is thus created based on the CT technique.

The neural network using convolutional tunneling techniqueis referred as a *CT Network*, or *CTN* in short. A classical optimization (e.g. Levenberg-Marquardt Algorithm by [10, 11]) is then performed to find the best solution for the network. As output of a CTN is the average (and thus smoothed) version of the original output, the error function with CTN is also smoothed and thus finding the minimum is less likely to be trapped by local minima.

The fact that a CTN is one single network (instead of ensemble of multiple networks) suggests less computation is required to construct and to use the network.

In this paper, we construct a neural network with one single hidden layer to demonstrate the concept of CTN and its ability of generalization. The paper is organized as follows: section 2 introduces the concept of CTN; section 3 demonstrates its generalization ability; section 4 describes a new schema of finding global minimum based on the concept of CTN and section 5 then discusses some potential extensions.

2 Convolutional Tunneling Network

As mentioned above, a CTN is a neural network whose output is the weighted average of outputs from all nearby networks. Mathematically, the weighted average means taking convolution of the output with Gaussian function over the entire parameter space.

Let $t(\mathbf{x} \mid \mathbf{p})$ denote the output of a network on input $\mathbf{x}$ with parameter $\mathbf{p}$, then output of the CTN is $t(\mathbf{x} \mid \mathbf{p}) * g(p_1|0,\sigma) * g(p_2|0,\sigma) * \ldots$, where function g is Gaussian function, σ is the bandwidth of the CTN. It's easy to see that the CTN will converge to the original network when the bandwidth is 0.

2.1 Convolutional tunneling of SLFN

In theory, a MLP with one single hidden layer is enough to approximate any continuous nonlinear function. In this section, single hidden layer FFNN (SLFN) is investigated.

SLFN consists of one input layer, one hidden layer and one output layer. Without loss of generality, assume the output layer has only one output node. The network is illustrated as follows:

The transfer function of a node can be viewed as two parts: the *activation function* that describes the total input of the node, and the *output function* that coverts the input of the node to an output value.

Let

- x_k denote the k-th input variable. In case of network with bias, the last one is fixed as 1;
- $\nu_{j,k}$ denote the weight from input k to hidden node j;
- z_j denote the input of hidden node j
  $z_j = \sum_k \nu_{j,k} x_k$;

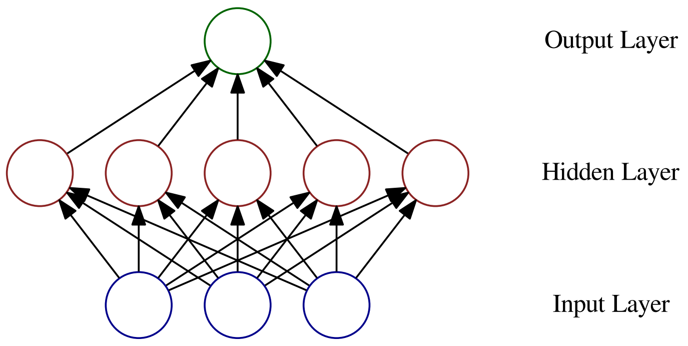

Figure 1: MLP with single hidden layer

- o_j denote the output of hidden node j
  $o_j = \Psi(z_j)$, where $\Psi(x)$ is the transfer function;
- ω_j denote the weight from hidden node j to the output node;
- t denote the output of the network
  $t = \sum_j \omega_j o_j$;
- y denote the observed value of a given training sample;
- e denote the error on the given training sample
  $e = t - y$

Here the activation function of the hidden nodes is an simple weighted sum, like most MLP does.

In the following discussion, the output of a CTN is presented with convolution.

The output function

Let $\bar{t}$ denote output of the CTN:

$$\bar{t} = t * g(\omega|0, \sigma) * g(\nu|0, \sigma)$$

It's hard (if not impossible) to obtain the convolution using classical output function like logistic function or Tanh. To make the convolution easier, the output function is chosen as the CDF of standard normal probability density function $\Phi(x)$ for its nice integral properties.

$$\Phi(x) = \int_{-\infty}^{x} \phi(t)dt$$

where $\phi(x)$ is standard normal probability density function

$$\phi(x) = \frac{1}{2\pi} e^{-\frac{x^2}{2}}$$

Gaussian function $g(x|0,\sigma)$ can be also expressed with standard normal probability density function as:

$$g(x|0,\sigma) = \frac{1}{\sigma}\phi\Big(\frac{x}{\sigma}\Big)$$

Output of the CTN

Let $\bar{e} = \bar{t} - y$, we have the error function $\bar{E}$ as:

$$\bar{E} = \sum_i \bar{e}_i^2 = \sum_i (\bar{t}_i - y)^2$$

According to the definition of $\bar{E}$, the smoothed error function will converge to the original error function when $\sigma = 0$.

As mentioned above, the output of the CTN is the convolution of the output of the original network and a high dimensional Gaussian function in parameter space.

As

$$\begin{aligned}(a+bx) * g(x|0,\sigma) &= \int_{-\infty}^{\infty} (a + b(x-t))\frac{1}{\sigma}\phi\Big(\frac{t}{\sigma}\Big)dt \\ &= a + bx - b\sigma \int_{-\infty}^{\infty} t\phi(t)dt \\ &= a + bx\end{aligned}$$

Thus

$$\begin{aligned}t * g(\omega_1|0,\sigma) * g(\omega_2|0,\sigma) * \ldots &= \sum_j \omega_j o_j * g(\omega_1|0,\sigma) * g(\omega_2|0,\sigma) * \ldots \\ &= \sum_j \omega_j o_j \\ &= t\end{aligned}$$

That is, the convolution of the output with ω is still the original output.

As

$$\begin{aligned}\Phi(a+bx) * g(x|0,\sigma) &= \int_{-\infty}^{\infty} \Phi(a+bt)\frac{1}{\sigma}\phi\left(\frac{t-x}{\sigma}\right)dt \\ &= \int_{-\infty}^{\infty} \Phi(a+bx+b\sigma t)\phi(t)dt \\ &= \Phi\left(\frac{a+bx}{\sqrt{1+b^2\sigma^2}}\right)\end{aligned}$$

Let $\bar{o}_j$ denote the output of j-th hidden node in the CTN. As $o_j = \Phi(z_j)$, we have

$$\begin{aligned}\bar{o}_j &= o_j * g(\nu_{j,1}|0,\sigma) * g(\nu_{j,2}|0,\sigma) * \dots \\ &= \Phi\left(\sum_k \nu_{j,k}x_k\right) * g(\nu_{j,1}|0,\sigma) * g(\nu_{j,2}|0,\sigma) * \dots \\ &= \Phi\left(\frac{\sum_k \nu_{j,k}x_k}{p_1(\mathbf{x};\sigma)}\right) \\ &= \Phi\left(\frac{z_j}{p}\right)\end{aligned}$$

where $p = \Pi_k\sqrt{1+\sigma^2 x_k^2}$.

Thus

$$\begin{aligned}\bar{t} &= \sum_j \omega_j \bar{o}_j \\ &= \sum_j \omega_j \Phi\left(\frac{z_j}{p}\right) \\ &= \sum_j \omega_j \Phi\left(\frac{\sum_k \nu_{j,k}x_k}{\Pi_k\sqrt{1+\sigma^2x_k^2}}\right)\end{aligned}$$

As shown above, the output of the CTN is similar to the original network. The difference is that the additive activation function is replaced by a nonlinear function.

The activation function of the CTN is denoted by

$$A_{S1}(\mathbf{x};\nu,\sigma) = \frac{\sum_k \nu_k x_k}{\Pi_k\sqrt{1+\sigma^2x_k^2}}$$

A_{S1} is referred as *superpose activation function* [1] . Thus the CTN can be also defined as a neural network that takes superpose activation function as its activation function and CDF function $\Phi(x)$ as its output function [2]. Accordingly, the function $\Phi(A_{S1}(\mathbf{x};\nu,\sigma))$ is referred as *superpose transfer function.*

In order to utilize classical approach such as LMA for optimization, the Jacobi matrix is deduced as follows:

The Jacobi matrix

As mentioned above, $\bar{e}$ denotes the error on one of the samples. $\bar{e} = \sum_j \omega_j \bar{o}_j$, thus it is straightforward that

$$\begin{aligned}
\frac{\partial \bar{e}}{\partial \omega_j} &= \bar{o}_j \\
\frac{\partial \bar{e}}{\partial \nu_{j,k}} &= \omega_j \frac{\partial \bar{o}_j}{\partial \nu_{j,k}} \\
&= \omega_j \frac{\partial \bar{o}_j}{\partial \bar{z}_j} \frac{\partial \bar{z}_j}{\partial \nu_{j,k}} \\
&= \omega_j \frac{1}{p} \phi\left(\frac{z_j}{p}\right) x_k
\end{aligned}$$

3 Generalization property

As mentioned above, the output of CTN is expected to be more generalized as it combines the nearby solutions. In this section, an experiment is carried out on solving the two-spiral problem with CTN.

There are a few reasons to choose two-spiral problem for the demonstration. First, the two-spiral problem is a well known benchmark; second, the problem was considered to be difficult [[3]], so that it's a suitable candidate to demonstrate the performance of a neural network; and last, as the input of the problem is in 2D plane, it is easy to be visualized for demonstration purpose.

Visual comparison

[1] Here the subscript 1 indicates the activation function is to be used with a sigmoid function range from 0 to 1.

[2] notice that the transfer function can be generalized to other sigmoid functions in section 5

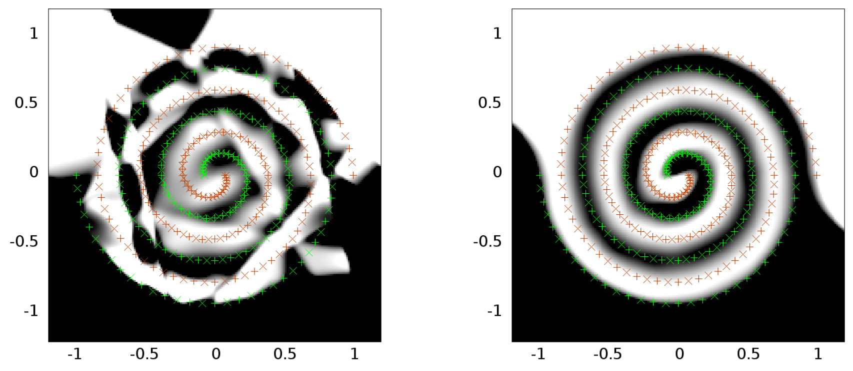

Figure 2: Classic result vs CTN

Figure 2 is the visualization of two solutions to the two-spiral problem. Both are produced by a MLP with single hidden layer with 30 hidden nodes. The transfer function used at left side is CDF function Φ. While the transfer function used at right side is superpose transfer function with bandwidth set as 2.0. Positive samples in the figure are red and the negative samples are green. Training samples are marked with "+" and test samples are marked with "x".

The image produced by the CTN solution shows little sign of irregularity. It looks more smooth and perfectly follows the spiral curvature. The generalization ability can be easily seen from the visual result. In fact this CTN solution gives correct classification on all samples from test set.

Comparison of generalization

For classification problems, the performance of a neural network is measured by *success rate*, which is the number of correct prediction divided by total number of test samples. We call a solution as a *perfect solution* when it has a 100% success rate. It's a common practice to choose the best solution from a couple of trials, thus it's important for an approach to have higher chance to produce acceptable solutions. This ability is measured by *acceptance rate*, which is the number of solutions with success rate greater than a threshold divided by total number of trials.

In this experiment, we set the network size (number of hidden nodes) as 20, 30, 50 and 80, and produced 100 solutions for each case. And plot the acceptance rate as a function of number of nodes when sr(success rate)

> 0.9, sr > 0.95 and sr = 1.

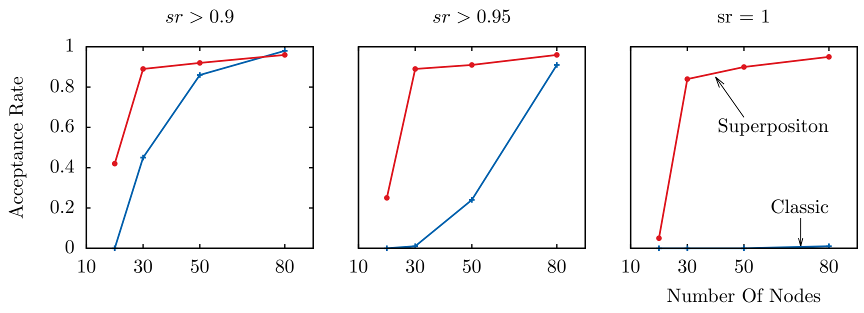

Figure 3: Acceptance rate Comparison

As we can see from the plot, CTN reaches notably higher acceptance rate when there are over 30 hidden nodes for all success rate thresholds, while the classic approach requires more hidden nodes to reach the same level.

It's noticeable that classic approach can hardly produce a perfect solution - there is only one observation of perfect solution out of 100 trials when there are 80 hidden nodes, while the CTN produce perfect solutions constantly (with more than 90% chance when there are over 30 hidden nodes). And there are even 5 observations of perfect solutions with only 20 hidden nodes.

Comparison of training time

The average training time is plotted as a function of the number of the hidden nodes.

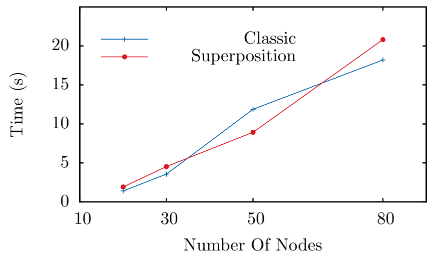

Figure 4: Time Comparison

As classic and CTN share the same network architecture and optimization method, they take similar training time when the network size is also the same. As the training time heavily depend on the size of the network, it's not surprising that it's much faster to get a better solution with CTN as it requires much smaller network to do the job.

Summary In this section, it's demonstrated that CTN is not only fast but also reliable.

First, comparing to its classic counterpart, a small number of nodes is enough to make a performing CTN, while a small network means less effort to train the network and also less effort to make prediction with the network. Second, comparing to neural network ensembles, there is no need to train multiple neural networks nor combine multiple results from those networks.

CTN is reliable as it is actually a combination of all nearby networks, thus it can be viewed as an ultimate neural network ensemble of infinity number of neural networks.

4 Finding global minimum with convolutional tunneling

In the above sections, we demonstrated how to construct a CTN with generalization as main concern. In this section, we will show how it could be used in searching global minimum.

Finding global minimum is referred as *global optimization*, which differs from local searching by its objective as to find minimum (or maximum) over all inputs. The most popular strategies to solve the problem can be categorized into deterministic methods [6], stochastic methods [7] and heuristics [12].

As maximization of a function can be viewed as minimization on the additive inverse of the function, we hereby refer optimization as minimization to save a few words. By definition, the target of global optimization is to find the smallest minimum out of all minima. In practice, this target is usually converted to searching the best possible solution. Thus the performance of a searching method can be measured by its efficiency to get a better solution.

4.1 The searching schema

Ideally, if function $F(x)$ has finite number of local minima, one can try local search from various starting points to find all of them and take the solution with the smallest value. However, this is not practical as the number of minima is usually incredibly huge in many real-life applications.

As we noticed above, in case the analytic expression of convolution of F(x) and Gaussian function can be obtained, the convolution of $F(x)$ will smooth out small ridges and valleys used to be in the objective function landscape. The smoothness increases along with the value of σ. One could expect that the number of local minima is also reduced during the process. According to the discussion above, when σ is big enough, it will be easy to find the global minimum of the smoothed version of objective function.

On the other hand, it is well known that the quality of the result from a local search heavily depends on the starting point. If the global minimum of $\bar{F}(x;\sigma)$ is obtained, and the smooth factor σ is small enough, then the solution can be used as the starting point to search the global minimum of $F(x)$. Furthermore, if global minimum of $\bar{F}(x;\sigma_1)$ is obtained, the solution can be used as the starting point to search the global minimum of $\bar{F}(x;\sigma_2)$ if $\sigma_1 > \sigma_2$ and $\sigma_1 - \sigma_2$ is small enough.

As the value of σ decreases, the smoothed version of target function $F(x)$ will eventually degenerate to the objective function $F(x)$ itself.

A **two-phase searching schema** can be constructed as follows:

- Phase I: search for multiple local minima for a CTN with a large σ and choose the solution with the smallest function value.
- Phase II: repeatedly reduce σ value and perform local search starting from previous solution until σ is small enough, then set σ as 0 to perform final search.

4.2 Experiment result

The optimization method is demonstrated with MLP with single hidden layer. Instead of using convolution of the error function, we use the convolution on the output of the network as described in section 2. As the convolution is to smooth out the objective function, the same reasoning mentioned above still holds true.

The configuration of the network is:

- number of nodes: 30;
- activation function: $\sum_k \nu_{j,k} x_k$
- output function: CDF function $\Phi(x)$

Following the searching schema described above, first find 5 different solutions of a CTN with $\sigma = 2.0$ and pick up the solution with the smallest

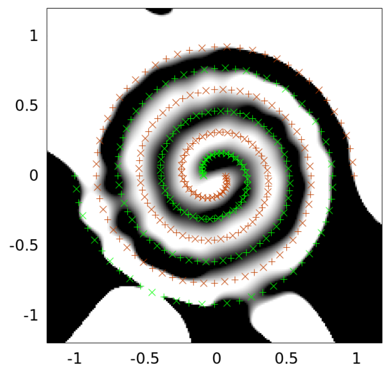

Figure 5: Output of global searching with CTN

error; then continuously replace σ by 0.7σ after each local search until $\sigma < 0.1$, then set $\sigma = 0$ and make the final search.

Figure 5 shows the result. Although the result is not as good as we can get from CTN, it is clearly better than those trained by classic approach with distinct spiral shape. The squared training error of this solution is 0.00980 and success rate is 99.5%. By comparison, as observed in the experiment described in section 3 the smallest training error of classic approach is 0.05297, and there is only 1 case out of 100 trials reaches similar success rate.

4.3 Variation and limitation

One limitation of this searching schema is that it requires analytic expression of the convolution. In some cases, this requirement is hard to be fulfilled if not impossible.

In the above example, the objective function (i.e. error function of the neural network) follows the form of $\sum f^2(x)$ and we take convolution on $f(x)$ instead of directly on $F(x)$ as mentioned in section 4.1. This approach suggests that $F(x) = \sum f^2(x)$ case may be generalized to the form of $F(x) = f(g(x))$. In this case, replace $g(x)$ with its convolution might also smooth out the objective function and therefore the searching schema can be applied. This will simplify the required convolution in case the direct convolution is hard to be obtained. However, This hypothesis requires further investigation.

It is also worth to mention that the searching schema mentioned above might fail if the global minimum is hidden in a deep but narrow valley. In this case though, it's also hard for all existing searching algorithms to find it.

5 Work with different sigmoids

Based on the knowledge discussed above, the CT approach can be extended to other sigmoid functions as well.

5.1 Gaussian error function (erf)

Gaussian error function ($erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$) can be expressed by CDF function $\Phi(x)$ as $erf(x) = 2\Phi\left(\sqrt{2}x\right) - 1$, thus it share the same integral property of $\Phi(x)$.

When use $erf(x)$ as the output function, the output of a MLP with single hidden layer is:

$$t = \sum_j \omega_j erf\left(\sum_k \nu_{j,k} x_k\right)$$

Follwing similar deducing in section 2, the output of the CTN is then

$$\bar{t} = \sum_j \omega_j erf\left(\frac{\sum_k \nu_{j,k} x_k}{\Pi_k \sqrt{\frac{1}{2} + \sigma^2 x_k^2}}\right)$$

Comparing $\bar{t}$ with t, the difference is the activation function. Define

$$A_{S2}(\mathbf{x}; \nu, \sigma) = \frac{\sum_k \nu_k x_k}{\Pi_k \sqrt{\frac{1}{2} + \sigma^2 x_k^2}}$$

then the output of node j can be expressed as

$$\bar{o}_j = erf(A_{S2}(\mathbf{x}; \nu_j, \sigma))$$

The error of the CTN on a sample is $\bar{e} = \bar{t} - y$. Its derivative is:

$$
\begin{aligned}
\frac{\partial \bar{e}}{\partial \omega_j} &= \bar{o}_j \\
\frac{\partial \bar{e}}{\partial \nu_{j,k}} &= \omega_j \frac{\partial \bar{o}_j}{\partial \nu_{j,k}} \\
&= \omega_j \frac{\partial \bar{o}_j}{\partial \bar{z}_j} \frac{\partial \bar{z}_j}{\partial \nu_{j,k}} \\
&= \omega_j \frac{1}{p} e^{-(z_j/p)^2} x_k
\end{aligned}
$$

where $p = \Pi_k \sqrt{\frac{1}{2} + \sigma^2 x_k^2}$.

5.2 Using tanh

Although the superpose transfer function is deduced from convolution of function Φ and Gaussian function, the form of the function is simply changed by its activation function. As function tanh looks very much like erf, one could expect that they behave similarly.

Following the reasoning above, the output of j-th node $\bar{o}_j$ is directly set as

$$\bar{o}_j = \tanh(A_{S2}(\mathbf{x}; \nu_j, \sigma))$$

The total output is still $\bar{t} = \sum_j \omega_j \bar{o}_j$.

As $\tanh'(x) = 1 - \tanh^2(x)$, we have the derivative of the error as:

$$
\begin{aligned}
\frac{\partial \bar{e}}{\partial \omega_j} &= \bar{o}_j \\
\frac{\partial \bar{e}}{\partial \nu_{j,k}} &= \omega_j \frac{1}{p} \big(1 - \bar{o}_j^2\big) x_k
\end{aligned}
$$

where $p = \Pi_k \sqrt{\frac{1}{2} + \sigma^2 x_k^2}$.

A test with two-spiral problem is carried out to check the theory. The test result is not included in the paper as it looks identical to the CTN result shown in right side of figure 2.

5.3 Using logistic function

Use similar approach above, as logistic function is similar with CDF function Φ, the output of j-th node is directly set as

$$\bar{o}_j = S(A_{S1}(\mathbf{x}; \nu_j, \sigma))$$

The total output is still $\bar{t} = \sum_j \omega_j \bar{o}_j$.

As $S'(x) = S(x)(1 - S(x))$, we have the derivative of the error as:

$$\frac{\partial \bar{e}}{\partial \omega_j} = \bar{o}_j$$
$$\frac{\partial \bar{e}}{\partial \nu_{j,k}} = \omega_j \frac{1}{p} \bar{o}_j (1 - \bar{o}_j) x_k$$

where $p = \Pi_k \sqrt{1 + \sigma^2 x_k^2}$.

The test result is also the same as the CTN result shown in right side of figure 2.

6 Discussion

In this paper, the concept of CTN is presented. Such a concept opens a new door towards better neural network design and introduces a new way to solve global optimization.

And it's worth to mention that the CTN is easy to be implemented based on existing implementations as CTN differs from classical network only with the activation function.

As far as noticed, there are following issues need to be investigated.

Bandwidth of CTN

In the demonstrations mentioned above, the bandwidth of CTN σ is set as 2. The result is not sensitive to the value of σ. However, if the value is too small, for example $\sigma = 0.5$, the result become less generalized. The output image looks similar to figure 5. While if the bandwidth is too big, say 5.0, the visual result becomes blurry and it starts to lose the ability to make prediction.

Currently the value is chosen with trial on error basis. However, it's desirable to have a method to choose the value of the bandwidth.

Proof of universal approximator

In section 2, the superpose activation function is deduced from the convolution when the output function is chosen as CDF function $\phi(x)$. But the test results suggest the generalization ability actually come from the superpose activation function $A_{S1}(\mathbf{x}; \nu_j, \sigma)$ and $A_{S2}(\mathbf{x}; \nu_j, \sigma)$.

This fact suggests that any transfer function with CTN activation function and sigmoid output function could be a universal approximator.

The mathematical proof is still missing yet.

Proof of global searching method

In section 4, a new global search schema is provided. And it's also demonstrated it gives better solution. But how far can we go from there? Is it possible to find the true global minimum under some conditions with this approach? Some mathematical analysis is required to reveal its potential and limitations.

References

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[2] Włodzisław Duch and Norbert Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212, 1999.

[3] Gary William Flake. Square unit augmented radially extended multilayer perceptrons. In *Neural Networks: Tricks of the Trade*, pages 145–163. Springer, 1998.

[4] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001, 1990.

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[6] Reiner Horst and Hoang Tuy. *Global optimization: Deterministic approaches.* Springer Science & Business Media, 2013.

[7] AHG Rinnooy Kan and GT Timmer. Stochastic global optimization methods part i: Clustering methods. *Mathematical programming*, 39(1):27–56, 1987.

[8] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.

[9] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[10] Kenneth Levenberg. A method for the solution of certain non–linear problems in least squares. 1944.

[11] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[12] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

[13] Amanda JC Sharkey and Noel E Sharkey. Combining diverse neural nets. *The Knowledge Engineering Review*, 12(03):231–247, 1997.