Optimization by TCNN in discrete and continuous domains

* HNN, TCNN, multimodal function optimization
* TCNN
  + Chaotic (bifurcation + lapunov)
  + Discreet (equations)
* Compare with GA, swarm, Monte Carlo, …
* TCNN analog
  + Chaotic functions
  + Equations
  + Simulink schema

<https://en.wikipedia.org/wiki/Differential_evolution>

<https://en.wikipedia.org/wiki/Memetic_algorithm>

<https://en.wikipedia.org/wiki/Graduated_optimization>

https://en.wikipedia.org/wiki/Global\_optimization

1. Introduction
2. TCNN
   1. Chaotic dynamic
   2. Compare
3. TCNN analog
   1. Chaotic analog function
   2. Experimental results
   3. Analog schema (Simulink)
4. Conclusions
5. Introduction

В данной статье представлено сравнение TCNN с различными алгоритмами для решения задач поиска глобального минимума. В статье так же рассматривается переход от дискретной модели TCNN к аналоговой, и результаты моделирования аналогового представления.

1. Описание TCNN

TCNN описанная и предложенная [] основывается на сети хопфилда и методе отжига.

TCNN описывается следующей системой уравнений

где

xi – выход нейрона

yi – внутреннее состояние нейрона

wij – весовой коэффициент между нейронами

Ii – смещение нейрона

zi – коэффициент обратной связи

Отличие от сети хопфилда заключается в том, что в начальный момент, в сети происходит хаотический поиск, который обуславливается хаотической функцией () налог температуры в методе отжига. Со временем влияние хаоса на систему уменьшается, что приводит к более выраженной динамики сети хопфилда, которая основана на градиентном методе.

Т.к. в итоге сеть сойдётся к сети хопфилда, то она гарантированно найдёт локальный минимум, но при введении начальной хаотической динамики сеть будет пытаться выйти из локального минимума.

Моделирование:

Динамика одного нейрона:

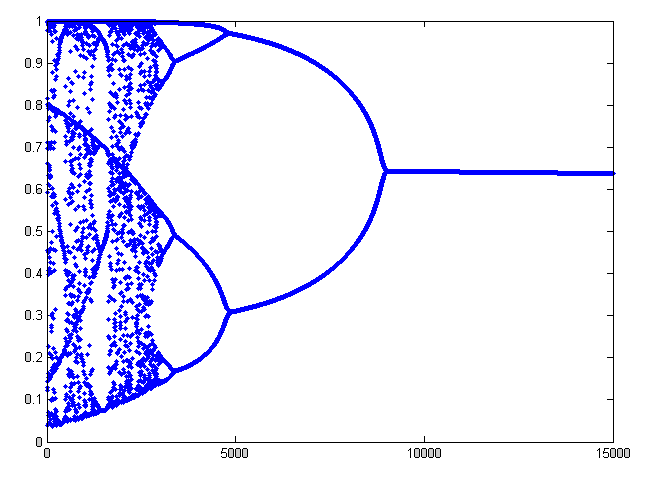
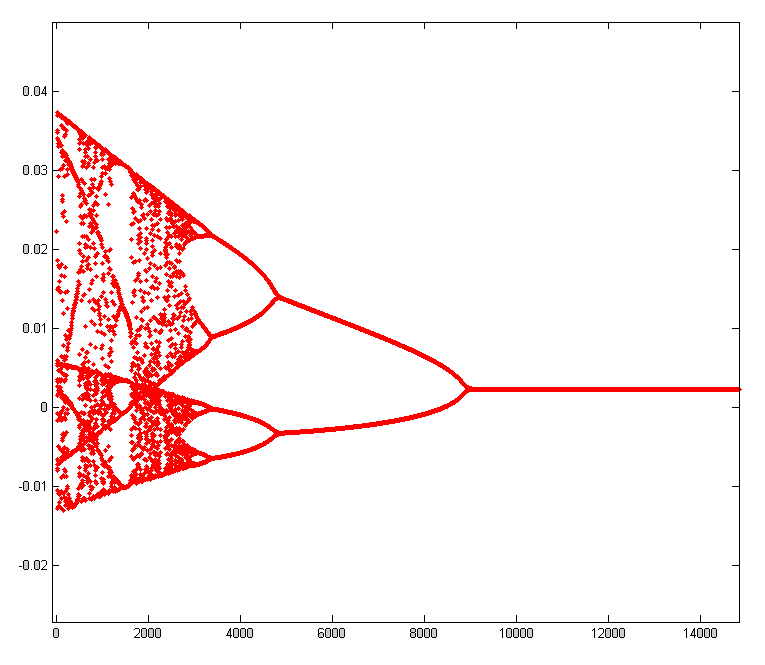


Рис.1. динамики одного нейрона

Тестовый пример поиска глобального минимума:

Sum((x(i))^2 - cos(3\*PI\*x(i)))

Rastrigin function: 10n + Sum(x(i)^2 - 10cos(2pi\*x(i)))

Такое представление модели сильно ограничено по диапазону входной функции, т.к. при довольно сильно изменяющейся функции необходимо сильно уменьшить коэффициент альфа при производной, что повлияет на схождение к глобальному оптимуму, а если альфа будет слишком велик, то динамика сети будет не хаотическая а пойдёт в разнос и на выходе будут осцилляции между максимальным и минимальным значениями.

-- Т.к.диапазон хаотической последовательности, которая описывается логистической картой будет между 0 и 1.

При таких ограничения, если даже перейти к аналоговой модели, как написано в статье, то всё рано логистическая карта в дифференциальном уравнении уже не будет хоатической функцией. Таким образом возьмём хаотическую функцию, которая также будет поддерживать аппаратную платформу на аналоговых микросхемах.

Аналоговое представление сети:

В статье TCNN есть описание сети в аналоговом виде, но представленная ими хаотическая функция не будет хаотической если перейти в аналоговый спектр. Поэтому для реализации аналоговой модели TCNN необходимо описать и аналоговую хаотическую функцию.

Для аппаратной поддержки возьмём функцию, которая может быть легко представлена с помощью аналоговых компонент.

Хаотическая функция:

Описание, фазовый портрет

Аналоговая математическая модель для TCNN:

Моделирование:

Тестовый пример поиска глобального минимума:

Начальные значения для нейронов 5 и -2.

Minimum f(0,0) = 0

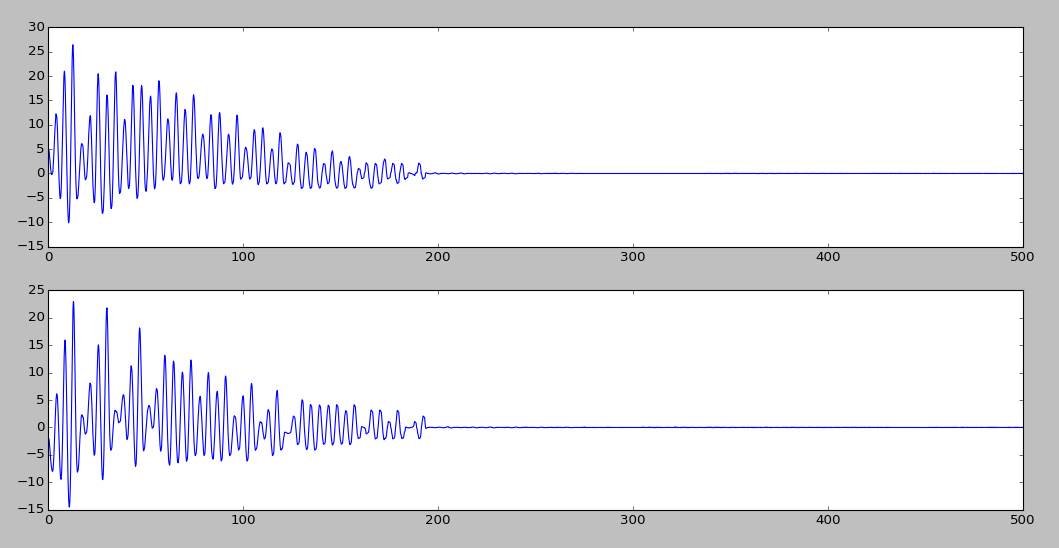


Рис.8. Динамика выходных значений нейронов

Найденные значения:

-0.0059729683838

-0.00508304486271

Сравнение с различными алгоритмами поиска глобального минимума:

- GA

- Ant Colony

- Simulating Annealing

- Monte Carlo

Пример использования аналоговой модели: ???

Выводы

В данной статье было проведено сравнение различных методов поиска глобального оптимума, среди которых TCNN отличается как метод, который может быть реализован в аналоговом представлении.

Summary table

|  |  |  |  |
| --- | --- | --- | --- |
| Algo | Global min | Average err | Max err (|f(Xopt) – f(Xfind)|) |
| Function: f(x) = Sum((x(i))^2 - cos(3\*PI\*x(i))) | | | |
| TCNN |  |  |  |
| aTCNN | 98 | ~10^-5 | 0.4346 |
|  |  |  |  |
|  |  |  |  |

---------------- TOTAL RESULT -------------------

result: 100/50

fails: 76

---------------- TOTAL RESULT -------------------

result: 100/0

fails: 0

subprocess.check\_call([WIN\_BIN\_OPT\_FUNC

, "--steps", "8000"

, "--step\_len", "0.001"

, "--chaotic\_step\_len", "0.15"

, "--alpha", "0.05"

, "--chaotic\_coeff", "20000" #800

, "--chaotic\_reduce", "0.99993"

, "--function", "5"

, "--dim", "30"

, "--init\_cond", "0, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8"] )

---------------- TOTAL RESULT -------------------

result: 10/0

fails: 0

subprocess.check\_call([WIN\_BIN\_OPT\_FUNC

, "--steps", "15000"

, "--step\_len", "0.001"

, "--chaotic\_step\_len", "0.15"

, "--alpha", "0.05"

, "--chaotic\_coeff", "20000" #800

, "--chaotic\_reduce", "0.5"

, "--function", "5"

, "--dim", "30"

, "--init\_cond", "0, 2.8, 3.8, 1.8, 2.8, 3.8, -1.2, -3.0, -0.8, 1.9, 1.8, 2.8, 3.8, 1.8, 2.8, 3.8, -1.2, -3.0, -0.8, 1.9, 1.8, 2.8, 3.8, 1.8, 2.8, 3.8, -1.2, -3.0, -0.8, 1.9, -1.9"])

---------------- TOTAL RESULT -------------------

result: 100/0

fails: 0

subprocess.check\_call([WIN\_BIN\_OPT\_FUNC

, "--steps", "15000"

, "--step\_len", "0.001"

, "--chaotic\_step\_len", "0.15"

, "--alpha", "0.05"

, "--chaotic\_coeff", "20000" #800

, "--chaotic\_reduce", "0.5"

, "--function", "5"

, "--dim", str(dim)

, "--init\_cond", init\_cond])

RANDOM initial conditions [-5, 5]

>>>>> TESTING INFO <<<<<<

tests complete: 100

result: 100/2

fails: 2

>>>>> TESTING INFO <<<<<<

---------------- TOTAL RESULT -------------------

result: 100/2

fails: 2

subprocess.check\_call([WIN\_BIN\_OPT\_FUNC

, "--steps", "30000"

, "--step\_len", "0.001"

, "--chaotic\_step\_len", "0.15"

, "--alpha", "0.2"

, "--chaotic\_coeff", '4000' #"1500" #800

, "--chaotic\_reduce", "0.4"

, "--function", "6"

, "--dim", str(dim)

, "--init\_cond", init\_cond])

RANDOM initial conditions [-5, 5]