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

Поиск глобального минимума с помощью аналоговой модели Trancident transiently chaotic neural network (TCNN)

1. Введение

В данной статье рассматривается аналоговая модель TCNN. С помощью которой, будет показано, что такая модель более подходит для решения задачи поиска глобального минимума, в отличие от дискретного представления. Также приводится сравнение с методом, на котором основана модель (метод отжига).

1. Описание TCNN

Изначально TCNN была предложена для решения задач комбинаторной оптимизации. За счёт хаотической динамики привнесённой в сеть появилась возможность расширить область поиска оптимального решения. В [] описывается модель и её применение к задачи TSP. В данной статье произведена попытка применения TCNN к задачам поиска оптимума аналоговой функции.

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

Данная модель похожа на сеть Хопфилда, отличие заключается в том, что в начальный момент, в сети происходит хаотический поиск, который обуславливается хаотической функцией аналог 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.

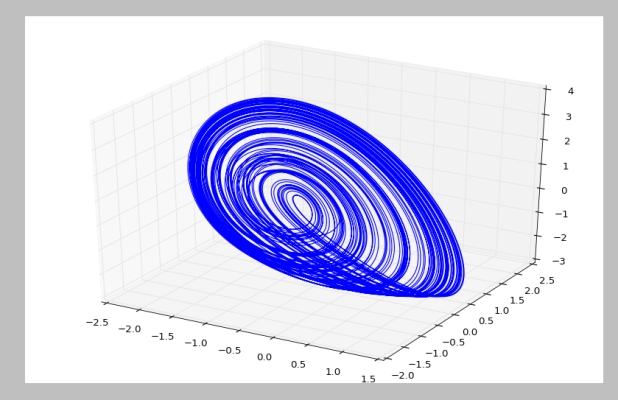
1. Описание аналоговой модели TCNN

Если переходить к аналоговому описанию, представленному в [], то хаотическая динамика, моделируемая уравнением:

в аналоговом представлении не имеет хаотического поведения. Следовательно, для аналоговой модели необходима хаотическая другая функция. Возьмём одну из функций представленных в [], который могут быть легко выполнены на аналоговых элементах.

Математическое описание хаотической функции:

Фазовый портрет:



В роли хаотической функции для TCNN возьмём третью производную из данной модели, тогда математическое описание аналоговой TCNN (aTCNN) для решения задач оптимизации аналоговой функции будет выглядеть следующим образом:

где

xi – хаотическая функция

yi – выход нейрона

zi – коэффициент влияния хаотической динамики

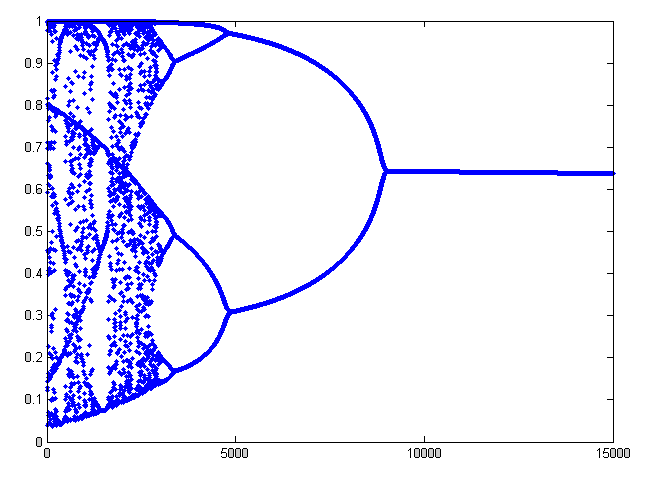
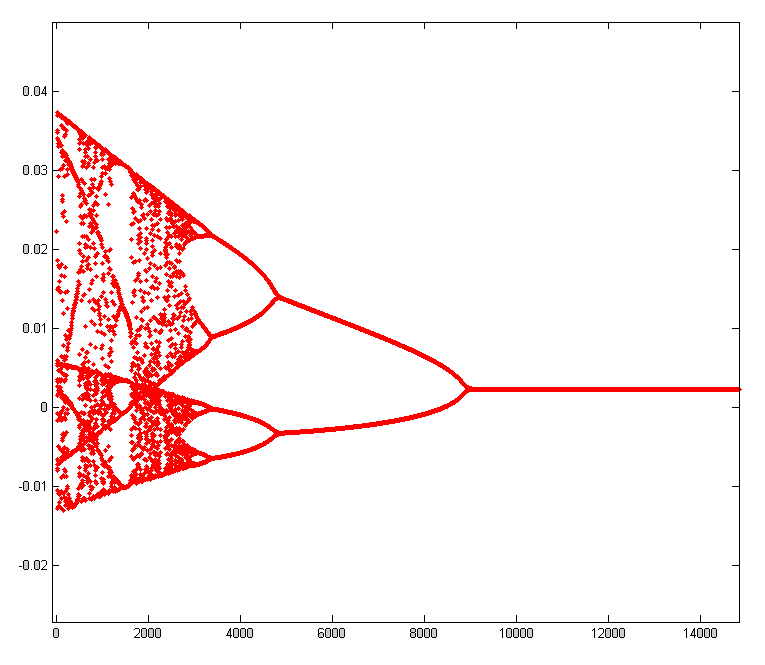
α – коэффициент сходимости

f – оптимизируемая функция

θ - коэффициент затухания хаотической динамики

1. Моделирование:

Ниже представлена динамика одного нейрона для TCNN внутреннего состояния(справа) и выхода (слева).



Как видим, хаотическая динамика присутствует на начальных этапах поиска, что позволяет сети иметь возможность выйти из локального оптимума. Но, как оказалось, на практике такой метод плохо подходит для поиска глобального минимума при большом количестве локальных оптимумов и большом диапазоне изменения значений оптимизируемой функции. Т.к. коэффициент alpha влияет как на скорость сходимости функции, так и на саму сходимость. Если этот коэффициент будет слишком маленьким, то оптимизируемая функция почти не будет влиять на динамику самой сети, но если этот коэффициент будет слишком большой, то динамика сети никогда не сойдётся, и будет колебаться между максимальным и минимальным значениями. Т.к. сама сеть изначально и не была предусмотрена для решения задача оптимизации аналоговых функций, поэтому ошибка при оптимизации будет значительная, что видно в Таблице 1.

Для теста aTCNN возьмём функцию с несколькими оптимумами следующего вида:

Параметры сети:

При моделировании для решения дифференциальных уравнений используется метод Рунге-Кутты 4й степени. Для поиска производной используется численное дифференцирование пятого порядка.

шаг моделирования сети = 0.01

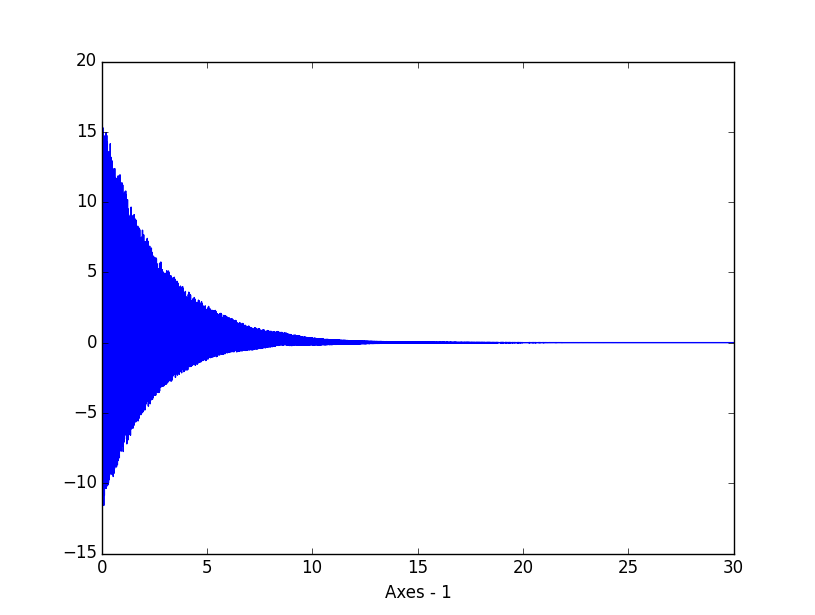
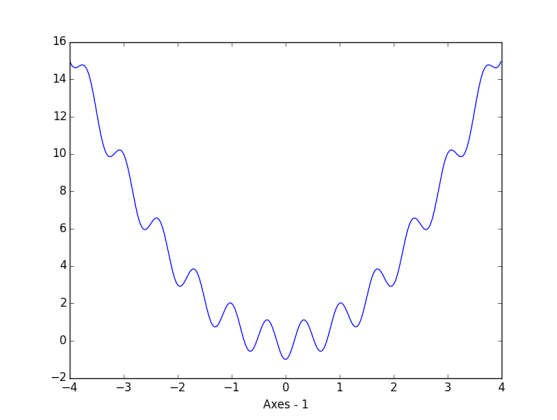
шаг моделирования хаотической функции = 0.15

z0 – 4000

α – 0.2

θ – 0.4

Количество шагов моделирование = 30000



Найденные значения соответствуют глобальному минимуму

X = '-4.42152e-005' F = -1

Проверим работу aTCNN на некоторых функциях с множеством локальных оптимумов и сравним с TCNN и Simulation Annealing алгоритмами. Также сравним эти алгоритмы с генетическим алгоритмом использующим размер популяции 300 и 200 поколений, сравнение которого также было поведено в [].

Для сравнения было выбрано 3 функции (N-размерность пространс):

1. Функция с несколькими локальными оптимумами, немного проще чем функция Растригина, была взята для того, чтобы показать, что TCNN не подходит для решения задач оптимизации аналоговых функций, притом aTCNN отлично справляется с этой задачей.
2. Функция Растригина:
3. Функция Schwefel

Параметры для моделирования aTCNN:

1. Те же, что и в примере

z0 – 150000

α – 0.2

θ – 0.4

Количество шагов моделирование = 30000

z0 – 500000

α – 0.6

θ – 0.2

Количество шагов моделирование = 100000

|  |  |  |  |
| --- | --- | --- | --- |
| Algo | Find Global min in 100 runs | Average err | Max err (|f(Xopt) – f(Xfind)|) |
| Function: f(x) = Sum((x(i))^2 - cos(3\*PI\*x(i))) | | | |
| TCNN | 0 | 350.377594289 | 350.377594289 |
| aTCNN | 98 | ~10^-5 | 0.4346 |
| Annealing |  |  |  |
| GA |  |  |  |
| Rastrigin | | | |
| TCNN |  |  |  |
| aTCNN | 68 | 0.4436 | 3.00309742856 |
| Annealing | 1 | 274.8335 | 414.9209 |
| GA |  |  |  |
|  |  |  |  |
| Schwefel function: f(x) = Sum(-x(i)\*sin(sqrt(abs(x(i))))) | | | |
| TCNN |  |  |  |
| aTCNN | 100 | 0.00038208 | 0.00038213 |
| Annealing |  |  |  |
| GA |  |  |  |

Выводы

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.

Данная сеть плохо применима к задачам отпимизации непрерывных функций. Но подходит к оптимизации комбинаторных задач, таких как к примеру TSP.

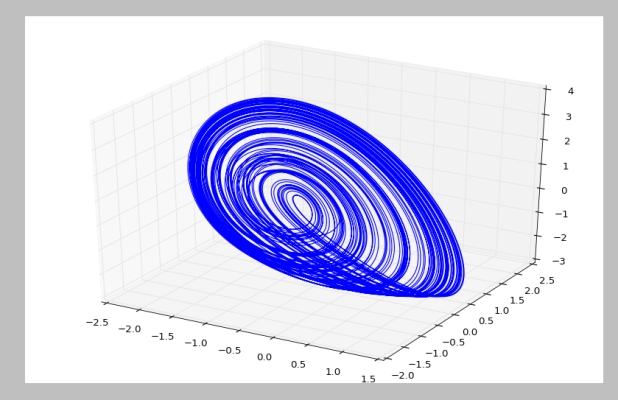
При таких ограничения, если даже перейти к аналоговой модели, как написано в статье, то всё рано логистическая карта в дифференциальном уравнении уже не будет хоатической функцией. Таким образом возьмём хаотическую функцию, которая также будет поддерживать аппаратную платформу на аналоговых микросхемах.

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:

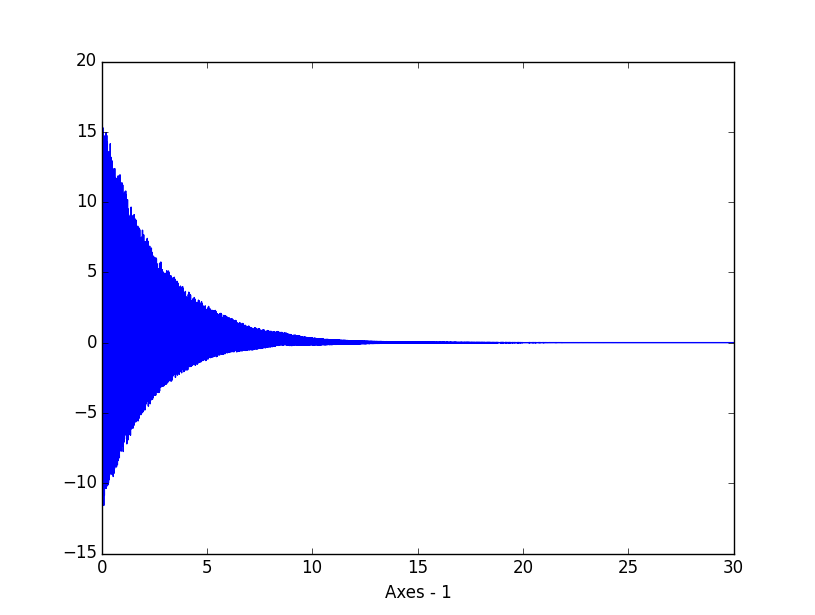
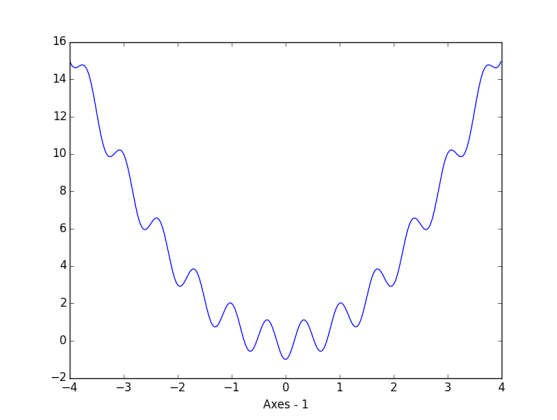
Описание, фазовый портрет



Аналоговая математическая модель для TCNN:

Моделирование:

Тестовый пример поиска глобального минимума:



X = '-4.42152e-005' F = -1

Параметры сети:

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])

init\_cond += ', ' + str((random.random()-0.5)\*10)

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.

Конфигурация тестовых параметров

Сравнение с различными алгоритмами поиска глобального минимума:

- GA

- Ant Colony

- Simulating Annealing

- Monte Carlo

Summary table

---------------- 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]