БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра ЭВМ

Лабораторная работа №2

Нейронная сеть Хопфилда

|  |  |
| --- | --- |
| Проверил:  Третьяков А. Г. | Выполнил: ст. гр. 250501 Андрадэ А. И. |

Минск 2016

1. **Цель работы**

Изучение топологии, алгоритма функционирования сети Хопфилда.

Имеется некоторое количество эталонных образов – изображений (бинарная матрица). Нам дают некий искажённый образ, и наша задача состоит в том, чтобы «распознать» в нём один из эталонных.

1. **О сети Хопфилда**

Каждый нейрон сети получает и передаёт сигналы другим. То, как нейроны связаны между собой, зависит от типа сети. Сеть Хопфилда является однослойной сетью, потому что в ней используется лишь один слой нейронов. Она так же является рекурсивной сетью, потому что обладает обратными связями. Она функционирует циклически.

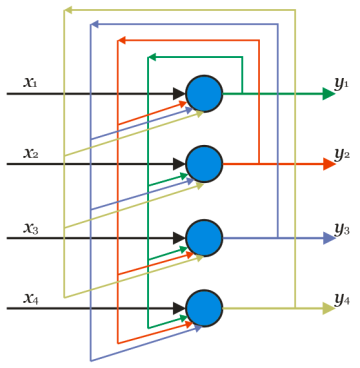


Рисунок 1. Пример сети Хопфилда из четырёх нейронов

Каждый из нейронов имеет выходы сигнала, который подаются на входы всех остальных нейронов, кроме себя самого.

Сеть, содержащая N нейронов может запомнить не более ~0.15\*N образов.Так что реальная сеть должна содержать достаточно внушительное количество нейронов. Это одно из существенных недостатков сети Хопфилда – небольшая ёмкость.

Плюс ко всему образы не должны быть очень похожи друг на друга, иначе в некоторых случаях возможно зацикливание при распознавании.

1. **Работа сети**

Образ, который сеть запоминает или распознаёт (любой входной образ) может быть представлен в виде вектора X размерностью n, где n – число нейронов в сети. Выходной образ представляется вектором Y с такой же размерностью. Каждый элемент вектора может принимать значения: +1 либо -1 (Можно свести к 0 и 1, однако +1 и -1 удобнее для расчётов).

Программная реализации не требует непосредственной реализации нейронов, можно всего лишь эмулировать их работу при помощи векторов и матрицы коэффициентов

1. **Обучение сети**

Как было сказано, обучение сети строится на вычислении весовых коэффициентов. Для этого мы будем поддерживать матрицу W размером (N, N). N – количество нейронов в векторе (как входном, так и выходном). При обучении сети некому образу X коэффициенты устанавливаются так:

def train(self,X):

for i in range(self.neurons\_num):

for j in range(i):

if i == j:

W[i][j] = 0

else:

W[i][j] += X[i] \* X[j]

W[j][i] = W[i][j]

Где, **X** – входной бинарный образ для обучения.

**W** –весовая весовых коэффициентов связей между нейронами.

Матрица весовых коэффициентов W – симметричная, поэтому рассчитываем значения коэффициентов только до главной диагонали и копируем их в симметричные диагонали элементы.

Если нам нужно обучить сеть следующему образу, мы просто меняем вектор X и заново повторяем эту процедуру. Вы видите, в элементах матрице сохраняется сумма значений для всех образов, которым мы обучили сеть. Установка значения элемента в 0 при i==j это отражения устройства сети, когда выход некого нейрона не попадает на его же вход.

1. **Распознавание образа**

После того как сеть обучена нескольким эталонным образам мы захотим подать ей на вход некоторый вектор, и попросить её распознать его. Нас могут ожидать несколько исходов.

В идеале сеть распознает образ и выдаст на выход эталонный вектор, соответствующий искажённому.

Другой вариант развития событий – если в памяти сети есть похожие образы, и входящий искажённый похож на их обоих, то сеть может впасть в бесконечный цикл. Так, если при некотором заданном количестве итераций распознавания образ не распознан, цикл прекращается, распознавание признаётся неудачным и сеть выводит вольную «импровизацию» своей работы.

Сеть выполняет следующую работу пока результат не совпадёт с одним из эталонным образов, либо не привесится порог итераций. Случайным образом выбирается нейрон r для обновления. Для него рассчитывается новое состояние s (асинхронный режим работы), используя нашу матрицу коэффициентов следующим образом:

net = 0

for i in range(neurons\_num):

net += Y[i] \* W[i][r]

s = signum(net,0)

Затем обновляем его состояние:

Y[r] = s

*signum(a,b)* – пороговая активационная функция, её реализация:

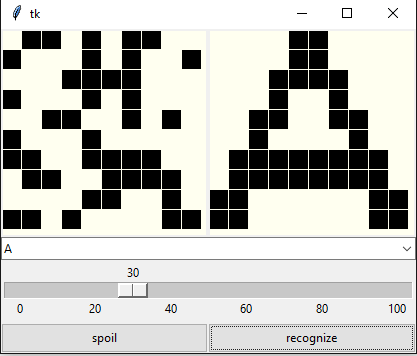
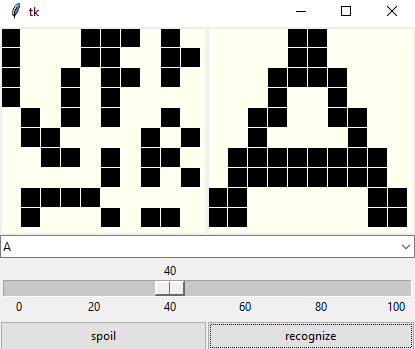
def signum(a,b):

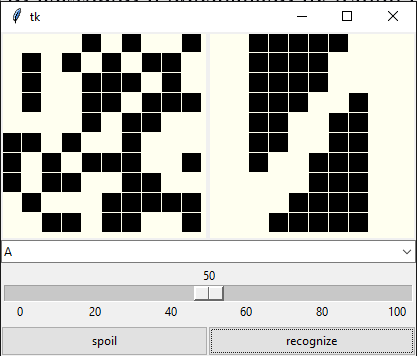
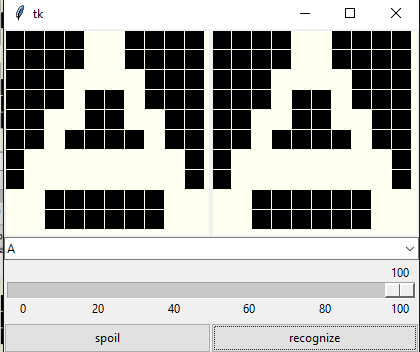
return int(a > b) - int(a < b)

В нашем случае мы не проводим чёткой разницы между входным и выходным вектором и обозначаем их одной переменной, поскольку выходной вектор рекурсивно подаётся на вход сети снова.

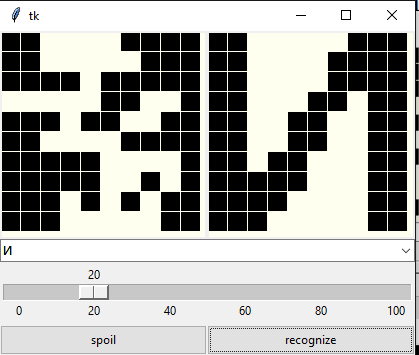
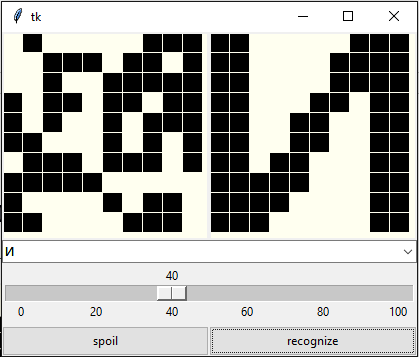
1. **Результаты**

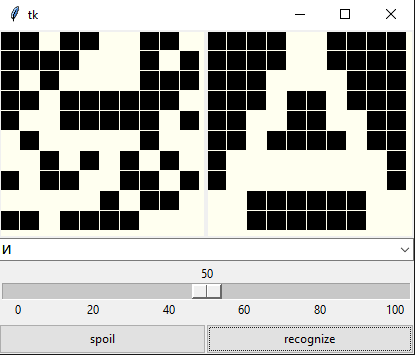
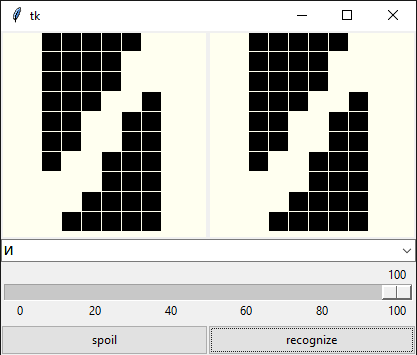
Буква А (бинарная матрица (10,10)  ):

Буква И:

1. **Итог**

Была продемонстрирована простая реализация нейронной сети Хопфилда, формирующую ассоциативную память. Как было показано, вся работа сводится к вычислению весовых коэффициентов сети.