

0 Установка и импорт библиотеки

1. Для установки библиотеки необходимо в терминале ввести следующую команду:

```
pip install widelearning
```

Дальнейшие действия выполнены в Jupyter Notebook.

2. Импорт библиотеки:

```
import widelearning as wdl
```

1 Полносвязный слой

1. Подготовка набора данных (датасета).

Для примера использован набор данных User Knowledge Modeling Data Set, который можно скачать по адресу:

<https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling>

Открытый в MS Excel файл выглядит следующим образом:

STG	SCG	STR	LPR	PEG	UNS	Attribute Information:	
0	0	0	0	0	very_low		
0.08	0.08	0.1	0.24	0.9	High	STG (The degree of study time for goal object materials).	
0.06	0.06	0.05	0.25	0.33	Low	SCG (The degree of repetition number of user for goal object materials)	
0.1	0.1	0.15	0.65	0.3	Middle	STR (The degree of study time of user for related objects with goal object)	
0.08	0.08	0.08	0.98	0.24	Low	LPR (The exam performance of user for related objects with goal object)	
0.09	0.15	0.4	0.1	0.66	Middle	PEG (The exam performance of user for goal objects)	
0.1	0.1	0.43	0.29	0.56	Middle	UNS (The knowledge level of user)	
0.15	0.02	0.34	0.4	0.01	very_low		
0.2	0.14	0.35	0.72	0.25	Low		
0	0	0.5	0.2	0.85	High		
0.18	0.18	0.55	0.3	0.81	High		
0.06	0.06	0.51	0.41	0.3	Low		
0.1	0.1	0.52	0.78	0.34	Middle		
0.1	0.1	0.7	0.15	0.9	High		
0.2	0.2	0.7	0.3	0.6	Middle		
0.12	0.12	0.75	0.35	0.8	High		
0.05	0.07	0.7	0.01	0.05	very_low		
0.1	0.25	0.1	0.08	0.33	Low		
0.15	0.32	0.05	0.27	0.29	Low		

Скопируем часть, содержащую данные, в новый файл формата csv и переместим столбец UNS (состоит из меток принадлежности к классам) в

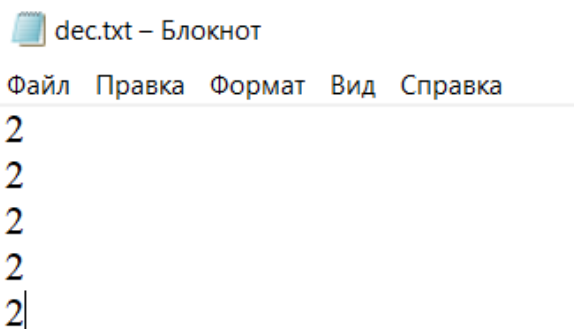
начало. Заменяем разделитель целой и дробной части вместо запятой «.» на точку «.»

Получившийся файл, открытый в текстовом редакторе:

```
UNS;STG;SCG;STR;LPR;PEG
very_low;0;0;0;0;0
High;0.08;0.08;0.1;0.24;0.9
Low;0.06;0.06;0.05;0.25;0.33
Middle;0.1;0.1;0.15;0.65;0.3
Low;0.08;0.08;0.08;0.98;0.24
Middle;0.09;0.15;0.4;0.1;0.66
Middle;0.1;0.1;0.43;0.29;0.56
very_low;0.15;0.02;0.34;0.4;0.01
Low;0.2;0.14;0.35;0.72;0.25
High;0;0;0.5;0.2;0.85
High;0.18;0.18;0.55;0.3;0.81
Low;0.06;0.06;0.51;0.41;0.3
Middle;0.1;0.1;0.52;0.78;0.34
High;0.1;0.1;0.7;0.15;0.9
Middle;0.2;0.2;0.7;0.3;0.6
High;0.12;0.12;0.75;0.35;0.8
very_low;0.05;0.07;0.7;0.01;0.05
Low;0.1;0.25;0.1;0.08;0.33
Low;0.15;0.32;0.05;0.27;0.29
```

2. Преобразование в целочисленный вид

Для выполнения вычислений преобразуем с помощью функции `data_int` представленные в вещественнозначном виде данные в целочисленный. Создадим файл `dec.txt`, который содержит количество знаков после запятой для каждого столбца файла `csv`. Значения количества знаков в файле `txt` идут на новой строке для каждого следующего столбца, то есть в нашем случае имеется 5 столбцов, для каждого столбца количество знаков равно 2. В итоге в файле `dec.txt` 5 строк со значением 2:




```
dec.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
2
2
2
2
2|
```


Первый параметр функции – путь к подготовленной выборке в формате csv, второй параметр – файл со значениями количества знаков после запятой для каждого столбца, третий параметр – желаемое название выходных преобразованных в целочисленный вид файлов.

```
wdl.data_int('kahraman.csv', 'dec.txt', 'KAHRAMAN')
```

	UNS	STG	SCG	STR	LPR	PEG	N
0	very_low	-49	-45	-47	-49	-46	0
1	High	-41	-37	-37	-25	44	1
2	Low	-43	-39	-42	-24	-13	2
3	Middle	-39	-35	-32	16	-16	3
4	Low	-41	-37	-39	49	-22	4
...
253	High	12	33	22	43	11	253
254	Middle	29	16	24	-30	14	254
255	High	5	37	24	-20	31	255
256	Middle	1	30	34	12	-20	256
257	High	17	45	29	38	28	257

В результате применения функции data_int получим целочисленные обучающую и тестовую выборки.

 KAHRAMAN_test.csv

 KAHRAMAN_train.csv

3. Первоначальное приближение первого нейрона

Используем функцию select_top для получения первоначального приближения первого нейрона:

```
wdl.select_top('KAHRAMAN_train.csv', 'UNS')
```

Здесь первый параметр – путь к обучающей выборке в целочисленном виде, второй параметр – название столбца с метками классов.

Вывод функции выглядит следующим образом:

```
Количество отсеченных сверху = 9
Количество НЕотсеченных сверху = 14
Количество отсеченных снизу = 8
Количество НЕотсеченных снизу = 201
===Нижняя категория: High
СУММА отсеченных сверху и снизу = 17
TOP - ['very_low']
OTHERS - ['High', 'Low', 'Middle']
+++++
Количество отсеченных сверху = 10
Количество НЕотсеченных сверху = 44
Количество отсеченных снизу = 15
Количество НЕотсеченных снизу = 163
===Нижняя категория: very_low
СУММА отсеченных сверху и снизу = 25
TOP - ['High']
OTHERS - ['very_low', 'Low', 'Middle']
+++++
Количество отсеченных сверху = 4
Количество НЕотсеченных сверху = 71
Количество отсеченных снизу = 7
Количество НЕотсеченных снизу = 150
===Нижняя категория: High
СУММА отсеченных сверху и снизу = 11
TOP - ['Low']
OTHERS - ['very_low', 'High', 'Middle']
+++++
Количество отсеченных сверху = 0
Количество НЕотсеченных сверху = 80
Количество отсеченных снизу = 7
Количество НЕотсеченных снизу = 145
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 7
TOP - ['Middle']
OTHERS - ['very_low', 'High', 'Low']
+++++
```

Переменная TOP обозначает целевой класс и соответственно, «Нижняя категория» - это противоположный класс, участвующий в отсечении.

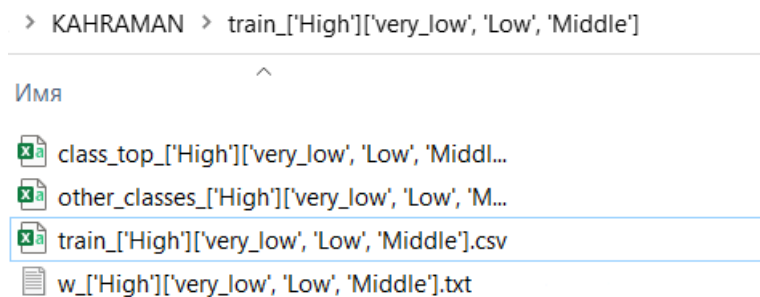
Были также сгенерированы папки:

```
train_['High']['very_low', 'Low', 'Middle']
train_['Low']['very_low', 'High', 'Middle']
train_['Middle']['very_low', 'High', 'Low']
train_['very_low']['High', 'Low', 'Middle']
```

Необходимо выбрать целевой класс. Выбор можно производить по разным критериям:

- по сумме отсеченных сверху и снизу;
- по количеству отсеченных сверху;
- по количеству отсеченных снизу и так далее.

От выбора целевого класса зависит итоговая структура нейронной сети. В нашем случае целевой класс будет выбираться по сумме отсеченных экземпляров сверху и снизу. Таким образом, в качестве целевого на первом нейроне выступает класс «High», поскольку сумма отсеченных равна 25. Если мы хотим оставить полученные результаты без проведения градиентного уточнения вектора весов, то необходимо открыть соответствующую папку и скопировать в отдельный каталог файлы w.txt и train.csv, которые являются вектором весов первого нейрона и сокращенной выборкой для второго нейрона:



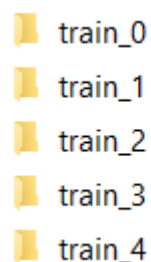
4. Градиентное уточнение первого нейрона

С помощью градиентного уточнения можно увеличить количество отсеченных экземпляров. Функция grad изменяет элементы вектора весов после первоначального приближения с некоторым шагом и фиксирует количество отсеченных. В приоритете стоит количество отсеченных сверху (требуется увеличение количества), далее количество отсеченных снизу (требуется увеличение количества), на последнем месте расстояние между границами отсечения. Если количество отсеченных сверху и снизу не изменяется, то элемент вектора изменяется, пока расстояние между границами уменьшается.

Первый параметр функции `grad` – путь к обучающей выборке, которая использовалась в первоначальном приближении, второй – целевой класс, третий – список с остальными классами, четвертый – название столбца с метками классов, пятый – вектор весов после первоначального приближения. Вектор весов берется из соответствующей целевому классу папки первоначального приближения (файл `w.txt`).

```
wdl.grad('KAHRAMAN_train.csv', ['High'], ['very_low', 'Low', 'Middle'], 'UNS',  
[455.02247191011236, 552.8764044943821, 273.38202247191015,  
518.2921348314607, 2276.179775280899])
```

В результате градиентного уточнения сгенерировались папки `train_{i}.csv`.







Необходимо проверить каждую из них и выбрать ту папку, в которой в файле `txt` во второй строке (количество отсеченных сверху) наибольшее значение. Если значения одинаковые, то необходимо проверить третью строку (количество отсеченных снизу) и выбрать ту папку, в которой данное значение наибольшее. В случае одинаковых значений сверху и снизу, необходимо выбирать папку по последней строке файла `txt`, которое обозначает расстояние между границами, оно должно быть минимальное.

Таким образом, в нашем случае все три значения для всех папок одинаковые, поэтому выбирается любая. Выберем папку под номером 0. Необходимо скопировать отдельно файл `train.csv` и `w.txt`. Первый файл является сокращенной обучающей выборкой для второго нейрона, второй файл – является вектором весов первого нейрона.

> KAHRAMAN > train_0

Имя

 class_top_0.csv
 other_classes_0.csv
 train_0.csv
 w_0.txt

После градиентного уточнения количество отсеченных экземпляров увеличилось: сверху до 49, снизу до 10.

5. Первоначальное приближение второго нейрона

Выбирается сокращенная выборка после градиентного уточнения.

```
wdl.select_top('train_2.csv', 'UNS')
```

```
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 4
Количество отсеченных снизу = 3
Количество НЕотсеченных снизу = 165
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 4
TOP - ['High']
OTHERS - ['Middle', 'Low', 'very_low']
+++++
Количество отсеченных сверху = 5
Количество НЕотсеченных сверху = 75
Количество отсеченных снизу = 6
Количество НЕотсеченных снизу = 87
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 11
TOP - ['Middle']
OTHERS - ['High', 'Low', 'very_low']
+++++
Количество отсеченных сверху = 7
Количество НЕотсеченных сверху = 68
Количество отсеченных снизу = 4
Количество НЕотсеченных снизу = 94
===Нижняя категория: Middle
СУММА отсеченных сверху и снизу = 11
TOP - ['Low']
OTHERS - ['High', 'Middle', 'very_low']
+++++
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 12
Количество отсеченных снизу = 1
Количество НЕотсеченных снизу = 159
===Нижняя категория: Middle
СУММА отсеченных сверху и снизу = 2
```

```
TOP - ['very_low']
OTHERS - ['High', 'Middle', 'Low']
+++++
```

Имеется два класса с количеством отсеченных экземпляров сверху и снизу, равным 11. Выберем в качестве целевого класс «Low».

6. Градиентное уточнение второго нейрона

Из соответствующей папки, в названии которой в первом списке «Low», копируем вектор весов.

```
wdl.grad('train_2.csv', ['Low'], ['High', 'Middle', 'very_low'], 'UNS', [-
540.3163265306123, -194.17346938775518, -400.48979591836735,
1046.591836734694, -1941.0408163265306])
```

Выбираем из папок train номер папки, в которой в файле txt наибольшее количество отсеченных сверху, иначе наибольшее количество отсеченных снизу, иначе наименьшее расстояние между границами отсечения. Во всех папках одинаковые значения, выберем папку 1. Количество отсеченных сверху стало 10, снизу 7. Скопируем в отдельный каталог файл train.csv и w.txt (сокращенная обучающая выборка третьего нейрона, вектор весов второго нейрона). Удалим сгенерированные папки.

7. Первоначальное приближение третьего нейрона

Загрузим соответствующую сокращенную выборку.

```
wdl.select_top('train_3.csv', 'UNS')
```

```
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 64
Количество отсеченных снизу = 1
Количество НЕотсеченных снизу = 90
===Нижняя категория: High
СУММА отсеченных сверху и снизу = 2
TOP - ['Low']
OTHERS - ['very_low', 'Middle', 'High']
+++++
Количество отсеченных сверху = 4
Количество НЕотсеченных сверху = 9
Количество отсеченных снизу = 1
```



```

Количество НЕотсеченных снизу = 142
===Нижняя категория: Middle
СУММА отсеченных сверху и снизу = 5
TOP - ['very_low']
OTHERS - ['Low', 'Middle', 'High']
+++++
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 72
Количество отсеченных снизу = 2
Количество НЕотсеченных снизу = 81
===Нижняя категория: very_low
СУММА отсеченных сверху и снизу = 3
TOP - ['Middle']
OTHERS - ['Low', 'very_low', 'High']
+++++
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 4
Количество отсеченных снизу = 2
Количество НЕотсеченных снизу = 149
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 3
TOP - ['High']
OTHERS - ['Low', 'very_low', 'Middle']
+++++

```

Наибольшая сумма отсеченных равна 5 для целевого класса «very_low».

8. Градиентное уточнение третьего нейрона

```

wdl.grad('train_3.csv', ['very_low'], ['Low', 'Middle', 'High'], 'UNS',
[47.81818181818181, -182.63636363636363, -141.54545454545456,
91.72727272727275, -387.27272727272725])

```

В папке под номером 0 расстояние минимальное и равно 7.34. Из данной папки копируются в отдельный каталог файлы train.csv и w.txt (обучающая выборка для четвертого нейрона и вектор весов третьего нейрона). Сгенерированные папки необходимо удалить.

9. Первоначальное приближение четвертого нейрона

В качестве целевого выбирается класс «High», поскольку сумма отсеченных наибольшая и равна 4.

```

wdl.select_top('train_4.csv', 'UNS')

```

```

Количество отсеченных сверху = 0
Количество НЕотсеченных сверху = 7
Количество отсеченных снизу = 1
Количество НЕотсеченных снизу = 141
===Нижняя категория: High
СУММА отсеченных сверху и снизу = 1
TOP - ['very_low']
OTHERS - ['Low', 'Middle', 'High']
+++++
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 64
Количество отсеченных снизу = 2
Количество НЕотсеченных снизу = 82
===Нижняя категория: High
СУММА отсеченных сверху и снизу = 3
TOP - ['Low']
OTHERS - ['very_low', 'Middle', 'High']
+++++
Количество отсеченных сверху = 0
Количество НЕотсеченных сверху = 72
Количество отсеченных снизу = 1
Количество НЕотсеченных снизу = 76
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 1
TOP - ['Middle']
OTHERS - ['very_low', 'Low', 'High']
+++++
Количество отсеченных сверху = 2
Количество НЕотсеченных сверху = 3
Количество отсеченных снизу = 2
Количество НЕотсеченных снизу = 142
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 4
TOP - ['High']
OTHERS - ['very_low', 'Low', 'Middle']
+++++

```

10. Градиентное уточнение четвертого нейрона

```

wdl.grad('train_4.csv', ['High'], ['very_low', 'Low', 'Middle'], 'UNS', [-63.5625,
75.15972222222223, -44.30555555555556, -83.38888888888889,
196.91666666666666])

```

Вектор весов и сокращенная обучающая выборка копируются из папки №3. Количество отсеченных сверху 5, снизу – 2, расстояние – 751.9. Класс «High» полностью отсекается.

Удаляются сгенерированные папки.

11. Первоначальное приближение пятого нейрона

```
wdl.select_top('train_5.csv', 'UNS')
```

```
Количество отсеченных сверху = 52
Количество НЕотсеченных сверху = 20
Количество отсеченных снизу = 1
Количество НЕотсеченных снизу = 69
===Нижняя категория: Low
СУММА отсеченных сверху и снизу = 53
TOP - ['Middle']
OTHERS - ['Low', 'very_low']
+++++
Количество отсеченных сверху = 1
Количество НЕотсеченных сверху = 62
Количество отсеченных снизу = 60
Количество НЕотсеченных снизу = 19
===Нижняя категория: Middle
СУММА отсеченных сверху и снизу = 61
TOP - ['Low']
OTHERS - ['Middle', 'very_low']
+++++
Количество отсеченных сверху = 0
Количество НЕотсеченных сверху = 7
Количество отсеченных снизу = 54
Количество НЕотсеченных снизу = 81
===Нижняя категория: Middle
СУММА отсеченных сверху и снизу = 54
TOP - ['very_low']
OTHERS - ['Middle', 'Low']
+++++
```

Целевым выбирается класс «Middle», поскольку несмотря на сумму отсеченных сверху отсекается только один экземпляр. Предпочтительнее выбрать вариант, в котором сверху отсекается большее количество экземпляров.

12. Градиентное уточнение пятого нейрона

```
wdl.grad('train_5.csv', ['Middle'], ['Low', 'very_low'], 'UNS',
[277.4000000000001, 201.71428571428555, 150.99999999999994, -
709.5714285714286, 2071.085714285714])
```

В результате класс «very_low» полностью отсекается.

Выберем папку под номером 0. Скопируем в отдельный каталог выборку для шестого нейрона train.csv и вектор весов пятого нейрона w.txt. Удалим сгенерированные папки.

13. Первоначальное приближение шестого нейрона

В обучающей выборке осталось только два класса. Количество экземпляров выборки равно 64. Необходимо использовать бинарные варианты функций с меткой «binary».

```
wdl.select_top_binary('train_6.csv', 'UNS')
```

```
Количество отсеченных сверху = 18
Количество отсеченных снизу = 45
===Верхняя категория: Low
===Нижняя категория: Low
TOP - ['Middle']
OTHER - ['Low']
СУММА отсеченных сверху и снизу = 63
[-929.5079365079366, -696.952380952381, -46.03174603174604, -
108.99999999999997, -1315.1269841269843]
DISTANCE = 342.9047619047633
+++++

Количество отсеченных сверху = 57
Количество отсеченных снизу = 6
===Верхняя категория: Low
===Нижняя категория: Low
TOP - ['Low']
OTHER - ['Middle']
СУММА отсеченных сверху и снизу = 63
[1009.0, 48.0, -929.0, -1056.0, -13.0]
DISTANCE = 205.0
+++++
```

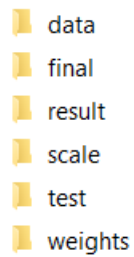
Выберем в качестве целевого класс «Low». Он полностью отсекается. В выборке остался только один экземпляр класса «Middle». Градиентное уточнение в данном случае уже не нужно.

Необходимо скопировать вектор весов из соответствующего файла:

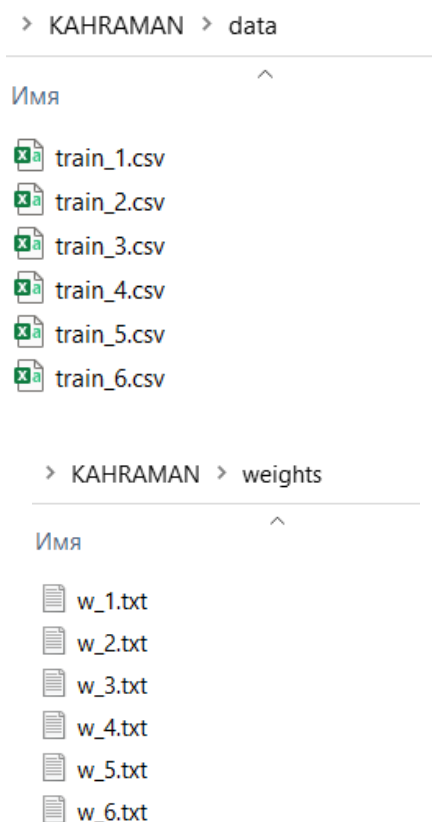
```
w_['Low']['Middle'].txt
w_['Middle']['Low'].txt
```

14. Проверка результатов

Создадим папки data, weights, test, scale, final, result.



В папку data переместим обучающие выборки для каждого нейрона. В папку weights соответствующие вектора весов. Тестовая выборка должна быть помещена в папку test. Принцип наименования должен соответствовать следующему виду:



Для генерации выходной логической функции используется функция `generate_fa`, в которой первый параметр — путь к папке, содержащей обучающие выборки каждого нейрона, второй параметр — путь к папке, содержащей вектора весов каждого нейрона, третий параметр — название столбца с метками классов.

```
wdl.generate_fa('data', 'weights', 'UNS')
```

2 Сверточный слой

Для примера работы использован набор данных MNIST и структура нейронной сети LeNet-5:

https://github.com/SlinkoIgor/Neural_Networks_and_CV/blob/master/module05_mnist_conv.ipynb

По ссылке используется библиотека Pytorch, код был преобразован для использования в библиотеке TensorFlow. Структура была сокращена до одного сверточного слоя с 4 ядрами и одного полносвязного слоя с 10 нейронами. Для работы необходимо установить библиотеку TensorFlow:

```
pip install tensorflow
```

В LeNet-5 в первом сверточном слое используется 6 сверточных ядер 3×3 с шагом 1. Число вычислительных операций для слоя равно:

```
wdl.count_conv_operations(28, 28, 1, 3, 1, 6)
```

```
Кол-во вычислительных операций = 73008  
Кол-во вычислительных операций (no_bias) = 72954
```

Первые два параметра функции — это высота и ширина исходного изображения, третий параметр — количество цветовых каналов, четвертый параметр — размерность сверточного ядра, пятый — шаг сдвига сверточного ядра по изображению, шестой — количество сверточных ядер.

Теперь сравним ресурсоемкость вычислений при использовании четырех сверточных ядер размерности 14×14 с шагом 7:

```
wdl.count_conv_operations(28, 28, 1, 14, 7, 4)
```

```
Кол-во вычислительных операций = 14112  
Кол-во вычислительных операций (no_bias) = 13328
```

Таким образом, это снизит количество вычислительных операций.

Зададим структуру модели следующим образом, отключим обучение сверточного слоя:

```
from keras import models
from keras import layers

model = models.Sequential([
    layers.Conv2D(4, (14,14), activation='relu', strides=(7, 7), input_shape=(28,28,1), use_bias=False, trainable=False),
    layers.Flatten(),
    layers.Dense(10, 'softmax')
])
```

Сгенерируем горизонтальное ядро размера 14*14, состоящее из значений диапазона от -7 до 7 с шагом в 2:

```
wdl.horizontal(14, [-7,-5,-3,-1,1,3,5,7])
```

```
[[-7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7],
 [-5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5],
 [-3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3],
 [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
 [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5],
 [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7],
 [-7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7],
 [-5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5],
 [-3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3],
 [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]]
```

Аналогичным образом сгенерируем вертикальное ядро:

```
wdl.vertical(14, [-7,-5,-3,-1,1,3,5,7])
```

```
[[-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3],
 [-7, -5, -3, -1, 1, 3, 5, 7, -7, -5, -3, -1, 1, 3]]
```

В случае генерации диагональных ядер сразу генерируются два ядра: главная и побочная диагональ:

```
wdl.diagonal([-7, -5, -3, -1], [1, 3, 5, 7], 1, 14)
```

```
[[1, -7, -5, -3, -1, -7, -5, -3, -1, -7, -5, -3, -1, -7],  
 [1, 1, -7, -5, -3, -1, -7, -5, -3, -1, -7, -5, -3, -1],  
 [3, 1, 1, -7, -5, -3, -1, -7, -5, -3, -1, -7, -5, -3],  
 [5, 3, 1, 1, -7, -5, -3, -1, -7, -5, -3, -1, -7, -5],  
 [7, 5, 3, 1, 1, -7, -5, -3, -1, -7, -5, -3, -1, -7],  
 [1, 7, 5, 3, 1, 1, -7, -5, -3, -1, -7, -5, -3, -1],  
 [3, 1, 7, 5, 3, 1, 1, -7, -5, -3, -1, -7, -5, -3],  
 [5, 3, 1, 7, 5, 3, 1, 1, -7, -5, -3, -1, -7, -5],  
 [7, 5, 3, 1, 7, 5, 3, 1, 1, -7, -5, -3, -1, -7],  
 [1, 7, 5, 3, 1, 7, 5, 3, 1, 1, -7, -5, -3, -1],  
 [3, 1, 7, 5, 3, 1, 7, 5, 3, 1, 1, -7, -5, -3],  
 [5, 3, 1, 7, 5, 3, 1, 7, 5, 3, 1, 1, -7, -5],  
 [7, 5, 3, 1, 7, 5, 3, 1, 7, 5, 3, 1, 1, -7],  
 [1, 7, 5, 3, 1, 7, 5, 3, 1, 7, 5, 3, 1, 1]]
```

Первый параметр – список значений, которые будут находиться выше диагонали, второй параметр – список значений, которые будут находиться ниже диагонали, третий параметр – значение, из которого будет состоять диагональ, последний параметр – размерность сверточного ядра.