Нейронные сети

Практические задания

Единый сборник описаний практических заданий

Содержание

Вводная часть

Данный файл не является самостоятельной частью практической составляющей курса «Нейронные сети». Актуальная навигация по материалам курса ведется через сайт дисциплины: https://gitverse.ru/gurgutan/nn-site. Текст сборника «Нейронные сети. Практические задания» не является самостоятельным учебным материалом. Он предназначен только для того, чтобы в одном файле собрать описания работ для удобной навигации по ним и создания комплекта документов по учебной дисциплине.

01-task-01. Проверка условий теорем представления и аппроксимации

Цель

Проверить простые свойства классов функций, которые используются в формулировках теорем представления и аппроксимации: наличие постоянной функции, замкнутость относительно операций, разделение точек и связь этих свойств с теоремами лекции 1.

Задание относится к лекции 1 «Основные теоремы нейроинформатики». В нем не требуется строить нейронную сеть, обучать модель или применять методы оптимизации.

Вариант

Работа выполняется на компакте:

```
X = [0,1]
```

Рассмотрите пять классов функций на **X**.

class_id	Класс функций
A	все функции вида **f(x)** = **alpha x** + **beta**
B	все постоянные функции **f(x)** = **beta**
C	все функции вида **f(x)** = **alpha x^2** + **beta**
D	все многочлены от **x**
E	все функции вида **f(x)** = **alpha sin(x)** + **beta**

Здесь **alpha** и **beta** - произвольные действительные числа.

Проверяемые свойства

Для каждого класса функций заполните значения следующих полей.

Поле	Значение **1**	Значение **0**
contains_one	класс содержит постоянную функцию **1**	класс не содержит постоянную функцию **1**
linear_closed	любая линейная комбинация двух функций класса снова принадлежит этому классу	найдется линейная комбинация функций класса, которая не принадлежит этому классу
product_closed	произведение любых двух функций класса снова принадлежит этому классу	найдется произведение функций класса, которое не принадлежит этому классу
separates_points	для любых разных **x_1, x_2** из **X** найдется функция класса, различающая эти точки	есть разные точки **x_1, x_2**, которые не различаются ни одной функцией класса
basic_conditions_ready	первые четыре свойства равны **1**	хотя бы одно из первых четырех свойств равно **0**

Поле **basic_conditions_ready** означает только то, что четыре перечисленных свойства проверены положительно для данного класса. Оно не является доказательством равенства класса всему **C(X)**.

Что нужно сделать

1. Для каждого класса **A**, **B**, **C**, **D**, **E** определить значения всех пяти свойств.
2. Создать файл **results/01-task-01-answer.csv**.
3. Записать ответ в точном формате из следующего раздела.
4. При необходимости создать **notes.md** с замечаниями. Этот файл не участвует в автоматической проверке.

Формат ответа

Файл **results/01-task-01-answer.csv** должен содержать ровно 6 непустых строк: строку заголовка и по одной строке для каждого **class_id**.

Столбцы должны идти строго в таком порядке:

```
class_id,contains_one,linear_closed,product_closed,separates_points,basic_conditions_ready
```

Каждое значение, кроме **class_id**, должно быть только **0** или **1**.

Шаблон файла:

```
class_id,contains_one,linear_closed,product_closed,separates_points,basic_conditions_ready
A,0,0,0,0,0
B,0,0,0,0,0
C,0,0,0,0,0
```

```
D,0,0,0,0,0
E,0,0,0,0,0
```

Числа в шаблоне не являются ответом. Они показывают только формат.

Шаблон ответа

Чтобы создать пустой шаблон CSV-файла, выполните из папки задания:

```
uv run python task.py
```

Скрипт создает **results/01-task-01-answer.csv**, если такого файла еще нет. Он не содержит эталонных ответов и не проверяет правильность решения.

Что сдавать

Сдайте папку задания в таком виде:

```
01-task-01/
├── task.py
├── results/
│   └── 01-task-01-answer.csv
└── notes.md
```

notes.md нужен только для замечаний о неоднозначности условия, технической проблеме или спорном пограничном случае.

02-task-01. Ориентированный граф и функция на графе

Цель
Научиться задавать простую вычислительную сеть как ориентированный граф с именованными вершинами, упорядочивать входящие дуги вершины, записывать функцию графа в префиксной скобочной форме и вычислять значение этой функции при заданных операциях в вершинах.

Материал лекции
Задание относится к лекции 2 «Формальное описание искусственных нейронных сетей» и опирается на понятия:

1. искусственная нейронная сеть как ориентированный граф вычислений;
2. вершина графа как вычислительный элемент;
3. направленная дуга как передача значения от одной вершины к другой;
4. входы вершины и порядок входящих дуг;
5. вычисление значения составной функции по ациклическому графу;
6. цикл как препятствие для однозначного прямого вычисления.

В этом задании не требуется обучать модель, подбирать веса или использовать библиотеки для нейронных сетей.

Формат дуги
Одна дуга задается строкой:

```
(a, b, n)
```

где:

1. **a** - имя начальной вершины;
2. **b** - имя конечной вершины;
3. **n** - порядковый номер этой дуги среди всех дуг, входящих в вершину **b**.

Например, строки

```
(v1, v3, 1)
(v2, v3, 2)
```

означают, что вершина **v3** получает первый аргумент из **v1**, а второй аргумент из **v2**.

Часть 1. Создание ориентированного графа
Для каждого случая задан список строк с дугами. Нужно:

1. разобрать строки;
2. проверить формат каждой строки;
3. проверить, что номера входящих дуг каждой вершины начинаются с **1**, не повторяются и идут без пропусков;
4. сериализовать корректный граф в JSON;

5. для некорректного графа указать номер строки с первой найденной ошибкой и краткую метку ошибки.

В сериализации используйте структуру:

```
{"arcs":[{"from":"v1","order":1,"to":"v3"}],"vertices":["v1","v3"]}
```

Вершины упорядочивайте лексикографически. Дуги в поле **arcs** оставляйте в том порядке, в котором они приведены во входном списке.

Базовый вариант:

graph_id	lines
G1	**(v1, v3, 1)**; **(v2, v3, 2)**; **(v3, v4, 1)**
G2	**(v1, v3, 1)**; **(v2, v3, 1)**
G3	**(v1, v2)**
G4	**(b, h, 1)**; **(x, h, 2)**; **(h, y, 1)**

Метки ошибок:

Метка	Смысл
invalid_arc_format	строка не имеет вида **(a, b, n)**
duplicate_incoming_order	у одной вершины повторился номер входящей дуги
missing_incoming_order	у одной вершины есть пропуск в нумерации входящих дуг

Формат файла 02-task-01-graphs.csv
Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **graph_id**.

Столбцы должны идти строго в таком порядке:

```
graph_id,serialized_graph,error_line,error_message
```

Правила заполнения:

1. **serialized_graph** - JSON-строка для корректного графа или пустая строка при ошибке;
2. **error_line** - номер строки с первой ошибкой или пустая строка для корректного графа;
3. **error_message** - метка ошибки или пустая строка для корректного графа.

Пример структуры файла:

```
graph_id,serialized_graph,error_line,error_message
G1,"{}",,
G2,,0,error
G3,,0,error
G4,"{}",,
```

Числа и JSON в примере не являются ответом. Они показывают только формат.

Часть 2. Создание функции по графу

Для каждого случая задан ациклический или циклический ориентированный граф и корневая вершина. Корневая вершина считается выходом функции.

Нужно построить линейную префиксную скобочную запись:

```
A(B(C),D)
```

Здесь **A** - корневая вершина, а **B** и **D** - ее аргументы, упорядоченные по номерам входящих дуг. Если у вершины нет входящих дуг, она записывается только своим именем.

Если в графе есть цикл, префиксную запись строить нельзя: в этом случае поставьте **has_cycle** = **1**, а поле **prefix_form** оставьте пустым.

Базовый вариант:

function_id	root	lines
F1	**v4**	**(v1, v3, 1)**; **(v2, v3, 2)**; **(v3, v4, 1)**
F2	**z**	**(x, a, 1)**; **(y, a, 2)**; **(a, z, 1)**; **(c, z, 2)**
F3	**a**	**(a, b, 1)**; **(b, a, 1)**
F4	**out**	**(x, e, 1)**; **(e, out, 1)**

Формат файла 02-task-01-functions.csv

Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **function_id**.

Столбцы должны идти строго в таком порядке:

```
function_id,has_cycle,prefix_form
```

Правила заполнения:

1. **has_cycle** - **1**, если в графе найден цикл, иначе **0**;
2. **prefix_form** - префиксная скобочная запись функции или пустая строка при цикле.

Часть 3. Вычисление значения функции на графе

Для каждого случая задан граф, корневая вершина и словарь операций в вершинах.

Допустимые операции:

Запись	Смысл
+	сумма входных значений вершины
*****	произведение входных значений вершины
exp	экспонента от единственного входного значения
число	числовая константа в вершине без входящих дуг

В базовом варианте **exp** применяется только к нулю, поэтому результат остается целым числом.

Базовый вариант:

value_id	root	lines	operations
V1	**out**	**(x, s, 1); (y, s, 2); (s, out, 1); (c, out, 2)**	**x:2, y:3, s:+, c:4, out:***
V2	**out**	**(x, e, 1); (e, out, 1); (c, out, 2)**	**x:0, e:exp, c:3, out:+**
V3	**out**	**(x, m, 1); (y, m, 2); (m, out, 1); (b, out, 2)**	**x:2, y:5, m:*, b:3, out:+**
V4	**out**	**(x, s, 1); (y, s, 2); (s, out, 1); (c, out, 2)**	**x:−1, y:4, s:+, c:2, out:***

Формат файла 02-task-01-values.csv

Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **value_id**.

Столбцы должны идти строго в таком порядке:

```
value_id,value_num,value_den
```

Правила заполнения:

1. **value_num** - числитель значения функции;
2. **value_den** - знаменатель значения функции.

Если значение является целым числом, знаменатель равен **1**.

03-task-01. Вычисления на ориентированном графе

Цель

Научиться записывать ориентированный граф вычислений через матрицу смежности, выполнять один шаг распространения сигнала по взвешенным дугам и прослеживать простую итерационную схему на графе с циклом.

Материал лекции

Задание относится к лекции 3 «Вычисления на графе» и опирается на понятия:

1. ориентированный граф **(V,E)**;
2. вершины и дуги графа вычислений;
3. матрица смежности ориентированного графа;
4. веса дуг;
5. один шаг вычислений **x = y A**;
6. покомпонентное применение функции активации;
7. рациональная сигмоида **f(z) = z / (1 + |z|)**;
8. итерационный процесс **y^((n)) = f(y^((n-1)) A)**.

В этом задании не требуется обучать модель, подбирать веса или реализовывать нейронную сеть программно.

Параметры варианта

Вариант состоит из трех независимых частей:

1. список ориентированных графов с заданными вершинами и дугами;
2. список взвешенных графов с начальными значениями вершин;
3. список одномерных итерационных схем на графе с одной вершиной и петлей.

Нумерация вершин начинается с **1**. Дуга **(i,j)** означает направление от **v_i** к **v_j**. В матрице смежности этому соответствует элемент **M_ij = 1**.

Часть 1. Структура ориентированного графа

Для каждой строки задано:

1. **graph_id** - идентификатор графа;
2. **vertex_count** - число вершин;
3. **edges** - список ориентированных дуг.

Базовый вариант:

graph_id	vertex_count	edges
G1	4	**(1,2);(1,3);(3,4)**
G2	4	**(1,2);(2,3);(3,1);(3,4)**
G3	3	**(1,1);(1,2);(2,3)**
G4	5	**(1,2);(2,3);(3,4);(4,5)**
G5	4	**(2,1);(2,3);(4,2);(4,3)**

Для каждого графа нужно определить:

1. число дуг;
2. исходящую степень вершины **v_1**;
3. входящую степень последней вершины графа;
4. содержит ли граф хотя бы один цикл;
5. содержит ли граф хотя бы одну стоковую вершину, то есть вершину без исходящих дуг.

Петля **(i,i)** считается циклом.

Формат файла 03-task-01-structure.csv
Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **graph_id**.

Столбцы должны идти строго в таком порядке:

```
graph_id,edge_count,out_degree_v1,in_degree_last,has_cycle,has_sink
```

Правила заполнения:

1. **edge_count** - число дуг в графе;
2. **out_degree_v1** - число дуг, выходящих из **v_1**;
3. **in_degree_last** - число дуг, входящих в последнюю вершину графа;
4. **has_cycle** - **1**, если в графе есть цикл, иначе **0**;
5. **has_sink** - **1**, если в графе есть хотя бы одна вершина без исходящих дуг, иначе **0**.

Пример структуры файла:

```
graph_id,edge_count,out_degree_v1,in_degree_last,has_cycle,has_sink
G1,0,0,0,0,0
G2,0,0,0,0,0
G3,0,0,0,0,0
G4,0,0,0,0,0
G5,0,0,0,0,0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 2. Один шаг вычислений на графе
Для каждого случая задано:

1. **case_id** - идентификатор случая;
2. **y** - строка текущих значений вершин;
3. **weighted_edges** - список взвешенных дуг **(i,j,w_ij)**.

Постройте матрицу **A**, где **A_ij** = **w_ij**, если есть дуга из **v_i** в **v_j**, и **A_ij** = **0**, если такой дуги нет. Затем вычислите

```
x = y A,
y_new = f(x),
f(z) = z / (1 + |z|).
```

Функция **f** применяется к каждой координате отдельно.

Базовый вариант:

case_id	y	weighted_edges
U1	**(1;-1;2)**	**(1,2,2);(2,3,-1);(3,1,1)**
U2	**(2;0;-1)**	**(1,2,-1);(1,3,2);(3,2,3)**
U3	**(1;1;1)**	**(1,1,-2);(2,3,3);(3,2,-1)**

Для каждой координаты каждого случая нужно указать:

1. значение до активации **x_j**;
2. значение после активации **f(x_j)** в виде несократимой дроби;
3. знак значения после активации.

Формат файла 03-task-01-step.csv
Файл должен содержать ровно 10 строк:

1. строка заголовка;
2. по одной строке для каждой пары **(case_id, vertex_id)**.

Столбцы должны идти строго в таком порядке:

```
case_id,vertex_id,pre_activation,output_num,output_den,output_sign
```

Правила заполнения:

1. **vertex_id** - номер вершины **1**, **2** или **3**;
2. **pre_activation** - значение **x_j** до применения функции активации;
3. **output_num** - числитель дроби **f(x_j)**;
4. **output_den** - положительный знаменатель дроби **f(x_j)**;
5. **output_sign** - **-1**, если **f(x_j)** < **0**, **0**, если **f(x_j)** = **0**, **1**, если **f(x_j)** > **0**.

Дробь должна быть несократимой. Например, значение **-2/3** записывается как **output_num** = **−2**, **output_den** = **3**.

Пример структуры файла:

```
case_id,vertex_id,pre_activation,output_num,output_den,output_sign
U1,1,0,0,1,0
U1,2,0,0,1,0
U1,3,0,0,1,0
```

```
U2,1,0,0,1,0
U2,2,0,0,1,0
U2,3,0,0,1,0
U3,1,0,0,1,0
U3,2,0,0,1,0
U3,3,0,0,1,0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 3. Две итерации на графе с циклом

Рассмотрим граф с одной вершиной **v_1** и одной петлей **(1,1)** веса **a**. Для начального значения **y^((0))** выполните две итерации:

```
y^((1)) = f(a y^((0))),
y^((2)) = f(a y^((1))),
f(z) = z / (1 + |z|).
```

Базовый вариант:

iteration_id	a	y^((0))
R1	1	1
R2	−1	1
R3	2	1
R4	−2	−1
R5	3	−1

Для каждого случая нужно указать:

1. итоговое значение **y^((2))** в виде несократимой дроби;
2. уменьшился ли модуль значения после двух итераций по сравнению с начальным значением.

Формат файла 03-task-01-iterations.csv

Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **iteration_id**.

Столбцы должны идти строго в таком порядке:

```
iteration_id,final_num,final_den,is_abs_decreased
```

Правила заполнения:

1. **final_num** - числитель дроби **y^((2))**;
2. **final_den** - положительный знаменатель дроби **y^((2))**;
3. **is_abs_decreased** - **1**, если $|y^{((2))}| < |y^{((0))}|$, иначе **0**.

Дробь должна быть несократимой.

Пример структуры файла:

```
iteration_id,final_num,final_den,is_abs_decreased
R1,0,1,0
R2,0,1,0
R3,0,1,0
R4,0,1,0
R5,0,1,0
```

Числа в примере не являются ответом. Они показывают только формат.

Что нужно сдать

Нужно сдать три файла:

1. **03-task-01-structure.csv**;
2. **03-task-01-step.csv**;
3. **03-task-01-iterations.csv**.

При необходимости можно дополнительно приложить файл **notes.md**. Он предназначен только для замечаний о неоднозначностях, спорных случаях или технических проблемах. **notes.md** не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. наличие всех трех обязательных файлов;
2. точное совпадение заголовков CSV-файлов;
3. наличие всех идентификаторов **G1**-**G5**, **U1**-**U3**, **R1**-**R5**;
4. корректность целочисленных значений и булевых полей **0/1**;
5. корректность несократимых дробей с положительным знаменателем;
6. отсутствие лишних строк и лишних столбцов.

Как можно варьировать задание

Для разных вариантов можно менять:

1. число вершин и список дуг в части 1;
2. начальный вектор **y** в части 2;
3. список взвешенных дуг **(i,j,w_ij)** в части 2;
4. вес петли **a** и начальное значение **y^((0))** в части 3;
5. число строк в каждой части при сохранении тех же столбцов ответа.

04-task-01. Функции потерь и шаг оптимизации

Цель

Научиться вычислять эмпирическую ошибку на небольшой обучающей выборке, сопоставлять функцию потерь с типом задачи и выполнять один шаг итерационной оптимизации по заданному направлению.

Материал лекции

Задание относится к лекции 4 «Обучение как задача многомерной оптимизации» и опирается на понятия:

1. обучающая выборка **D = {(x^((m)), y^((m)))}**;
2. функция потерь **ell(f_theta(x), y)** на одном примере;
3. эмпирический риск **D_D(theta)**;
4. средняя абсолютная ошибка;
5. среднеквадратичная ошибка;
6. бинарная кросс-энтропия;
7. вектор параметров **theta**;
8. общий шаг итерационной оптимизации **theta_(k+1) = theta_k + eta_k p_k**;
9. шаг градиентного спуска **theta_(k+1) = theta_k - eta_k nabla D_D(theta_k)**;
10. классы методов оптимизации.

В этом задании не требуется обучать модель, вычислять градиент самостоятельно или реализовывать нейронную сеть программно. Все предсказания, направления движения и градиенты заданы в условии.

Параметры варианта

Вариант состоит из трех независимых частей:

1. наборы целевых значений и готовых предсказаний для расчета MAE и MSE;
2. наборы бинарных вероятностных прогнозов для расчета кросс-энтропии;
3. шаги итерационной оптимизации с заданными параметрами, направлениями и градиентами.

Нумерация примеров и координат начинается с **1**.

Часть 1. Эмпирический риск для регрессии

Для каждого случая задано:

1. **risk_id** - идентификатор случая;
2. **y_true** - целевые значения на обучающей выборке;
3. **y_pred** - предсказания модели при некотором фиксированном **theta**.

Базовый вариант:

risk_id	y_true	y_pred
L1	**(1;2;3)**	**(1;1;5)**
L2	**(0;-1;2;4)**	**(1;-1;0;5)**
L3	**(2;2;2)**	**(0;2;4)**
L4	**(-1;1;3;5)**	**(-2;0;4;8)**

Для каждого случая нужно вычислить:

1. число примеров **sample_count**;
2. сумму абсолютных ошибок;
3. сумму квадратов ошибок;
4. среднюю абсолютную ошибку **MAE**;
5. среднеквадратичную ошибку **MSE**;
6. какая из двух функций потерь сильнее штрафует крупные ошибки в данном случае.

Для поля **larger_penalty_loss** используйте словарь:

Метка	Смысл
mse	**MSE > MAE**
mae	**MAE > MSE**
equal	**MAE = MSE**

Формат файла 04-task-01-regression.csv
Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **risk_id**.

Столбцы должны идти строго в таком порядке:

```
risk_id,sample_count,abs_error_sum,squared_error_sum,mae_num,mae_den,mse_num,mse_den,larger_pena
```

Правила заполнения:

1. **sample_count** - число примеров в случае;
2. **abs_error_sum** - сумма абсолютных ошибок;
3. **squared_error_sum** - сумма квадратов ошибок;
4. **mae_num**, **mae_den** - числитель и положительный знаменатель несократимой дроби **MAE**;
5. **mse_num**, **mse_den** - числитель и положительный знаменатель несократимой дроби **MSE**;
6. **larger_penalty_loss** - одна из меток **mse**, **mae**, **equal**.

Пример структуры файла:

```
risk_id,sample_count,abs_error_sum,squared_error_sum,mae_num,mae_den,mse_num,mse_den,larger_pena
L1,0,0,0,0,1,0,1,equal
L2,0,0,0,0,1,0,1,equal
L3,0,0,0,0,1,0,1,equal
L4,0,0,0,0,1,0,1,equal
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 2. Бинарная кросс-энтропия

Для бинарной классификации используйте функцию потерь

```
ell(z,y) = - (y log z + (1 - y) log(1 - z)),
```

где **z** - предсказанная вероятность класса **1**, а **y** - правильная метка.

Во всех строках базового варианта вероятности выбраны так, что каждая потеря равна целому коэффициенту, умноженному на **ln 2**. Например, если правильному классу назначена вероятность **1/4**, то потеря равна **2 ln 2**.

Базовый вариант:

ce_id	y	z
B1	1	**1/2**
B2	0	**1/2**
B3	1	**1/4**
B4	0	**3/4**
B5	1	**1/8**
B6	0	**7/8**

Для каждого случая нужно определить:

1. вероятность, назначенную правильному классу;
2. коэффициент **c** в записи **ell(z,y) = c ln 2**;
3. является ли прогноз уверенным, но ошибочным.

Считайте прогноз уверенным, но ошибочным, если правильному классу назначена вероятность меньше или равная **1/4**.

Формат файла 04-task-01-cross-entropy.csv

Файл должен содержать ровно 7 строк:

1. строка заголовка;
2. по одной строке для каждого **ce_id**.

Столбцы должны идти строго в таком порядке:

```
ce_id,true_prob_num,true_prob_den,loss_ln2_coeff,is_confident_error
```

Правила заполнения:

1. **true_prob_num**, **true_prob_den** - числитель и положительный знаменатель вероятности правильного класса;
2. **loss_ln2_coeff** - целый коэффициент **c** в формуле **ell(z,y)** = **c ln 2**;
3. **is_confident_error** - **1**, если правильному классу назначена вероятность меньше или равная **1/4**, иначе **0**.

Пример структуры файла:

```
ce_id,true_prob_num,true_prob_den,loss_ln2_coeff,is_confident_error
B1,0,1,0,0
B2,0,1,0,0
B3,0,1,0,0
B4,0,1,0,0
B5,0,1,0,0
B6,0,1,0,0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 3. Один шаг оптимизации

Для каждого случая задано:

1. **step_id** - идентификатор случая;
2. **theta** - текущий вектор параметров;
3. **eta** - длина шага;
4. **direction_type** - способ задания направления;
5. **p** или **grad** - направление движения или градиент.

Если **direction_type** = **general**, используйте общий шаг

```
theta_next = theta + eta p.
```

Если **direction_type** = **gradient_descent**, используйте шаг градиентного спуска

```
theta_next = theta - eta grad.
```

Базовый вариант:

step_id	theta	eta	direction_type	p	grad
O1	**(1;2)**	1	**general**	**(-1;3)**	пусто
O2	**(0;-2;4)**	**1/2**	**general**	**(2;0;-2)**	пусто
O3	**(3;1)**	**2**	**gradient_descent**	пусто	**(1;-1)**
O4	**(-1;0;2)**	**1/8**	**gradient_descent**	пусто	**(-2;4;0)**
O5	**(2;-2)**	**1/4**	**gradient_descent**	пусто	**(4;8)**

Для каждого случая нужно вычислить следующий вектор параметров и указать класс метода.

Для поля **method_order** используйте словарь:

Метка	Смысл
zero_order	направление задано без градиента
first_order	используется градиент

Формат файла 04-task-01-optimization.csv
Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **step_id**.

Столбцы должны идти строго в таком порядке:

```
step_id,theta_next,method_order,is_gradient_descent
```

Правила заполнения:

1. **theta_next** - следующий вектор параметров в формате **(a;b;c)** без пробелов;
2. дробные координаты записываются как несократимые дроби, например **(1/2;-3;0)**;
3. **method_order** - одна из меток **zero_order**, **first_order**;
4. **is_gradient_descent** - **1**, если используется формула градиентного спуска, иначе **0**.

Пример структуры файла:

```
step_id,theta_next,method_order,is_gradient_descent
01,(0;0),zero_order,0
02,(0;0;0),zero_order,0
03,(0;0),zero_order,0
04,(0;0;0),zero_order,0
05,(0;0),zero_order,0
```

Числа и метки в примере не являются ответом. Они показывают только формат.

Что нужно сдать
Нужно сдать три файла:

1. **04-task-01-regression.csv**;
2. **04-task-01-cross-entropy.csv**;
3. **04-task-01-optimization.csv**.

При необходимости можно дополнительно приложить файл **notes.md**. Он предназначен только для замечаний о неоднозначностях, спорных случаях или технических проблемах. **notes.md** не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. наличие всех трех обязательных файлов;

2. точное совпадение заголовков CSV-файлов;
3. наличие всех идентификаторов **L1**-**L4**, **B1**-**B6**, **O1**-**O5**;
4. корректность целых чисел, дробей и булевых полей **0/1**;
5. принадлежность строковых меток фиксированным словарям;
6. отсутствие лишних строк и лишних столбцов.

Как можно варьировать задание

Для разных вариантов можно менять:

1. целевые значения и готовые предсказания в части 1;
2. число примеров в каждом случае части 1;
3. вероятности правильного класса в части 2, сохраняя степени **1/2**, если нужна проверка через коэффициент при **ln 2**;
4. векторы **theta**, шаги **eta**, направления **p** и градиенты **grad** в части 3;
5. число строк в каждой части при сохранении тех же столбцов ответа.

05-task-01. Градиентный шаг и наискорейший спуск

Цель

Научиться выполнять один шаг градиентного спуска, проверять направление убывания функции потерь и вычислять шаг наискорейшего спуска для простого линейного нейрона с квадратичной функцией потерь.

Материал лекции

Задание относится к лекции 5 «Общая схема градиентного обучения» и опирается на понятия:

1. эмпирическая функция потерь **L(w)**;
2. общая итерационная схема **w_(t+1)** = **w_t** + **eta_t p_t**;
3. градиент функции потерь;
4. антиградиентное направление **p_t** = **- nabla L(w_t)**;
5. градиентный спуск **w_(t+1)** = **w_t - eta_t nabla L(w_t)**;
6. метод наискорейшего спуска;
7. линейный нейрон **hat(y)** = **X w**;
8. квадратичная функция потерь **L(w) = 1/2 ||Xw - y||_2^2**;
9. формула шага **eta_t = ||nabla L(w_t)||_2^2 / ||X nabla L(w_t)||_2^2**.

В этом задании не требуется обучать модель до сходимости или реализовывать нейронную сеть программно.

Параметры варианта

Вариант состоит из двух независимых частей:

1. список готовых градиентов и шагов обучения для одного обновления параметров;
2. список малых линейных задач, где нужно вычислить шаг наискорейшего спуска.

Координаты векторов записываются через точку с запятой: **(a;b;c)**. Дробные координаты в ответе записываются как несократимые дроби.

Часть 1. Один градиентный шаг

Для каждого случая задано:

1. **case_id** - идентификатор случая;
2. **w** - текущий вектор параметров;
3. **grad** - градиент **nabla L(w)**;
4. **eta** - шаг обучения.

Базовый вариант:

case_id	w	grad	eta
G1	**(2;-1)**	**(1;3)**	1
G2	**(0;4;2)**	**(-2;0;4)**	**1/2**
G3	**(3;3)**	**(0;0)**	2
G4	**(-1;2)**	**(4;-2)**	**1/4**
G5	**(1;-2;0)**	**(2;-4;6)**	**1/2**

Для каждого случая нужно вычислить:

1. антиградиентное направление **p** = **-grad**;
2. следующий вектор параметров **w_next** = **w - eta grad**;
3. скалярное произведение **p^T grad**;
4. является ли **p** направлением убывания по линейной аппроксимации.

Считайте, что направление является направлением убывания, если **p^T grad < 0**.

Формат файла 05-task-01-gradient-step.csv
Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **case_id**.

Столбцы должны идти строго в таком порядке:

```
case_id,p,w_next,inner_product,is_descent_direction
```

Правила заполнения:

1. **p** - антиградиентное направление в формате **(a;b;c)**;
2. **w_next** - следующий вектор параметров в формате **(a;b;c)**;
3. дробные координаты записываются как несократимые дроби;
4. **inner_product** - значение **p^T grad**;
5. **is_descent_direction** - **1**, если **p^T grad < 0**, иначе **0**.

Пример структуры файла:

```
case_id,p,w_next,inner_product,is_descent_direction
G1,(0;0),(0;0),0,0
G2,(0;0;0),(0;0;0),0,0
G3,(0;0),(0;0),0,0
G4,(0;0),(0;0),0,0
G5,(0;0;0),(0;0;0),0,0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 2. Шаг наискорейшего спуска
Для линейного нейрона с квадратичной функцией потерь используйте формулы:

```
r = Xw - y,
grad = X^T r,
eta = ||grad||_2^2 / ||X grad||_2^2,
w_next = w - eta grad.
```

Для каждого случая задано:

1. **steepest_id** - идентификатор случая;
2. **X** - матрица признаков;
3. **w** - текущий вектор параметров;
4. **y** - целевой вектор.

Матрица **X** записана построчно: **[(a;b);(c;d)]**.

Базовый вариант:

steepest_id	X	w	y
S1	**[(1;0);(0;1)]**	**(2;-1)**	**(0;0)**
S2	**[(2;0);(0;1)]**	**(1;2)**	**(0;0)**
S3	**[(1;1);(1;-1)]**	**(1;0)**	**(0;2)**
S4	**[(1;0);(0;3)]**	**(3;1)**	**(1;1)**

Для каждого случая нужно вычислить:

1. остаток **r**;
2. градиент **grad**;
3. числитель **eta_num** и знаменатель **eta_den** несократимой дроби для **eta**;
4. следующий вектор параметров **w_next**.

Формат файла 05-task-01-steepest.csv

Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **steepest_id**.

Столбцы должны идти строго в таком порядке:

```
steepest_id,residual,grad,eta_num,eta_den,w_next
```

Правила заполнения:

1. **residual** - вектор **r** = **Xw - y**;
2. **grad** - вектор **X^T r**;
3. **eta_num**, **eta_den** - числитель и положительный знаменатель несократимой дроби **eta**;
4. **w_next** - следующий вектор параметров;
5. все векторы записываются без пробелов в формате **(a;b)**.

Пример структуры файла:

```
steepest_id,residual,grad,eta_num,eta_den,w_next
S1,(0;0),(0;0),0,1,(0;0)
```

```
S2,(0;0),(0;0),0,1,(0;0)
S3,(0;0),(0;0),0,1,(0;0)
S4,(0;0),(0;0),0,1,(0;0)
```

Числа в примере не являются ответом. Они показывают только формат.

Что нужно сдать
Нужно сдать два файла:

1. **05-task-01-gradient-step.csv**;
2. **05-task-01-steepest.csv**.

При необходимости можно дополнительно приложить файл **notes.md**. Он предназначен только для замечаний о неоднозначностях, спорных случаях или технических проблемах. **notes.md** не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. наличие обоих обязательных файлов;
2. точное совпадение заголовков CSV-файлов;
3. наличие всех идентификаторов **G1**-**G5** и **S1**-**S4**;
4. корректность векторов, целых чисел, дробей и булевых полей **0**/**1**;
5. положительность знаменателей и несократимость дробей;
6. отсутствие лишних строк и лишних столбцов.

Как можно варьировать задание
Для разных вариантов можно менять:

1. векторы **w**, **grad** и шаги **eta** в части 1;
2. матрицы **X**, текущие параметры **w** и целевые значения **y** в части 2;
3. размерность векторов при сохранении того же формата ответа;
4. число строк в каждой части при сохранении тех же столбцов.

05-task-02. Полный градиент и потоковый SGD

Цель

Научиться отличать полный градиент по обучающей выборке от градиента по одному объекту, выполнять несколько шагов потокового стохастического градиентного спуска и сопоставлять варианты градиентного обучения с их вычислительными свойствами.

Материал лекции

Задание относится к лекции 5 «Общая схема градиентного обучения» и опирается на понятия:

1. средняя эмпирическая функция потерь **L(w) = 1/N sum_n L_n(w)**;
2. полный градиент **nabla L(w) = 1/N sum_n nabla L_n(w)**;
3. пакетный градиентный спуск;
4. потоковый стохастический градиентный спуск;
5. обновление **w_(t+1) = w_t - eta_t nabla L_(n_t)(w_t)**;
6. эпоха как один полный проход по обучающей выборке;
7. преимущества и недостатки потокового SGD.

В этом задании не требуется случайно выбирать объекты или программно обучать модель. Последовательности объектов и градиентов уже заданы.

Параметры варианта

Вариант состоит из трех независимых частей:

1. списки градиентов отдельных объектов для вычисления полного градиента;
2. последовательности одномерных потоковых SGD-обновлений;
3. описания вычислительных режимов, которые нужно классифицировать фиксированными метками.

Координаты векторов записываются через точку с запятой: **(a;b;c)**.

Часть 1. Полный градиент по выборке

Для каждого случая задан список градиентов отдельных объектов **gradients**. Полный градиент равен среднему арифметическому этих векторов.

Базовый вариант:

batch_id	gradients
B1	**(2;0);(0;2);(-1;1)**
B2	**(1;-1);(-1;1)**
B3	**(3;3);(1;-1);(-2;0);(0;2)**
B4	**(4;0);(0;4);(2;2);(-2;2)**

Для каждого случая нужно вычислить:

1. число объектов **sample_count**;
2. полный градиент **full_grad**;
3. направление пакетного градиентного спуска **p_batch** = **-full_grad**;
4. равен ли полный градиент нулю.

Формат файла 05-task-02-full-gradient.csv
Файл должен содержать ровно 5 строк:

1. строка заголовка;
2. по одной строке для каждого **batch_id**.

Столбцы должны идти строго в таком порядке:

```
batch_id,sample_count,full_grad,p_batch,is_full_gradient_zero
```

Правила заполнения:

1. **sample_count** - число градиентов отдельных объектов;
2. **full_grad** - средний градиент в формате **(a;b)**;
3. **p_batch** - антиградиентное направление в формате **(a;b)**;
4. дробные координаты записываются как несократимые дроби;
5. **is_full_gradient_zero** - **1**, если полный градиент равен нулевому вектору, иначе **0**.

Пример структуры файла:

```
batch_id,sample_count,full_grad,p_batch,is_full_gradient_zero
B1,0,(0;0),(0;0),0
B2,0,(0;0),(0;0),0
B3,0,(0;0),(0;0),0
B4,0,(0;0),(0;0),0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 2. Несколько шагов потокового SGD
В каждом случае задано начальное значение одного параметра **w0**, постоянный шаг **eta** и последовательность градиентов отдельных объектов. Нужно выполнить обновления:

```
w_(t+1) = w_t - eta g_t.
```

Базовый вариант:

online_id	w0	eta	gradients
O1	0	1	**2;-1;3**
O2	1	**1/2**	**2;2**
O3	−2	1	**-1;-1;4**
O4	3	**1/4**	**4;-8;4;0**
O5	0	**1/3**	**3;-6;0**

Для каждого случая нужно вычислить:

1. число выполненных обновлений;
2. итоговое значение параметра **w_final**;
3. изменился ли параметр после всех обновлений.

Формат файла 05-task-02-online.csv
Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **online_id**.

Столбцы должны идти строго в таком порядке:

```
online_id,update_count,final_w_num,final_w_den,is_changed
```

Правила заполнения:

1. **update_count** - число примененных градиентов;
2. **final_w_num**, **final_w_den** - числитель и положительный знаменатель несократимой дроби для **w_final**;
3. **is_changed** - **1**, если **w_final** != **w0**, иначе **0**.

Пример структуры файла:

```
online_id,update_count,final_w_num,final_w_den,is_changed
O1,0,0,1,0
O2,0,0,1,0
O3,0,0,1,0
O4,0,0,1,0
O5,0,0,1,0
```

Числа в примере не являются ответом. Они показывают только формат.

Часть 3. Классификация режимов обучения
Для каждого описания нужно выбрать тип режима и указать его свойства.

Используйте словарь **mode_type**:

Метка	Смысл
batch_gradient_descent	на каждом шаге используется полный градиент по всей выборке
online_sgd	на каждом шаге используется градиент одного объекта
steepest_descent	используется антиградиентное направление и одномерный поиск шага

Базовый вариант:

mode_id	description
M1	На каждом шаге усредняются градиенты всех объектов выборки.
M2	После перемешивания выборки параметры обновляются по одному объекту.
M3	Направление равно антиградиенту, а шаг находится минимизацией вдоль прямой.
M4	Один полный проход по всем объектам называется эпохой.
M5	Одна итерация дешева, но направление шага зашумлено.

Для каждого случая нужно определить:

1. тип режима **mode_type**;
2. использует ли режим полный градиент на каждом шаге;
3. использует ли режим один объект на каждом шаге;
4. требует ли режим одномерного поиска шага.

Для **M4** и **M5** укажите режим, к которому относится описанное свойство.

Формат файла 05-task-02-modes.csv

Файл должен содержать ровно 6 строк:

1. строка заголовка;
2. по одной строке для каждого **mode_id**.

Столбцы должны идти строго в таком порядке:

```
mode_id,mode_type,uses_full_gradient,uses_single_object,uses_line_search
```

Правила заполнения:

1. **mode_type** - одна из меток **batch_gradient_descent**, **online_sgd**, **steepest_descent**;
2. **uses_full_gradient** - **1**, если на каждом шаге используется полный градиент, иначе **0**;
3. **uses_single_object** - **1**, если на каждом шаге используется один объект, иначе **0**;
4. **uses_line_search** - **1**, если нужен одномерный поиск шага, иначе **0**.

Пример структуры файла:

```
mode_id,mode_type,uses_full_gradient,uses_single_object,uses_line_search
M1,batch_gradient_descent,0,0,0
M2,batch_gradient_descent,0,0,0
M3,batch_gradient_descent,0,0,0
M4,batch_gradient_descent,0,0,0
M5,batch_gradient_descent,0,0,0
```

Числа и метки в примере не являются ответом. Они показывают только формат.

Что нужно сдать

Нужно сдать три файла:

1. **05-task-02-full-gradient.csv**;
2. **05-task-02-online.csv**;
3. **05-task-02-modes.csv**.

При необходимости можно дополнительно приложить файл **notes.md**. Он предназначен только для замечаний о неоднозначностях, спорных случаях или технических проблемах. **notes.md** не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. наличие всех трех обязательных файлов;
2. точное совпадение заголовков CSV-файлов;
3. наличие всех идентификаторов **B1**-**B4**, **O1**-**O5**, **M1**-**M5**;
4. корректность векторов, дробей и булевых полей **0**/**1**;
5. принадлежность **mode_type** фиксированному словарю;
6. отсутствие лишних строк и лишних столбцов.

Как можно варьировать задание

Для разных вариантов можно менять:

1. списки градиентов отдельных объектов в части 1;
2. начальные значения, шаги и последовательности градиентов в части 2;
3. описания режимов и фиксированный словарь меток в части 3;
4. число строк в каждой части при сохранении тех же столбцов ответа.

06-task-01. Обратный проход по скалярному вычислительному графу

Цель

Научиться выполнять обратный проход по простому вычислительному графу и записывать сопряженные величины для узлов.

Материал лекции

Задание относится к лекции 6 «Автоматическое дифференцирование и обратное распространение ошибки» и опирается на понятия:

1. вычислительный граф;
2. локальная производная;
3. правило цепочки;
4. режим обратного прохода;
5. сопряженная величина узла;

Базовый вариант

Граф: **u = x*y, v = u + b, L = v^2. **Значения входных узлов:** x=2, y=-3, b=5. **Нужно заполнить значения всех узлов и сопряженные величины** dL/d node**.

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 06-task-01-reverse.csv

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
node_id,value,adjoint
```

Пример структуры:

```
node_id,value,adjoint
x,,
y,,
b,,
u,,
v,,
L,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **06-task-01-reverse.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. значения всех узлов графа;
2. сопряженные величины **dL/d node**;
3. строгий порядок строк по **node_id** из таблицы задания;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

07-task-01. Один шаг практических градиентных методов

Цель

Сравнить один шаг нескольких методов оптимизации на скалярных параметрах и фиксированных градиентах.

Материал лекции

Задание относится к лекции 7 «Практические методы градиентного обучения» и опирается на понятия:

1. метод тяжелого шарика;
2. метод Нестерова с моментом;
3. AdaGrad;
4. RMSprop;
5. Adam;

Базовый вариант

Используйте один шаг для каждого метода. Дроби считайте точно.

method_id	метод	параметры варианта
M1	heavy_ball	**w=1**, **eta=1/10**, **mu=1/2**, **v_prev=2**, **grad=2**; **v_new = mu v_prev + grad**, **w_next = w - eta v_new**
M2	nesterov	**w=43/40**, **eta=1/10**, итоговое направление **v_new=5/2**, **w_next = w - eta v_new**
M3	adagrad	**w=1**, **eta=3/10**, **grad=3**, **sum_sq_prev=0**; **sum_sq_new = sum_sq_prev + grad^2**, **w_next = w - eta grad / sqrt(sum_sq_new)**
M4	rmsprop	**w=3/10**, **eta=1/10**, **rho=9/10**, **s_prev=1**, **grad=3**; **s_new = rho s_prev + (1-rho) grad^2**, **w_next = w - eta grad**
M5	adam	**w=1**, **eta=1/10**, **beta_1=9/10**, **beta_2=99/100**, **m_prev=0**, **v_prev=0**, **grad=3**; используйте bias-correction для первого шага

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 07-task-01-updates.csv

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
method_id,method,w_next_num,w_next_den,state_a_num,state_a_den,state_b_num,state_b_den
```

Пример структуры:

```
method_id,method,w_next_num,w_next_den,state_a_num,state_a_den,state_b_num,state_b_den
M1,,,,,,,
```

```
M2,,,,,,,
M3,,,,,,,
M4,,,,,,,
M5,,,,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **07-task-01-updates.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. следующее значение параметра;
2. обновленное состояние метода;
3. несократимые дроби с положительным знаменателем;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

08-task-01. Планы шага обучения и масштабы инициализации

Цель

Вычислить значения нескольких планов шага обучения и масштабы Xavier/He-инициализации.

Материал лекции

Задание относится к лекции 8 «Настройка обучения и практический выбор оптимизатора» и опирается на понятия:

1. постоянный шаг;
2. ступенчатое уменьшение шага;
3. экспоненциальное уменьшение;
4. разогрев;
5. Xavier-инициализация;
6. He-инициализация;

Базовый вариант

Планы шага обучения:

schedule_id	правило	параметры
S1	постоянный шаг	**eta=1/10**
S2	ступенчатое уменьшение	**eta_0=1/10**, множитель **1/2**, выполнено 2 уменьшения
S3	экспоненциальное уменьшение	**eta_t = eta_0 gamma^t**, **eta_0=1/10**, **gamma=9/10**, **t=2**
S4	линейный разогрев	максимум **1/10**, текущий шаг разогрева **3** из **5**
S5	уменьшение при плато	**eta_0=1/10**, множитель **1/2**, одно плато

Инициализация:

init_id	метод	параметры
I1	Xavier	**fan_in=4**, **fan_out=8**
I2	Xavier	**fan_in=6**, **fan_out=10**
I3	He	**fan_in=128**
I4	He	**fan_in=256**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 08-task-01-schedules.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
schedule_id,eta_num,eta_den
```

Пример структуры:

```
schedule_id,eta_num,eta_den
S1,,
S2,,
S3,,
S4,,
S5,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **08-task-01-initialization.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
init_id,method,variance_num,variance_den
```

Пример структуры:

```
init_id,method,variance_num,variance_den
I1,,,
I2,,,
I3,,,
I4,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **08-task-01-schedules.csv**
2. **08-task-01-initialization.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. значение шага обучения на указанной эпохе;
2. дисперсия инициализации;
3. несократимые дроби;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

09-task-01. Размерности стандартных блоков

Цель

Определить выходные размерности линейного слоя, свертки, субдискретизации и эмбеддинга.

Материал лекции

Задание относится к лекции 9 «Стандартные вычислительные блоки нейросетей» и опирается на понятия:

1. линейный слой;
2. Conv1d;
3. Conv2d;
4. max pooling;
5. embedding;
6. размерность выхода;

Базовый вариант

Определите форму выхода без размерности batch.

block_id	блок	параметры
B1	linear	вход **(4;5)**, выходных признаков **3**, есть bias
B2	conv1d	**in_channels**=**3**, **out_channels**=**2**, **length**=**10**, **kernel**=**3**, **stride**=**1**, **padding**=**0**, есть bias
B3	conv2d	**in_channels**=**3**, **out_channels**=**6**, вход **32 x 32**, **kernel**=**5**, **stride**=**2**, **padding**=**2**, есть bias
B4	maxpool2d	вход **(3;16;16)**, **kernel**=**2**, **stride**=**2**
B5	embedding	длина последовательности **5**, размер эмбеддинга **4**, размер словаря **7**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 09-task-01-shapes.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
block_id,block_type,output_shape,parameter_count
```

Пример структуры:

```
block_id,block_type,output_shape,parameter_count
B1,,,
B2,,,
B3,,,
B4,,,
B5,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **09-task-01-shapes.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. форма выходного тензора;
2. число обучаемых параметров;
3. нулевое число параметров для pooling;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

10-task-01. Штрафы регуляризации и инвертированное dropout-масштабирование

Цель

Вычислить L1/L2-штрафы и проверить ожидаемое значение активаций при случайном выключении нейронов.

Материал лекции

Задание относится к лекции 10 «Методы регуляризации» и опирается на понятия:

1. L1-регуляризация;
2. L2-регуляризация;
3. случайное выключение нейронов;
4. инвертированное масштабирование;
5. добавление шума;

Базовый вариант

Штрафы регуляризации:

case_id	w	lambda_1	lambda_2
R1	**(1;-2)**	**1/2**	**7/10**
R2	**(3;-1;2)**	**3/8**	**3/16**
R3	**(1;-1;0;2)**	**1/2**	**1/3**

Dropout:

dropout_id	исходная активация	вероятность сохранения
D1	**6**	**3/4**
D2	**-2**	**1/2**
D3	**4**	**4/5**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 10-task-01-penalties.csv

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
case_id,l1_num,l1_den,l2_num,l2_den
```

Пример структуры:

```
case_id,l1_num,l1_den,l2_num,l2_den
R1,,,,
R2,,,,
R3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **10-task-01-dropout.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
dropout_id,keep_prob_num,keep_prob_den,scaled_active_value,expected_value
```

Пример структуры:

```
dropout_id,keep_prob_num,keep_prob_den,scaled_active_value,expected_value
D1,,,,
D2,,,,
D3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **10-task-01-penalties.csv**
2. **10-task-01-dropout.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. L1 и L2 штрафы;
2. масштаб активного нейрона;
3. сохранение математического ожидания;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

11-task-01. Таксономия и метрики генеративных моделей

Цель

Классифицировать модели обучения без учителя и вычислить простые агрегаты качества генеративной модели.

Материал лекции

Задание относится к лекции 11 «Обучение без учителя» и опирается на понятия:

1. скрытая переменная;
2. генеративная модель;
3. правдоподобие;
4. оценка Inception;
5. покрытие распределения;

Базовый вариант

Таксономия:

model_id	описание
U1	кодировщик переводит данные в скрытое представление без явного правдоподобия
U2	вероятностная генеративная модель задает **p(x
U3	генератор строит пример из скрытого кода, но не дает явного правдоподобия
U4	алгоритм кластеризации сопоставляет объекту скрытую метку

Метрики:

metric_id	правило	данные
M1	средний log-likelihood	**-2**, **-5/2**
M2	упрощенная оценка Inception	**3/2**
M3	покрытие кластеров	покрыто **5** из **6** кластеров

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 11-task-01-taxonomy.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
model_id,direction,is_generative,uses_likelihood
```

Пример структуры:

```
model_id,direction,is_generative,uses_likelihood
U1,,,
U2,,,
U3,,,
U4,,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **11-task-01-metrics.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
metric_id,value_num,value_den,better_when
```

Пример структуры:

```
metric_id,value_num,value_den,better_when
M1,,,
M2,,,
M3,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **11-task-01-taxonomy.csv**
2. **11-task-01-metrics.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. тип направления модели;
2. булевы признаки генеративности и правдоподобия;
3. простые дробные метрики;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

12-task-01. Двумерная свертка, дополнение границ и шаг

Цель

Вычислить размерность выхода и несколько элементов двумерной свертки для малых изображений.

Материал лекции

Задание относится к лекции 12 «Сверточные сети для обработки изображений» и опирается на понятия:

1. двумерная свертка;
2. ядро свертки;
3. дополнение границ;
4. шаг свертки;
5. рецептивное поле;

Базовый вариант

Используйте корреляционную форму свертки, принятую в практических библиотеках: ядро не переворачивается.

case_id	вход и ядро	padding	stride	какие ячейки записать
C1	вход **[[1,2,3], [4,5,6], [7,8,9]]**, ядро **[[1,0], [0,-1]]**	0	1	левую верхнюю, правую верхнюю, правую нижнюю
C2	вход **[[1,2], [3,4]]**, ядро **[[13/4,0], [0,-1]]**	1	1	левую верхнюю, центральную, правую нижнюю
C3	вход **[[1,2,3,4], [5,6,7,8],[9,10,11,12], [13,14,15,16]]**, ядро **[[1,0], [0,1]]**	0	2	левую верхнюю, правую верхнюю, правую нижнюю

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если

знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 12-task-01-conv2d.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
case_id,output_shape,top_left,center_or_next,bottom_right
```

Пример структуры:

```
case_id,output_shape,top_left,center_or_next,bottom_right
C1,,,,
C2,,,,
C3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **12-task-01-conv2d.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. формула размера выхода;
2. значения выбранных ячеек карты признаков;
3. учет padding и stride;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

13-task-01. Развертка рекуррентной сети во времени

Цель
Выполнить несколько шагов простой рекуррентной формулы и оценить множитель затухания градиента.

Материал лекции
Задание относится к лекции 13 «Рекуррентные сети» и опирается на понятия:

1. развертка во времени;
2. скрытое состояние;
3. рекуррентная матрица;
4. затухание градиента;
5. BPTT;

Базовый вариант
Скалярная рекуррентная формула: **h_t = a h_(t-1) + b x_t**.

seq_id	**a**	**b**	**h_0**	последовательность **x_1,x_2,x_3**
R1	**1/2**	**1**	**0**	**1, 1/2, 1/2**
R2	**-1/3**	**1**	**0**	**-1, 2/3, 1/3**
R3	**1**	**1**	**1**	**1, -2, 2**

Градиентный множитель для трех шагов равен **a^3**.

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 13-task-01-unroll.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
seq_id,h1,h2,h3,grad_multiplier_num,grad_multiplier_den
```

Пример структуры:

```
seq_id,h1,h2,h3,grad_multiplier_num,grad_multiplier_den
R1,,,,,
R2,,,,,
R3,,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **13-task-01-unroll.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. последовательные скрытые состояния;
2. произведение рекуррентных множителей;
3. дробный формат градиентного множителя;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

14-task-01. Сырые оценки самовнимания и причинная маска

Цель

Посчитать матрицу **QK^T**, определить разрешенные позиции при причинной маске и выбрать максимальную оценку.

Материал лекции

Задание относится к лекции 14 «Трансформеры» и опирается на понятия:

1. запрос;
2. ключ;
3. значение;
4. самовнимание;
5. причинная маска;
6. сложность внимания;

Базовый вариант

Векторы ключей: **K1=(1;0)**, **K2=(0;1)**, **K3=(1;0)**. Векторы запросов: **Q1=(1;0)**, **Q2=(0;2)**, **Q3=(1;1)**. Для **Qi** причинная маска разрешает только ключи **K1…Ki**. Softmax вычислять не нужно. Для сложности внимания используйте число пар **n^2** для **n=4,8,16**.

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 14-task-01-attention.csv

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
query_id,scores,allowed_count,best_key_id
```

Пример структуры:

```
query_id,scores,allowed_count,best_key_id
Q1,,,
Q2,,,
Q3,,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **14-task-01-complexity.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
length_id,token_count,pair_count
```

Пример структуры:

```
length_id,token_count,pair_count
L1,,
L2,,
L3,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **14-task-01-attention.csv**
2. **14-task-01-complexity.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. скалярные произведения запрос-ключ;
2. учет причинной маски;
3. квадратичный рост числа пар токенов;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

15-task-01. Уравнение Беллмана и один шаг Q-обучения

Цель
Вычислить значения по уравнению Беллмана и выполнить одно обновление Q-обучения.

Материал лекции
Задание относится к лекции 15 «Обучение с подкреплением» и опирается на понятия:

1. марковский процесс принятия решений;
2. функция ценности;
3. уравнение Беллмана;
4. Q-обучение;
5. дисконтирование;

Базовый вариант
Беллмановские значения:

state_id	правило
S1	**V = 1 + 3/4 * 8**
S2	**V = −1 + 1/2 * 8**
S3	терминальное состояние, **V=0**

Q-обучение использует **Q_next = Q + alpha * (target - Q)**.

update_id	**Q**	**alpha**	**target**
Q1	**1**	**1/2**	**5**
Q2	**1**	**1/2**	**-1/2**
Q3	**4**	**1/3**	**4**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 15-task-01-bellman.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
state_id,value_num,value_den
```

Пример структуры:

```
state_id,value_num,value_den
S1,,
S2,,
S3,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **15-task-01-qlearning.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
update_id,target_num,target_den,q_next_num,q_next_den
```

Пример структуры:

```
update_id,target_num,target_den,q_next_num,q_next_den
Q1,,,,
Q2,,,,
Q3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **15-task-01-bellman.csv**
2. **15-task-01-qlearning.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. целевое значение Беллмана;
2. одно Q-обновление;
3. несократимые дроби;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

16-task-01. Прямой процесс зашумления и цель предсказания шума

Цель
Вычислить параметры прямого диффузионного процесса и квадратичную ошибку предсказания шума.

Материал лекции
Задание относится к лекции 16 «Диффузионные модели» и опирается на понятия:

1. прямой процесс зашумления;
2. alpha;
3. накопленное alpha_bar;
4. предсказание шума;
5. обратный процесс;

Базовый вариант
Прямой процесс:

case_id	**alpha_bar_t**
D1	**9/10**
D2	**3/5**
D3	**1/4**

Для цели предсказания шума считайте MSE между истинным и предсказанным шумом.

loss_id	истинный шум	предсказанный шум
E1	**(1;0)**	**(4/5;-1/10)**
E2	**(0;1)**	**(1/2;1/2)**
E3	**(2;-1)**	**(2;-1)**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 16-task-01-forward.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
case_id,alpha_bar_num,alpha_bar_den,signal_var_num,signal_var_den,noise_var_num,noise_var_den
```

Пример структуры:

```
case_id,alpha_bar_num,alpha_bar_den,signal_var_num,signal_var_den,noise_var_num,noise_var_den
D1,,,,,,
D2,,,,,,
D3,,,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **16-task-01-noise-loss.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
loss_id,mse_num,mse_den
```

Пример структуры:

```
loss_id,mse_num,mse_den
E1,,
E2,,
E3,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать

Нужно сдать файлы:

1. **16-task-01-forward.csv**
2. **16-task-01-noise-loss.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки

Автопроверка может проверять:

1. накопленное значение alpha_bar;
2. доли дисперсии сигнала и шума;
3. MSE предсказанного шума;

Как можно варьировать задание

Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.

17-task-01. MAP-оценка и апостериорное предсказание

Цель
Связать гауссов априорный закон с L2-регуляризацией и вычислить простые апостериорные предсказательные агрегаты.

Материал лекции
Задание относится к лекции 17 «Байесовы нейронные сети» и опирается на понятия:

1. априорное распределение;
2. апостериорное распределение;
3. MAP-оценка;
4. апостериорное предсказательное распределение;
5. эпистемическая неопределенность;

Базовый вариант
MAP-часть использует цель **data_loss + lambda ||w||_2^2**.

case_id	**lambda**	**	
B1	**1/4**	**4**	**1/4**
B2	**1/2**	**5**	**1**
B3	**1/8**	**2**	**5/16**

Апостериорное предсказание считается по конечному набору предсказаний.

pred_id	предсказания
P1	**1, 2, 3**
P2	**-1, 0, 1**
P3	**1, 3/2, 2**

Все дроби в ответе записываются как несократимые дроби через числитель и знаменатель в отдельных столбцах либо как целые числа, если знаменатель не требуется форматом файла. Векторы записываются без пробелов, например **(1;-2;3)**.

Формат файла 17-task-01-map.csv
Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
case_id,lambda_num,lambda_den,objective_num,objective_den
```

Пример структуры:

```
case_id,lambda_num,lambda_den,objective_num,objective_den
B1,,,,
B2,,,,
B3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом. ### Формат файла **17-task-01-predictive.csv**

Файл должен содержать строку заголовка и строки для всех идентификаторов базового варианта. Столбцы идут строго в таком порядке:

```
pred_id,mean_num,mean_den,epistemic_var_num,epistemic_var_den
```

Пример структуры:

```
pred_id,mean_num,mean_den,epistemic_var_num,epistemic_var_den
P1,,,,
P2,,,,
P3,,,,
```

Числа в примере, кроме идентификаторов, не являются ответом.

Что нужно сдать
Нужно сдать файлы:

1. **17-task-01-map.csv**
2. **17-task-01-predictive.csv**

При необходимости можно дополнительно приложить **notes.md**. Этот файл не участвует в автопроверке.

Критерии автоматической проверки
Автопроверка может проверять:

1. коэффициент L2 из гауссова prior;
2. значение MAP-цели;
3. среднее и дисперсия предсказаний;

Как можно варьировать задание
Для разных вариантов можно менять числовые параметры базового варианта, число строк и идентификаторы случаев при сохранении тех же формул и того же формата CSV.