

## Задача 1:

Предположим, мы тренируем нейронную сеть с наличием  $L2$ -регуляризации. При повышении коэффициента регуляризации, мы ожидаем что точность предсказания после окончания тренировки (final accuracy) на тренировочном наборе (training set):

- не изменится или увеличится
- не изменится или уменьшится

## Решение:

Правильный ответ: «Не изменится или уменьшится»

Обоснование:

$L2$ -регуляризация добавляет штраф за большие веса модели (нормализует их), что снижает риск переобучения (overfitting).

При увеличении коэффициента регуляризации модель сильнее ограничивается в свободе подстройки под тренировочные данные.

В результате:

- Если регуляризация была слишком слабой, её усиление может улучшить обобщающую способность (accuracy на тестовых данных).
- Но на тренировочном наборе (training set) точность либо останется прежней (если модель уже была близка к оптимальной), либо уменьшится (из-за наложенных ограничений на веса).

Таким образом, на training set final accuracy не увеличится — возможны только нейтральный или отрицательный эффект.

## Задача 2:

Предположим, мы добавили Batch Normalization к сверточной сети, использующей ReLU в качестве активационной функции, которая ранее тренировалась без него. Мы ожидаем:

- значение функции ошибки во время тренировки будет уменьшаться быстрее
- значение функции ошибки во время тренировки будет уменьшаться медленнее

## Решение:

Правильный ответ: «Значение функции ошибки во время тренировки будет уменьшаться быстрее»

Обоснование:

Batch Normalization (BN) стабилизирует распределение входов каждого слоя, устраняя проблему internal covariate shift.

Для ReLU это особенно полезно:

- BN предотвращает "затухание" активаций (из-за отрицательных сдвигов), сохраняя активными больше нейронов.
- Градиенты становятся устойчивее, что ускоряет сходимость.

Следствия:

- Модель может использовать более высокий learning rate, не рискуя расходиться.
- Уменьшение ошибки происходит быстрее, так как оптимизация становится эффективнее.

Таким образом, добавление BN обычно ускоряет обучение и снижает loss на тренировочных данных быстрее, чем без него.

## Задача 3:

Нейронная сеть для задачи классификации симплов на 1000 классов заканчивается слоем Softmax

с 1000 выходами, по одному на класс. Сеть инициализировали случайными весами и пока не тренировали.

Приведите оценку среднего значения cross-entropy loss такой не тренированной сети на достаточно большом наборе данных. Напомним, среднее значение будет усреднение значения функции потерь для примеров в наборе данных.

## Решение:

Среднее значение cross-entropy loss для необученной сети с Softmax на 1000 классов можно оценить аналитически.

Рассуждение:

Softmax на необученной сети:

- Веса инициализированы случайно, поэтому выходы каждого нейрона перед Softmax (logits) — случайные числа.

- После Softmax все выходы — равномерное распределение (каждый класс имеет вероятность  $\sim \frac{1}{1000}$ ).

Формула cross-entropy loss:

$$L = -\frac{1}{N} \sum_{i=1}^N \log(p_{y_i}),$$

где  $p_{y_i}$  — вероятность правильного класса для  $i$ -го примера.

Оценка для равномерного распределения:

Если модель предсказывает  $p_{y_i} = \frac{1}{1000}$  для всех примеров, то:

$$L = -\log\left(\frac{1}{1000}\right) = \log(1000)$$

В ML используют натуральный логарифм или логарифм по основанию 2 (в битах). То есть  $\log$  — это  $\ln$  или  $\log_2$ .

Итоговый ответ:  $L = \log(1000)$ .

## Задача 4:

В статье Very Deep Convolutional Networks for Large-Scale Image Recognition описана архитектура VGG:

Эта архитектура получает на вход картинку 224x224.

- Сколько параметров в сумме у всех сверточных слоев варианта A? Не забудьте про параметры сдвига!

- Какое рецептивное поле (receptive field) у нейронов первого слоя conv3-256 варианта A?

## Решение:

Решение задачи об архитектуре VGG:

Количество параметров в сверточных слоях VGG-A.

Архитектура VGG-A (11 слоёв) содержит:

- 8 сверточных слоёв (чередуются conv3-64, conv3-128, conv3-256, conv3-512)
- 3 полносвязных слоя (не учитываем)

Формула для расчёта параметров одного сверточного слоя:

$$\text{Params} = (\text{input\_ch} * \text{kernel\_size}^2 * \text{output\_ch}) + \text{output\_ch}$$

Пошаговый расчёт:

1. **conv3-64** (3 входных канала  $\rightarrow$  64 фильтра):

$$(3 * 3^2 * 64) + 64 = 1,728 + 64 = 1,792$$

2. **conv3-128** (64  $\rightarrow$  128):

$$(64 * 3^2 * 128) + 128 = 73,728 + 128 = 73,856$$

3. **conv3-256** (128  $\rightarrow$  256):

$$(128 * 3^2 * 256) + 256 = 294,912 + 256 = 295,168$$

4. **conv3-512** (256  $\rightarrow$  512, 2 слоя):

$$2 * [(256 * 3^2 * 512) + 512] = 2 * (1,179,648 + 512) = 2,360,320$$

Итого параметров:

$$1,792 + 73,856 + 295,168 + 2,360,320 = \boxed{2,731,136}$$

Рецептивное поле первого conv3-256 в VGG-A:

Формула для рецептивного поля:

$$RF_{new} = RF_{prev} + (\text{kernel\_size} - 1) * \prod \text{previous\_strides}$$

Последовательность слоёв до первого conv3-256:

1.  $2 \times \text{conv3-64}$  (stride=1)
2. max-pool ( $2 \times 2$ , stride=2)
3.  $2 \times \text{conv3-128}$  (stride=1)
4. max-pool ( $2 \times 2$ , stride=2)
5. conv3-256 (первый слой)

Расчёт:

После  $2 \times \text{conv3-64}$ :  $RF = 3 + (3-1) * 1 = 5$

После max-pool:  $RF = 5 + (2-1) * 1 = 6$ , stride = 2

После  $2 \times \text{conv3-128}$ :  $RF = 6 + (3-1) * 2 = 10$

$10 + (3-1) * 2 = 14$

После max-pool:  $RF = 14 + (2-1) * 2 = 16$ , stride = 4

conv3-256:  $RF = 16 + (3-1) * 4 = \boxed{24}$

Альтернативный метод:

$$RF = 1 + \sum_{i=1}^L (k_i - 1) * \prod_{j=1}^{i-1} s_j$$

Для первого conv3-256 (6-й слой):

$$1 + 2 * 1 + 2 * 1 + 2 * 2 + 2 * 2 + 2 * 4 = \boxed{24}$$

Задача 5:

Пусть есть некая матрица  $Y$ , заданная как произведение двух других матриц:  $Y = X * W$ . Предположим, есть дифференцируемая функция  $f(Y)$ , которая в точке  $Y_0$  равна числу  $L$ . Выразите  $\nabla_X f$ , предполагая что  $\nabla_Y f$  в точке  $Y_0$  известен. Задача эквивалентна вычислению градиента во время обратного прохода через аффинную часть полносвязного слоя.

**Подсказка:** попробуйте расписать производную  $f$  по одному из элементов матрицы  $X$  через производные элементов матрицы  $Y$ :  $\frac{\partial f}{\partial x_{ij}} = ?$

## Решение:

Вычисление градиента  $\nabla_X f$  для  $Y = X * W$

Дано:

- Матрицы  $X \in R^{m \times n}$ ,  $W \in R^{n \times p}$
- $Y = X * W$ , где  $Y \in R^{m \times p}$
- Функция  $f(Y)$  дифференцируема, и в точке  $Y_0$  известно значение  $\nabla_Y f$

Решение:

- Поэлементное разложение:

Элементы матрицы  $Y$  выражаются как:

$$y_{ik} = \sum_{j=1}^n x_{ij} * w_{jk}$$

Производная  $f$  по элементу  $x_{ij}$ :

$$\frac{\partial f}{\partial x_{ij}} = \sum_{k=1}^p \frac{\partial f}{\partial y_{ik}} * \frac{\partial y_{ik}}{\partial x_{ij}} = \sum_{k=1}^p \frac{\partial f}{\partial y_{ik}} * w_{jk}$$

- Матричная форма:

Заметим, что:

$$\frac{\partial f}{\partial x_{ij}} = \sum_{k=1}^p \left( \frac{\partial f}{\partial Y} \right)_{ik} * (W^T)_{kj}$$

где  $W^T$  - транспонированная матрица  $W$ .

Это соответствует матричному произведению:

$$\nabla_X f = \frac{\partial f}{\partial X} = \left( \frac{\partial f}{\partial Y} \right) * W^T$$

- Проверка размерностей:

1.  $\frac{\partial f}{\partial Y} \in R^{m \times p}$
2.  $W^T \in R^{p \times n}$
3. Результат  $\nabla_X f \in R^{m \times n}$  (совпадает с размером  $X$ )

Ответ:  $\nabla_X f = (\nabla_Y f) * W^T$ .