

Задание 1. Свёрточные автокодировщики для улучшения качества классификации изображений

Курс: Практикум на ЭВМ, весна 2018

Начало выполнения задания: 17 февраля.

Мягкий дедлайн: **2 марта, 23:59.**

Жёсткий дедлайн: **16 марта, 23:59.**

Дата последнего обновления задания: 27 февраля 2018 г.

Формулировка задания

Данное задание направлено на ознакомление с нейронными сетями и концепцией transfer learning.

В задании необходимо:

1. Написать на языке Python собственную реализацию свёрточного автокодировщика с произвольным количеством слоёв. Прототипы функций должны строго соответствовать прототипам, описанным в спецификации и проходить все выданные тесты. Задание, не проходящее все выданные тесты, приравнивается к невыполненному. При написании необходимо пользоваться стандартными средствами языка Python, библиотеками numpy, scipy, pytorch, scikit-learn и matplotlib.
2. Провести описанные ниже эксперименты на датасете stl-10.
3. Написать отчёт о проделанной работе (формат PDF, подготовленный в системе L^AT_EX, или формат html, конвертированный из jupyter-notebook).
4. Сдать в систему anytask .zip архив с отчётом и jupyter-notebook с экспериментами (если есть дополнительные модули с кодом, их тоже нужно поместить в архив). Архив должен иметь название task1_фамилия_имя.zip

Многослойные нейронные сети

Предположим, что решается классическая задача обучения с учителем и в распоряжении имеется набор объектов вместе с соответствующими метками $X = \{x_i\}_{i=1}^n$, $y = \{y_i\}_{i=1}^n$.

Нейронные сети позволяют строить сложные нелинейные алгоритмы. Минимальная структурная единица нейронной сети — нейрон, «вычислительный узел», принимающий на вход значения x^1, \dots, x^n (а также константный единичный вход), а на выходе выдающий значение $a(x) = h(\sum_{i=1}^n w_n x_n + w_0)$, где функция $h : \mathbb{R} \rightarrow \mathbb{R}$ называется функцией активации.

Некоторые из функций активации:

- сигмоидная

$$h(z) = \frac{1}{1 + \exp(-z)}.$$

- тангенциальная

$$h(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- relu

$$h(z) = \max(0, x)$$

- leaky-relu

$$h(z) = \mathbb{I}[x < 0]0.01x + \mathbb{I}[x \geq 0]x$$

Проблемы первых двух функций активации заключаются в том, что при $x \rightarrow \infty$, функции слабо меняются, градиент на этих участках близок к нулю. В процессе обратного распространения ошибки локальный градиент умножается на общий градиент. Следовательно, если локальный градиент очень мал, он фактически обнуляет общий градиент. Функция `relu` лишена такого недостатка на положительной полуоси, а `leaky-relu` на всей оси.

Нейронная сеть в общем случае строится как соединение множества нейронов, объединенных в *слои* так, что выходы одного слоя являются входами следующего. Полносвязным слоем нейронной сети называется слой, в котором входами каждого нейрона являются выходы всех нейронов с предыдущего слоя. Самый левый слой сети называется входным, самый правый — выходным (обычно он состоит из одного нейрона, итоговой функции потерь), остальные слои называют скрытыми, потому что их «правильные» значения отсутствуют в обучающем наборе.

Традиционная функция потерь для задачи классификации — кросс-энтропия. При обучении сети часто используется l2-регуляризация всех весов сети, в `pytorch` коэффициент l2-регуляризации задаётся с помощью параметра `weight_decay` алгоритма оптимизации. Веса сети обучаются с помощью алгоритма обратного распространения ошибки (back propagation).

Линейные автокодировщики

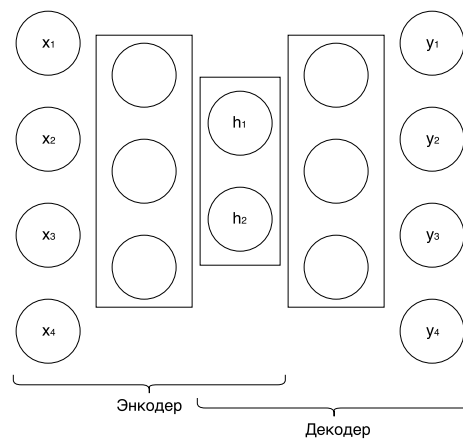


Рис. 1: Структура автокодировщика

Пусть помимо обучающей размеченной выборки имеется неразмеченная выборка \hat{X} , $|\hat{X}| \gg |X|$. Логично, попытаться использовать эту выборку для повышения качества решения исходной задачи.

Автокодировщиком (автоэнкодером) называется нейросеть, целевой вектор которой полагается равным её входу, т.е. $y^{(i)} = x^{(i)}$. Автокодировщик тем самым строит приближение функции $J(x) \approx x$, т.е. фактически приближение для тождественной функции. Хотя тождественная функция и не выглядит разумной целью для аппроксимации сама по себе, обучая её мы получаем возможность перевода исходных объектов в новое признаковое пространство (возможно, гораздо меньшей размерности). В качестве признаков в новом пространстве выступают значения активации нейронов одного из внутренних слоёв. Часть автокодировщика, преобразующая исходный объект в новое представление, называется энкодером, часть, преобразующая объект из нового представления в исходное, декодером. Типичная функция потерь для автокодировщика — MSE. Автокодировщик не обязан быть полностью симметричным, важна лишь одинаковая размерность входного и выходного слоёв.

Таким образом, автокодировщик можно использовать для генерации признаков объектов. Также веса автокодировщика можно использовать для инициализации части весов сети, решающей задачу классификации, такая инициализация в большинстве случаев будет лучше случайной.

Накладывая ограничения на веса автокодировщика, можно обнаруживать скрытые закономерности в данных. Один из вариантов: наложить ограничение на среднюю (по объектам) величину активации нейронов скрытого слоя.

А именно, если под активным нейроном понимать нейрон со значением функции активации, близким к 1, а под неактивным — близким к 0 (для случая сигмоидной функции активации), то можно наложить ограничение, при котором каждый из нейронов большую часть времени был бы неактивен. Пусть $\rho_{ij} = h(\sum_{i=1}^n W_{ji}x_i + w_0)$. Обозначим, $\rho_j = \frac{1}{N} \sum_{i=1}^N \rho_{ij}$. Будем требовать приближенного выполнения ограничения $\rho_j = \rho$, где ρ — параметр, отвечающий за разреженность, обычно достаточно малая величина, близкая к 0 (порядка 0.05). Чтобы достичь этого, мы добавим дополнительно ограничение для функции потерь нашей нейросети, а именно добавим

такое слагаемое:

$$\sum_{j=1}^h \rho \log \frac{\rho}{\rho_j} + (1 - \rho) \log \frac{(1 - \rho)}{(1 - \rho_j)}$$

Такая регуляризация возможна только в случае, когда модель обучается полным градиентным спуском либо стохастическим по батчам достаточно большого размера.

Фильтры для обработки изображений

Прежде чем говорить о специальных нейросетевых архитектурах для изображений, необходимо немного вспомнить о классических методах обработки изображений. Напомним, что изображение в общем случае представляется в памяти компьютера как трёхмерный тензор (RGB) или как матрица (оттенки серого). Для выделения признаков применяются фильтры — специальные преобразования, позволяющие обнаружить участки изображений, обладающие некоторыми особыми свойствами. Например, фильтр Собеля позволяет выделять края объектов на изображении:

$$G = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Фильтр $G \in \mathbb{R}^{2s_1+1 \times 2s_2+1}$ применяется к изображению $x \in \mathbb{R}^{a \times b}$ с помощью операции свёртки. Результат свёртки можно записать так:

$$F(i, j) = \sum_{k_1=-s_1}^{s_1} \sum_{k_2=-s_2}^{s_2} x_{i+k_1, j+k_2} G_{k_1, k_2}$$

Аналогично, можно записать применение фильтров чётных размерностей и трёхмерных фильтров к трёхмерным изображениям. Есть некоторая вариативность в применении операции свёртки (обработка краёв, изменение "шага" фильтра). Исходное изображение может дополняться с разных сторон нулевыми пикселями (padding) (например, для выравнивания размера изображений). Свёртка может применяться не в каждой точке (i, j) изображения, а, например, через одну (stride).

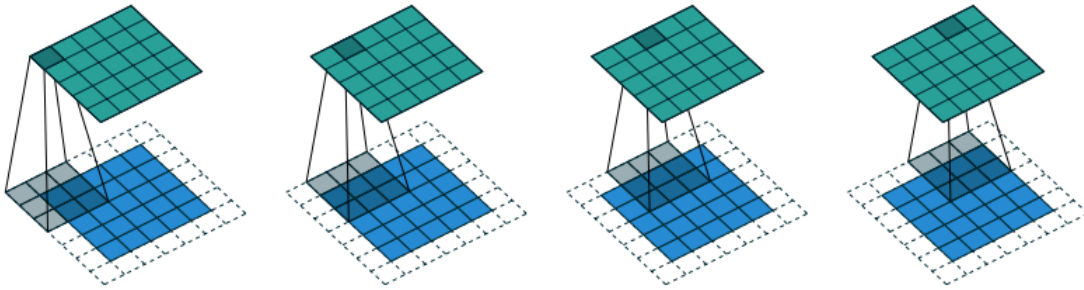


Рис. 2: Пример использования свёртки с $s_1 = s_2 = 3, a = b = 5, \text{padding} = 1, \text{stride} = 0$

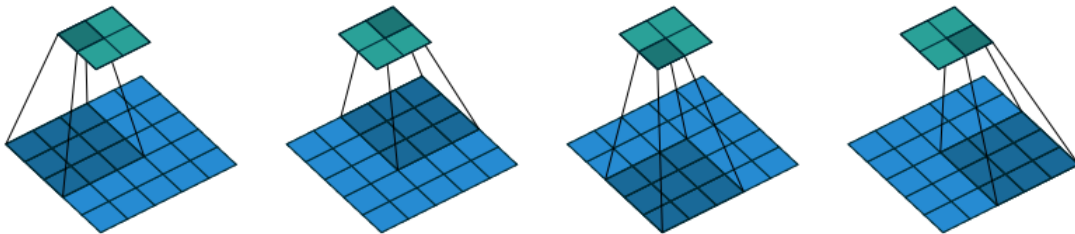


Рис. 3: Пример использования свёртки с $s_1 = s_2 = 3, a = b = 5, \text{padding} = 0, \text{stride} = 2$

Свёрточные нейронные сети

В классических методах обработки изображений исследователь должен был формировать различные фильтры для выделения признаков в своей задаче. Современный подход, лежащий в основе архитектуры свёрточных нейронных сетей (convolutional networks) — обучение фильтров под решаемую задачу. В свёрточных нейронных сетях появляются два новых типа слоя — свёрточный слой (convolutional layer) и pooling-слой.

Свёрточный слой — это свёртка нескольких фильтров с выходами предыдущего слоя. Веса фильтров обучаются как и обычные полносвязные слои в ходе алгоритма back-propagation.

Pooling-слой нужен для того, чтобы «сжать» изображение (рис. 5). Результат применения max-pooling слоя размера $2s_1 + 1 \times 2s_2 + 1$ можно записать так:

$$F(i, j) = \max(x_{i+k_1, j+k_2} | k_1 \in \{-s_1, \dots, s_1\}, k_2 \in \{-s_2, \dots, s_2\})$$

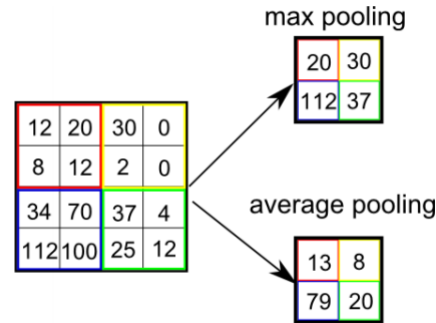


Рис. 4: Пример применения max-pooling и average-pooling слоёв, stride=2

Как и в случае свёрточного слоя, pooling-слой может применять с padding и stride. Обычно, pooling-слой применяется сразу за свёрточным слоем. Pooling-слой не требует обучения, но влияет на обучение других слоёв. Классическая архитектура свёрточной сети для решения задачи классификации — несколько свёрточных блоков (свёрточный слой + функция активации + pooling-слой) и полносвязный слой в конце.

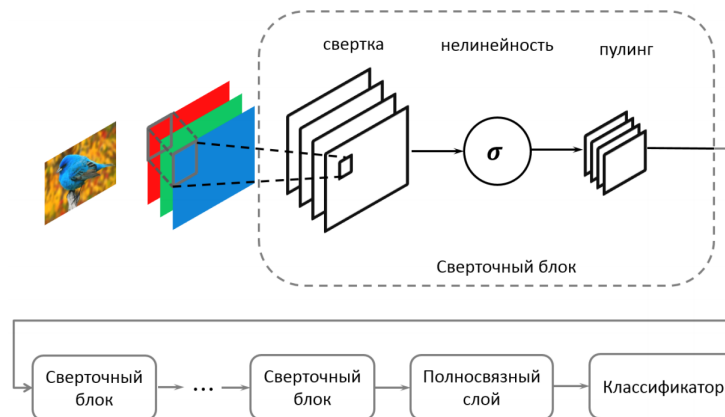


Рис. 5: Классическая архитектура свёрточной сети

Свёрточные автокодировщики

Свёрточный автокодировщик, как и линейный, старается построить приближение тождественной функции $J(x) \approx x$, но использует для этого свёрточные, pooling-слои, а также два новых слоя — convolutional-transposed layer и upsampling layer. Нетрудно догадаться, что два новых типа слоя должны быть в некотором смысле обратны свёрточному и pooling слоям соответственно.

Convolutional-transposed слой применяет свёрточный слой к изображению с большим padding. В случае convolutional-transposed padding понимается «наоборот», чем больше padding, тем меньшим количеством нулей будет дополняться изображение перед свёрткой.

Upsampling слой восстанавливает исходное изображение после применения max-pooling. Есть несколько типов таких слоёв, самый простой из них работает следующим образом. Если $f(i, j)$ результат предыдущего (не обязательно прямо перед upsampling слоем) max-pooling, то результат upsampling, применённый к изображению \hat{x} , будет такой:

$$\phi(i, j) = \begin{cases} \hat{x}_{i,j}, & \text{если } f(i, j) = x_{i,j} \\ 0, & \text{иначе} \end{cases}$$

Т.е. для unpooling слоя требуется дополнительно хранить индексы максимальных элементов.

Свёрточный автокодировщик в общем случае состоит из следующей последовательности слоёв:

1. Несколько свёрточных блоков
2. Несколько обратных свёрточных блоков (upsampling + convolutional-transposed + активация)
3. Функция потерь MSE

Требования реализации

Требуется реализовать класс для свёрточного автокодировщика ConvAutoEncoder и класс для свёрточной сети ConvNet. Спецификацию классов необходимо выбрать самостоятельно.

Замечание. Рекомендуются придумать спецификацию, которая позволит легко менять параметры сети. Например, для автокодировщика подойдёт такая спецификация:

```
class ConvAutoEncoder(nn.Module):
    def __init__(self,
                  input_size=(3, 32, 32),
                  layers_num=2,
                  conv_out_channels=(6, 6),
                  conv_kernel_size=(2, 3),
                  conv_stride=(2, 2),
                  pool_kernel_size=(2, 1),
                  pool_stride=(1, 1))
```

Полезные функции из библиотеки pytorch:

```
nn.Conv2d, nn.MaxPool2d, nn.Linear, nn.ConvTranspose2d, nn.MaxUnpool2d, nn.MSELoss,
nn.CrossEntropyLoss, optim.SGD, optim.Adam, nn.functional.relu, nn.functional.tanh
```

Замечание. Перед применением кросс-энтропии не нужно делать предварительной softmax активации

Исследовательская часть.

Все эксперименты в этом задании проводятся на датасете stl-10. Скачать датасет можно с помощью команды `torchvision.datasets.STL10`. Необходимо скачать три части датасета: неразмеченную **unsupervised**, размеченные **train** и **test**. Рекомендуются уменьшить изображения перед процедурами обучения (например, до размера 32×32). Значения пикселей всех изображений перед подачей в сеть необходимо перевести в отрезок $[-1, 1]$.

Требуется провести следующие исследования:

1. Протестируйте на train/test выборках мультиномиальную логистическую регрессию (можно реализовать как однослойную нейронную сеть) и какой-нибудь из методов, основанных на деревьях. В дальнейшем, используйте полученное качество (точность, log loss) в качестве бейзлайна.
2. Реализуйте небольшую свёрточную сеть, работающую на train/test выборке. В качестве функции потерь используйте кросс-энтропию. Минимальный размер сети: 1 свёрточный слой, 1 pooling слой, 1 полносвязный слой. Проведите исследования, как влияют на качество и скорость работы сети:
 - размер ядра свёртки (рекомендуется брать небольшие значения, от 2 до 8)
 - количество фильтров на свёрточном слое (достаточно больше значения, от 5 до 40)
 - количество свёрточных блоков (свёрточный слой + pooling) в сети (1, 2 и больше)
 - стратегий использования momentum в методе SGD
 - (опционально) использования специальных методов оптимизации для нейросетей (Adam, Adagrad ...)

Замечание. Обязательно используйте при обучении l2 регуляризацию!

3. Реализуйте однослойный (один слой на энкодер, один слой на декодер) свёрточный автокодировщик, обучающийся только по unsupervised части датасета. Подберите параметры, при которых автокодировщик определён корректно (размерность выхода равна размерности входа). Визуализируйте выход автокодировщика для нескольких изображений. Подберите параметры слоёв (padding, strides, функции потерь), при которых достигается хорошее качество работы автокодировщика (низкое значение MSE).

Замечание. Так как элементы входных массивов находятся в отрезке $[-1, 1]$, элементы на выходе находятся в таких же границах.

4. Используйте признаки, выделенные с помощью автоэнкодера, как признаки для моделей из первого пункта. Сравните результаты, полученные после обучения на разных признаковых пространствах, проанализируйте результаты (какой модели преобразование лучше помогло и почему). Сравните результаты обучения с нейросетевой моделью из второго пункта, проанализируйте, помогли ли неразмеченные данные улучшить качество классификации.
5. Попробуйте модифицировать процесс обучения нейросети. После обучения автокодировщика, используйте параметры энкодера в качестве первых слоёв новой сети. Обучите сеть, добучая в том числе и параметры автокодировщика. Проанализируйте, помогли ли неразмеченные данные улучшить качество классификации.

Бонусная часть

1. (до 5 баллов) Используйте автокодировщик из нескольких слоёв (2 и более). Проанализируйте, можно ли с помощью такой архитектуры улучшить качество работы автокодировщика.

Замечание. Если не удаётся обучить несколько слоёв в единой архитектуре, обучите несколько stacked-автоэнкодеров.

2. (до 10 баллов) Реализуйте архитектуру разреженного линейного автокодировщика. Из исходного датасета stl-10 вырежьте небольшие цельные фрагменты размера $3 \times 8 \times 8$, обучите автокодировщик на этих фрагментах. Визуализируйте средние активации нейронов на скрытом слое, сделайте выводы об их форме.
3. (до 5 баллов) Проведите 5 эксперимент основного задания с линейным автокодировщиком. Для получения нового признакового описания, используйте свёртку с фильтрами, обученными линейным автокодировщиком.