Искусственные нейронные сети

Приведите пример задачи, которая может быть решена обучением без учителя

Как правило, пригодна только для задач, в которых известны описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Обучение без учителя может применено к следующим задачам:

1. Кластеризация;
   Данная задача решалась на втором практическом занятии: бинарная классификация множеств точек. Очевидно, можно обобщить до многолассовой классификации
2. Обобщения;
3. Поиск правил ассоциации;
   Среди представленных наборов данных выделить наборы признаков и их значения, встречающие в данных ноборах наиболее часто.
4. Сокращение размерности;
   У нас есть множество параметров, описывающих какие-то объекты. При этом часть параметров являются определяющими, часть оказывают влияние в меньшей мере. Обучением без учителя можно решить задачу сокращения размерности исходных параметров минимизируя при этом потери.
5. Визуализации данных;
   Преобразование многомерных данных к размерности два. Это позволяет визуализировать данные, что способствует лучшему пониманию данных с сути решаемой задачи.

Что такое карта признаков?

В соответствии с лекцией по свёрточным нейронным сетям:

> Карта признаков - это набор нейронов, каждый из которых имеет тем большее значение, чем больша связанный с ним фрагмент изображения похож на ядро.

На каждую свёртку приходится одна карта признаков.

> Каждый свёрточный уровень преобразует предыдущую карту признаков в несколько последующих признаков.

Получается, что CNN, чередуя слои свёртки и субдескритизации, приобразует исходное изображение в карту признаков, которую можно преобразовать в скаляр или вектор, а затем скормить полносвязной нейронной сети для дальнейшей обработки, например, для

задачи классификации карты признаков на 10 классов цифр, как в данной лабораторной работе.

Что такое субдискретизирующий слой?

Субдискретезирующий слой - слой пуллинга.

Слой пуллинга по своей сути является просто сжимающим изображение слоем. Исходя ис предположения, что соседние пиксели на изображении должны отличаться не сильно, можно сжать это изображение таким образом, что блоку пикселей будет соответствовать один пиксель среднего цвета.

> Линейный субдискретезирующий слой S_k, заключающий в себе принцип пространственной субдискретизации, понижает размерность карт признаков, используя технику усреднения значений соседних пикселей. Для этого отображаемая карта признаков разбивается на равные, непересекающиеся области. Формула отображения карты признаков Y_i^{k-1} на карту признаков Y_j^k выглядит следующим образом:
>
> $$y_j^k(x,y) = \varphi(w_{ij}^k \sum_{(u_x,u_y)\in U} y_i^{k-1}(sx+u_x, sy+u_y))$$
>
> где k - коэффициент децимации, $U = (1 \dots s)x(1 \dots s)$. При моделировании субдискретезирубщего слоя часто используются следующие параметры:
>
> $$s = 2; w_{ij}^k = \frac{1}{s^2} = const$$

Наиболее эффективным способом субдискретизации является ***max-pooling***, использующийся для обучения СНС. Формула отображения карт признаков при этом:

$$y_j^k(x,y) = \max_{(u_x,u_y)\in U} (y_i^{k-1}(sx+u_x, sy+u_y))$$

За оформление отчета минус балл (по сути он отсуствует)

Насколько я понимаю, эти отчёты всё равно никто печатать не будет, и пишем мы их номинально, поэтому я воспользовался форматом оформления наиболее удобным для меня. Я переживаю за ограниченные ресурсы нашей планеты: так или иначе бесполезное увеличение объёма отчёта за счёт использования .docx формата влечёт, как следствие, повышенную нагрузку на интернет, перерасход электрической энергии, ускоряет выход из строя накопителей серверов github-a. В конечном счёте это приводит у ускорению истощения планеты Земля.

В своём отчёте я указал ровно столько информации, сколько бы указал при использовании стандартного оформления. На мой взгляд, нельзя говорить о том, что "по сути он отсутствует". Так или иначе я переписал этот отчёт в формате .docx вне зависимости от того, повлияет ли это на что-то.

Можно ли обучать CNN с ненормированными данными?

При нормировании данных необходимо получить значения со средним значением равным нулю и дисперсией равной единице.

Делается это для того, чтобы стабилизировать и ускорить сетевое обучение с помощью спуска градиента. Если данные не нормированны, то в процессе обучения потери могут быть вычислены как NaN.

Проще говоря, CNN с ненормированными данными обучать можно. Но при этом возможны серьёзные проблемы со скоростью обучения и точностью обученной модели.

В чем разница методов Adamax и Adagrad?

Оба относятся к методам градиентного спуска. Каждый оптимизатор или семейство оптимизатор использует определённые ухищрения для увеличения точности или скорости обучения модели.

Adagrad

Adagrad - adaptive gradient. Редко встречающиеся входные параметры, преобразуемые в признак только при прохождении через несколько слоёв нейронной сети могут быть определены более успешно с использование оптимизации Adagrad. Для каждого параметра сети, она хранит сумму квадратов его обновлений, что позволяет определяет типичность этого параметра, поскольку появляется возможность обнаруживать то, как часто данный параметр приводит к обновлению нейрона. Формула обновления для данного оптимизатора выглядит следующим образом:

$$G_t = G_t + g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

, где G_t - сумма квадратов обновлений, а ϵ - сглаживающий параметр, необходимый для исключения деления на 0. Обычно ϵ берут равным 10^{-6} или 10^{-8}

Таким образом, идея семества Adagrad в том, чтобы использовать ***что-то***, что позволит уменьшить обновления для элементов, которые и так обновляются часто.

Достоиством Adagrad является отсутствие необходимости подбирать скорость обучения. Достаточно выбрать её в меру большой.

Adamax

Adamax - модификация алгоритма Adam (adaptive moment estimation). Метод совмещает в себе идею накопления движения и идею более слабого обновления весов для типичных признаков.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Для определения того, как часто изменяет градиент авторы предложили оценивать инерционный момент распределения градиентов произвольной степени p (формула накопления импульса Нестерова):

$$v_t = \sqrt[p]{\beta_2^p v_{t-1} + (1 - \beta_2^p)|g_t|^p}$$

Искусственно увеличиваем значения параметров m_t и v_t на первых шагах обучения:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Итоговое правило обновления выглядит следующим образом:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

Почему выбрана именно такая архитектура сети?

Не в полной мере понимаю данный вопрос. Архитектура выбирается эмпирически исходя из необходимости максимизовать точность сети, увеличить скорость обучения, уменьшить объём используемой памяти. Очевидно, при таком количестве параметров нужно искать консенсус.

В данной работе изображения являются небольшими по размеру (28 на 28 пикселей), количество изображения тоже является не самым большим, поэтому объём требуемой памяти корректировать особой необходимости нет.

Обучение модели из исходного примера на моём процессоре занимает порядка 10 секунд, что я считаю вполне вменяемым.

Минимальной удовлетворительной точностью модели является 95%. Собственно, выбранная архитектура позволяет получить порядка 97%-98%.

Стандартная конфигурация сети справляется со всеми требованиями, поэтому и была оставлена без изменений. В случае, если бы исходные изображения были большего размера, я бы производил предварительное сжатие вместо увеличения `input_shape`. Если бы модель обучалась слишком медленно, добавил бы слои `Dropout`, чтобы прорядить число нейронов в сети. Если бы точность была низкая, то как и в третьей лабораторной работе, исследовал бы влияние количества нейронов на скрытых слоях, при необходимости добавил бы второй скрытый слой.