Реферат по статье: «Contrastive Adaptation Network for Unsupervised Domain Adaptation»

https://arxiv.org/pdf/1901.00976.pdf

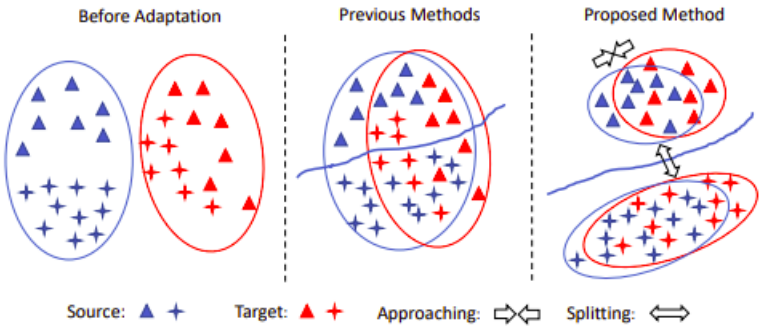
Чиж Ярослав

1 Введение

Задача Unsupervised Domain Adaptation (UDA) заключается в том, чтобы перенести имеющиеся знания об исходном домене с множеством размеченных примеров на целевой домен, для которого нет никакой разметки. Рассмотрим следующую задачу UDA в контексте данного реферата: есть два домена (source — исходный, target — целевой; например, графические модели предметов и их реальные фото) с изображениями нескольких классов; требуется обучить классификатор, который будет хорошо работать на target домене, но при условии, что метки классов есть только для source домена.

2 Проблема предыдущих методов

Проблема предыдущих методов, решающих поставленную выше задачу, заключается в том, что они измеряют несоответствие доменов (в фунцкии потерь) только на уровне самих доменов, игнорируя то, к каким классам принадлежат объекты. Рассмотрим следующий рисунок для более четкого понимания:



Примечание: разные домены помечены разными цветами, разные классы помечены треугольниками и крестиками.

Слева мы видим, как располагаются объекты из разных доменов в признаковом пространстве до того, как сделана какая-либо доменная адаптация.

В центре показано расположение доменов при применении какого-либо из предшествующих методов доменной адаптации. Адаптация без учета классовых меток приводит к тому, что объекты из target домена одного класса плохо совместимы с объектами того же класса из source домена. Поэтому решающее правило классификатора, обученного на исходном домене, плохо подходит для разделения классов целевого домена.

Справа приведено изображение доменов при применении предложенного в статье метода Contrastive Adaptation Network (CAN). В данном методе минимизируется внутриклассовое несоответствие до-

менов, а также максимизируется межклассовое несоответствие, чтобы повысить обобщающую способность модели.

3 Предложенный метод

Чтобы достичь описанных выше целей, вводится функция потерь Contrastive Domain Discrepancy (CDD). Введем следующие обозначения:

$$\mu_{cc'}(y, y') = \begin{cases} 1, \text{ если } y = c \text{ и } y' = c' \\ 0, \text{ иначе} \end{cases}$$

x_i^s — i-ый объект из source домена, x_j^t — j-ый объект из target домена, $\phi(x)$ — выходное значение нейронной сети на входе x (значение эмбеддинга изображения), $k(a, b)$ — некоторая ядровая функция, y_i^s — класс i-го объекта из source домена, $\hat{y}_j^t$ — предсказанный класс j-го объекта из target домена, n_s — кол-во рассматриваемых объектов из source домена, n_t — из target домена. Тогда CDD можно записать следующим образом:

$$D^{cdd} = \frac{1}{M}\sum_{c=1}^{M} D^{cc}(\hat{y}_{1:n_t}^t, \phi) - \frac{1}{M(M-1)}\sum_{c=1}^{M}\sum_{c'=1}^{M} D^{cc'}(\hat{y}_{1:n_t}^t, \phi);\ D^{c_1c_2}(\hat{y}_{1:n_t}^t, \phi) = e_1 + e_2 - 2e_3$$

где

$$e_1 = \frac{\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\mu_{c_1c_1}(y_i^s, y_j^s)k(\phi(x_i^s), \phi(x_j^s))}{\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\mu_{c_1c_1}(y_i^s, y_j^s)}$$

$$e_2 = \frac{\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\mu_{c_2c_2}(\hat{y}_i^t, \hat{y}_j^t)k(\phi(x_i^t), \phi(x_j^t))}{\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\mu_{c_2c_2}(\hat{y}_i^t, \hat{y}_j^t)}$$

$$e_3 = \frac{\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\mu_{c_1c_2}(y_i^s, \hat{y}_j^t)k(\phi(x_i^s), \phi(x_j^t))}{\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\mu_{c_1c_2}(y_i^s, \hat{y}_j^t)}$$

Заметим, что во время обучения у нас нет истинных меток классов для target домена, поэтому эти метки $\hat{y}_j^t$ предсказываются следующим образом: объекты source домена прогоняются через нейронную сеть и для каждого класса считается его центр (по полученным эмбеддингам). Далее объекты target домена тоже прогоняются через нейросеть и над полученными эмбеддингами выполняется алгоритм кластеризации Spherical K-means, центры в котором инициализируются центрами классов для source домена. Итого, каждый объект target домена принадлежит кластеру, центр которого имеет определенный класс — этот класс и является предсказанной меткой $\hat{y}_j^t$ для j-го объекта target домена. Также для большей стабильности всего алгоритма далее производится фильтрация «неуверенно» кластеризованных объектов и классов по следующим правилам:

- Если расстояние от объекта до центра его кластера больше некоторой константы D_0 (она подбирается для конкретного датасета), то объект исключается из рассмотрения на текущей итерации алгоритма
- После ислючения «неуверенно» кластеризованных объектов исключаются целые классы объектов: если в классе меньше N_0 объектов (данный параметр тоже подбирается под конкретный датасет), то класс исключается из рассмотрения на текущей итерации алгоритма

Итого, весь предложенный алгоритм состоит из цикла по «эпохам», каждая из которых содержит следующие шаги:

1. Получить метки классов для target домена с помощью описанной выше кластеризации
2. В течение K итераций (данный параметр тоже можно настраивать):

(a) Семплировать объекты из source и target доменов с учетом исключенных объектов и классов, а также посчитать CDD функцию потерь D^{cdd} для данных объектов

(b) Семплировать объекты из source домена и посчитать для них кроссэнтропию l^{ce}

(c) применить метод обратного распространения ошибки для итоговой функции потерь, которая имеет вид $l = l^{ce} + \beta D^{cdd}$, и обновить параметры нейронной сети

4 Результаты

Предложенный алгоритм показал state-of-the-art результаты на датасетах Office-31 и VisDA-2017. Чтобы воспроизвести результаты данной статьи самостоятельно, можно воспользоваться ее имплементацией на PyTorch по следующей ссылке: Contrastive Adaptation Network.