Московский государственный технический университет им. Н.Э.

Баумана

Кафедра «Системы обработки информации и управления»

Домашнее Задание №1

по дисциплине

«Методы машинного обучения»

Выполнил:

студент группы ИУ5-24М

Чжоу Хан

Москва — 2022 г.

Задание

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

- выбор задачи;
- теоретический этап;
- практический этап.

задачи

Выбранная задача: «Анализ настроения».

Выбранные статьи для анализа:

- Convolutional Neural Networks for Sentence Classification;
- Character-level Convolutional Networks for Text Classification.

Теоретическая часть

Статья 1:

Мы сообщаем о серии экспериментов со сверточными нейронными сетями (CNN), обученными поверх предварительно обученных векторов слов для задач классификации на уровне

предложений. Мы показываем, что простая CNN с небольшим количеством настроек гиперпараметров и статическими векторами достигает отличных результатов на нескольких эталонах. Обучение векторов, специфичных для конкретной задачи, с помощью тонкой настройки обеспечивает дальнейшее увеличение производительности. Кроме того, мы предлагаем простую модификацию архитектуры, позволяющую использовать как специфические для конкретной задачи, так и статические векторы. Рассмотренные здесь модели CNN улучшают уровень техники в 4 из 7 задач, которые включают анализ настроения и классификацию вопросов.

Архитектура модели, показанная на рисунке 1, представляет собой небольшой вариант архитектуры CNN Коллобера и др. (2011). Пусть $x_i \in \mathbb{R}^k$ - k-мерный вектор слов, соответствующий i-му слову в предложении. Предложение длины n (при необходимости с добавлением прокладок) представляется как

$$x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n, \quad (1)$$

где $\oplus$ - оператор конкатенации. В общем случае, пусть $x_{i:i+j}$ означает конкатенацию слов x_i , $x_{i+1}, \ldots, x_{i+j}$. Операция свертки

включает фильтр w ∈ R hk, который применяется к окну из h слов для получения новой характеристики. Например, признак ci генерируется из окна слов xi:i+h-1 следующим образом

$$ci = f(w \cdot xi:i+h-1 + b). \quad (2)$$

Здесь b ∈ R - член смещения, а f - нелинейная функция, например, гиперболический тангенс. Этот фильтр применяется к каждому возможному окну слов в предложении {x1:h, x2:h+1, . . . , xn-h+1:n} для получения карты признаков

$$c = [c1, c2, \ldots, cn-h+1], \quad (3)$$

c c ∈ R n-h+1. Затем мы применяем операцию объединения с максимальным временем (Collobert et al., 2011) к карте признаков и берем максимальное значение cˆ = max{c} в качестве признака, соответствующего данному конкретному фильтру. Идея заключается в том, чтобы захватить наиболее важную особенность - одну с наибольшим значением - для каждой карты признаков. Эта схема объединения естественным образом справляется с переменной длиной предложения. Мы описали процесс, в котором одна характеристика извлекается из одного фильтра. Модель использует несколько фильтров (с разными размерами окна) для получения нескольких признаков. Эти признаки формируют

предпоследний слой и передаются в полностью подключенный слой softmax, выходом которого является распределение вероятностей по меткам. В одном из вариантов модели мы экспериментируем с двумя "каналами" векторов слов, один из которых остается статичным на протяжении всего обучения, а другой настраивается с помощью обратного распространения (раздел 3.2).2 В многоканальной архитектуре, показанной на рисунке 1, каждый фильтр применяется к обоим каналам, а результаты складываются для вычисления ci в уравнении (2). В остальном модель эквивалентна одноканальной архитектуре.

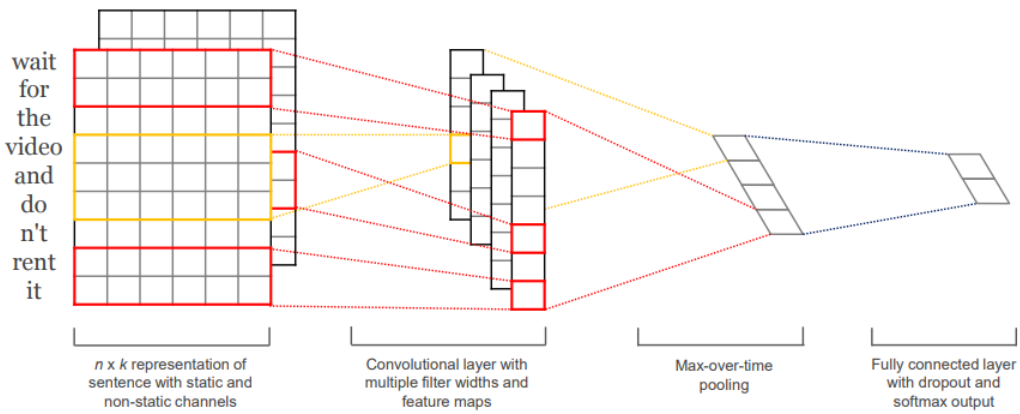

Рисунок 1: Архитектура модели с двумя каналами для примера предложения.

Для регуляризации мы используем отсев на предпоследнем слое с ограничением на l2-нормы весовых векторов (Hinton et al., 2012). Отсев предотвращает совместную адаптацию скрытых блоков путем случайного отбрасывания - т.е. установки на ноль - доли p скрытых блоков во время обратного распространения. То есть, учитывая предпоследний слой z = [ˆc1, . . . , cˆm] (заметим, что здесь у нас m фильтров), вместо того, чтобы использовать

$$y = w \cdot z + b \quad (4)$$

для выходной единицы y при прямом распространении, отсев использует

$$y = w \cdot (z \circ r) + b, \quad (5)$$

где ◦ - оператор умножения по элементам, а $r \in Rm$ - "маскирующий" вектор случайных величин Бернулли с вероятностью p равной 1. Уклоны передаются только через незамаскированные единицы. Во время тестирования выученные весовые векторы масштабируются по p так, что wˆ = pw, и wˆ используется (без отсева) для оценки не замеченных предложений. Мы дополнительно ограничиваем l2-нормы весовых векторов путем изменения масштаба w так, чтобы ||w||2 = s всякий раз, когда ||w||2 > s после шага градиентного спуска.

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Таблица 1: Сводная статистика для наборов данных после токенизации.

c: Количество целевых классов. l: Средняя длина предложения. N: Размер набора данных. |V |: Размер словаря. |Vpre|: Количество слов, присутствующих в наборе предварительно обученных векторов слов. Тест: Размер тестового набора (CV означает, что не было стандартного разделения тренировка/тест, поэтому использовался 10-кратный CV). Мы экспериментировали с несколькими вариантами модели.

- CNN-rand: Наша базовая модель, в которой все слова инициализируются случайным образом, а затем изменяются в процессе обучения.

- CNN-static: Модель с предварительно обученными векторами из word2vec. Все слова, включая неизвестные, которые

инициализируются случайным образом, остаются статичными, и только другие параметры модели обучаются.

- CNN-non-static: То же самое, что и выше, но предварительно обученные векторы точно настраиваются для каждой задачи.

- CNN-multichannel: Модель с двумя наборами векторов слов. Каждый набор векторов рассматривается как "канал", и каждый фильтр применяется к обоим каналам, но градиенты обратно распространяются только по одному из каналов. Таким образом, модель может точно настраивать один набор векторов, сохраняя другой статичным. Оба канала инициализируются с помощью word2vec.

Результаты сравнения наших моделей с другими методами приведены в таблице 2. Наша базовая модель со всеми случайно инициализированными словами (CNN-rand) сама по себе не показала хороших результатов. Хотя мы ожидали прироста производительности за счет использования предварительно обученных векторов, мы были удивлены величиной этого прироста. Даже простая модель со статическими векторами (CNN-static) показывает замечательные результаты, конкурируя с более сложными моделями глубокого обучения, которые используют сложные схемы объединения (Kalchbrenner et al., 2014) или

требуют предварительного вычисления деревьев разбора (Socher et al., 2013). Эти результаты говорят о том, что предварительно обученные векторы являются хорошими, "универсальными" экстракторами признаков и могут быть использованы в разных наборах данных. Тонкая настройка предварительно обученных векторов для каждой задачи дает еще большее улучшение (CNN-non-static).

Статья 2:

В этой статье предлагается эмпирическое исследование использования символьных сверточных сетей (ConvNets) для классификации текстов. Мы создали несколько крупных наборов данных, чтобы показать, что конволюционные сети на уровне символов могут достигать современных или конкурентоспособных результатов. Сравнение проводится с традиционными моделями, такими как мешок слов, n-граммы и их варианты TFIDF, а также с моделями глубокого обучения, такими как конволютные сети на основе слов и рекуррентные нейронные сети. В этом разделе мы представляем конструкцию конвентных сетей на уровне символов для классификации текста. Конструкция является модульной, где градиенты получаются методом обратного распространения для выполнения оптимизации.

Основным компонентом является временной сверточный модуль, который просто вычисляет одномерную свертку. Предположим, у нас есть дискретная входная функция g(x) ∈ [1, l] → R и дискретная функция ядра f(x) ∈ [1, k] → R. Свертка h(y) ∈ [1, b(l - k)/dc + 1] → R между f(x) и g(x) с шагом d определяется как

$$h(y) = \sum_{x=1}^{k} f(x) \cdot g(y \cdot d - x + c),$$

где c = k - d + 1 - константа смещения. Как и в традиционных сверточных сетях зрения, модуль параметризуется набором таких функций ядра fij (x) (i = 1, 2, . . . , m и j = 1, 2, ... , n), которые мы называем весами, на множестве входов gi(x) и выходов hj (y). Мы называем каждый gi (или hj) входным (или выходным) признаком, а m (или n) - размером входного (или выходного) признака. Выходные данные hj (y) получаются суммой по i сверток между gi(x) и fij (x). Одним из ключевых модулей, который помог нам обучить более глубокие модели, является временная max-pooling. Это 1-D версия модуля max-pooling, используемого в компьютерном зрении. Учитывая дискретную входную функцию g(x) ∈[1, l] → R, функция max-pooling h(y) ∈ [1, b(l - k)/dc + 1] → R для g(x) определяется как

$$h(y) = \max_{x=1}^{k} g(y \cdot d - x + c),$$

где c = k - d + 1 - константа смещения. Именно этот модуль объединения позволил нам обучить ConvNets глубже 6 слоев, где все остальные потерпели неудачу. Анализ [3] может пролить свет на это. Нелинейность, используемая в нашей модели, - это выпрямитель или пороговая функция h(x) = max{0, x}, что делает наши конволюционные слои похожими на выпрямленные линейные модули (ReLU) . Используемый алгоритм - стохастический градиентный спуск (SGD) с минибатчем размера 128, с использованием импульса 0.9 и начальным размером шага 0.01, который уменьшается вдвое каждые 3 эпохи в течение 10 раз. В каждую эпоху берется фиксированное количество случайных обучающих выборок, равномерно распределенных по классам. Это число будет позже подробно описано для каждого набора данных в отдельности. Реализация выполнена с использованием Torch 7.

Мы разработали 2 конвсети - одну большую и одну маленькую. Они обе имеют глубину 9 слоев с 6 сверточными слоями и 3 полностью связанными слоями. Рисунок 1 дает иллюстрацию.

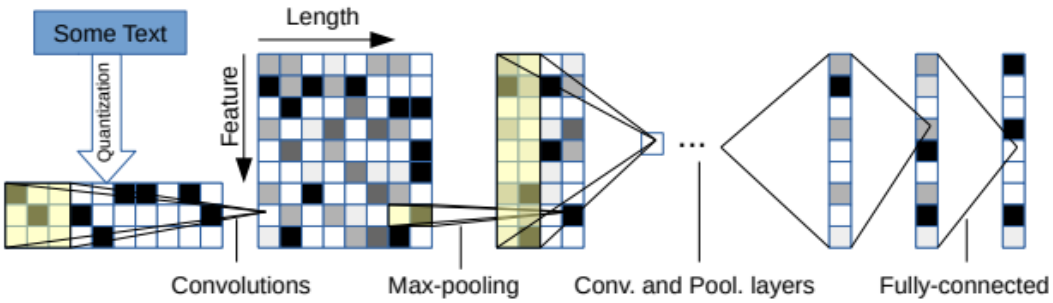

Рисунок 1: Иллюстрация нашей модели

Число признаков на входе равно 70 благодаря нашему методу квантования символов, а длина входного признака равна 1014. Кажется, что 1014 символов уже могут охватить большую часть интересующих нас текстов. Для регуляризации мы также вставляем 2 модуля dropout между тремя полностью связанными слоями. Вероятность их отсева составляет 0,5. В таблице 1 приведены конфигурации конволюционных слоев, а в таблице 2 - конфигурации полностью связанных (линейных) слоев.

Таблица 1: Конволюционные слои, используемые в наших экспериментах. Конволюционные слои имеют страйд 1, а пулинговые слои являются непересекающимися, поэтому мы опускаем описание их страйдов.

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Мы инициализируем веса с помощью гауссовского распределения. Среднее и стандартное отклонение, используемые для инициализации большой модели, составляют (0, 0.02), а малой модели (0, 0.05).

Таблица 2: Полносвязные слои, использованные в наших экспериментах. Количество выходных единиц для последнего слоя определяется задачей. Например, для задачи классификации на 10 классов оно будет равно 10.

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

Для разных задач длина входных данных может быть разной (например, в нашем случае $l0 = 1014$), как и длина кадров. Из нашей модели легко понять, что при длине входного сигнала $l0$ длина выходного кадра после последнего сверточного слоя (но до любого

из полностью связанных слоев) равна l6 = (l0 - 96)/27. Это число, умноженное на размер кадра на слое 6, даст размер входного сигнала, который принимает первый полностью связанный слой.

Вывод:

Мы сравнили большое количество традиционных моделей и моделей глубокого обучения, используя несколько крупных наборов данных. С одной стороны, анализ показывает, что ConvNet на уровне символов является эффективным методом. С другой стороны, то, насколько хорошо наша модель работает в сравнении, зависит от многих факторов, таких как размер набора данных, наличие текстов и выбор алфавита. В будущем мы надеемся применять конвсети на уровне символов для более широкого круга задач обработки языка, особенно когда требуются структурированные результаты. Мы описали серию экспериментов с конволюционными нейронными сетями, построенными на базе word2vec. Несмотря на незначительную настройку гиперпараметров, простая CNN с одним слоем свертки работает очень хорошо. Наши результаты дополняют уже имеющиеся доказательства того, что предварительное обучение векторов слов без надзора является важным компонентом глубокого обучения для НЛП.

Список литературы:

1. Yoon Kim, Convolutional Neural Networks for Sentence Classification // New York University

2. Xiang Zhang, Junbo Zhao, Yann LeCun, Character-level Convolutional Networks for Text Classification // Courant Institute of Mathematical Sciences, New York University