A1. Sentiment analysis using Naive Bayes classifier

Описание формата сдачи задания см. на mdl.cs.msu.ru (Assignment 1 Submission).

Теоретическая часть

1. Наивный байесовский классификатор, модель Бернулли

Пусть V – множество $\{v_1,v_2,...,v_M\}$ всевозможных слов (словарь). В модели Бернулли для каждого класса c_j рассматривается случайная M-мерная переменная $(B_{1j},B_{2j},...,B_{Mj})$, компоненты которой независимы, i-ая компонента соответствует v_i (i-ому слову словаря) и имеет распределение Бернулли с вероятностью успеха p_{ij}. Другими словами, для каждого класса c_j у нас имеется свой набор из M несимметричных монет. Чтобы сгенерировать документ (множество слов) из класса c_j, нужно подбросить все эти монеты и выбрать из словаря слова, соответствующие монетам, на которых выпал орел. Документы, отличающиеся только порядком слов или их частотами, считаются одинаковыми.

В каждом из следующих вопросов выпишите сначала формулу в общем виде, а затем ее оценку на обучающей выборке с использованием аддитивного сглаживания (сглаживания Лапласа).

a) Чему равна вероятность $P(v_i \in d \mid d \in c_j)$ встретить i-ое слово из словаря в случайном документе класса c_j?

b) Вывести $P(d=(k_1,k_2,...,k_M) \mid d \in c_j)$ – вероятность того, что случайный документ d, принадлежащий классу c_j, будет состоять из $k_1,k_2,...,k_M$ вхождений слов $v_1,v_2,...,v_M$? Для вывода использовать "наивное" предположение о независимости признаков.

c) Вывести вероятность $P(c_j \mid d)$, что данный документ принадлежит классу c_j. Для вывода использовать формулу Байеса.

d) Какой класс c_j будет будет выдан для документа d классификатором, если предположить, что $P(c_j)$ и $P(d \mid c_j)$ заданы? Как можно оценить вероятность ошибки?

2. Мультиномиальный наивный байесовский классификатор

В мультиномиальной модели для каждого класса c_j рассматривается случайная переменная W с категориальным распределением $Cat(r_{1j},..., r_{Mj})$ (кубик с M гранями, вероятность выпадения i-ой грани – r_{ij}) и случайная переменная N с некоторым распределением P(N), N=1..inf. Для генерации документа вначале сэмплируется его длина n~P(N), затем n раз сэмплируется очередное слово $w \sim Cat(r_{1j},..., r_{Mj})$: n раз бросаем один и тот же несимметричный кубик с M гранями. Мы будем предполагать, что длина документа не зависит от класса: $P(N|c_j)=P(N)$ – в этом случае при построении байесовского классификатора ее можно игнорировать (как мы это делаем с вероятностью документа P(d)). Документы, отличающиеся только порядком слов, считаются одинаковыми.

a)–d) Те же вопросы, что и для модели Бернулли.

Практическая часть

Классификаторы должны быть **совместимы с версиями python и библиотек, указанными на странице сдачи задания, и не должны зависеть от других библиотек**. Мы рекомендуем инсталлировать Anaconda и создать соответствующий environment.

1. Реализовать наивный байесовский классификатор Бернулли и мультиномиальный наивный байесовский классификатор. Сравнить их результаты на FILIMDB Dataset (https://github.com/nvanva/filimdb_evaluation). В качестве метрики для сравнения использовать точность (accuracy) – процент правильно классифицированных примеров из тестовой выборки. Сравнить точность на train/dev/test частях.

a) Загрузить обучающую выборку в 2 списка – позитивные и негативные отзывы. Чему равна минимальная, максимальная, средняя, медианная длина (в символах) позитивных / негативных отзывов?

b) Сделать предобработку. Перевести отзывы в нижний регистр. Вставить пробелы вокруг всех символов, не являющихся цифрами или буквами. Учитывайте, что в отзывах встречаются не только буквы латинского алфавита (например, слово cliché может быть хорошим признаком отрицательного отзыва, поэтому с ним нужно обходиться аккуратно).

c) Сделать токенизацию – то есть представить каждый отзыв в виде списка токенов (слов, чисел, знаков пунктуации). Каждый токен станет элементом словаря. При токенизации учитывайте, что в текстах знаки пунктуации приклеиваются к предшествующему слову (не отделяются пробелом). Однако, если предобработка выполнена правильно, это уже не проблема. Предобработку и токенизацию полезно выделить в отдельные функции, т.к. они понадобятся и при обработке тестовой выборки. Почему важно, чтобы тестовая и обучающая выборка обрабатывались одинаково?

d) Построить 2 Питоновских словаря слово→частота с частотами каждого слов в позитивных и негативных отзывах. Для позитивных и негативных отзывов распечатайте по 30 самых частотных слов и их частоты. Распечатайте 30 слов с максимальными и 30 слов с минимальными наивными байесовскими весами (наивным байесовским весом слова будем называть log [P(w|pos) / P(w|neg)]), сами веса, а также абсолютные частоты этих слов в позитивном и негативном классах. Результаты приведите в виде таблицы. Какие слова из полученных списков кажутся позитивно/негативно окрашенными, но имеют низкий вес?

e) Используя формулы из Теоретической части реализовать байесовские классификаторы. Обучить классификаторы на обучающей выборке и разметить с их помощью тестовую выборку. Сколько времени обучается классификаторы и сколько времени уходит на обработку тестовой выборки? Чему равна точность классификаторов на обучающей, валидационной и тестовой выборках? Результаты приведите в виде таблицы. Как вы думаете, с чем связана разная точность на обучающей и валидационной выборках? А на валидационной и тестовой выборках?

f) Словарь может состоять не только из униграмм (одиночных слов), но и n-грамм (n подряд идущих слов). Добавить в словарь биграммы (пары подряд идущих слов) – помогает ли их добавление улучшить результат и насколько? Помогает ли добавление триграмм? Нужно ли оставлять униграммы или достаточно использовать только биграммы и триграммы? А при n>3? Результаты приведите в виде таблицы.

Исследовательская часть

1. Оценки $P(w|c_j)$ для низкочастотных слов недостоверны. Например, если слово встретилось только 2 раза и оба в положительных отзывах – это может быть случайность. Если оно встретилось 1000 раз и только в положительных отзывах – это уже скорее закономерность. Попробуйте удалить из словаря слова, которые встретились < min_cnt раз. Помогает ли это улучшить качество для униграммной модели? А для биграммной? Какое значение min_cnt оптимально в каждом случае?

Сравните отсечку по document frequency и term frequency. Для этого Нарисуйте график зависимости точности от размера словаря. Подберите значение порога так, чтобы получить одинаковый размер словаря для каждого из вариантов.

2. Мультиномиальный наивный байес может учитывать или не учитывать частоты слов (во втором случае достаточно удалить повторяющиеся слова при предобработке). Есть ли разница в результатах?

3. Предобработка может сильно влиять на качество работы классификатора – поэтому имеет смысл попробовать разные варианты предобработки! Как влияет на качество перевод в нижний регистр? Стоит ли удалять или оставлять знаки препинания? Заменять все числа на специальный токен NUMBER?

4. Не все слова, которые классификатор увидит в процессе работы, будут присутствовать в словаре. Отсутствующие в словаре слова можно игнорировать, но в некоторых приложениях они могут играть важную роль. Например, для спам фильтра большое число незнакомых слов может оказаться хорошим признаком того, что пример - спам. Попробуйте заменять слова, отсутствующие в словаре, на UNK. Как оценить $P(UNK|c_j)$? Можно заменить слова, которые встретились 1 раз в обучающей выборке, на UNK, и мы автоматически получим нужные оценки для UNK.

5. Проверьте независимость длины документа от класса. Для этого нарисуйте распределения длин документов из позитивного и негативного классов. Поможет ли классификатору учет длины документа?

6. Помимо размеченных отзывов, для обучения доступно большое число неразмеченных (train_unlabeled.texts) - попробуйте ими воспользоваться для улучшения качества классификатора! Предложите способы задействовать неразмеченную выборку. Реализуйте и сравните несколько наиболее перспективных на Ваш взгляд способов.