ВМК

Задание 2. Градиентные методы обучения линейных моделей. Применение линейных моделей для определения токсичности комментария

Практикум 317 группы, 2024

Начало выполнения задания: 29 октября 2024 года, 17:00.
Мягкий Дедлайн: **18 ноября 2024 года, 10:00.**
Жёсткий Дедлайн: **22 ноября 2024 года, 23:00.**

Формулировка задания

Данное задание направлено на ознакомление с линейными моделями и градиентными методами обучения. В задании необходимо:

1. Написать на языке Python собственную реализацию линейного классификатора с произвольной функцией потерь. Прототипы функций должны строго соответствовать прототипам, описанным в спецификации и проходить все тесты. Задание, не проходящее все тесты, приравнивается к невыполненному. При написании необходимо пользоваться стандартными средствами языка Python, библиотеками `numpy`, `scipy` и `matplotlib`. Библиотекой `scikit-learn` пользоваться запрещается, если это не обговорено отдельно в пункте задания.
2. Вывести все необходимые формулы, привести выкладки в отчёте.
3. Провести описанные ниже эксперименты с модельными данными и приложенным датасетом.
4. Написать отчёт о проделанной работе (формат PDF). Отчёт должен быть подготовлен в системе LaTeX.

Теоретическая часть

1. Выведите формулу градиента функции потерь для задачи бинарной логистической регрессии. Запишите вывод.
2. Выведите формулу градиента функции потерь для задачи многоклассовой (мультиномиальной) логистической регрессии. Запишите вывод.
3. Покажите, что при количестве классов $= 2$, задача мультиномиальной логистической регрессии сводится к бинарной логистической регрессии.

Список экспериментов

Эксперименты этого задания необходимо проводить на датасете, содержащим комментарии из раздела обсуждений английской Википедии, который был преобразован для решения задачи бинарной классификации: является ли данный комментарий токсичным или нет. Подробнее об исходных данных здесь. Требуемый для выполнения данного задания датасет можно найти по следующей ссылке. Данные в датасете записаны в формате csv.

1. Произведите предварительную обработку текста. Приведите все тексты к нижнему регистру. Замените в тексте все символы, не являющиеся буквами и цифрами, на пробелы.

   Замечание. Полезные функции: `str.lower`, `str.split`, `str.isalnum`, `re.sub`, `re.split`.

2. Преобразуйте выборку в разреженную матрицу `scipy.sparse.csr_matrix`, где значение `x` в позиции `(i,j)` означает, что в документе `i` слово `j` встретилось `x` раз. Разрешается воспользоваться конструктором `sklearn.feature_extraction.text.CountVectorizer`.

   Замечание 1. У `CountVectorizer` есть несколько методов для работы, используйте `fit_transform` и `fit` для обучающей выборки, используйте `transform` для тестовой.

   Замечание 2. Используйте параметр `min_df`, чтобы уменьшить размерность данных и ускорить проведение экспериментов.

3. Реализуйте методы градиентного и стохастического градиентного спуска в соответствии с требованиями к реализации. Сравните численный подсчет градиента функции потерь из модуля utils.py с вычислением по аналитической формуле.

4. Исследуйте поведение градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

   - параметр размера шага `step_alpha`
   - параметр размера шага `step_beta`
   - начального приближения

   Исследование поведения подразумевает анализ следуюших зависимостей:

   - зависимость значения функции потерь от итерации метода (эпохи в случае стохастического варианта)
   - зависимость точности (accuracy) итерации метода (эпохи в случае стохастического варианта)
   - При желании можно проанализировать зависимость значения функции потерь и зависимость точности от реального времени работы метода и их отличия от зависимости от итерации.

5. Исследуйте поведение стохастического градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

   - параметр размера шага `step_alpha`
   - параметр размера шага `step_beta`
   - размер подвыборки `batch_size`
   - начального приближения

6. Сравните поведение двух методов между собой, сделайте выводы.

7. Примените алгоритм лемматизации (например, `WordNetLemmatizer` из библотеки `nltk`) к коллекции. Удалите из текста стоп-слова (например, используя список стоп-слов из `nltk`). Исследуйте, как предобработка корпуса повлияла на точность классификации, время работы алгоритма и размерность признакового пространства.

8. Исследуйте качество, время работы алгоритма и размер признакового пространства в зависимости от следуюших факторов:

   - использовалось представление `BagOfWords` или `Tfidf`
   - параметров `min_df` и `max_df` конструкторов.

   Замечание. Для построения tf-idf представления воспользуйтесь `TfidfTransformer` или `TfidfVectorizer` из библиотеки `sklearn`.

9. Выберите лучший алгоритм для тестовой выборки. Проанализируйте ошибки алгоритма. Проанализируйте и укажите общие черты объектов, на которых были допущены ошибки.

Требования к реализации

Прототипы всех функций описаны в файлах, прилагающихся к заданию.

Среди предоставленных файлов должны быть следующие модули и функции в них:

1. Модуль `oracles.py` с реализациями функций потерь и их градиентов.

   Обратите внимание на то, что все функции должны быть полностью векторизованы (т.е. в них должны отсутствовать циклы).

   Замечание 1. В промежуточных вычислениях стоит избегать вычисления $\exp(-b_i\langle x_i, w\rangle)$, иначе может произойти переполнение. Вместо этого следует напрямую вычислять необходимые величины с помощью специализированных для этого функций: `np.logaddexp`, `scipy.special.logsumexp` и `scipy.special.expit`. В ситуации, когда вычисления экспоненты обойти не удаётся, можно воспользоваться процедурой «клипинга» (функция `numpy.clip`).

   Замечание 2. При вычислении нормировки $\frac{\exp(\alpha_i)}{\sum_k \exp(\alpha_k)}$ может произойти деление на очень маленькое число, близкое к нулю. Необходимо воспользоваться следующим трюком:

$$\frac{\exp(\alpha_i)}{\sum_k \exp(\alpha_k)} = \frac{\exp(\alpha_i - \max \alpha_j)}{\sum_k \exp(\alpha_k - \max \alpha_j)}$$

2. Модуль `utils.py` с реализацией функции численного подсчёта градиента произвольного функционала.

3. Модуль `optimization.py` с реализацией методов обыкновенного и стохастического градиентного спуска.

Замечание. Для всех функций можно задать аргументы по умолчанию, которые будут удобны вам в вашем эксперименте. Ко всем функция можно добавлять необязательные аргументы, а в словарь `history` разрешается сохранять необходимую в ваших экспериментах информацию.

Бонусная часть

1. **(до 2 баллов)** Добавьте в признаковое пространство n-граммы (измените параметр `ngramm_range` у `TfidfTransformer`). Исследуйте, как влияет размер максимальных добавленных n-грамм на качество и скорость работы алгоритма.

2. **(до 5 баллов)** Реализуйте режим работы алгоритма `SGDClassifier`, при котором вся обучающая выборка не будет храниться в оперативной памяти.

   - Перемешайте документы в исходном файле, чтобы большое количество документов одного класса не шли подряд друг за другом.
   - Реализуйте итератор, который будет считывать следующие `batch_size` строк из созданного файла и преобразовывать их в `numpy.array` или `sparse.csr_matrix` соответствующего размера. При считывании последнего документа, итератор начинает считывание документов заново.
   - Реализуйте специальный режим обучения для алгоритма стохастического градиентного спуска, который будет принимать на вход вместо матрицы объекты-признаки рассмотренный выше итератор.

   Попробуйте оценить сколько памяти удаётся сэкономить, используя такой подход, и насколько медленнее такой подход по времени.

   Алгоритм, реализованный таким образом, на каждой итерации не будет выбирать произвольные документы, а будет брать следующую пачку документов, пришедшую из итератора. Таким образом, алгоритм станет «менее случаен». Исследуйте, повлияет ли это на качество работы алгоритма. Рассмотрите несколько различных значений параметра `batch_size`.

   Замечание. Критерий остановки оптимизации алгоритма связанный с разностью значений функции на соседних эпохах для такого режима обучения неэффективен, необходимо использовать другие критерии, не зависящие от объектов всей выборки, либо производить константное число итераций метода.

3. **(до 5 баллов)** Улучшить качество работы линейных алгоритмов на датасете с помощью средств, не использующихся в задании. Например, можно реализовать ансамбль линейных алгоритмов, использовать продвинутые методы выделения коллокаций, выделять специальные термины или сущности в тексте. Обратите также внимание на специфику задачи: вполне возможно, что альтернативная обработка текста (учет регистра, знаков препинания и т.п.) поможет значительно улучшить качество вашей модели. Размер бонуса зависит от величины улучшения и от изобретательности подхода.