ВМК

Задание 3. Метод опорных векторов

Курс: Практикум на ЭВМ, осень 2016

Начало выполнения задания: 3 ноября.
Срок сдачи: **20 ноября, 23:59.**
Среда для выполнения задания: Python 3.4 или Python 2.7

1 Ликбез

Зафиксируем обозначения:

- N — число объектов в обучающей выборке.
- D — размерность признакового пространства.
- $\mathbf{x}_n \in \mathbb{R}^D$ — вектор признаков объекта n.
- $y_n \in \{-1, 1\}$ — правильный ответ для объекта n.

Прямая задача SVM

$$\min_{\mathbf{w}, \boldsymbol{\xi} \geqslant 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n,$$
$$\text{s.t. } y_n(\mathbf{w}^\intercal \mathbf{x}_n + w_0) \geqslant 1 - \xi_n,\ n = 1, \ldots, N.$$

Прямая задача SVM без ограничений В предыдущей задаче можно избавиться от переменных ξ_n, если учесть, что $\xi_n \geqslant 0$ и $\xi_n \geqslant 1 - y_n\mathbf{w}^\intercal\mathbf{x}_n$:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \max\{0, 1 - y_n(\mathbf{w}^\intercal \mathbf{x}_n + w_0)\}.$$

Двойственная задача SVM

$$\max_{\mathbf{a}} \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$$
$$\text{s.t. } 0 \leqslant a_n \leqslant C,\ n = 1, \ldots, N,$$
$$\sum_{n=1}^{N} a_n y_n = 0,$$

где $k(\mathbf{x}_n, \mathbf{x}_m)$ — ядровая функция, в линейном случае ее значение равно $\mathbf{x}_n^\intercal \mathbf{x}_m$.

Субградиент Вектор $\mathbf{g} \in \mathbb{R}^n$ является субградиентом выпуклой функции $f\colon \mathbb{R}^n \to \mathbb{R}$ в точке $\mathbf{x} \in \mathbb{R}^n$, если $\forall \mathbf{z} \in \mathbb{R}^n$ выполнено неравенство

$$f(\mathbf{z}) \geqslant f(\mathbf{x}) + \mathbf{g}^\intercal(\mathbf{z} - \mathbf{x}).$$

Если функция f дифференцируема в точке $\mathbf{x}$, ее субградиент в этой точке совпадает с градиентом. *Субдифференциалом* функции f в точке $\mathbf{x}$ называют множество субградиентов в этой точке, обозначают $\partial f(\mathbf{x})$.

Рассмотрим пример вычисления субдифференциала для функции $f(x) = |x|$. При $x < 0$ субградиент единственен: $\partial f(x) = -1$; аналогично при $x > 0$: $\partial f(x) = 1$. При $x = 0$ субдифференциал определяется неравенством $|z| \geqslant gz$ для любого $z \in \mathbb{R}$, это неравенство выполнено только при $g \in [-1, 1]$, таким образом $\partial f(0) = [-1, 1]$.

Субградиентный спуск — метод аналогичный методу градиентного спуска, в котором вместо градиента используется субградиент.

2 Формулировка задания

Требуется реализовать следующие методы для решения задачи SVM:

1. Метод внутренней точки для решения прямой задачи. Рекомендуется использовать библиотеку cvxopt, метод `cvxopt.solvers.qp`.
2. Метод внутренней точки для решения двойственной задачи. Рекомендуется использовать библиотеку cvxopt, метод `cvxopt.solvers.qp`.
3. Метод субградиентного спуска для решения прямой задачи, а также его стохастический вариант. Рассмотреть критерий останова как по значению целевой функции, так и по норме аргумента. Реализовать полностью самостоятельно. Для этого потребуется вывести формулу для субградиента функционала в прямой задаче SVM без ограничений, вывод вставить в отчет.
4. Метод, используемый в библиотеке liblinear. Рекомендуется использовать биндинги из библиотеки scikit-learn, класс `sklearn.svm.LinearSVC`.
5. Метод, используемый в библиотеке libsvm. Рекомендуется использовать биндинги из библиотеки scikit-learn, класс `sklearn.svm.SVC`.

Исследовательская часть. Для проведения исследований необходимо генерировать модельные данные, которые не являются линейно разделимыми, для этого удобно использовать многомерные нормальные распределения. Минимальный размер выборки — по 100 объектов в каждом классе.

Требуется провести следующие исследования:

1. Исследовать зависимость времени работы реализованных методов для решения задачи линейного SVM от размерности признакового пространства и числа объектов в обучающей выборке. Исследовать скорость сходимости методов. Сравнить методы по полученным значениям целевой функции.
2. Провести эти исследования для случая SVM с RBF ядром для тех методов, где возможен ядровой переход.
3. Реализовать процедуру поиска оптимального значения параметра C и ширины RBF ядра с помощью кросс-валидации (можно воспользоваться библиотекой scikit-learn). Исследовать зависимость ошибки на валидационной выборке от значений этих параметров. Рассмотреть случаи хорошо и трудно разделимых выборок.
4. Сравнить (по скорости сходимости и точности решения) несколько стратегий выбора шага α_t в методе субградиентого спуска: α, $\frac{\alpha}{t}$, $\frac{\alpha}{t^\beta}$, где α, β — некоторые константы, t — номер итерации.
5. Исследовать, как размер подвыборки, по которой считается субградиент, в методе стохастического субградиентного спуска влияет на скорость сходимости метода и на точность решения. В этом и предыдущем пунктах за точное решение можно взять решение, полученное с помощью одного из методов внутренней точки.
6. Для двумерного случая:
   - Провести визуализацию выборки.
   - Для линейного SVM и для SVM с RBF ядром провести визуализацию разделяющей поверхности
   - Отобразить объекты, соответствующие опорным векторам.

3 Требования к оформлению

Для сдачи задания необходимо предоставить:

1. Отчет в формате pdf (оформленный в системе LaTeX) или IPython notebook с описанием всех проведенных исследований со всеми графиками и выводами.
2. Python модуль svm.py со классом `SVM`, в соответствии со спецификациями, приведенными ниже.
3. Если предоставлен отчёт в формате pdf, предоставить также IPython notebook с кодом для воспроизведения всех результатов из отчета: таблиц, графиков и т.д.

4 Спецификации

Необходимо предоставить модуль svm.py, в котором должен быть реализован класс `SVM`.

Замечание 1. Все описанные ниже функции должны работать для бинарного случая, в этом случае метки классов принимают значения 1 и -1.

Замечание 2. Для проверки задания используются автоматические тесты, которые чувствительны к неверным прототипам функций/изменённым названиям атрибутов класса. В используемые прототипы можно добавлять аргументы по умолчанию, в класс можно добавлять свои методы и атрибуты.

В классе должны присутствовать следующие атрибуты:

1. Атрибуты, которые задаются при инициализации:
   - `C` — параметр регуляризации, `float`
   - `method` — метод для решения задачи SVM, должен принимать следующие значения:
     - `'primal'` — метод внутренней точки для решения прямой задачи
     - `'dual'` — метод внутренней точки для решения двойственной задачи
     - `'subgradient'` — метод субградиентного спуска для решения прямой задачи
     - `'stoch_subgradient'` — метод стохастического субградиентного спуска для решения прямой задачи
     - `'liblinear'` — метод, используемый в библиотеке liblinear
     - `'libsvm'` — метод, используемый в библиотеке libsvm
2. Атрибуты, которые появляются после обучения:
   - `w` — вектор весов SVM для прямой задачи, переменная типа `numpy.array`, матрица размера $D \times 1$ (только в случае, когда решается прямая задача)
   - `A` — значения двойственных переменных, матрица размера $N \times 1$ (только в случае, когда решается двойственная задача)
3. Атрибуты, которые необходимо задать, если решается двойственная задача:
   - `kernel` — ядро, использующееся для решения. Может принимать следующие значения:
     - `'linear'` — линейный случай
     - `'rbf'` — RBF ядро
   - `gamma` — ширина RBF ядра (если `kernel` = `'rbf'`), `float`

Замечание 4. Здесь и далее полагается, что:

`X` — переменная типа `numpy.array`, матрица размера $N \times D$, признаковые описания объектов из обучающей выборки,

`y` — переменная типа `numpy.array`, матрица размера $N \times 1$, правильные ответы на обучающей выборке,

В классе должны быть реализованы следующие методы:

1. `compute_primal_objective(self, X, y)` — метод для подсчета целевой функции SVM для прямой задачи

   Функция должна возвращать одно число — значение целевой функции.
2. `compute_dual_objective(self, X, y)` — метод для подсчета целевой функции SVM для двойственной задачи

   Функция должна возвращать одно число — значение целевой функции.

   Замечание. Учтите, что для разных ядер — разные целевые функции.
3. `fit(self, X, y, tol, max_iter, verbose, stop_criterion, batch_size, lamb, alpha, beta)` — функция для обучения svm с помощью метода, заданного параметром `method`.

   Функция принимает следующие аргументы:
   - `tol` — требуемая точность для метода обучения
   - `max_iter` — максимальное количество итераций в методе обучения
   - `verbose` — в случае `True`, требуется выводить при обучении отладочную информацию на экран (например номер итерации, значение целевой функции)

   Атрибуты, которые необходимо задать, если задача решается с помощью субградиентных методов:

- `alpha` — параметр α для шага субградиентного спуска
- `beta` — параметр β для шага субградиентного спуска
- `stop_criterion` — критерий останова, может принимать следующие значения:
  - 'objective' — остановка по значению целевой функции в случае субградиентного метода, для стохастического субградиентного метода необходимо использовать экспоненциальное скользящее среднее
  - 'argument' — остановка по норме аргумента
- `batch_size` — размер подвыборки, по которой считается субградиент (только если используется стохастический градиентный спуск)
- `lamb` — параметр забывания (только если используется стохастический градиентный спуск и критерий останова `'objective'`)

Функция должна заполнять следующие атрибуты класса:

- `w` в случае прямой задачи и `A` в случае двойственной задачи

Функция должна вернуть словарь со следующими полями:

- `'status'` — 0 или 1, 0 — если после обучения метод вышел по критерию останова, 1 — если по числу итераций (кроме методов 'liblinear' и 'libsvm')
- `'objective_curve'` — список значений целевой функции по итерациям метода (только для 'subgradient' и 'stoch_subgradient')
- `'time'` — время обучения метода

4. `predict(self, X_test, return_classes=False)` — метод для предсказаний ответов по новым данным.

   Описание параметров:

   - `X_test` — переменная типа `numpy.array`, матрица размера $M \times D$, признаковые описания объектов из тестовой выборки
   - `return_classes` — если значение равно `True`, необходимо вернуть метки классов, иначе вернуть оценки, выдаваемые классификатором

   Функция должна возвращать `numpy array` размера $M \times 1$, ответы алгоритма на тестовой выборке.

5. `compute_support_vectors(self, X, y)` — метод для определения опорных векторов.

   Метод работает только в случае двойственной задачи. Метод должен возвращать `numpy.array`, матрицу размера $K \times D$, где K — число найденных опорных векторов.

6. `compute_w(self, X, y)` — метод для получения прямых переменных `w` по двойственным `A`.

   Работает только в случае двойственной задачи без ядер. Метод должен возвращать вектор весов SVM для прямой задачи, переменную типа `numpy.array`, матрицу размера $D \times 1$

Помимо класса необходимо также написать функцию визуализации для двумерного случая, соответствующую требованиям пункта 6 исследовательской части задания:

`visualize(X, y, alg_svm, show_vectors=False)`

Описание параметров:

- `alg_svm` — обученный SVM, объект класса `SVM`
- `show_vectors` — если значение равно `True`, необходимо отразить объекты, соответствующие опорным векторам (только в случае решения двойственной задачи)

Бонус + 0.3 — даётся за обработку всех некорректных вызовов методов и обращений к атрибутам (т.е. например, при вызове функций для двойственной задачи при решении прямой должно выбрасываться исключение).