ВМК

Задание 3. Метод опорных векторов

Курс: Практикум на ЭВМ, осень 2015

Начало выполнения задания: 5 ноября.
Срок сдачи: **22 ноября, 23:59.**
Среда для выполнения задания: Python 3.4.

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) с описанием всех проведенных исследований со всеми графиками и выводами.

2. IPython notebook с кодом для воспроизведения всех результатов из отчета: таблиц, графиков и проч.

3. Python модуль со всеми требуемыми функциями, в соответствии со спецификациями, приведенными ниже.

4 Спецификация

В предоставленном модуле должны быть реализованы функции:

1. Функции подсчета целевой функции SVM, для прямой и двойственных задач:

   ```
   compute_primal_objective(X, y, w, C)
   compute_dual_objective(X, y, w, C, gamma=0)
   ```

   Описание параметров:

   - `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,
   - `w` — переменная типа `numpy.array`, матрица размера $D \times 1$, вектор весов SVM,
   - `C` — параметр регуляризации,
   - `gamma` — ширина RBF ядра. Если `gamma=0`, рассматривается линейный случай.

   Функции должны возвращать одно число — значение целевой функции.

2. Функции для решения задачи SVM:

   ```
   svm_subgradient_solver(X, y, C, tol=1e-6, max_iter=100, verbose=False)
   svm_qp_primal_solver(X, y, C, tol=1e-6, max_iter=100, verbose=False)
   svm_qp_dual_solver(X, y, C, tol=1e-6, max_iter=100, verbose=False, gamma=0)
   svm_liblinear_solver(X, y, C, tol=1e-6, max_iter=100, verbose=False)
   svm_libsvm_solver(X, y, C, tol=1e-6, max_iter=100, verbose=False, gamma=0)
   ```

   Описание параметров:

   - `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,
   - `C` — параметр регуляризации,
   - `tol` — требуемая точность,
   - `max_iter` — максимальное число итераций,
   - `verbose` — в случае `True`, требуется выводить отладочную информацию на экран (номер итерации, значение целевой функции),
   - `gamma` — ширина RBF ядра. Если `gamma=0`, рассматривается линейный случай.

   Функции должны возвращать словарь с полями:

   - `'w'` — `numpy.array`, матрица размера $K \times 1$,
   - `'A'` — только в случае решения двойственной задачи, `numpy.array`, матрица размера $N \times 1$, значения двойственных переменных,
   - `'status'` — 0 или 1, 0 — если метод вышел по критерию останова, 1 — если по числу итераций,
   - `'objective_curve'` — список значений целевой функции по итерациям метода,
   - `'time'` — время работы метода.

3. Функция определения опорных векторов:

   ```
   compute_support_vectors(X, y, A)
   ```

   Описание параметров:

   - `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,
   - `A` — переменная типа `numpy.array`, матрица размера $N \times 1$, значения двойственных переменных.

   Функция должна возвращать `numpy.array`, матрицу размера $K \times D$, где K — число найденных опорных векторов.

4. Функция получения прямых переменных $\mathbf{w}$ по двойственным a_n:

   `compute_w(X, y, A)`

   Описание параметров:

   - `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,
   - `A` — переменная типа `numpy.array`, матрица размера $N \times 1$, значения двойственных переменных.

   Функция должна возвращать `numpy.array`, матрицу размера $D \times 1$.

Все описанные выше функции должны работать для бинарного случая, в этом случае метки классов принимают значения 1 и -1. В случае $K \geqslant 2$ классов они нумеруются от 0 до $K - 1$.

Требуется также реализовать функцию визуализации, соответствующую требованиям пункта 6 исследовательской части задания:

`visualize(X, y, w, A=None)`

Описание параметров:

- `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,
- `w` — переменная типа `numpy.array`, матрица размера $D \times K$ или $D \times 1$ в случае двух классов, вектор весов SVM,
- `A` — переменная типа `numpy.array`, матрица размера $N \times K$ или $N \times 1$ в случае двух классов, значения двойственных переменных. Если этот параметр не задан, функция не должна отображать опорные вектора.