Домашние задания по курсу машинного обучения

Dmitry Mukhutdinov

10 октября 2016 г.

Содержание

1 Задание 1.

1.1 Постановка задачи

1. Реализовать метрический классификатор kNN
2. Сделать кросс-валидацию; обосновать выбор числа фолдов для нее
3. Выполнить визуализацию данных
4. Настроить классификатор с 2-3 метриками и 2-3 пространственными преобразованиями
5. Для оценки качества можно использовать метрику accuracy, но лучше - f1-measure

1.2 Датасет

Ссылка: `https://www.dropbox.com/s/lolwyijk22xu7na/chips.txt?dl=0`
Датасет представляет собой набор 2d-точек, разбитых на 2 класса.

1.3 Hints

1. Нарисуйте график, изображающий точки из разных классов разными цветами. Сразу станет понятно, где примерно должна проходить граница разделения классов.
2. Добавьте пространственные преобразования, которые хорошо разделяют датасет. Таким, например, является параболоид с центром в среднем арифметическом всех точек.

1.4 FAQ

1. **Вопрос:** Что такое метрический классификатор?

   Ответ: Это алгоритм классификации, который основан на понятии **сходства** между объектами. При классификации очередного объекта решение принимается на основе известных ответов для объектов, схожих с данным. Обычно функция сходства является **метрикой**, но необязательно (например, может нарушаться равенство треугольника).

2. **Вопрос:** Что будет, если $k = 0$? А если $k \approx N$? (N - размер датасета)

   Ответ: Первый случай - вырожденный, в этом случае классификатор просто не работает.

   Во втором случае классификатор любой объект будет относить к тому классу, представителей которого в исходном датасете большинство.

3. **Вопрос:** Как выбиралось количество фолдов для кросс-валидации? Почему именно такое?

   Ответ: Около 5 фолдов. Датасет очень маленький, поэтому в ином случае фолды будут слишком мелкие. Сослаться на слайды первой лекции[1] , стр. 18.

4. **Вопрос:** Как правильно подобрать k - количество ближайших соседей, на которых мы ориентируемся?

   Ответ: Аналогично с Leave-One-Out кросс-валидацией (Первая лекция[1], стр. 24)

5. **Вопрос:** Что можно делать для улучшения качества классификации, кроме подбора гиперпараметров (т. е. числа k, выбора метрики и ядра)?

   Ответ: Prototype selection и распознавание аномалий (упоминалось в лекции[1]), чтобы почистить датасет от кривых данных. Как именно это делать, не спрашивают.

6. **Вопрос:** Как устроено k-d tree? Как работают kNN-запросы в нем?

   Ответ: Вспоминаем вычгеом и/или читаем статью[2]. Обычно достаточно помахать руками и ляпнуть что-нибудь про разделение точек по медианам при построении и отсекание квадратиков при запросе.

2 Задание 2

2.1 Постановка задачи

1. Реализовать линейную регрессию
2. Настраивать вектор коэффициентов двумя способами - градиентным спуском и генетическим алгоритмом
3. Для оценки качества использовать MSE (среднеквадратичную ошибку)
4. Выбирать гиперпараметры можно произвольным образом, но придется обосновать свое решение
5. Модель должна уметь дообучаться по произвольным точкам (с консоли, если у вас консольное приложение)

2.2 Датасет

Ссылка: `https://www.dropbox.com/s/eoyz1uvis41xgrw/prices.txt?dl=0`

Датасет представляет собой зависимость цен на жилье от площади и количества комнат.

2.3 Hints

1. Нормализуйте свой датасет (сдвиньте точки по каждой оси на среднее значение и разделите на стандартное отклонение). Если вы пишете на Python, можно воспользоваться готовым инструментом ‘sklearn.preprocessing.StandardScaler‘[3]. Это сильно упростит работу как градиентного спуска, так и эволюционного алгоритма.
2. При реализации эволюционного алгоритма особью является вектор коэффициентов Θ. Для хорошего результата достаточно делать один вид мутации - добавление к вектору коэффициентов случайного, нормально распределенного шума (в Python сгенерировать случайный вектор из нормального распределения можно с помощью функции ‘numpy.random.randn‘[4]). Можно даже без

[1] Слайды первой лекции: `https://www.dropbox.com/sh/0fk38jg1f5ty1oz/AAD8Z_Hf8Gs6EsE3WNCBh2bWa/02-Distance.pdf?dl=0`

[2] Nearest Neighbor with k-d trees: `http://courses.cs.washington.edu/courses/cse599c1/13wi/slides/lsh-hashkernels-annotated.pdf`

[3] Scikit-learn documentation - StandardScaler: `http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

[4] NumPy documentation - numpy.random.randn: `http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randn.html`

скрещивания. С размером потомства и процентом выживаемости можно поэкспериментировать, экспериментально хорошо работает увеличение популяции в 6 раз и выживаемость 1/6 популяции.

3. Если вы чувствуете в себе силы, в качестве эволюционного алгоритма можно выбрать алгоритм дифференциальной эволюции[5]. Он сходится лучше, чем наивная эволюция.

2.4 FAQ

1. **Вопрос:** Как работает градиентный спуск?

   Ответ: См. презентацию[6].

   Вкратце: мы минимизируем функцию ошибки $Q(w)$, аргументом которой является $w = (w_1, w_2, ...w_k)$ - вектор коэффициентов линейной модели. Мы делаем это, вычисляя функцию в какой-либо начальной точке, и сдвигая эту точку в направлении, противоположном градиенту $Q(w)$. Постоянно перемещаясь в направлении антиградиента, мы приходим к минимуму.

2. **Вопрос:** Какое у вас условие сходимости?

   Ответ: Алгоритм сошелся, когда разница в значениях функции ошибки между шагами перестала превышать некоторый маленький порог: $Q(w^{[k+1]}) - Q(w^{[k]}) \leqslant \varepsilon$.

3. **Вопрос:** Как подбирать размер шага α и количество итераций?

   Ответ: Максимальное количество итераций можно сразу выставить каким-то разумно большим (в районе нескольких тысяч), и перебирать α от больших (5-10) к маленьким, пока алгоритм не начнет сходиться. Особо продвинутые могут сделать динамическую зависимость α от номера итерации и/или других условий.

   На деле же датасет таков, что заходит просто потыкать несколько значений α руками, особенно если предварительно его нормализовать.

4. **Вопрос:** Как работает ваш эволюционный алгоритм? Как вы подбирали параметры для него?

   Ответ: Как написали, так и отвечайте. Задача может решаться всевозможными эволюционными алгоритмами, описание одного из вариантов реализации можно увидеть в подпункте Hints. Гиперпараметры (размер потомства, процент выживаемости и пр.) можно попытаться подобрать с помощью кросс-валидации, но на деле лучше всего работает метод "от фонаря".

[5] Wikipedia - Differential evolution: `https://en.wikipedia.org/wiki/Differential_evolution`

[6] Слайды второй лекции: `https://www.dropbox.com/sh/0fk38jg1f5ty1oz/AABrdOgBrCJPEI5fQtlL5GHja?dl=0&preview=03-Linear.pdf`