Прогнозирование оптимальных суперпозиций в задачах регрессии

Шибаев Иннокентий, 474 группа (ФУПМ МФТИ)

10 июня 2018 г.

1. Введение

1.1. Актуальность темы.

Во многих задачах машинного обучения возникает проблема выбора наилучшей в терминах некоторого функционала качества модели (суперпозиции базовых функций) описывающей исходные данные. Задача выбора модели (задача определения данной суперпозиции) является переборной, и ввиду этого имеет высокую вычислительную сложность.

1.2. Цель работы.

Предложить метод прогнозирования оптимальных суперпозиций в задачах регрессии исходя из прецедентов выборов моделей.

2. Постановка задачи

2.1. Описание выборки и исходная постановка

Решается задача регрессии. Дан некоторый набор выборок $\mathcal{A} = \{\mathbf{A}_1, \dots, \mathbf{A}_N\}$, $\mathbf{A}_i = (\mathbf{X}_i, \mathbf{y}_i)$ где $\mathbf{X}_i$ — признаковое описание n_i объектов i-й выборки. а $\mathbf{y}_i$ - ответы на них. Для каждой пары A_i также задана некоторая f_i — суперпозиция базовых функций являющаяся порождающей функцией этой выборки (т.е. $f_i(\mathbf{X}_i) = y_i$).

Также предполагается, что данные однородны в том смысле, что $f_i \in \mathcal{F}$ — семейство порождающих функций m переменных и $\mathcal{F} = \{f : f = sup(g_1, \dots, g_l)\}$ где $sup(g_1, \dots, g_l)$ — суперпозиция l заданных базовых функций $g_1, \dots, g_l$.

Для пары $\mathbf{A} = (\mathbf{X}, \mathbf{y})$Требуется найти суперпозицию f^* наилучшим образом приближающую данную выборку

$$f^* = \arg\min_{f \in \mathcal{F}} S(f|w^*, \mathbf{X}, \mathbf{y}) \tag{1}$$

где S — заданная функция ошибки, w^* - оптимальный набор параметров для модели f при заданных $\mathbf{X}, \mathbf{y}$:

$$w^* = \arg\min_{f \in \mathcal{F}} S(w|f, \mathbf{X}, \mathbf{y}).$$

В качестве функции потерь S будем использовать разность квадратов регрессионных остатков:

$$S(f|w, \mathbf{X}, \mathbf{y}) = |f(\mathbf{X}, w) - \mathbf{y}|_2^2$$

2.2. Метод решения

Рассмотрим отображение $h : \mathcal{A} \to \mathcal{F}$ которое дает решение (1). Тогда можно переписать задачу в виде

$$h^* = \arg\min_{h \in H} \sum_{i=1}^{N} \|h(\mathbf{A}_i)(\mathbf{X}_i) - \mathbf{y}_i\|_2^2$$

где H - некоторое параметрическое множество допустимых отображений. Теперь воспользуемся тем, что для каждого $\mathbf{A}_i$ нам также дана функция f_i порождающая эту выборку: $|f_i(\mathbf{X}_i) - \mathbf{y}_i|_2 = 0$. Заменяя, получим

$$h^* = \arg\min_{h \in H} \sum_{i=1}^{N} \|h(\mathbf{A}_i)(\mathbf{X}_i) - \mathbf{y}_i\|_2^2$$

2.3. Описание алгоритма

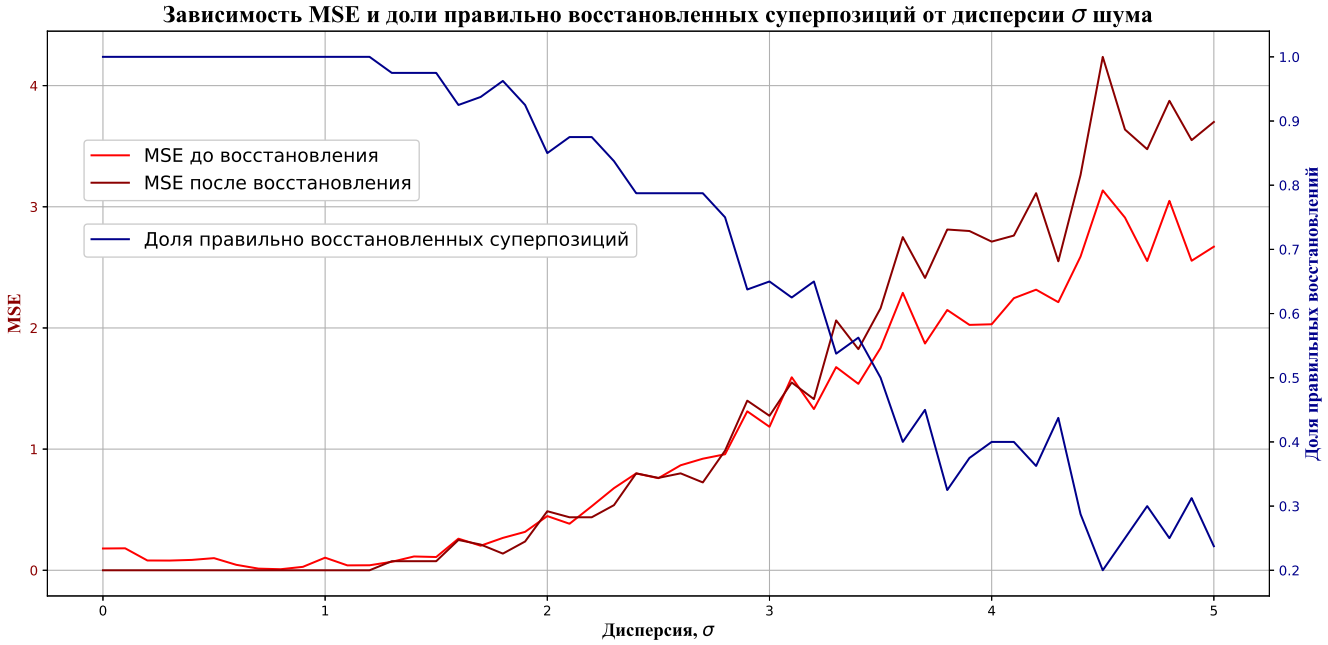
- В начале генерируется достаточно большое обучающее множество синтетических суперпозиций (линейные суперпозиции простых функций, возможно также суперпозиции следующего уровня). Для них делается зашумление.
- На этом множестве обучаем какой-нибудь классификатор (LogReg, SVC или что-нибудь такое). Для векторизации (нам нужно предугадывать много ответов) используем sklearn.multioutput.MultiOutputClassifier (для каждого столбца в Y строится свой классификатор)

- Запускаем обученный классификатор на реальных данных, и смотрим на полученную матрицу задающую суперпозицию, восстанавливаем тем или иным способом (важно, что мы не predict-им сами матрицы, мы используем *predict_probe* чтобы считать матрицу вероятностей, после чего по ней восстанавливаем матрицу суперпозиции (это хотя бы гарантирует корректность последней)).

  Надежда на то что уже просмотренные нами суперпозиции в некотором смысле хорошо аппроксимируют нашу новую функцию.

2.4. Эксперименты на синтетических данных

Технология описана выше, здесь речь о том что вообще можно проверять.



- Во-первых, можно посмотреть на зависимость от шума. Это я даже сделал, см графики выше. Хотя я не определился с размещением подписей, если есть предложения - пишите. У нас достаточно странный шаблон (там дико большие поля, 14 шрифт и т.д.) так что не хочу делать картинки в несколько колонок, постараюсь запихивать все в одну.

  Итак - генерируем набор порождаюзих функций, генерируем суперпозиции, для каждой генерируем несколько наборов зашумленных данных, после чего делим на test и train, учимся и предсказываем. Метрики основные - две: первая это accuracy (просто доля матриц суперпозиций что были точно восстановлены), а вторая - среднее отклонение (квадрат).

- Во-вторых (это кажется реально важным): посмотреть на то как работает алгоритм если в качестве train и test давать зашумленные функции (суперпозиции) но порожденные одним и тем же набором порождающих функций. По идее качество должно быть похуже, но надо проверить. Это как раз ближе к реальному использованию.

- В-третьих надо проверить как сильно влияет параметр сложности суперпозиции. Я считаю сложность так:

$$e^{\sin(\ln(x))}$$

  имеет "глубину"равную трем. Этому я и полагаю равной сложность. Глубина равная 1 соответствует линейным комбинациям базовых функций (к примеру $sin(x) + ln(x) + e^x$).

  Понятно, что тут качество должно быстро падать, все же я использую достаточно слабые алгоритмы классификации. Но все же то же интересно.

- В-четвертых надо придумать какие реальные данные использовать. У меня пока нет идей, хотя можно попробовать использовать те же временные ряды (USC-HAD и прочая).

Из проблем - не знаю как прикрутить коэффициенты. Т.е. варьирование к примеру $sin(3x)$ и $sin(x)$ у меня достигается простым добавлением обеих порождающих функций. Это не страшно, когда речь идет о линейных суперпозициях (см. ниже) но все же не очень приятно.

2.5. Некоторые теоретические замечания

Хочется отметить, что когда рассматриваются простые суперпозиции сложности 1 (глубины) то это на самом деле очень интересно. Потому что в тех же терминах признакового описания временных рядов больше ничего и не надо.

А плюсы огромны, как минимум стоит упомянуть то что в случае использования линейнх комбинаций матрицы суперпозиций имеют огромное число нулевых элементов (собственно ненулевые это только первый ряд (функция суммы) и последний столбец (столбец переменной)). Что позволяет строить не n^2 классификаторов (n — число порождающих функций) а $O(n)$.