Майнор «Интеллектуальный анализ данных». Вторая консультация по проектам.

Надежда Чиркова

8 июня 2016 г.

1 Этапы решения задачи анализа данных

Вспомним основные этапы решения задачи анализа данных:

1. Подготовка выборки
2. Визуализация данных (поиск особенностей и закономерностей)
3. Предобработка данных
4. Решение задачи:
   (a) Придумывание новых признаков
   (b) Настройка разных моделей (подбор гиперпараметров), выбор лучших моделей
   (c) Отбор признаков
   (d) Построение композиций
   (e) Трансформация предсказаний
   (f) Все это много раз и в разном порядке, пока результаты нас не удовлетворят

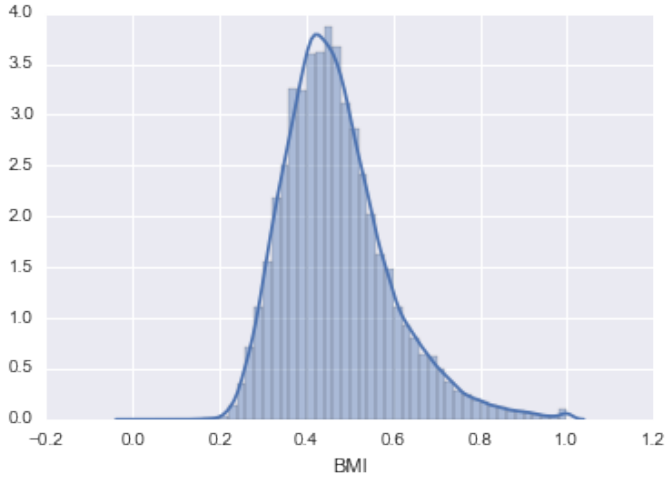
На первой консультации были освещены вопросы 1 и 2 (и они вошли в задание первого модуля), в курсе — вопросы 3, 4(b) (первое задание второго модуля), 4(c) — см. лекцию про отбор признаков, частично 4(d). Сегодня поговорим об оставшихся этапах - 4(a), 4(d), 4(e) которые вошли во вторую часть задания второго модуля.

2 Предобработка данных

Это очень важный этап, поэтому еще раз коснемся его. Что должно быть сделано обязательно:

- Корректное кодирование категориальных признаков (чтобы они не оказались упорядоченными)
- Удаление пропусков
- Стандартизация или масштабирование выборки
- Удаление константных или почти константных признаков (обсуждали еще в первом модуле)
- Удаление сильно коррелированных признаков (тоже связано с визуализацией)

К этапу предобработки данных можно отнести обнаружение аномальных, или шумовых, объектов. Для этого есть специальные методы, однако их непросто настраивать. Наиболее простой метод — через визуализацию. Если признак неплохо описывается каким-нибудь распределением, но в какой-нибудь точке есть нелогичный скачок, такой объект можно удалить. Пример такого явления (скачок в конце):



Все равно таких объектов единицы, и нет смысла на них настраиваться.

3 Feature engineering & selection

Выбор подходящего метода машинного обучения — это только малая часть успешного решения задачи. Гораздо важнее правильно выбрать признаковое пространство.

С одной стороны, в нашей постановке признаки уже даны, и мы не можем просить новую информацию об объектах или новые объекты. Но с другой стороны, ни откуда не следует, что данная нам информация представлена в наиболее подходящем для предсказаний виде.

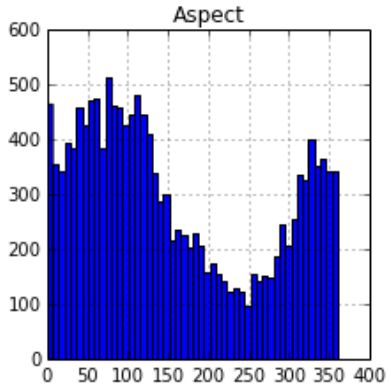
Пример. Пусть мы предсказываем стоимость участка земли, признаки: длина, ширина, район. Логично, что стоимость участка напрямую зависит от его площади, и хоть она не дана, ее можно легко вычислить как произведение длины на ширину.

Итак, мы хотим получить наиболее информативное признаковое представление из наших данных. Помимо логики, можно применять ряд приемов.

Признаки времени. Если известна дата, можно добавлять категориальный признак дня недели, категориальные или номинальные месяц, год.

Бинаризация вещественных признаков Для некоторых вещественных признаков можно указать какой-нибудь значимый порог (или несколько порогов), который делит объекты на две смысловые категории. Например, в задаче предсказания уровня образования признак возраста можно отсекать по порогу 18 — время окончания школы. Конечно, введение порога зависит от задачи.

Иногда порог напрашивается по гистограмме. Пример:



Кластеры. Можно попробовать кластеризовать объекты и добавить номер кластера как **категориальную** переменную.

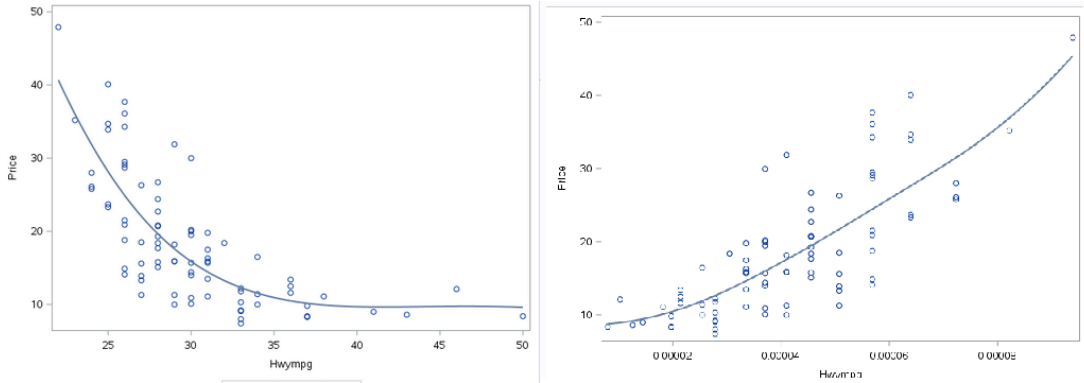
Понижение размерности. Результат применения методов понижения размерности (PCA) можно добавлять как новые признаки.

Изучение групп признаков. Иногда известно несколько похожих признаков, например объект - город, группа признаков - доля расходов бюджета на разные статьи. Тогда имеет смысл создавать агрегированные признаки: минимум / максимум, сумма, среднее, медиана, число ненулевых и т. д. Не забывайте следить, чтобы не образовывалось константных и коррелирующих признаков!

Добавление произведений признаков, квадратов. Если это логично (пример с площадью) или улучшает качество. Осторожно, возможно переобучение, полная загрузка памяти и бесконечное обучение алгоритма!

Нелинейное преобразование признаков. Иногда нелинеййным преобразованием, например $\log(x_i + 1)$ или $\frac{1}{x_i}$ или возведением в степень можно сделать распределение вещественного признака более нормальным. Если это удается, применяйте такое преобразование.

Если вы решаете задачу регрессии, полезно нарисовать скаттеры признак-целевой признак и попробовать применить такие преобразования к каждому признаку, чтобы зависимость стала линейной. Пример (до и после преобразования):



Расстояние до эталонных объектов Если у вас есть возможность выбрать несколько эталонных объектов класса (или получить их как среднее), можно добавить расстояния до них как признак. Это близко к идее добавления меток кластеров и к идее метрических методов.

Внимательное изучение строковых признаков. Строковые признаки сложно обрабатывать напрямую. Но следует внимательно изучить их содержание, там может быть важная информация. Например, по приставке «Mr.» или «Mrs.» можно определить пол.

Преобразование строк к мешку слов или n-грамм. Про мешок слов мы уже говорили. Есть другой интересный подход[1]: превращать текст в n-граммы. Например, в триграммы:

«Сегодня холодно» → сег, его, год, одн, дня, нях, яхо, хол, оло, лод, одн, дно

Такой подход устойчив к опечаткам (поменяются только несколько триграмм) и к изменению формы слова, может оказаться компактнее мешка слов, но нужно настраивать длину n-граммы.

Очевидно, далеко не все добавленные признаки окажутся полезны. Нужны методы отбора признаков.

Вспомните, какие методы отбора признаков рассказывались на лекции.

Жадные методы могут переобучиться, а полный перебор — долго. Рекомендуется использовать регуляризованные линейные модели, обращать внимание на важность признаков, оцениваемую деревьями, случайными лесами и другими методами. Можно использовать корреляционные, одномерные методы.

[1]Подслушано на выступлении Александра Дьяконова

4 Композиции алгоритмов

В курсе не раз упоминалось, что композиции алгоритмов гораздо мощнее одиночных алгоритмов и чаще всего помогают делать более качественные предсказания.

На лекции и семинарах рассматривались классические композиции, которые стали основой известных методов машинного обучения: случайных лесов, градиентного бустинга над решающими деревьями. Особенность этих композиций в том, что они усиливают **большое количество слабых алгоритмов**. Например, решающие деревья слабые, потому что переобучаются.

Есть другая группа композиций, которые усиливают **несколько хороших алгоритмов**. Их мы рассмотрим сейчас. В их основе лежат простые логичные идеи.

Для тех и других композиционных методов важно, чтобы ответы базовых предсказателей были некоррелированы, то есть не похожи друг на друга. Иначе композиция не даст сильного выигрыша. Так происходит, если базовые алгоритмы и так очень хорошо решают задачу (относительно того, насколько ее вообще теоретически можно хорошо решить).

Пусть у нас есть обучающая выборка X и вектор правильных ответов y, тестовая выборка $\tilde{X}$, несколько алгоритмов, обученных на X и y, и предсказания $\tilde{y}^1, \dots, \tilde{y}^n$ этих алгоритмов на $\tilde{X}$. Нужно из этих векторов составить итоговый вектор предсказаний $\tilde{y}$.

Простое голосование или усреднение. Это наиболее простой способ сделать композицию. Для задачи классификации на K классов предсказание для $\tilde{x}_i$:

$$\tilde{y}_i = \max_{k=1,\dots,K} \sum_{j=1}^{n} [\tilde{y}_i^j = k],$$

то есть мы выбираем класс, который чаще всего встречается среди предсказаний для объекта с номером i.

Для задачи регрессии еще проще:

$$\tilde{y}_i = \frac{1}{n} \sum_{i=1}^{n} \tilde{y}_i^j.$$

Если базовые классификаторы предсказывают вероятность класса 1, то для них также применяется усреднение.

Задача. Поясним, почему это работает. Пусть у нас есть выборка, на которой все правильные ответы 1 (объектов класса 0 в тестовой выборке нет). Даны три базовых бинарных классификатора, каждый из которых на тестовой выборке выдает 70% единиц (считаем, что в случайных позициях). Найдиет вероятность того, что классификатор простого голосования отнесет объект к классу 1.

Решение. Посчитаем вероятности:

Все три классификатора выдают единицу в позиции i:	$0.7 \cdot 0.7 \cdot 0.7 = 0.3429$
Два классификатора выдают единицу в позиции i	$3 \cdot 0.7 \cdot 0.7 \cdot 0.3 = 0.4409$
Один классификатор выдает едиинцу в позиции i	$3 \cdot 0.7 \cdot 0.3 \cdot 0.3 = 0.189$
Все ошибаются	$0.3 \cdot 0.3 \cdot 0.3 = 0.027$

Вероятность корректного ответа композиционного классификатора: $0.3429+0.4409 = 0.7838$, что больше вероятности правильного ответа одного базового классификатора!

Стекинг и блендинг Разумный следующий шаг — усреднять предсказания алгоритмов с весами:

$$\tilde{y}_i = \frac{1}{n}\sum_{i=1}^{n} w^j \tilde{y}_i^j. \tag{1}$$

Желательно, чтобы такая линейная комбинация была выпуклой: $w^j \geqslant 0$, $\sum_j w^j = 1$.

Как настроить веса?

Обучить линейную регрессию!

Но нужно не переобучиться. Поэтому нужно выделить часть выборки для валидации. Сначала мы обучаем базовые классификаторы (регрессоры) на 80% выборки обучающей выборки X, далее делаем предсказания для оставшихся 20% X. Эти предсказания считаем новыми признаками на 20% X и настраиваем $w^1, \ldots, w^n$. Наконец, делаем прогноз для $\tilde{X}$: настраиваем базовые классификаторы (регрессоры) и используем формулу (1).

Не забудьте перемешать выборку!

Этот подход к построению композиции называется **блендинг**.

Есть похожий способ — **стекинг**. Он несколько похож на кросс-валидацию. Мы делим выборку X на K блоков. Затем по очереди исключаем каждый блок, на оставшейся выборке обучаем базовые классификаторы (регрессоры). Делаем предсказания всех базовых алгоритмов на выбранном блоке. В итоге получаем предсказания для всех объектов обучающей выборки. Их используем как признаки в линейной регрессии для итогового предсказания.

Таким образом, в стекинге мы обучаем линейную регрессию не на валидационной выборке, а на всей X. Но приходится обучать базовые алгоритмы несколько раз.

Оба подхода очень популярны при решении конкурсных задач. Рекомендуем вам выбрать несколько хорошо настроенных алгоритмов решения вашей проектной задачи и сделать на них блендинг или стекинг (если выборка не очень большая).

Что делать, если $w^j \leqslant 0$? Следует исключить такой алгоритм из модели и заново настроить веса.

Что делать, если сумма весов не равна единице? Это странно, потому что такие ответы смещенные. Можно попробовать нормировать вектор весов к единице (разделить его компоненты на сумму вектора) и оценить качество на отложенной выборке.

Пример поиска коэффициентов w^j в python (y_i — предсказание на валидационной выборке):

```
from sklearn.linear_model import LinearRegression
blender = LinearRegression()
blender.fit(np.vstack((y_1, y_2, y_3)).T, y_true)
```

```
blender.coef_
```

5 Трансформация предсказаний

В конце предыдущей секции был приведен пример, когда предсказания итоговой модели могут быть смещенные. К чему это приводит? К тому, что распределение ответов на тестовой выборке не совпадает с распределением ответов на обучающей выборке.

А распределения должны совпадать, если объекты разделены на обучение и контроль случайно. Это очень просто проконтролировать: достаточно построить две гистограммы ответов.

Если распределения не совпадают, нужно применить трансформацию к предсказанию (некоторую функцию), чтобы распределения стали похожи. Если речь идет о задаче классификации, то трансформировать нужно вещественные предсказания — вероятности классов.

Есть другой случай, когда необходимо преобразование: если вы предсказываете не исходную целевую переменную, а некоторую обратимую функцию от нее (например, логарифм). Тогда в конце решения задачи нужно не забыть вернуться к исходному виду целевой переменной (в примере применить функцию экспоненты к предсказаниям). Эта процедура может быть осмысленна например в случае, когда преобразование делает распределение целевого признака больше похожим на нормальное.

6 Один из главных врагов датамайнера — переобучение!

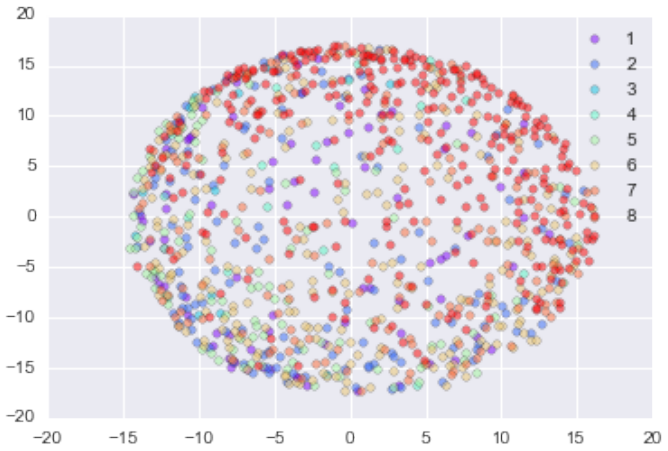
Поэтому самое главное для вас — не переобучиться. Ни в коем случае не оценивайте работу алгоритма на данных, которые хоть чуть-чуть участвовали в обучении. В идеале отложить часть выборки (например, 20%) отдельно и узнать качество на них только в конце работы над проектом. Эта работа уже сделана для тех, кто предсказывает зарплату по объявлениям.

Если вы решили делать композицию блендингом, то нужно искать веса базовых алгоритмов на валидационной выборке. Если вы уже использовали всю обучающую выборку для тестирования алгоритмов, то разделите вашу тестовую на две части — валидация (для построения композиции) и собственно тестовая (на которой оценивается качество в самом конце). Итак, для блендинга у вас должны быть три выборки: обучающая, валидационная и тестовая. На первой вы пробуете алгоритмы (подбираете гиперпараметры каждого алгоритм кросс-валидацией) и выбираете лучшие по качеству на кросс-валидации, на валидационной — подбираете коэффициенты базовых алгоритмов, на тестовой — получаете итоговое значение метрики качества (просто делаете предсказания и вычисляете метрику).

Для усреднения и простого голосования шаг с блендингом пропускается, вы все делаете на обучающей выборке.

7 Как оценить, насколько хорошо задача вообще может быть решена, или вновь визуализация

Примените метод MDS или t-SNE для визуализации своих данных. Это нелинейные методы понижения размерности, которые часто используют для отображения выборки в двумерное пространство (на плоскость). Ключевая особенность MDS в том, что он стремится сохранить расстояния между объектами при отображении их в плоскость. Идея t-SNE похожа. Оба метода реализованы в sklearn. Пример работы MDS с косинусной метрикой:



В этой задаче классы пронумерованы от 1 до 8 (упорядочены). Мы видим, что классы 1 и 8 хорошо разделяются, а остальные распределены между ними.

Если при визуализации ваши классы почти никак не разделяются, то и решать эту задачу вы, скорее всего, сможете с очень плохим качеством. Если классы разделяются хорошо, то и качество решения задачи будет хорошее.

Чтобы применить метод к регрессии, можно выделить несколько групп значений целевого признака (только для визуализации!) и свести задачу к предсказанию номинального признака.

Применение MDS с косинусной метрикой:

```
from sklearn.manifold import MDS
from sklearn.metrics.pairwise import pairwise_distances

mds = MDS(metric="precomputed")
MDS_transformed_cos = mds.fit_transform(pairwise_distances(data, metric="cosine"))
```

С евклидовой метрикой — просто создать объект `MDS()` передать data как параметр fit. Учтите, что MDS работает не очень быстро, и ему стоит подавать не больше 5000 объектов.