Соревнования и линейные методы

Евгений Соколов
sokolov.evg@gmail.com

11 Февраля 2015

Содержание

Данные

Задача: предсказание зарплаты по тексту объявления.

Признаки:

- название должности
- текстовое описание вакансии
- регион (**есть иерархия регионов!**)
- вид контракта (постоянный/временный, full-/part-time)
- категория вида работы
- название и электронный адрес работодателя

Целевая переменная: годовая зарплата.

Метрика качества: Mean Absolute Error (MAE)

$$\text{MAE} = \sum_{i=1}^{\ell} |a(x_i) - y_i|.$$

Правила

Срок: до 10 марта.

Баллы:

- 1-е место — 15 баллов
- 2-е место — 13 баллов
- 3-е место — 11 баллов
- Остальные места выше бейзлайна — от 1 до 10 баллов по равномерной сетке
- Если решения будут тривиальными, то максимальный балл будет уменьшен

Форма отчетности:

- Код, воспроизводящий решение
- Краткий отчет
- Для занявших первые три места — доклад на семинаре

Leaderboard и форум

Таблица результатов:

- Результаты в таблице вычисляются по 30% тестовой выборки
- Опасно настраиваться на это число, появляется риск переобучения

На странице конкурса есть форум!

- Можно делиться интересными идеями или кодом
- Активность будет поощряться

Какие у нас данные?

В основном у нас есть тексты и категориальные признаки.

- Категориальный признак x_{ij} с областью значений $\{c_1, \ldots, c_m\}$ кодируется бинарным вектором

$$([x^j = c_k])_{k=1}^{m}.$$

- Текстовый признак x_{ij} — это множество $\{w_1, \ldots, w_{n_i}\}$, который можно закодировать бинарным вектором

$$([w_k \in x_{ij}])_{k=1}^{|W|},$$

где $W = \{w_1, \ldots, w_{|W|}\}$ — множество всех возможных слов.

- В обоих случаях получаем разреженные признаки.

Что делать с разреженными признаками?

На разреженных данных хорошо работают линейные методы!

- Линейная регрессия (L2, L1, ElasticNet) — есть в Vowpal Wabbit;
- Support Vector Regression — `sklearn.svm.SVR`.

Но не стоит забывать про другие подходы:

- kNN — может хорошо заработать при грамотной настройке метрики
- Gradient Boosted Decision Trees — XGBoost, `sklearn.ensemble.GradientBoostingRegressor`
- Random Forest — `sklearn.ensemble.RandomForestRegressor`

Работа с текстами

Тексты можно предобрабатывать:

- удалять редкие/частотные слова
- делать стэмминг или лемматизацию
- **Hashing Trick**: заменить каждое слово w на $h(w)$, где h — хэш-функция с 2^b возможными значениями; получаем «кластеризацию» слов, если $2^b < |W|$.
- кластеризовать по-умному: например, объединить все редкие слова в одно.
- строить тематическую модель.

Работа с текстами

Можно генерировать новые признаки:

- индикаторы вхождения всех пар слов:

$$[w_k \in x_{ij}][w_l \in x_{ij}]$$

- n-граммы — индикаторы того, что данные два слова встретились рядом; для текста «мама мыла раму» получаем биграммы «мама мыла» и «мыла раму»
- k-skip-n-граммы — как n-граммы, только разрешаем словам быть отдаленными не больше чем на k

Blending

Алгоритмы можно объединять в один:

- Пусть даны два алгоритма $b_1(x)$ и $b_2(x)$
- Построим их выпуклую комбинацию:

$$a(x) = \alpha b_1(x) + (1 - \alpha) b_2(x); \quad \alpha \in [0, 1]$$

- Параметр α настраивается с помощью кросс-валидации
- Можно объединять и больше алгоритмов
- Пример: линейная модель и случайный лес
- Почти всегда решения-победители на Kaggle представляют собой линейную комбинацию нескольких алгоритмов

Stacking

Выход одних алгоритмов можно подавать на вход другим:

- Известно, что GBDT плохо работает над разреженными признаками
- Можно сначала обучить на них линейную регрессию, а ее выход использовать как признак в GBDT
- Есть риск переобучения — линейную регрессию лучше обучать на подвыборке

Feature Engineering

Самое ценное в анализе данных — умение придумывать признаки!

- Подумайте, по каким признакам вы сами стали бы прогнозировать зарплату
- Читайте статьи
- Читайте форумы на kaggle

Содержание

Что такое Vowpal Wabbit

- очень качественная реализация стохастического градиентного спуска для линейных моделей
- считывает с диска по одному объекту и делает шаг только по нему, нет необходимости хранить выборку в памяти
- может быть запущен на кластере
- нормализация признаков, взвешивание объектов, адаптивный градиентный шаг
- матричные разложения, тематическое моделирование, активное обучение, обучение с подкреплением
- разнообразие методов оптимизации: сопряженные градиенты, квазиньютоновские методы (L-BFGS)

Входные данные

Одна строка — один объект:

```
123 10 | 1:0.43 5:2.1 age:20 some raw text here
```

- 123 — целевая переменная
- 10 — вес объекта (можно не указывать, по умолчанию 1)
- `name:value` — описание признака
  - если `name` — строка, то она хэшируется (см. Hashing Trick)
  - по умолчанию `value`=1
  - если признак не описан для данного объекта, то он считается равным нулю

Пространства признаков

Признаки можно разделять на группы:

```
123 10 |integer 1:0.43 5:2.1 age:20 |text some raw text here age:120
```

- `integer` и `text` — два пространства признаков
- в обоих пространствах есть признак age, так можно

Обучение

Пусть выборка записана в файле `train.txt`.

Обучение:

```
vw -d train.txt --passes 10 -c -f model.vw
```

- `-d filename` — имя входного файла
- `--passes n` — количество проходов по выборке
- `-c` — включает кэширование, позволяет ускорить все проходы после первого
- `-f filename` — имя файла с моделью

Применение

Как получить прогноз:

```
vw -d test.txt -i model.vw -t -p predictions.txt
```

- `-d filename` — имя входного файла
- `-i filename` — имя файла с моделью
- `-t` — режим применения существующей модели
- `-p filename` — имя файла с прогнозами

Функционалы

Поддерживаемые функционалы (`--loss_function`):

- squared
$$\frac{1}{2}(y - a(x))^2$$
- classic — quadratic без перевзвешивания объектов
- quantile (обратите внимание!)
$$\tau(a(x) - y)[y \leqslant a(x)] + (1 - \tau)(y - a(x))[y \geqslant a(x)]$$
- logistic
$$\log(1 + \exp(-ya(x)))$$
- hinge
$$\max(0, 1 - ya(x))$$

Регуляризация

К функционалу можно добавить регуляризаторы:

- `--l1 coef`
- `--l2 coef`

Можно обучать SVM:

```
vw -d train.txt -f svm.vw --loss_function hinge --l2
0.1
```

Градиентный спуск

Градиентный шаг:

$$w^{(t+1)} = w^{(t)} - \alpha_t \nabla Q(x_{i_t}).$$

Как выбирать α_t?

$$\alpha_t = s \left(\frac{i}{i+t} \right)^p,$$

где

- `-l s`
- `--initial_t i`
- `--power_t p`

Эти параметры сильно влияют на качество!

Работа с признаками

- `-b n`: логарифм количества возможных значений хэш-функции для hashing trick
- `-q ab`: генерирует все парные признаки, где первый признак берется из пространств с именем «a*», второй — из «b*»
- `--cubic abc`: тройки признаков
- `--ngram an`: генерирует n-граммы для пространств «a*»
- `--skips ak`: разрешает делать пропуски длины k в n-граммах пространств «a*»

Что еще?

Читайте документацию!

- `--holdout_after`: ранний останов, использует отложенную выборку
- `--bfgs`: квазиньютоновская оптимизация, должна работать лучше
- `--ksvm`: SVM с kernel trick

Напутствие

- На слайдах было много идей — экспериментируйте!
- Не забывайте настраивать параметры в ваших алгоритмах
- Изучайте данные, делитесь находками
- Если используете нестандартные алгоритмы или библиотеки — расскажите всем

Удачи!