Прикладные задачи анализа данных

Методы решения задач классификации на два класса с категориальными признаками

Дьяконов А.Г.

Московский государственный университет
имени М.В. Ломоносова (Москва, Россия)

Терминология:

Категориальные признаки

Факторные

Номинальные

Примеры:

профессия,

должность,

идентификационный номер,

номер группы студента,

тарифный план,

издательство,

область науки

Сами значения признаков бесполезны!

Имеет смысл лишь операция сравнения…

Прикладные задачи:

1. Рекомендательная система для службы безопасности

пользователь	отдел	доступ	ресурс	разрешение
Иванов	07	A	Б316	+
Петров	06	A	Б316	+
Сидоров	06	C	Б315	-

Номер признака	1	2	3	4	5	6	7	8
Число категорий	7518	4243	128	177	449	343	2358	67

2. Анализ конверсии

источник	ссылка	браузер	время	CA
google	none	chrome	12:13:40	-
yandex	none	opera	12:13:42	+
ozon	www.ozon.ru/id218	opera	12:14:02	-

Код: автоматическое определение категориальности

- если значения – строки
- если мало уникальных значений

	city	class	degree	income
0	Moscow	A	1	10.2
1	London	B	1	11.6
2	London	A	2	8.8
3	Kiev	A	2	9.0
4	Moscow	B	3	6.6
5	Moscow	B	3	10.0
6	Kiev	A	1	9.0
7	Moscow	A	1	7.2

```
# найти все признаки, в которых первое значение - строка
def find_cat(data):
    for name in data.columns:
        s = ''
        s += name
        if (type(data[name][0]) == str):
            s += ' строка,'
        if (data[name].nunique()<=3):
            s += ' мало уникальных'
        if (s!=name):
            print (s)

find_cat(data)
```

```
city строка, мало уникальных
class строка, мало уникальных
degree мало уникальных
```

Что делать с категориальными признаками

Что делать?

- **ONE-кодирование**
- **ONE + SVD**
- **Вещественное кодирование**

Какие могут быть проблемы?

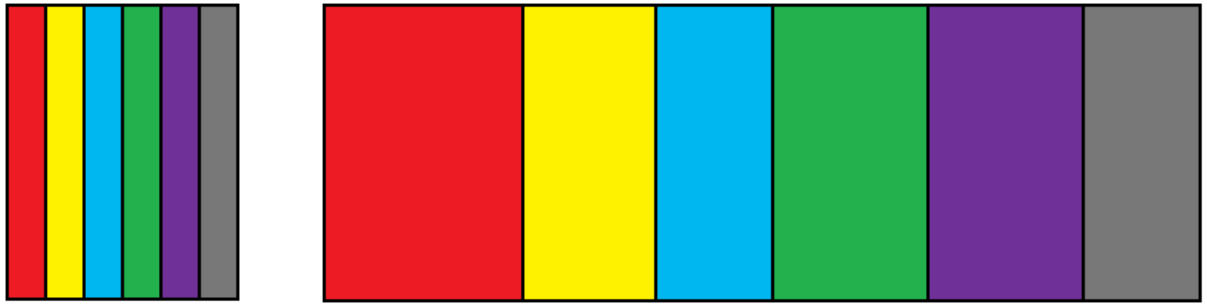
- **Проблема малых категорий**
- **Проблема большого признакового пространства**
- **Нужны «взаимодействия» признаков**

One-hot-кодировки (Dummy)

Должность	
асс.	1
доц.	2
проф.	3
зав.каф.	4
декан	5

Должность	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
1	1	0	0	0	0
1	1	0	0	0	0
3	0	0	1	0	0
2	0	1	0	0	0
2	0	1	0	0	0
4	0	0	0	1	0

Проблема:



Код: Dummy-кодирование

- sklearn.preprocessing.OneHotEncoder
- через сравнения
- pandas.get_dummies

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(sparse=False)
new_ohe_features = ohe.fit_transform(data.city_le.values.reshape(-1, 1))
tmp = pd.DataFrame(new_ohe_features, columns=['city=' + str(i) for i in range(new_ohe_features.shape[1])])
data = pd.concat([data, tmp], axis=1)
data
```

	city	class	degree	income	city=0	city=1	city=2
0	Moscow	A	1	10.2	0	0	1
1	London	B	1	11.6	0	1	0
2	London	A	2	8.8	0	1	0
3	Kiev	A	2	9.0	1	0	0
4	Moscow	B	3	6.6	0	0	1
5	Moscow	B	3	10.0	0	0	1
6	Kiev	A	1	9.0	1	0	0
7	Moscow	A	1	7.2	0	0	1

```
# ручной способ
def code_myohe(data, feature):
    for i in data[feature].unique():
        data[feature + '=' + i] =
            (data[feature] == i).astype(float)
code_myohe(data, 'city')
data
```

Проблемы реализации

OneHotEncoder по умолчанию порождает sparse-ответ

OneHotEncoder работает только с числами
(ручное решение и со строками)

Проблемы глобальные

Число признаков может быть большим.

Но:

Признаки бинарные,
Матрица разреженная

(хранят «положения единиц»,
есть специальные методы)

Конъюнкции признаков

f	*g*
1	2
1	1
0	2
2	1
2	1

f & *g*
(1,2)
(1,1)
(0,2)
(2,1)
(2,1)

перенумерация
1
2
3
4
4

Получаем новый факторный признак

При добавлении конъюнкций – разрастание пространства

Порядок конъюнкции	1	2	3	4
Число признаков	15283	205298	882617	2076324

Код: конъюнкция признаков

	city	class	degree	income	city + degree
0	Moscow	A	1	10.2	Moscow + 1
1	London	B	1	11.6	London + 1
2	London	A	2	8.8	London + 2
3	Kiev	A	2	9.0	Kiev + 2
4	Moscow	B	3	6.6	Moscow + 3
5	Moscow	B	3	10.0	Moscow + 3
6	Kiev	A	1	9.0	Kiev + 1
7	Moscow	A	1	7.2	Moscow + 1

```
# конъюнкция двух признаков
def make_conj(data, feature1, feature2):
    data[feature1 + ' + ' + feature2] =
            data[feature1].astype(str) +
            ' + ' +
            data[feature2].astype(str)

    return (data)

# пример использования
make_conj(data, 'city', 'degree')
```

Очень простое решение!

Код: как делать sparse-матрицу в Python

```
from sklearn.preprocessing import LabelEncoder
FinelineNumber = LabelEncoder().fit_transform(data['FinelineNumber'].fillna(-1))
DepartmentDescription = LabelEncoder().fit_transform(data['DepartmentDescription'].fillna('n'))

from scipy.sparse import csc_matrix
S1 = csc_matrix((data.ScanCount, (data.VisitNumber, DepartmentDescription))).astype(numpy.float64)
S2 = csc_matrix((data.ScanCount, (data.VisitNumber, FinelineNumber))).astype(numpy.float64)

# вектор ответов и индексы обучения
y = data.groupby('VisitNumber')['TripType'].first()
indextrain = y.index.values
y = y.values

encodery = LabelEncoder()
y = encodery.fit_transform(y)

# контроль по фолдам - логистическая регрессия
from sklearn.cross_validation import cross_val_score
from sklearn.cross_validation import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from scipy.sparse import hstack
print (cross_val_score(LogisticRegression(), hstack([S1[indextrain,:], S2[indextrain,:]]), y,
cv=StratifiedKFold(y, n_folds=2), scoring='log_loss'))
```

Просто кодировки вещественными числами

Должность	*f*	*g*
асс.	19	25
доц.	25	35
проф.	30	68
зав.каф.	34	71
декан	50	67

Один признак можно кодировать несколько раз…

но как?

Просто кодировки вещественными числами

Должность	f	g
асс.	19	25
доц.	25	35
проф.	30	68
зав.каф.	34	71
декан	50	67

Один признак можно кодировать несколько раз…

- просто номерами категорий (установить какой-то порядок)
- несколько раз случайными номерами
- по мощности категорий (самая естественная кодировка)
- по другим вещественным признакам
- по другим факториальным признакам
- байесовское кодирование

Код: Простейшее кодирование – по номеру категории

- sklearn.preprocessing.LabelEncoder
- dict + map

	city	class	degree	income	city_le
0	Moscow	A	1	10.2	2
1	London	B	1	11.6	1
2	London	A	2	8.8	1
3	Kiev	A	2	9.0	0
4	Moscow	B	3	6.6	2
5	Moscow	B	3	10.0	2
6	Kiev	A	1	9.0	0
7	Moscow	A	1	7.2	2

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(data.city)
data['city_le'] = le.transform(data.city)
data
```

```
# ручная альтернатива
# словарь для кодировки
dct = {'Kiev': 0, 'London': 1, 'Moscow': 2}
data['city_le'] = data['city'].map(dct)
data
```

```
train_d['Manager_Gender'] = train_d['Manager_Gender'].map({'M':1, 'F':-1, nan:0})
```

Код: По значениям категориального признака

- **просто по мощности – одна строка**

```
feature = 'city'
newfeature = 'city_c'
data[newfeature] = data[feature].map(data.groupby(feature).size())
data
```

	city	class	degree	income	city_c
0	Moscow	A	1	10.2	4
1	London	B	1	11.6	2
2	London	A	2	8.8	2
3	Kiev	A	2	9.0	2
4	Moscow	B	3	6.6	4
5	Moscow	B	3	10.0	4
6	Kiev	A	1	9.0	2
7	Moscow	A	1	7.2	4

Кодирование относительно других (вещественных) признаков

книга	120
книга	200
книга	180
принтер	7000
принтер	3500

Книга
(120+200+180)/3

Принтер
(7000+3500)/2

Интерпретация

Средняя цена в категории

Максимальная цена

Дисперсия цен

Число скидок в категории

Но: у нас могут быть не вещественные признаки

Код: кодировка по значениям вещественного признака

```
# функция возвращает значения нового признака
def code_mean(data, cat_feature, real_feature):
    return (data[cat_feature].map(data.groupby(cat_feature)[real_feature].mean()))

data['city_mean_income'] = code_mean(data, 'city', 'income')
data
```

	city	class	degree	income	city_mean_income
0	Moscow	A	1	10.2	8.5
1	London	B	1	11.6	10.2
2	London	A	2	8.8	10.2
3	Kiev	A	2	9.0	9.0
4	Moscow	B	3	6.6	8.5
5	Moscow	B	3	10.0	8.5
6	Kiev	A	1	9.0	9.0
7	Moscow	A	1	7.2	8.5

Методы решения задач с категориальными признаками

Как решать?

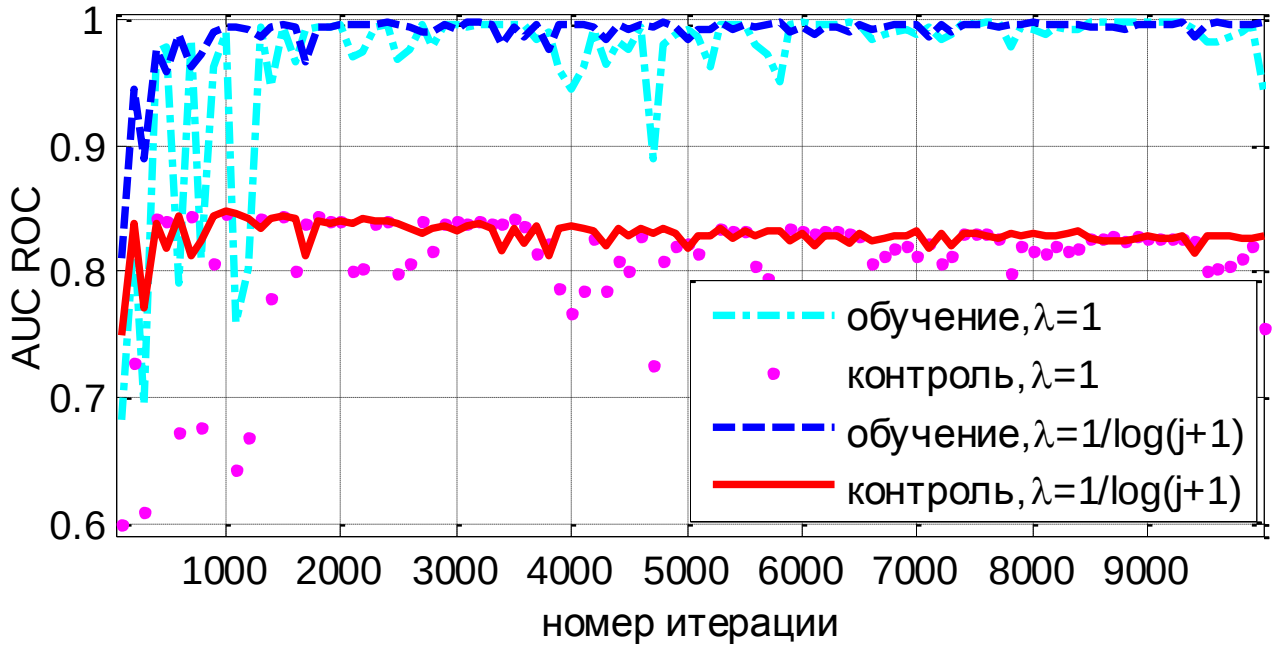
Линейные методы

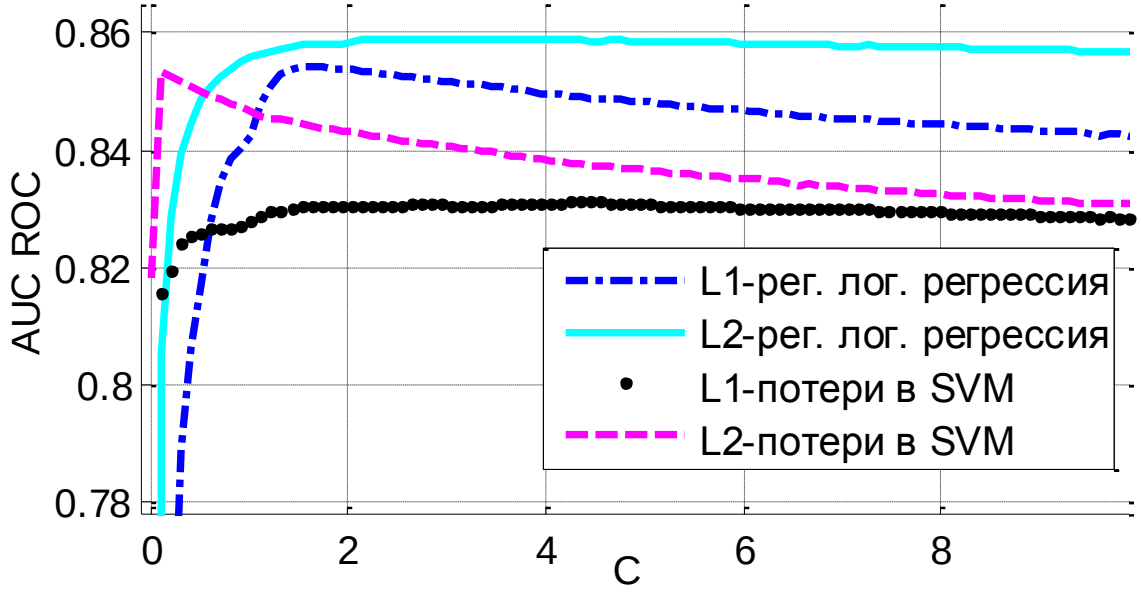
$$L = w_1 f_1 + \ldots + w_n f_n$$

f_i **– не обязательно исходные признаки**
(м.б. их кодировки, конъюнкции и т.п.)

Вспоминаем…

$$(w_1,\ldots,w_n) = (w_1,\ldots,w_n) + \lambda \sum_{i=1}^{m} \left(y_i - \frac{1}{1+e^{-(w_1 f_1+\ldots+w_n f_n)}} \right) (f_1,\ldots,f_n)$$


AUC ROC
1
0.9
0.8
0.7
0.6
1000 2000 3000 4000 5000 6000 7000 8000 9000
номер итерации
обучение, λ=1
контроль, λ=1
обучение, λ=1/log(j+1)
контроль, λ=1/log(j+1)


AUC ROC
0.86
0.84
0.82
0.8
0.78
0 2 4 6 8
C
L1-рег. лог. регрессия
L2-рег. лог. регрессия
L1-потери в SVM
L2-потери в SVM

Качество алгоритмов пакета LIBLINEAR

после one-hot-кодирования признаков

Порядок конъюнкции	1	2	3	4
ROC AUC	0.8591	0.8703	0.8713	0.8704

- Маленькие категории имеет смысл удалять (схлопывать в одну)
- Некоторые категории могут быть только в контроле / в обучении

Байесовская кодировка

Чем плохая?

1	+	0.75	+
1	+	0.75	+
1	-	0.75	-
1	+	0.75	+
0	-	0.5	-
0	+	0.5	+

- **содержит данные об ответе!**

1	2
A	E
A	D
A	E
B	D
B	F
C	F
C	E

1	2
0.0	0.33
0.0	0.00
0.0	0.33
0.5	0.00
0.5	0.50
0.5	0.50
0.5	0.33

Какой ответ (целевой вектор)?

Байесовская кодировка

Мнимый выход – LOO-кодировка

1	1
A	0.66
A	0.66
A	0.66
A	1.00
B	0.00
B	0.50
B	0.50

Проблема с контролем:

A	0.75
B	0.33

Какой ответ (целевой вектор)?

Стало ещё хуже!

Байесовская кодировка

А ведь это имеет отношение к стекингу;)

Выход:

- **разделение выборки (для кодирования / для обучения)**
- **«регуляризация»**
- **добавление случайного шума**

Байесовские алгоритмы – как можно решать

Перекодировка значений признаков в оценку вероятности

$$g_j = \frac{|I_j(f_j) \cap Y_1| + \Delta_j \cdot c}{|I_j(f_j)| + c}$$

м.б. с регуляризацией Лапласа

Значения признака может не быть в обучении…

1	+	0.75	+
1	+	0.75	+
1	-	0.75	-
1	+	0.75	+
0	-	0.5	-
0	+	0.5	+

$$L(f_1, \ldots, f_n) = \frac{\sum_{\substack{j=1 \\ f_j \in F_j}}^{n} w_j \varphi(g_j)}{\sum_{\substack{j=1 \\ f_j \in F_j}}^{n} w_j}$$

Байесовские алгоритмы – чуть лучше...

в признаковом пространстве

$$(\varphi(g_1),\ldots,\varphi(g_n),\gamma_1,\ldots,\gamma_n)$$

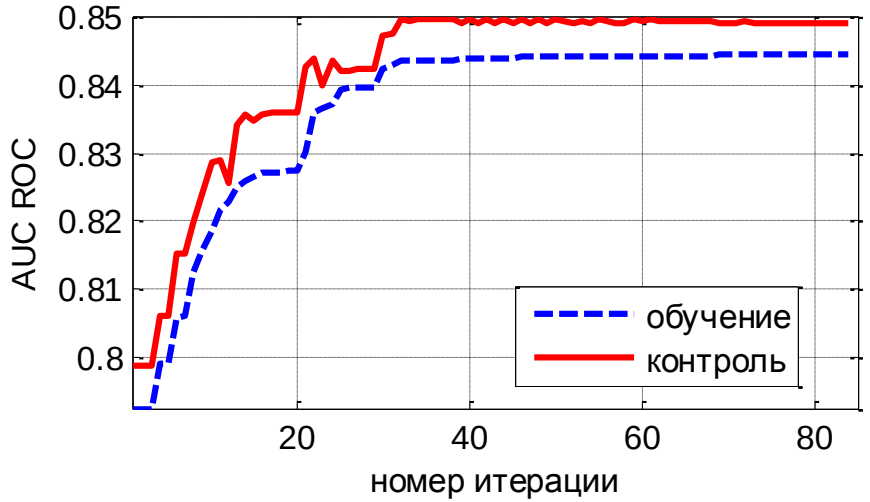
$\varphi(\cdot)$ – **преобразование признака**

Если используем log, то сумма – произведение в наивном Байесе

$$\gamma_j = \begin{cases} 0, & f_j \in F_j, \\ 1, & f_j \notin F_j \end{cases}$$

– характеристический вектор «известности значения признаков»

Настройка покоординатным спуском



Байесовские алгоритмы

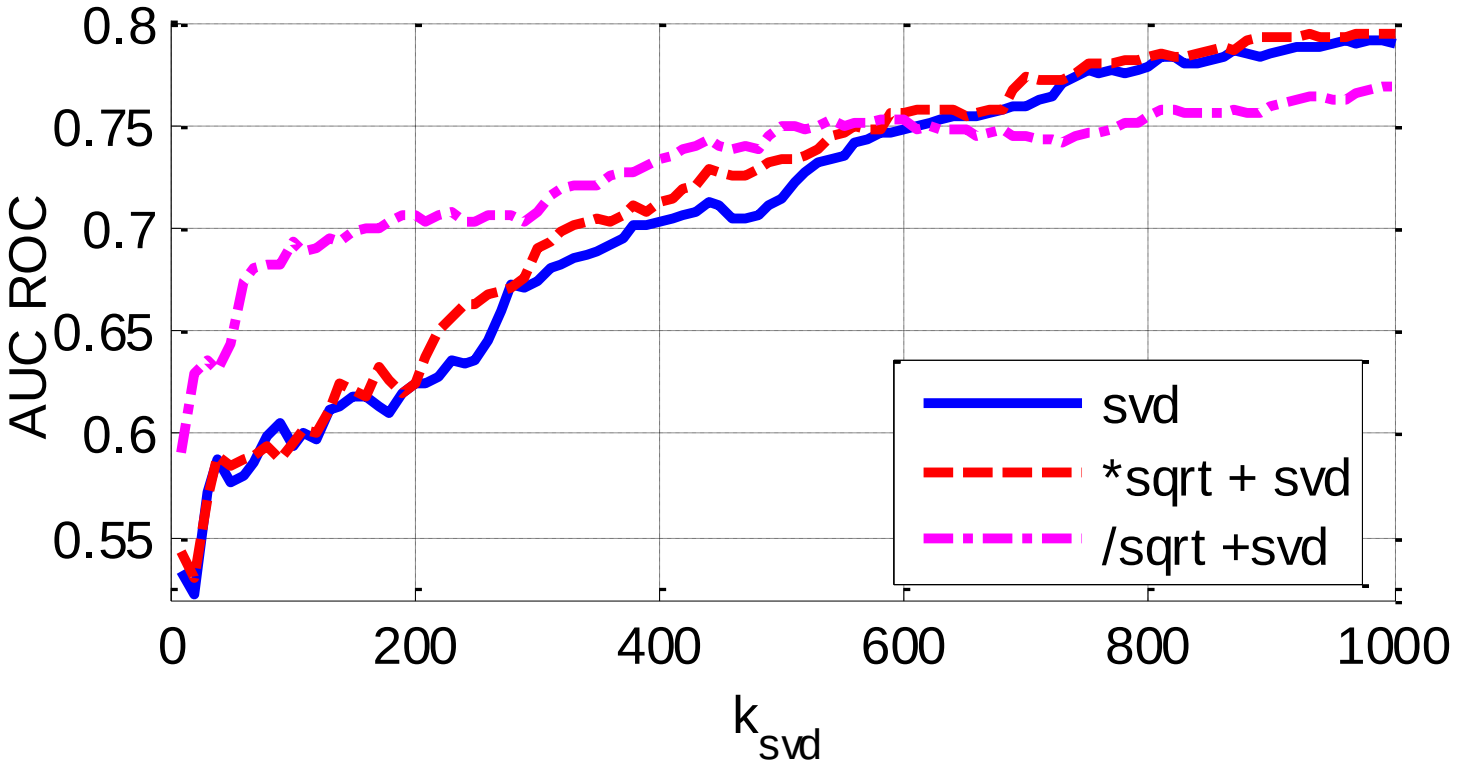
Порядок конъюнкции	1	2	3	4	5
x-призн.пр-во	0.8491	0.8814	0.8864	0.8878	0.8868
x-Лаплас	0.8475	0.8746	0.8801	0.8834	0.8842
log-призн.пр-во	0.8101	0.8234	0.8247	0.8242	0.8246
sqrt-Лаплас	0.8465	0.8726	0.8790	0.8809	0.8821

Лучше не логарифмировать…

AUC оптимизирует сумма вероятностей!

Методы, основанные на сингулярном разложении матрицы бинарных признаков

$$F' \approx U\Lambda V \qquad Uw = Y \qquad w = (U^{\mathrm{T}}U + \lambda I)^{-1}U^{\mathrm{T}}Y$$



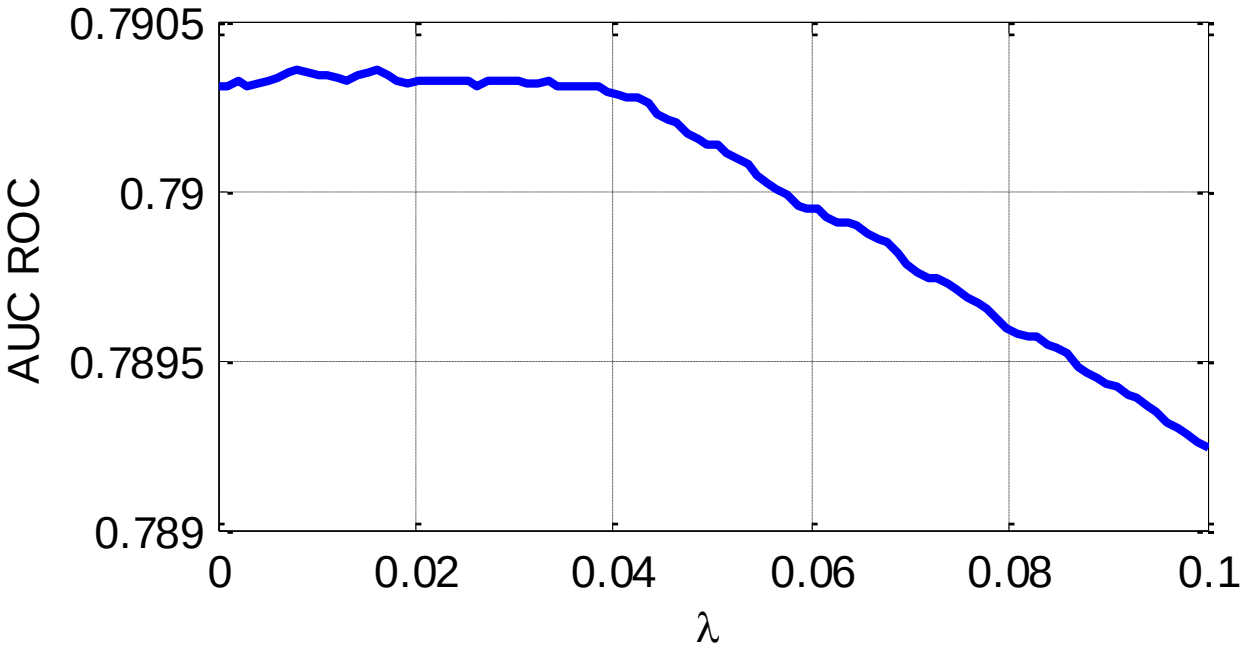
Качество от числа слагаемых в SVD

Дальше – больше 1 часа

Хорошо умножать строку на корень из суммы строки!

Методы, основанные на сингулярном разложении матрицы бинарных признаков

Качество от коэффициента регуляризации



Методы, основанные на близости

$$\frac{\sum_{i=1}^{m} r_i y_i}{\sum_{i=1}^{m} r_i}$$

обобщение взвешенного метода ближайших соседей –

k совпадает с числом объектов в обучении

$$r_i = \left(\sum_{\Omega \in \Omega^*} w_\Omega \prod_{j \in \Omega} I[f_j = f_{ij}] \right)^d$$

веса зависят от сходства объектов

Степень конъюнкции	1	2	3	4
Качество	0.8681	0.8884	0.8900	0.8919

Методы, основанные на близости

- «ленивая модель» алгоритмов (для классификации необходимо хранить всю обучающую выборку)

- процедура настройки параметров достаточно долгая

- При использовании конъюнкций – качество растёт, но растёт и время настройки

Методы, основанные на тензорных разложениях

$$\begin{bmatrix} f_{11} & f_{12} \\ \vdots & \vdots \\ f_{m1} & f_{m2} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

информация о значениях матрицы

$$Z = \| z_{ij} \|_{n_1 \times n_2}$$

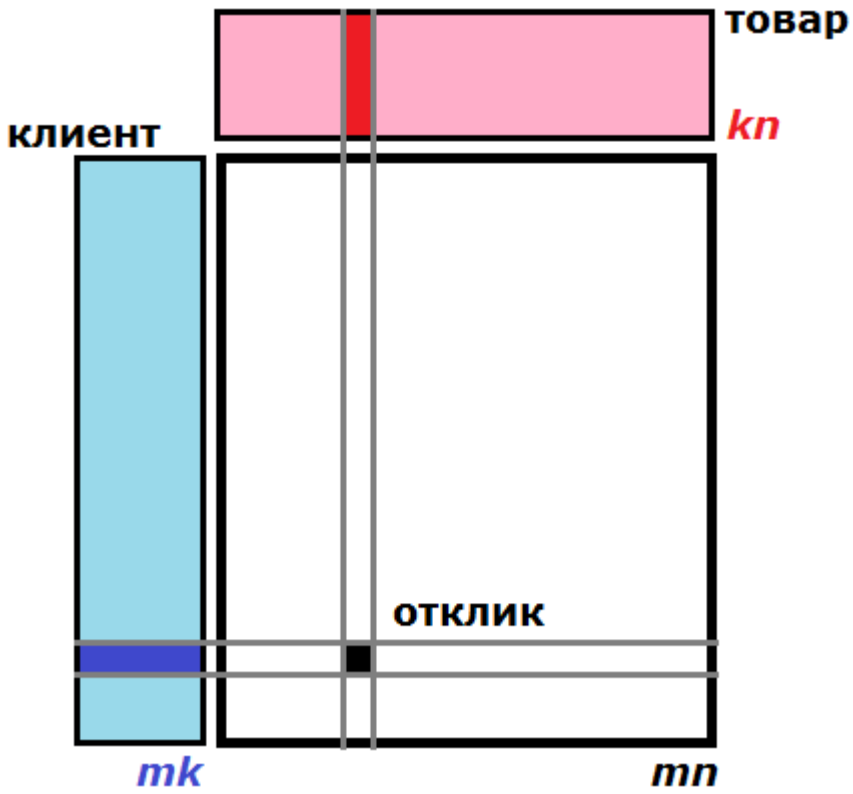
$$z_{f_{t1}, f_{t2}} = y_t$$

$$Z = UV$$

f	g	
1	2	1
2	1	1
1	3	0
2	2	0

	1	2	3
1		1	0
2	1	0	

Идея разложения



Методы, основанные на тензорных разложениях

$$J = \sum_{t=1}^{m} e_t^2 + \lambda_1 \sum_{s=1}^{k} \sum_{i=1}^{n_1} u_{is}^2 + \lambda_2 \sum_{s=1}^{k} \sum_{j=1}^{n_2} v_{sj}^2$$

$$e_t = \sum_{s=1}^{k} (u_{f_{t1},s} v_{s,f_{t2}}) - y_t$$

Аналогично в многомерном случае – но там тензор!

Как минимизировать?

Как минимизировать…

1. SGD

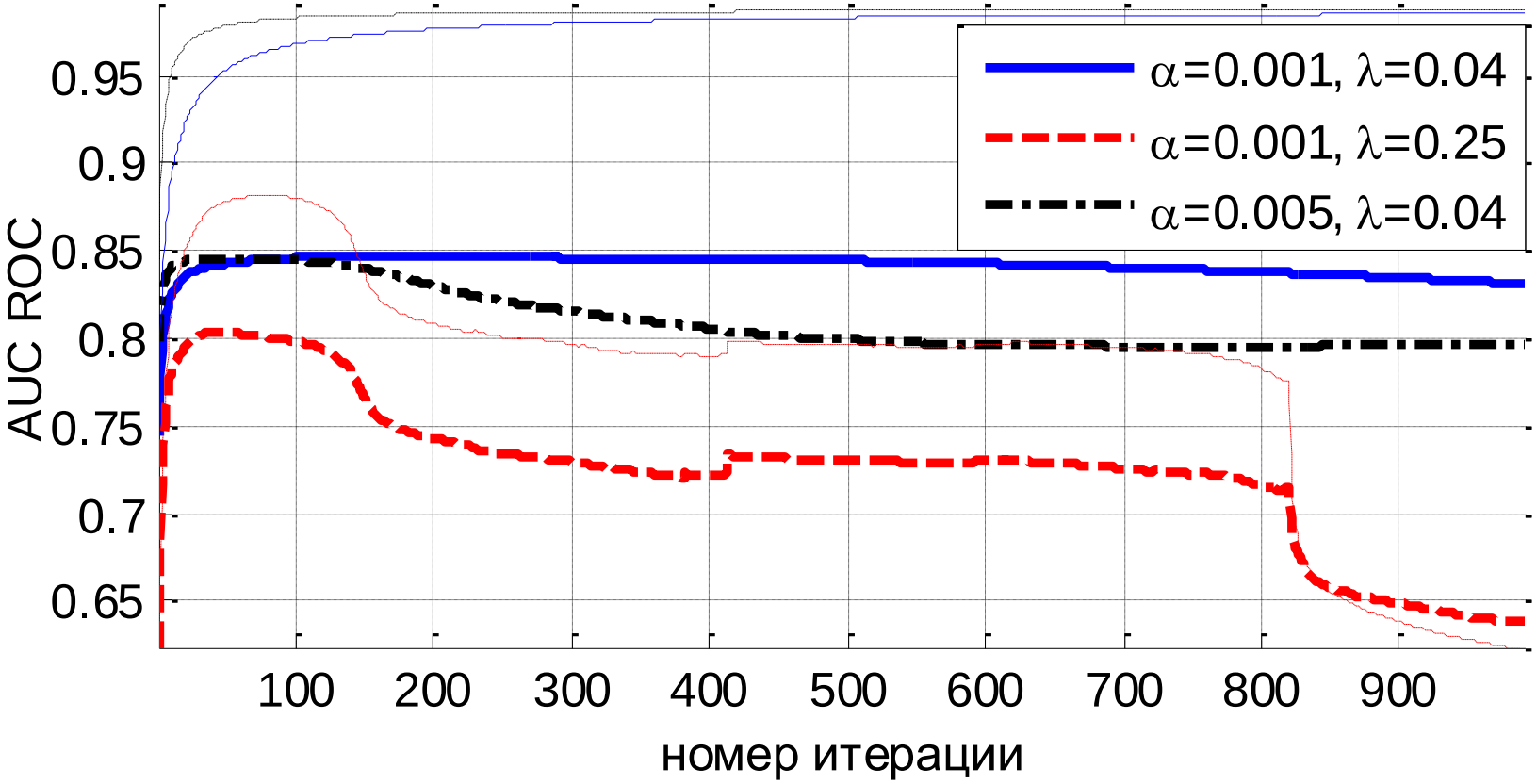
$$\frac{\partial J}{\partial u_{is}} = 2\sum_{t:\ f_{t1}=i} e_t v_{s,f_{t2}} + 2\lambda_1 u_{is} \qquad \frac{\partial J}{\partial v_{sj}} = 2\sum_{t:\ f_{t2}=j} e_t u_{f_{t1},s} + 2\lambda_2 v_{sj}$$

Итерационный процесс:

$$u_{is} = u_{is} - \alpha\sum_{t:\ f_{t1}=i} e_t v_{s,f_{t2}} - \lambda_1 u_{is} \qquad v_{sj} = v_{sj} - \alpha\sum_{t:\ f_{t2}=j} e_t u_{f_{t1},s} - \lambda_2 v_{sj}$$

2. ALS

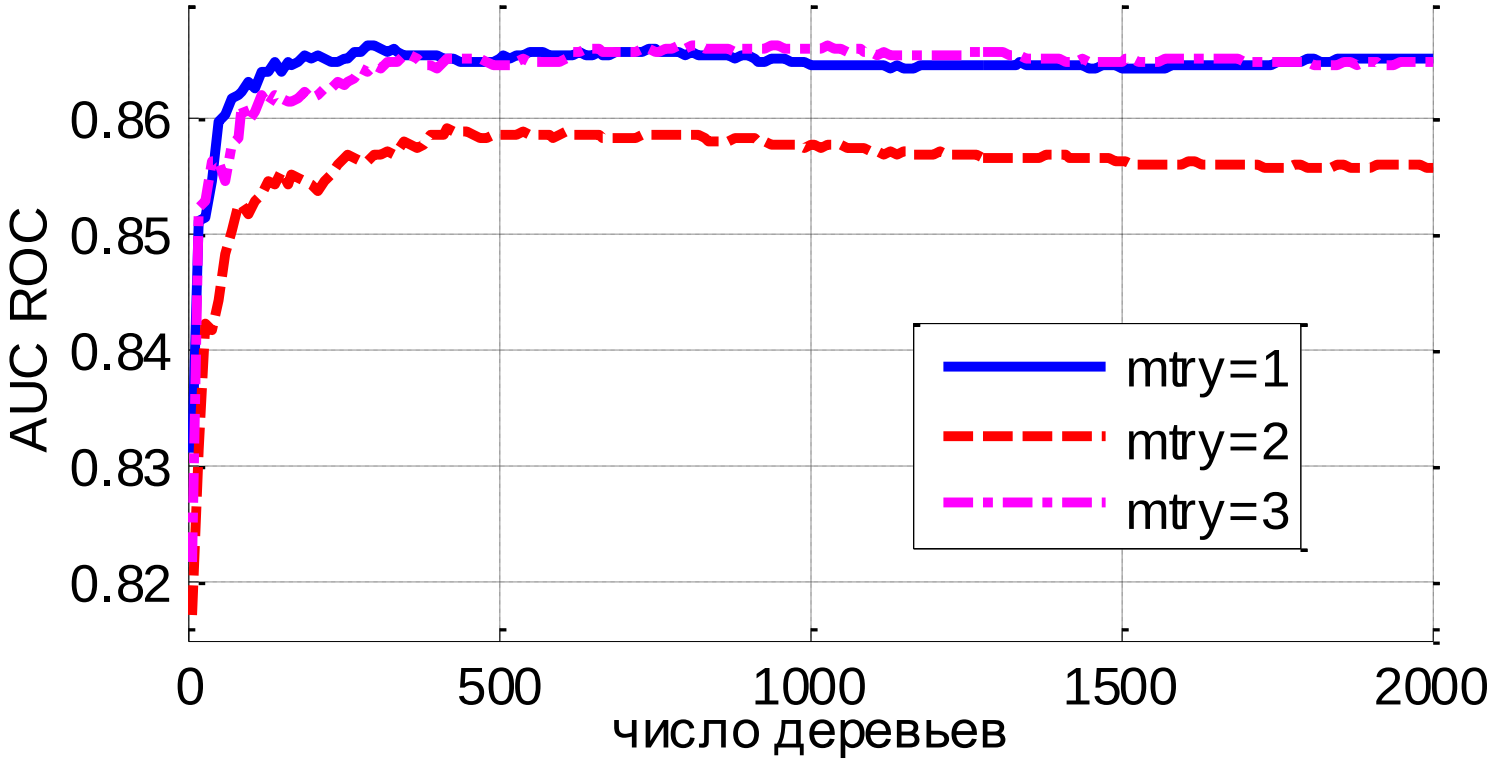
Методы, основанные на тензорных разложениях



Качество на обучении (тонкие графики) и контроле (толстые)

Методы, основанные на кодировках признаков

Качество случайного леса от числа деревьев при случайных кодировках



Кодирование относительно других (факторных) признаков

A	1
B	1
A	2
A	2
A	3
A	3
B	3
C	3
B	4
C	4
B	4
B	4

	1	2	3	4
A	1	2	2	0
B	1	0	0	3
C	0	0	1	1

В этой матрице всё содержится…

SVD-разложение характеристической (по Матросову) матрицы

```
[u,l,v] = svd(A)

u =

    0.5043    0.8419   -0.1922
    0.7932   -0.5396   -0.2822
    0.3413    0.0102    0.9399


l =

    3.4535         0         0         0
         0    2.8954         0         0
         0         0    0.8307         0


v =

    0.3757    0.1044   -0.5711   -0.7223
    0.2920    0.5815   -0.4628    0.6019
    0.3909    0.5850    0.6686   -0.2408
    0.7879   -0.5556    0.1122    0.2408
```

A	0.5043	0.8419	1	0.3757	0.1044
B	0.7932	-0.5396	1	0.3757	0.1044
A	0.5043	0.8419	2	0.2920	0.5815
A	0.5043	0.8419	2	0.2920	0.5815
A	0.5043	0.8419	3	0.3909	0.5850
A	0.5043	0.8419	3	0.3909	0.5850
B	0.7932	-0.5396	3	0.3909	0.5850
C	0.3413	0.0102	3	0.3909	0.5850
B	0.7932	-0.5396	4	0.7879	-0.5556
C	0.3413	0.0102	4	0.7879	-0.5556
B	0.7932	-0.5396	4	0.7879	-0.5556
B	0.7932	-0.5396	4	0.7879	-0.5556

кодировка при использовании двух компонент в разложениях

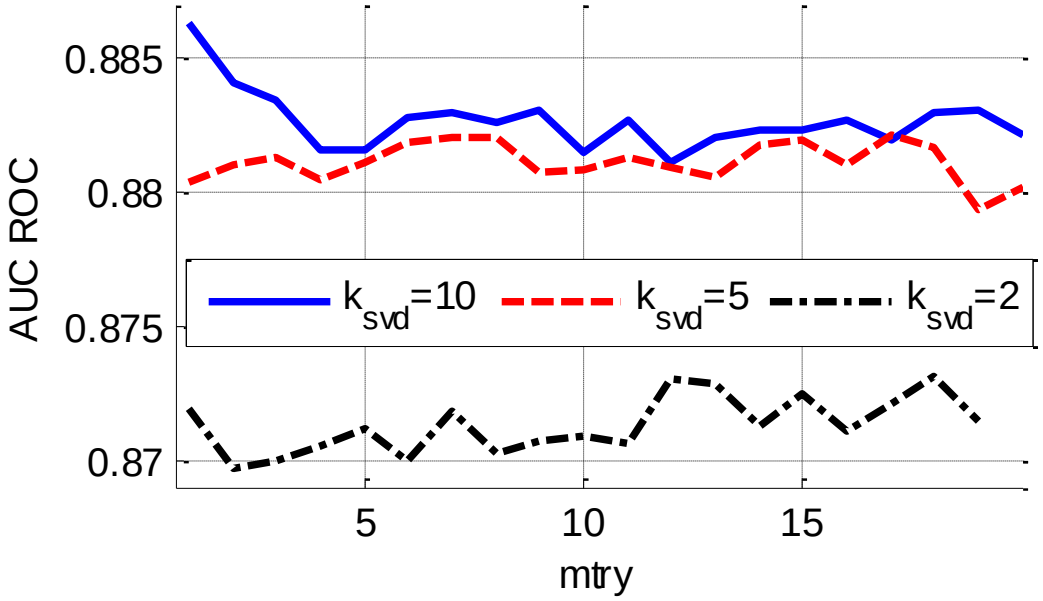
Код: кодирование по ДРУГОМУ категориального признака

- **по характеристической матрице**

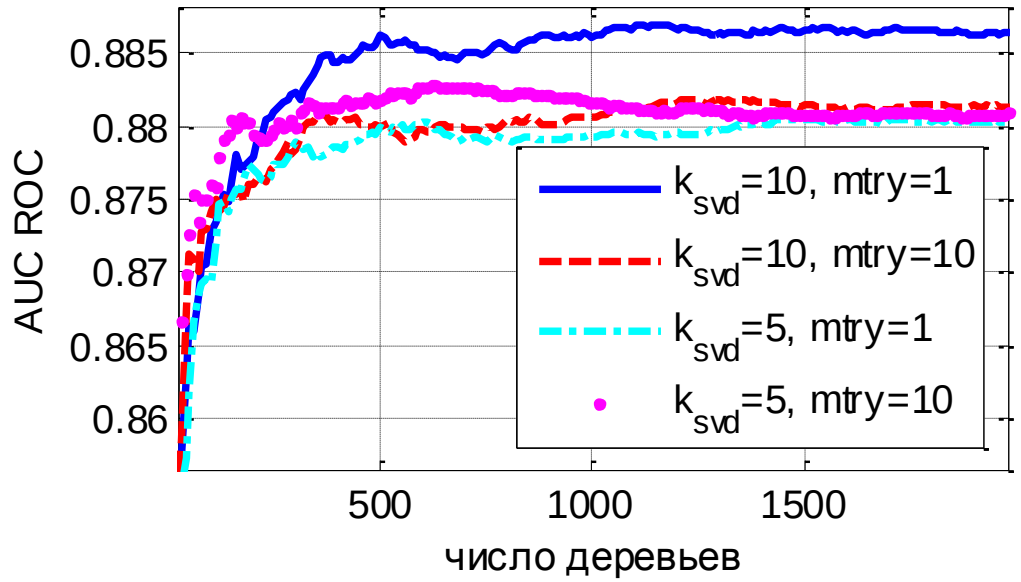
```
import numpy as np
from numpy.linalg import svd
def code_factor(data, cat_feature, cat_feature2):
    ct = pd.crosstab(data[cat_feature], data[cat_feature2])
    u, _, _ = svd(ct.values)
    coder = dict(zip(ct.index, u[:,0])) # если кодировать первой компонентой
    return (data[cat_feature].map(coder))

data['city_degree_code'] = code_factor(data, 'city', 'degree')
```

	city	class	degree	income	city_mean_income	city_degree_code
0	Moscow	A	1	10.2	8.5	-0.888074
1	London	B	1	11.6	10.2	-0.325058
2	London	A	2	8.8	10.2	-0.325058
3	Kiev	A	2	9.0	9.0	-0.325058
4	Moscow	B	3	6.6	8.5	-0.888074
5	Moscow	B	3	10.0	8.5	-0.888074
6	Kiev	A	1	9.0	9.0	-0.325058
7	Moscow	A	1	7.2	8.5	-0.888074

Методы, основанные на кодировках признаков



Зависимость качества от параметра mtry случайного леса



Зависимость качества от числа деревьев в случайном лесе

Кодировки

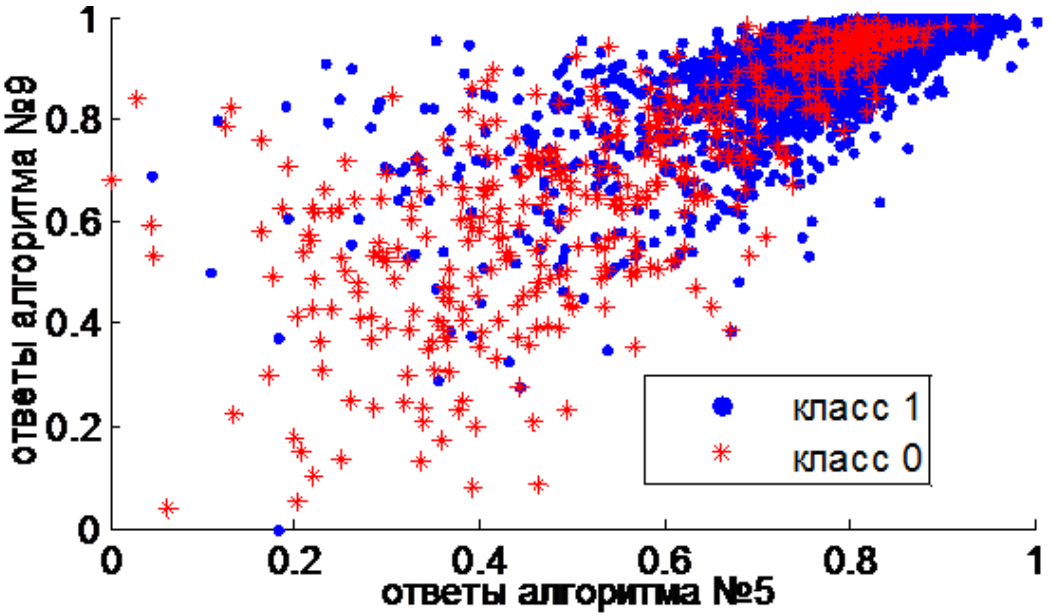
- случайные
- на основе других
- SVD (на основе других + ортогонализация)
- байесовская (нет экспериментов)

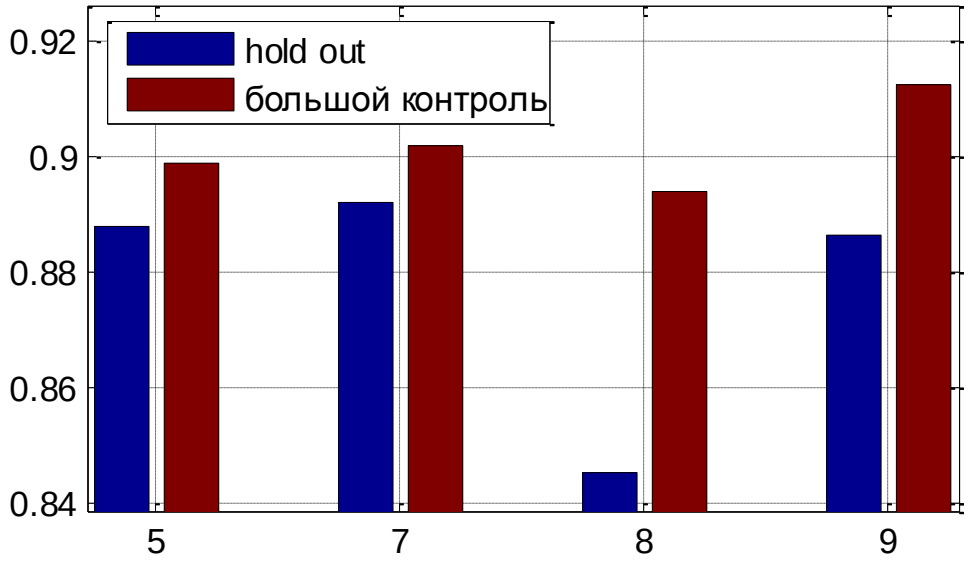
Ансамбли алгоритмов

$$\alpha A_1 + (1-\alpha) A_2, \alpha \in [0,1]$$

алгоритмы	Линейный 4	Байес 5	kNN+ABO 7	Тензоры 8	Кодировки 9
линейный	0.8713	0.8914	0.8919	0.8731	0.8879
Байес		0.8878	0.8972	0.8884	0.8974
kNN+ABO			0.8919	0.8924	0.8919
тензоры				0.8453	0.8872
кодировки					0.8863

Ансамбли алгоритмов





AUC хороший, а разделимости нет...

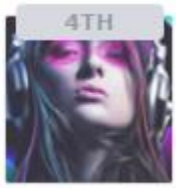
и корреляции нет...

Факторизационные машины

1st/239

4th/657

4th/163



4th/133

Steffen Rendle

libFM: Factorization Machine Library

http://www.libfm.org/

Супермодель, иммитирует
SVD, SVD++, FPMC, Pairwise interaction tensor factorization,
SVM с полином. ядром и т.п.

Ask Peter Norvig

Q5: What, say, 3 recent papers in machine learning do you think will be influential to directing the cutting edge of research these days? (41 Up-votes, 26.08.2014)

I've never been able to pick lasting papers in the past, so don't trust me now, but here are a few:

Rendle's "Factorization Machines"

Wang et al. "Bayesian optimization in high dimensions via random embeddings"

Dean et al. "Fast, Accurate Detection of 100,000 Object Classes on a Single Machine"

Факторизационные машины

$$r_{ui} \sim w_0 + w_u + w_i + v_u^{\mathrm{T}} v_i$$

модель второго порядка:

$$w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{1 \le i < j \le n} v_i^{\mathrm{T}} v_j x_i x_j \sim w_0 + w^{\mathrm{T}} x + x^{\mathrm{T}} \underbrace{W}_{\sim \mathrm{rg}=k} x$$

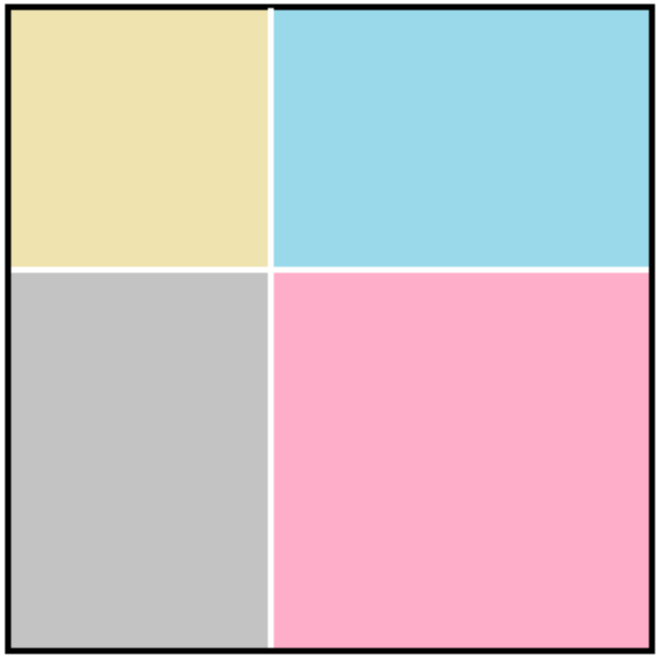
«факторизация» – в предположении, какая у нас матрица весов, иначе была бы просто «модель второго порядка»

Факторизационные машины

Что ещё…

- **факторизация отдельных блоков (FFM – field-aware factorization machine)**
- **эффективное блочное хранение**

FFM – field-aware factorization machine



Линейная модель

$$\phi(\mathbf{w},\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{j\in C_1} w_j x_j$$

Полиномиальная модель (Poly2)

$$\phi(\mathbf{w},\mathbf{x}) = \sum_{j_1,j_2\in C_2} w_{j_1,j_2} x_{j_1} x_{j_2}$$

Факторизационная машина

$$\phi(\mathbf{w},\mathbf{x}) = \sum_{j_1,j_2\in C_2} \langle \mathbf{w}_{j_1}, \mathbf{w}_{j_2}\rangle x_{j_1} x_{j_2}$$

Факторизационная машина с полями

$$\phi(\mathbf{w},\mathbf{x}) = \sum_{j_1,j_2\in C_2} \langle \mathbf{w}_{j_1,f_2}, \mathbf{w}_{j_2,f_1}\rangle x_{j_1} x_{j_2}$$

Оптимизационная задача

$$\min_{\mathbf{w}} \sum_{i=1}^{L} \left(\log\left(1 + \exp(-y_i \phi(\mathbf{w}, \mathbf{x}_i)\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \right),$$

$$\phi(\mathbf{w}, \mathbf{x}) = \sum_{j_1, j_2 \in C_2} \langle \mathbf{w}_{j_1, f_2}, \mathbf{w}_{j_2, f_1} \rangle x_{j_1} x_{j_2},$$

LogLoss + регуляризация

Что такое поля...

Field name		Field index
User	$\rightarrow$	field 1
Movie	$\rightarrow$	field 2
Genre	$\rightarrow$	field 3
Price	$\rightarrow$	field 4

Что ещё?

- неотрицательные матричные разложения
- вероятностные разложения
- специальные регуляризаторы
- локальная низкоранговость
- бикластеризация
- тензоры (тензорное разложение)

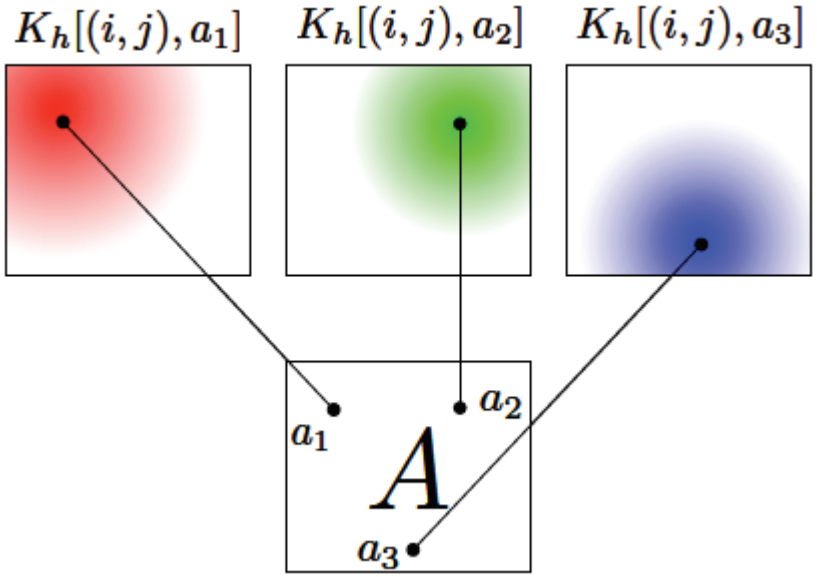


рис. из дипломной работы М.Трофимова

Литература

Дьяконов А. Методы решения задач классификации с категориальными признаками // Прикладная математика и информатика. Труды факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова. — 2014. — № 46. — С. 103–127
http://istina.msu.ru/media/publications/article/972/9eb/7537819/sw-factors-dyakonov.pdf

Y. Koren, R.M. Bell, C. Volinsky Matrix Factorization Techniques for Recommender Systems // IEEE Computer 42(8): 30-37 (2009).

Соревнование «Amazon.com – Employee Access Challenge» //
http://www.kaggle.com/c/amazon-employee-access-challenge

S. Funk Netflix Update: Try This at Home //
http://sifter.org/~simon/journal/20061211.html

libFM: Factorization Machine Library // http://www.libfm.org/

Литература

Python: Кодирование категориальных признаков //
https://github.com/Dyakonov/python_hacks/blob/master/dj_cat_coding.ipynb

FFM – field-aware factorization machine (слайды) //
http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf