

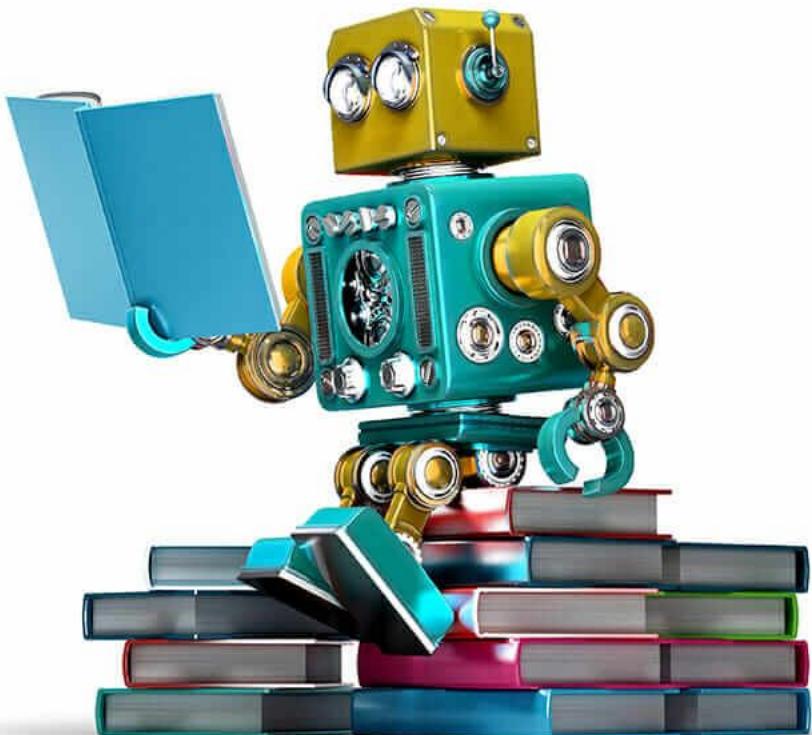
# Lecture 4

---

Bulat Ibragimov  
*Yandex Research*

# План

1. Кластеризация
2. Предобработка и визуализация данных
3. Работа с признаками



# Clustering

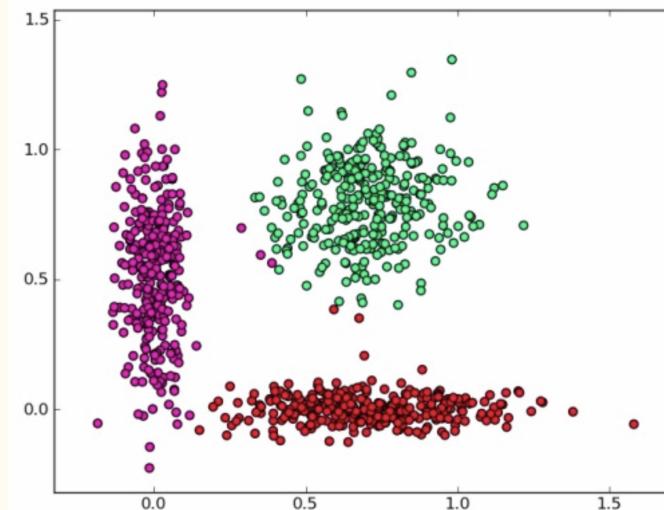
---

100  
010

# Кластеризация

**Кластерный анализ** (Data clustering) — задача разбиения заданной [выборки](#) объектов (ситуаций) на непересекающиеся подмножества, называемые [кластерами](#), так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

Задача кластеризации относится к широкому классу задач [обучения без учителя](#).



# Кластеризация

## Формальная постановка задачи кластеризации

Пусть  $\mathbf{X}$  — множество объектов,  $\mathbf{Y}$  — множество номеров (имён, меток) кластеров. Задана функция расстояния между объектами  $\rho(x, x')$ . Имеется конечная обучающая выборка объектов  $\mathbf{X}^m = \{x_1, \dots, x_m\} \subset \mathbf{X}$ . Требуется разбить выборку на непересекающиеся подмножества, называемые *кластерами*, так, чтобы каждый кластер состоял из объектов, близких по метрике  $\rho$ , а объекты разных кластеров существенно отличались. При этом каждому объекту  $x_i \in \mathbf{X}^m$  приписывается номер кластера  $y_i$ .

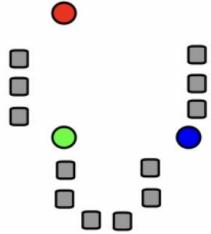
Алгоритм кластеризации — это функция  $a: \mathbf{X} \rightarrow \mathbf{Y}$ , которая любому объекту  $x \in \mathbf{X}$  ставит в соответствие номер кластера  $y \in \mathbf{Y}$ . Множество  $\mathbf{Y}$  в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Кластеризация ([обучение без учителя](#)) отличается от [классификации \(обучения с учителем\)](#) тем, что метки исходных объектов  $y_i$  изначально не заданы, и даже может быть неизвестно само множество  $\mathbf{Y}$ .

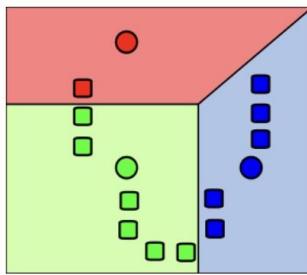
Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- Не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты.
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.
- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

# k-means

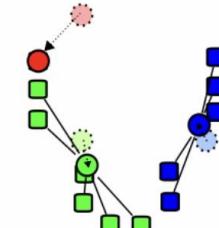


Исходные точки и  
случайно выбранные  
начальные точки.

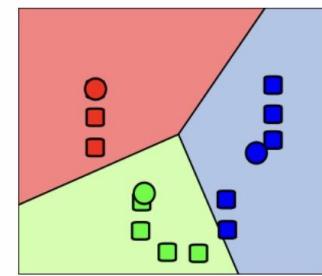


Точки, отнесённые к  
начальным центрам.

Разбиение на  
плоскости —  
**диаграмма Вороного**  
относительно  
начальных центров.

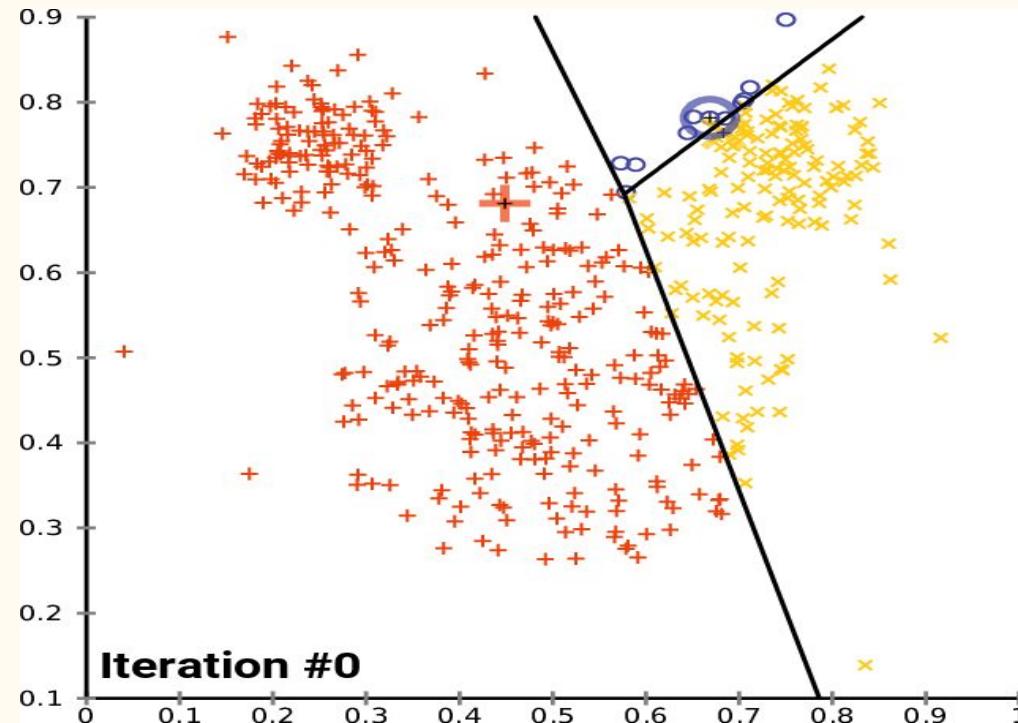


Вычисление новых  
центров кластеров  
(Ищется **центр масс**).



Предыдущие шаги  
повторяются, пока  
алгоритм не сойдётся.

# k-means



# Другие алгоритмы

- k-means
- EM algorithm
- DBSCAN
- Agglomerative clustering
- Divisive clustering

# Preprocessing

## And

## Visualization

---

100  
010

## Standardization

Mean removal and variance scaling

```
>>> from sklearn import preprocessing  
>>> import numpy as np  
>>> X_train = np.array([[ 1., -1.,  2.],  
...                      [ 2.,  0.,  0.],  
...                      [ 0.,  1., -1.]])  
>>> X_scaled = preprocessing.scale(X_train)  
  
>>> X_scaled  
array([[ 0. ..., -1.22...,  1.33...],  
       [ 1.22...,  0. ..., -0.26...],  
       [-1.22...,  1.22..., -1.06...]])
```

```
>>> X_scaled.mean(axis=0)  
array([0., 0., 0.])  
  
>>> X_scaled.std(axis=0)  
array([1., 1., 1.])
```

- RBF Kernel for SVM
- L1, L2 regularizations for LM

## Scaling features to a range

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                      [ 2.,  0.,  0.],
...                      [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[0.5        , 0.         , 1.         ],
       [1.         , 0.5        , 0.33333333],
       [0.         , 1.         , 0.         ]])
```

- robustness to very small standard deviations of features
- preserving zero entries in sparse data

## Mapping to a uniform distribution

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
>>> quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
>>> X_train_trans = quantile_transformer.fit_transform(X_train)
>>> X_test_trans = quantile_transformer.transform(X_test)
>>> np.percentile(X_train[:, 0], [0, 25, 50, 75, 100])
array([ 4.3,  5.1,  5.8,  6.5,  7.9])
```

- it smooths out unusual distributions and is less influenced by outliers than scaling methods.
- It does, however, distort correlations and distances within and across features

## Normalization

```
>>> X = [[ 1., -1.,  2.],  
...        [ 2.,  0.,  0.],  
...        [ 0.,  1., -1.]]  
>>> X_normalized = preprocessing.normalize(X, norm='l2')  
  
>>> X_normalized  
array([[ 0.40..., -0.40...,  0.81...],  
       [ 1. ...,  0. ...,  0. ...],  
       [ 0. ...,  0.70..., -0.70...]])
```

- metric algorithms
- quadratic kernels

# Encoding categorical features

```
>>> enc = preprocessing.OrdinalEncoder()
>>> X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses Firefox']]
>>> enc.fit(X)
OrdinalEncoder(categories='auto', dtype=<... 'numpy.float64'>)
>>> enc.transform([['female', 'from US', 'uses Safari']])
array([[0., 1., 1.]])
```

## Binarization

```
>>> X = [[ 1., -1.,  2.],
...        [ 2.,  0.,  0.],
...        [ 0.,  1., -1.]]
>>> binarizer = preprocessing.Binarizer().fit(X) # fit does nothing
>>> binarizer
Binarizer(copy=True, threshold=0.0)
>>> binarizer.transform(X)
array([[1., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

## Discretization

```
>>> X = np.array([[ -3.,  5., 15 ],
...                 [  0.,  6., 14 ],
...                 [  6.,  3., 11 ]])
>>> est = preprocessing.KBinsDiscretizer(n_bins=[3, 2, 2], encode='ordinal').fit(X)
```

```
>>> est.transform(X)
array([[ 0.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 2.,  0.,  0.]])
```

discretizes features into k equal width bins

## Generating polynomial features

```
>>> import numpy as np
>>> from sklearn.preprocessing import PolynomialFeatures
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```

- Generating new features
- Train polynomial algorithms

# Как выглядит обучающая выборка

Fisher's Iris Data

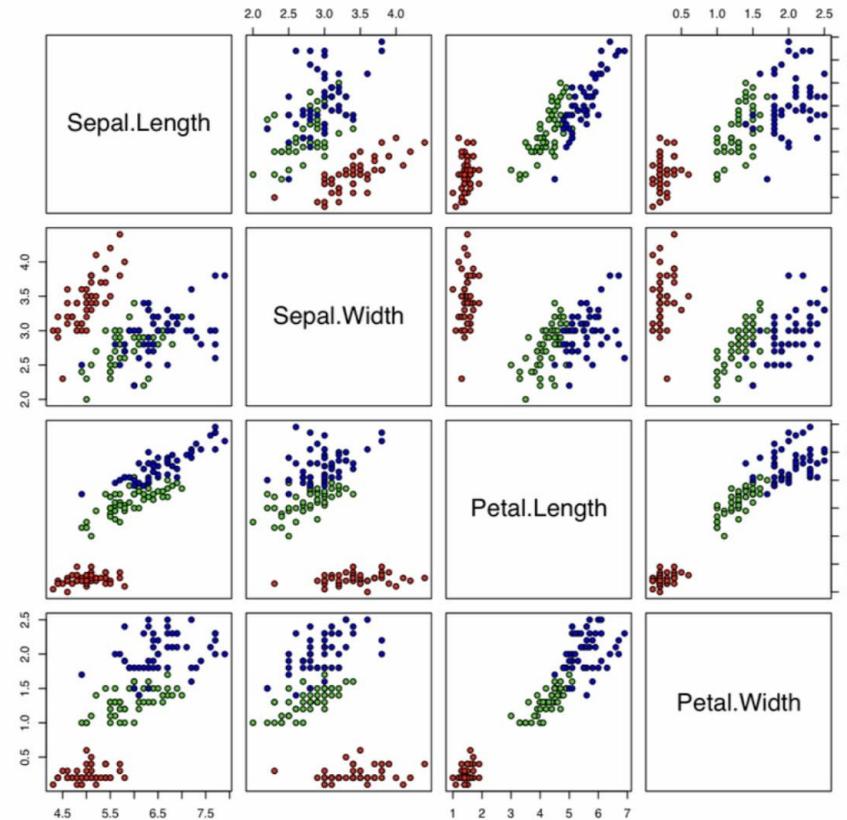
Sepal length ↴	Sepal width ↴	Petal length ↴	Petal width ↴	Species ↴
5.0	2.0	3.5	1.0	<i>I. versicolor</i>
6.0	2.2	5.0	1.5	<i>I. virginica</i>
6.2	2.2	4.5	1.5	<i>I. versicolor</i>
6.0	2.2	4.0	1.0	<i>I. versicolor</i>
6.3	2.3	4.4	1.3	<i>I. versicolor</i>
5.5	2.3	4.0	1.3	<i>I. versicolor</i>
5.0	2.3	3.3	1.0	<i>I. versicolor</i>
4.5	2.3	1.3	0.3	<i>I. setosa</i>
5.5	2.4	3.8	1.1	<i>I. versicolor</i>
5.5	2.4	3.7	1.0	<i>I. versicolor</i>
4.9	2.4	3.3	1.0	<i>I. versicolor</i>
6.7	2.5	5.8	1.8	<i>I. virginica</i>
5.7	2.5	5.0	2.0	<i>I. virginica</i>
6.3	2.5	5.0	1.9	<i>I. virginica</i>
6.3	2.5	4.9	1.5	<i>I. versicolor</i>
4.9	2.5	4.5	1.7	<i>I. virginica</i>

# Что хотелось бы уметь

- Визуализировать обучающую выборку, когда признаков больше трёх
- Уменьшать количество признаков, переходя к новым, более информативным

# Визуализируем выборку

Iris Data (red=setosa,green=versicolor,blue=virginica)

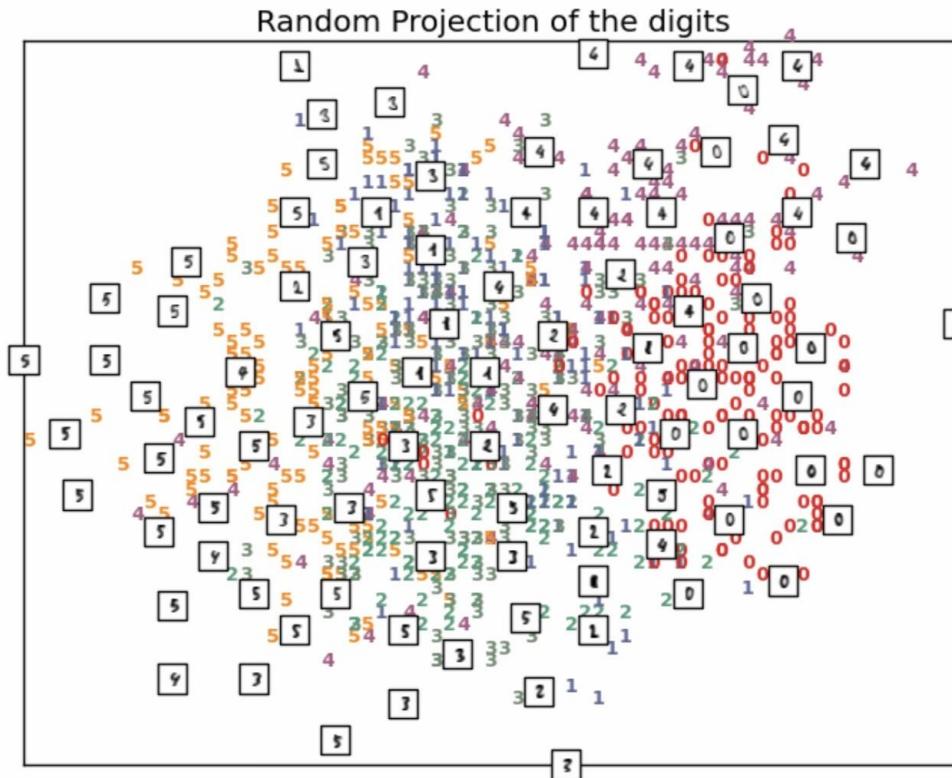


# Более сложный случай

Что делать, если признаков еще больше?

1	6	4	12	5	5	3	4	1	67	3	2	1	2	1	0	0	1	0	0	0	1	0	0	1	1	
2	48	2	60	1	3	2	2	1	22	3	1	1	1	1	0	0	1	0	0	0	1	0	0	1	2	
4	12	4	21	1	4	3	3	1	49	3	1	2	1	1	0	0	1	0	0	0	1	0	0	1	1	
1	42	2	79	1	4	3	4	2	45	3	1	2	1	1	0	0	0	0	0	0	0	0	0	1	1	
1	24	3	49	1	3	3	4	4	53	3	2	2	1	1	1	0	1	0	0	0	0	0	0	1	2	
4	36	2	91	5	3	3	4	4	35	3	1	2	2	1	0	0	1	0	0	0	0	0	1	0	1	
4	24	2	28	3	5	3	4	2	53	3	1	1	1	1	0	0	0	1	0	0	1	0	0	1	1	
2	36	2	69	1	3	3	2	3	35	3	1	1	2	1	0	1	1	0	1	0	0	0	0	0	1	
4	12	2	31	4	4	1	4	1	61	3	1	1	1	1	0	0	1	0	0	0	1	0	0	1	0	
2	30	4	52	1	1	4	2	3	28	3	2	1	1	1	1	0	1	0	0	0	1	0	0	0	2	
2	12	2	13	1	2	2	1	3	25	3	1	1	1	1	1	0	1	0	0	1	0	0	0	1	2	
1	48	2	43	1	2	2	4	2	24	3	1	1	1	1	1	0	0	1	0	0	1	0	0	0	1	2
2	12	2	16	1	3	2	1	3	22	3	1	1	2	1	0	0	1	0	0	0	1	0	0	1	1	
1	24	4	12	1	5	3	4	3	60	3	2	1	1	1	1	0	1	0	0	0	1	0	0	1	0	
1	15	2	14	1	3	2	4	3	28	3	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	
1	24	2	13	2	3	2	2	3	32	3	1	1	1	1	1	0	0	1	0	0	0	1	0	0	2	
4	24	4	24	5	5	3	4	2	53	3	2	1	1	1	0	0	1	0	0	0	1	0	0	1	1	
1	30	0	81	5	2	3	3	3	25	1	3	1	1	1	0	0	1	0	0	0	1	0	0	1	1	
2	24	2	126	1	5	2	2	4	44	3	1	1	2	1	0	1	1	0	0	0	0	0	0	0	2	
4	24	2	34	3	5	3	2	3	31	3	1	2	2	1	0	0	1	0	0	0	1	0	0	0	1	1
4	9	4	21	1	3	3	4	3	48	3	3	1	2	1	1	0	1	0	0	1	0	0	0	1	1	
1	6	2	26	3	3	3	3	1	44	3	1	2	1	1	0	0	1	0	0	1	0	0	0	1	1	
1	10	4	22	1	2	3	3	1	48	3	2	2	1	2	1	0	1	0	0	1	0	0	0	1	0	
2	12	4	18	2	2	3	4	2	44	3	1	1	1	1	0	1	1	0	0	0	1	0	0	0	1	1
4	10	4	21	5	3	4	1	3	26	3	2	1	1	2	0	0	1	0	0	0	1	0	0	0	1	1
1	6	2	14	1	3	3	2	1	36	1	1	1	2	1	0	0	1	0	0	0	1	0	0	1	0	
4	6	0	4	1	5	4	4	3	39	3	1	1	1	1	0	0	1	0	0	0	1	0	0	1	0	
3	12	1	4	4	3	2	3	1	42	3	2	1	1	1	0	0	1	0	0	1	0	0	0	1	1	
2	7	2	24	1	3	3	2	1	34	3	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	1
1	60	3	68	1	5	3	4	4	63	3	2	1	2	1	0	0	1	0	0	0	1	0	0	0	1	2
2	18	2	19	4	2	4	3	1	36	1	1	1	2	1	0	0	1	0	0	0	1	0	0	0	1	1
1	24	2	40	1	3	3	2	3	27	2	1	1	1	1	0	0	1	0	0	0	1	0	0	0	1	1

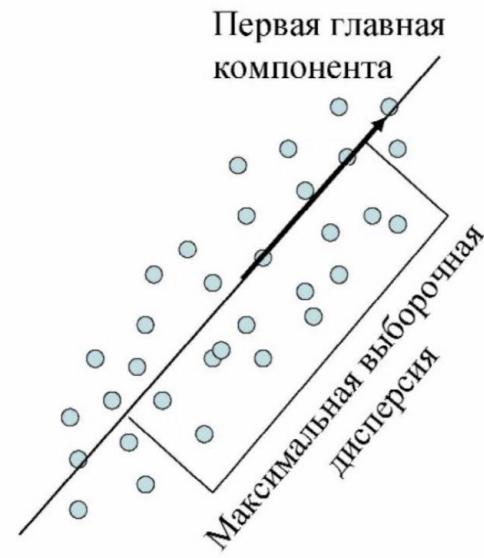
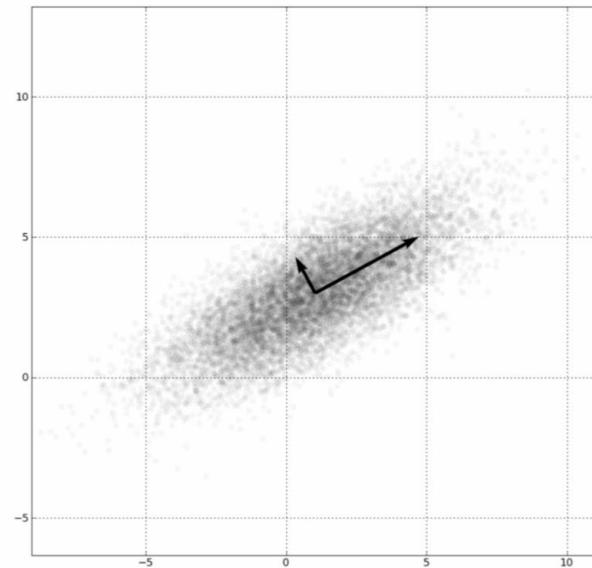
# Пример случайной проекции для рукописных цифр



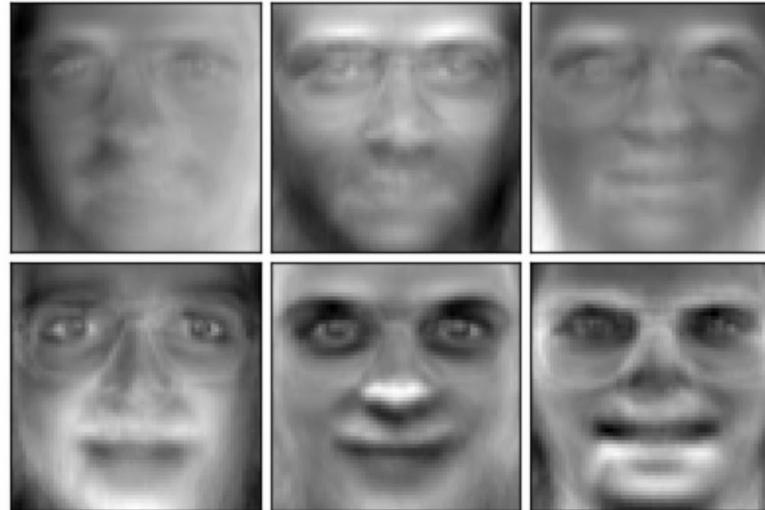
# Principal Component Analysis

- Идея: давайте выделять в пространстве признаков направления, вдоль которых разброс точек наибольший (они кажутся наиболее информативными)

# PCA



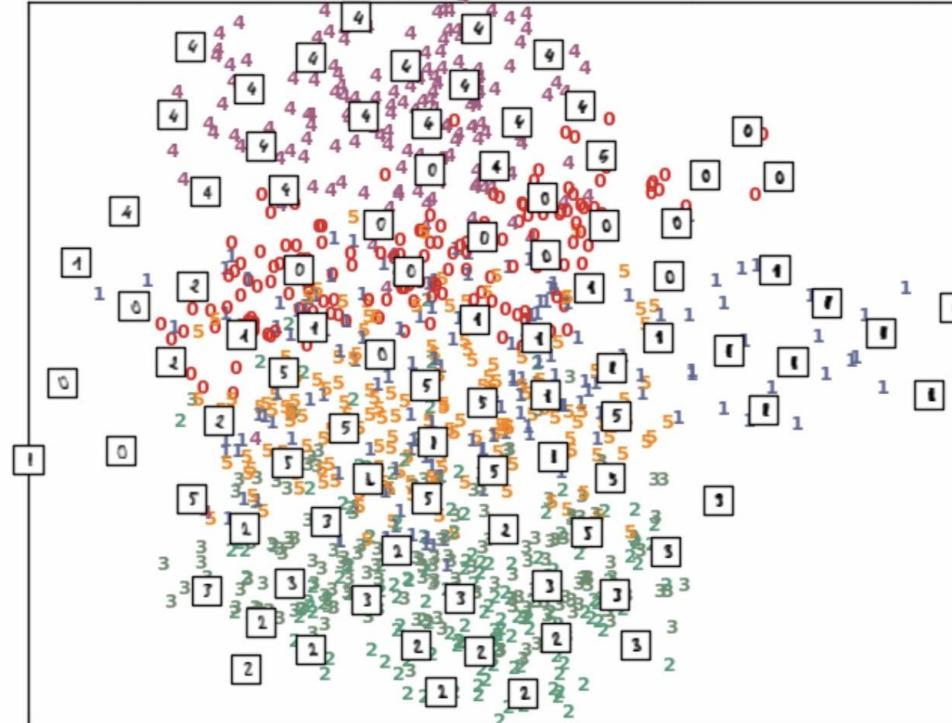
# Пример: eigenfaces



$$= \text{mean} + 0.9 * \text{eigenface}_1 - 0.2 * \text{eigenface}_2 + 0.4 * \text{eigenface}_3 + \dots$$

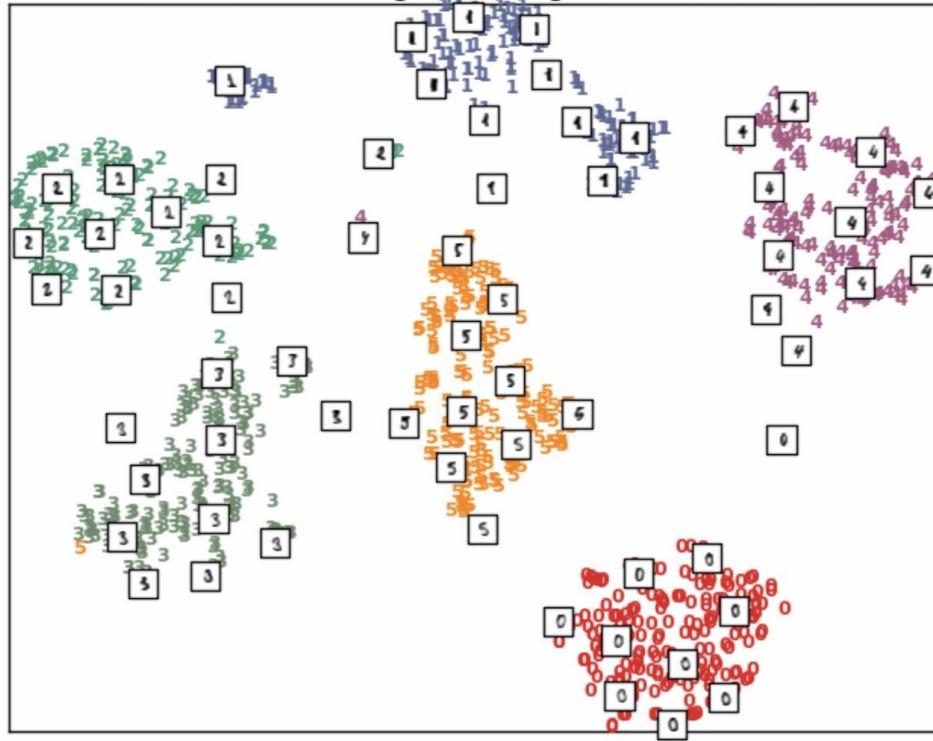
# Рукописные цифры: проекция на главные компоненты

Principal Components projection of the digits (time 0.02s)



# Manifold learning: t-SNE

t-SNE embedding of the digits (time 23.50s)



# Features

---

100  
010

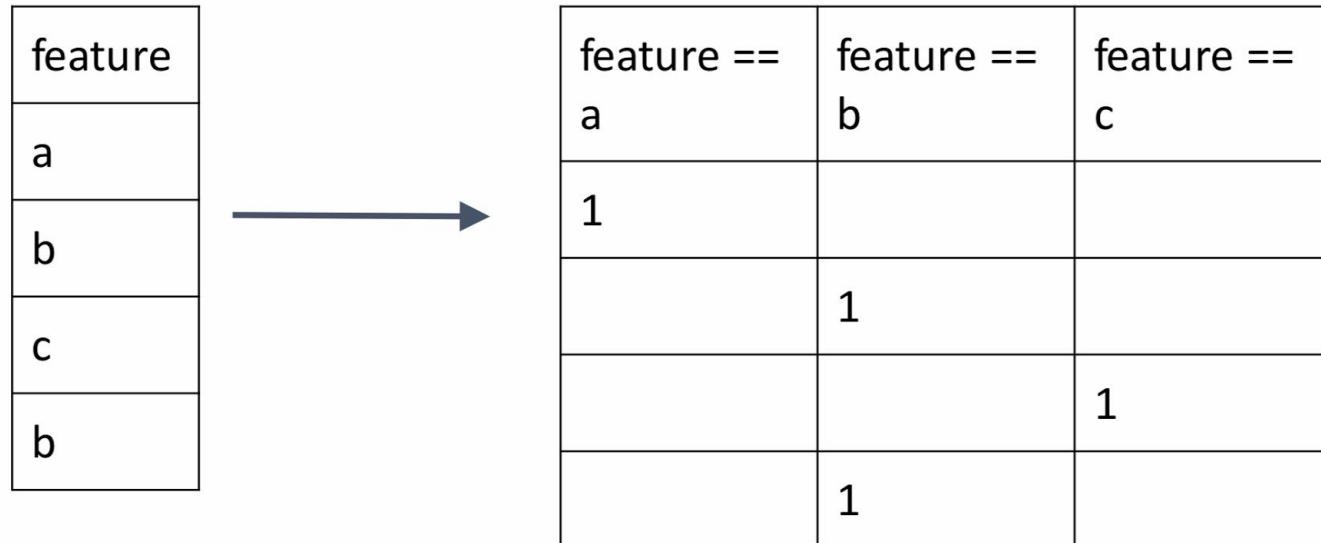
# Категориальные признаки (строки)

Из колонок “name”, “ticket”, “cabin” можно сгенерировать новые признаки

	A	B	C	D	E	F	G	H	I	J	K
1	survived	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
2	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
3	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
4	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2.	7.925		S
5	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
6	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
7	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
8	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
9	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
10	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
11	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
12	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
13	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S

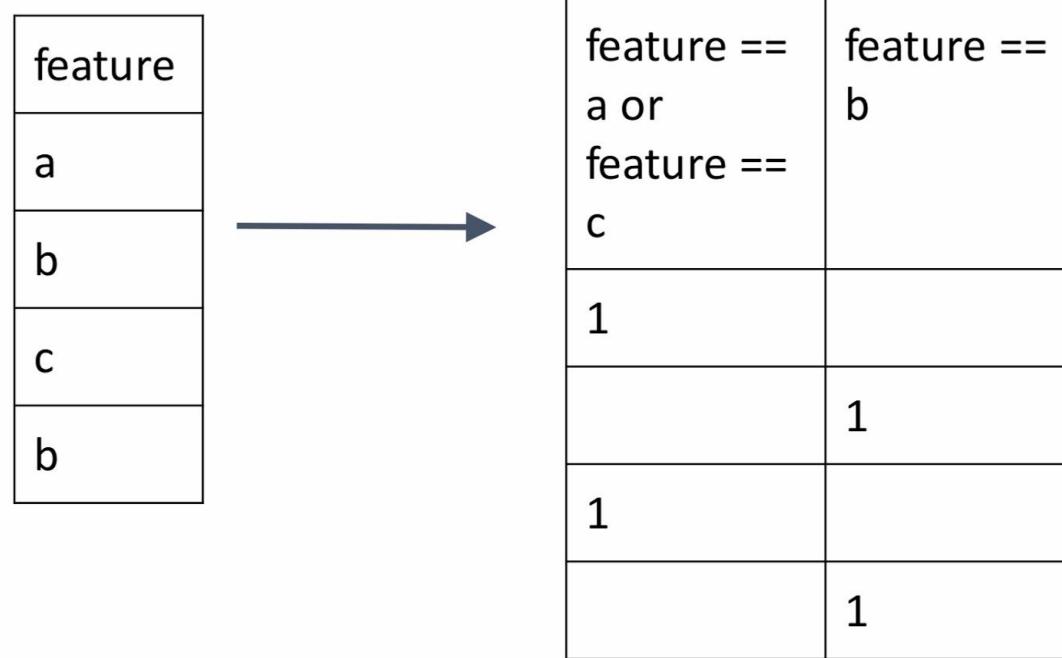
# Категориальные признаки

## Бинаризация



# Категориальные признаки

Hashing trick



# Категориальные признаки

№ склада	Город	...	Продано вина (ящиков)
2343	Москва	...	56000
185	Самара	...	10500
121	Ростов	...	11300
...	...	...	...

# Категориальные признаки

№ склада	В среднем продают в городе	...	Продано вина (ящиков)
2343	59000	...	56000
185	11200	...	10500
121	12100	...	11300
...	...	...	...

# Методы обработки пропущенных значений

- Удаление объектов/признаков с пропущенными значениями
- Замена специальным значением
- Замена агрегированной статистикой (по любой характеристике)
- Матричная факторизация ( $X \sim AB$ )
- Метод ближайшего соседа
- Восстановление методами машинного обучения