Методы формирования пространства признаков

[Искусственный интеллект и экспертные системы]

Капырин Николай, старший преподаватель каф. 305

Москва, 2018

Московский Авиационный Институт

Пространство признаков

Пространство признаков

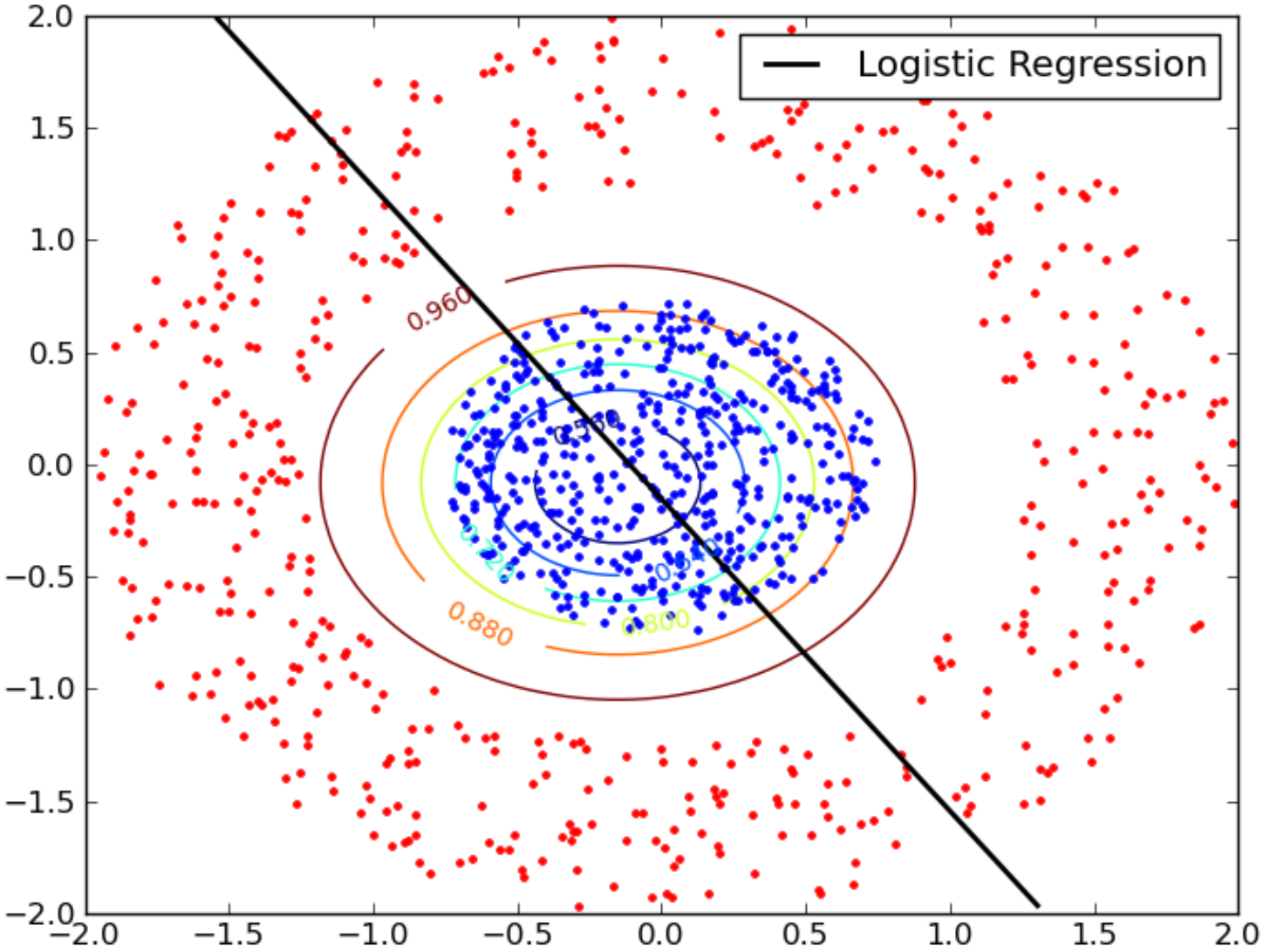
Мы видели что обучение (эмпирическое обобщение) опирается на разные структуры данных.

Структура системы машинного обучения определяет качество ответа на задачу анализа, но количественный ответ зависит от параметров модели, и чем их больше, тем...

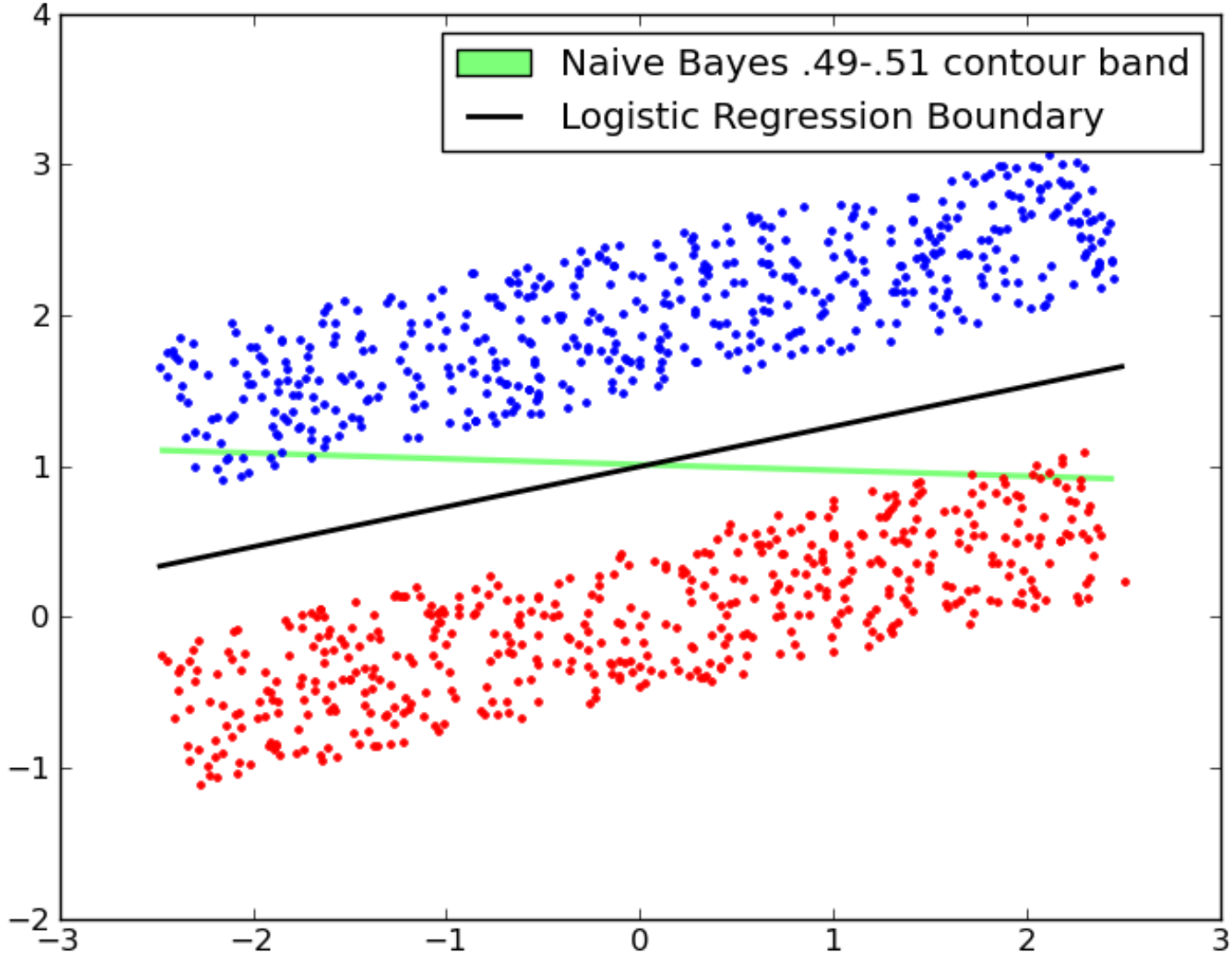
Мотивация снижения размерности задачи:

- Визуализация – представить в голове 5-мерные данные довольно трудно
- Много признаков – дольше обучать
- Качество обучения – чтобы параметры были стабильными, данные должны быть из большой несмещённой выборки
- Небольшие модели экономнее при эксплуатации

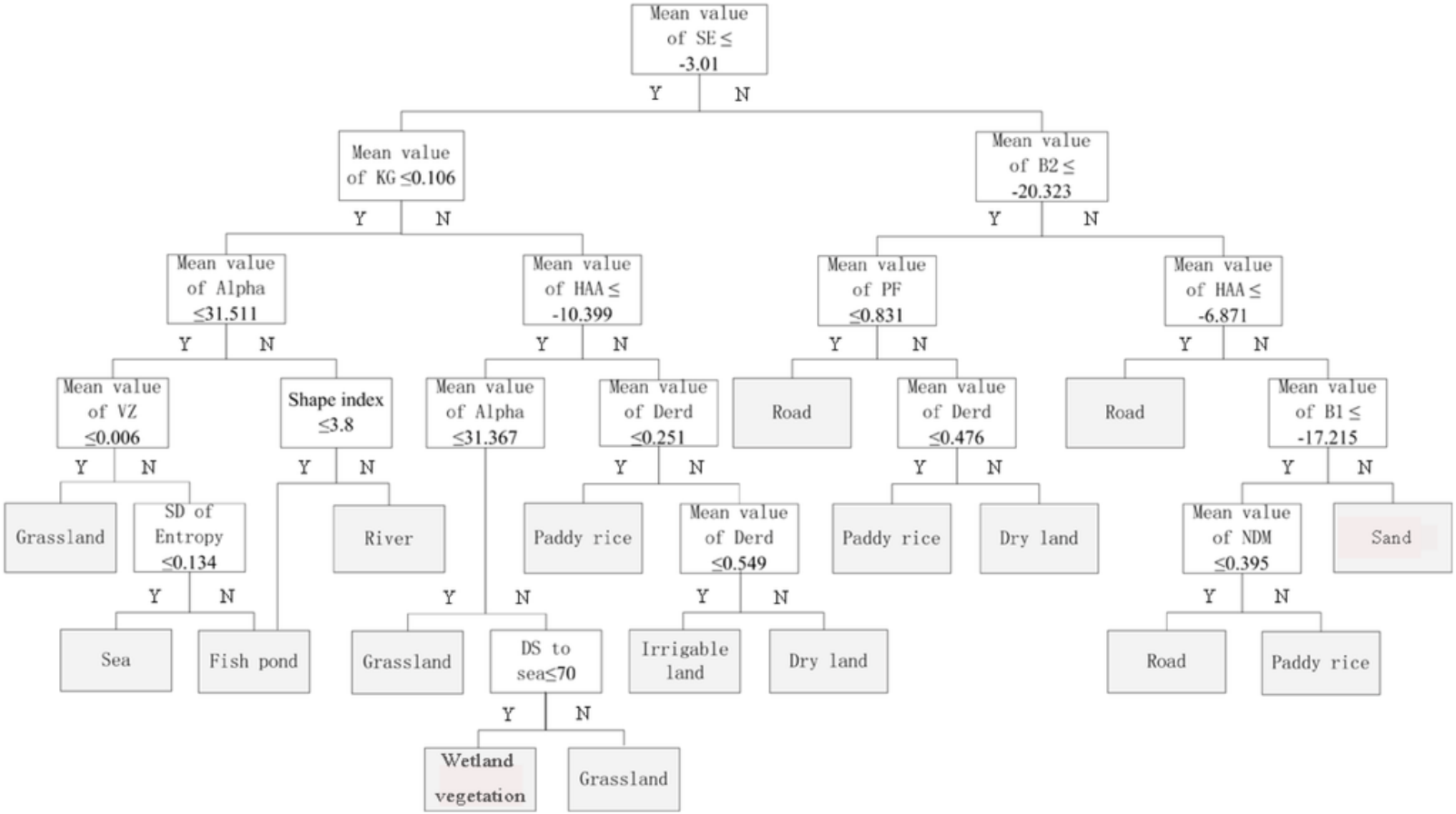
Локализованные параметры



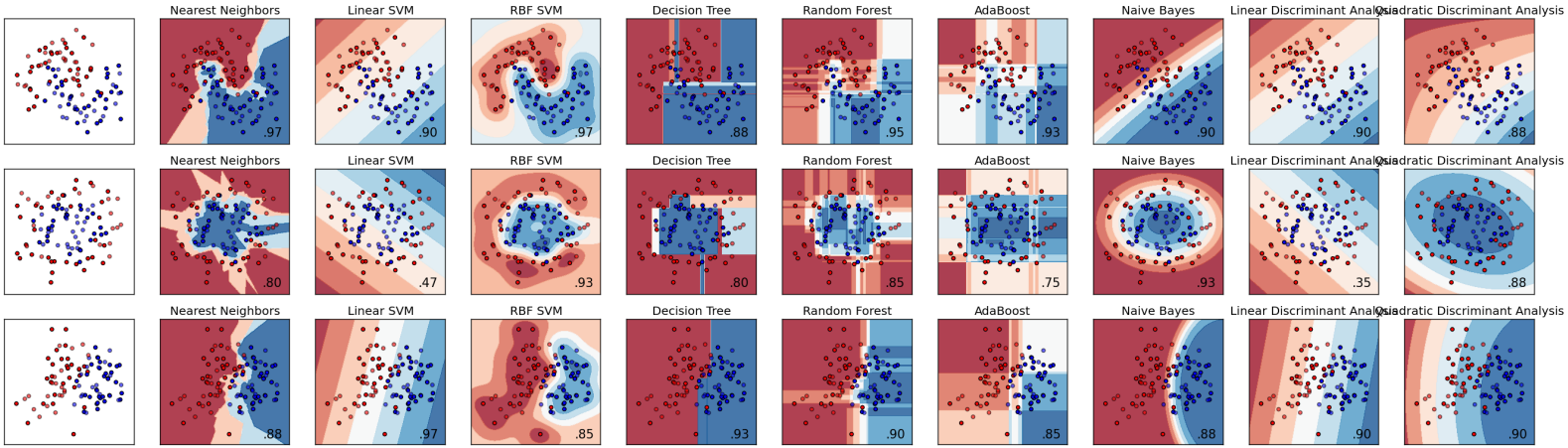
Локализованные параметры



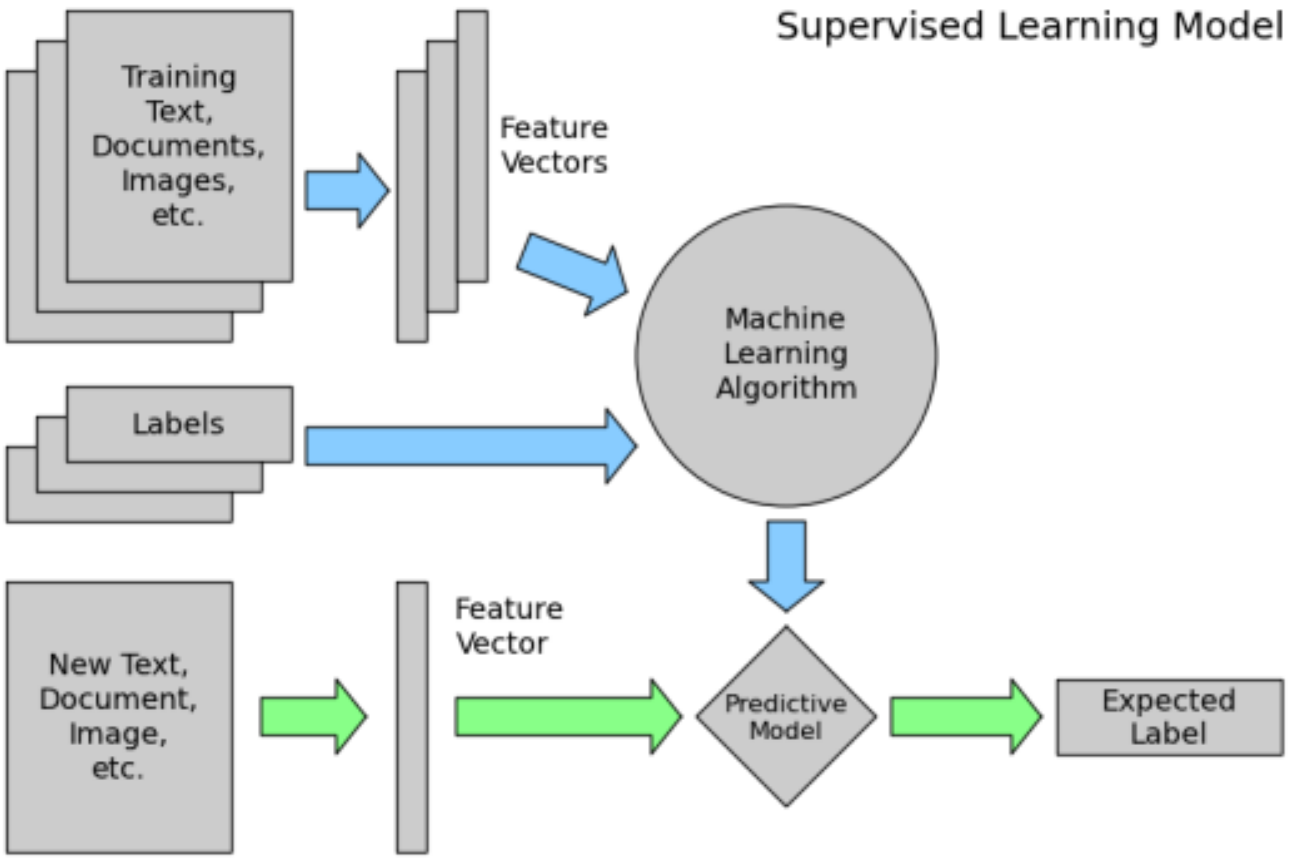
Локализованные параметры



Локализованные параметры



Признаки и обучение с учителем



Каждый объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нет.

Признаком называется отображение $f : X \to D_f$

где D_f – множество допустимых значений признака.

Если заданы признаки $f_1, \ldots, f_n$, то вектор $\mathbf{x} = (f_1(x), \ldots, f_n(x))$ называется признаковым описанием объекта $x \in X$.

Признаковые описания допустимо отождествлять с самими объектами. При этом множество $X = D_{f_1} \times \cdots \times D_{f_n}$ называют признаковым пространством.

В зависимости от характера множества D_f признаки делятся на следующие типы:

- бинарный признак: $D_f = \{0, 1\}$
- номинальный признак: D_f – конечное множество
- порядковый признак: D_f – конечное упорядоченное множество
- количественный признак: D_f – множество действительных чисел

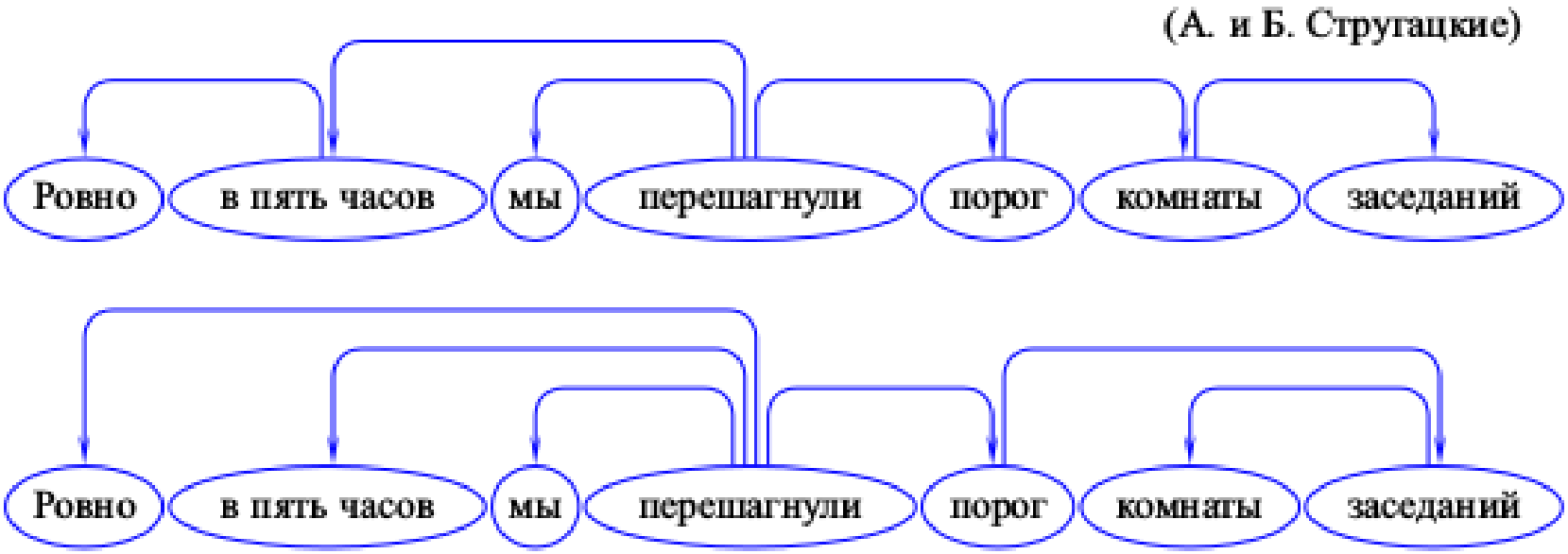
Кроме этого, ситуация с признаками бывает разной:

- признаки привязаны к точкам данных (распознавание чисел)
- признаки отражают связи между данными (синтаксический разбор предложения)
- отсутствие признаков – lazy learning

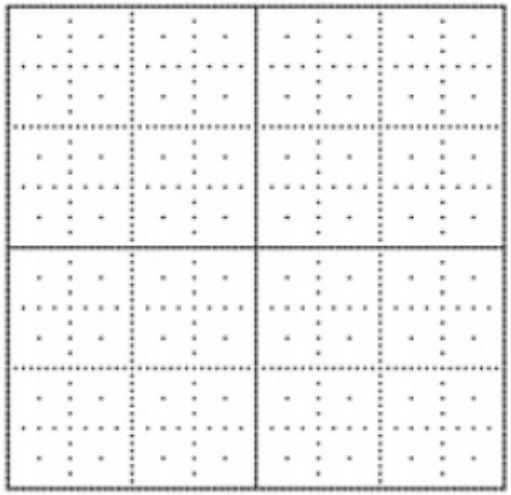
Признаковое описание – наиболее распространённый случай входных данных.

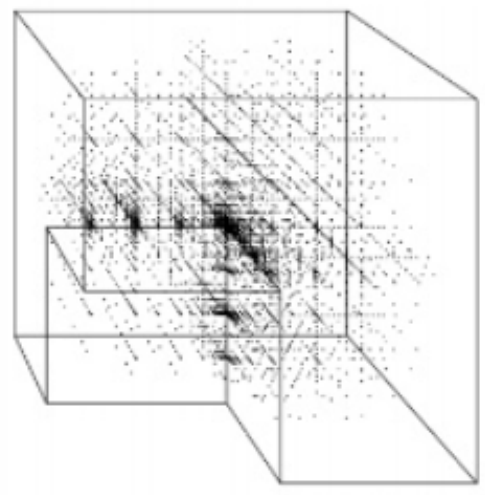
В реальности информация не разделена по признакам, их нужно синтезировать.

Синтаксический разбор – построение *проективного дерева минимального веса*, где веса подбираются на основе большой модели разобранного вручную текста (см. *Корпус русского языка*).



Проклятие размерности





- Сложность вычислений возрастает экспоненциально
- Требуется хранить огромное количество данных
- Большое число признаков являются шумными
- В линейных классификаторах увеличение числа признаков приводит к мультиколлинеарности и переобучению. Для метрических классификаторов (в пространствах с lp нормой) согласно закону больших чисел расстояния становятся неинформативны

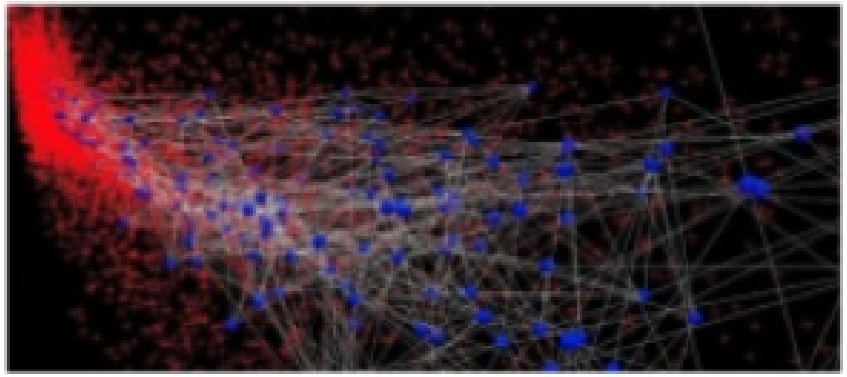
Методы подбора признаков

Два подхода к снижению размерности

Feature extraction

Data space $\rightarrow$ Feature space

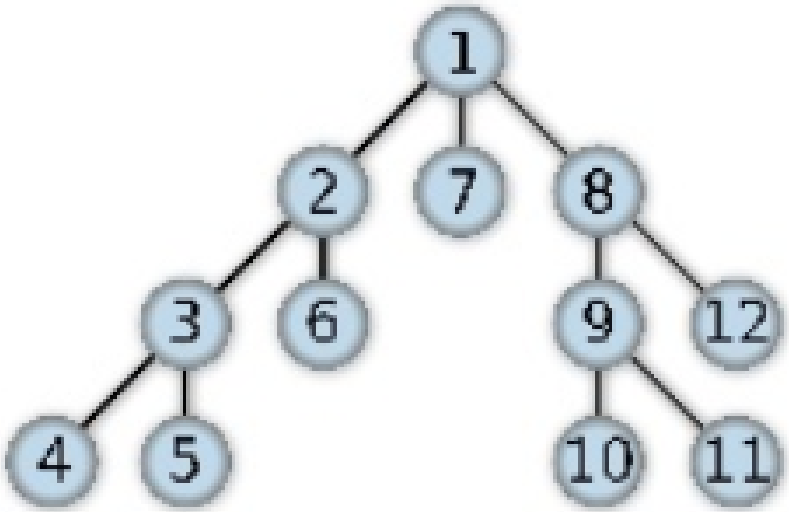
Пространство данных может быть представлено сокращённым количеством «эффективных» признаков



Feature Selection

Data space $\rightarrow$ Data subspace

Отбирается некоторое подмножество наиболее «полезных» признаков



Задача выделения/синтеза признаков

Дано. N обучающих D-мерных объектов $x_i \in \mathcal{X}$, образующих тренировочный набор данных (training data set) $\mathbf{X}$.

Найти. Найти преобразование $A : \mathcal{X} \to P$, $dim(P) = d < D$, сохранив при этом наибольшую часть «полезной информации» об $\mathcal{X}$.

Что мы рассмотрим:

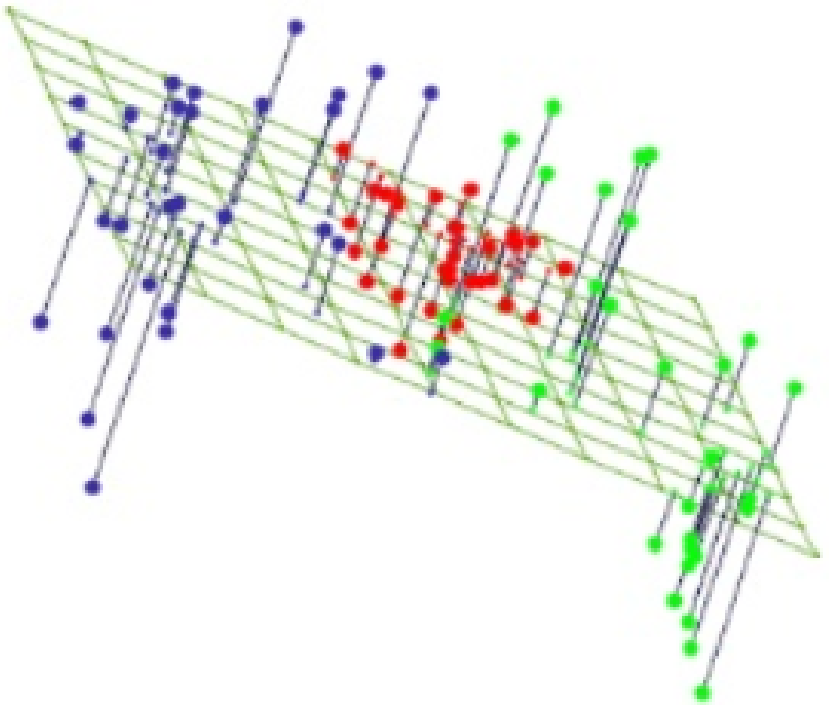
- PCA – principal component analysis
- ICA – independent component analysis
- Методы основанные на автоэнкодерах (autoencoders with bottlenecks)

Principal Component Analysis

PCA (Principal Component Analysis) – анализ главных компонент выборки. В теории информации также известен как «преобразование Карунена-Лоева».

Суть метода

Находим гиперплоскость заданной размерности, такую что ошибка проектирования выборки на данную гиперплоскость будет минимальной.



Будем искать преобразование в семействе линейных функций:

$$x = Ap + b, \text{ где}$$

- $x \in \mathcal{R}^D$ – представление объекта в исходном пространстве
- $p \in \mathcal{R}^d$ – новые координаты объекта
- $b \in \mathcal{R}^D$, $A \in \mathcal{R}^{D \times d}$

Исходные точки: $x_j = \sum_{i=1}^{D} (x_j^T a_i) a_i$

Проекции: $\tilde{x}_j = \sum_{i=1}^{d} p_{j,i} a_i + \sum_{i=d+1}^{D} b_i a_i$

Критерий выбора гиперплоскости:

$$J = \frac{1}{N} \sum_{j=1}^{N} \|x_j - \tilde{x}_j\|^2 \to \min_{q,z,b}$$

Решение будет иметь вид:

$$p_{i,j} = x_j^T a_i$$

$$b_i = \bar{x}^T a_i$$

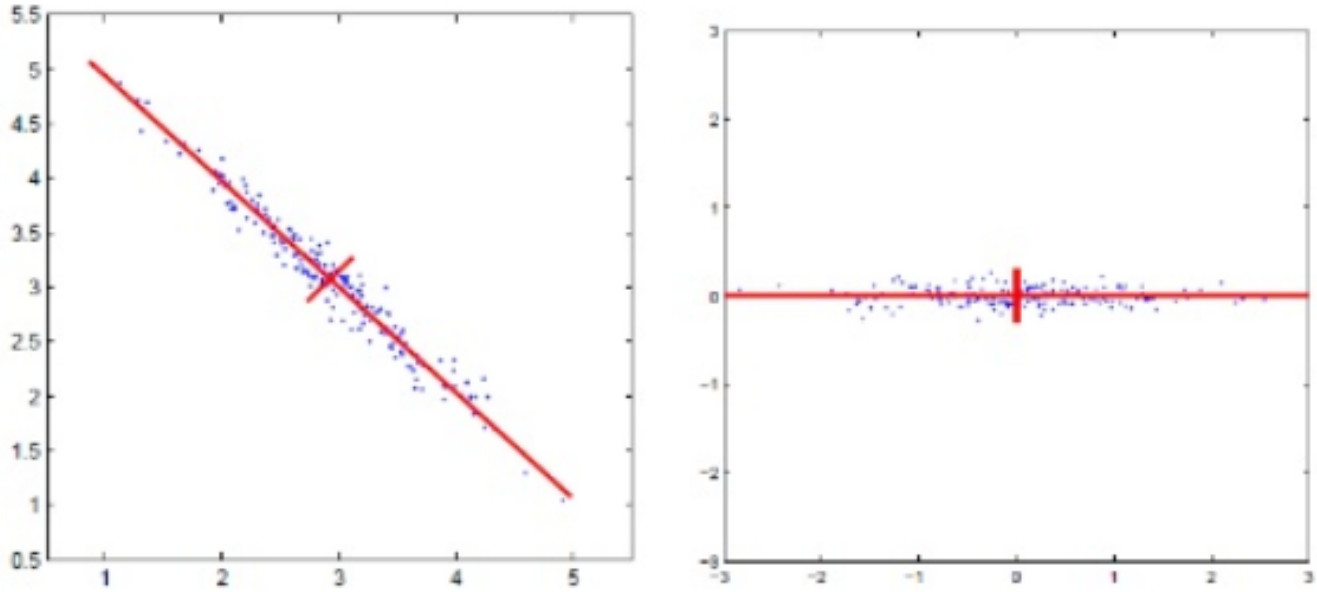
где

$$\bar{x} = \frac{1}{N} \sum_{j=1}^{N} x_j$$

$$\mathbf{R} = cov(\mathbf{X}) = \frac{1}{N}\sum_{j=1}^{N}(x_j - \bar{x})^T(x_j - \bar{x})$$

$a_i, i = 1 \dots d$ – базис из собственных векторов ковариационной матрицы $\mathbf{R}$, отвечающих d наибольших собственным значениям $\lambda_1 \geq \lambda_1 \cdots \geq \lambda_d$

Иллюстрация PCA



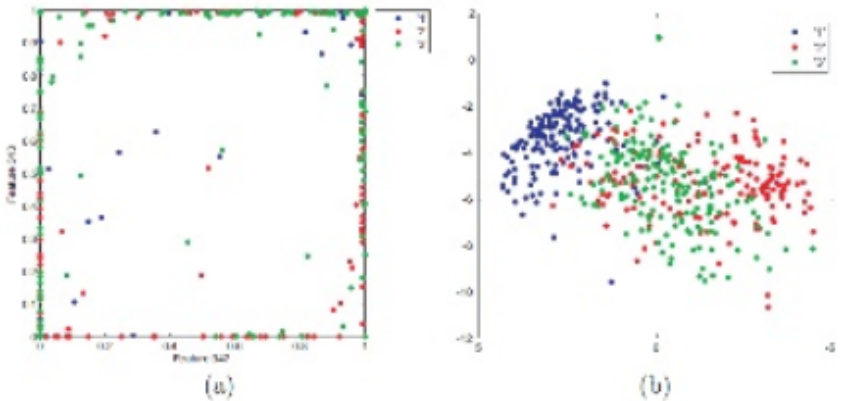
(a) Исходное пространство (b) Итоговое пространство

- Сдвигаем начало координат в центр выборки
- Поворачиваем оси так, чтобы признаки не коррелировали
- Избавляемся от координат с малой дисперсией

Иными словами...

PCA – максимизация дисперсии проекции данных

Пример распознавания рукописных данных из БД MNIST



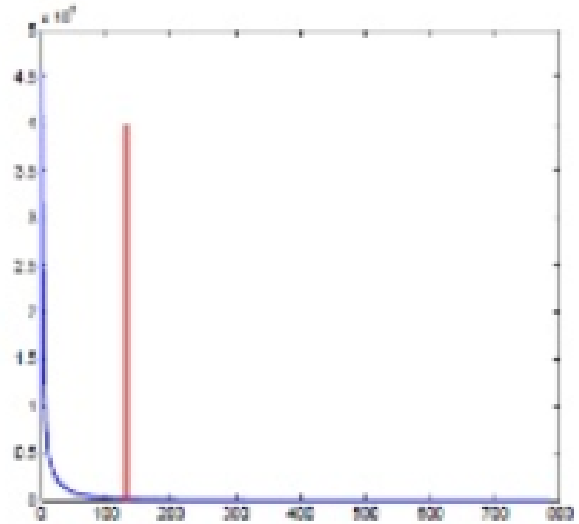
Выбор размерности редуцированного пространства

Поскольку собственные значения ковариационной матрицы **R** отсортированы в порядке убывания ($\lambda_1 \geq \lambda_1 \cdots \geq \lambda_d$)

критерий выбора размерности будет иметь вид:

$$d : \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \leq \eta$$

где $\eta \in \{0.95 \ldots 0.99\}$



Связь PCA и ICA

$$X = U\Sigma V^T$$

где

- $U(m \times m)$ – ортогональная матрица левых собственных векторов (собственные векторы матрицы XX^T)
- $V(n \times n)$ – ортогональная матрица правых собственных векторов (собственные векторы матрицы X^TX)
- $\Sigma(m \times n)$ – диагональная матрица с сингулярными числами на главной диагонали

Матрица главных компонент может быть вычислена:

$$XV = U\Sigma$$

Применение PCA

(a) Data Visualization

(b) Image procesing

(c) Prospect

(d) Data compression

Достоинства и недостатки PCA

+ Простой алгоритм
+ Можно адаптировать для любого нелинейного случая, совершив преобразование ядра (см. Kernel trick)
– Вычислять собственные веторы ковариационной матрицы в случае больших данных – проблематично
– Координаты объектов в новом пространстве неоднозначно определены

ICA – Анализ независимых компонентов

$$X = A \cdot S$$

$$x_j = a_{j,1}s_1 + a_{j,2}s_2 + \cdots + a_{j,N}s_N, j = 1, \ldots, N$$

- x_j, s_k – случайные величины
- X – наблюдаемые данные
- A – матрица смешивания
- S – неизвестный сигнал

Задача: Оценить и восстановить исходные сигналы $S = A^{-1}X$

Два предположения:

- s_j – статистически независимы $p(s_1, s_2) = p(s_1)p(s_2)$
- не-гауссово распределение

ICA – Анализ независимых компонентов: Схема решения

1. Центрируем данные: $x_i \leftarrow (x_i - \bar{x}) : \bar{x} \leftarrow \frac{1}{N} \sum_{i=1}^{N} x_i$
2. Отбеливаем данные

$$X = U\Sigma V,\ X \leftarrow U\Sigma^{-\frac{1}{2}} U^T X$$

   - $Cov(X) = I$
   - $AA^T = I$
3. Находим ортогональную матрицу A
   - Infomax
   - FastICA
   - JADE

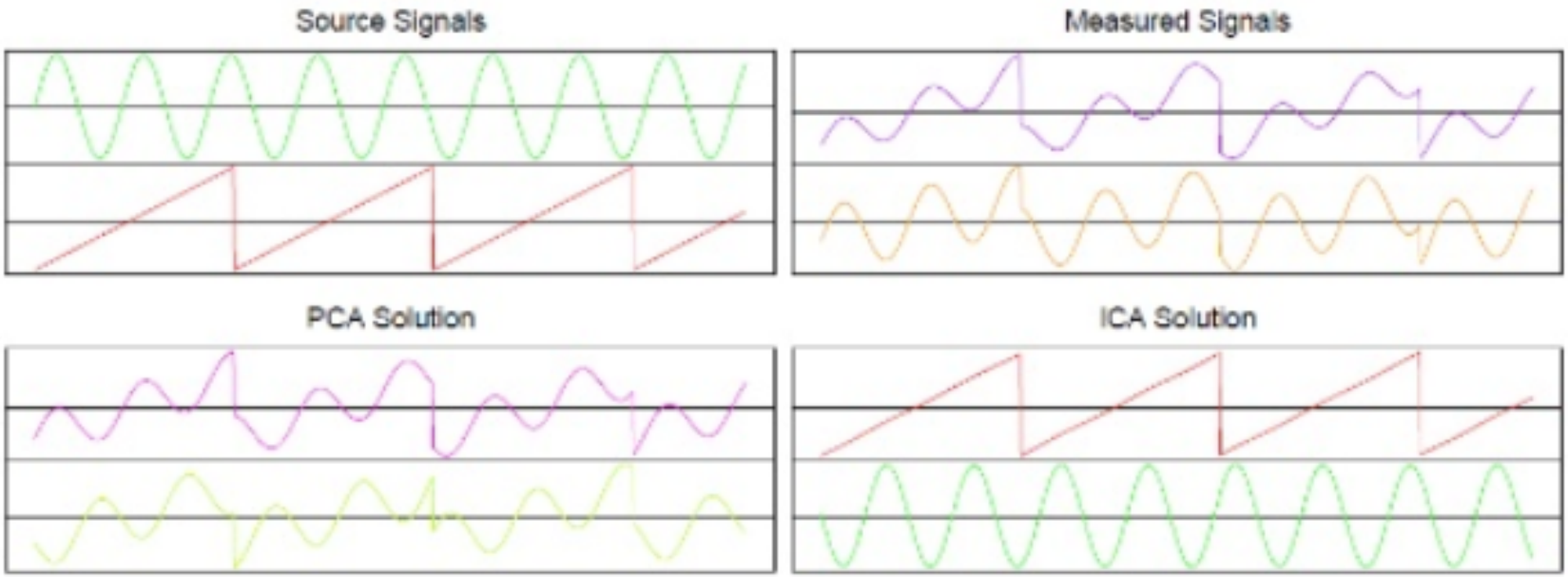
PCA vs ICA: Геометрическая интерпретация

(a) PCA
(ортогональны)

(b) ICA
(не ортогональны)

PCA vs ICA: Пример

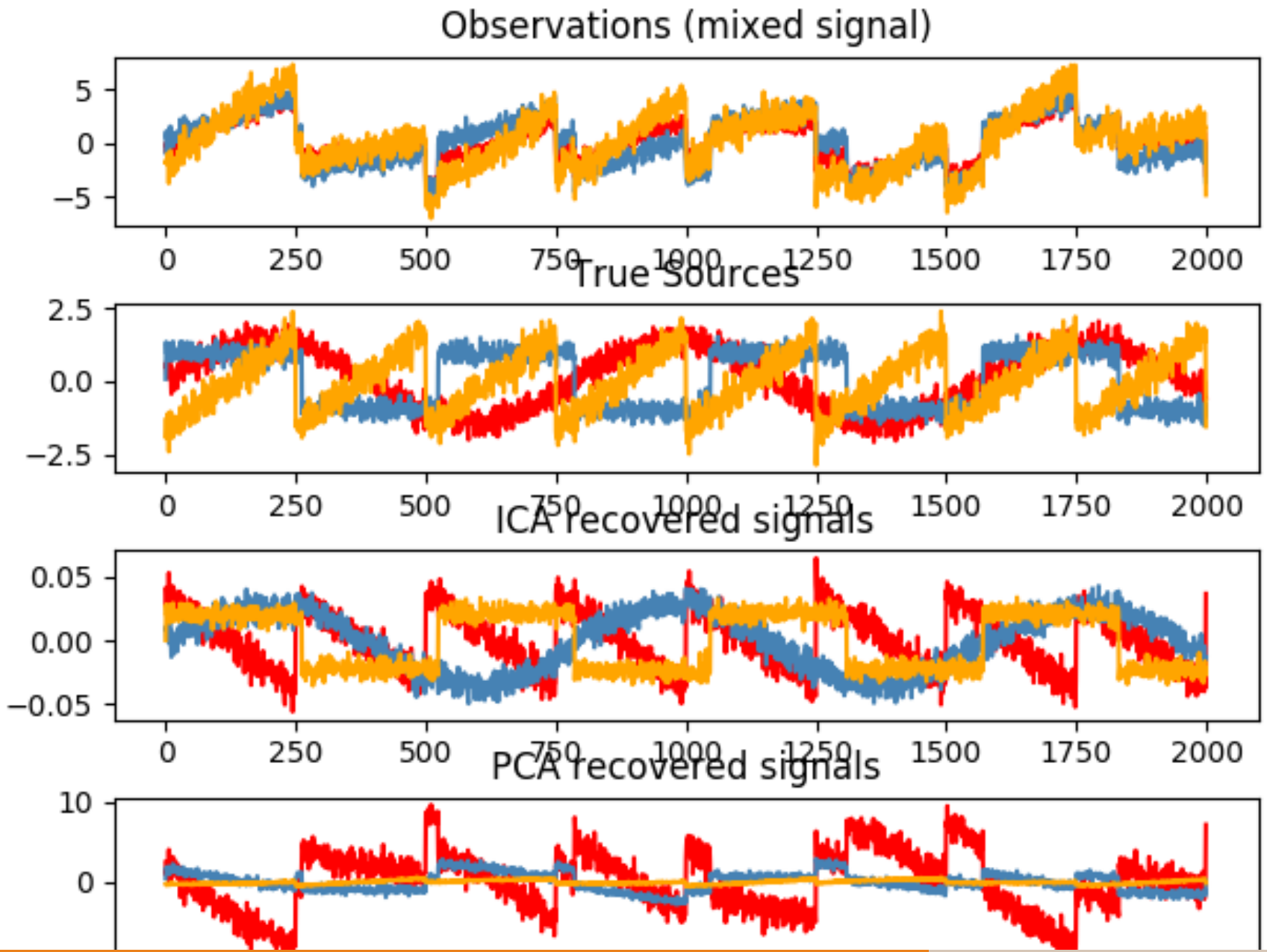
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$



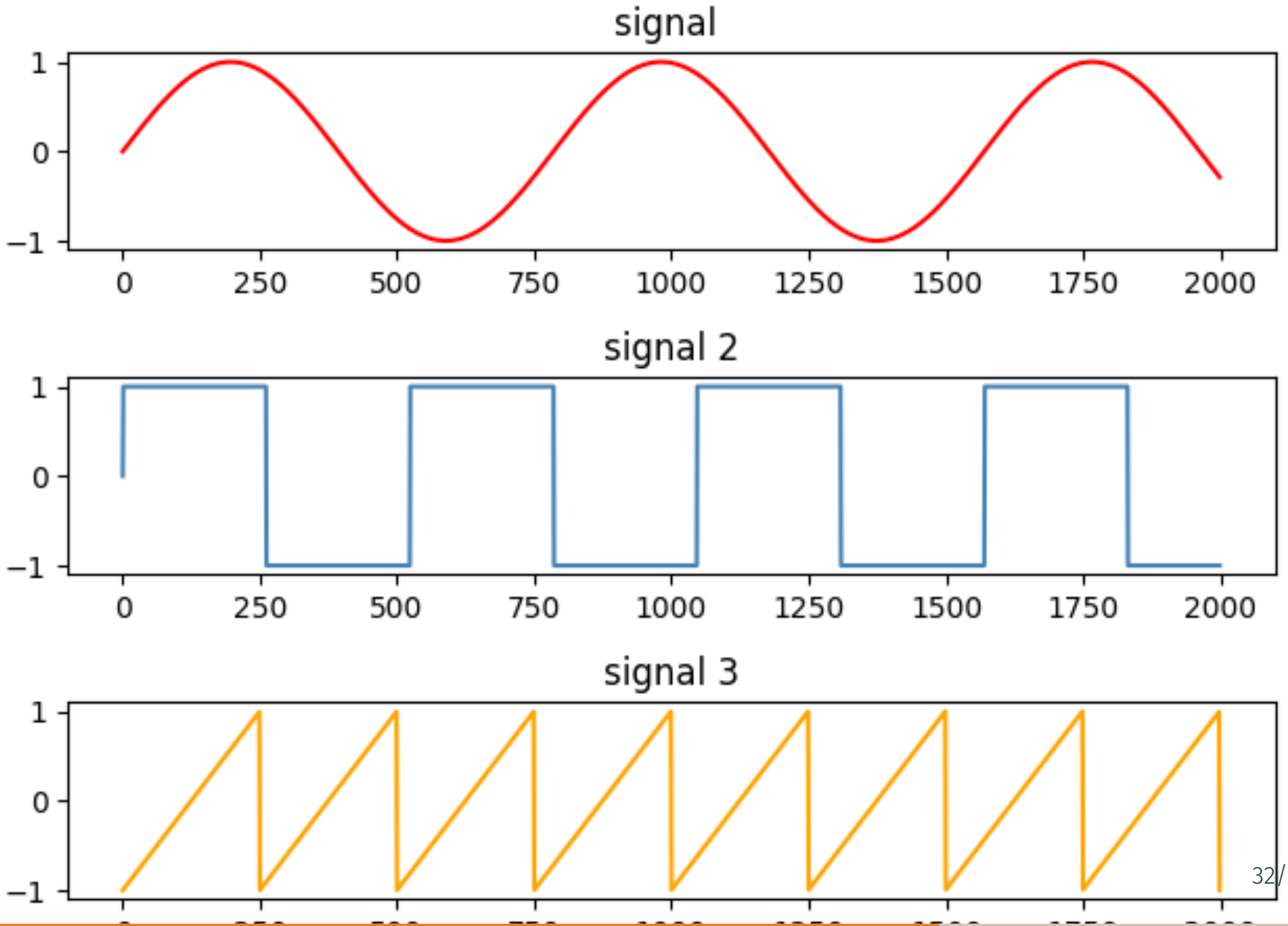
PCA vs ICA: Пример (источник)

```
np.random.seed(0)
n_samples = 2000
time = np.linspace(0, 8, n_samples)

s1 = np.sin(2 * time)  # Signal 1 : sinusoidal signal
s2 = np.sign(np.sin(3 * time))  # Signal 2 : square signal
s3 = signal.sawtooth(2 * np.pi * time)  # Signal 3: saw tooth signal

S = np.c_[s1, s2, s3]
S += 0.2 * np.random.normal(size=S.shape)  # Add noise

S /= S.std(axis=0)  # Standardize data
# Mix data
A = np.array([[1, 1, 1], [0.5, 2, 1.0], [1.5, 1.0, 2.0]])  # Mixing
    matrix
X = np.dot(S, A.T)  # Generate observations

# Compute ICA
ica = FastICA(n_components=3)
S_ = ica.fit_transform(X)  # Reconstruct signals
A_ = ica.mixing_  # Get estimated mixing matrix

# We can `prove` that the ICA model applies by reverting the unmixing.
assert np.allclose(X, np.dot(S_, A_.T) + ica.mean_)

# For comparison, compute PCA
pca = PCA(n_components=3)
H = pca.fit_transform(X)  # Reconstruct signals based on orthogonal
    components
```

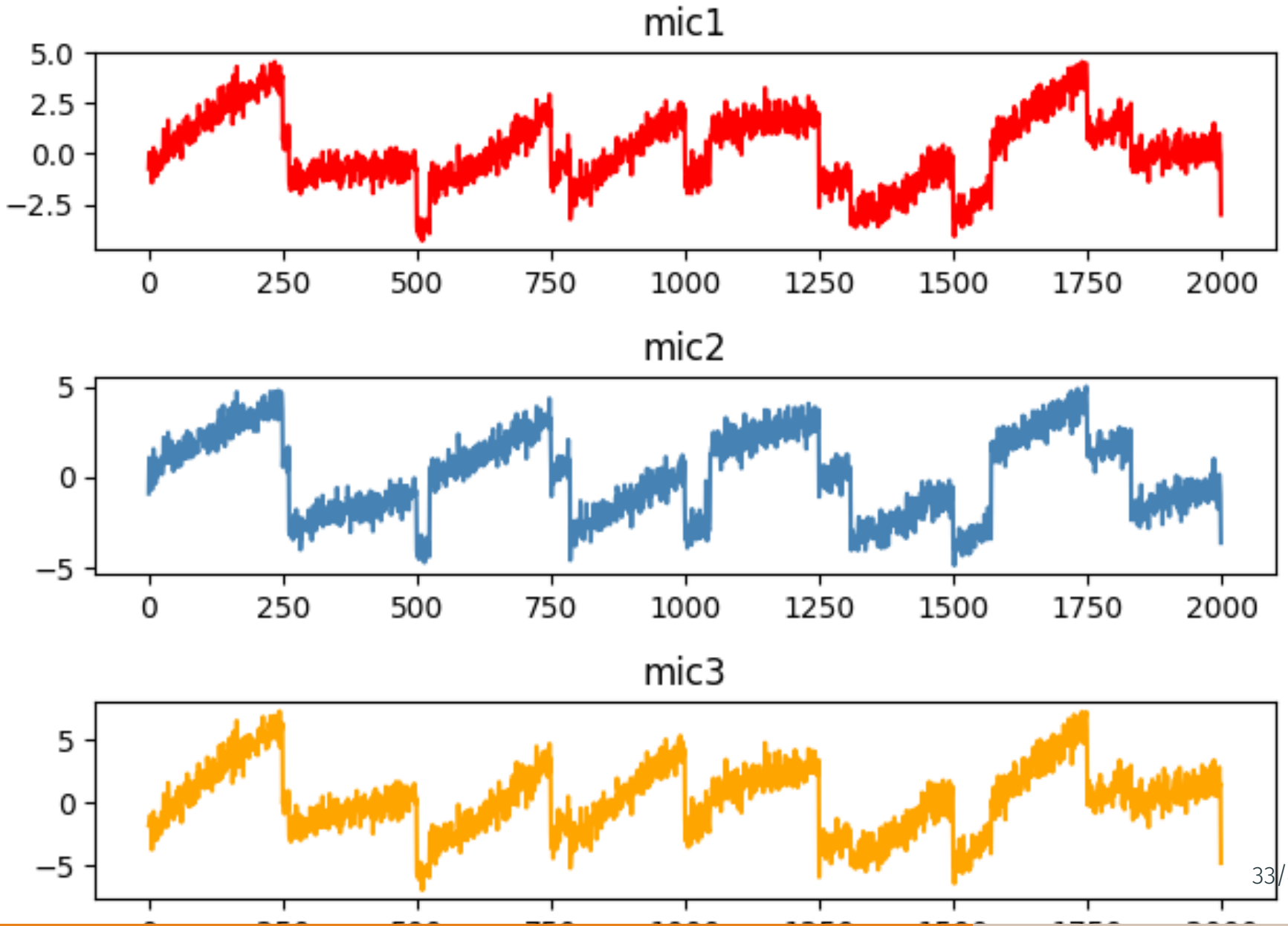
Результаты слепого разделения сигналов



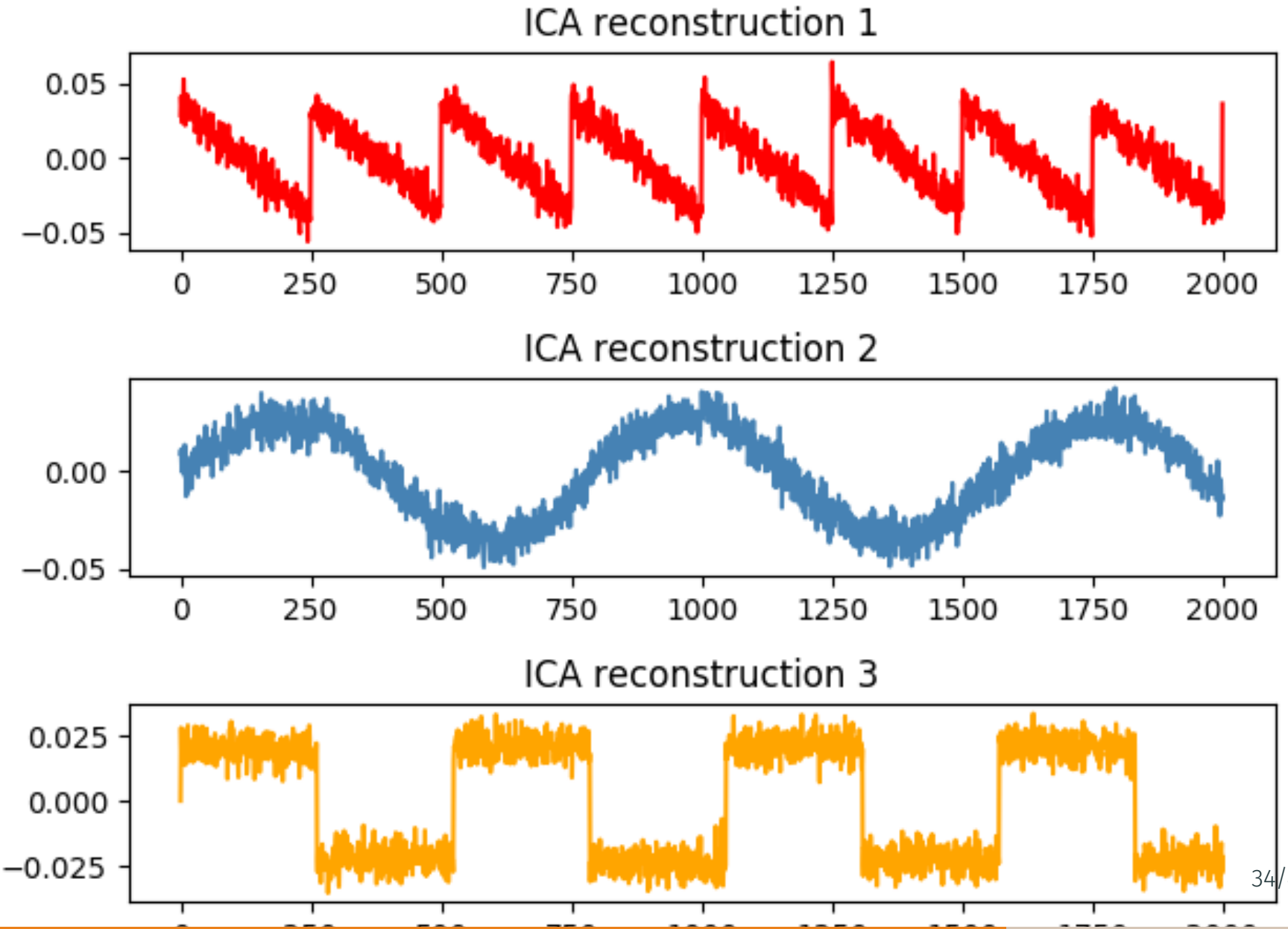
Результаты слепого разделения сигналов: Исходные данные



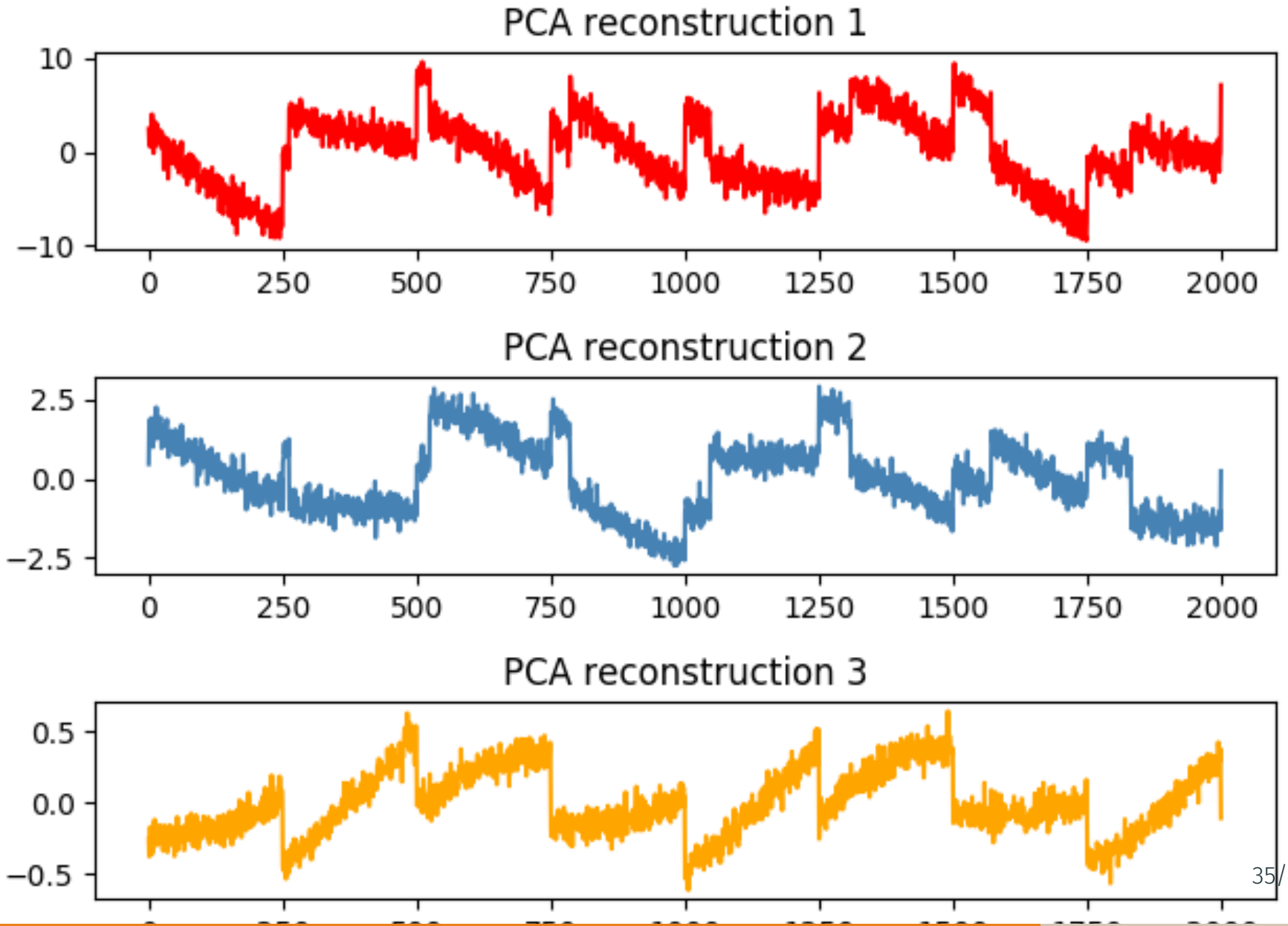
Результаты слепого разделения сигналов: Измеренные данные



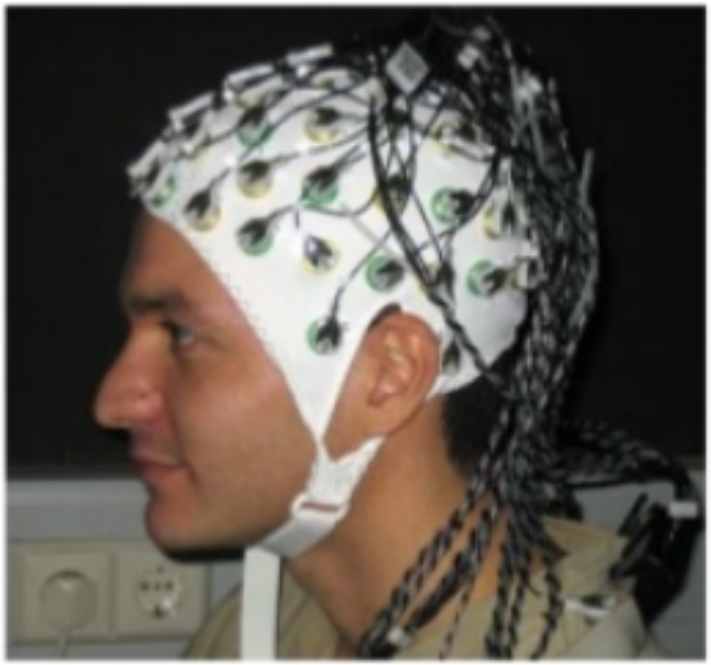
Результаты слепого разделения сигналов: ICA



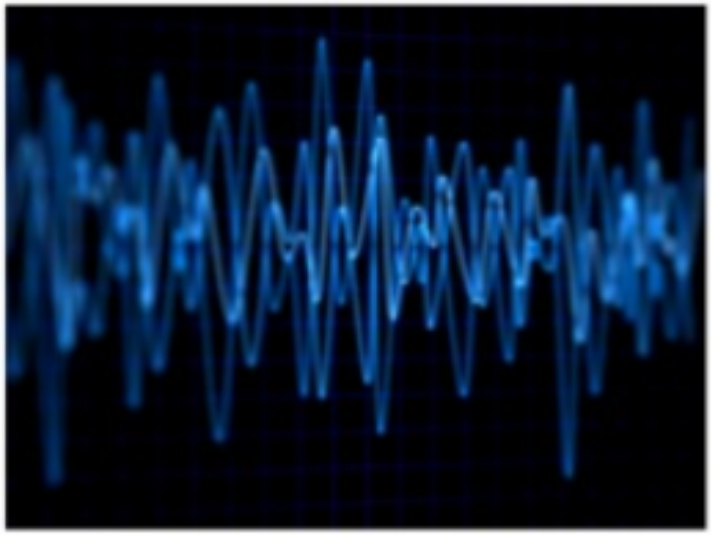
Результаты слепого разделения сигналов: PCA



Применение ICA

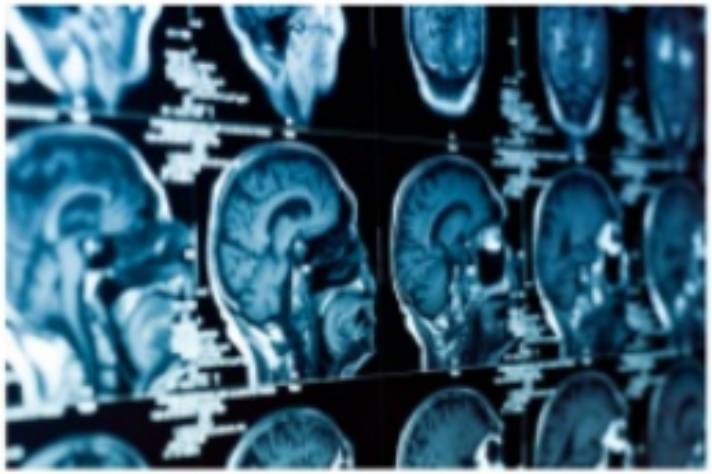


(a) EEG



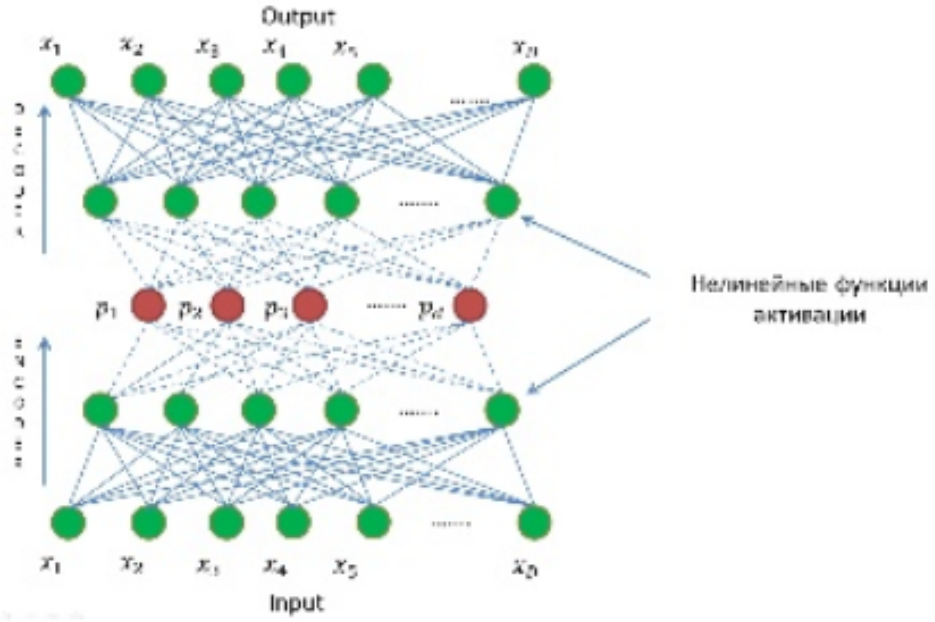
(b) Audio procesing

(c) Finance



(d) Medical data

Методы основанные на автоэнкодерах



$$J(\mathbf{w}) = \sum_{i=1}^{N} \|f(\mathbf{x}_i, \mathbf{w}) - \mathbf{x}_i\|^2 \to \min$$

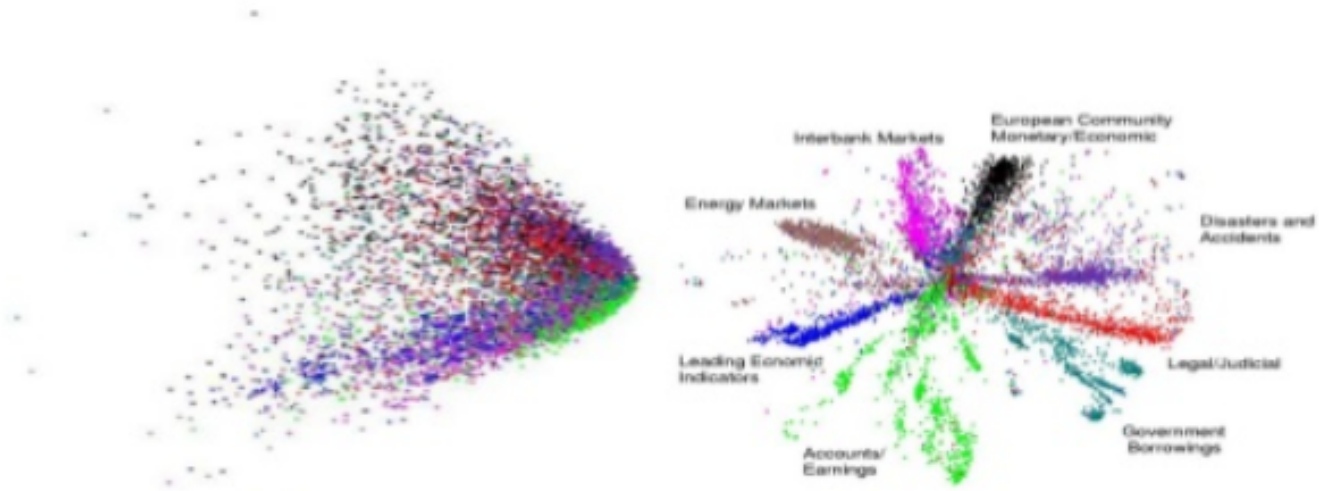
Замечание

Если в сети всего один скрытый слой, тогда результат эквивалентен PCA.

PCA vs Autoencoder

Задача визуализации тематических текстовых документов

- $D = 2000$ - "мешок слов"
- $N = 4 \cdot 10^5$ документов



(a) PCA (b) Deep Autoencoder

Задача отбора признаков

Feature Selection

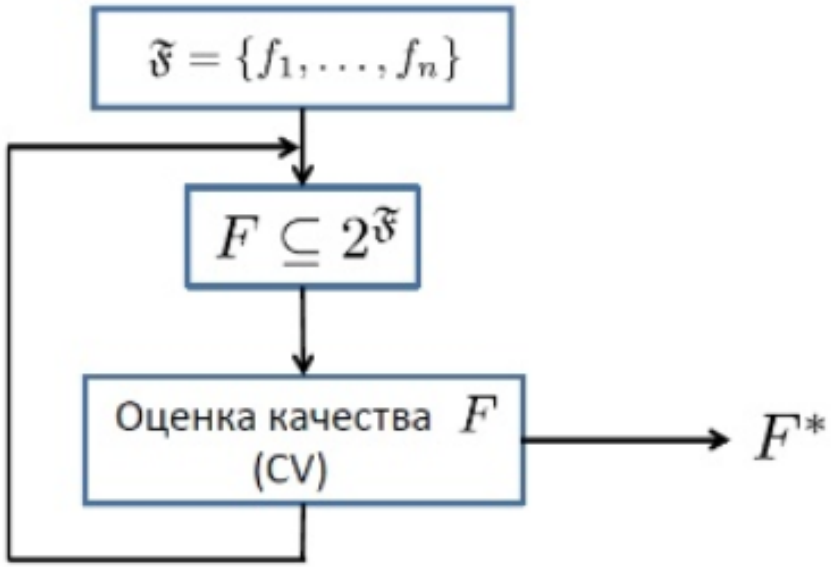
Дано. N обучающих D-мерных объектов $\mathbf{x}_i \in \mathcal{X}$, образующих тренировочный набор данных (training data set) $\mathbf{X}$, а также каждому $\mathbf{x}_i$ соответсвует метка $c_i \in \mathcal{R}$.

Найти. Найти подмножество признаков F исходного признакового пространства $\mathcal{F} = \{f_1, f_2, \ldots, f_D\}$, содержащее наиболее "информативные" признаки.

Что мы рассмотрим:

- Переборные алгоритмы
- Методы основанные на корреляции/взаимной информации
- Embedded methods

Отбор признаков “в лоб”



- Экспертный подход
- Full Search (NP hard)
- Жадные алгоритмы (Forward selection, Backward elimination, Bidirectional elimination etc.)

Жадные алгоритмы отбора признаков

Forward selection

```
function forwardselection(F, J, n):
    # F - original feature set
    # J - external criterion
    # n - parameter
    initialize F_0 = {} # empty set
    initialize Q = J(F_0) # compute score
    for j in 1..D:
        fbest = find_best_feature(J, F_j-1, F)
        F_j = add_new_feature(F_j-1, fbest) # add feature
        if J(F_j) < Q:
            jbest = j
            Q = J(F_j) # save best
        if j - jbest >= n:
            return F_jbest
```

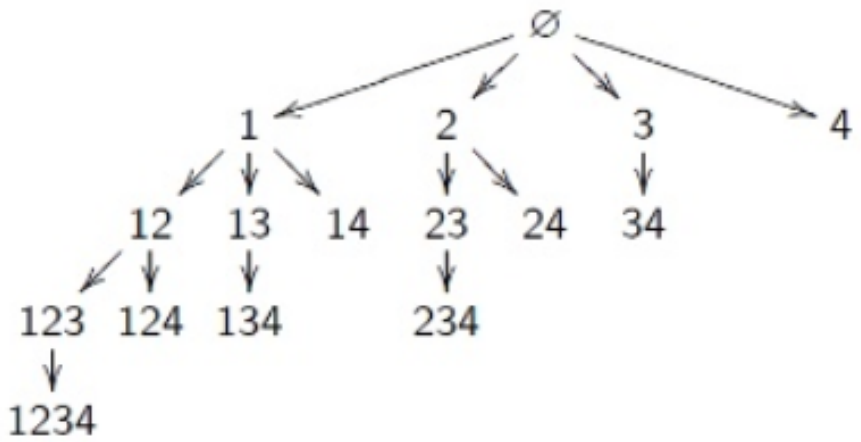
Backward elimination

- Все аналогично. Только ислючаем

Жадные алгоритмы отбора признаков

DFS. Основные идеи:

- Избегаем повторов при переборе
- Если подмножество признаков бесперспективно, то не будем пытаться его дальше наращивать.



Оценка бесперспективности:

$$\exists j: \quad J(F) \geq \eta J(F_j^*), \quad |F| \geq j + n$$

Выводы по «жадным» алгоритмам формирования признаков

- не всё то – признак, что – блестит (не все признаки полезны)
- отбор признаков происходит по внешним критериям
- любые эвристики хороши для сокращения перебора
- ...при условии что перебор устойчив по подмножествам признаков
- если не делать никакой декомпозиции, постоянно нужно переобучать алгоритм под новые данные