13:10-14:40 БИСО-1,2-16 Метод наименьших квадратов. Остаточная ошибка.

Лекция + Практическое занятие по дисциплине "Методы и алгоритмы цифровой обработки сигналов"

*Кафедра КБ-**3** "Управление и моделирование систем" Института Комплексной Безопасности и Специального Приборостроения, МИРЭА, **2020**г.*

Дата проведения занятия: 18.04.2020.

Время: 13:10-14:40.

Место/Форма проведения занятия: Дистанционное обучение/Конференция **Skype.** Ссылка на видео-конференцию будет предоставлена непосредственно перед занятием по почте.

Постоянная ссылка: https://colab.research.google.com/drive/160GFQucFiMR-egmaLBQOlww3RH9Z0uOZ

Цель: Анализ остаточной ошибки при оптимальной фильтрации. Разбор примеров решения задач.

Материал прошлых лекций: Метод МНК позволяет решить переопределенную задачу вида:

$$V\vec{c} = \vec{x}$$

Путем минимизации критерия:

$$Q = \left(V\vec{c} - \vec{x}\right)^T \left(V\vec{c} - \vec{x}\right)$$

Такое решение можно представить в виде:

$$\vec{c} = \left(V^T V\right)^{-1} V^T x$$

Решение является оптимальным в том смысле, что не существует другого набора коэффициентов $\vec{c}$, который обеспечивал бы меньшее значение критерия Q.

Остаточная ошибка оптимального линейного фильтра

Матрица V составлена из вектор-столбцов (регрессоров) - см. предыдущую лекцию

$$V = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1K} \\ a_{21} & a_{22} & \dots & a_{2K} \\ \vdots & \vdots & \dots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NK} \end{pmatrix} = \begin{pmatrix} \vec{a}_1 & \vec{a}_2 & \dots \vec{a}_K \end{pmatrix}$$

Здесь $\vec{a}_k$ - регрессор (другое название - терм) - вектор столбец.

Введем обозначение

$$R = V^T V$$

R - это квадратная (в отличие от V) матрица размером $K \times K$. Каждый элемент матрицы **R** можно представить в виде скалярного произведения столбцов матрицы V:

$$r_{k,m} = \sum_{n=1}^{N} a_{k,n} a_{m,n}$$

Легко заметить, что выражение для $r_{k,m}$ является оценкой корреляции между столбцами k и m матрицы V. Если предположить, что размерность регрессоров N стремится к бесконечности, то оценка корреляции будет стремиться к истинному значению корреляции между случайными процессами выступающими в качестве регрессоров. Будем называть матрицу R **корреляционной матрицей.**

Элементы корреляционной матрицы обладают следующими свойствами (*приведите доказательнство*):

Элементы корреляционной матрицы обладают следующими свойствами (*приведите доказательство*):

- $r_{k,m} = r_{m,k}$ - для действительнозначных регрессоров.
- $r_{k,k} \geq 0$

Также можно доказать, что корреляционная матрица является симметричной неотрицательно определенной матрицей. Т.е. $\forall \vec{y}$ билинейная форма вида неотрицательна:

$$\vec{y}^T R \vec{y} \geq 0$$

Это свойство позволяет сократить объем вычислений при поиске R^{-1} (можно использовать разложение Холесского, см.**numpy.linalg.cholesky(a))**.

Введем обозначение

$$\vec{\rho} = V^T x$$

Каждый элемент вектора $\vec{\rho}$ можно представить в виде:

$$\rho_k = \sum_{n=1}^{N} a_{n,k} x_n$$

Т.е. ρ_k - определяет оценку корреляции между k-ым вектором-регрессором $\vec{a}_k$ и вектором требуемого выхода $\vec{x}$.

В таких обозначениях решение переопределенной системы линейных уравнений методом наименьших квадратов можно представить в виде:

$$\vec{c} = \left(V^T V\right)^{-1} V^T x = R^{-1} \vec{\rho} \tag{1}$$

Уравнение (???) в форме:

$$R\vec{c} = \vec{\rho} \tag{2}$$

называется *уравнением Винера*.

Рассмотрим критерий оптимизации - сумму квадратов отклонений:

$$Q = (V\vec{c} - \vec{x})^T (V\vec{c} - \vec{x}) = (\vec{c}^T V^T - \vec{x}^T)(V\vec{c} - \vec{x}) \\ = \\ \vec{c}^T V^T V\vec{c} - \vec{c}^T V^T \vec{x} - \vec{x}^T V\vec{c} + \vec{x}^T \vec{x}$$

Учитывая, что для вычислений в действительных числах $\vec{c}^T V^T \vec{x} = \vec{x}^T V\vec{c}$ (*докажите*):

$$Q(\vec{c}) = \vec{c}^T V^T V\vec{c} - 2\vec{c}^T V^T \vec{x} + \vec{x}^T \vec{x} = \vec{x}^T \vec{x} - 2\vec{c}^T \vec{\rho} \\ + \vec{c}^T R\vec{c} \tag{3}$$

Подставим уравнение Винера $R\vec{c} = \vec{\rho}$ в выражение (???):

$$Q_{opt} = \vec{x}^T \vec{x} - 2\vec{c}^T \vec{\rho} + \vec{c}^T \vec{\rho} = \\ \vec{x}^T \vec{x} - \vec{c}^T \vec{\rho} = \\ \vec{x}^T \vec{x} - \vec{c}^T R\vec{c} \tag{3}$$

- Остаточная ошибка оптимального фильтра Q_{opt} - это квадрат модуля кратчайшего расстояния от линейной оболочки, образованной столбцами (регрессорами) матрицы V до точки $\vec{x}$.
- Остаточную ошибку оптимального фильтра Q_{opt} нельзя уменьшить в рамках заданного набора регрессоров и вектора $\vec{x}$.
- Выражение $\vec{c}^T R\vec{c} \geq 0$, поэтому значение ошибки лежит в диапазоне $0 \ldots \vec{x}^T \vec{x}$.

Расчитаем остаточную ошибку для задачи **1** из предыдущей лекции:

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

N     = 100
alpha = 2
beta  = -3
```

```
x     = np.array([np.cos( np.pi/N*4*np.arange(0.,N) )+np.cos( np.pi/N*20*np.arange(0.,N
) )+np.cos( np.pi/N*30*np.arange(0.,N) )]).transpose()
eta   = 0.9 * np.random.randn( N,1 )
xi    = 2.1 * np.random.randn( N,1 )
a     = alpha*x + eta
b     = beta*x + xi

V     = np.hstack((a,b))
R     = V.transpose()@V
rho   = V.transpose()@x
c     = np.linalg.inv(R)@rho
print('c={0}'.format(c))

sol_x = V@c
```

```
c=[[ 0.34311871]
 [-0.08261957]]
```

Сначала рассчитаем ошибку напрямую $(\hat{\vec{x}}-\vec{x})^T(\hat{\vec{x}}-\vec{x}) = \sum_{n=1}^{N}(\hat{x}_n - x_n)^2$

In [5]:

```
Q = (sol_x-x).transpose()@(sol_x-x)
print( 'Q={0}\n'.format(Q) )
```

```
Q=[[11.42294313]]
```

Теперь будем использовать формулу $Q_{opt} = \vec{x}^T\vec{x} - \vec{c}^T R\vec{c}$

In [6]:

```
Q = x.transpose()@x - c.transpose()@R@c
print( 'Q={0}\n'.format(Q) )
```

```
Q=[[11.42294313]]
```

Максимальная ошибка для данного примера:

In [7]:

```
Qmax = x.transpose()@x
print( 'Qmax={0}\n'.format(Qmax) )
```

```
Qmax=[[150.]]
```

Разделив сумму квадратов ошибок на максимальное теоретическое значение суммы квадратов отклонений, получим относительную ошибку:

In [8]:

```
print('Относительная ошибка {0}'.format(Q/Qmax))
```

```
Относительная ошибка [[0.07615295]]
```

В цифровой обработке традиционно используются логарифмические величины для относительной ошибки:

In [9]:

```
nmseDb = 10*np.log10(Q/Qmax)
print('Относительная ошибка {0:5.2f}dB'.format(nmseDb[0][0]))
```

```
Относительная ошибка -11.18dB
```

Сравним с относительной ошбкой во входных векторах $\vec{a}$ и $\vec{b}$:

In [0]:

```
nmse_a      = (a-x*alpha).transpose()@(a-x*alpha)
nmse_b      = (b-x*beta).transpose()@(b-x*beta)
rms_x       = x.transpose()@x
nmse_a_db   = 10*np.log10(nmse_a[0][0]/rms_x[0,0]/(alpha*alpha))
nmse_b_db   = 10*np.log10(nmse_b[0][0]/rms_x[0,0]/(beta*beta))
print( 'Относительные ошибки входных данных a={:5.2f}dB b={:5.2f}dB'.format( nmse_a_db, n
mse_b_db ) )
```

```
Относительные ошибки входных данных a=-8.65dB b=-4.84dB
```

Таким образом, относительная ошибка

- по сравнению со входным вектором $\vec{a}$ улучшилась на **-8.65-(-10.47)=1.82**дБ, т.е. в $10^{1.82/10} = 1.52$ раза.
- по сравнению со входным вектором $\vec{b}$ улучшилась на **-4.84-(-10.47)=5.63**дБ, т.е. в $10^{5.63/10} = 3.65$ раза.

Рассмотрим пример с очисткой звукового файла. Нам понадобится запись с "чистым" звуком (вектор $\vec{x}$).

In [4]:

```
from IPython.display import Audio
x = np.loadtxt("https://raw.githubusercontent.com/RF-Lab/lab_sources/master/x_sound_16000
.txt")
x = x[0:250000]
Audio(x,rate=16000)
```

Out[4]:

Your browser does not support the audio element.

Добавим к чистому сигналу шум - получим вектор $\vec{a} = \vec{x} + \vec{\eta}$

In [7]:

```
eta   = 500*np.random.randn( *x.shape )
a     = x + eta
Audio(a,rate=16000)
#plt.plot(x)
#plt.plot(eta)
```

Out[7]:

Your browser does not support the audio element.

$$\hat{x}_n = \sum_{m=-M}^{M} c_m x_{n-m}$$

In [8]:

```
M = 512
V = np.zeros((a.shape[0],2*M+1))
for k in np.arange(-M,M+1):
  V[:,k]  = np.roll( a, k )
```

```
print('Shape of V is {}'.format(V.shape))

R = V.transpose() @ V

print('Shape of R is {}'.format(R.shape))
#print(R)

rho = V.transpose()@x

c   = np.linalg.inv(R) @ rho
#print('Filter coefs={}'.format(c))

sol_x   = V @ c

Audio(sol_x,rate=16000)
```

```
Shape of V is (250000, 1025)
Shape of R is (1025, 1025)
```

Out[8]:

Your browser does not support the audio element.

Вопросы и задания

- Построить частотный отклик полученного фильтра **(freqz)**
- Построить графики спектральной плотности сигнала на ходе в фильтр**,** сигнала без шума и сигнала на выходе фильтра **(pwelch)** и спектральную плотность шума**.**
- Проинтерпретировать результаты в частотном домене**.** Пояснить стратегию фильтрации в частотном домене**(**где сигнал усиливался, а где подавлялся**).**