Лабораторная работа №3

рынок: **Китай**
период: **2018 год**

Седунов Илья,
Альперович Вадим,
Славутин Александр,
17ПМИ.

In [7]:
```
import random
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
import warnings
import seaborn as sns
from seaborn import set_style
set_style('dark')
warnings.simplefilter('ignore')
```

Подготовка модели.

- Выберите на рынке 20 активов (N=20).
- По наблюдениям за 2018 год оцените математические ожидания доходностей и матрицу ковариаций доходностей (используйте выборочную матрицу ковариаций).
- Найденные вектор средних и матрица ковариаций будут далее использованы в экспериментах как «истинные» вектор E=(E1, E2, …, EN) и матрица ковариаций (σi,j).
- Убедитесь, что матрица ковариаций невырожденная (если она близка к вырожденной, то измените состав активов).

In [8]:
```
# Сбор случайных 250 активов Китайского рынка за 2018 год

# random.seed(100)

df = pd.read_excel('data/china_stocks.xlsx').drop_duplicates(['Symbol'])
symbols = list(df['Symbol'])
symbols = random.choices(symbols, k=1000)
print('Количество китайских тикеров =', len(symbols))
```

```
Количество китайских тикеров = 1000
```

In [9]:
```
from IPython.display import clear_output
start = "2018-01-01"
end   = "2018-12-31"
stocks = {}

for symbol in tqdm_notebook(symbols):
    stock = yf.download(symbol, start=start, end=end, progress=False)
    if stock[~np.isnan(stock['Close'])].shape[0] < 220:
        print('Drop because of nan values', symbol)
        continue
    stock['return'] = stock['Close'] / stock['Close'].shift(1)
    stock['log_return'] = np.log(stock['return'])
    stocks[symbol] = stock
clear_output()
print('Акции собраны!')

# удаление пустых записей

for symbol in symbols:
    try:
        stocks[symbol] = stocks[symbol].dropna()
    except:
        continue
```

```
Акции собраны!
```

In [10]:
```python
# сбор общей статистики

stock_stat = pd.DataFrame(columns=['symbol',  'E', 'Sigma', 'mean_vol', 'n_observations'])
for symbol in tqdm_notebook(symbols):
    try:
        stock = stocks[symbol].dropna()
        stock_stat.loc[symbol] = [symbol,
                                  stock['log_return'].mean(),
                                  stock['log_return'].std(),
                                  stock['Volume'].mean(),
                                  len(stock)]
    except:
        continue
stock_stat.dropna(inplace=True)
print('Осталось активов после обработки', len(stock_stat))
```

100% 1000/1000 [00:08<00:00, 112.29it/s]

```
Осталось активов после обработки 771
```

In [19]:
```python
def get_return_mean_cov(df, sse_components=False):
    # получить по выбранным активам матрицу их доходностей,
    # вектор средних доходностей и матрицу ковариации

    r_matrix = {}
    if sse_components:
        for i in range(len(df)):
            symbol = df.index[i]
            r_matrix[symbol] = sse_stocks[symbol]['log_return']
    else:
        for symbol in df['symbol']:
            r_matrix[symbol] = stocks[symbol]['log_return']
    r_df = pd.DataFrame(r_matrix).dropna()
    return r_df.values, r_df.mean().values, r_df.cov().values

def plot_mean_var_map(df,x='Sigma', y='E', title='Карта активов:σ от E', figsize=(12, 6)):
    # получить карту риск от дохоности

    ax = df.plot(x=x, y=y, s=np.log(df['mean_vol']**3),
                         kind='scatter',
                         figsize=figsize,
                         edgecolor='black',
                         grid=True)
    plt.xlabel('Sigma', size=15)
    plt.ylabel('E', size=15)
    plt.title(title, size=16)
```

Выберем 20 активов с наибольшей платой за риск

In [316]:
```
stock_stat['sharp'] = (stock_stat.E)**2 / stock_stat.Sigma
selected20 = stock_stat.sort_values(['sharp'], ascending=False).iloc[:20]
selected20
```

Out[316]:

	symbol	E	Sigma	mean_vol	n_observations	sharp
300266.SZ	300266.SZ	-0.006689	0.034849	3.210197e+07	242	0.001284
300032.SZ	300032.SZ	-0.006487	0.034231	1.301785e+07	242	0.001229
002676.SZ	002676.SZ	-0.007110	0.043493	2.116599e+07	242	0.001162
000533.SZ	000533.SZ	-0.005908	0.033415	1.564525e+07	242	0.001045
002512.SZ	002512.SZ	-0.005061	0.025443	1.030791e+07	242	0.001007
002178.SZ	002178.SZ	-0.005503	0.033449	2.348156e+07	242	0.000905
002210.SZ	002210.SZ	-0.005018	0.028322	2.598738e+07	242	0.000889
002573.SZ	002573.SZ	-0.004757	0.025483	9.584909e+06	242	0.000888
002617.SZ	002617.SZ	-0.004140	0.021047	1.052856e+07	242	0.000814
300355.SZ	300355.SZ	-0.004902	0.029514	2.779174e+07	242	0.000814
300083.SZ	300083.SZ	-0.004881	0.029478	2.101371e+07	242	0.000808
000056.SZ	000056.SZ	-0.004263	0.022853	5.172684e+06	242	0.000795
300415.SZ	300415.SZ	-0.004457	0.026538	3.406898e+06	242	0.000749
600595.SS	600595.SS	-0.004245	0.025032	1.195527e+07	242	0.000720
300055.SZ	300055.SZ	-0.004330	0.026337	7.706688e+06	242	0.000712
000839.SZ	000839.SZ	-0.004304	0.026390	3.772130e+07	242	0.000702
603696.SS	603696.SS	-0.004372	0.027874	1.725513e+06	242	0.000686
300320.SZ	300320.SZ	-0.004467	0.029484	6.474994e+06	242	0.000677
002496.SZ	002496.SZ	-0.004416	0.028997	1.818392e+07	242	0.000672
002043.SZ	002043.SZ	-0.003772	0.021387	8.452362e+06	242	0.000665

Получим вектор средних и выборчную матрицу ковариации выбранных акций

In [317]:
```
return_matrix, mean_vec, cov_matrix = get_return_mean_cov(selected20)
return_matrix.shape, mean_vec.shape, cov_matrix.shape
```

Out[317]:
```
((242, 20), (20,), (20, 20))
```

Посмотрим на вектор собственных значений

In [318]:
```
eigenvec = np.linalg.eigvals(cov_matrix)
# np.around(eigenvec,4)
print('Conditional number: ', max(eigenvec) /  min(eigenvec))
```

```
Conditional number:  35.2991644198731
```

Попытаемся выбрать активы с наилучшей обусловленностью

In [320]:
```
gen_lst = []
for i in tqdm_notebook(range(2500)):
    selected20 = stock_stat.sample(35).sort_values(['sharp'], ascending=False).iloc[:20]
    return_matrix, mean_vec, cov_matrix = get_return_mean_cov(selected20)
    eigenvec = np.linalg.eigvals(cov_matrix)
    conditional_number = max(eigenvec) /  min(eigenvec)
    gen_lst.append((conditional_number, selected20))
```

100% 2500/2500 [03:51<00:00, 10.81it/s]

In [354]:
```
gen_lst = sorted(gen_lst, key=lambda x: x[0], reverse=True)
selected20 = gen_lst[0][1]
return_matrix, mean_vec, cov_matrix = get_return_mean_cov(selected20)
eigenvec = np.linalg.eigvals(cov_matrix)
print('Conditional number: ', max(eigenvec) /  min(eigenvec))
```

```
Conditional number:  216.8678620697297
```

In [322]:
```
from cvxopt import matrix, solvers
from scipy.optimize import minimize


def risk_portfolio(X, cov_matrix):
    return np.sqrt(np.dot(np.dot(X, cov_matrix), X.T))


def objective_function(X, mean_vec, cov_matrix, b):
    return (-np.dot(mean_vec, X)) + b * np.dot(np.dot(X, cov_matrix), X.T)


def optimize_portfolio(mean_vec,
                       cov_matrix,
                       b,
                       bounds,
                       objective_function=objective_function,
                       cvxopt=False):
    if cvxopt:
        r_avg = matrix(mean_vec)
        sigma = matrix(b*cov_matrix)
        n = mean_vec.shape[0]
        P = sigma
        q = matrix(-mean_vec)

        # inequality constraint
        G = matrix( -np.identity(n) )
        h = matrix( np.zeros((n,1)) )

        # equality constraint Ax = d; captures the constraint sum(x) == 1
        A = matrix(1.0, (1,n))
        d = matrix(1.0)
        sol = solvers.qp(P, q, G, h, A, d, show_progress=False)
        clear_output()
        return np.array([x for x in sol['x']])

    else: # scipy.minimize
        N = cov_matrix.shape[0]
        X = np.ones(N)
        X = X / X.sum()
        bounds = bounds * N

        constraints=[]
        constraints.append({'type': 'eq',
                            'fun': lambda X: np.sum(X) - 1.0})

        return minimize(objective_function, X,
                        args=(mean_vec, cov_matrix, b), method='SLSQP',
                        constraints=constraints,
                        bounds=bounds).x
```

1. Истинный оптимальный портфель в модели Марковица с заданным отношением к риску.

Задана константа b. Решите задачу оптимизации

$$-E(x) + b\sigma(x) \to min,$$
$$x_1 + x_2 + \cdots + x_N = 1$$
$$x_i \geq 0$$

(т.е. найдите оптимальный портфель с отношением к риску, равным b). Найдите и зафиксируйте веса портфеля и значение целевой функции.

Здесь $E(x) = E_1 x_1 + E_2 x_2 + \cdots + E_N x_N$, $\sigma(x) = \sum\sum \sigma_{ij} x_i x_j$

Примечание 1.

Константа b подобрана таким образом, что истинный оптимальный CVaR портфель совпадает с истинным оптимальным портфелем п.1. Значение константы смотри в упражнениях к теме

$$b = \frac{1}{\sqrt{2\pi}}\frac{1}{1-\beta}exp(-\frac{(\Phi^{-1}(\beta))^2}{2})$$

где Φ - ф-ция стандартного нормального распределения, а β - уверенность для CVaR

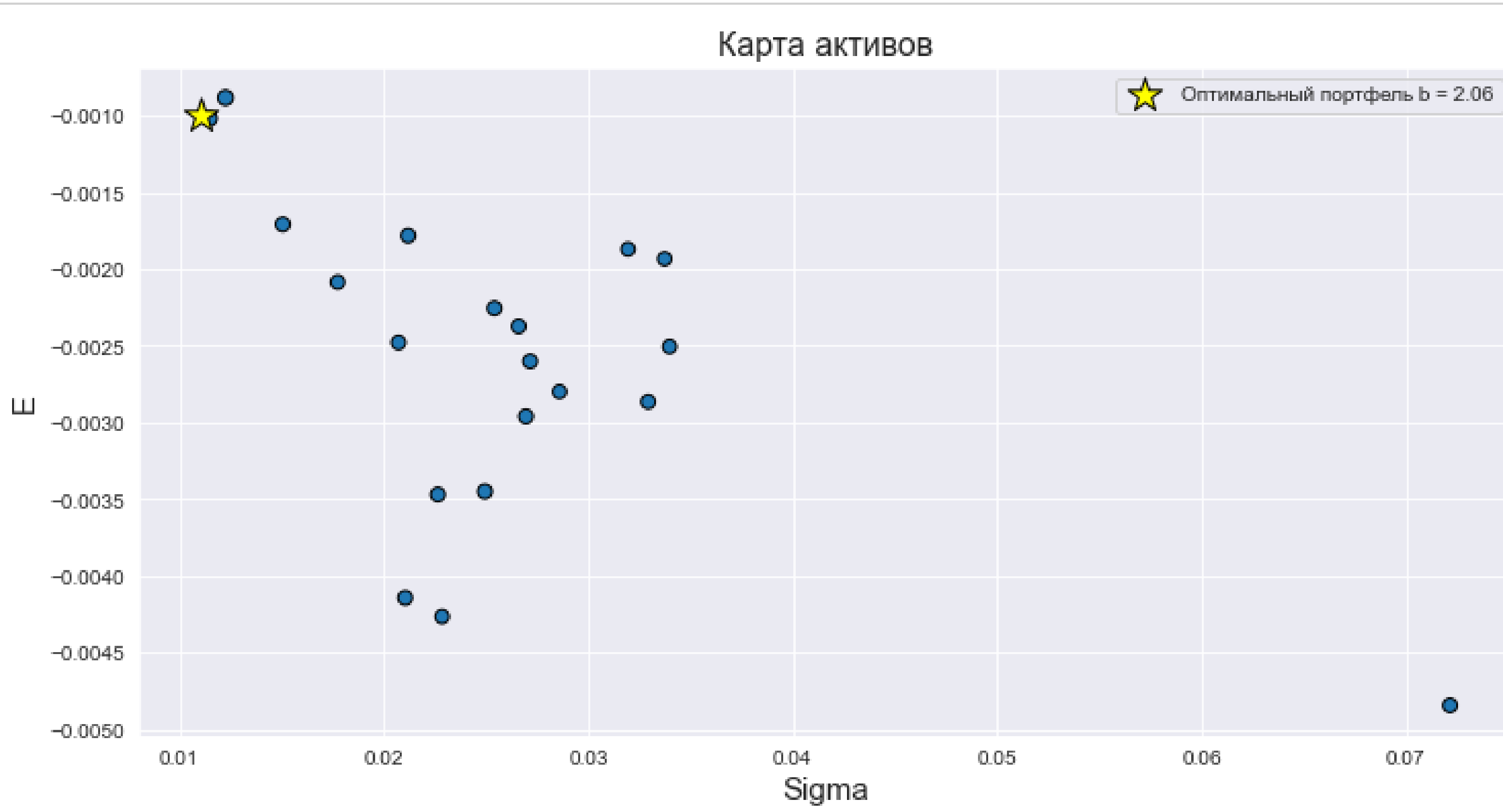
In [323]:
```python
from scipy.stats import norm

beta = 0.95

# ppf - percent point function for standart normal distribution
b = (np.sqrt(2 * np.pi) *(1 - beta) )**(-1) * np.exp(-(norm.ppf(beta)**2 / 2))
print('b: ', b)
```

```
b:  2.0627128075074257
```

In [324]:
```python
X = optimize_portfolio(mean_vec, cov_matrix, b, bounds=((0, 1),))
```

In [325]:
```python
def plot_weights_histogram(weights, data):
    plt.figure(figsize=(6,2))
    try:
        x_values = data['symbol'].values
    except:
        x_values = data['names'].values
    x = np.arange(len(weights))
    plt.xlabel('Акция')
    plt.ylabel('Вес')
    height = weights
    plt.bar(x, height=height)
    plt.xticks(x, x_values, rotation='50')
    plt.grid()

plot_weights_histogram(X, selected20)
plt.title('Полученный веса портфеля')
pass
```

In [326]:
```python
plot_mean_var_map(selected20, title='Карта активов')
plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)
plt.legend()
pass
```



2. Оценка неопределенности оптимального портфеля в модели Марковица с заданным отношением к риску.

2.1. Задайте число наблюдений T=30.

С помощью генератора многомерного нормального распределения создайте выборку размера T из нормального распределения с вектором математических ожиданий E=(E1, E2, …, EN) и матрицей ковариаций (σi,j).

In [327]:
```
T = 30
r_matrix_gen = np.random.multivariate_normal(mean_vec, cov_matrix, T)
r_matrix_gen.shape
```

Out[327]:
```
(30, 20)
```

2.2. По построенной выборке сделайте оценку E^{est} вектора математических ожиданий и оценку σ^{est} матрицы ковариаций.

In [328]:
```
mean_vec_est = np.mean(r_matrix_gen, axis=0)
cov_matrix_est = np.cov(r_matrix_gen.T)
mean_vec_est.shape, cov_matrix_est.shape
```

Out[328]:
```
((20,), (20, 20))
```

In [329]:
```
import pprint

print('Истинный вектор средних:')
pprint.pprint(np.around(mean_vec, 3))
print()
print('Оценки вектора средних:')
pprint.pprint(np.around(mean_vec_est, 3))
```

```
Истинный вектор средних:
array([-0.004, -0.004, -0.003, -0.003, -0.003, -0.005, -0.002, -0.003,
       -0.003, -0.003, -0.002, -0.002, -0.002, -0.002, -0.003, -0.002,
       -0.002, -0.002, -0.001, -0.001])

Оценки вектора средних:
array([-0.002, -0.004, -0.003,  0.001,  0.001,  0.008,  0.002, -0.   ,
       -0.005, -0.005,  0.001,  0.004, -0.001, -0.   ,  0.003, -0.002,
        0.   , -0.006, -0.   ,  0.   ])
```

2.3 Используя эти оценки решите задачу оптимизации

$$
\begin{gathered}
-E^{est}(x) + b\sigma^{est}(x) \to min, \\
x_1 + x_2 + \cdots + x_N = 1 \\
x_i \geq 0
\end{gathered}
$$

Здесь $E^{est}(x) = E_1^{est}x_1 + E_2^{est}x_2 + \cdots + E_N^{est}x_N$, $\sigma^{est}(x) = \sum\sum \sigma_{ij}^{est} x_i x_j$

(т.е. найдите выборочный оптимальный портфель с отношением к риску, равным b). Найдите и зафиксируйте веса портфеля и значение целевой функции.

In [330]:
```
X_est = optimize_portfolio(mean_vec_est, cov_matrix_est, b, bounds=((0, 1),))
```

2.4 Сравните два портфеля: истинный (п.1) и выборочный (п.2.3).

Оцените относительную ошибку в определении весов портфеля в норме Manhattan (L1 норма Минковского). Сделайте выводы. Сделайте сравнение в системе координат (σ, E).

In [331]:
```
print('Истинные веса портфеля:')
pprint.pprint(np.around(X, 3))
print()
print('Оценки весов портфеля:')
pprint.pprint(np.around(X_est, 3))
```

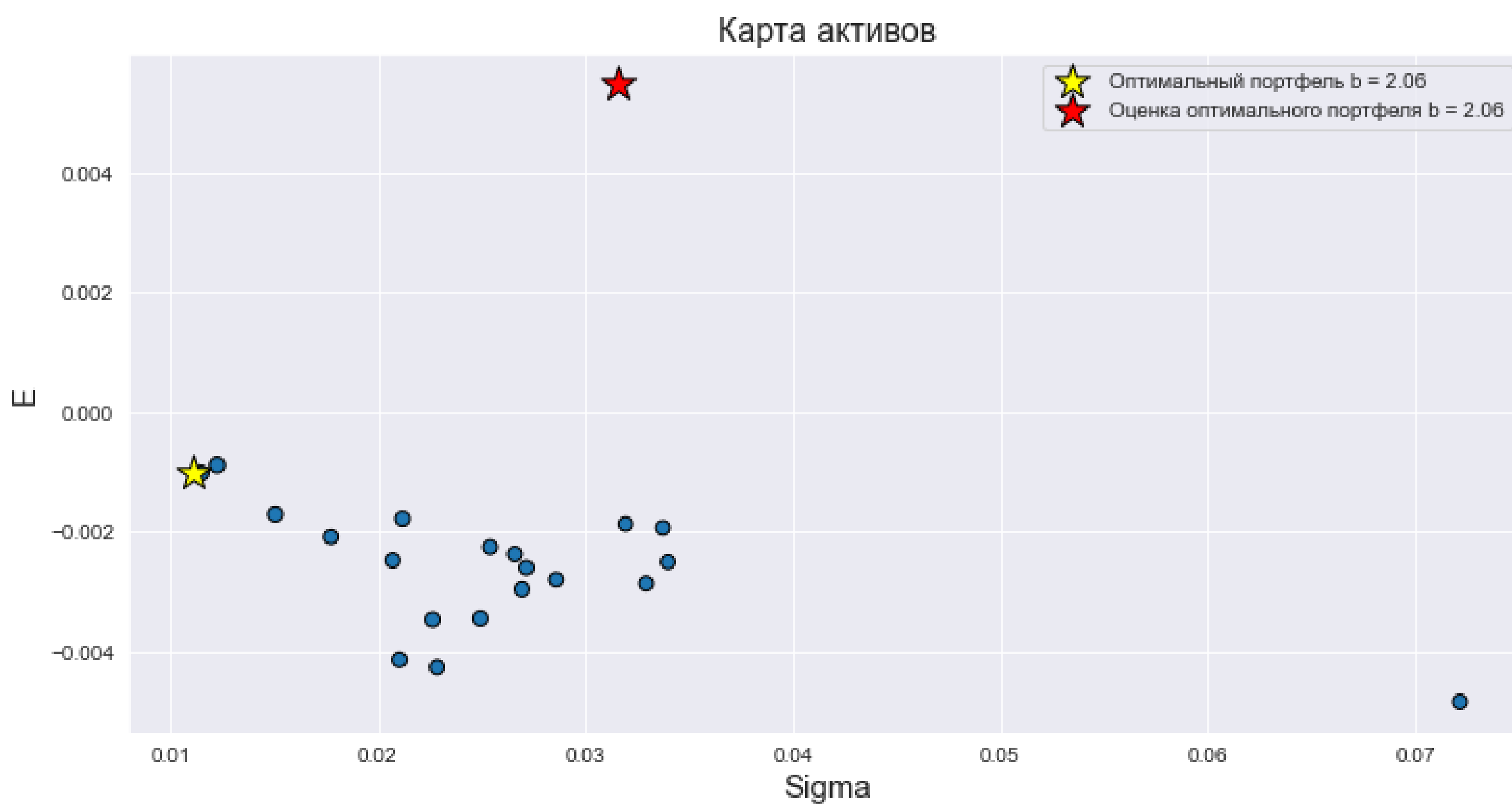
```
Истинные веса портфеля:
array([0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.067, 0.   , 0.   , 0.   , 0.   ,
       0.443, 0.49 ])

Оценки весов портфеля:
array([0.   , 0.   , 0.   , 0.   , 0.   , 0.287, 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.692, 0.   , 0.   , 0.022, 0.   , 0.   , 0.   ,
       0.   , 0.   ])
```

In [332]:
```
plot_weights_histogram(X, selected20)
plt.title('Истинные веса портфеля')
plt.show()
plot_weights_histogram(X_est, selected20)
plt.title('Оценки полученных весов портфеля')
pass
```

In [333]:
```
print('L1 норма вектора X - Xest:', np.around(np.linalg.norm(X - X_est, ord=1), 3))
```

```
L1 норма вектора X - Xest: 2.0
```

In [334]:
```
plot_mean_var_map(selected20, title='Карта активов')
plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)
plt.scatter(risk_portfolio(X_est, cov_matrix_est),
            np.dot(mean_vec_est, X_est),
            c='red',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оценка оптимального портфеля b = %.2f' % b)
plt.legend()
pass
```



Вывод:

Исходя из значений весов портфеля и их оценок, L1-нормы этих векторов, а также сравнения портфелей по карте E от σ можно сделать вывод, что разница существенна. Оценка оптимального портфеля с задданным параметром b имеет завышенную доходность и риск.

2.5. Повторите эксперимент S=40 раз и оцените среднюю относительную ошибку по S повторениям эксперимента.

Сделайте выводы.

Сделайте сравнение в системе координат (σ, E).

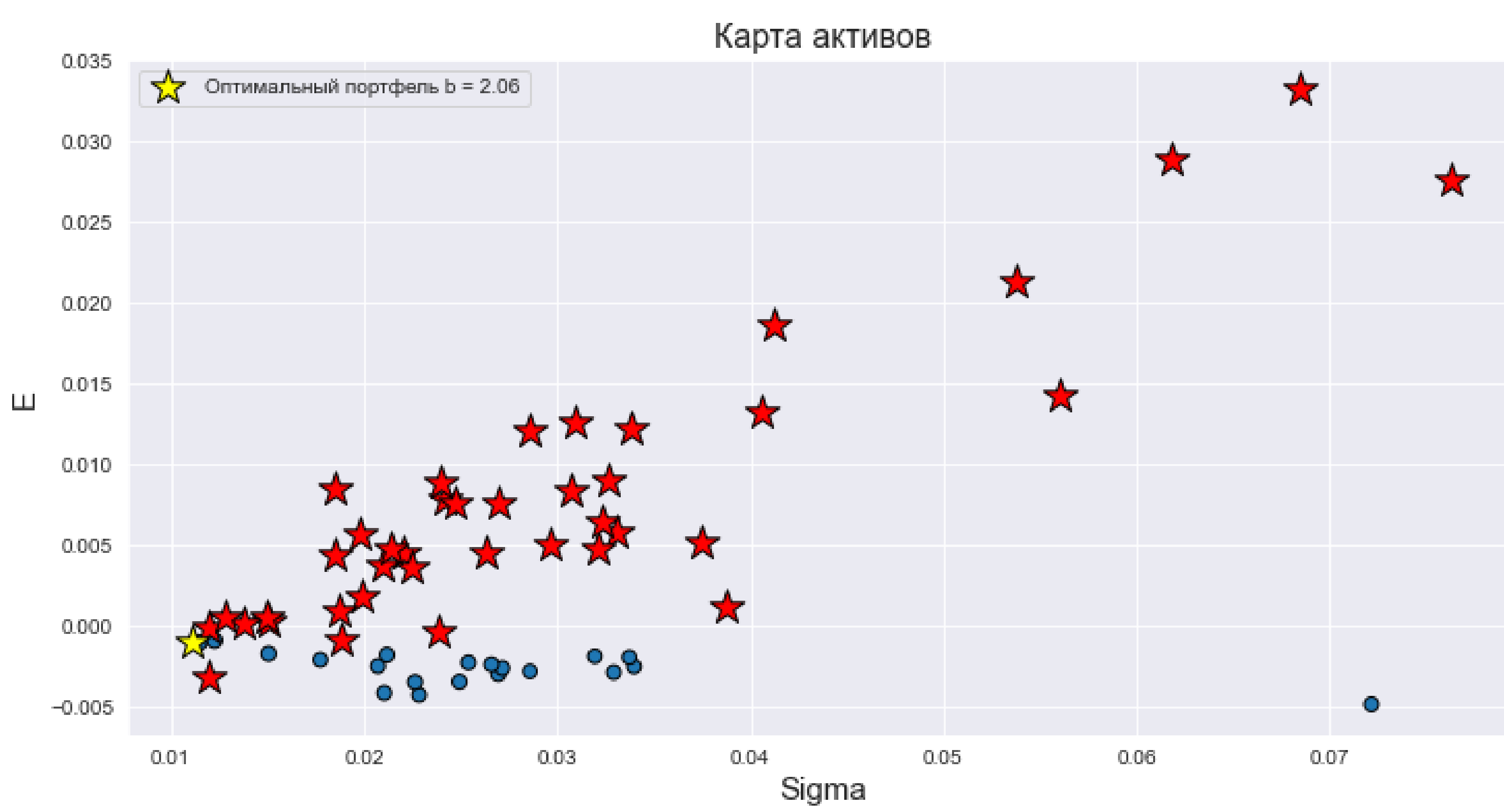
In [335]:
```
S = 40

experiments = []

for i in tqdm_notebook(range(S)):
    experiment = {}
    experiment['i'] = i
    experiment['r_matrix_gen'] = np.random.multivariate_normal(mean_vec, cov_matrix, T)
    experiment['mean_vec_est'] = np.mean(experiment['r_matrix_gen'], axis=0)
    experiment['cov_matrix_est'] = np.cov(experiment['r_matrix_gen'].T)
    experiment['X_est'] =  optimize_portfolio( experiment['mean_vec_est'],
                                              experiment['cov_matrix_est'],
                                              b,
                                              bounds=((0, 1),))
    experiment['L1-norm'] = np.linalg.norm(X - experiment['X_est'], ord=1)
    experiments.append(experiment)
```

100%   40/40 [00:01<00:00, 20.37it/s]

In [336]:
```
l1_norms = [experiment['L1-norm'] for experiment in experiments]
l1_norms_mean  = np.mean(l1_norms)
l1_norms_std  = np.std(l1_norms)
print('Средняя L1-норма по %d экспериментам: %.3f' % (S, l1_norms_mean))
print('Примерный 95 прц. доверительный интервал: [%.3f, %.3f]' %
      (l1_norms_mean - 2*l1_norms_std, l1_norms_mean + 2*l1_norms_std))
```

```
Средняя L1-норма по 40 экспериментам: 1.893
Примерный 95 прц. доверительный интервал: [1.336, 2.450]
```

In [337]:
```
plot_mean_var_map(selected20, title='Карта активов')

for experiment in experiments:
    plt.scatter(risk_portfolio(experiment['X_est'], experiment['cov_matrix_est']),
                np.dot(experiment['mean_vec_est'], experiment['X_est']),
                c='red',
                marker='*',
                s=300,
                edgecolors='black')

plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)
plt.legend()
pass
```



Вывод:

Исходя из 40 проведенных экспериментов можно видеть, что в среднем оценки весов существенно далеки от истинных, хоть некоторые портфели, построенные по данным оценкам, относительно близки к истинному на карте активов, в целом можно отметить сильную зависимость от сгенерированных наблюдений.

2.6 Предположите, что нам известны точные значения математических ожиданий E=(E1, E2, ..., EN).

Повторите пп. 2.2-2.5. используя оценку только матрицы ковариаций (т.е. решайте задачу оптимизации

$$-E^{est}(x) + b\sigma^{est}(x) \to min,$$
$$x_1 + x_2 + \dots + x_N = 1$$
$$x_i \geq 0$$

Здесь $E^{est}(x) = E_1^{est} x_1 + E_2^{est} x_2 + \dots + E_N^{est} x_N$, $\sigma^{est}(x) = \sum \sum \sigma_{ij}^{est} x_i x_j$

Сравните точность этих портфелей и портфелей п.2.3

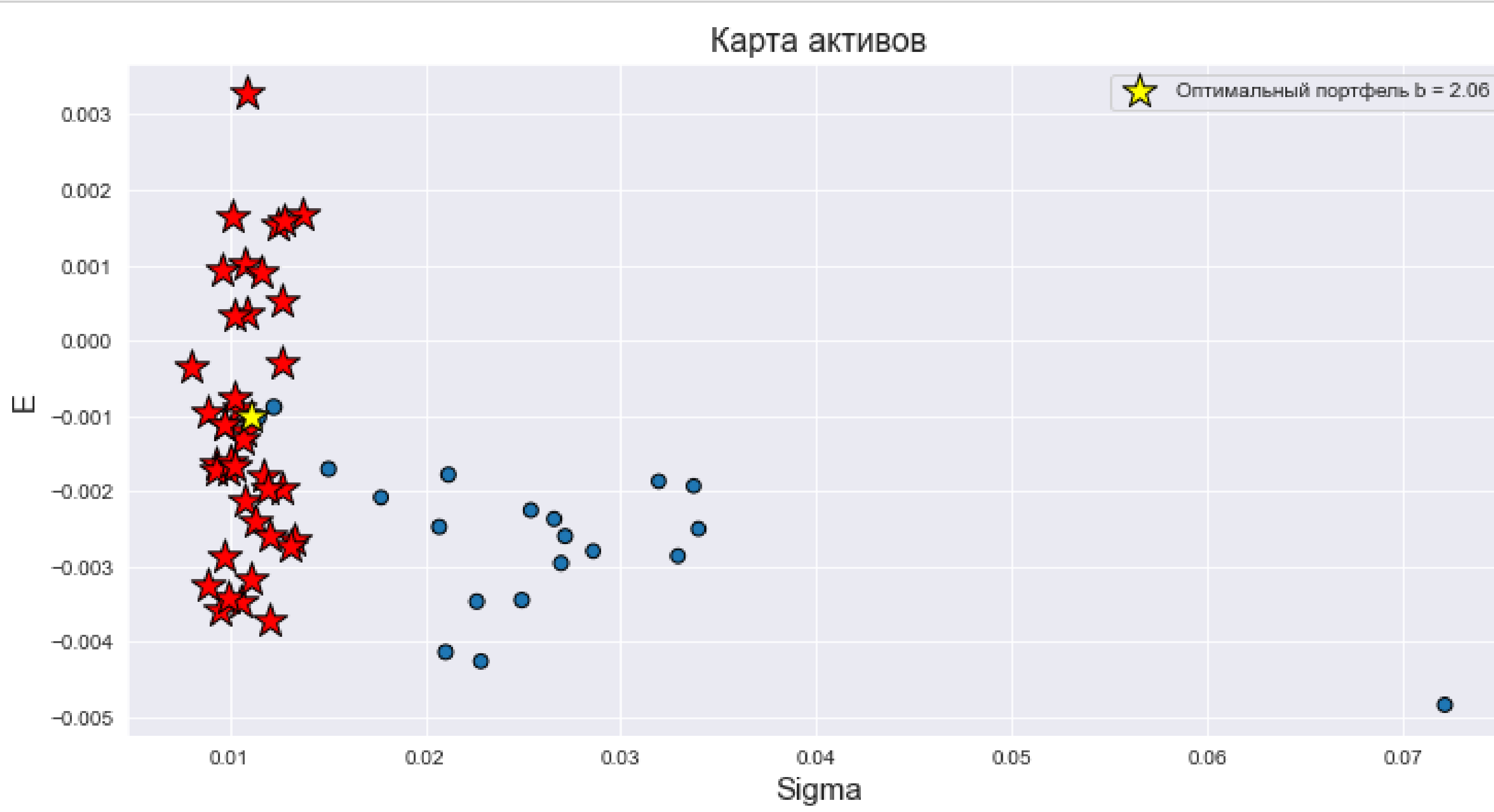
In [338]:
```
experiments_with_true_mean = []

for i in tqdm_notebook(range(S)):
    experiment = {}
    experiment['i'] = i
    experiment['r_matrix_gen'] = np.random.multivariate_normal(mean_vec, cov_matrix, T)
    experiment['mean_vec_est'] = np.mean(experiment['r_matrix_gen'], axis=0)
    experiment['cov_matrix_est'] = np.cov(experiment['r_matrix_gen'].T)
    experiment['X_est'] =  optimize_portfolio(mean_vec,                    # используем истинный
                                              experiment['cov_matrix_est'],
                                              b,
                                              bounds=((0, 1),))
    experiment['L1-norm'] = np.linalg.norm(X - experiment['X_est'], ord=1)
    experiments_with_true_mean.append(experiment)
```

100% 40/40 [00:02<00:00, 18.73it/s]

In [339]:
```
l1_norms = [experiment['L1-norm'] for experiment in experiments_with_true_mean]
l1_norms_mean  = np.mean(l1_norms)
l1_norms_std  = np.std(l1_norms)
print('Средняя L1-норма по %d экспериментам: %.3f' % (S, l1_norms_mean))
print('Примерный 95 прц. доверительный интервал: [%.3f, %.3f]' %
      (l1_norms_mean - 2*l1_norms_std, l1_norms_mean + 2*l1_norms_std))
```

```
Средняя L1-норма по 40 экспериментам: 0.109
Примерный 95 прц. доверительный интервал: [-0.002, 0.220]
```

In [340]:
```
plot_mean_var_map(selected20, title='Карта активов')

for experiment in experiments_with_true_mean:
    plt.scatter(risk_portfolio(experiment['X_est'], experiment['cov_matrix_est']),
                np.dot(experiment['mean_vec_est'], experiment['X_est']),
                c='red',
                marker='*',
                s=300,
                edgecolors='black')
plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)
plt.legend()
pass
```



Вывод:

Исходя из 40 проведенных экспериментов с истинным вектором средних можно видеть, что в оценки весов существенно улучшились (судя по L1-норме примерно в 10 раз) и стали ближе к истинным, такой же вывод можно сделать и по по карте E от σ, где сгенерированные портфели стали располагаться существенно ближе к истинному портфелю не только по доходности, но и по риску.

3. Оценка неопределенности оптимального CVaR портфеля

3.1 Уровень значимости β выбран 0,95. Число наблюдений T.

Используя сгенерированные наблюдения из п.2.1 решите задачу ЛП для определения оптимального CVaRβ портфеля. Найдите и зафиксируйте веса портфеля и значение целевой функции CVaR.

In [341]:
```
def cvar_objective_function(UXalpha, T, beta):
    return UXalpha[-1] + 1 / (T * (1 - beta)) * np.sum(UXalpha[:T])

def cvar_optimize_portfolio(r_matrix,
                            beta,
                            cvar_objective_function=cvar_objective_function,
                            cvxopt=False):
    alpha  = 0
    N = r_matrix.shape[1]
    X = np.ones(N)/ N

    T = r_matrix.shape[0]
    U = np.dot(r_matrix,  X) - alpha

    UXalpha = np.zeros(T+N+1)
    UXalpha[:T] = U
    UXalpha[T:N+T]= X
    UXalpha[-1] = alpha

    bounds_U = ((0, 100000000000),) * T
    bounds_X = ((0, 1.1),) * N
    bounds_alpha = ((-100000, 100000),)
    bounds = bounds_U + bounds_X + bounds_alpha


    constraints = []
    constraints.append({'type': 'eq', 'fun': lambda X: sum(X[T:N+T]) -1})
    def u_x_con(UXalpha, r_matrix, i):
        return np.dot(r_matrix[i], UXalpha[T:N+T]) + UXalpha[-1] - UXalpha[i],
    for i in range(T):
        constraints.append({'type': 'ineq',
                            'fun': u_x_con,
                            'args': (r_matrix, i)})


    return minimize(cvar_objective_function, UXalpha,
                    args=(T, beta), method='SLSQP',
                    constraints=constraints,
                    bounds=bounds).x
```

In [342]:
```
%%time
result = cvar_optimize_portfolio(r_matrix_gen, beta)
```

```
Wall time: 504 ms
```

In [343]:
```
T = 30
N = 20
UXalpha = result
cvar_X_est = UXalpha[T:N+T]
alpha_est  = UXalpha[-1]
```

In [356]:
```
print('Alpha CVAR портфеля:' ,round(alpha_est, 4))
```

```
Alpha CVAR портфеля: 0.0121
```

In [355]:
```
print('VAR CVAR портфеля на уровне 0.95:' ,round(np.quantile(-np.dot(r_matrix_gen, cvar_X_est), 0
```

```
VAR CVAR портфеля на уровне 0.95: 0.0121
```

In [346]:
```
# np.quantile(-np.dot(r_matrix_gen, X_est), 0.95)
```

In [347]:
```
plot_weights_histogram(X, selected20)
plt.title('Истинные веса портфеля')
plt.show()
plot_weights_histogram(X_est, selected20)
plt.title('Оценки полученных весов портфеля')
plot_weights_histogram(cvar_X_est, selected20)
plt.title('Оценки полученных весов портфеля CVAR')
pass
```

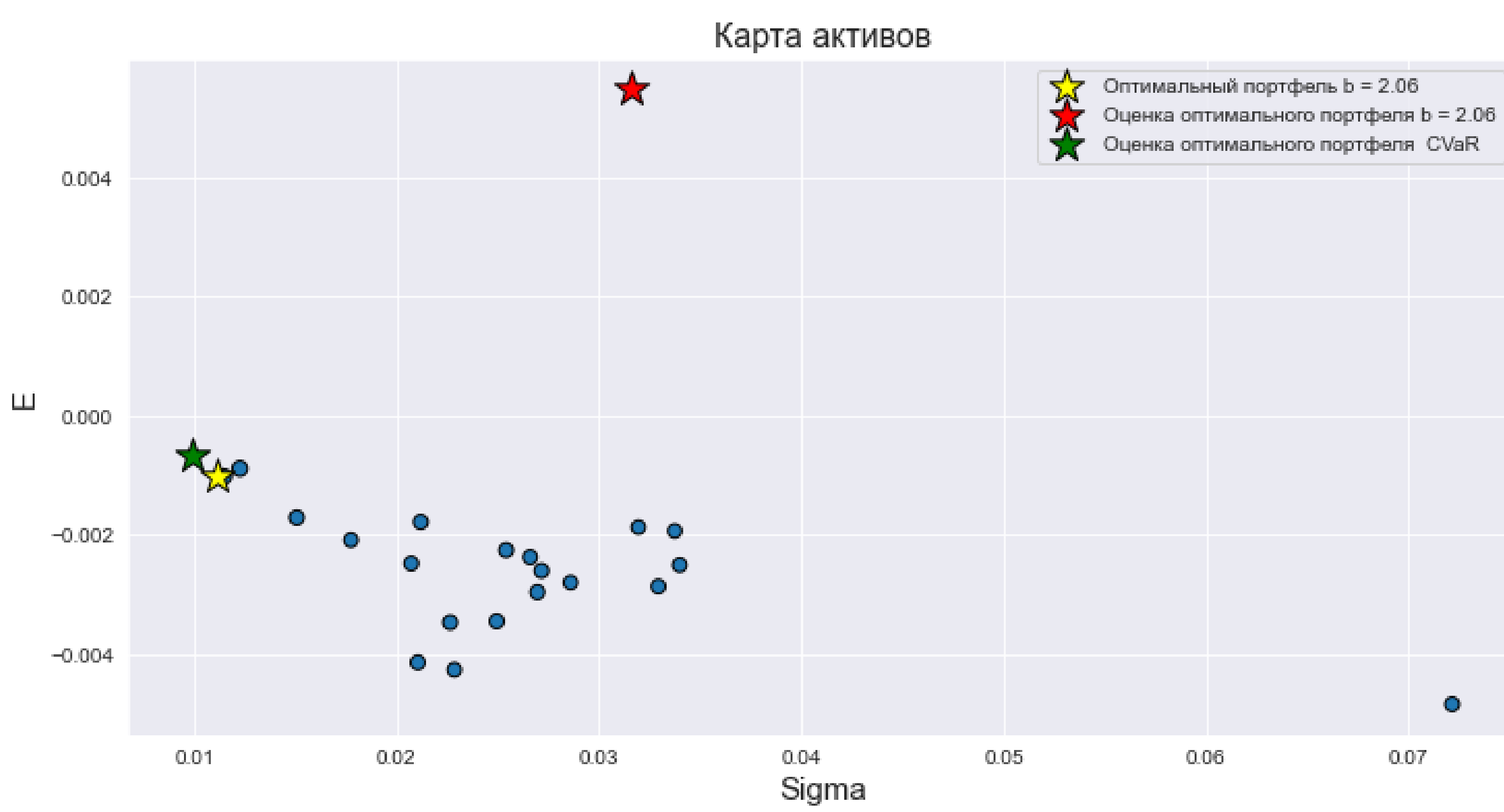
3.2 Сравните два портфеля: истинный (п.1) и найденный в п.3.1.

Оцените относительную ошибку в определении весов портфеля в норме Manhattan (L1 норма Минковского). Сравните с ошибкой портфеля из п. 2.3

In [358]:
```
print('L1 норма вектора X-Xest в 2.3 :', np.around(np.linalg.norm(X - X_est, ord=1), 3))
print('L1 норма вектора для CVAR X-cvar_Xest:', np.around(np.linalg.norm(X - cvar_X_est, ord=1), 
```

```
L1 норма вектора X-Xest в 2.3 : 2.0
L1 норма вектора для CVAR X-cvar_Xest: 1.081
```

In [350]:
```
plot_mean_var_map(selected20, title='Карта активов')
plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)

plt.scatter(risk_portfolio(X_est, cov_matrix_est),
            np.dot(mean_vec_est, X_est),
            c='red',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оценка оптимального портфеля b = %.2f' % b)

plt.scatter(risk_portfolio(cvar_X_est, cov_matrix_est),
            np.dot(mean_vec_est, cvar_X_est),
            c='green',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оценка оптимального портфеля  CVaR' % b)
plt.legend()
pass
```



Вывод:

Исходя из полученной L1 нормы для CVAR портфеля, можно сделать вывод, что он намного ближе к истинному (в 2 раза по L1), так же это видно на карте активов

3.3. Повторите эксперимент S раз и оцените среднюю относительную ошибку по S повторениям эксперимента.

Сделайте выводы. Сравните с ошибкой из п. 2.5

In [351]:
```
S = 40

cvar_experiments = []

for i in tqdm_notebook(range(S)):
    cvar_experiment = {}
    cvar_experiment['i'] = i
    cvar_experiment['r_matrix_gen'] = np.random.multivariate_normal(mean_vec, cov_matrix, T)
    cvar_experiment['mean_vec_est'] = np.mean(cvar_experiment['r_matrix_gen'], axis=0)
    cvar_experiment['cov_matrix_est'] = np.cov(cvar_experiment['r_matrix_gen'].T)
    cvar_experiment['cvar_X_est'] =  cvar_optimize_portfolio(cvar_experiment['r_matrix_gen'], beta
    cvar_experiment['L1-norm'] = np.linalg.norm(X - cvar_experiment['cvar_X_est'][T:N+T], ord=1)
    cvar_experiments.append(cvar_experiment)
```

100% 40/40 [00:15<00:00, 2.56it/s]

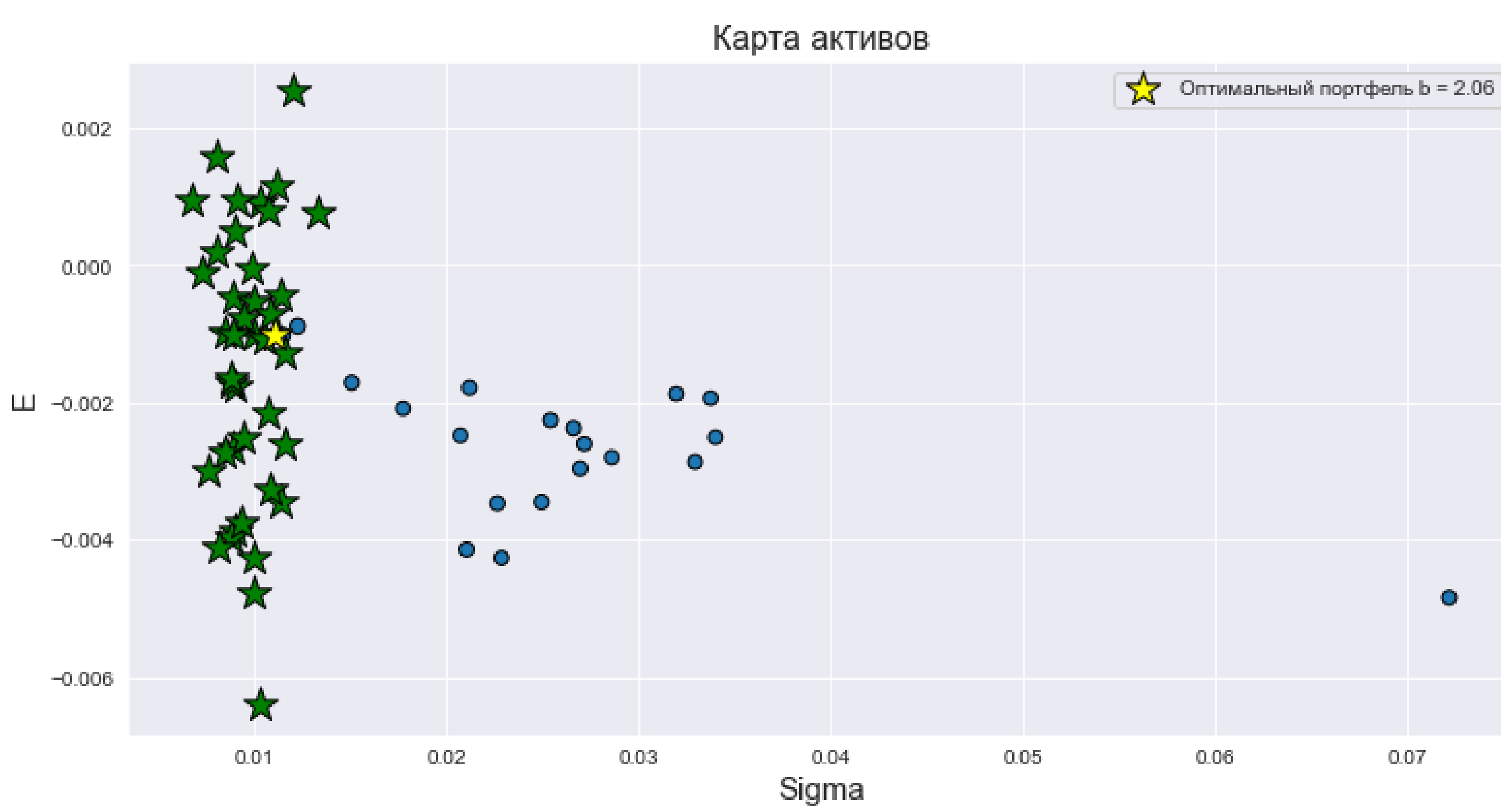
In [360]:
```
l1_norms = [experiment['L1-norm'] for experiment in experiments]
l1_norms_mean  = np.mean(l1_norms)
l1_norms_std  = np.std(l1_norms)
print('Средняя L1-норма из пункта 2.5 по %d экспериментам: %.3f' % (S, l1_norms_mean))
print('Примерный 95 прц. доверительный интервал: [%.3f, %.3f]' %
      (l1_norms_mean - 2*l1_norms_std, l1_norms_mean + 2*l1_norms_std))
```

```
Средняя L1-норма из пункта 2.5 по 40 экспериментам: 1.893
Примерный 95 прц. доверительный интервал: [1.336, 2.450]
```

In [352]:
```
l1_norms = [cvar_experiment['L1-norm'] for cvar_experiment in cvar_experiments]
l1_norms_mean  = np.mean(l1_norms)
l1_norms_std  = np.std(l1_norms)
print('Средняя L1-норма CVAR по %d экспериментам: %.3f' % (S, l1_norms_mean))
print('Примерный 95 прц. доверительный интервал: [%.3f, %.3f]' %
      (l1_norms_mean - 2*l1_norms_std, l1_norms_mean + 2*l1_norms_std))
```

```
Средняя L1-норма CVAR по 40 экспериментам: 1.286
Примерный 95 прц. доверительный интервал: [0.557, 2.015]
```

In [353]:
```
plot_mean_var_map(selected20, title='Карта активов')

for experiment in cvar_experiments:
    plt.scatter(risk_portfolio(experiment['cvar_X_est'][T:N+T], experiment['cov_matrix_est']),
                np.dot(experiment['mean_vec_est'], experiment['cvar_X_est'][T:N+T]),
                c='green',
                marker='*',
                s=300,
                edgecolors='black')
plt.scatter(risk_portfolio(X, cov_matrix),
            np.dot(mean_vec,X),
            c='yellow',
            marker='*',
            s=300,
            edgecolors='black',
            label='Оптимальный портфель b = %.2f' % b)
plt.legend()
pass
```



Вывод:

Исходя из 40 проведенных экспериментов для CVAR портфелей можно видеть, что оценки оценка истинного портфеля лучше, чем в пункте 2.5 (по L1 норме 1.28 против 1.89)