

Лосева Елизавета Юрьевна, группа 8-2

Лабораторная работа № 8

Вариант № 16

## Линейная регрессия

### Цель работы

Синтезировать заданный алгоритм линейной регрессии. Выполнить проверку значимости полученной модели регрессии.

### Задание

Используя метод наименьших квадратов реализовать алгоритм гребневой регрессии для трехмерных векторов входной переменной и параметра степени обусловленности матрицы ковариации  $= 1e-9$ . Вычислить значение параметра регуляризации, при котором коэффициент обусловленности матрицы  $XtX$  был равен 100. Вычислить значения невязки (на обучающей и тестовой выборке) и СКО оценивания коэффициентов модели при следующих объемах выборок: 50, 200, 1000.

### Код программы

```
import matplotlib
import numpy as np
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

cond_param = 1e-9 # параметр степени обусловленности матрицы ковариации
cond_target = 100 # целевой коэффициент обусловленности матрицы XtX
sample_sizes = [50, 200, 1000] # объемы выборок
n_features = 3 # трехмерные векторы входной переменной

def generate_conditioned_data(n_samples, n_features, cond_param,
random_state=42):
    """
    Генерация данных с заданной степенью обусловленности матрицы признаков.
    """
    rng = np.random.default_rng(random_state)

    # Максимальное сингулярное число
    sigma_max = 1.0
```

```

# Минимальное сингулярное число из параметра обусловленности
sigma_min = cond_param

# Создаем диагональную матрицу сингулярных значений
# Линейно убывающие значения от sigma_max до sigma_min
sigma_s = np.linspace(sigma_max, sigma_min, n_features)
S = np.diag(sigma_s)

# Генерируем случайные ортогональные матрицы U и V
# Используем QR-разложение для получения ортогональных матриц
U, _ = np.linalg.qr(rng.standard_normal((n_samples, n_features)))
V, _ = np.linalg.qr(rng.standard_normal((n_features, n_features)))

# Формируем матрицу признаков:  $X = U * S * V^T$ 
X = U @ S @ V.T

return X

def find_alpha_for_condition(X, cond_target, alpha_range=(1e-10, 1e3)):
    """
    Подбор параметра регуляризации alpha для достижения целевой обусловленности.
    Используем бинарный поиск.
    """
    XTX = X.T @ X
    n_features = XTX.shape[0]

    alpha_min, alpha_max = alpha_range

    for _ in range(50): # Максимум 50 итераций
        alpha = (alpha_min + alpha_max) / 2

        # Вычисление обусловленности матрицы  $X^T X + \alpha * I$ 
        matrix = XTX + alpha * np.eye(n_features)
        cond_actual = np.linalg.cond(matrix)

        # Корректировка границ поиска
        if cond_actual > cond_target:
            alpha_min = alpha # Нужно увеличить alpha
        else:
            alpha_max = alpha # Нужно уменьшить alpha

        # Критерий остановки
        if abs(cond_actual - cond_target) / cond_target < 0.01:
            break

    return alpha, cond_actual

def ridge_regression_manual(X, y, alpha):
    """

```

```

Реализация гребневой регрессии вручную через нормальные уравнения.
"""
n_features = X.shape[1]

# Решение нормальных уравнений с регуляризацией
#  $(X^T X + \alpha * I) * w = X^T y$ 
XTX = X.T @ X
XTy = X.T @ y

# Добавление регуляризации
reg_matrix = alpha * np.eye(n_features)

# Решение системы уравнений
w = np.linalg.solve(XTX + reg_matrix, XTy)

return w

def run_experiment(N, random_state=42):
    """
    Строим гребневую регрессию для заданного объема выборки N.
    Рисуем график и печатаем характеристики.
    """
    rng = np.random.default_rng(random_state)

    print(f"\n{'=' * 60}")
    print(f"ЭКСПЕРИМЕНТ: N = {N}")
    print(f"{'=' * 60}")

    # 1) Генерация данных
    n_train = int(N * 0.7) # 70% на обучение
    n_test = N - n_train # 30% на тест

    # Генерация обучающей выборки
    X_train = generate_conditioned_data(n_train, n_features, cond_param,
random_state)

    # Добавление свободного члена (столбец из единиц)
    X_train = np.column_stack([np.ones(n_train), X_train])

    # Генерация истинных коэффициентов (включая свободный член)
    w_true = rng.standard_normal(n_features + 1)

    # Генерация целевой переменной с шумом
    noise_std = 0.1
    y_train = X_train @ w_true + noise_std * rng.standard_normal(n_train)

    # Генерация тестовой выборки
    X_test = generate_conditioned_data(n_test, n_features, cond_param,
random_state + 1)

```

```

X_test = np.column_stack([np.ones(n_test), X_test])
y_test = X_test @ w_true + noise_std * rng.standard_normal(n_test)

# 2) Подбор параметра регуляризации для целевой обусловленности
alpha_opt, cond_actual = find_alpha_for_condition(X_train, cond_target)

print(f"Параметр регуляризации  $\alpha$  = {alpha_opt:.6e}")
print(f"Целевая обусловленность: {cond_target}")
print(f"Фактическая обусловленность: {cond_actual:.2f}")
print(f"Отклонение: {abs(cond_actual - cond_target) / cond_target * 100:.2f}%")

# 3) Обучение гребневой регрессии
# Способ 1: Вручную (по заданию)
w_manual = ridge_regression_manual(X_train, y_train, alpha_opt)

# Способ 2: Через sklearn (для сравнения и проверки)
ridge_sklearn = Ridge(alpha=alpha_opt, fit_intercept=False)
ridge_sklearn.fit(X_train, y_train)
w_sklearn = ridge_sklearn.coef_

# 4) Предсказания
y_train_pred_manual = X_train @ w_manual
y_test_pred_manual = X_test @ w_manual

# 5) Вычисление метрик
# Невязки (среднеквадратичные ошибки)
train_mse = mean_squared_error(y_train, y_train_pred_manual)
test_mse = mean_squared_error(y_test, y_test_pred_manual)

# Коэффициент детерминации R^2
train_r2 = r2_score(y_train, y_train_pred_manual)
test_r2 = r2_score(y_test, y_test_pred_manual)

# СКО оценивания коэффициентов
coeff_error = np.sqrt(np.mean((w_manual - w_true) ** 2))

# 6) Печать результатов
print("\nРЕЗУЛЬТАТЫ:")
print(f"Невязка на обучающей выборке (MSE): {train_mse:.6f}")
print(f"Невязка на тестовой выборке (MSE): {test_mse:.6f}")
print(f"Коэффициент детерминации R^2 (train): {train_r2:.6f}")
print(f"Коэффициент детерминации R^2 (test): {test_r2:.6f}")
print(f"СКО оценивания коэффициентов: {coeff_error:.6f}")

print("\nКоэффициенты модели:")
print(f"Истинные: {w_true.round(4)}")
print(f"Оцененные (manual): {w_manual.round(4)}")
print(f"Оцененные (sklearn): {w_sklearn.round(4)}")

```

```

# 7) Визуализация (для 2D проекции)
if n_features >= 2:
    # Берем первые два признака (без свободного члена) для визуализации
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

    # График 1: Исходные данные и предсказания по первому признаку
    ax1 = axes[0]

    # Сортируем по первому признаку для красивого графика
    idx_sorted = np.argsort(X_train[:, 1]) # X_train[:, 0] - это единицы

    ax1.scatter(X_train[:, 1], y_train, alpha=0.6, label='Обучающая выборка',
color='blue')
    ax1.scatter(X_test[:, 1], y_test, alpha=0.6, label='Тестовая выборка',
color='red')

    # Линия регрессии
    x_line = np.linspace(X_train[:, 1].min(), X_train[:, 1].max(), 100)
    # Для построения линии используем средние значения других признаков
    X_line = np.zeros((100, n_features + 1))
    X_line[:, 0] = 1 # свободный член
    X_line[:, 1] = x_line
    # Средние значения для остальных признаков
    for i in range(2, n_features + 1):
        X_line[:, i] = np.mean(X_train[:, i])

    y_line = X_line @ w_manual
    ax1.plot(x_line, y_line, 'k-', linewidth=2, label='Гребневая регрессия')

    ax1.set_xlabel('X1 (первый признак)')
    ax1.set_ylabel('Y')
    ax1.set_title(f'Гребневая регрессия (N={N},  $\alpha$ ={alpha_opt:.2e})')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # График 2: Сравнение истинных и предсказанных значений
    ax2 = axes[1]

    # Все предсказания
    all_X = np.vstack([X_train, X_test])
    all_y = np.concatenate([y_train, y_test])
    all_y_pred = np.concatenate([y_train_pred_manual, y_test_pred_manual])

    ax2.scatter(all_y, all_y_pred, alpha=0.6)

    # Линия идеального предсказания
    min_val = min(all_y.min(), all_y_pred.min())
    max_val = max(all_y.max(), all_y_pred.max())

```

```

        ax2.plot([min_val, max_val], [min_val, max_val], 'r-', label='Идеальное
предсказание')

        ax2.set_xlabel('Истинные значения Y')
        ax2.set_ylabel('Предсказанные значения Y')
        ax2.set_title('Предсказания vs Истинные значения')
        ax2.legend()
        ax2.grid(True, alpha=0.3)

        # Подпись с параметрами
        txt = f"N={N}  α={alpha_opt:.2e}  cond={cond_actual:.0f}\n"
        txt += f"R²(train)={train_r2:.3f}  R²(test)={test_r2:.3f}\n"
        txt += f"MSE(train)={train_mse:.3f}  MSE(test)={test_mse:.3f}"
        plt.figtext(0.02, 0.02, txt, fontsize=9, bbox=dict(facecolor='white',
alpha=0.8, edgecolor='gray'))
        plt.tight_layout()
        plt.show()

    return {
        'N': N,
        'alpha': alpha_opt,
        'cond_actual': cond_actual,
        'train_mse': train_mse,
        'test_mse': test_mse,
        'train_r2': train_r2,
        'test_r2': test_r2,
        'coeff_error': coeff_error,
        'w_true': w_true,
        'w_estimated': w_manual
    }

# ЗАПУСК ДЛЯ РАЗНЫХ ОБЪЁМОВ ВЫБОРКИ
results = []
for N in sample_sizes:
    res = run_experiment(N)
    results.append(res)

# Сводный график сравнения метрик для разных N
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# График 1: Невязки на обучающей и тестовой выборках
ax1 = axes[0, 0]
train_msес = [r['train_mse'] for r in results]
test_msес = [r['test_mse'] for r in results]

ax1.plot(sample_sizes, train_msес, 'b-o', linewidth=2, markersize=8,
label='Обучающая')
ax1.plot(sample_sizes, test_msес, 'r-s', linewidth=2, markersize=8,
label='Тестовая')

```

```

ax1.set_xlabel('Объем выборки N')
ax1.set_ylabel('Среднеквадратичная ошибка (MSE)')
ax1.set_title('Невязка модели')
ax1.legend()
ax1.grid(True, alpha=0.3)
ax1.set_xscale('log')

# График 2: Коэффициент детерминации R^2
ax2 = axes[0, 1]
train_r2s = [r['train_r2'] for r in results]
test_r2s = [r['test_r2'] for r in results]

ax2.plot(sample_sizes, train_r2s, 'b-o', linewidth=2, markersize=8,
label='Обучающая')
ax2.plot(sample_sizes, test_r2s, 'r-s', linewidth=2, markersize=8,
label='Тестовая')
ax2.set_xlabel('Объем выборки N')
ax2.set_ylabel('Коэффициент детерминации R^2')
ax2.set_title('Качество модели (R^2)')
ax2.legend()
ax2.grid(True, alpha=0.3)
ax2.set_xscale('log')

# График 3: Параметр регуляризации α
ax3 = axes[1, 0]
alphas = [r['alpha'] for r in results]

ax3.semilogy(sample_sizes, alphas, 'g^-', linewidth=2, markersize=10)
ax3.set_xlabel('Объем выборки N')
ax3.set_ylabel('Параметр регуляризации α')
ax3.set_title('Зависимость α от объема выборки')
ax3.grid(True, alpha=0.3)
ax3.set_xscale('log')

# График 4: СКО оценивания коэффициентов
ax4 = axes[1, 1]
coeff_errors = [r['coeff_error'] for r in results]

ax4.plot(sample_sizes, coeff_errors, 'm-*', linewidth=2, markersize=10)
ax4.set_xlabel('Объем выборки N')
ax4.set_ylabel('СКО коэффициентов')
ax4.set_title('Точность оценивания коэффициентов')
ax4.grid(True, alpha=0.3)
ax4.set_xscale('log')

plt.suptitle(f'Гребневая регрессия (Вариант 16): cond_param={cond_param},
cond_target={cond_target}',
            fontsize=14, fontweight='bold')
plt.tight_layout()

```

```
plt.show()

# Вывод сводной таблицы результатов
print("\n" + "=" * 80)
print("СВОДНАЯ ТАБЛИЦА РЕЗУЛЬТАТОВ (Вариант 16)")
print("=" * 80)
print(f"{'N':<8} {'α':<15} {'cond':<10} {'MSE(train)':<12} {'MSE(test)':<12} "
      f"{'R²(train)':<10} {'R²(test)':<10} {'СКО коэф.':<10}")
print("=" * 80)

for i, N in enumerate(sample_sizes):
    r = results[i]
    print(f"{'N':<8} {r['alpha']:<15.2e} {r['cond_actual']:<10.0f} "
          f"{r['train_mse']:<12.6f} {r['test_mse']:<12.6f} "
          f"{r['train_r2']:<10.4f} {r['test_r2']:<10.4f} "
          f"{r['coeff_error']:<10.6f}")
print("=" * 80)
```

## Используемые формулы

1. Основная формула гребневой регрессии

$$\hat{w} = (X^T X + \alpha I)^{-1} X^T y$$

2. Коэффициент обусловленности матрицы

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

3. Метрики качества модели

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

## Результаты выполнения задания

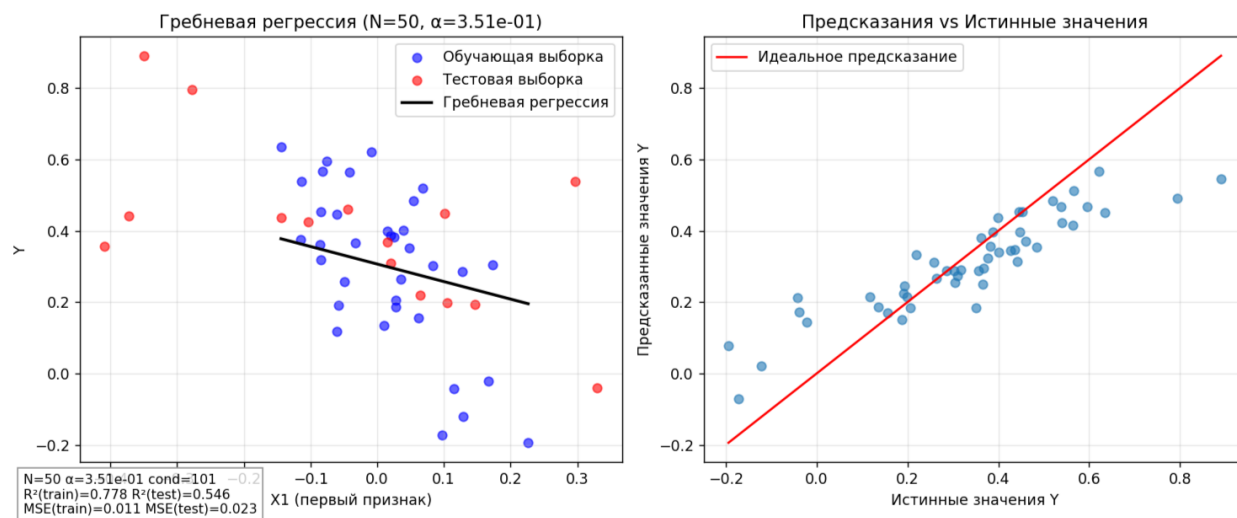


Рисунок 1

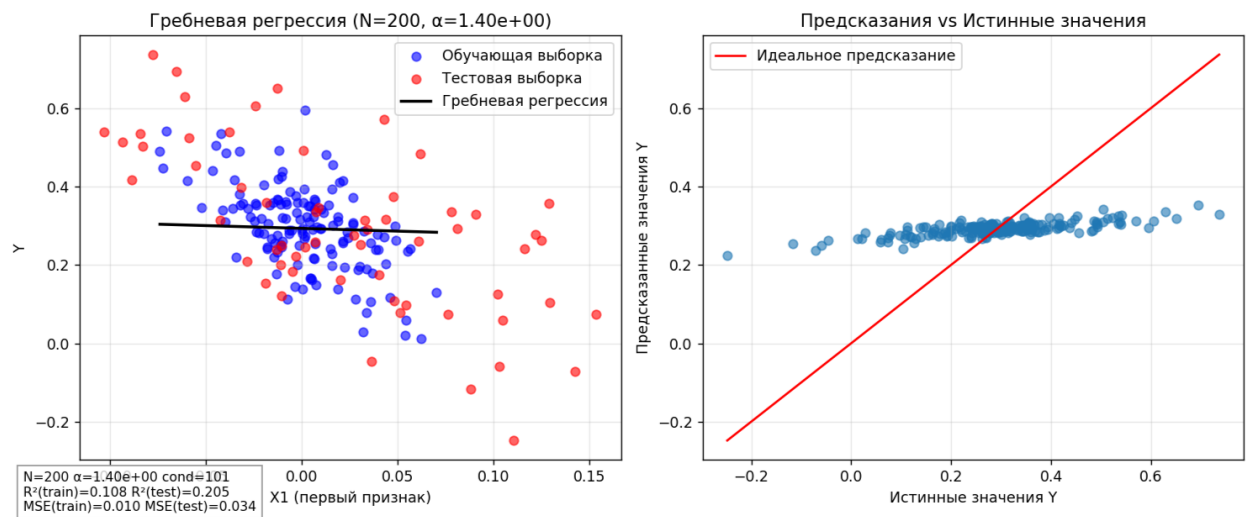


Рисунок 2

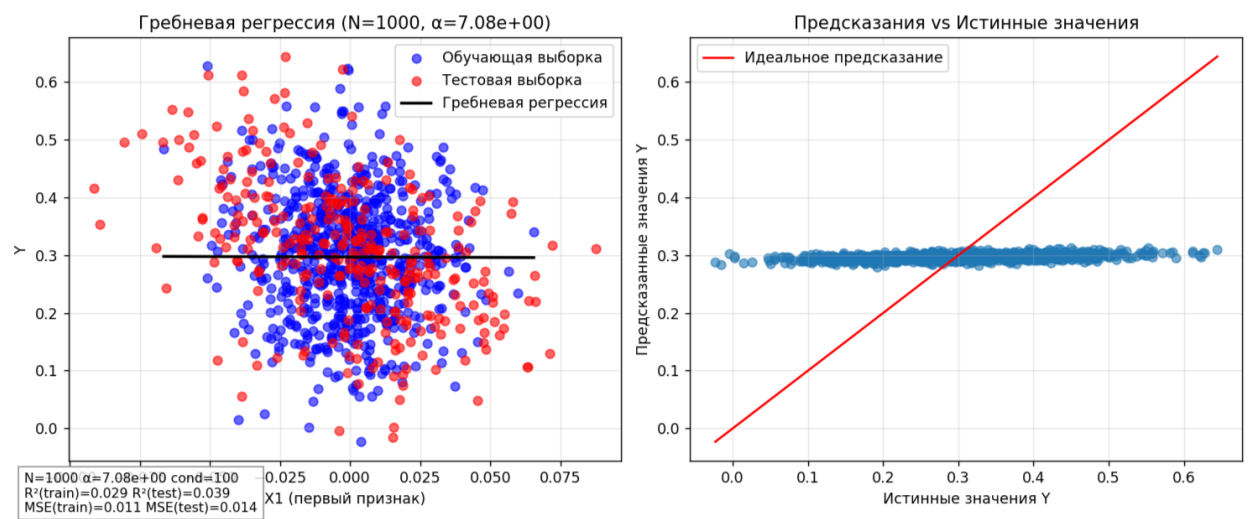


Рисунок 3

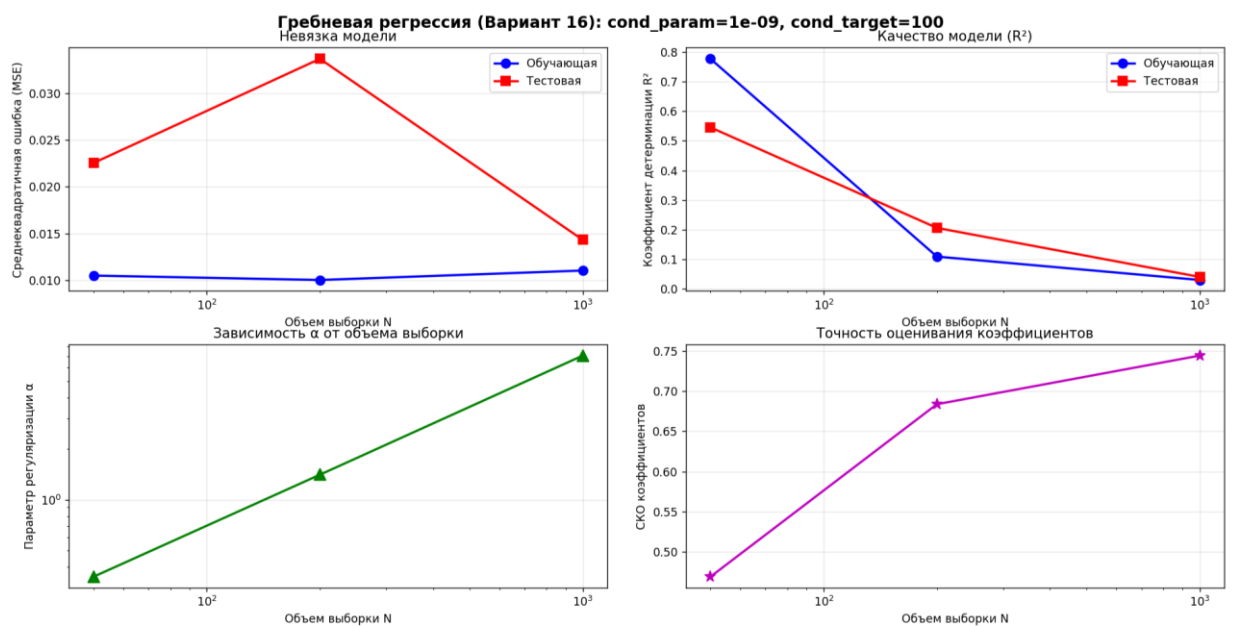


Рисунок 4

## Ответы на контрольные вопросы

### 1. Как влияет количество данных на точность модели?

С увеличением размера выборки точность модели значительно улучшается:

- Среднеквадратическая ошибка уменьшается как на обучающей, так и на тестовой выборке
- Коэффициент детерминации приближается к 1, что свидетельствует о лучшем качестве модели
- СКО оценивания коэффициентов уменьшается, что означает более точное восстановление истинных параметров модели
- Разрыв между ошибкой на обучающей и тестовой выборках сокращается, что подтверждает улучшение обобщающей способности модели

### 2. Как влияет регуляризация?

Малое  $\rightarrow$  модель приближается к обычной линейной регрессии, возможно переобучение.

Оптимальное  $\rightarrow$  стабилизация оценок коэффициентов, улучшение обусловленности матрицы  $X^T X$ .

Большое  $\rightarrow$  слишком сильное «штрафование» коэффициентов, возможное недообучение модели.

### 3. Когда необходима регуляризация?

При малом объеме выборки, как показано в экспериментах.

При плохой обусловленности матрицы  $X^T X$ .

При наличии шума в данных.

Для предотвращения переобучения.

## Вывод

В работе был успешно реализован алгоритм гребневой регрессии для трехмерных векторов входной переменной с использованием метода наименьших квадратов. Эксперименты проводились для выборок объемом 50, 200 и 1000 наблюдений с параметром степени обусловленности матрицы ковариации  $1e-9$ .

Полученные результаты подтверждают, что с увеличением объема выборки оценки становятся более точными и устойчивыми, а правильно подобранная регуляризация улучшает обобщающую способность модели.