Команда ГАДы:
Крылов Андрей m3234
Фунин Георгий m3234
Дударев Денис m3235

Отчет
Лабораторная работа №3

Введение

В этой лабораторной работе мы реализовали и исследовали методы стохастической оптимизации поиска минимума многомерных функций, а также сравнили их с методами из прошлых лабораторных работы. В работе были использованы Python 3 и библиотека scipy-optimyze.

1 Описание методов

SGD

Реализация SGD

Листинг 1: Реализация метода SGD

```python
def sgd(
        X, y,
        lr_fun,
        lr0=0.01,
        n_epochs=800,
        batch=1,
        decay=0.003,
        momentum=0,
        reg_type=None,
        tol=1e-4,
        patience=20,
        log_every=1
):
    m, n = X.shape
    w = np.zeros((n, 1))
    v = np.zeros_like(w)

    loss_hist, iter_hist = [], []
    best_loss, wait, it = np.inf, 0, 0

    for epoch in range(1, n_epochs + 1):
        lr = lr_fun(epoch - 1, lr0, decay)
        perm = np.random.permutation(m)

        for i in range(0, m, batch):
            xi = X[perm[i:i + batch]]
            yi = y[perm[i:i + batch]]

            g = calc_gradient(xi, yi, w, reg_type)
            v = momentum * v - lr * g
            w += v
            it += 1

            if it % log_every == 0:
                cur_loss = calc_loss(X, y, w, reg_type)
                loss_hist.append(cur_loss)
                iter_hist.append(it)

                if best_loss - cur_loss > tol:
                    best_loss = cur_loss
                    wait = 0
                else:
                    wait += 1
                    if wait >= patience:
                        return w, iter_hist, loss_hist
    return w, iter_hist, loss_hist
```

2 Графики

Реализуем отображение графиков на Python, который:

- отображает визуализацию 2D
- отрисовывает траекторию градиентного спуска

Используемые библиотеки

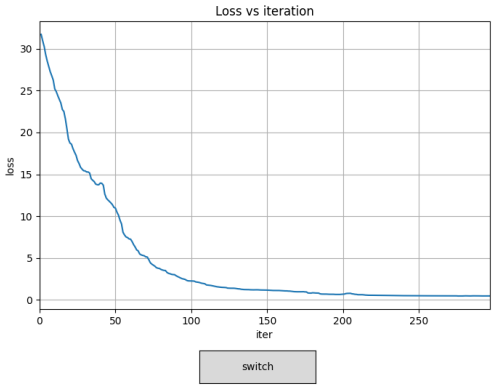
- `numpy` — работа с массивами данных
- `matplotlib.pyplot` — создание 2D-графиков

3 Описание результатов

Результаты для разных размеров батчей и learning rate = 10^5

3.1 Реализация SGD с разными batch size

batch size	Iterations	Mem (MiB)	MSE	MAE	R^2
1	297	1.3	0.425	0.526	0.350
10	402	0.9	0.394	0.508	0.397
100	449	1.0	0.391	0.503	0.402
500	466	1.5	0.390	0.503	0.403
1000	465	1.0	0.390	0.503	0.403



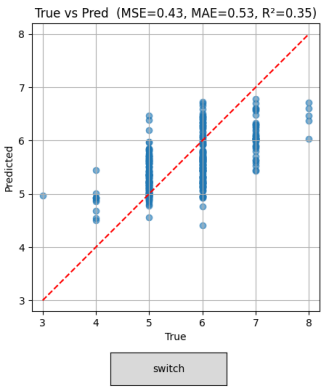
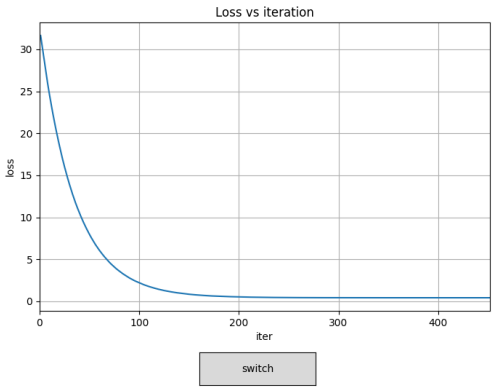


Рис. 1: batch size 1



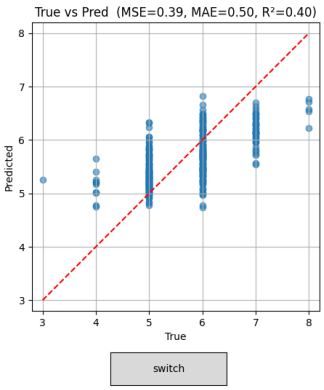
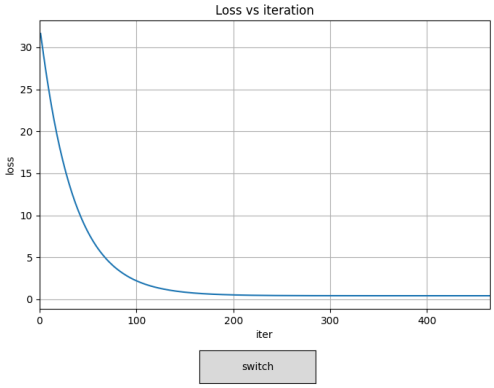


Рис. 2: batch size 100



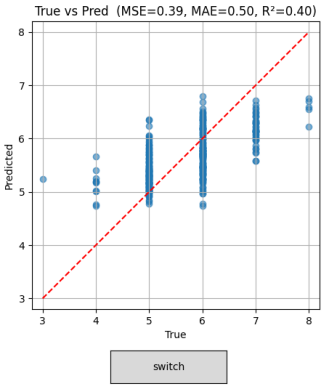


Рис. 3: batch size 1000

3.2 Выбор шага и регуляция

Method	Iterations	Mem (MiB)	MSE	MAE	R^2
exp	427	1.6	0.400	0.512	0.388
step	938	0.4	0.429	0.513	0.344
const	413	0.7	0.401	0.513	0.386
exp + l1(0.1)	437	2.0	0.401	0.513	0.387
step + l1(0.1)	933	0.5	0.429	0.513	0.344
const + l1(0.1)	426	1.2	0.401	0.513	0.387
exp + l2(0.1)	430	2.3	0.400	0.513	0.387
step + l2(0.1)	929	0.4	0.429	0.513	0.344
const + l2(0.1)	386	0.7	0.402	0.512	0.385
exp + elasticnet	430	2.6	0.400	0.512	0.388
step + elasticnet	932	1.0	0.429	0.513	0.344
const + elasticnet	374	0.1	0.403	0.513	0.384

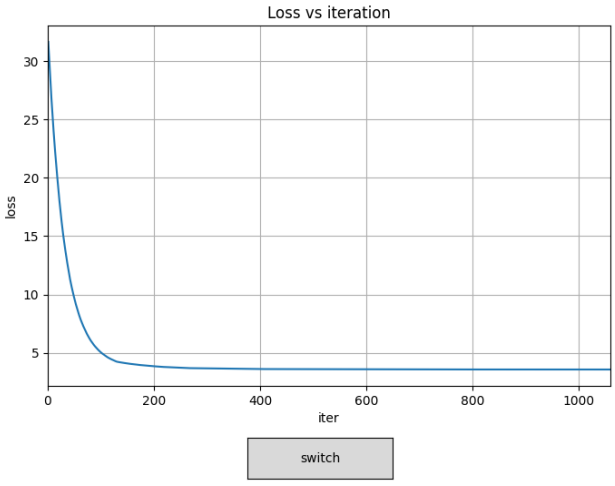


Рис. 4: step + l1(0.1)

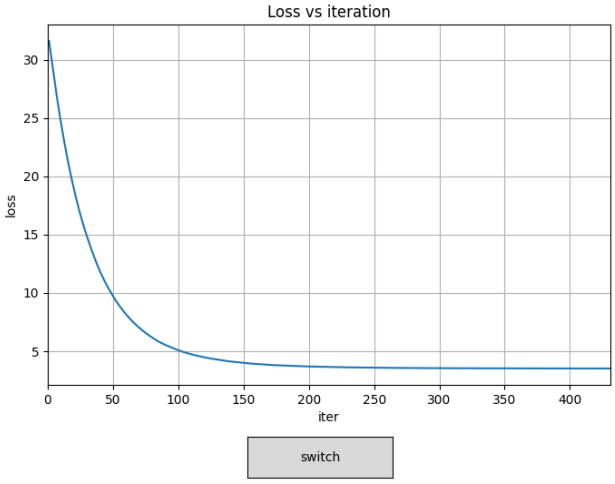


Рис. 5: exp + elasticnet

Регуляция меняет не так много, выбор шага реально влияет на результат, что наглядно видно. Так, exp хорошо себя показал в малом количестве итераций, но достаточно много тратил памяти. Противоположно же работал метод step. Чем то средним между двумя этими методами был const.

3.3 Keras и torch

Method	**MSE**	**MAE**	$\mathbf{R^2}$
KERAS	0.691	0.665	-0.057
TORCH	0.482	0.539	0.263

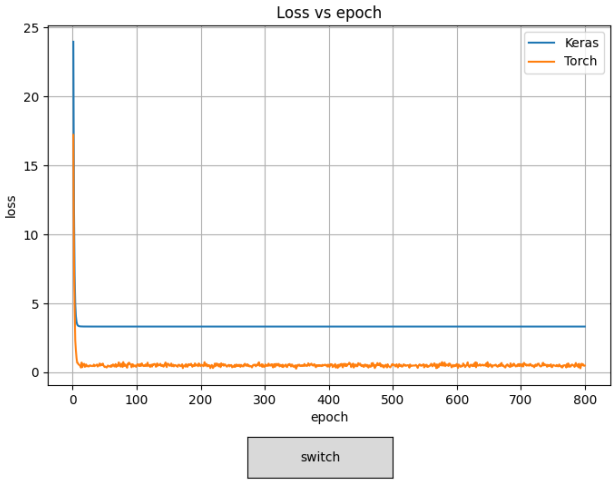
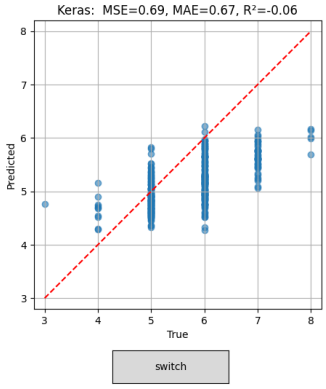


Рис. 6: Keras и Torch



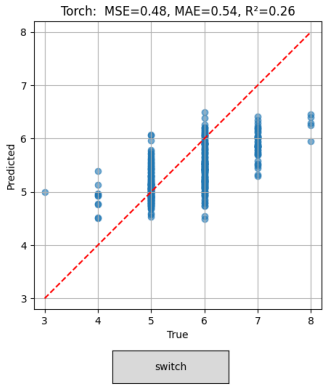


Рис. 7: batch size 100

Ключевое различие состоит в том, что в Keras вы просто описываете архитектуру модели и вызываете метод fit, передавая нужные параметры (число эпох, размер батча, разделение на тренировочную и валидационную выборки)

В PyTorch же вам нужно вручную реализовать этот цикл: в каждом проходе по эпохам формировать батчи, вычислять выходы модели, считать лосс, выполнять backward() и step(), то есть напрямую управлять всеми этапами оптимизации.

3.4 Momentum

Реализуем моментум

Momentum	**Iterations**	**Mem (MiB)**	**MSE**	**MAE**	**R^2**
0	417	1.9	0.401	0.512	0.387
0.3	460	2.0	0.401	0.514	0.386
0.6	434	2.2	0.400	0.512	0.388
0.9	453	1.8	0.399	0.512	0.389

Как видим моментум незначительно улучшил нащ рузультат

4 Поставим задачу регрессии с ε-insensitive-потерями

Пусть $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ — обучающая выборка. Будем искать линейный предсказатель в (возможно) «скрытом» пространстве

$$f(x) = w^\top \varphi(x) + b, \qquad w \in \mathcal{H},\ b \in \mathbb{R},$$

где $\varphi : \mathbb{R}^d \to \mathcal{H}$ — фиксированное отображение (ядровой трюк позволит не выписывать φ явно).

-insensitive-loss игнорирует ошибки радиусом до $\varepsilon > 0$:

$$\big|y - f(x)\big|_\varepsilon := \max\big(0,\ |y - f(x)| - \varepsilon\big).$$

Её график — «плоская» центральная полка шириной 2ε.

Прямая задача SVR

Чтобы свести абсолютную величину к линейным ограничениям, вводим неотрицательные slack-переменные ξ_i^+, ξ_i^- для превышений сверху/снизу трубы.

$$\min_{w,b,\xi^+,\xi^-} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^m \big(\xi_i^+ + \xi_i^-\big)$$

$$\begin{cases} y_i - (w^\top \varphi(x_i) + b) \le \varepsilon + \xi_i^+, \\ (w^\top \varphi(x_i) + b) - y_i \le \varepsilon + \xi_i^-, \\ \xi_i^+, \xi_i^- \ge 0, \qquad i = 1, \dots, m. \end{cases}$$

$\frac{1}{2}\|w\|^2$ — регуляризация, эквивалентная максимизации ширины зазора $2/\|w\|$; $C > 0$ — жёсткость модели: чем больше C, тем дороже ошибаться маленькое $C \to$ модель более плоская и допускает больше выбросов.

Лагранжиан и условия KKT

Введём множители $\alpha_i, \alpha_i' \ge 0$ для первых двух ограничений и $\eta_i^+, \eta_i^- \ge 0$ для $\xi_i^\pm \ge 0$. Лагранжиан:

$$\mathcal{L} = \tfrac{1}{2}\|w\|^2 + C\sum_i (\xi_i^+ + \xi_i^-) - \sum_i \alpha_i(\varepsilon + \xi_i^+ - y_i + w^\top \varphi_i + b)$$
$$- \sum_i \alpha_i'(\varepsilon + \xi_i^- + y_i - w^\top \varphi_i - b) - \sum_i \eta_i^+ \xi_i^+ - \sum_i \eta_i^- \xi_i^-,$$

где $\varphi_i = \varphi(x_i)$.

Условия стационарности $\partial\mathcal{L}/\partial(\cdot) = 0$:

$$\begin{cases} w = \sum_i (\alpha_i - \alpha_i')\varphi_i, \\ \sum_i (\alpha_i - \alpha_i') = 0, \\ 0 = C - \alpha_i - \eta_i^+, \quad 0 = C - \alpha_i' - \eta_i^-. \end{cases}$$

$\eta_i^\pm \ge 0 \Rightarrow 0 \le \alpha_i, \alpha_i' \le C$.

Двойственная задача

Подставляя выражение для w в $\mathcal{L}$ и используя $K(x_i, x_j) = \varphi_i^\top \varphi_j$, получаем:

$$\max_{\alpha, \alpha'} \; -\tfrac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(x_i, x_j) - \varepsilon\sum_i(\alpha_i + \alpha_i') + \sum_i (\alpha_i - \alpha_i')y_i$$
$$\text{s.t. } \sum_i (\alpha_i - \alpha_i') = 0, \qquad 0 \le \alpha_i, \alpha_i' \le C.$$

Размерность задачи — 2m вместо d; ядро $K(\cdot,\cdot)$ позволяет работать в бесконечномерном $\mathcal{H}$.

Разреженность. В оптимуме большинство α_i, α_i' обнулены. Точки с ненулевыми коэффициентами называются опорными (SV) и полностью определяют модель.

Восстановление прямого решения

$$w^\star = \sum_{i \in SV} (\alpha_i - \alpha_i')\varphi(x_i), \quad b^\star = y_k - w^\top \varphi(x_k) - \operatorname{sgn}(\alpha_k - \alpha_k')\,\varepsilon,$$

где x_k — любой SV, находящийся внутри трубы ($0 < \alpha_k < C$ или $0 < \alpha_k' < C$). Прогноз для новой точки x:

$$f(x) = \sum_{i \in SV} (\alpha_i - \alpha_i')\, K(x_i, x) + b^\star.$$