# Восстановление регрессии с использованием алгоритма Kernel RLS (KRLS)

Казакевич Анна Юрьевна

15 апреля 2025 г.

# 1 Постановка задачи

Пусть дана обучающая выборка

$$\{(x_i, y_i)\}_{i=1}^{l}, \quad x_i \in \mathbb{R}, \quad y_i \in \mathbb{R}.$$

Цель состоит в том, чтобы найти функцию $\hat{f}(x)$, минимизирующую квадратичную функцию потерь:

$$L(f) = \sum_{i=1}^{l} \left(y_i - f(x_i)\right)^2.$$

При использовании ядерных методов аппроксимация функции производится в виде:

$$\hat{f}(x) = \sum_{j=1}^{m} \alpha_j \, k(x_j, x),$$

где $k(x, x')$ — выбранное ядро (например, гауссово), а $\{x_j\}_{j=1}^{m}$ формируют словарь, т.е. подмножество обучающих примеров, используемых для аппроксимации.

**Основная задача:** построение линейной регрессии в пространстве признаков.

# 2 Базовая идея метода построения разрежённой аппроксимации

Классические ядерные методы используют все обучающие точки, что приводит к высоким вычислительным затратам и проблемам обобщения. Для решения этой проблемы применяется метод разрежённой аппроксимации, основанный на следующей идее:

(a) Каждый новый образец $x_t$ отображается в пространство признаков $\phi(x_t)$.

(b) Проверяется, можно ли аппроксимировать $\phi(x_t)$ как линейную комбинацию уже выбранных векторов $\{\phi(\tilde{x}_j)\}_{j=1}^{m_{t-1}}$ из словаря.

Эту проверку количественно выражают условием approximate linear dependency (ALD):

$$\delta_t \triangleq \min_{\mathbf{a}} \left\| \sum_{j=1}^{m_{t-1}} a_j \, \phi(\tilde{x}_j) - \phi(x_t) \right\|^2 \leq \nu,$$

где $\nu > 0$ — порог аппроксимации. Применяя ядровую функцию $k(x, x') = \langle \phi(x), \phi(x') \rangle$, данное условие преобразуется в:

$$\delta_t = k(x_t, x_t) - \mathbf{k}_{t-1}(x_t)^\top \, K_{t-1}^{-1} \, \mathbf{k}_{t-1}(x_t),$$

где:

- $\mathbf{k}_{t-1}(x_t) = \big(k(\tilde{x}_1, x_t),\, k(\tilde{x}_2, x_t),\, \ldots,\, k(\tilde{x}_{m_{t-1}}, x_t)\big)^\top$;

- $K_{\tilde{t-1}} = \left[k(\tilde{x}_i, \tilde{x}_j)\right]_{i,j=1}^{m_{t-1}}$.

Если $\delta_t > \nu$, образец $x_t$ добавляется в словарь, что позволяет обеспечить разрежённое представление решения и уменьшить вычислительные затраты.

# 3 Алгоритм Kernel RLS (KRLS)

Классический алгоритм RLS ищет вектор весов $\mathbf{w}$, минимизирующий сумму квадратов ошибок в модели

$$f(x) = \langle \mathbf{w},\, \phi(x) \rangle.$$

В ядерном случае, с использованием представления:

$$\mathbf{w}_t = \sum_{j \in D_t} \alpha_j\, \phi(x_j),$$

решение принимает вид:

$$\hat{f}_t(x) = \sum_{j \in D_t} \alpha_j\, k(x_j, x).$$

Основное различие с классическим RLS состоит в том, что словарь $D_t$ формируется динамически с использованием ALD-проверки, а матрица, инвертированная в ходе обновлений, имеет размер, равный числу опорных точек, а не общему числу образцов.

Вновь поступивший вектор добавляется в словарь, т.е. становится опорным, если он не является приближенной линейной аппроксимацией векторов, составляющих словарь к моменту его поступления.

## 3.1 Обновление при добавлении новой точки (ALD не выполнено)

Если для нового образца $x_t$ выполнено условие

$$\delta_t = k(x_t, x_t) - \mathbf{k}_{t-1}(x_t)^\top\, K_{\tilde{t-1}}^{-1}\, \mathbf{k}_{t-1}(x_t) > \nu,$$

то $x_t$ добавляется в словарь, а инвертированная матрица $K_{\tilde{t}}^{-1}$ обновляется по блочной формуле:

$$K_{\tilde{t}}^{-1} = \frac{1}{\delta_t} \begin{pmatrix} \delta_t\, K_{\tilde{t-1}}^{-1} + a_t\, a_t^\top & -a_t \\ -a_t^\top & 1 \end{pmatrix},$$

где $a_t = K_{\tilde{t-1}}^{-1}\, \mathbf{k}_{t-1}(x_t)$.

Обновление коэффициентов записи решения производится так:

$$\tilde{\alpha}_t = \begin{pmatrix} \tilde{\alpha}_{t-1} - a_t\, \dfrac{e_t}{\delta_t} \\ \dfrac{e_t}{\delta_t} \end{pmatrix}, \quad \text{где } e_t = y_t - \mathbf{k}_{t-1}(x_t)^\top\, \tilde{\alpha}_{t-1}.$$

### 3.2 Обновление без расширения словаря (ALD выполнено)

Если же $\delta_t \leq \nu$, новая точка аппроксимируется имеющимся словарём, и обновление производится без его расширения. В этом случае вводится вспомогательная матрица $P$ (инициализируется как единичная), которая корректируется по рекуррентной формуле:

$$P_t = P_{t-1} - \frac{P_{t-1}\, a_t\, a_t^\top\, P_{t-1}}{1 + a_t^\top\, P_{t-1}\, a_t},$$

а обновление коэффициентов осуществляется по схеме:

$$\tilde{\alpha}_t = \tilde{\alpha}_{t-1} + K_{\tilde{t-1}}^{-1}\, q_t\, e_t,$$

где $q_t = P_{t-1}\, a_t$.

### 3.3 Псевдокод алгоритма

Ниже представлен псевдокод алгоритма KRLS с разрежённой аппроксимацией:

**Algorithm 1** Kernel RLS с разрежённой аппроксимацией (ALD)

**Инициализация:**

- $D \leftarrow \{x_1\}$;
- $K_{\tilde{D}}^{-1} \leftarrow [\,1/k(x_1, x_1)\,]$;
- $\tilde{\alpha}_1 \leftarrow y_1/k(x_1, x_1)$;
- $P \leftarrow [1]$.

**for** $t = 2, \ldots, l$ **do**

Получить новый образец $(x_t, y_t)$.

Вычислить $\mathbf{k}_{t-1}(x_t) = \begin{pmatrix} k(\tilde{x}_1, x_t) \\ \vdots \\ k(\tilde{x}_{m_{t-1}}, x_t) \end{pmatrix}$ и $k(x_t, x_t)$.

Вычислить $a_t = K_{\tilde{t-1}}^{-1}\, \mathbf{k}_{t-1}(x_t)$.

Рассчитать

$$\delta_t = k(x_t, x_t) - \mathbf{k}_{t-1}(x_t)^\top\, a_t.$$

**if** $\delta_t > \nu$ **then**

**(Добавление новой точки)**

- $D_t \leftarrow D_{t-1} \cup \{x_t\}$.
- Обновить $K_{\tilde{t}}^{-1}$ по формуле:

$$K_{\tilde{t}}^{-1} = \frac{1}{\delta_t} \begin{pmatrix} \delta_t\, K_{\tilde{t-1}}^{-1} + a_t\, a_t^\top & -a_t \\ -a_t^\top & 1 \end{pmatrix}.$$

- Вычислить $e_t = y_t - \mathbf{k}_{t-1}(x_t)^\top\, \tilde{\alpha}_{t-1}$.
- Обновить коэффициенты:

$$\tilde{\alpha}_t = \begin{pmatrix} \tilde{\alpha}_{t-1} - a_t\, \dfrac{e_t}{\delta_t} \\ \dfrac{e_t}{\delta_t} \end{pmatrix}.$$

- Обновить матрицу $P$ (расширив размерность на 1).

**else**

**(Обновление без расширения словаря)**

- Обновить $P$ по формуле:

$$P_t = P_{t-1} - \frac{P_{t-1}\, a_t\, a_t^\top\, P_{t-1}}{1 + a_t^\top\, P_{t-1}\, a_t}.$$

- Вычислить $e_t = y_t - \mathbf{k}_{t-1}(x_t)^\top\, \tilde{\alpha}_{t-1}$.
- Определить $q_t = P_{t-1}\, a_t$ и обновить

$$\tilde{\alpha}_t = \tilde{\alpha}_{t-1} + K_{\tilde{t-1}}^{-1}\, q_t\, e_t.$$

**end if**

**end for**

**return** Словарь $D_l$ и коэффициенты $\tilde{\alpha}_l$, задающие аппроксимацию

$$\hat{f}(x) = \sum_{j \in D_l} \tilde{\alpha}_j\, k(\tilde{x}_j, x).$$

# 4 Экспериментальные результаты

Для демонстрации работы алгоритма KRLS были проведены эксперименты на модельных данных с использованием функции $\sin(x)$.

## 4.1 Практическая реализация метода

Метод реализован на python.

Код можно найти по ссылке: KRLS SVM algorithm

## 4.2 Восстановление регрессии по точным данным

Для точных данных генерируется равномерная выборка $x \in [-5, 5]$ с истинной зависимостью

$$y = \sin(x),$$

при этом шум отсутствует. Алгоритм KRLS восстанавливает функцию, используя число опорных векторов $m \ll 200$.

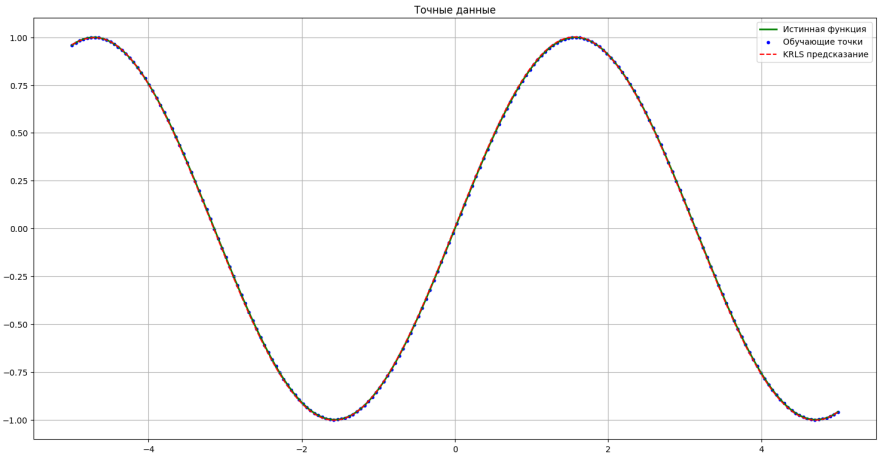


Рис. 1: Восстановление регрессии по точным данным (без шума)

**Вывод программы**
Число опорных векторов = 31
Финальный MSE = 2.7781835056972007e-05

## 4.3 Восстановление регрессии по зашумлённым данным

При зашумлённых данных к истинным значениям добавляется Гауссов шум с, например, $\sigma_{noise} = 0.2$:

$$y = \sin(x) + \eta, \quad \eta \sim \mathcal{N}(0, 0.2).$$

При этом число опорных векторов может увеличиться для адекватного аппроксимирования зависимости, однако точность предсказания остается высокой.

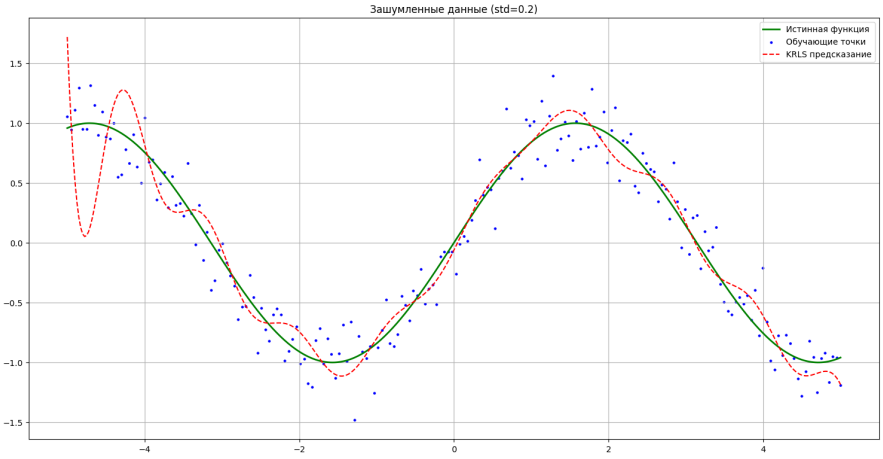


Рис. 2: Восстановление регрессии по зашумлённым данным

**Вывод программы**

Число опорных векторов = 31
Финальный MSE = 0.03445072605449376

## 4.4 Зависимость MSE от числа опорных векторов

В ходе экспериментов фиксируется динамика количества опорных векторов, а также измеряется среднеквадратичная ошибка (MSE) на тестовой выборке. Результаты показывают, что уменьшение порога $\nu$ приводит к увеличению числа опорных точек и уменьшению MSE (до определённого уровня), что иллюстрирует компромисс между размером модели и точностью аппроксимации.

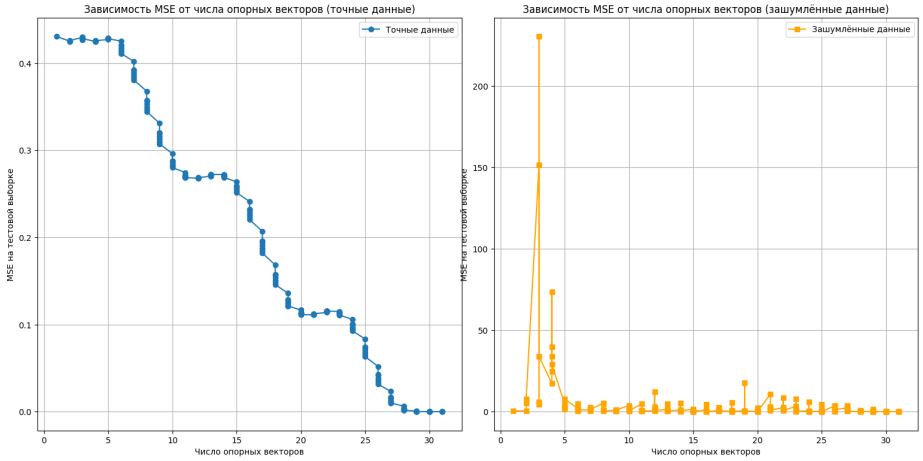


Рис. 3: Зависимость среднеквадратичной ошибки (MSE) от числа опорных векторов. На оси $x$ — число опорных векторов, на оси $y$ — значение MSE.

# 5 Выводы

- Алгоритм KRLS расширяет классический метод рекурсивного наименьших квадратов (RLS) в контексте ядерных методов, позволяя решать задачи регрессии в высокоразмерном (или даже бесконечномерном) пространстве признаков. Основная идея заключается в представлении искомой функции в виде линейной комбинации ядерных функций, а также в последовательном построении разрежённого словаря, состоящего только из тех обучающих примеров, которые вносят существенную информацию в аппроксимацию.

- Процедура отбора опорных точек на основе условия approximate linear dependency (ALD) позволяет существенно сократить размер модели. Это даёт два основных преимущества: во-первых, уменьшается вычислительная сложность как при обучении, так и при использовании модели для предсказаний; во-вторых, разрежённое представление способствует улучшению обобщающих свойств алгоритма, предотвращая переобучение и уменьшая зависимость от шума в данных.

- Порог $\nu$ определяет точность аппроксимации в процедуре ALD. При меньшем $\nu$ модель стремится включить больше опорных точек, что приводит к повышенной точности, но увеличивает вычислительную сложность и риск переобучения. При увеличении $\nu$ модель становится более разрежённой, что снижает затраты по времени и памяти, однако может снизиться точность аппроксимации. Таким образом, оптимальный выбор $\nu$ обеспечивает баланс между точностью и эффективностью модели.

- Алгоритм KRLS способен обновлять модель в режиме онлайн за фиксированное количество операций, поскольку вычислительная сложность обновления зависит от размера словаря (а не общего числа обучающих точек). Благодаря динамическому формированию разрежённого представления, алгоритм сохраняет стабильное время работы даже при поступлении большого количества данных, что делает его подходящим для реального времени и потоковых приложений.

# 6 Литература

1. Engel Y., Mannor S., Meir R. The Kernel Recursive Least Squares Algorithm // Advances in Neural Information Processing Systems (NeurIPS). – 2003.