Лабораторная работа №4

Выполнили: Гуревич Михаил, Трохан Александр и Соловьёв Роман, M33001

✅ Полный код лабораторной работы с комментариями можно найти на Github, а отчёт в **Notion**.

Цель работы

Построение стационарного ряда с помощью модели авторегрессии, его анализ и предсказание.

Задачи

- Построить авторегрессионную модель временного ряда третьего порядка.
- Проверить, является ли построенный ряд стационарным.
- Сгенерировать следующие значения ряда с помощью модели и построить график ряда.
- Преобразовать ряд к последовательности векторов задержек, обучить машину опорных векторов и определить авторегрессионные параметры.
- Построить продолжение временного ряда по найденным параметрам и сравнить с реальным рядом.
- Проанализировать результаты при различных ядрах и других гиперпараметрах модели.

Ход работы

Стационарный временной ряд — ряд, свойства которого не меняются с течением времени (т.е. матожидание, дисперсия и ковариация между равными отсчётами времени не меняется).

Пусть есть стационарный временной ряд $\{x_t\}$. Если ряд подчиняется авторегрессионной модели порядка p, то его t-ый записывается как:

$$x_t = \mu + a_1 x_{t-1} + a_2 x_{t-2} + ... + a_p x_{t-p} + \varepsilon_t$$
$$x_t \in \mathrm{AR}(p)$$

где ε_t — шумовая компонента, $\mathrm{AR}(p)$ — авторегрессионная модель порядка p.

Также введём обозначение $\gamma_p \equiv \mathrm{cov}(x_t, x_{t+p})$, соответсвенно $\gamma_0 \equiv \mathrm{cov}(x_t, x_t)$.

Построение стационарного ряда

В данной работе будет рассматриваться авторегрессия $\mathrm{AR}(3)$ где $\mu = 0$. Для работы с авторегрессией реализуем класс `AR3`.

```
class AR3:
    def __init__(self, coefs: np.ndarray, x_start: np.ndarray):
        if len(coefs) != 4:
            raise ValueError("Wrong number of coefficients.")

        if len(x_start) != 3:
            raise ValueError("Wrong number of first Xs.")

        self.coefs = coefs.copy()
        self.x = np.zeros(LEN)
        self.x[0] = x_start[0]
        self.x[1] = x_start[1]
        self.x[2] = x_start[2]
        self.eps = np.random.normal(0, 5, LEN)
```

Так как нам требуется получить стационарный ряд с помощью авиорегрессии, то для выбора коэффициентов $a_1, ..., a_3$ воспользуемся анализом автокорреляционной функции:

Умножим уравнение модели $x_t = a_1 x_{t-1} + a_2 x_{t-2} + a_3 x_{t-3} + \varepsilon_t$ на x_{t-1}, x_{t-2} и x_{t-3}, а затем возьмём математическое ожидание от данных величин. В данном случае $\mathrm{E}(x_t \cdot x_{t-p}) = \gamma_p$ и $\mathrm{E}(x_t \cdot \varepsilon_t) = 0$, т.к $\mathrm{E}(\varepsilon_t) = 0$. Тогда получим систему из следующих уравнений:

$$\begin{cases} \gamma_1 - a_1\gamma_0 - a_2\gamma_1 - a_3\gamma_2 = 0 \\ \gamma_2 - a_1\gamma_1 - a_2\gamma_0 - a_3\gamma_1 = 0 \\ \gamma_3 - a_1\gamma_2 - a_2\gamma_1 - a_3\gamma_0 = 0 \end{cases}$$

Разделив каждое из уравнений на γ_0 и приняв $r_i = \dfrac{\gamma_i}{\gamma_0}$ (коэффциенты автокорреляции), получим:

$$\begin{cases} r_1 - a_1 - a_2 r_1 - a_3 r_2 = 0 \\ r_2 - a_1 r_1 - a_2 - a_3 r_1 = 0 \\ r_3 - a_1 r_2 - a_2 r_1 - a_3 = 0 \end{cases}$$

Решая систему, получим уравнения для r:

$$r_1 = \frac{a_2 a_3 + a_1}{1 - a_2 - a_1 a_3 - a_3^2}$$
$$r_2 = (a_1 + a_3) r_1 + a_2 = \frac{(1 - a_2) r_1 - a_1}{a_3}$$
$$r_3 = a_1 r_2 + a_2 r_1 + a_3$$

Отсюда делаем вывод, что $a_2 - a_1 a_3 - a_3^2 \neq 1$, $a_3 \neq 0$. Также для достижения стационарности ряда коэффициенты автокорреляции должны удовлетворять условию $r_i < \left|1\right|$. На основании этого был написан код, который генерирует случайные коэффициенты, пока не будут выполнены все условия:

```
def r1(a: np.ndarray):
    return (a[2] * a[3] + a[1]) / (1 - a[2] - a[1] * a[3] - a[3] * a[3])

def r2(a: np.ndarray):
    return (a[1] + a[3]) * r1(a) + a[2]

def r3(a: np.ndarray):
    return a[1] * r2(a) + a[2] * r1(a) + a[3]

def find():
    while (True):
        a = np.random.normal(0, 1, 4)
        a[0] = 0
        if (a[2] + a[1] * a[3] + a[3] * a[3]) != 1 and \
                abs(r1(a)) < 1 and \
```

```
                abs(r2(a)) < 1 and \
                abs(r3(a)) < 1 and \
                a[3] != 0:
            return [a, r1(a), r2(a), r3(a)]
```

В нашем случае был сгенерирован ряд $x_t = 0.5426x_{t-1} - 0.4634x_{t-2} - 0.4657x_{t-3}$.

Проверка ряда на стационарность

Несмотря на то, что сгенерированный ряд уже является стационарным из-за правильного выбора коэффициентов, можем дополнительно проверить стационарность с помощью нахождения корней характеристического уравнения $1 - a_1 z - a_2 z^2 - a_3 z^3 = 0$:

```
def _check_stationarity(self):
    """Check if model is stationarity."""

    z = smp.Symbol("z")
    equation_solution = smp.solve(
        smp.Eq(1 - self.coefs[1] * z - self.coefs[2] * z ** 2 - self.coefs[3] * z ** 3, 0))

    return equation_solution
```

В нашем случае имеем корни:

$$
\begin{aligned}
z_0 &= -2.0637 \\
z_1 &= 0.5343 - 0.8689i \\
z_2 &= 0.5343 + 0.8689i
\end{aligned}
$$

Модуль всех корней больше 1, а значит ряд стационарный.

Генерация ряда

Сгенерируем первые 1000 значений ряда с помощью функции:

```
def _generate_first_values(self, length=LEN):
        """Generate first `length` values of the series."""
```

```
        for i in range(3, length):
            self.x[i] = self.coefs[0] \
                + self.coefs[1] * self.x[i - 1] \
                + self.coefs[2] * self.x[i - 2] \
                + self.coefs[3] * self.x[i - 3] \
                + self.eps[i]
```

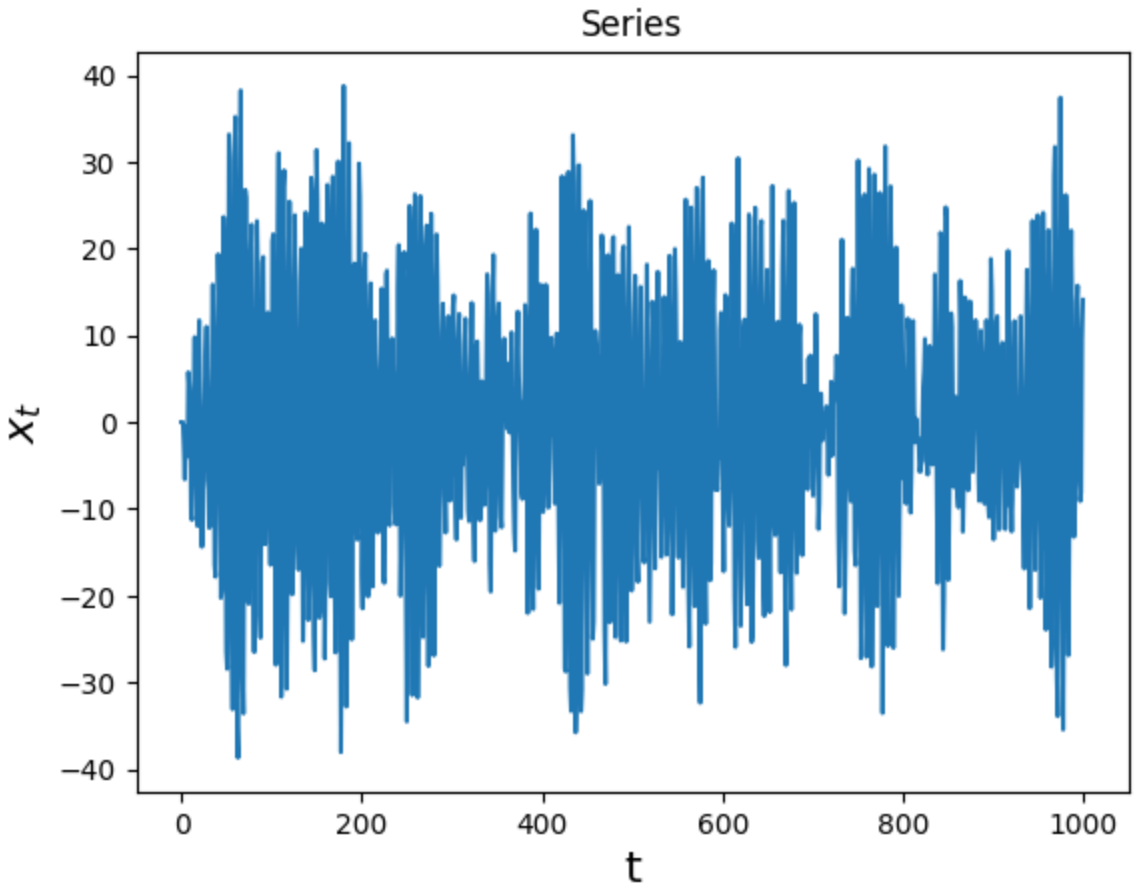


График ряда

Предсказание ряда

Преобразуем ряд к последовательности векторов задержек:

```
def _delay_vector(self):
    """Create delay vectors."""

    vectors = np.ndarray(shape=(LEN - 3, 3))

    for i in range(2, LEN - 1):
```

```
        vector_i = np.array([self.x[i - 2], self.x[i - 1], self.x[i]])
        vectors[i - 2] = vector_i

    self.vectors_train = vectors[:int(0.8 * (LEN - 3))].copy()
    self.last_vec = vectors[int(0.8 * (LEN - 3))].copy()
    self.y_train = self.x[3:int(0.8 * (LEN - 3)) + 3].copy()
    self.y_test = self.x[int(0.8 * (LEN - 3)) + 3:].copy()
```

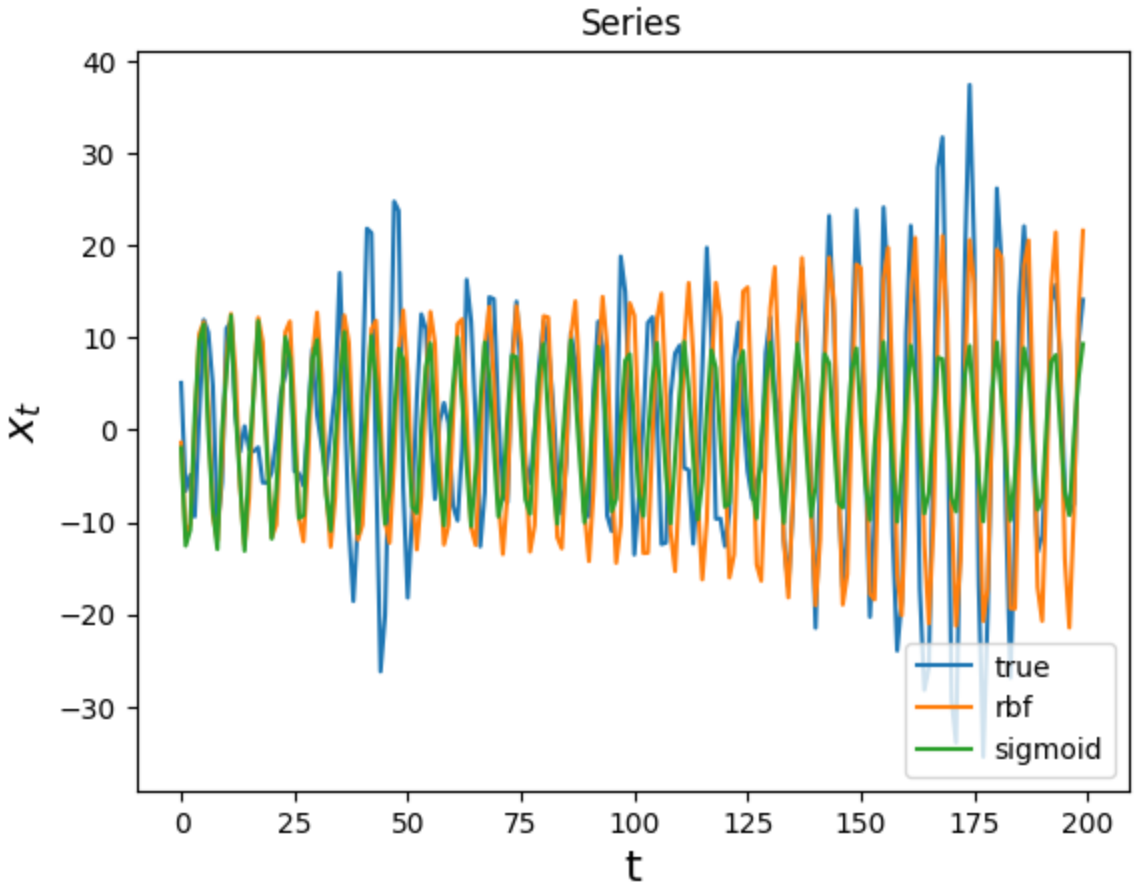
С помощью них обучим готовую реализацию машины опорных векторов из `Scilit-learn`. Рассмотрим разные варианты модели с ядрами `linear,` `poly`, `rbf` и `sigmoid`. Обучение производилось на первых 800 значениях ряда.

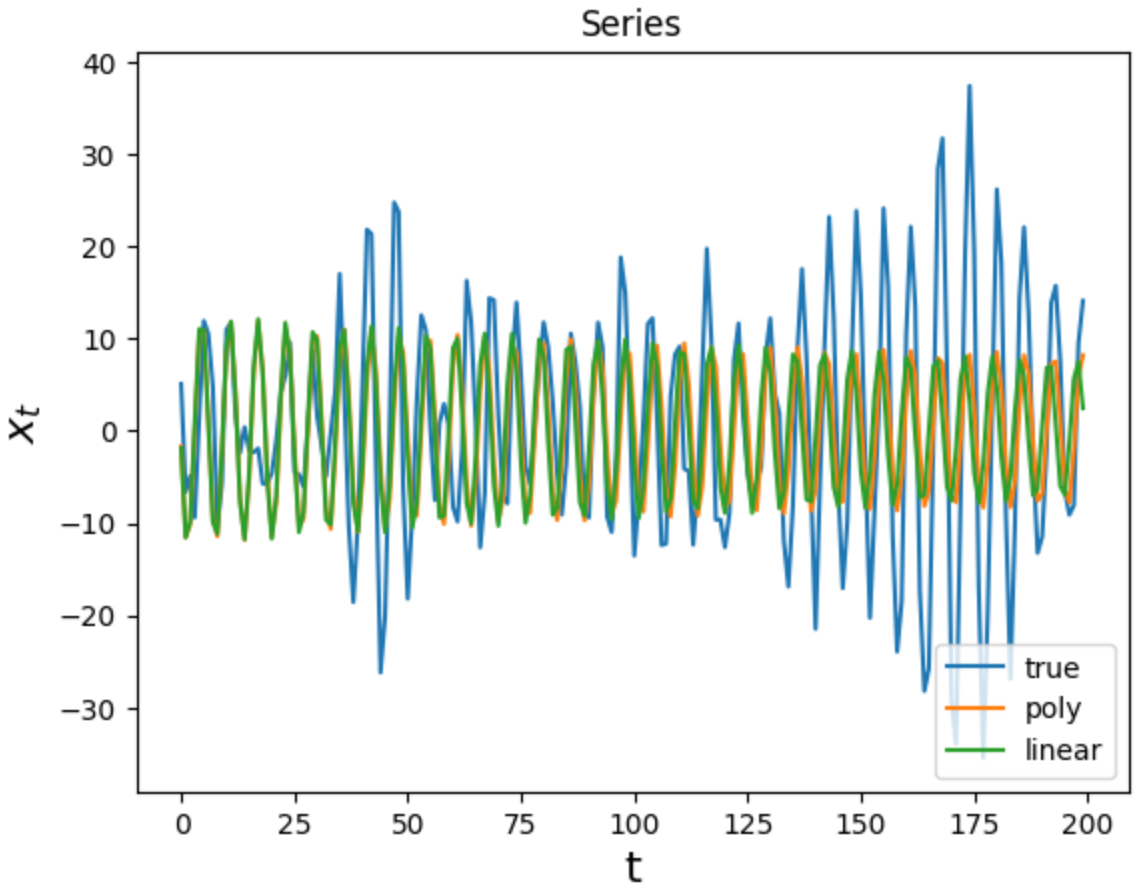
Код для подбора гиперпараметров для разных ядер:

```
if kernel == "linear":
    for C in range(2, 32, 2):
        res = self._predict_linear(C=C / 10)
        if lowest_error == -1 or self._calc_error(res) < lowest_error:
            lowest_error = self._calc_error(res)
            best_params = [C / 10]
            best_predict = res

elif kernel == "poly":
    for C in range(20, 205, 5):
        for degree in range(2, 5, 1):
            for coef0 in range(0, 22, 2):
                gamma = "scale"
                res = self._predict_poly(
                    C=C / 100, gamma=gamma, coef0=coef0 / 10, degree=degree)
                if self._calc_error(res) < lowest_error:
                    lowest_error = self._calc_error(res)
                    best_params = [C /100, gamma, coef0 / 10, degree]
                    best_predict = res

elif kernel == "rbf":
    for C in range(20, 205, 5):
        gamma = "scale"
        res = self._predict_rbf(C=C / 100, gamma=gamma)
        if self._calc_error(res) < lowest_error:
            lowest_error = self._calc_error(res)
            best_params = [C / 100, gamma]
            best_predict = res

else:  # sigmoid
    for C in range(2, 32, 2):
        for i in range(0, 2, 1):
            for coef0 in range(-20, 22, 2):
                gamma = "scale"
```

```
                if i == 1:
                    gamma = "auto"

                res = self._predict_sigmoid(C=C / 10, gamma=gamma, coef0=coef0 / 10)
                if self._calc_error(res) < lowest_error:
                    lowest_error = self._calc_error(res)
                    best_params = [C / 10, gamma, coef0 / 10]
```

Ниже приведены графики последних 200 значений ряда и предсказание моделей:





Итоговые гиперпараметры моделей и их ошибки можно найти в коде лабораторной работы.

Вывод

Была сгенерирован стационарный ряд с помощью модели авторегрессии и построены его предсказания с помощью машины опорных векторов с различными гиперпараметрами. Также было выявлено, что ядра `linear` и `poly` хуже работают при предсказании ряда, чем `rbf` и `sigmoid`.