Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математической физики

Лытов Дмитрий Андреевич

Применение Байесовских методов машинного обучения для решения обратной задачи модели Лотки — Вольтерры

КУРСОВАЯ РАБОТА

Научный руководитель:
к.ф-м.н. С.Б.Березин

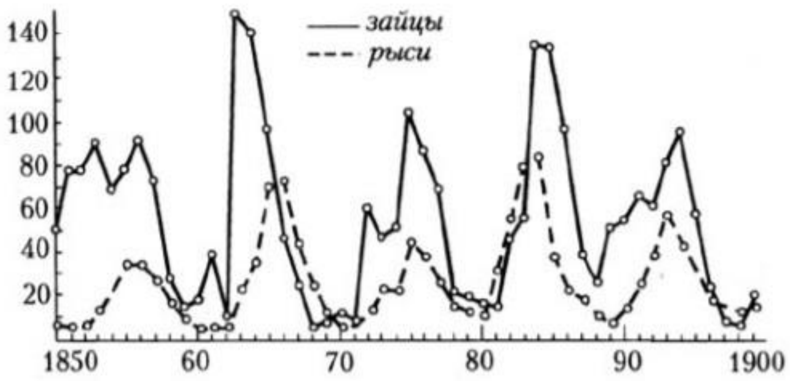
Москва, 2019

Оглавление

Введение

Система «хищник — жертва» — сложная экосистема, для которой реализованы долговременные отношения между видами хищника и жертвы.

На (рис. 1) приведены примеры реальной динамики численности популяции зайцев и *рысей [1]*



Данные промысла зайца (сплошная кривая) и рыси (пунктирная) в Гудзоновом заливе в течение второй половины XIX века

Рис. 1

Одна из простейших математических моделей для описания динамики численности популяций - ***Модель Лотки — Вольтерры***. В ней полагаются известными некоторые параметры, характеризующие регуляцию численности популяций, что позволяет смоделировать их динамику. Но что, если наша задача в обратном? Допустим, что уже имеем измерения численности популяций за последние несколько лет наблюдений за видами, но нам хотелось бы по этим данным найти исходные параметры – числовые значения, характеризующие регуляцию их численности?

Цель работы

1. По данным измерениям численности популяций видов хищников и жертв приблизить исходные параметры модели Лотки – Вольтерры.

2. Реализовать алгоритм, решающий данную задачу с помощью Байесовских методов машинного обучения и выяснить, насколько точен этот подход в сравнении с другими известными моделями решения обратной задачи для модели Лотки – Вольтерры.

Основная часть

Модель Лотки – Вольтерры

$$\begin{cases} dx = (\alpha - \beta y(t))x(t) \\ dy = (-\gamma + \delta x(t))y(t) \end{cases} \quad (1)$$

$x > 0$ – численность популяции жертв

$y > 0$ – численность популяции хищников

$\alpha > 0$ – коэффициент рождаемости жертв (на единицу особи, на единицу времени)

$\beta > 0$ – коэффициент смертности жертв

(на единицу особи жертвы, на единицу особи хищника, на единицу времени)

$\gamma > 0$ – коэффициент смертности хищников

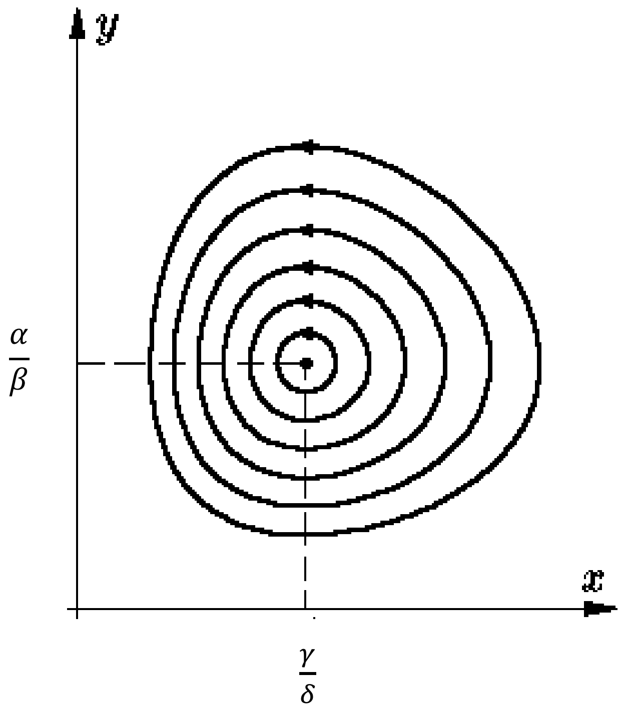
(на единицу особи, на единицу времени)

$\delta > 0$ – коэффициент смертности хищников

(на единицу особи жертвы, на единицу особи хищника, на единицу времени)

Решением на фазовой плоскости является семейство замкнутых траекторий с центром в равновесной точке: $x_e = \frac{\gamma}{\delta}; y = \frac{\alpha}{\beta}$

Координаты равновесной точки находятся [2]



Описание данных

Данные о численности популяций видов будет моделировать самостоятельно. Для этого был реализован алгоритм для прямого решения поставленной задачи с начальными условиями $x(0) = x_0; y(0) = y_0$ После того, как я передаю ему на вход некоторые исходные параметры, он моделирует динамику популяций (в виде двух непрерывных функций от времени), и возвращает случайным образом сделанные измерения (берет точки на графике /*каком*/ с некоторыми отклонениями, соответствующими погрешности измерения):

$$\{(x_i, y_i, t_i)\}_{i=1}^{sample\_size} \tag{2}$$

Постановка задачи

На основе данных (2) приблизить исходные параметры системы $\alpha_0, \beta_0, \gamma_0, \delta_0$.

Для оценки точности работы алгоритма используем функционал $\sqrt{\frac{(\alpha_0-\alpha')^2+(\beta_0-\beta')^2+(\gamma_0-\gamma')^2+(\delta_0-\delta')^2}{4}}$.

Для визуализации будем строить решение прямой задачи с найденными параметрами и оценивать схожесть с решением задачи с исходными параметрами.

Возможные варианты решения

Для решения поставленной задачи я использовал Байесовские методы машинного обучения. *Байесовский вывод* — статистический вывод, в котором свидетельство и/или наблюдение используются, чтобы обновить или вновь вывести вероятность того, что гипотеза может быть верной. Название байесовский происходит от частого использования в процессе вывода теоремы Байеса. Вывод производится с помощью метода «Expectation propagation». Но при обучении использовал две различные модели:

1. Разностная модель.

   Пусть имеется выборка из n точек (x_i, y_i, t_i), тогда имеем

$$\begin{cases} (dx)_i = (\alpha - \beta y_i)x_i \\ (dy)_i = (-\gamma + \delta x_i)y_i \end{cases}$$

$(dx)_i, (dy)_i, x_i, y_i$
— случайные величины, для которых мы можем задать наблюдаемые значения

$$(dx)_1 = \frac{x_2 - x_1}{t_2 - t_1} \ ; \ (dx)_i = \frac{x_{i+1} - x_{i-1}}{t_{i+1} - t_{i-1}} \ ; \ (dx)_n = \frac{x_n - x_{n-1}}{t_n - t_{n-1}}$$

Аналогично для $(dy)_i$

2. Модель с использованием первого интеграла

Найдем первый интеграл системы (1):

$$\frac{dx}{dy} = \frac{\big(\alpha - \beta y(t)\big)x(t)}{\big(-\gamma + \delta x(t)\big)y(t)}$$

$$\frac{\big(-\gamma + \delta x(t)\big)dx}{x(t)} = \frac{\big(\alpha - \beta y(t)\big)dy}{y(t)}$$

$$\left(-\frac{\gamma}{x(t)} + \delta\right)dx = \left(\frac{\alpha}{y(t)} - \beta\right)dy$$

$$d(-\gamma \ln(x) + \delta x) = d(\alpha \ln(y) - \beta y)$$

$$d(\alpha \ln(y) - \beta y + \gamma \ln(x) - \delta x) = 0$$

$$\delta x + \beta y - \gamma \ln(x) - \alpha \ln(y) = C$$

По определению, значение функции слева принимает одинаковое значение на всех точках наперед взятой фазовой траектории. Воспользуемся этим, чтобы описать модель.

Пусть имеется выборка из n точек (x_i, y_i), тогда имеем

$$\sum_{i=1}^{n} \delta x_i + \beta y_i - \gamma \ln(x_i) - \alpha \ln(y_i) - C = 0$$

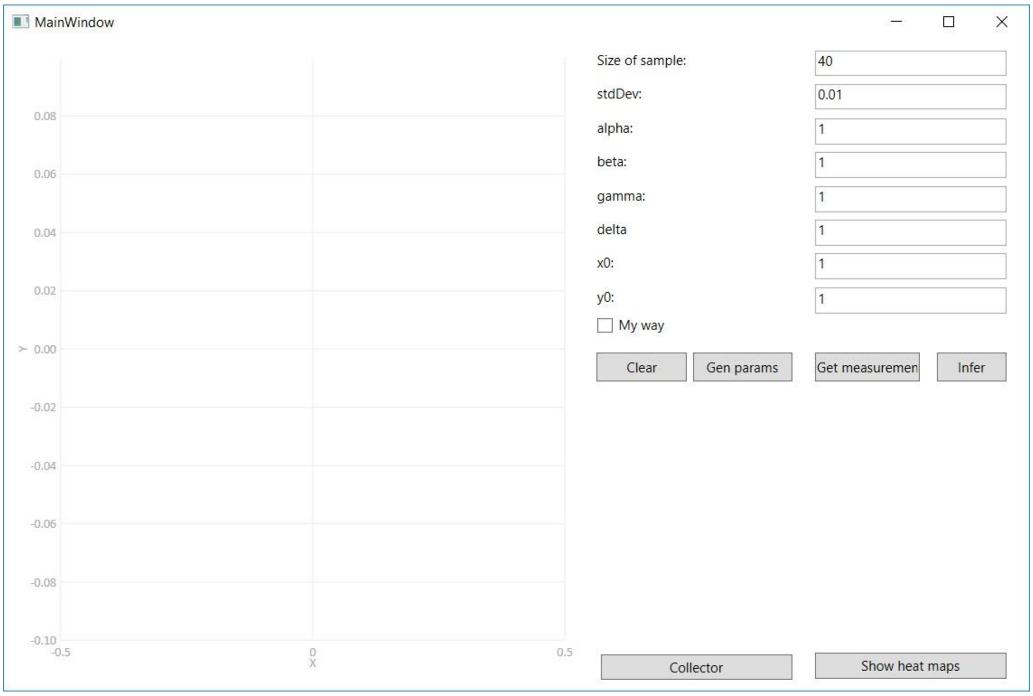
После обучения получаем значения $\alpha', \beta', \gamma', \delta', C'$. Дополнительно, C' предоставляет данные о конкретной траектории, входные данные не требуют временных отметок, но и параметры получены с точностью до мультипликативной константы (из-за отсутствия данных о времени)

Решение

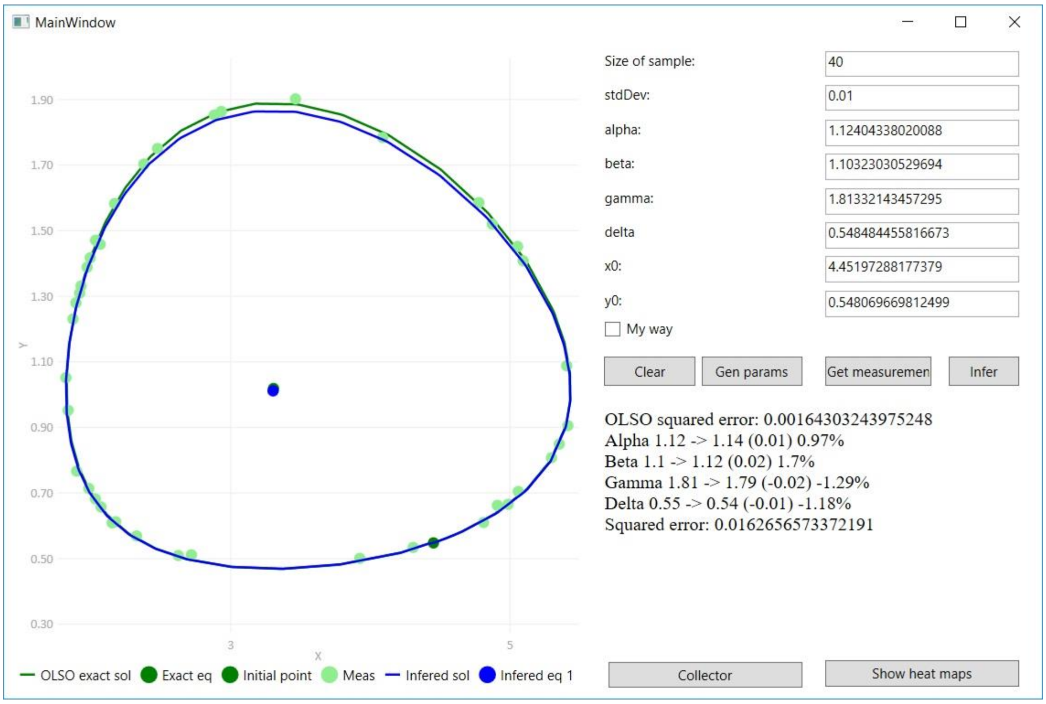
Для описания модели и ее обучения я выбрал программный пакет *Microsoft Infer .NET* – библиотеку, реализующую Байесовские методы машинного обучения, разработанную Microsoft Research. В качестве языка программирования и среды разработки я выбрал C# и Microsoft Visual Studio 2015. Для отрисовки графиков использовал пакет LiveCharts.

В составе решения были написаны 5 проектов:

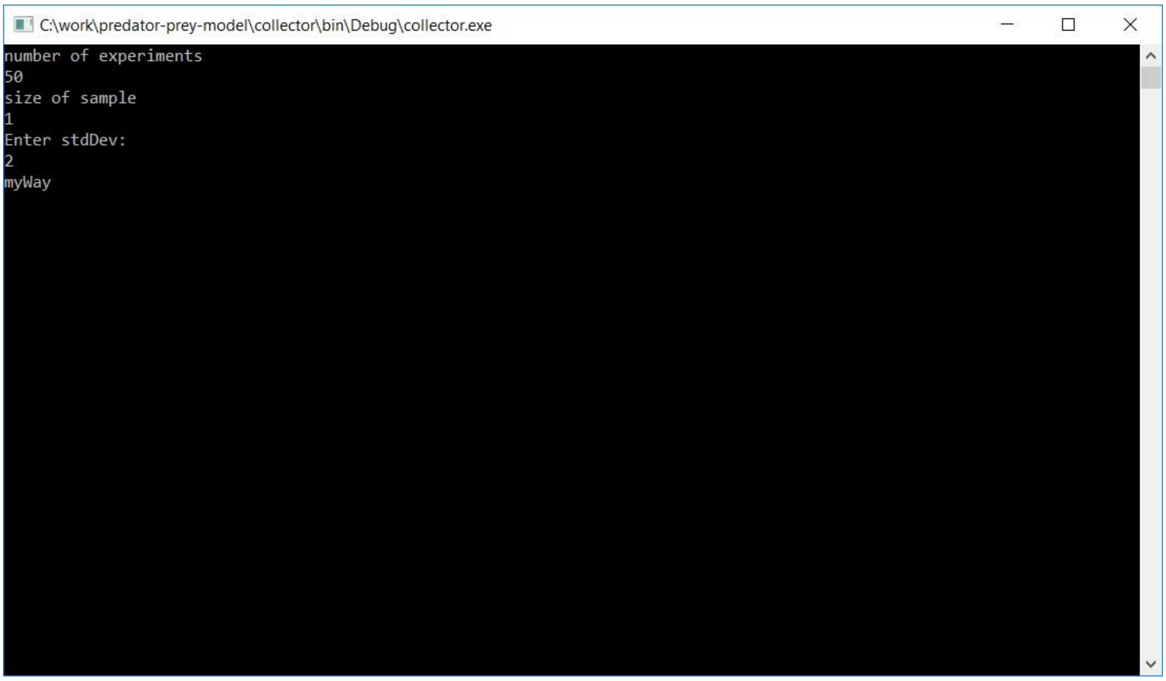
1. User Interface – пользовательский интерфейс, обеспечивающий возможность использования функционала для пользователя и просмотр графиков;



Пользователю предлагается пронаблюдать эксперимент. Ввести исходные параметры системы или сгенерировать их случайно, решить прямую задачу, снять измерения с выбранной погрешностью, решить обратную задачу одним из двух способов. После чего предлагается оценить визуально сходство решений для исходных и приближенных параметров, а также оценить точность их нахождения:



2. Solver – проект, отвечающий за решение прямой дифференциальной задачи;
3. Randomizer – проект, предоставляющий случайные исходные параметры для экспериментов;
4. Predictor – проект, отвечающий за решение обратной задачи;
5. Collector – проект, запускающий серию экспериментов в автоматическом режиме и собирающий их результаты в выходной файл;



Код

Ниже приведен код обучения модели и вывода (первый способ):

```
Variable<double> alpha = Variable.GaussianFromMeanAndVariance(5, 2).Named("alpha");
Variable<double> beta = Variable.GaussianFromMeanAndVariance(5, 2).Named("beta");
Variable<double> gamma = Variable.GaussianFromMeanAndVariance(5, 2).Named("gamma");
Variable<double> delta = Variable.GaussianFromMeanAndVariance(5, 2).Named("delta");
Range dataRange = new Range(points.Count);
VariableArray<double> dxdt = Variable.Array<double>(dataRange);
VariableArray<double> dydt = Variable.Array<double>(dataRange);
VariableArray<double> x = Variable.Array<double>(dataRange);
VariableArray<double> y = Variable.Array<double>(dataRange);
using (Variable.ForEach(dataRange))
{
      dxdt[dataRange] = (alpha - beta * y[dataRange]) * x[dataRange];
      dydt[dataRange] = (-gamma + delta * x[dataRange]) * y[dataRange];
}
double[] xObserved = new double[points.Count];
double[] yObserved = new double[points.Count];
double[] dxdtObserved = new double[points.Count];
double[] dydtObserved = new double[points.Count];
for (int i = 0; i < points.Count; i++)
{
      xObserved[i] = points[i][0];
      yObserved[i] = points[i][1];
}
dxdtObserved[0] = (xObserved[1] - xObserved[0]) / (points[1][2] - points[0][2]);
dydtObserved[0] = (yObserved[1] - yObserved[0]) / (points[1][2] - points[0][2]);
dxdtObserved[points.Count - 1] = (xObserved[points.Count - 1] - xObserved[points.Count - 2]) / (points[points.Count - 1][2] - points[points.Count - 2][2]);
dydtObserved[points.Count - 1] = (yObserved[points.Count - 1] - yObserved[points.Count - 2]) / (points[points.Count - 1][2] - points[points.Count - 2][2]);
for (int i = 1; i < points.Count - 1; i++)
```

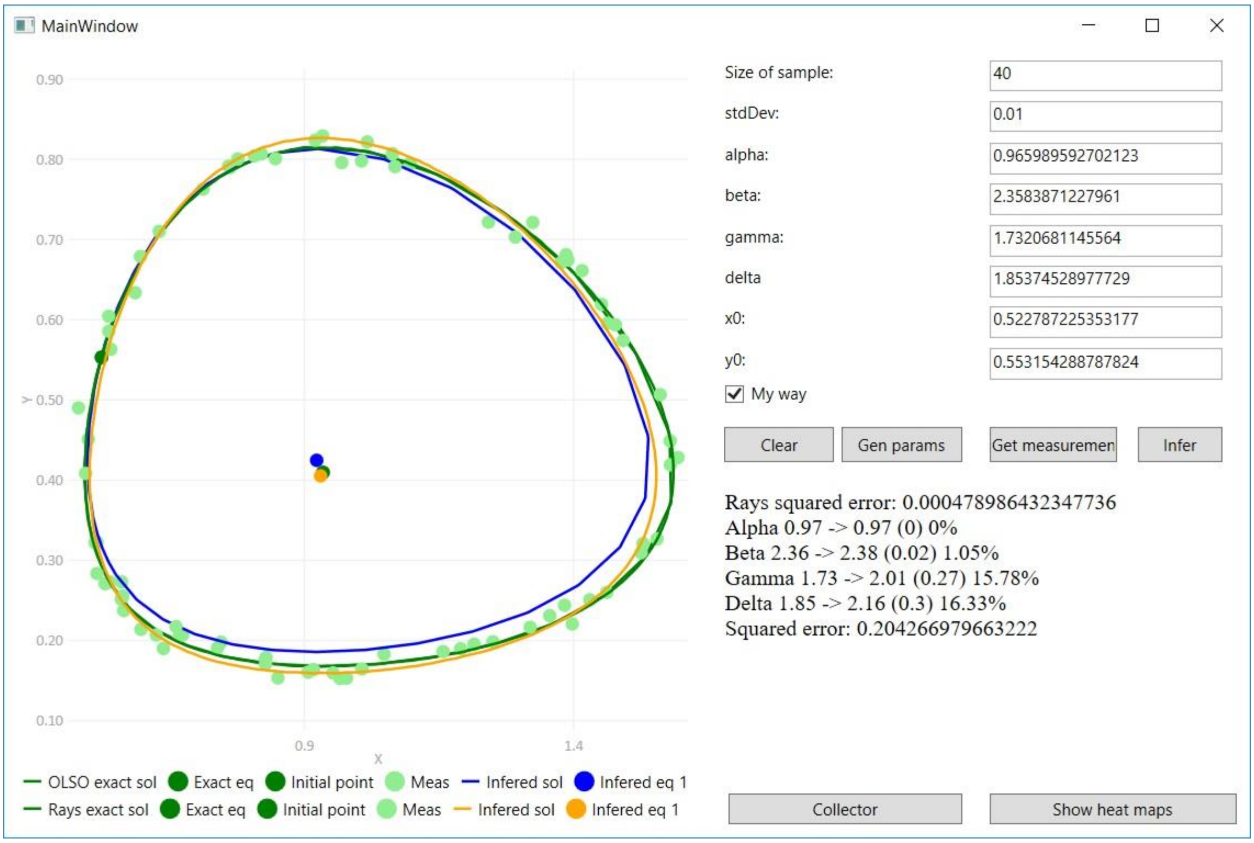
```
{
      dxdtObserved[i] = (xObserved[i + 1] - xObserved[i - 1]) / (points[i + 1][2] - points[i - 1][2]);
      dydtObserved[i] = (yObserved[i + 1] - yObserved[i - 1]) / (points[i + 1][2] - points[i - 1][2]);
}

x.ObservedValue = xObserved;
y.ObservedValue = yObserved;
dxdt.ObservedValue = dxdtObserved;
dydt.ObservedValue = dydtObserved;

InferenceEngine engine = new InferenceEngine(new ExpectationPropagation());
```

Пример работы программы

Сгенерируем случайные параметры и проведем эксперимент.



Темно-зеленым обозначена исходная траектория, её начальная и равновесная точки.

Светло-зеленым обозначены сделанные по ней измерения.

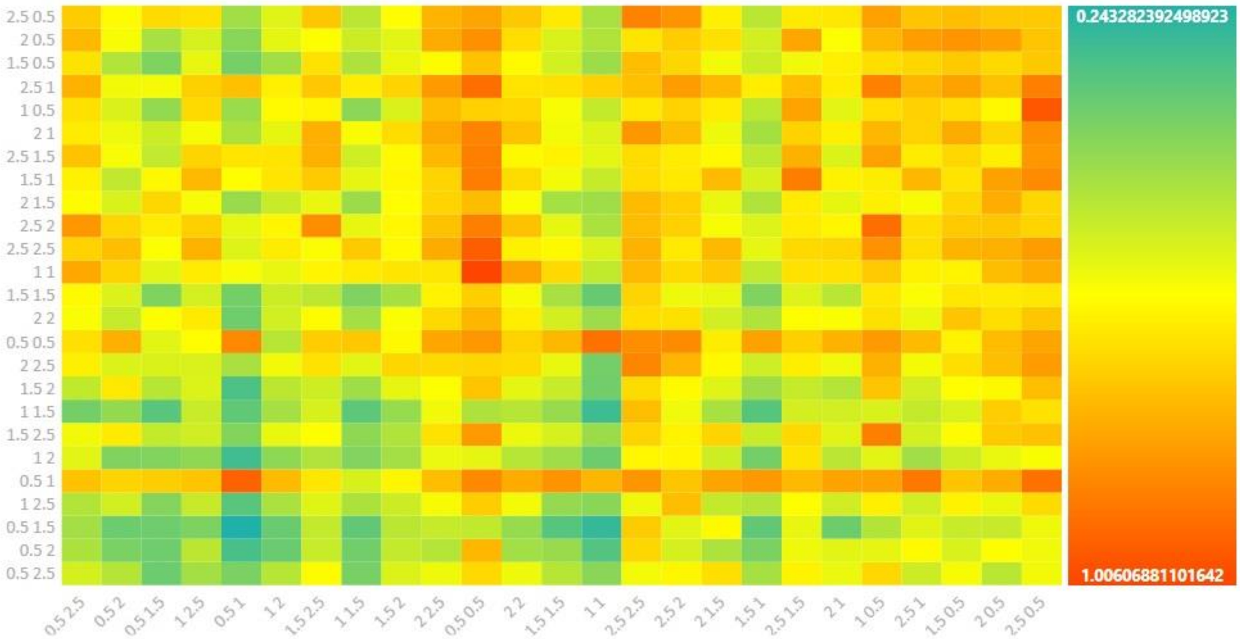
Синим обозначены траектория и её равновесная точка, полученные первым способом.

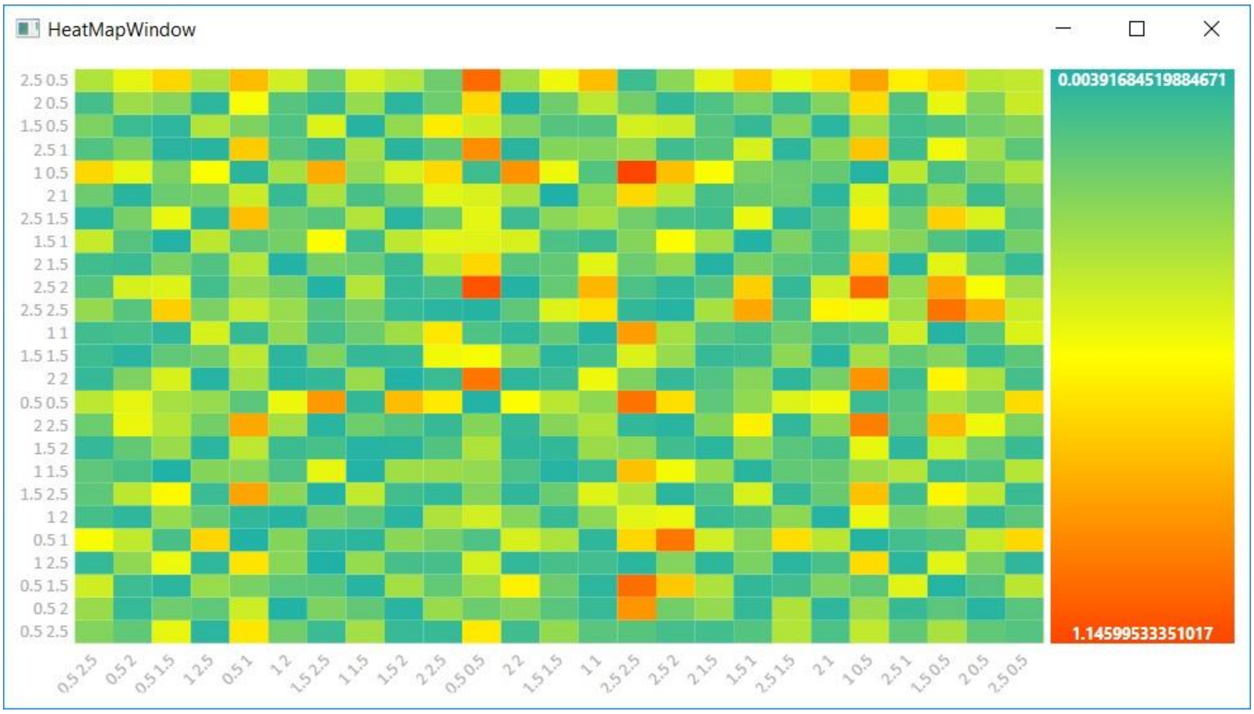
Желтым обозначены траектория и её равновесная точка, полученные вторым способом.

Эксперимент

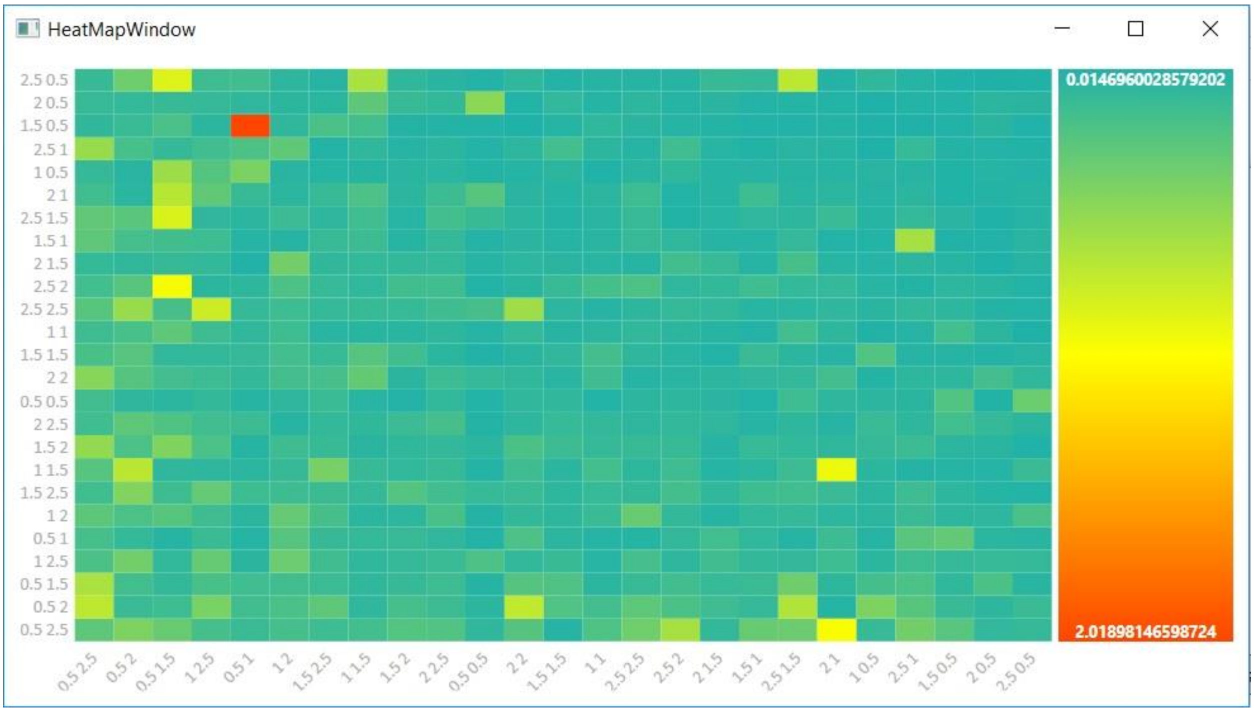
Была проведена серия экспериментов, в ходе которых были получены данные о работе каждого из двух методов для выборки из 5 точек и погрешности измерения с нормальным отклонением при $\sigma = 0{,}001$ и для выборки из 50 точек и погрешности измерения при $\sigma = 0{,}1$. Для каждой комбинации α , $\beta, \gamma, \delta \in [0.5, 1, 1.5, 2, 2.5]$ было проведено 50 экспериментов. На тепловой карте цвет подобран в соответствии с медианой ошибки.

N = 5, stdDev = 0.001 (первый и второй способы, соответственно):





N = 50, stdDev = 0.01 (первый и второй способы, соответственно):



Оценка

Точность приближения параметров напрямую зависит от погрешности измерений. Так при погрешности, представленной нормальным распределением с $\delta = 0{,}001$ оба метода показали точность в пределах 1, а при $\delta = 0{,}01$ уже в пределах 2. Количество точек не помогает с решением задачи. Модель одинаково работает и при 5, и при 50 точках в выборке.

Заключение

По данным измерениям численности популяций видов хищников и жертв мне удалось относительно точно приблизить исходные параметры модели Лотки – Вольтерры.

Реализовано два алгоритма, решающих данную задачу с помощью Байесовских методов машинного обучения.

Список литературы

1. «Хищник и жертва» К. Богданов

2. «Очерки по теории обыкновенных дифференциальных уравнений» Дифференциальные уравнения в биологии, химии, медицине

3. «Model-Based Machine Learning» Christopher Bishop, Thomas Diethe

4. «Байесовские методы машинного обучения» К. Струминский, Д. Молчанов

5. «Bayesian Reasoning and Machine Learning» Barber D.