Применение нейронных сетей в решении задачи факторизации Винера-Хопфа*

*при поддержке Российского фонда фундаментальных исследований,
грант № 23-21-00474

Алымова Елена Владимировна [1]
Кудрявцев Олег Евгеньевич [1, 2]

[1] Российская таможенная академия (Ростовский филиал)
[2] ООО НПФ «ИнВайз Системс»

**Международная научно-техническая конференция
"Актуальные проблемы прикладной математики, информатики и механики“
Воронеж, 04 - 06 декабря, 2023**

Актуальность и цель исследования

Вычисление цен опционов в моделях Леви остается актуальной математической и вычислительной задачей финансовой математики.

Методы вычисления цен барьерных опционов: недостатки

- Методы Монте-Карло: *медленно*
- Конечно-разностные схемы: *необходим детальный анализ процесса Леви в основе модели*
- Методы факторизации Винера-Хопфа: *в общем случае требуются нетривиальные формулы аппроксимации*

Основная цель исследования

Предложить гибридный численный метод вычисления цен барьерных опционов, моделируемых процессом Леви. Основное преимущество метода: аппроксимация факторов Винера-Хопфа с помощью нейронной сети.

Постановка задачи

Пусть задана следующая последовательность чисел:

$$p_0, p_1, \cdots, p_{M-2} \mid p_k \geq 0, \sum_{k=0}^{M-2} p_k = 1, M = 2^N, N \in \mathrm{N}$$

При $\mathrm{k} > \frac{\mathrm{M}-2}{2}$ пос-ть $\mathrm{p_k}$ возрастает, при $\mathrm{k} \leq \frac{\mathrm{M}-2}{2}$ – убывает.

Тогда $\phi(\xi) = \sum_{\mathrm{k}=0}^{\mathrm{M}-2} p_k e^{i\xi\left(k-\frac{M}{2}+1\right)}$ является характеристической функцией дискретной случайной величины X, принимающей значения в точках $x_k = k - \frac{M}{2} + 1$ с вероятностями $\mathrm{p_k}$, $k = 0, \cdots M-2$.

;

Постановка задачи

Задача: представить характеристическую функцию *X* в виде следующего произведения характеристических функций двух независимых дискретных случайных величин, принимающих неотрицательные и неположительные значения:

$$\phi(\xi) = \sum_{k=0}^{\frac{M}{2}} \beta_k e^{i\xi k} \cdot \sum_{k=0}^{\frac{M}{2}-1} \alpha_k e^{-i\xi k}$$

$$\alpha_0, \alpha_1, \ldots, \alpha_{M/2} \mid \alpha_i \geq 0, \sum_{i=0}^{M/2} \alpha_i = 1, 0 \leq \alpha_i < \alpha_{i+1} < 1$$

$$\beta_0, \beta_1, \ldots, \beta_{M/2} \mid \beta_i \geq 0, \sum_{i=0}^{M/2} \beta_i = 1, 1 \geq \beta_{i+1} > \beta_i > 0$$

Задача факторизации полинома

Указанная задача сводится к поиску факторизации

$$p(x) = q(x) \cdot r(x)$$

Где

$$\mathrm{q(x)} = \alpha_0 + \alpha_1 x + \cdots + \alpha_{M/2} x^{M/2-1}$$

$$\mathrm{r(x)} = \beta_0 + \beta_1 x + \cdots + \beta_{M/2} x^{M/2-1}$$

$$\mathrm{p(x)} = p_0 + p_1 x + \cdots + p_{M-1} x^{M-2}$$

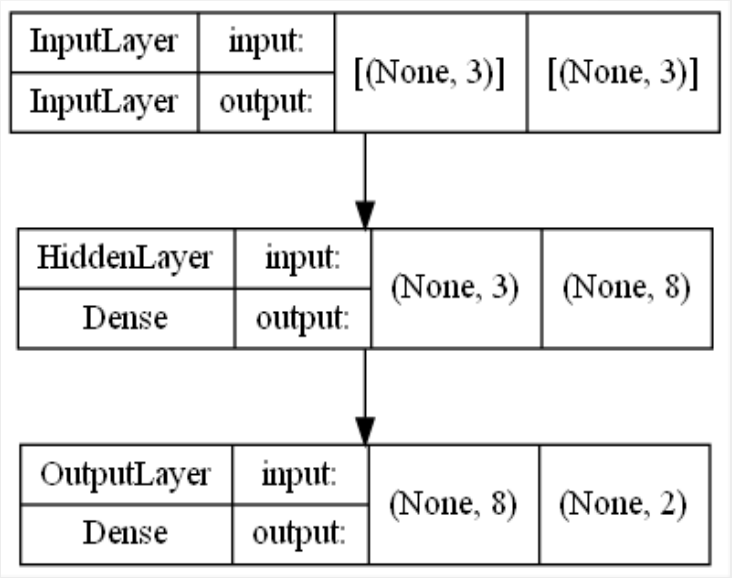
Нейросеть используется для поиска коэффициентов многочленов q и r, произведение которых равно заданному многочлену p

Аппроксимация факторов

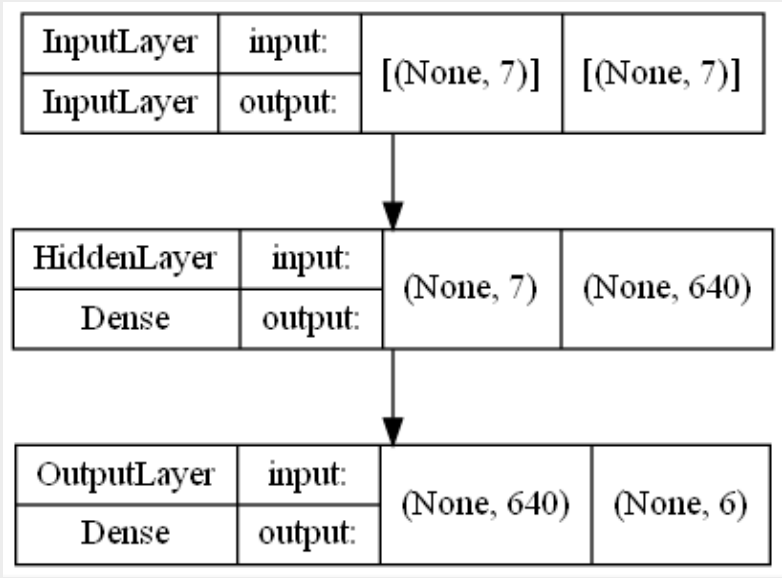
Нейросеть аппроксимирует функцию $F:(p_i)\to(\alpha_j,\beta_k)$

$$\alpha_0,\alpha_1,\ldots,\alpha_{M/2}\mid \alpha_i\geq 0,\ \sum_{i=0}^{M/2}\alpha_i=1,\ 0\leq\alpha_i<\alpha_{i+1}<1$$

$$\beta_0,\beta_1,\ldots,\beta_{M/2}\mid \beta_i\geq 0,\ \sum_{i=0}^{M/2}\beta_i=1,\ 1\geq\beta_{i+1}>\beta_i>0$$



Факторизация полинома 2-й степени (M = 4)



Факторизация полинома 6-й степени (M = 8)

Факторизация полинома 2-й степени

Пусть M = 4

$$\frac{p_0}{x} + p_1 + p_2 x^2 = \left(\frac{\alpha_0}{x} + \alpha_1\right)(\beta_0 + \beta_1 x),$$

$\alpha_i -$ возрастают, β_i убывают $(i = 0, 1)$

$$\alpha_0 = 1 - \alpha_1$$

$$\beta_1 = 1 - \beta_0$$

Разделим и умножим на X:

$$\frac{p_0 + p_1 x + p_2 x^2}{x} = \frac{(\alpha_0 + \alpha_1 x)(\beta_0 + \beta_1 x)}{x}$$

$$p_0 + p_1 x + p_2 x^2 = (\alpha_0 + \alpha_1 x)(\beta_0 + \beta_1 x)$$

Функции генерации обучающих данных

```python
def gen_pair():
    import random as rnd
    alpha1 = rnd.random()
    while alpha1 == 0:
        alpha1 = rnd.random()
    alpha2 = 1 - alpha1
    pair = [alpha1, alpha2]
    pair.sort()
    return pair
```

```python
def gen_factor_coeffs():
    alpha = gen_pair()
    beta = gen_pair()
    beta.sort(reverse=True)
    return alpha, beta
```

```python
def calc_abc(alpha, beta):
    a = alpha[1] * beta[1]
    b = alpha[1] * beta[0] + alpha[0] * beta[1]
    c = alpha[0] * beta[0]
    return [a, b, c]
```

Функция потерь

- Стандартная функция $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y}_i - Y_i\right)^2$
- Собственная функция потерь:

Пусть

$$\mathrm{A} = \left[1-\alpha_1, \alpha_1, 0, 0\right]$$
$$\mathrm{B} = \left[\beta_0, 1-\beta_0, 0, 0\right]$$
$$\mathrm{C} = \left[p_0, p_1, p_2, 0\right]$$

Определим вектор **D** следующим образом:

$$D = \mathrm{FFT}(A) \cdot \mathrm{FFT}(B) - \mathrm{FFT}(C),$$

где FFT() – функция быстрого преобразования Фурье.

Функция потерь: $CLOSS2 = \frac{1}{m}\sum_{i=1}^{m}\left(real(D_i) + imag(D_i)\right)^2$

Факторизация полинома 6-й степени

Пусть M = 8

$$\sum_{i=0}^{6} p_i x^i = \left(\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3\right) \cdot \left(\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3\right)$$

α_i – возрастают, β_i убывают $(i = 0, 3)$

$$\alpha_0 = 1 - \sum_{i=1}^{3} \alpha_i$$

$$\beta_3 = 1 - \sum_{i=0}^{2} \beta_i$$

Функции генерации обучающих данных

```python
def gen_facor_coef_seq(n=999, l=8):
    import random
    randomList = [random.randint(0, n) for x in range(l)]
    mid = len(randomList) // 2 - 1
    while (sum(randomList[mid + 1:]) == 0) or (sum(randomList[:mid + 1]) == 0):
        randomList = [random.randint(0, n) for x in range(l)]
    randomList.sort()
    alphaCoefs = randomList[mid + 1:]
    betaCoefs = randomList[:mid + 1]
    betaCoefs.sort(reverse=True)
    alphaCoefsNorm = [alphaCoefs[i] / sum(alphaCoefs) for i in range(len(alphaCoefs))]
    betaCoefsNorm = [betaCoefs[i] / sum(betaCoefs) for i in range(len(betaCoefs))]
    alphaCoefsNorm[0] = 1 - sum(alphaCoefsNorm[1:])
    betaCoefs[-1] = 1 - sum(betaCoefs[:-1])
    alphaCoefsNorm.sort()
    betaCoefs.sort(reverse=True)
    return alphaCoefsNorm, betaCoefsNorm
```

```python
def calc_abc7_by4(alpha, beta):
    abc7 = [0 for i in range(7)]
    abc7[0] = alpha[0] * beta[0]
    abc7[1] = alpha[0] * beta[1] + alpha[1] * beta[0]
    abc7[2] = alpha[0] * beta[2] + alpha[1] * beta[1] + alpha[2] * beta[0]
    abc7[3] = alpha[0] * beta[3] + alpha[1] * beta[2] + alpha[2] * beta[1] + alpha[3] * beta[0]
    abc7[4] = alpha[1] * beta[3] + alpha[2] * beta[2] + alpha[3] * beta[1]
    abc7[5] = alpha[2] * beta[3] + alpha[3] * beta[2]
    abc7[6] = alpha[3] * beta[3]
    return abc7
```

Функция потерь

Пусть:

$$\mathrm{A} = \left[1 - \sum_{\mathrm{j}=0}^{2} \alpha_{\mathrm{j}}, \alpha_0, \alpha_1, \alpha_2, 0, 0, 0, 0\right]$$

$$\mathrm{B} = \left[\beta_0, \beta_1, \beta_2, 1 - \sum_{\mathrm{k}=0}^{2} \beta_{\mathrm{k}}, 0, 0, 0, 0\right]$$

$$\mathrm{C} = \left[p_0, p_1, p_2, p_3, p_4, p_5, p_6, 0\right]$$

Штраф за нарушение условия возрастания/убывания:

$$\mathrm{PenaltyA} = \left|\min\left(0, \alpha_1 - \alpha_0, \alpha_2 - \alpha_1\right)\right|$$

$$\mathrm{PenaltyB} = \left|\min\left(0, \beta_0 - \beta_1, \beta_1 - \beta_2\right)\right|$$

Функция потерь:

$$CLOSS6 = \frac{1}{m}\sum_{i=1}^{m}\left(real(D_i) + imag(D_i)\right)^2 + \mathrm{PenaltyA} + \mathrm{PenaltyB}$$

Результаты тренировки моделей (степень 2)

Стандартная функция потерь (MSE)

- Быстрое обучение – 75 эпох
- Оценка:

```
31250/31250 [==============================] - 40s 1ms/step - loss: 2.1537e-04
```

- Среднее время предсказания: 0.0420 сек.

Собственная функция потерь:

N – количество нейронов на скрытом слое

Среднее время предсказания: 0.0434 сек.

N	CLOSS2
4	2.62e-05
8	3.45e-06

Результаты тренировки моделей (степень 6)

Собственная функция потерь:

N – количество нейронов на скрытом слое

Среднее время предсказания: 0.0581 сек.

N	CLOSS6
128	7.82e-03
256	6.27e-04
512	1.04e-04
1024	6.31e-06

Основные результаты

- Построены нейронные сети для разложения полиномов 2-й и 6-й степени на полиномы-сомножители.
- Нейросеть предсказывает n-1 коэффициентов, самый маленький коэффициент вычисляется как разность суммы предсказанных коэффициентов с единицей.
- Введена функция штрафов за нарушение условий возрастания/убывания последовательностей коэффициентов.
- Разработанная **функция потерь не зависит от фактических значений коэффициентов**, что позволяет использовать ее для факторизации полиномов больших степеней.