Sigma Fold

Предсказание вторичной структуры белков методами ML

Иктисанов А. В., Галигузов Д. А., Хохлов А. А.

12 декабря 2025 г.

Задача и мотивация

Входные данные:

- Последовательность аминокислот (20 типов)
- Длина: 50 – 1000+ позиций

Выход:

- Метка для каждой позиции:
  - H – спираль
  - E – слой
  - C – петля

Почему это важно?

- Структура $\rightarrow$ функция белка
- Фундамент для 3D-предсказания
- Приложения в медицине и биотехе

Тип задачи:

Последовательная многоклассовая классификация

Методология CRISP-DM

1	2	3	4	5	6
Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment

Преимущества:

- Структурированность и воспроизводимость
- Цикличность (возврат к предыдущим этапам)
- Универсальность (любой ML-проект)

Этап 1-2: Понимание данных

Ноутбук 01: Data Exploration

Анализ:

- **Распределение длин:** диапазон, среднее, медиана
- **Баланс классов:** доля H/E/C в датасете
- **Пропуски:** нулевые значения, аномалии
- **Визуализация:** гистограммы, графики

Результат: определяем max_length для padding'а, тип балансировки, метрики

Этап 3: Подготовка данных

Ноутбук 02: Preprocessing

Кодирование:

- AA: A, C, ..., Y $\rightarrow$ 1, 2, ..., 20
- Labels: H, E, C $\rightarrow$ 0, 1, 2

Padding:

- Макс длина $L_{\max}$
- Дополнение нулями

Маски: бинарный вектор для игнорирования padding'а в loss
Балансировка: веса классов $w_c = \frac{N_{total}}{K \cdot N_c}$
Split: стратифицированное разделение (сохранение пропорций):

$$\text{Train: Val: Test} = 70\% : 15\% : 15\%$$

Результат: готовые train/val/test данные в NumPy

Этап 4а: Базовые методы

Ноутбук 03: Baseline Models

Сравнение трёх классических методов:

Метод	Q3	F1 (avg)	Время
Logistic Regression	0.2870	0.27	$<$ 1 сек
LinearSVC	0.45	0.30	$\sim$ 5 сек
Random Forest	0.4635	0.45	$\sim$ 30 сек

Вывод: Q3 $<$ 50% $\Rightarrow$ нужны методы, учитывающие **контекст**!

Этап 4b: Архитектура Transformer

Идея: каждая позиция должна видеть всю последовательность

Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head: h параллельных heads для разных типов зависимостей

Feed-Forward: нелинейность между attention слоями

Структура:

- Embedding + Positional Encoding
- 6 Transformer блоков
- Классификационный head

Параметры:

- $d_{model} = 128$
- $n_{heads} = 8$
- $\sim 500K$ параметров

Self-Attention механизм

Основная идея:

$$\text{новое представление позиции} = \sum_j \alpha_{ij} \cdot v_j$$

Веса α_{ij} вычисляются как:

$$e_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}} \quad \Rightarrow \quad \alpha_{ij} = \text{softmax}(e_{ij})$$

Интерпретация: позиция i обращает внимание на позиции, чьи ключи близки к её запросу

Пример: при предсказании типа аминокислоты в позиции i:

- Большой вес на соседних позициях (локальный контекст)
- Маленький вес на далёких позициях
- Специальный вес на родственных аминокислотах

Positional Encoding

Проблема: Self-attention перестановочно-инвариантен!

Решение: явно добавить информацию о позициях:

$$PE(i, 2k) = \sin\left(\frac{i}{10000^{2k/d}}\right)$$

$$PE(i, 2k+1) = \cos\left(\frac{i}{10000^{2k/d}}\right)$$

Входное представление:

$$\hat{x}_i = \text{Embedding}(x_i) + PE(i)$$

Преимущества синусоидального кодирования:

- Детерминировано (не требует обучения)
- Экстраполяция на длинные последовательности
- Разные частоты кодируют расстояния на разных масштабах

Обучение Transformer'а

Ноутбук 04: Transformer

Оптимизатор: AdamW с learning rate schedule
Loss: взвешенная кросс-энтропия с маской:

$$\mathcal{L} = -\frac{1}{N_{valid}} \sum_i m_i \sum_c w_c \cdot y_i^c \log \hat{p}_{i,c}$$

ESM-эмбеддинги:

- Вместо случайной инициализации embedding'а
- Использовать предобученные представления
- $\Rightarrow$ лучше качество, меньше требуется данных

Метрики качества

Q3-Accuracy (3-state accuracy):

$$Q3 = \frac{\sum_i m_i \cdot \mathbb{1}(\hat{y}_i = y_i)}{\sum_i m_i} \times 100\%$$

Стандартная в биоинформатике, результаты публикуемых методов: 75–85%

Per-Class F1:

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Confusion Matrix:

$$C = \begin{pmatrix} n_{HH} & n_{HE} & n_{HC} \\ n_{EH} & n_{EE} & n_{EC} \\ n_{CH} & n_{CE} & n_{CC} \end{pmatrix}$$

Показывает, какие классы путаются друг с другом

Сравнение методов

Результаты:

Метод	Q3 (%)	Параметры	Тип
Random Forest	46	$\sim 10K$	Классический ML
Transformer	> 52	$\sim 500K$	Deep Learning

Структура проекта

```
project-root/
    data/processed/      # обработанные данные
    notebooks/
        01_data_exploration.ipynb
        02_preprocessing.ipynb
        03_baseline_models.ipynb
        04_transformer.ipynb
    models/transformer/   # сохранённые веса
    results/metrics/      # метрики и отчёты
    README.md, .gitignore
```

Воспроизводимость:

- Фиксированные сиды (NumPy, PyTorch)
- requirements.txt с версиями
- Все этапы документированы

Ключевые выводы

1. **Методология работает**
   - CRISP-DM обеспечивает структурированный подход
   - Каждый этап производит артефакты для следующего
2. **Контекст критичен**
   - Базовые методы: Q3 $\approx$ 45%
   - Трансформер: Q3 $>$ 52%
3. **Подготовка данных важна**
   - Балансировка классов (веса)
   - Маски для padding'a
   - Стратифицированный split
4. **Масштабируемость**
   - Pre-trained ESM-эмбеддинги улучшают результаты
   - Трансфер обучения работает для биологических данных

Дальнейшие направления

1. **Ensemble методы:** комбинация нескольких трансформеров
2. **Attention visualization:** интерпретация решений модели
3. **Тестирование на CASP/CAMEO:** независимые бенчмарки
4. **Web-сервис:** развёртывание как API для практического использования

GitHub: https://github.com/bert0n0sez/Sigma-Fold