«Сейфуллин оқулары – 18: « Жастар және ғылым – болашаққа көзқарас» халықаралық ғылыми -практикалық конференция материалдары = Материалы международной научно-практической конференции «Сейфуллинские чтения – 18: « Молодежь и наука – взгляд в будущее» - 2022.- Т.I, Ч.IV. - С. 22-27

КЛАССИФИКАЦИЯ БЕЛКОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Кадыркеш С.М., магистрант 2 курса
Казахский агротехнический университет им. С. Сейфуллина, г. Нур-Султан.

Введение.

Хотя гены несут информацию, белки играют главную роль в обеспечении всех функциональных возможностей живого организма. Огромное количество различных белков участвует в каждой функции, которая происходит в клетке. Характеристика белков для выявления их структуры и функции проводится с помощью точных лабораторных экспериментов. С быстрым увеличением количества новых белковых последовательностей эти эксперименты стало трудно проводить, поскольку они дороги, требуют много времени и усилий. Поэтому для классификации белков и прогнозирования их функциональных свойств применяется множество вычислительных методов классификации. С ростом производительности вычислительных методов глубокое обучение играет ключевую роль во многих областях. В последнее время были представлены новые модели машинного обучения, такие как BLSTM, ProtCNN, для классификации белков по семействам. В этой статье будет рассмотрено реализация модели ProtCNN на популярном наборе данных Pfam, т. к. авторы этой модели не представили практическую реализацию. Модель достигла точности 98,77% для классификации семейства белков.

Белки являются основным функциональным органом живых организмов. Внутри клеток существует большое количество белков, участвующих в различных уникальных функциях, таких как рост и поддержание, вызывание биохимических реакций, выполнение функций посыльного, обеспечение структуры и защиты. Поэтому понимание функциональности белков необходимо для различных областей, таких как разработка лекарств и выявление заболеваний. В соответствии с основными функциональными функциями, выполняемыми белками, их можно разделить на различные группы, такие как структурные, сократительные, транспортные, ферментные, накопительные, гормональные и защитные [1]. С тех пор, как появились геномные проекты и технологический прогресс, количество известных новых белковых последовательностей быстро увеличилось. Поэтому в базах данных имеется огромное количество не охарактеризованных белков. Согласно статистике 2020 года, в Uniprot насчитывается около 175 миллионов белковых последовательностей, длина

полипептидных цепей которых варьируется от 6 до 37 000 аминокислотных остатков [2].

Для точного определения структуры и функции белков используются такие биологические эксперименты, как рентгеновская кристаллография или ядерный магнитный резонанс [8]. Этим экспериментальным методам трудно справиться с быстро растущим огромным количеством новых белковых последовательностей, поскольку биологические лабораторные эксперименты дороги, отнимают много времени и сил. Поэтому с развитием компьютерных программ и аппаратных средств для определения характеристик белков стали использоваться вычислительные методы [9].

Методы машинного обучения широко применяются для классификации белков. Среди других методов классификации белков Наивный Байес — это вероятностный классификатор, основанный на теореме Байеса, используемый для классификации белков. Классификатор Наивного Байеса с отбором признаков был использован для классификации белков фаговых вирионов (фаг — это вирус, который использует бактерии в качестве хозяев) с использованием эталонного набора данных из 307 последовательностей. Этот эталонный набор включает 99 последовательностей белков вириона фага и 208 последовательностей невирионных белков фага из базы данных UniProt [10]. Этот метод достиг точности 79,6% при использовании метода тестирования "джек-нож", чем другие известные традиционные методы, используемые для сравнения производительности, такие как Random Forest и Support vector machine. В исследованиях по классификации белков также использовалось дерево Наивного Байеса для многоуровневой классификации G-белок-связанных рецепторов (GPCR) [3]. Их результаты показывают более высокую точность по сравнению с другими методами классификации. Однако Наивный Байес предполагает, что все атрибуты в наборе данных взаимно независимы и равны, и этот метод также будет страдать от чрезмерной чувствительности, если в наборе данных есть избыточные или нерелевантные атрибуты.

В другой недавней работе глубокая CNN [25] также была обучена классифицировать 521 527 последовательностей из базы данных Uniprot с 698 семействами меток классов, показав точность AUC 99,99% [26]. Она состоит из 6 конволюционных слоев и 2 полностью связанных слоев с почти 1 миллионом общих параметров сети. Однако более глубокие нейронные сети с большим количеством параметров дают более высокую точность, но при этом приводят к избыточной подгонке, если количество параметров слишком велико. ProtCNN [17] (одиночная глубокая сверточная нейронная сеть) - недавний подход, который использует остаточные сети [27] на основе глубокой сверточной архитектуры для классификации белковых последовательностей из полного набора данных Pfam [6]. В этой работе они сравнили производительность ProtCNN с профильными HMM и BLASTp [28] на эталонном наборе данных Pfam seed dataset. Их модель превзошла BLASTp, в 200 раз быстрее, чем BLAST, обучив 80% набора данных Pfam [6], а также имела меньшую ошибку.

Для дальнейшей оценки эффективности модели ProtCNN по сравнению с методами, которые были опробованы с использованием набора данных Pfam [6], из-за ограниченной вычислительной мощности использовали 1000 семейств с соответствующей максимальной длиной последовательности 100 и 200 (см. таблицу 1).

Источник набора данных	**Имя набора данных**	**Кол-во семейств**	**Кол-во последовательностей**		**Кол-во последовательностей по семейству (минимум)**
Pfam Database	Pfam seed random split	1000	**Тренировочный (80%)**	**Валидация (10%) Тест (10%)**	200
			439493	54378	

Таблица 1. Подробная информация о наборе данных

Предобработка данных.

Человек способен работать с категориальными данными напрямую, поскольку его мозг может очень быстро извлекать признаки и классифицировать их с помощью запоминания. Для сетей глубокого обучения наборы данных необходимо преобразовать в понятный для сети формат без потери информации об исходном формате. В данной исследовательской работе мы используем подход, схожий с упорядоченным одношаговым кодированием [17,18,18], для преобразования выровненной последовательности белков в числовые векторы. Использовалось 3 шага при кодировании исходной последовательности на вход модели.

1. Вставка последовательности аминокислот. На первом этапе мы подготовили длину последовательности к фиксированному значению: 1000 для ввода в сеть, так как каждый белок имеет различную длину, которая в основном меньше 1000 [35]. Например, когда есть аминокислотная последовательность с 650 аминокислотами, ее длина преобразуется в 1000 путем добавления любого специального символа, такого как знак подчеркивания "_", который не является аминокислотой.
2. Составление таблицы аминокислот с цифровыми кодами: В данной работе мы используем список наименований аминокислот IUPAC [36], чтобы предоставить каждой аминокислоте порядковый номер.
3. Представление каждой аминокислоты в виде вектора в двумерном пространстве: берем ось x как упорядоченные коды аминокислот, а ось y - как аминокислоты в последовательности, как показано на рис. 1.

Использовали не аминокислотные позиции как нули, а аминокислотные позиции как единицы.

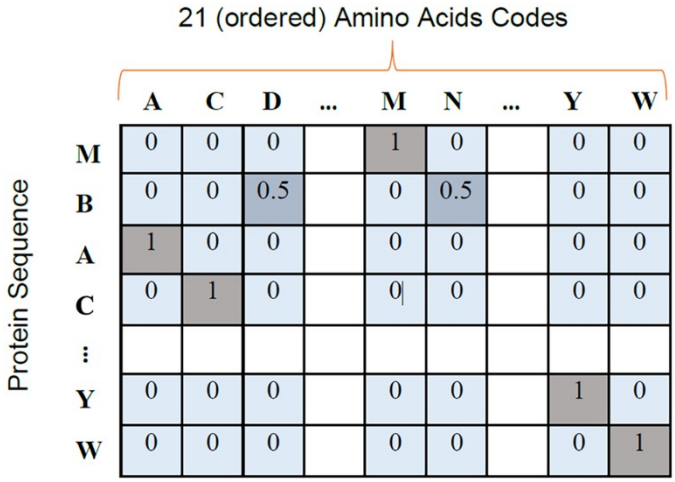

Рис 1. *Представление последовательности в виде матрицы (двумерного массива) после кодирования исходной аминокислотной последовательности.*

Реализация модели на Python.

Последовательности аминокислот преобразуются в одну кодировку с формой (batch_size, 100, 21) в качестве входных данных для модели. Начальная операция свертки применяется к входу с размером ядра 1 для извлечения основных свойств. Затем два идентичных остаточных блока используются для захвата сложных паттернов в данных, вдохновленных архитектурой ResNet, что поможет нам обучить модель с большим количеством эпох и с лучшей производительностью модели.

Определил остаточный блок, немного отличающийся от сети ResNet. Вместо выполнения трех сверток используются только две свертки с добавлением еще одного параметра (скорости расширения) к первой свертке, чтобы иметь более широкое поле зрения при одинаковом количестве параметров модели.

```
def residual_block(data, filters, d_rate):
  """
  _data: input
  _filters: convolution filters
  _d_rate: dilation rate
  """

  shortcut = data

  bn1 = BatchNormalization()(data)
  act1 = Activation('relu')(bn1)
  conv1 = Conv1D(filters, 1, dilation_rate=d_rate, padding='same', kernel_regularizer=l2(0.001))(act1)

  #bottleneck convolution
  bn2 = BatchNormalization()(conv1)
  act2 = Activation('relu')(bn2)
  conv2 = Conv1D(filters, 3, padding='same', kernel_regularizer=l2(0.001))(act2)

  #skip connection
  x = Add()([conv2, shortcut])

  return x
```

Рис 3. Определение остаточного блока

Каждая операция свертки в остаточном блоке следует базовому шаблону BatchNormalization => ReLU => Conv1D. В остаточном блоке первая

свертка выполняется с размером ядра 1x1 со скоростью расширения, а вторая операция свертки выполняется с большим размером ядра 3x3.

Наконец, после применения операций свертки формируется пропущенное соединение путем добавления начального ввода(ярлыка) и вывода из примененных операций свертки.

После двух остаточных блоков применяется максимальное объединение для уменьшения пространственного размера представления. Для регуляризации добавляется отсев, чтобы предотвратить чрезмерную подгонку модели.

```
# model

x_input = Input(shape=(100, 21))

#initial conv
conv = Conv1D(128, 1, padding='same')(x_input)

# per-residue representation
res1 = residual_block(conv, 128, 2)
res2 = residual_block(res1, 128, 3)

x = MaxPooling1D(3)(res2)
x = Dropout(0.5)(x)

# softmax classifier
x = Flatten()(x)
x_output = Dense(1000, activation='softmax', kernel_regularizer=l2(0.0001))(x)

model2 = Model(inputs=x_input, outputs=x_output)
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model2.summary()
```

Рис 4. Эта модель обучается с 10 эпохами, batch_size 256 и проверяется на данных валидации.

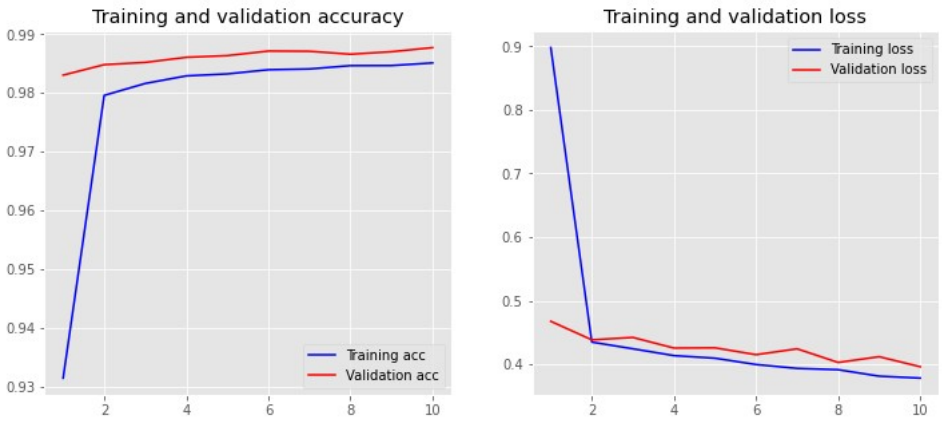

Рис 5. Результаты точности обучения и валидации модели ProtCNN в зависимости от количества эпох

В этой статье все вычислительные эксперименты проводились с помощью Google CoLab Pro с поддержкой GPU Tesla. Модель была реализована на Python с использованием библиотеки TensorFlow Keras. Для визуализации метрик по эпохам использовался GNU Plot.

Заключение.

В этой статье мы исследовали модель машинного обучение ProtCNN, которая изучает взаимосвязь между несогласованными аминокислотными последовательностями и их функциональными аннотациями. Классификация белковых семейств очень важна для прогнозирования функции белка, разработки лекарств и выявления заболеваний. В классификации белковых семейств модели, основанные на машинном обучении, достигли более высокой точности. Модель достигла значительных результатов, которые являются более точными и вычислительно эффективными, чем современные методы, такие как BLASTp для аннотирования белковых последовательностей. Эти результаты предполагают, что модели машинного обучения станут основным компонентом будущих инструментов прогнозирования функции белка.

Список используемой литературы

1. Buxbaum E. Fundamentals of Protein Structure and Function 2007. 1–367 p.
2. Levitt M. Nature of the protein universe. Proceedings of the National Academy of Sciences. 2009;106(27):11079. pmid:19541617
3. Davies MN, Secker A, Freitas AA, Mendao M, Timmis J, Flower DR. On the hierarchical classification of G protein-coupled receptors. Bioinformatics. 2007;23(23):3113–8. pmid:17956878
4. Andreeva A, Kulesha E, Gough J, Murzin AG. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. Nucleic Acids Res. 2020;48(D1):D376–D82. pmid:31724711
5. Galperin MY, Makarova KS, Wolf YI, Koonin EV. Expanded microbial genome coverage and improved protein family annotation in the COG database. Nucleic Acids Res. 2015;43(Database issue):D261–9. pmid:25428365
6. El-Gebali S, Mistry J, Bateman A, Eddy SR, Luciani A, Potter SC, et al. The Pfam protein families database in 2019. Nucleic Acids Res. 2019;47(D1):D427–D32. pmid:30357350
7. UniProt C. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019;47(D1):D506–D15. pmid:30395287
8. Szymczyna BR, Taurog RE, Young MJ, Snyder JC, Johnson JE, Williamson JR. Synergy of NMR, computation, and X-ray crystallography for structural biology. Structure. 2009;17(4):499–507. pmid:19368883
9. Shehu A, Nussinov R. Computational Methods for Exploration and Analysis of Macromolecular Structure and Dynamics. PLoS Comput Biol. 2015;11(10):e1004585. pmid:26505479
10. Min S, Lee B, Yoon S. Deep learning in bioinformatics. Briefings in Bioinformatics. 2016;18(5):851–69.
11. Paliwal K, Lyons J, Heffernan R. A Short Review of Deep Learning Neural Networks in Protein Structure Prediction Problems. Advanced Techniques in Biology & Medicine. 2015;03.

12. Bileschi ML, Belanger D, Bryant D, Sanderson T, Carter B, Sculley D, et al. Using Deep Learning to Annotate the Protein Universe. bioRxiv. 2019:626507.
13. More AS, Rana DP, editors. Review of random forest classification techniques to resolve data imbalance. 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM); 2017 5–6 Oct. 2017.
14. Hou J, Adhikari B, Cheng J. DeepSF: deep convolutional neural network for mapping protein sequences to folds. Bioinformatics (Oxford, England). 2018;34(8):1295–303. pmid:29228193
15. Söding J. Protein homology detection by HMM-HMM comparison. Bioinformatics. 2005;21(7):951–60. pmid:15531603
16. Carter B, Bileschi M, Smith J, Sanderson T, Bryant D, Belanger D, et al. Critiquing Protein Family Classification Models Using Sufficient Input Subsets. bioRxiv. 2019:674119. pmid:31874057