Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Наименование школы: ИШИТР
Наименование отделения: ОИТ
Наименование направления: Искусственный интеллект и машинное обучение

Отчёт по Нейроэволюционные вычисления

Выполнил: студент гр. 8ВМ22 Цяо Иньсюань (Подпись) Цяо Иньсюань (Ф.И.О.)

Проверил: Старший преподаватель
ОИТ ИШИТР _____ (Подпись) Григорьев Д. С. (Ф.И.О.)

Томск 2023

Нейроэволюционные вычисления

Григорьев Дмитрий Сергеевич2023

1 Задание

Необходимо реализовать алгоритм согласно варианту.

2 Общие требования к выполнению работы

1. Требуется реализовать сохранение/загрузку весов связей/структуры сети в отдельном фай- ле;
2. По результатам работы отчет должен содержать описание используемого алгоритма, эта- пы имплементации, визуальное отображение структуры сети на разных эпохах, описание целеввых метрик, графики сходимости (зависимость целевой метрики от количества эпох);
3. Необходимо использовать систему контроля версий;
4. Допускается использование сторонних наборов данных или из архива (Data.rar):

(a)	cancer1	(f)	diabetes2
(b)	diabetes1	(g)	glass2
(c)	diabetes1	(h)	cancer3
(d)	glass1	(i)	diabetes3
(e)	cancer2	(j)	glass3

5. Допускается использование сторонних библиотек.

3 Баллы

Всего за выполнение работы предусмотрено 55 баллов:

1. Код, отчет, ответы на вопросы - 35
2. Коммиты - 5
3. Собственная реализация НЭ алгоритма в соответствии с вариантом - 10
4. Отчет в формате LaTeX- 5

4 Варианты

Вариант выбирается в соответствии с порядковым номером в группе.

Вариант	Алгоритм	Топология
1.	NEAT	Без ограничений на структуру
2.	SANE	Полносвязная 1 Скрытый слой
3.	ESP	Полносвязная Рекуррентная 1 скрытый слой
4.	NEAT	Прямого распространения 2 скрытых слоя
5.	SANE	Полносвязная 1 скрытый слой
6.	H-ESP	Полносвязная 1 Скрытый слой
7.	HyperNEAT	Прямого распространения 1 скрытый слой
8.	ESP	Полносвязная рекуррентная 1 скрытый слой
9.	SANE	Полносвязная 1 скрытый слой
10.	NEAT	Прямого распространения 2 скрытых слоя
11.	SANE	Полносвязная 1 скрытый слой
12.	NEAT	Без ограничений на структуру
13.	H-ESP	Полносвязная Рекуррентная 1 скрытый слой
14.	HyperNEAT	Прямого распространения 2 скрытых слоя
15.	SANE	Прямого распространения 1 скрытый слой
16.	H-ESP	Полносвязная Рекуррентная 1 скрытый слой
17.	ESP	Прямого распространения 1 скрытый слой
18.	ESP	Прямого распространения 1 скрытый слой
19.	SANE	Прямого распространения 1 скрытый слой
20.	NCL	Без ограничений на структуру
21.	NCL	Без ограничений на структуру
22.	SANE	Прямого распространения 1 скрытый слой
23.	H-ESP	Полносвязная Рекуррентная 1 скрытый слой
24.	HyperNEAT	Прямого распространения 2 скрытых слоя
25.	NEAT	Без ограничений на структуру
26.	SANE	Прямого распространения 1 скрытый слой
27.	NCL	Без ограничений на структуру
28.	ESP	Полносвязная 1 Скрытый слой

Цель работы –Реализация структурной оптимизации нейронных сетей на основе алгоритма SANE (Symbiotic Adaptive NeuroEvolution). Алгоритм SANE оптимизирует структуру нейронной сети с помощью кроссинговера и мутации генов, тем самым улучшая производительность сети. В приведенном коде с помощью алгоритма SANE была обучена нейронная сеть с двумя скрытыми слоями, с размерностью входного слоя X_train.shape[1] и размерностью выходного слоя 1. В процессе итеративной оптимизации была записана структура сети на разных эпохах и визуализированы веса первого скрытого слоя. Затем лучшая модель, полученная в результате обучения, использовалась для построения графика сходимости на обучающем и проверочном множествах, показывая изменение точности в процессе обучения. Наконец, были рассчитаны и выведены показатели точности на тестовом наборе.

Ход работы

1. Описание:

Алгоритм SANE (Symbiotic Adaptive NeuroEvolution) - это метод обучения, сочетающий нейронную сеть с эволюционным алгоритмом. Он использует эволюционные алгоритмы для оптимизации структуры и весов нейронной сети в соответствии с требованиями конкретной задачи. Алгоритм сочетает генетические алгоритмы и обучение нейронной сети для оптимизации структуры и весов нейронной сети посредством эволюционного и адаптивного процесса.

Предложен Дэвидом Мориарти . Представляет вариант коэволюционного алгорит-

ма для эволюции весов и структуры ИНС[1].

Используется для ИНС прямого распространения с 1 скрытым слоем.

Хромосома — список связей одного нейрона скрытого слоя. Информация о связи:

Мегка нейрона (8 бит)

Вес связи (16 бит)

Определение входного/выходного нейрона

— Метка > 127 - нейрон выходной

— Индекс нейрона: метка шоа №

— M - количество входных или выходных нейронов (в зависимости от значения метки)

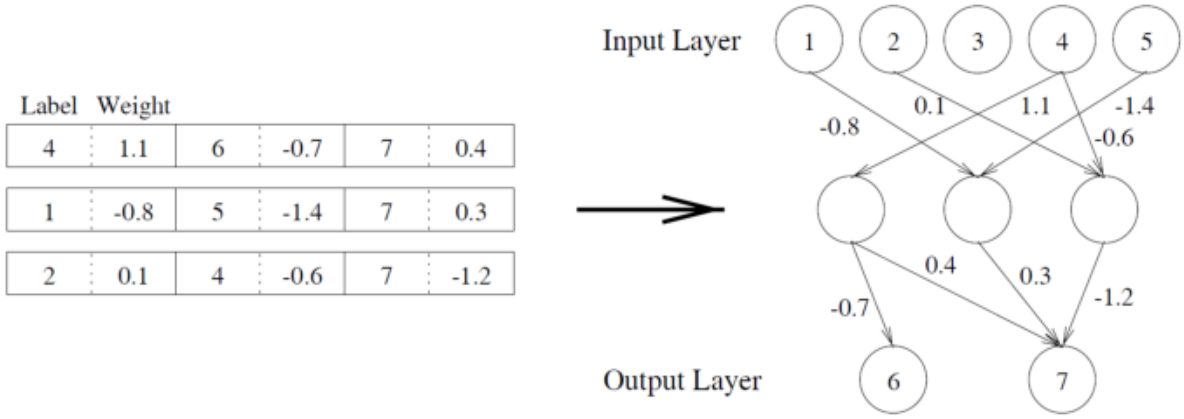

Рис. 1 Кодирование информации 0б ИНС в алгоритме SANE.

Т.к. вся информация кодируется в бинарном виде, то для скрещивания и мутации нейронов применяются 1-точечный кроссинговер и битовая мутация.

Алгоритм — коэволюционный. Информация об ИНС формируется из нескольких нейронов (Рис. 1).

Количество скрытых нейронов фиксированное и должно определяться заранее. Количество связей у каждого нейрона также фиксировано. Эти два параметра (количество нейронов и количество связей) и определяют свойства структуры ИНС.

Структура алгоритма SANE - стандартая для эволюционного алгоритма. Работа

одного поколения представлена алгоритмом В.Т

Анализ разнообразия весов связей

Для оценки результатов обучения нейроно рассматривалось также и разнообразие весов связей. Это тем более важно, что популяция была одна и никакого искусственного разграничения нейронов в этой популяции не было предусмотрено. Т.е. в стандартном эволюционном алгоритме веса нейронов скорее всего сошлись бы к одному и тому же вектору. Однако, благодаря использованию популяции комбинаций в популяции нейронов присутствовало разнообразие .
Анализ весов связей производился с применением метода главных компонент, т.е. веса проецировались на плоскость двух собственных векторов для матрицы ковариации весов с максимальными собственными числами. Пример распределение проекций на последнем поколении эволюции показан на рис. Из рисунка видно, что можно выделить кластеры нейронов, что говорит о появлении аналога специализации для отдельных нейронов. В результате этой специализации нейроны выполняют различные функции и в успешную ИНС должны быть включены нейроны с разной специализацией.

2. **набор данных:** набор данных был взят из набора данных Breast Cancer Wisconsin в репозитории UCI Machine Learning Repository. Целью является выполнение задачи бинарной классификации. В частности, используется алгоритм SANE для оптимизации структуры нейронной сети для прогнозирования классификации доброкачественных и злокачественных опухолей молочной железы на заданном наборе данных по раку молочной железы. Набор данных, используемый в коде, представляет собой набор данных по раку молочной железы

(breast-cancer.csv), который содержит характеристики опухолей молочной железы (например, радиус, текстура, окружность образования и т.д.) и соответствующие диагнозы (M для злокачественных опухолей и B для доброкачественных опухолей). Прогнозирование классификации рака молочной железы с помощью нейронной сети, оптимизированной алгоритмом SANE.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_worst
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54

5 rows × 32 columns

```
1 df.columns
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

Рис.2 Индекс имени столбца данных

3. этапы реализации:

1) Инициализация популяции: случайным образом генерируется начальный набор особей нейронной сети, каждая из которых представлена набором весов связей и топологией.

```
def create_population(self):
    population = []
    for i in range(self.population_size):
        model = Sequential()
        model.add(Dense(16, input_shape=self.input_shape, activation='relu'))
        model.add(Dense(16, activation='relu'))
        model.add(Dense(self.output_shape, activation='sigmoid'))
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
        model.build((None, self.input_shape))
        population.append(model)
    return population
```

В приведенном выше коде метод create_population используется для создания начальной популяции нейронной сети. Он выполняет цикл population_size несколько раз, каждый раз создавая модель нейронной сети. Модель построена с использованием класса Sequential и имеет входной слой, скрытый слой из 16 нейронов и выходной слой. Функции активации - ReLU и Sigmoid, функция потерь - бинарная перекрестная энтропия, а оптимизатор - Adam. Затем модель строится с помощью метода build и добавляется в популяцию.

Этот код гарантирует, что каждый индивид в начальной популяции имеет одинаковую структуру сети, но его веса инициализируются случайным образом. В ходе итераций алгоритма SANE особи в популяции постепенно оптимизируют свою структуру и веса посредством операций отбора, кроссинговера и мутации для улучшения производительности задачи классификации рака молочной железы.

2) Оценка пригодности: каждая особь используется для оценки задачи, вычисляется ее значение пригодности и используется для измерения производительности особи.

```
def get_fitness_scores(self, X_train, y_train):
    scores = []
    for model in self.population:
        model.fit(X_train, y_train, epochs=1, verbose=0)
        score = model.evaluate(X_train, y_train, verbose=0)[1]
        scores.append(score)
    return scores
```

“get_fitness_scores” Метод используется для оценки фитнес-балла каждой особи в популяции. Он итерирует каждую модель нейронной сети в популяции и обучает ее один раз (используя x_train и y_train в качестве входных данных). Во время обучения эпохи устанавливаются на 1, что означает, что выполняется только одна итерация обучения. Затем с помощью метода evaluate вычисляется точность модели на обучающем множестве, и эта точность добавляется в список оценок в качестве оценки адаптации. Наконец, возвращается список оценок приспособленности для всех особей.

В каждом поколении алгоритма SANE фитнес-оценки всех особей в текущей популяции получаются путем вызова метода get_fitness_scores. Эти оценки используются для выбора лучших особей в качестве родителей для операций кроссинговера и мутации, в результате чего формируется следующее поколение популяций.

3) Операция отбора: на основе значения пригодности определенное количество особей выбирается в качестве родителей для воспроизводства следующего поколения.

```
def select_parents(self):
    fitness_scores = np.array(self.fitness_scores)
    fitness_scores = fitness_scores / np.sum(fitness_scores)  # Normalization
    parent_indices = np.random.choice(range(self.population_size), size=2, replace=False,
p=fitness_scores)
    parent1 = self.population[parent_indices[0]]
    parent2 = self.population[parent_indices[1]]
    return parent1, parent2
```

В приведенном выше коде метод select_parents используется для выбора родительских особей для операции кроссинговера. Сначала показатели приспособленности нормализуются так, чтобы сумма показателей была равна 1. Затем два неповторяющихся индекса из диапазона от 0 до population_size-1 случайным образом выбираются в качестве индексов родителей с помощью метода np.random.choice. В процессе выбора вероятность выбора родительской особи определяется на основе нормализованного показателя пригодности, при этом особи с более высокими показателями имеют большую вероятность быть выбранными. Наконец, на

основе индекса выбранного родителя из популяции извлекаются и возвращаются соответствующие родительские особи.

В каждом поколении алгоритма SANE родительские особи выбираются для кроссинговера путем вызова метода select_parents. Выбранные родительские особи будут использоваться для генерации новых дочерних особей.

4) Вариационные операции: вариационные операции над родительскими особями, включая корректировку весов, добавление/удаление связей или нейронов и т.д.

```
def mutate(self, model):
    weights = model.get_weights()
    for i, w in enumerate(weights):
        if len(w.shape) == 1:  # Bias invariant
            continue
        mask = np.random.rand(*w.shape) < self.mutation_rate
        weights[i] = np.where(mask, w + np.random.normal(size=w.shape), w)
    model.set_weights(weights)
    return model
```

Для выполнения мутации особи используется метод "mutate". Сначала получают весовые матрицы особей. Затем для каждой весовой матрицы определяется форма, если она одномерная (смещение без мутации). Если весовая матрица двумерна, то генерируется маска той же формы, что и весовая матрица, на основе скорости мутации "mutation_rate". Каждый элемент маски - случайное число от 0 до 1, причем элементы, меньшие, чем "mutation_rate " элементы будут установлены в True, указывая на то, что веса в соответствующей позиции должны быть мутированы. Затем с помощью "np.random.normal" генерируется матрица случайных чисел той же формы, что и матрица весов, и добавляется к исходной матрице весов для получения мутировавшей матрицы весов. Наконец, мутировавшая матрица веса сбрасывается на индивидуума, и возвращается мутировавший индивидуум.

В каждом поколении алгоритма SANE операции мутации выполняются над дочерними особями путем вызова метода mutate для введения новых генетических вариантов.

5) Операции кроссинговера: операции кроссинговера выполняются над мутировавшими особями для создания нового потомства.

```
def crossover(self, parent1, parent2):
    child = Sequential()
    child.add(Dense(16, input_shape=self.input_shape, activation='relu'))
    child.add(Dense(16, activation='relu'))
    child.add(Dense(self.output_shape, activation='sigmoid'))

    weights1 = parent1.get_weights()
    weights2 = parent2.get_weights()

    new_weights1, new_weights2 = self.adjust_weights_shape(weights1, weights2)

    child_weights = []
    for w1, w2 in zip(new_weights1, new_weights2):
```

```
        mask = np.random.choice([0, 1], size=w1.shape)  # Случайный выбор соответствующих весов родителей
        child_weights.append(np.where(mask, w1, w2))

    child.set_weights(child_weights)
    child.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    child.build((None, self.input_shape))
    return child

def adjust_weights_shape(self, weights1, weights2):
    """
    Adjusts the shape of the weight arrays to ensure they have the same dimensions
    """
    new_weights1, new_weights2 = [], []
    for w1, w2 in zip(weights1, weights2):
        if w1.shape != w2.shape:
            # adjust the shape of w1
            tmp_w1 = np.zeros(w2.shape)
            tmp_w1[:w1.shape[0], :w1.shape[1]] = w1
            new_weights1.append(tmp_w1)

            # adjust the shape of w2
            tmp_w2 = np.zeros(w1.shape)
            tmp_w2[:w2.shape[0], :w2.shape[1]] = w2
            new_weights2.append(tmp_w2)
        else:
            new_weights1.append(w1)
            new_weights2.append(w2)

    return new_weights1, new_weights2
```

В приведенном выше коде метод кроссинговера используется для скрещивания двух родительских особей для создания дочерних особей. Сначала создается пустая модель ребенка. Затем получаются веса родительских особей, weights1 и weights2, и вызывается метод adjust_weights_shape для корректировки весов, чтобы убедиться, что они имеют одинаковую размерность. Если две весовые матрицы не имеют одинаковой формы, они приводятся к одинаковой форме с помощью нулевых прокладок. Наконец, скорректированные матрицы весов пересекаются. Для каждой весовой матрицы в соответствующей позиции генерируются дочерние веса путем случайного выбора соответствующих весов родителя. Используя функцию np.where, веса родительской особи 1 или родительской особи 2 присваиваются дочерним особям на основе случайно выбранной маски. Наконец, пересеченные веса устанавливаются в модель дочерней особи, и дочерние особи возвращаются.

В каждом поколении алгоритма SANE операция кроссинговера выполняется над выбранными родительскими особями путем вызова метода кроссинговера для генерации дочерних особей.

6) Оценка пригодности: особи потомства оцениваются по заданию и вычисляются значения их пригодности.

```
def get_fitness_scores(self, X_train, y_train):
    scores = []
    for model in self.population:
        model.fit(X_train, y_train, epochs=1, verbose=0)
        score = model.evaluate(X_train, y_train, verbose=0)[1]
        scores.append(score)
    return scores
```

В приведенном выше коде метод "get_fitness_scores" используется для расчета фитнес-балла для каждой особи в популяции. Для каждой отдельной модели сначала проводится однократное обучение модели на обучающих данных, которая подгоняется путем вызова метода model.fit. Затем те же самые обучающие данные используются для оценки точности модели путем вызова метода "model.evaluate" для получения метрики оценки, где точность используется в качестве оценки пригодности. Наконец, оценки пригодности для каждого индивидуума сохраняются в списке оценок, и этот список возвращается.

В каждом поколении алгоритма SANE фитнес-оценки всех особей в текущей популяции получаются путем вызова метода "get_fitness_scores", который используется для выбора особей из родительского поколения и отслеживания лучших особей в эволюционном процессе.

```
Generation 1 Best score: 0.6314554214477539
Generation 2 Best score: 0.6314554214477539
Generation 3 Best score: 0.6314554214477539
Generation 4 Best score: 0.6314554214477539
Generation 5 Best score: 0.6361502408981323
Generation 6 Best score: 0.6338028311729431
Generation 7 Best score: 0.6525821685791016
Generation 8 Best score: 0.6314554214477539
Generation 9 Best score: 0.6408450603485107
Generation 10 Best score: 0.6408450603485107
```

Рис.3 Лучшая модель для прогнозирования баллов

Операция замены: на основе значений пригодности выбираются особи из потомства и родителя для формирования следующего поколения популяции.

Шаги 3-6 повторяются: операции отбора, мутации, кроссинговера и замены выполняются в несколько итераций, пока не будут выполнены условия завершения (например, достигнуто максимальное количество итераций или достигнута желаемая производительность).

Для визуального представления структуры сети в разные эпохи используйте библиотеку визуализации (например, matplotlib) для построения топологии нейронной сети. Изобразите связи между входным, скрытым и выходным слоями и распределение нейронов в скрытых слоях. Это поможет увидеть, как нейронная сеть изменяется и адаптируется в течение эпохи.

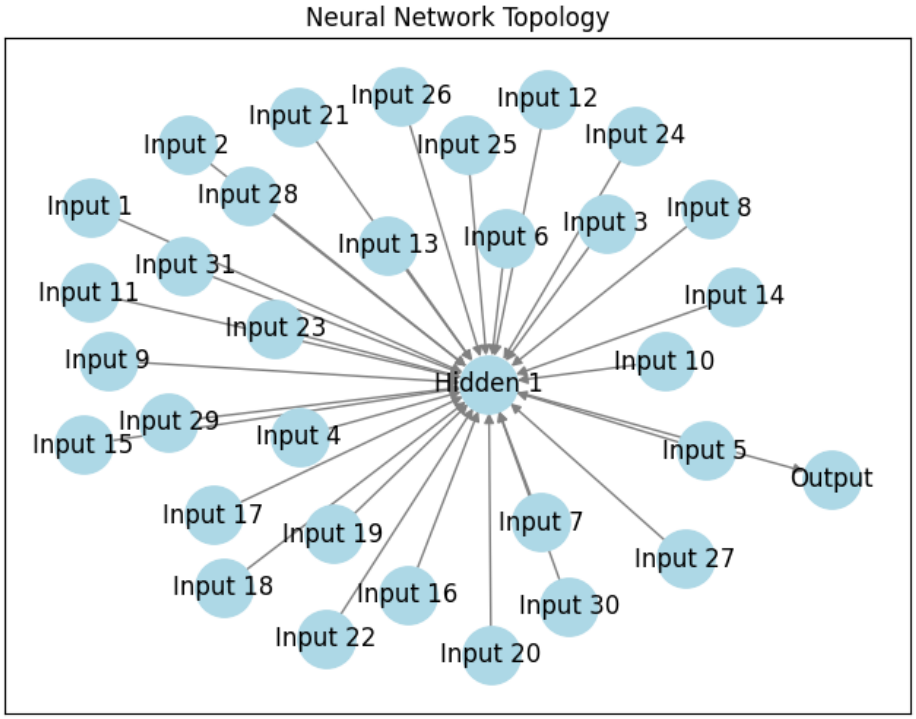

Рис.4 Топология нейронной сети

Связь узлов в нейронной сети, где каждый узел входного слоя соединен с первым узлом скрытого слоя ("Hidden 1"), а первый узел скрытого слоя соединен с узлом выходного слоя ("Output").

Общие целевые метрики включают среднюю квадратичную ошибку (MSE) и точность классификации. Значения целевых метрик могут рассчитываться и записываться на каждой итерации, а их изменения и тенденции описываются в отчетах.

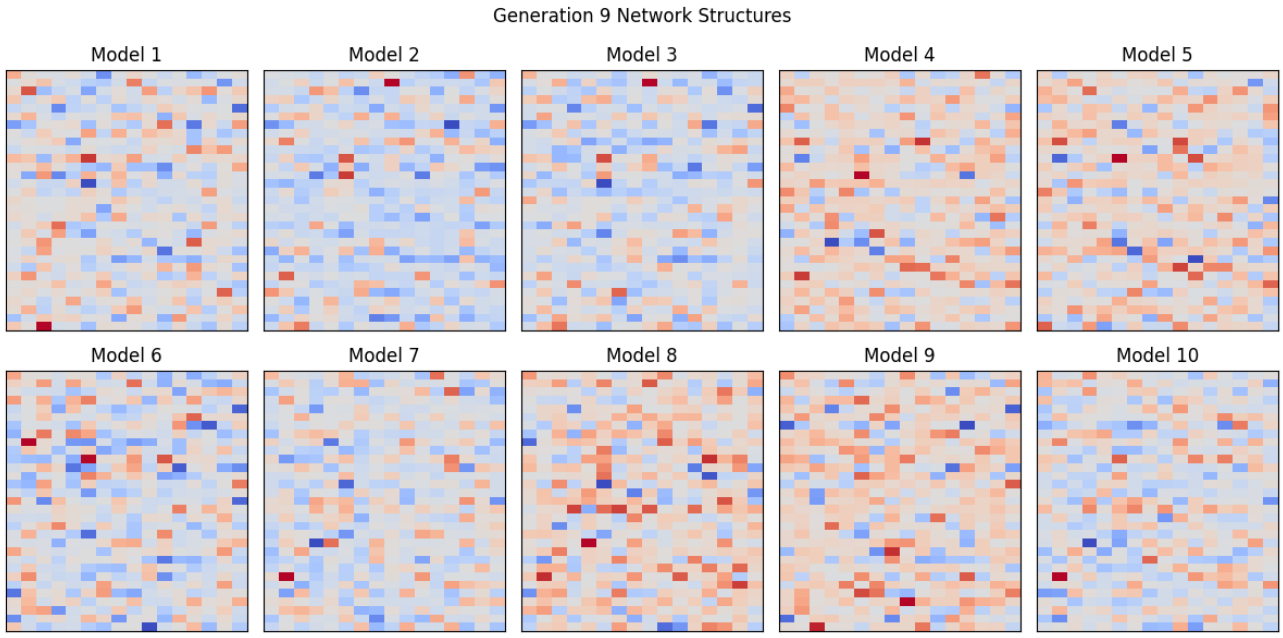

Рис.5 Взвешивание различных моделей

В каждое календарное время веса различных моделей, находящихся в этом календарном времени, обходятся в другом цикле. Для каждой модели мы создаем новую модель Sequential и устанавливаем для нее сохраненные веса. Затем мы используем plt.subplot() для создания субплощадки и задаем заголовок субплощадки, а также масштаб осей. В этом примере мы выбрали визуализацию весов первого скрытого слоя, используем plt.imshow() для отображения матрицы весов и задаем цветовое отображение с помощью cmap='coolwarm'. В окне графика в каждое календарное время визуализируются веса каждой модели, расположенные в ряд, что позволяет сравнивать и анализировать структуру нейронной сети.

Веса первого скрытого слоя визуализируются в виде изображения. Каждый пиксель изображения представляет собой значение веса связи между входным нейроном и скрытым нейроном. Для представления положительных и отрицательных весов использовалось цветовое отображение coolwarm, где синий цвет обозначает отрицательные веса, а красный - положительные. Полученные графики показывают структуру сети (веса первого скрытого слоя) для различных моделей в каждом поколении. Каждый подплан представляет модель, а цветные графики дают представление о величине и направлении весов.

Графики сходимости можно показать, построив график изменения целевой метрики по количеству календарных часов. Горизонтальная ось представляет собой количество календарных часов, а вертикальная - значение целевого показателя. Глядя на график сходимости, можно определить, сходится ли алгоритм, и понять ход обучения и производительность сети.

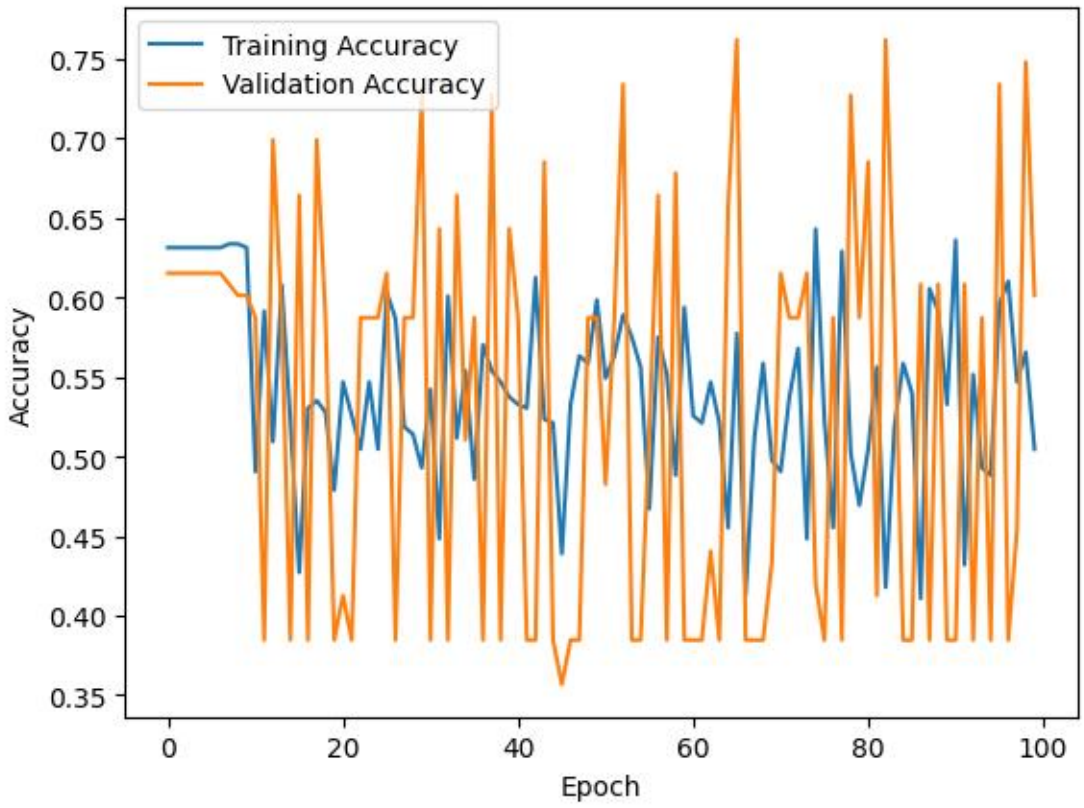

Рис.6 Диаграмма сходимости лучших моделей

Горизонтальная ось представляет собой количество итераций (Epoch), а вертикальная ось - коэффициент точности. Метки для горизонтальной и вертикальной осей задаются с помощью вызовов plt.xlabel() и plt.ylabel(), легенда добавляется с помощью plt.legend(), а график отображается с помощью plt.show(). Таким образом, мы можем наблюдать за работой модели в процессе обучения с помощью графика сходимости, определить, сходится ли модель, и оценить ее точность как на обучающем, так и на тестовом множестве.

Сравнительные эксперименты

Таблица 1

mutation_rate,num_generations	Generation :	Best score:	Test Accuracy:
0.35 , 100	32	0.64	0.58
0.4 , 150	56	0.74	0.74
0.35 ,30	14	0.65	0.73

В таблице 1 показаны результаты работы алгоритма SANE с различными коэффициентами мутации и суррогатными значениями. Ниже приводится объяснение каждого столбца:

Скорость мутации: В этом столбце указана скорость мутации, используемая в алгоритме SANE. Он указывает на вероятность возникновения мутаций в процессе воспроизводства.

Num Generations: В этом столбце указано количество поколений, выполняемых алгоритмом SANE. Это указывает на количество итераций или циклов алгоритма.

Поколения: В этом столбце указано количество поколений алгоритма SANE в процессе эволюции.

Лучший результат: В этом столбце показан лучший результат, достигнутый в каждом поколении. Он представляет собой наивысший показатель пригодности или точности, полученный для любой отдельной модели в этом поколении.

Точность теста: В этом столбце показана точность лучшей модели, полученной в каждом поколении, при оценке на тестовом наборе.

Из этой таблицы мы можем заметить следующее:

При коэффициенте мутации 0,35 и 100 поколениях наилучший результат составляет 32 балла, что соответствует точности теста 0,58.

Для коэффициента мутации 0,4 и 150 поколений лучший результат - 56, а соответствующая точность теста - 0,74.

Для коэффициента мутации 0,35 и 30 поколений наилучший результат - 14 баллов, а соответствующая точность теста - 0,73.

Исходя из этих результатов, можно сделать вывод, что более высокая скорость мутации (0,4) и большее число поколений (150) приводит к лучшей производительности, о чем свидетельствуют более высокие показатели наилучшей оценки и точности теста. Однако важно отметить, что эти результаты не могут быть окончательными, поскольку они основаны на ограниченном количестве экспериментов. Рекомендуется выполнить несколько запусков и учесть другие факторы, такие как вычислительные ресурсы и ограничения по времени, чтобы определить наилучшее сочетание скорости мутации и числа поколений для данной задачи.

Вывод:

Алгоритм SANE (Stochastic Amortized Network Evolution) - это алгоритм поиска структуры нейронной сети на основе генетического алгоритма, который используется для автоматического поиска наилучшей структуры нейронной сети. В экспериментах по классификации рака молочной железы алгоритм SANE может быть использован для поиска наилучшей структуры нейронной сети с целью повышения точности классификации.

Поиск наилучшей структуры нейронной сети для данной задачи осуществлялся с помощью эволюционного алгоритма. После нескольких поколений эволюции и настройки структуры сети алгоритм SANE выбирает модель нейронной сети с лучшей производительностью.

Обучив и оценив лучшую модель, можно получить точность этой модели на тестовом множестве. Построив график сходимости, мы можем наблюдать изменение производительности модели в процессе обучения, как с точки зрения точности обучения, так и точности проверки. Эти метрики можно использовать для оценки производительности и способности модели к обобщению.

Выводы экспериментов по классификации рака молочной железы во многом зависят от конечной производительности модели и метрик оценки. Точность, сильные и слабые стороны лучших моделей еще нуждаются в оптимизации и улучшении. Для практических задач классификации рака молочной железы, помимо производительности алгоритма, необходимо учитывать такие факторы, как характеристики набора данных, выбор признаков и интерпретация модели. Поэтому выводы, сделанные по результатам экспериментов по классификации рака молочной железы, должны учитывать эти факторы, подвергаться дальнейшему анализу и интерпретации.

Литература

[1] David E. Moriarty. Symbiotic Evolution Of Neural Networks In Sequential Decision Tasks. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1997.