Методы машинного обучения

Лекция 11

Эволюционные алгоритмы

Эволюционный алгоритм
Evalutionary algorithm

В искусственном интеллекте и машинном обучении эволюционные алгоритмы — это раздел эволюционных вычислений (моделирования), в которых используются модели процессов естественного отбора (размножение, рекомбинация, мутация и отбор) и принципы природной эволюции для решения **задач оптимизации**.

Решения задачи рассматриваются как особи популяции (агенты или индивидуумы). Множество особей принято называть популяцией. Качество каждого решения оценивается с помощью специальной целевой функции приспособленности (fitness function — фитнес-функция), которая задается окружающей средой. Такие алгоритмы относятся к адаптивным поисковым механизмам, включают эволюцию популяции и содержат следующие шаги:

1. Формируется начальная популяция методом случайного отбора (первое поколение).
2. Оценивается пригодность каждого члена популяции с помощью фитнес-функции.
3. Повторяются следующие действия (эволюция):
   - **отбор** — выбор наиболее приспособленных особей для размножения (родители);
   - **размножение** — формирование новых особей путем скрещивания и мутации, а затем оценка их пригодности;
   - **рекомбинация** — наименее приспособленные особи предыдущего поколения заменяются наиболее приспособленными особями нового поколения.

Эволюционный алгоритм Преимущества и недостатки

Преимущества:

- применимы к широкому классу задач оптимизации;
- легко комбинируются с другими методами;
- позволяют получать хорошо интерпретируемые результаты;
- обладают интерактивностью — можно включить в популяцию предложенные пользователем решения;
- рассматривают большое количество альтернативных решений.

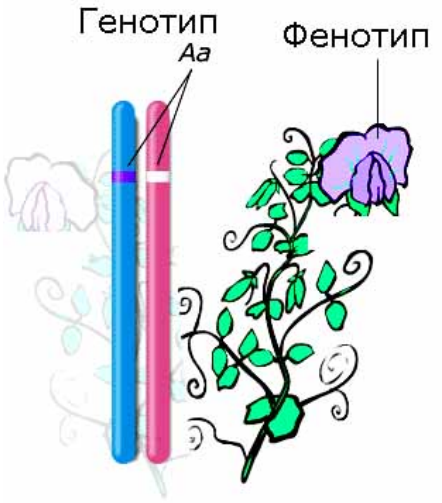
Недостатки:

- отсутствие гарантии нахождения оптимального решения за конечное время — алгоритм реализует локальную оптимизацию;
- слабая теоретическая база — алгоритм является эвристическим, т.е. точность и строгость постановки приносятся в жертву реализуемости;
- высокие вычислительные затраты.

Генетический алгоритм (ГА)

Эвристический алгоритм поиска, предложенный в 1975 году Джоном Холландом (John Holland) из Мичиганского университета, используется для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений

Работает с кодовыми последовательностями (КП), безотносительно к их смысловой интерпретации, поэтому КП и ее структура задается понятием **генотипа**, а его интерпретация, с точки зрения решаемой задачи, понятием **фенотипа**.



- **Фенотип** определяет, чем является объект в реальном мире.
- **Генотип** содержит всю информацию об объекте на уровне хромосомного набора.

Каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе, а совокупность генов называют хромосомой.

Каждая КП представляет, по сути, точку пространства поиска и называется особью или индивидуумом. Набор КП (особей) образует исходное множество решений *K* (популяцию). Количество особей в популяции характеризуется ее размером.

Особи в популяции оцениваются с использованием «функции приспособленности» (fitness function), в результате чего с каждым генотипом ассоциируется значение, которое определяет, насколько хорошо им описываемый фенотип решает поставленную задачу.

Кодирование признаков

Для представления генотипа объекта применяются битовые строки, т.е. ген — битовая строка, чаще всего фиксированной длины, которая представляет собой значение признака (атрибута) объекта.

1. Целые числа
- битовое значение признака.
- использование кода Грея.

2. Действительные числа
- битовое представление.
- интервальное с кодом Грея:
  - Разбивают весь интервал допустимых значений признака на участки с требуемой точностью.
  - Принимают значение гена как целочисленное число, определяющее номер интервала (используя код Грея).
  - В качестве значения параметра принимают число, являющиеся серединой этого интервала.

3. Нечисловые признаки
- Кодирование - преобразование в числовые

Число	Бинарный код	Код Грея
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100

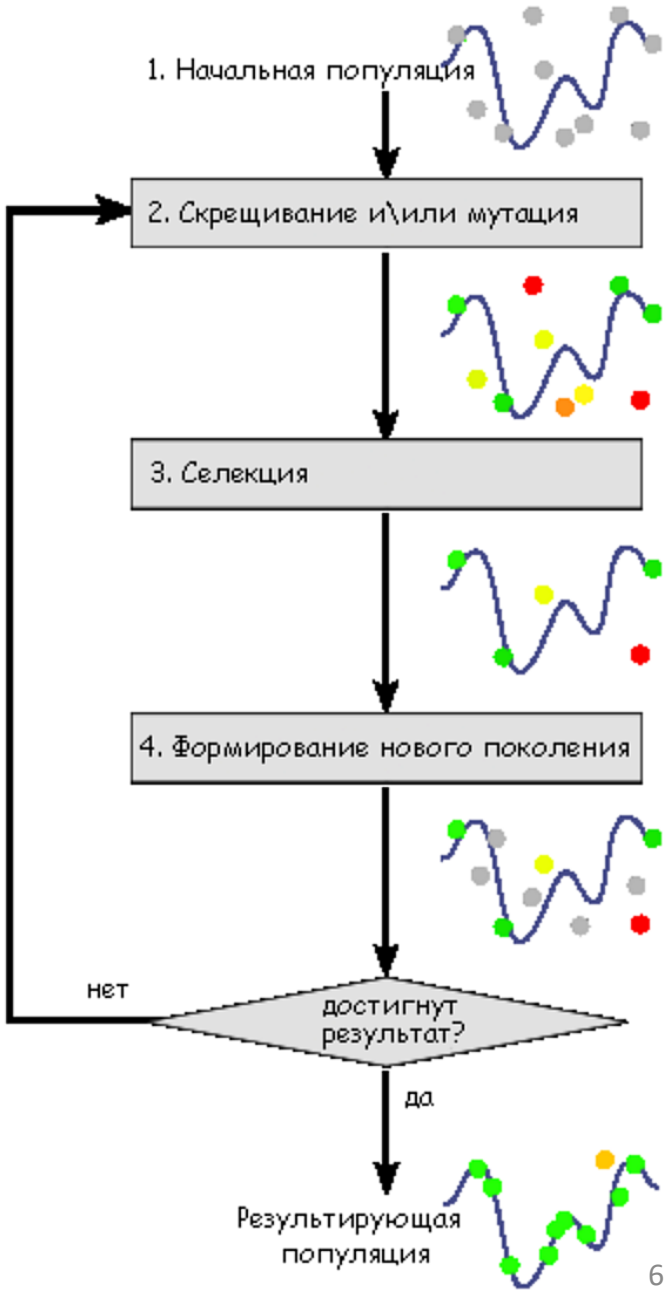
Код Грея *— двоичный код, в котором две «соседние» кодовые комбинации различаются только цифрой в одном двоичном разряде. Иными словами, расстояние Хэмминга между соседними кодовыми комбинациями равно 1.*

Основные этапы ГА

1. Задать целевую функцию приспособленности (fitness function) для особей популяции
2. Создать начальную популяцию
3. Начало цикла эволюции
   - Выбор родительских особей
   - «Скрещивание» (Crossover)
   - «Мутация» (Mutation) / Инверсия
   - Вычисление значений fitness function
   - Формирование нового поколения (селекция)
   - Если выполняются условия остановки, то «конец эволюционного процесса», иначе переход в начало цикла эволюции.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (*поколений*), пока не будет выполнен критерий остановки алгоритма. Критерием может быть:

- нахождение глобального, либо субоптимального решения;
- исчерпание числа поколений, отпущенных на эволюцию;
- исчерпание времени, отпущенного на эволюцию.



Целевая функция приспособленности (fitness function)

Вещественная или целочисленная функция одной или нескольких переменных, подлежащая оптимизации в результате работы генетического алгоритма, направляет эволюцию в сторону оптимального решения. Позволяет оценить степень приспособленности конкретных особей в популяции.

На выбор (построение) фитнесс-функции оказывают влияние следующие факторы:

- тип задачи – максимизация или минимизация;
- содержание шумов окружающей среды в фитнесс-функции;
- возможность динамического изменения фитнесс-функции в процессе решения задачи;
- объем допустимых вычислительных ресурсов;
- насколько различные значения для близких особей должна давать фитнесс-функция для упрощения отбора родительских особей;
- должна ли она содержать ограничения решаемой задачи;
- может ли она совмещать различные подцели (например, для многокритериальных задач).

В ГА фитнесс-функция может использоваться в виде черного ящика, т.е. для данной хромосомы она вычисляет значение, определяющее качество данной особи, и при этом внутри может быть реализована: в виде математической функции, программы моделирования (в том числе имитационного), нейронной сети, экспертной оценки и др.

Создание начальной популяции

Множество генотипов начальной популяции обычно создаётся случайным образом, тем самым выбирается нужное количество точек поискового пространства. Важно чтобы они соответствовали формату особей популяции, и на них можно было подсчитать функцию приспособленности.

Шаги:

- Инициировать начальный момент времени $t = 0$. Случайным образом сформировать начальную популяцию, состоящую из *k* особей $B_0 = A_1, A_2, \dots, A_k$.
- Вычислить приспособленность каждой особи:

$$F_{A_i} = fit(A_i), \;\; i = 1 \dots k,$$

и популяции в целом:

$$F_t = fit(B_t) = \sum_{i=1}^{k} fit(A_i)$$

Итогом первого шага является популяция B_0, состоящая из *k* особей.

Выбор родительских особей

- Панмиксия — оба родителя выбираются случайно, каждая особь популяции имеет равные шансы быть выбранной
- Селективный выбор - родителями могут быть только те особи, значение приспособленности которых не меньше среднего значения по популяции.
- Рулетка – вероятность выбора хромосомы определяется ее приспособленностью, то есть:

$$P_{Get\ A_i} = \frac{Fit(A_i)}{Fit(B_t)}$$

- Турнирный отбор – случайно выбираются несколько особей из популяции и победителем выбирается особь с наибольшей приспособленностью.
- Инбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наиболее похож на первого родителя
- Аутбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наименее похож на первого родителя

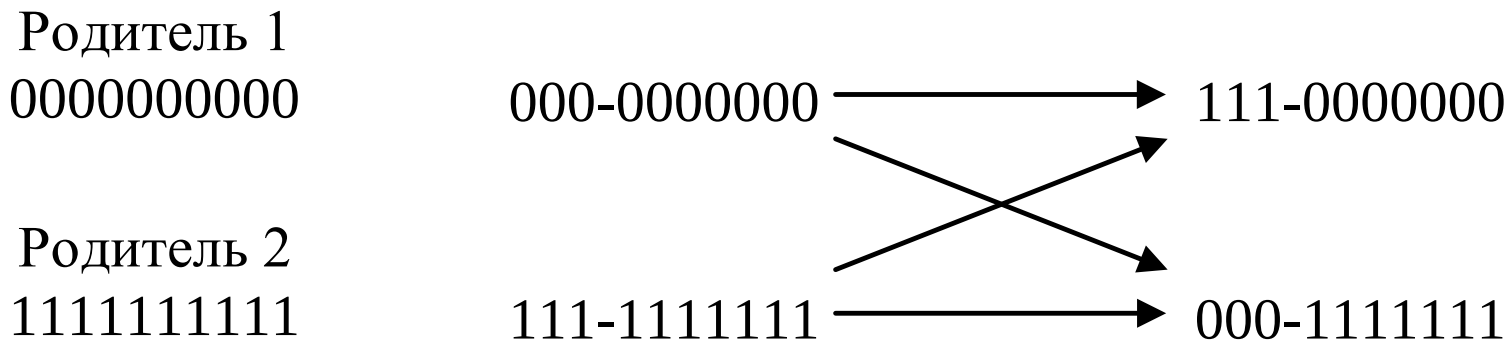
Инбридинг и аутбридинг построены на основе близкого и дальнего родства и бывают в двух формах: фенотипной и генотипной. В случае фенотипной формы похожесть измеряется в зависимости от геометрического расстояния между особями популяции или значениями функции приспособленности. В случае генотипной формы похожесть измеряется в смысле хеммингговского расстояния между хромосомными наборами особей (чем меньше отличий между генотипами особей, тем особи похожее).

Скрещивание (Crossover)

Различают: одноточечные, двухточечные и равномерные операторы скрещивания.

- Одноточечный оператор случайным образом выбирает одну точку разрыва. Обе родительские строки разрываются на два сегмента по этой точке. Затем соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков (Рисунок внизу слайда).
- В двухточечном операторе выбирается две точки разрыва, и родительские хромосомы обмениваются сегментом, который находится между этими точками.
- В равномерном операторе скрещивания каждый бит первого родителя наследуется первым потомком с заданной вероятностью, в противном случае этот бит передается второму потомку.

Потомки добавляются в популяцию.



Мутация (mutation) / Инверсия

Оператор мутации предназначен для поддержания разнообразия особей в популяции и представляет собой случайное изменение КП. Применение оператора позволяет выскочить из локального экстремума и продолжить движение в поисках лучших решений. Выполнение задается некоторой вероятностью. При использовании каждый бит в хромосоме с определенной вероятностью инвертируется.

Важно!

- Если оператор мутации имеет слишком большую вероятность применения, это приводит к разрушению генетического материала популяции и переключению на обычный поиск в произвольном порядке (метод перебора).
- Если вероятность слишком мала, то может возникнуть проблема с недостаточным разнообразием в популяции, алгоритм застрянет в локальном минимуме или поиск займет слишком много времени.

Оператор инверсии заключается в том, что хромосома делится на две части, и затем они меняются местами. Схематически это можно представить следующим образом:

000 1111111	**⇒**	**1111111 000**

Формирование нового поколения (Селекция)

На этапе отбора из всей популяции выбирают определённую долю особей, которая останется «в живых» и перейдет на следующий этап эволюции. Вероятность выживания особи A_i зависит от функции приспособленности $Fit(A_i)$. Из *N* особей популяции B_t должны остаться *S* особей, которые войдут в популяцию B_{t+1}, остальные отбрасываются.

- Турнирная селекция —случайно выбирается установленное количество особей (обычно две), затем из них выбирается особь с лучшим значением функции приспособленности.
- Метод рулетки — вероятность выбора особи тем вероятнее, чем лучше её значение функции приспособленности $P_{A_i} = \frac{Fit(A_i)}{\sum_{j=1}^{N} Fit(A_j)}$, где P_{A_i}— вероятность выбора i особи.
- Метод ранжирования — вероятность выбора зависит от места в списке особей отсортированных по значению функции приспособленности $P_{A_i} = \frac{1}{N}\left(a - (a-b)\frac{i-1}{N-1}\right)$, где $a \in [1,2]$, $b = 2 - a$, i- порядковый номер особи в отсортированном списке (т.е. $\forall i\ \forall j$ если $i < j$, то $Fit(A_i) \leq Fit(A_j)$ для поиска минимума фитнесс-функции).
- Равномерное ранжирование — вероятность выбора особи определяется выражением:

$$P_{A_i} = \begin{cases} \frac{1}{\mu}, & if\ 1 \leq i \leq \mu \\ 0, & if\ \mu \leq i \leq N \end{cases}$$, где $\mu \leq N$ параметр метода.

- Сигма-отсечение — для предотвращения преждевременной сходимости используются методы, масштабирующие значение фитнесс-функции. Вероятность выбора особи тем больше, чем оптимальнее значение масштабируемой функции приспособленности $P_{A_i} = \frac{F_i}{\sum_{j=1}^{N} F_j}$, где $F_i = 1 + \frac{Fit(A_i) - Fit_{Avg}}{2\sigma}$, Fit_{Avg} - среднее значение фитнесс-функции для всей популяции, σ— среднеквадратичное отклонение значения фитнесс-функции.
- Элитный отбор, который заключается в переносе некоторого количества (например, 10%) самых здоровых хромосом в следующее поколение.

Настройка параметров и процессов ГА

Не существует оптимального набора методов и параметров, которые сумели бы решить любую задачу. В связи с этим важную роль приобретает настройка параметров и процессов ГА. В частности, можно менять не только множество параметров (например, метод отбора и рекомбинирования), но и управлять коэффициентами применения генетических операторов.

- В зависимости от того, какой метод кодирования выбран для задачи, некоторые генетические операторы могут стать деструктивными. Важное значение приобретает правильный выбор кодировки и понимания эффектов использования операторов.
- Большое значение имеет параметр, отвечающий за размер популяции ГА. Если популяция мала, генетического материала может не хватить для решения задачи. Размер популяции также влияет на коэффициент применения операторов мутации и скрещивания.
- Особое внимание стоит уделять выбору типа и вероятностям применения генетических операторов.

Основная сложность, которая может возникать при использовании генетических алгоритмов и которая может быть преодолена путем настройки параметров и процессов.

- Преждевременное схождение, связанное с недостаточным разнообразием хромосом в популяции. Если большинство хромосом в популяции схожи между собой, то для селекции доступно мало рабочего материала.
- Обнаружение эффекта преждевременного схождения возможно сравнением среднего здоровья популяции с максимальным здоровьем. Если они схожи, это означает, что произошло преждевременное схождение популяции.
- Причиной может быть применяемый алгоритм отбора. Например, при использовании метода элиты выбираются хромосомы с самым высоким здоровьем, что приводит к сильному уменьшению размера популяции по сравнению с начальным.
- Если популяция слишком рано сошлась и решение не было найдено, следует перезапустить алгоритм с целью добавления нового генетического материала.

Непрерывный генетический алгоритм

В стандартном ГА используется двоичное кодирование признаков характеризующих объект, что не всегда удобно при работе с наборами вещественных чисел. Использование непрерывного генетического алгоритма может быть продиктовано необходимостью осуществлять оптимизацию в непрерывном пространстве значений вещественных чисел, тогда как двоичное представление может резко снизить точность найденного решения, в совокупности с резким увеличением длины хромосомы.

Основными преимуществами данного типа алгоритмов являются:

- возможность поиск в больших пространствах, что затруднительно в случае двоичных генов, когда увеличение пространства поиска сокращает точность решения при неизменной длине хромосомы;
- удобство представления решений, что обусловлено отсутствием операций кодирования/декодирования.

Непрерывный ГА - Основные операторы

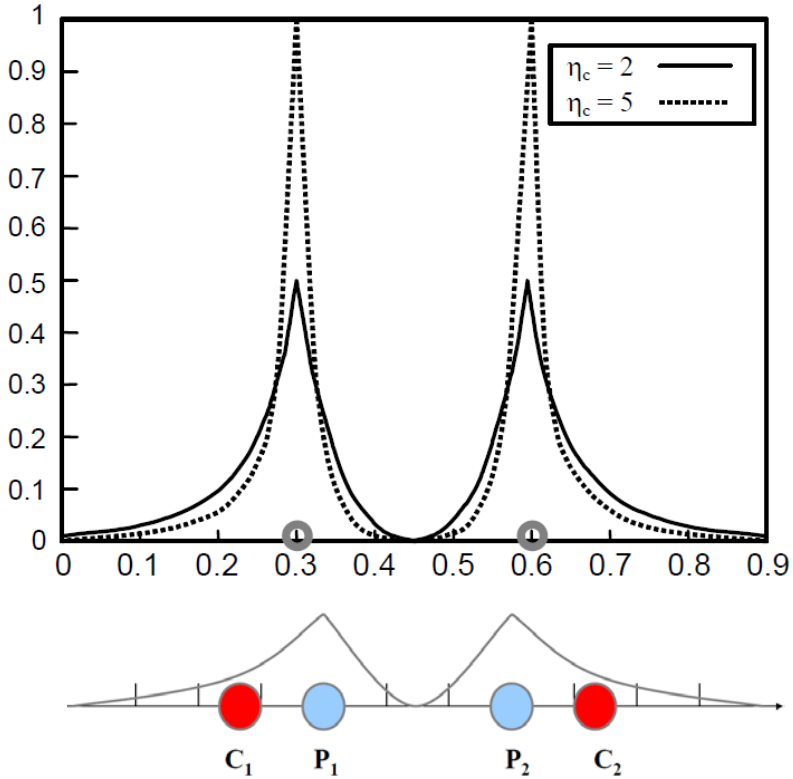
Отбор родительских особей для создания потомков осуществляется любым стандартным способом: рулетка, турнирный, случайный.

Оператор скрещивания - SBC (Simulated Binary Crossover) – кроссовер, который имитирует работу двоичного оператора скрещивания. Из двух векторов вещественных чисел осуществляется формирование двух новых векторов.

- Выбрать две родительские особи $\{a_0^1, a_1^1, a_2^1 \ldots a_N^1\}$ и $\{a_0^2, a_1^2, a_2^2 \ldots a_N^2\}$;
- Сгенерировать случайное число $u \in [0,1)$;
- Вычислить значение переменной β, характеризующей распределение вероятностей случайной величины;

$$\beta = \begin{cases} (2 \cdot u)^{1/(\eta+1)}, & u \le 0.5 \\ \left(\dfrac{1}{2 \cdot (1-u)}\right)^{1/(\eta+1)}, & u > 0.5 \end{cases}$$

- где η - параметр, влияющий на появления потомков вдали от родительских особей.



- Вычислить значения компонент векторов новых хромосом по формулам:

$$a_i^{1(new)} = 0.5[(1+\beta) \cdot a_i^1 + (1-\beta) \cdot a_i^2]$$

$$a_i^{2(new)} = 0.5[(1-\beta) \cdot a_i^1 + (1+\beta) \cdot a_i^2]$$

Среднее значение функции приспособленности остается неизменным у родителей и их потомков.

$$\frac{a_i^{1(new)} + a_i^{2(new)}}{2} = \frac{a_i^1 + a_i^2}{2}$$

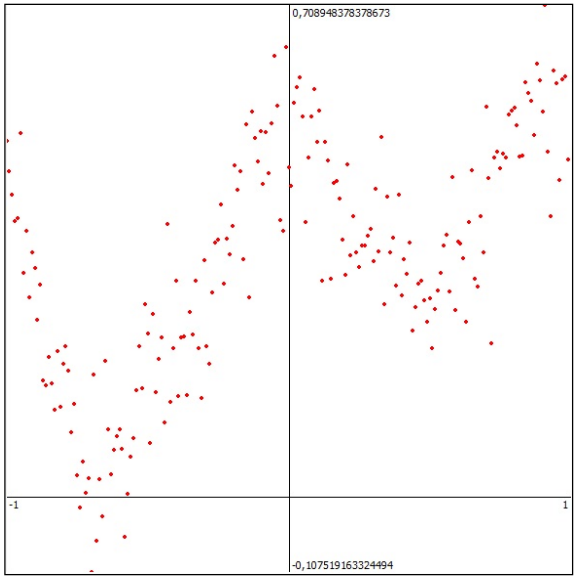
Оператор мутации выполняется с малой вероятностью, не превышающей 10%, для случайно выбранной компоненты вектора хромосомы на случайную величину в соответствии с нормальным законом распределения.

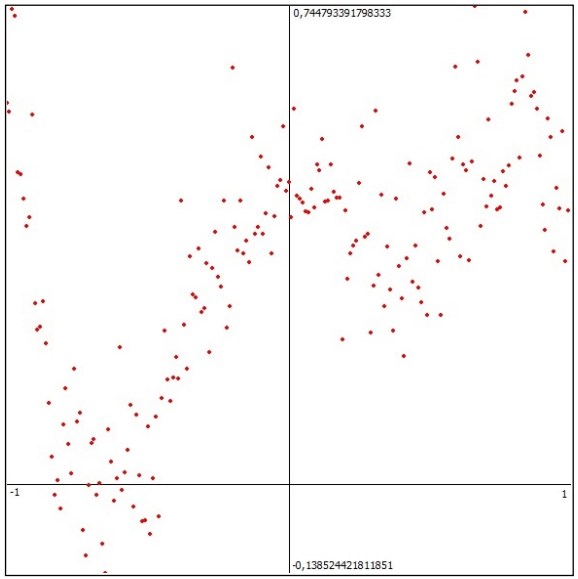
Пример: Построение аппроксимирующего полинома N-ой степени

Задача заключается в поиске таких значений коэффициентов $\{a_0, a_1, a_2 \ldots a_N\}$ аппроксимирующего полинома N-ой степени вида: $a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_N \cdot x^N$, которые бы наилучшим образом соответствовали исходному набору данных.

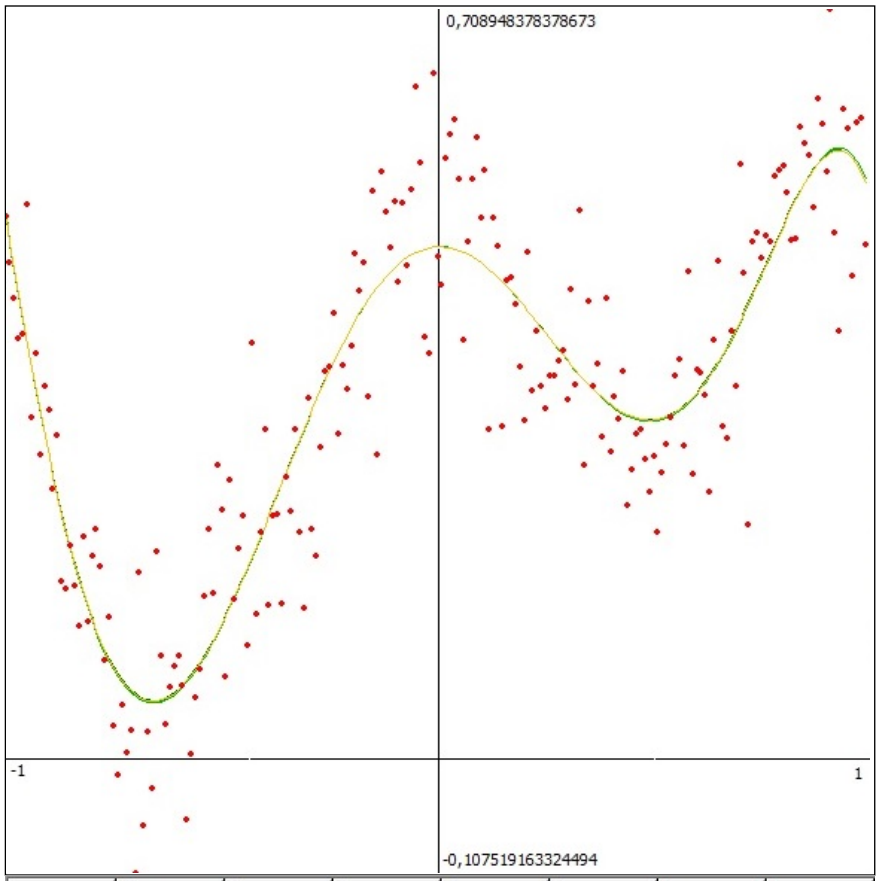
Целевая функция характеризует величину ошибки аппроксимации числовой последовательности с помощью полинома N-ой степени: $F_k = \frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{m}$

где y_i и $\hat{y}_i$ - вычисленное и измеренное значения соответственно, *m* – количество значений в числовой последовательности, *k* – особь популяции. Цель процесса оптимизации заключается в поиске таких значений $\{a_0, a_1, a_2 \ldots a_N\}$, для которых: $F_k \to 0$, $k = 1 \div K$, где *K* – количество особей в популяции.

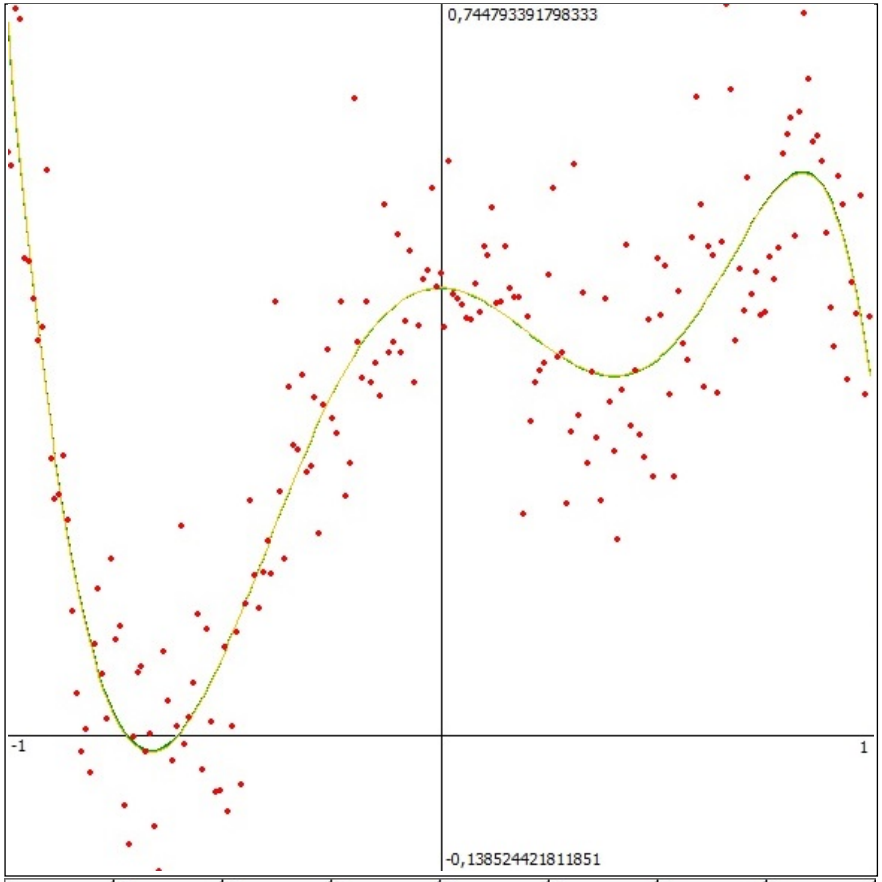




Результат моделирования работы непрерывного генетического алгоритма



	0	1	2	3	4	5	6
Polynom	0,5	0	-2	1	4	-1	-2
1	0,484218615	0,013165295	-1,79066290	0,959346537	3,399814320	-0,96267479	-1,57163313



	0	1	2	3	4	5	6
Polynom	0,5	0	-2	1,8	4	-2	-2
1	0,455694960	-0,00465775	-1,53303664	1,685238103	2,901641777	-1,87500155	-1,29677961

Использование генетического алгоритма для обучения нейронной сети

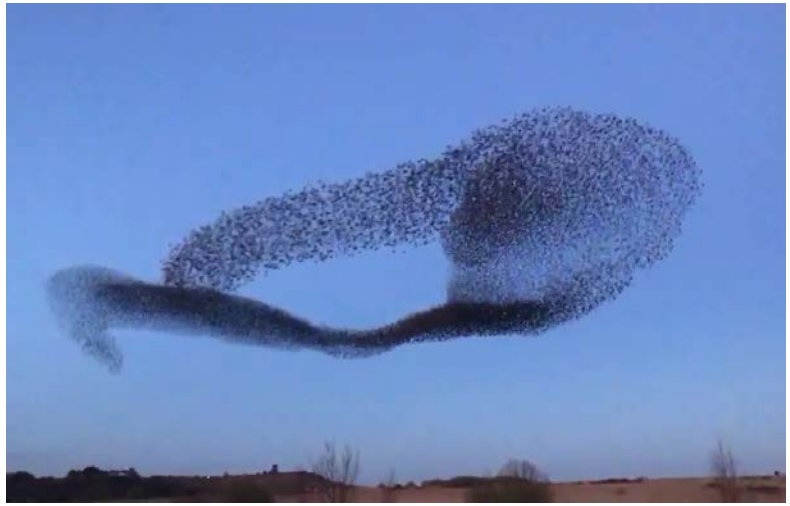
Генетические алгоритмы применимы для отыскания глобального экстремума многоэкстремальной функции и могут быть использованы для минимизации суммарной квадратической ошибки нейронной сети типа многослойный персептрон путем подстройки весовых коэффициентов.

Соотнесение основных структурных характеристик нейронной сети и параметров ГА осуществляется следующим образом:

- ген – весовой коэффициент нейронной сети;
- хромосома – набор генов или упорядоченный набор весовых коэффициентов нейронной сети, при этом каждая хромосома является возможным решением (т.е. таким набором весовых коэффициентов, который лучше подходят для решения имеющейся задачи);
- популяция – множество хромосом, вариантов наборов весовых коэффициентов (множество вариантов нейросети);
- эпоха – итерация, соответствующая созданию нового поколения хромосом.
- Над хромосомами осуществляются операции скрещивания, мутации, отбора и редукции, таким образом, происходит параллельная обработка множества альтернативных решений.

Алгоритм роя частиц - предпосылки

Пример коллективного поведения животных - стая птиц. Стая двигается плавно и скоординировано, словно ей кто-то управляет, представляя собой роевой интеллект. Отдельные птицы в ней действуют согласно определенным – довольно простым – правилам. Кружа в небе, каждая из птиц следит за своими сородичами и координирует свое движение согласно их положению, а найдя источник пищи, она оповестит их об этом.

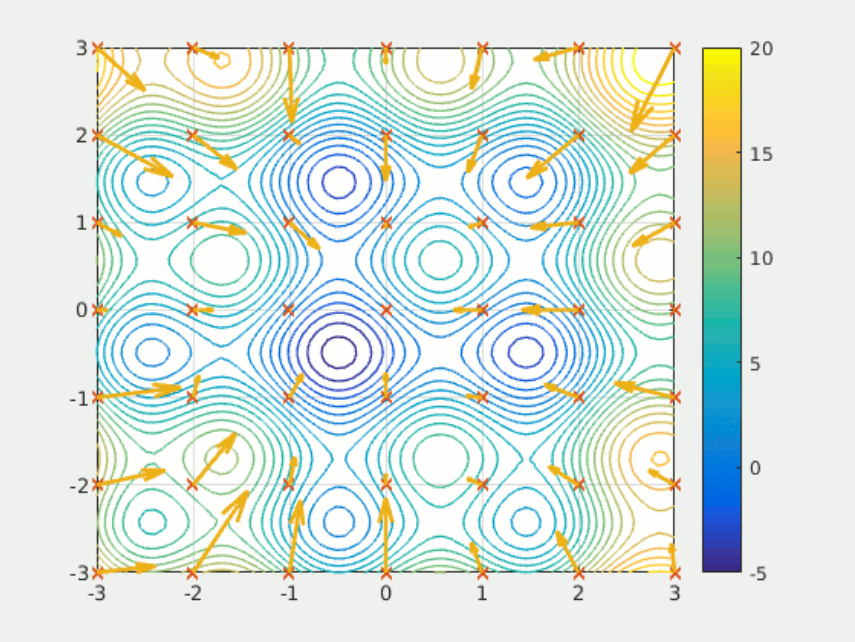


Наблюдение за птицами вдохновило Крейга Рейнольдса (Craig Reynolds) на создание в 1986 году компьютерной модели, которую он назвал Boids. Для имитации поведения стаи птиц, Рейнольдс запрограммировал поведение каждой из них в отдельности, а также их взаимодействие. При этом он использовал три простых принципа:

1. каждая птица в его модели стремилась избежать столкновений с другими птицами;
2. каждая птица двигалась в том же направлении, что и находящиеся неподалеку птицы;
3. птицы стремились двигаться на одинаковом расстоянии друг от друга.

Классический алгоритм роя частиц

Метод роя частиц (МРЧ) был предложен Кеннеди, Эберхартом и Ши (1995 год) и изначально предназначался для имитации социального поведения. Алгоритм моделирует многоагентную систему, где агенты-частицы (популяция возможных решений) двигаются к оптимальным решениям, обмениваясь при этом информацией с соседями.



Текущее состояние частицы характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними решением), а также вектором скорости перемещения. Оба этих параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит координаты лучшего из найденных ей решений, а также лучшее из пройденных всеми частицами решений – этим имитируется мгновенный обмен информацией между птицами.

Алгоритм роя частиц - инициализация

Пусть $f: \mathbb{R}^n \to \mathbb{R}$ — целевая функция, которую требуется минимизировать, S - количество частиц в рое, каждой из которых сопоставлена координата $x_i \in \mathbb{R}^n$ в пространстве решений и скорость $v_i \in \mathbb{R}^n$. Пусть также p_i— лучшее из известных положений частицы с индексом i, а g — наилучшее известное состояние роя в целом.

Для каждой частицы $i = 1, \dots, S$:

- Сгенерировать начальное положение частицы с помощью случайного вектора $x_i \sim U(b_{lo}, b_{up})$, имеющего многомерное равномерное распределение, где b_{lo}, b_{up}— нижняя и верхняя границы пространства решений соответственно.
- Присвоить лучшему известному положению частицы его начальное значение: $p_i = x_i$.
- Если $f(p_i) < f(g)$, то обновить наилучшее известное состояние роя: $g = p_i$.
- Присвоить значение скорости частицы:

$$v_i \sim U(-(b_{up} - b_{lo}), (b_{up} - b_{lo})).$$

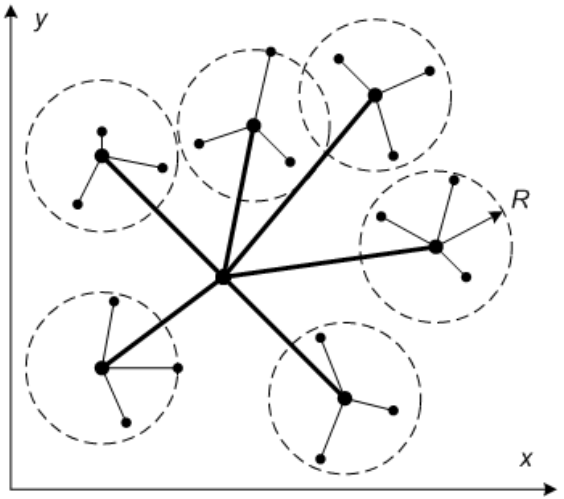
Алгоритм роя частиц - выполнение

- Пока не выполнен критерий остановки (например, достижение заданного числа итераций или необходимого значения целевой функции), повторять для каждой частицы $i = 1, \dots, S$ следующие действия:
  - Сгенерировать случайные векторы $r_p, r_g \sim U(0,1)$.
  - На каждой итерации направление и длина вектора скорости каждой из частиц изменяются в соответствие со сведениями о найденных оптимумах:
    - Обновить скорость частицы: $v_{i,t+1} = \omega v_{i,t} + \varphi_p r_p \odot (p_i - x_{i,t}) + \varphi_g r_g \odot (\mathrm{g} - x_{i,t})$, где операция $\odot$ - покомпонентное умножение, φ_p,φ_g-постоянные ускорения, ω – коэффициент инерции.
    - Обновить положение частицы x_i на вектор скорости: $x_{i,t+1} = x_{i,t} + v_{i,t+1}$. Этот шаг выполняется вне зависимости от улучшения значения целевой функции.
  - Если $f(x_i) < f(p_i)$, то:
    - Обновить лучшее известное положение частицы: $p_i = x_i$.
    - Если $f(p_i) < f(g)$, то лучшее известное состояние роя: $g = p_i$
- g содержит лучшее из найденных решений.

Параметры $\omega, \phi\varphi_p, \phi\varphi_g$ выбираются вычислителем и определяют поведение и эффективность метода в целом. Эти параметры составляют предмет многих исследований.

Алгоритм пчелиной колонии

Алгоритм оптимизации подражанием пчелиной колонии (*artificial bee colony optimization, ABC*) — один из полиномиальных эвристических алгоритмов для решения дискретных (комбинаторных) и непрерывных задач глобальной оптимизации в области информатики и исследования операций. Относится к категории стохастических бионических алгоритмов, основан на имитации поведения колонии медоносных пчел при сборе нектара в природе. Предложен Д. Карабога в 2005 г.



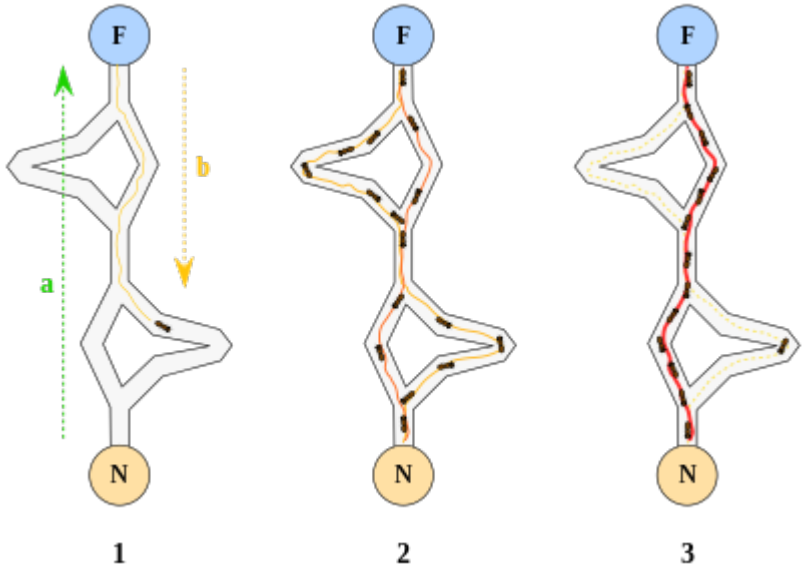
Алгоритм включает два основных этапа:

- При инициализации (начальной разведке) производится выполнение разведки пространства признаков с целью определения *K* его наиболее перспективных точек с наилучшими значениями целевой функции $f(X) = f(x_1, x_2, \dots, x_n)$ (в простейшем случае с использованием метода случайного перебора), которые запоминаются в улье.
- Последующая работа пчел улья. В окрестностях выбранных точек производится локальная разведка в пределах заданного радиуса разведки *R* с целью попытки уточнения решения, при этом при достижении улучшения в улье сохраняется обновленное значение f^* и соответствующий ему вектор параметров целевой функции X^*. Комбинируя работу пчел-разведчиков и рабочих пчел на протяжении заданного числа итераций *C* алгоритм обеспечивает постепенное улучшение запоминаемой выборки $\mathcal{R} = [X_1, X_2, \dots, X_K]$ из *K* решений.

По завершении его работы из указанного множества решений выбирается наилучшее, что является результатом работы алгоритма.

Муравьиный алгоритм

Муравьиный алгоритм (*алгоритм оптимизации подражанием муравьиной колонии - ant colony optimization*, ACO)— алгоритм для нахождения приближенных решений задач оптимизации на графах, таких как задача коммивояжера, транспортная задача и аналогичных задач поиска маршрутов на графах. Вычислительные затраты алгоритма полиномиально зависят от объема исходных данных.



В основе метода лежит поведение муравьев некоторых видов, которые изначально перемещаются в поисках пищи случайным образом, но, найдя ее, возвращаются в свою колонию, помечая путь феромонами. Это избавляет других муравьев от необходимости случайного поиска пищи и делает его более целенаправленным: найдя путь, помеченный феромонами, муравьи движутся по нему, усиливая плотность феромонов.

Однако со временем плотность феромонов уменьшается. И происходит это тем быстрее, чем длиннее путь, поскольку интервал перемещения по нему муравьев велик. Напротив, чем путь короче и чем интенсивнее он используется, тем выше плотность феромона на нем. Таким образом, выбирая пути с наибольшей плотностью феромона, муравьи сокращают расстояние, проходимое в поисках пищи, что эквивалентно поиску кратчайшего пути на графе.

На коротком пути, для сравнения, прохождение будет более быстрым, и, как следствие, плотность феромонов остаётся высокой. Если бы феромоны не испарялись, то путь, выбранный первым, был бы самым привлекательным.

Муравьиный алгоритм

Работа начинается с размещения муравьёв в вершинах графа (городах), затем начинается движение муравьёв — направление определяется вероятностным методом, на основании формулы вида:

Рёбра: Муравей будет двигаться от узла i к узлу j с вероятностью:

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

$\tau_{i,j}^{\alpha}$ - количество феромонов на ребре i, j;

α - параметр, контролирующий влияние $\tau_{i,j}^{\alpha}$ (определяет «стадность» алгоритма);

$\eta_{i,j}^{\beta}$ - привлекательность ребра i, j (начальное значение $1/d_{i,j}$, где d расстояние);

β - параметр, контролирующий влияние $\eta_{i,j}^{\beta}$ (определяет «жадность» алгоритма);

$\alpha + \beta = 1.$

Обновление феромонов

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j}$$

$\tau_{i,j}$ - количество феромонов на ребре i, j

ρ - скорость испарения феромона,

$\Delta\tau_{i,j}$ - количество отложенного феромона, обычно определяется как:

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/L_k & if\ Ant\ k\ travels\ on\ edge\ i,j \\ 0 & otherwise \end{cases}$$

где L_k-стоимость k-ого пути муравья (обычно длина).

Спасибо за внимание