Optymalizacja a uczenie się

Algorytmy optymalizacji stanowią funkcje przekształcające pewien zbiór punktów startowych w rozwiązanie jak najbliższe optymalnemu. Proces uczenia wygląda podobnie – zbiór parametrów początkowych przekształcany jest w taki sposób, aby uczony model jak najlepiej rozwiązywał postawiony przed nim problem.

Proces uczenia jako problem przeszukiwania

Uczenie możemy przedstawić jako przeszukiwanie przestrzeni parametrów z pewną określoną funkcją kosztu, którą się minimalizuje.

Uczenie sieci neuronowych

Jednym z przykładów może być uczenie sieci neuronowej. Sieć neuronowa przeprowadza obliczenia na podstawie parametrów w niej zawartych – wag i obciążeń. Te właśnie parametry podlegają zmianom w procesie uczenia. Można spojrzeć na zbiór aktualnych wartości parametrów sieci, jako na punkt w przestrzeni parametrów (każdy parametr byłby wymiarem przestrzeni - osią hipotetycznego układu współrzędnych). Zmiana wartości parametrów powoduje zmianę położenia punktu w przestrzeni.

Optymalizujemy parametry sieci poprzez dążenie do minimalizacji funkcji kosztu. Funkcja kosztu jest miarą tego, jak dobrze sieć wykonuje swoje zadanie, a dokładniej jak bardzo jej wyniki odbiegają od oczekiwanych. Na podstawie wartości różnicy między wyjściem faktycznym a oczekiwanym, szacuje się kierunek w którym należy przejść do innego punktu w przestrzeni parametrów w taki sposób, aby zmniejszyć wartość funkcji kosztu. Powszechnym podejściem do tego problemu jest wykorzystanie algorytmu wstecznej propagacji gradientu[1].

Indukcja drzew decyzyjnych

Innym ciekawym przykładem jest budowanie drzew decyzyjnych dla problemów klasyfikacji. Drzewo jako struktura jest acyklicznym grafem skierowanym, w którym wierzchołki posiadające tylko jednego sąsiada, z których nie wychodzą krawędzie nazywane są liśćmi, krawędzie gałęziami, a wierzchołek posiadający jedynie krawędzie z niego wychodzące – korzeniem.

Drzewo decyzyjne jest klasyfikatorem przyjmującym na wejściu zbiór atrybutów obiektu (przykład) oraz zawierającym tzw. reguły decyzyjne. Reguła składa się z testu dla pewnego atrybutu wejścia oraz decyzji, czyli akcji podejmowanej w zależności od wyniku testu. Akcją może być zaklasyfikowanie wejścia do konkretnej klasy (odwiedzenie liścia drzewa) lub przejście do kolejnego testu (węzła potomnego aktualnie rozpatrywanego testu). Odpowiedź drzewa następuje w momencie przejścia ścieżką od korzenia do liścia wskazującego, do jakiej klasy należy przykład. Uczeniu podlegają przyjęte testy oraz pozycje liści.

Drzewa zazwyczaj buduje się w procesie uczenia przy pomocy algorytmów zachłannych, które generalnie znajdują drzewa niekoniecznie optymalne, ale akceptowalnie dobre. Takie zadanie

---

[1] Aby dowiedzieć się więcej sprawdź: https://en.wikipedia.org/wiki/Backpropagation lub Paweł Wawrzyński - „Podstawy sztucznej inteligencji”, Oficyna Wydawnicza Politechniki Warszawskiej

można również przedstawić jako problem przeszukiwania, co może ułatwić znalezienie optimum globalnego. Wyróżnia się wiele podejść do tego problemu.

Przestrzeń drzew decyzyjnych można spróbować przeszukiwać algorytmem ewolucyjnym, w którym osobniki reprezentują drzewa decyzyjne. Mutacją w takim algorytmie będzie dodanie lub usunięcie węzła bądź zmianach wśród testów w węzłach i klas w liściach. Krzyżowanie może być zrealizowane poprzez zamianę poddrzew dwóch osobników.

Przeszukiwanie takiej przestrzeni można zdefiniować też inaczej. Przyjmijmy jako operację elementarną usunięcie węzła testowego, którego dzieci są liśćmi lub zmianie liścia w test, a jako metrykę ilość operacji elementarnych, które należy wykonać aby przekształcić jedno drzewo decyzyjne w drugie. Otrzymujemy spójną przestrzeń z ograniczoną metryką. Taką przestrzeń przyjęto w metodzie KIS, wykorzystywanej do klasyfikacji sekwencji DNA.[2] Metoda zakłada również przeszukiwanie przestrzeni atrybutów w celu wyznaczenia jak najlepszych testów dla drzewa.

Inne podejścia zakładają przeszukiwanie przestrzeni reguł. Przestrzeń stanowi drzewo z pustą regułą w korzeniu. Kolejne poziomy drzewa zawierają bardziej złożone reguły. Przestrzeń utworzoną w ten sposób można przeszukiwać w głąb z wykorzystaniem różnego rodzaju heurystyk.

Model zastępczy

Model zastępczy jest metodą używaną w optymalizacji w sytuacji, gdy funkcja celu jest nieznana bądź bardzo kosztowna do obliczenia. Model zastępczy jest rodzajem symulacji funkcji celu, zastępczą funkcją znacznie łatwiejszą do obliczenia.

Generowanie modelu zastępczego polega na przemyślanym próbkowaniu funkcji celu, którą traktuje się jako czarną skrzynkę. Na podstawie próbek model uczy się najistotniejszych cech funkcji celu, dzięki czemu stanowi łatwe do obliczenia i akceptowalnie dobre przybliżenie właściwej funkcji celu. Istotne jest dobranie najmniejszej możliwej liczby próbek, by zredukować koszt utworzenia modelu.[3]

Model tworzony jest w następujący sposób:

1. Następuje wybór punktów przestrzeni do zbioru danych dla modelu zastępczego – model zastępczy znać będzie tylko położenie tych punktów oraz wartości funkcji celu dla nich, nie posiada żadnych więcej informacji na temat funkcji celu.
2. Dla wybranych punktów przeprowadza się wyliczenie funkcji celu. W ten sposób powstają pary X (punkt) – Y (wartość), które można wykorzystać w uczeniu modelu.
3. Następuje wybór modelu zastępczego (sieć neuronowa, drzewo decyzyjne etc.)
4. Podobnie jak w innych zadaniach uczenia, zbiór danych dzielony jest na dane treningowe, walidacyjne i testowe. Z użyciem danych treningowych i walidacyjnych przeprowadzany jest proces uczenia.

---

[2] Więcej o metodzie KIS w rozprawie doktorskiej dr. inż. Rafała Biedrzyckiego „Konstruktywna indukcja w zadaniu klasyfikacji sekwencji DNA”

[3] Więcej informacji: https://en.wikipedia.org/wiki/Surrogate_model, „Surrogate-Based Optimization”, Zhong-Hua Han and Ke-Shi Zhang School of Aeronautics, Northwestern Polytechnical University, Xi’an, P.R. China

5. Wytrenowany model testowany jest przy użyciu danych testowych. Jeżeli wynik jest satysfakcjonujący, model może zostać wykorzystany do zadania optymalizacji. Jeżeli nie, należy zmienić model, jego parametry lub sposób próbkowania i wykonać kroki ponownie.

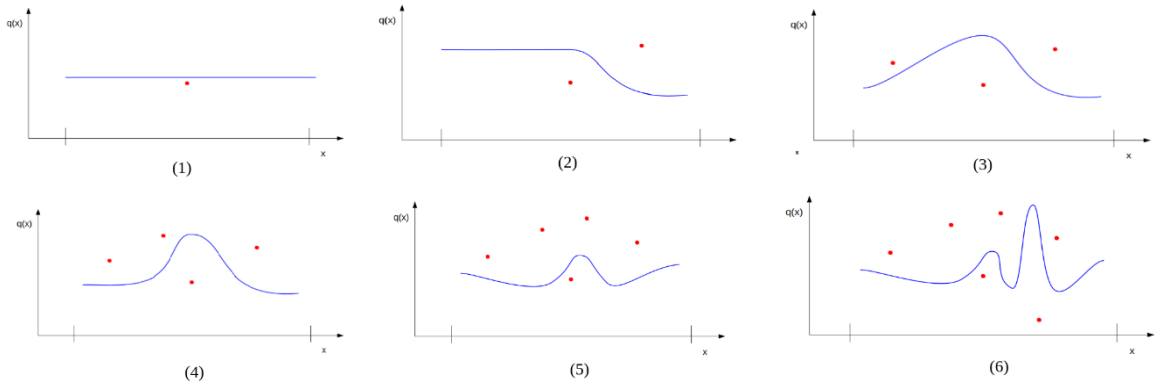
Po utworzeniu modelu możemy go wykorzystać do przybliżania wartości funkcji celu w rozpatrywanej przestrzeni, zamiast dokonywać kosztownych obliczeń „prawdziwej" funkcji celu. Model może zostać wykorzystany przez dowolną metodę optymalizacji – algorytm ewolucyjny, symulowane wyżarzanie itd.

Wykorzystanie modelu zastępczego może wydawać się świetnym pomysłem ze względu na niski koszt aproksymacji funkcji celu, dość intuicyjną ideę i duże możliwości dopasowania modelu – można w końcu wykorzystać dowolnie wybrany aproksymator i w zależności od potrzeb zmieniać jego parametry.

To podejście ma też pewne wady. Po pierwsze trudno oszacować w jakim stopniu model zastępczy musi być zgodny z faktyczną funkcją celu, aby uznać go za wystarczająco dobry – w niektórych przypadkach może to być 80% w innych 99%. Dodatkowo może się zdarzyć, że model będzie bardzo dokładny dla jednego podzbioru, ale dla innego dawać słabe przybliżenie, co może utrudnić interpretację wyników algorytmu optymalizacji.[4]

Uczenie jako adaptacja stanu

Algorytmy optymalizacji mogą wykorzystywać do tworzenia nowych punktów zmienną losową, której rozkład da się analizować. Jest to rozkład próbkowania, który może mieć możliwość adaptacji. W procesie generacji kolejnych punktów można zaobserwować, że im lepsza jest wartość funkcji celu w danym punkcie, tym chętniej jego otoczenie jest próbkowane.

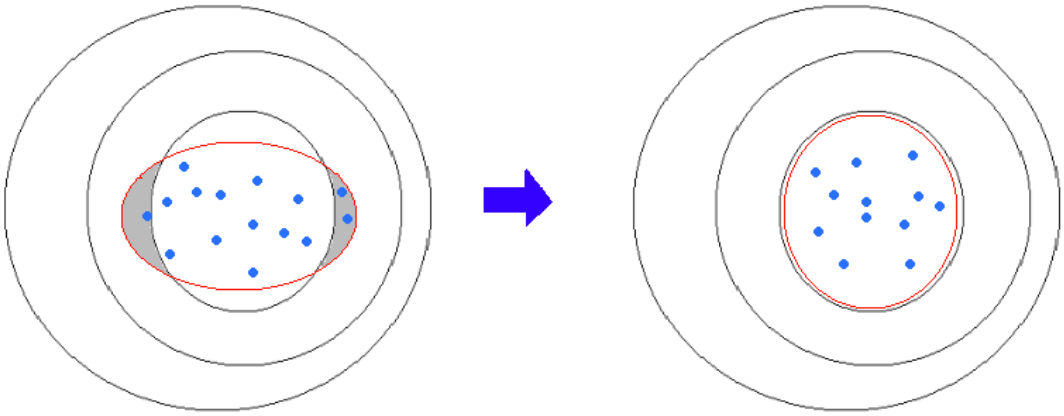


Adaptacja stanu w CMAES

W algorytmach takich jak CMAES stanem są parametry zmiennej losowej opisującej rozkład generowanych punktów. W kolejnych iteracjach algorytmu parametry dostosowywane są w taki sposób, aby zbliżyć się z generowanymi punktami do optimum funkcji celu.

[4] Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. https://christophm.github.io/interpretable-ml-book/.

Algorytm CMAES ma pewną ciekawą właściwość. W wyniku dostosowywania macierzy kowariancji rozkładu zmienia się kształt obwiedni generowanych punktów, gdy znajdują się one w otoczeniu optimum lokalnego. Obwiednia przyjmuje kształt bardzo zbliżony do poziomic funkcji celu wokół optimum. Można stwierdzić, że CMAES uczy się kształtu funkcji celu, ponieważ rozkład zmiennej losowej zaczyna naśladować wyciągnięcie poziomic funkcji celu we wszystkich kierunkach.



Aproksymacja hesjanu

Hesjan, czyli macierz drugich pochodnych funkcji podwójnie różniczkowalnej w pewnym punkcie, ma powszechne zastosowanie w algorytmach uczenia. Obliczenie hesjanu jest kosztowne czasowo, dlatego zamiast dokładnych wyliczeń stosuje się jego aproksymację, co redukuje złożoność problemu z $O(n^2)$ do $O(n)$ – opierają się na tym tzw. metody quasi-Newtona. W tych metodach optymalizacji hesjan jest przybliżany poprzez analizowanie kolejnych wektorów gradientu w czasie działania algorytmu[5].

Analiza danych jako optymalizacja

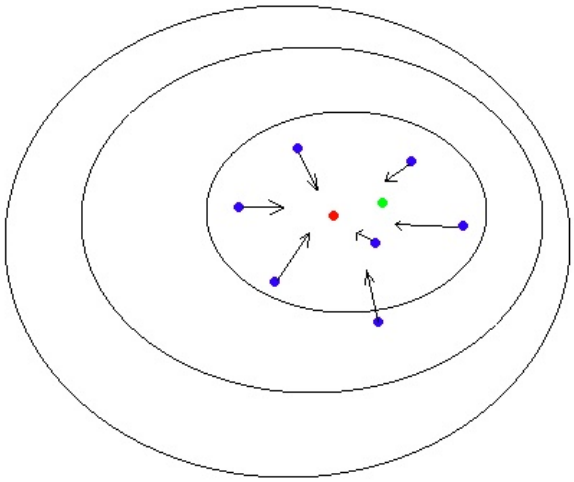
Rozwiązując zadanie optymalizacji, możemy przeglądać i analizować wszystkie punkty generowane przez algorytm i umieszczane w logu – niezależnie od tego, czy zostaną one wybrane na etapie selekcji, mogą one nieść pewne informacje pozwalające przybliżyć położenie optimów lokalnych.

Środek ciężkości

Najprostszym możliwym przykładem takiej analizy danych może być szukanie środka ciężkości punktów. Wyobraźmy sobie działający w pewnej przestrzeni algorytm roju cząstek. Po napotkaniu „wniesienia”, lokalnego maksimum funkcji celu, cząstki będą skupiać się wokół niego. Można zatem przypuszczać, że faktyczne optimum lokalne znajduje się gdzieś w okolicy środka ciężkości roju.

---

[5] Więcej na temat aproksymacji hesjanu na przykład w „A Stochastic Quasi-Newton Method for Large-Scale Optimization”, R. H. Byrd, S.L. Hansen, Jorge Nocedal,Y. Singer, February 19, 2015)

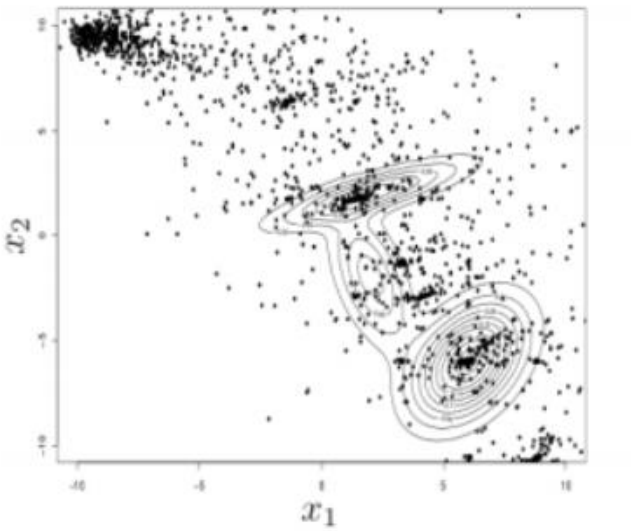


Na rysunku kolorem zielonym zaznaczono optimum lokalne, kolorem czerwonym środek ciężkości. Nie pokrywają się, ale położenie środka jest pewnym przybliżeniem optimum.

Podobnie środek ciężkości można wykorzystać w algorytmach ewolucyjnych. Monitorowanie środka ciężkości w czasie wykonania algorytmu może polepszyć jego skuteczność. [6]

Analiza skupień

Lepszy przykład stanowi analiza skupień. Rozważmy algorytm ewolucji różnicowej. W kolejnych krokach algorytmu generowane punkty zbliżają się do optimów lokalnych, tworząc wokół nich skupienia. Analiza skupień pozwala do pewnego stopnia przewidzieć jak wygląda funkcja celu w przeszukiwanej przestrzeni.



Generowane w kolejnych iteracjach punkty mają większą gęstość wokół wzniesień.

[6] Jarosław Arabas, Rafał Biedrzycki „Improving Evolutionary Algorithms in a Continuous Domain by Monitoring the Population Midpoint”, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 21, NO. 5, OCTOBER 2017