Design Patterns - Peer to Peer

Miłosz Kluczek, Lidia Moryc, Patryk Przybysz, Adam Stajek

November 2024

1 Wstęp

Peer-to-peer (P2P) to sposób komunikacji i wymiany danych bezpośrednio między użytkownikami (tzw. „równymi sobie" w sieci), bez centralnego serwera. Każdy uczestnik sieci pełni jednocześnie rolę klienta i serwera, co umożliwia bezpośrednie przesyłanie plików lub informacji.

W naszym projekcie projektujemy i wdrażamy system przetwarzania danych w architekturze peer-to-peer, który rozwiąże problem optymalizacji hiperparametrów modelu za pomocą przeszukiwania siatki (grid search). W ramach systemu, różne węzły w sieci będą odpowiedzialne za testowanie różnych kombinacji hiperparametrów równolegle, a wyniki będą dostępne online podczas obliczeń. Celem projektu jest znalezienie najlepszej kombinacji hiperparametrów, przy zastosowaniu odpowiednio dobranej heurystyki, co pozwoli zoptymalizować wydajność wybranego modelu.

2 Architektura

W zaprojektowanym systemie przetwarzania danych w architekturze peer-to-peer (P2P) każdy węzeł uczestniczący w optymalizacji hiperparametrów działa autonomicznie i współpracuje z pozostałymi węzłami, aby efektywnie przeprowadzić obliczenia i wybrać optymalną konfigurację hiperparametrów. Proces działania węzłów obejmuje następujące kroki:

1. Wymiana informacji o przetestowanych konfiguracjach Każdy węzeł przeprowadza obliczenia dla przypisanej mu konfiguracji hiperparametrów modelu, a następnie ocenia wyniki za pomocą wybranej metryki. Po zakończeniu testu danej konfiguracji, węzeł wysyła do innych węzłów informację o przetestowanej architekturze wraz z uzyskaną wartością metryki.

2. Przechowywanie logów przetestowanych konfiguracji Każdy węzeł przechowuje lokalnie log przetestowanych konfiguracji hiperparametrów oraz uzyskanych wyników.

3. Wysyłanie wyników po zakończeniu zadania Po zakończeniu zadania obliczeniowego każdy węzeł wysyła wynik wraz z przetestowaną konfiguracją i wartościami metryki do pozostałych węzłów w sieci. Informacje te są automatycznie dodawane do logów na innych węzłach, co zapewnia ich spójność w całym systemie.

4. Wybór nowej konfiguracji do testowania Po zakończeniu przetwarzania bieżącej konfiguracji każdy węzeł wybiera nowy zestaw hiperparametrów do przetestowania. Proces ten odbywa się na podstawie heurystyki, która ma na celu jak najbardziej efektywne przeszukiwanie przestrzeni hiperparametrów.

3 Struktura repo

Nasza biblioteka, którą można znaleźć na Githubie, znajduje się w folderze Peer2Peer. Aplikację prezentującą możliwości modułu można uruchomić z pliku app.py. Klasy *Logger* oraz *Database* implementują metaklasę *SingletonMeta*, co zapewnia globalny dostęp do tylko jednej ich instacji. Klasy dziedziczące po *Command* wraz z klasami *MessageManager* oraz *UserInterface* są zaimplementowane zgodnie ze wzorcem projektym *Command*, z kolei cały folder *strategies* realizuje wzorzec *Strategy*. Centralnym punktem naszego programu jest klasa *Node*, a użytkownik powinien używać tylko publicznego interfejsu *UserInterfaceImplementation*.

4 Przebieg interakcji w systemie

Pierwszy węzeł dołącza do sieci Gdy pierwszy węzeł dołącza do sieci, rozpoczyna proces przetwarzania przez grid searcha na określonej kombinacji hiperparametrów. Na początku, użytkownik w aplikacji webowej definiuje parametry początkowe, które są przesyłane do węzła.

Aktualizacja bazy danych Po rozpoczęciu grid searcha, węzeł aktualizuje bazę danych, dodając nową kombinację hiperparametrów, ale bez wartości wyników obliczeń. Dzięki temu inne węzły w sieci są świadome, że dany zestaw hiperparametrów jest w trakcie testowania, co pozwala na lepszą synchronizację danych i uniknięcie duplikacji testów.

Zakończenie obliczeń i zapis wyniku Po zakończeniu obliczeń na danej kombinacji hiperparametrów, węzeł zapisuje wynik w bazie danych. Dodatkowo, interfejs aplikacji webowej może na bieżąco wyświetlać logi, które informują użytkownika o postępie obliczeń i uzyskanych wynikach, co pozwala na monitorowanie procesu predykcji.

5 Wzorce projektowe

1. **Strategia (Strategy Pattern)**
   Opis: Wzorzec strategii umożliwia definiowanie różnych algorytmów w oddzielnych klasach i pozwala na dynamiczne wybieranie jednej z nich w zależności od sytuacji. Ułatwia to elastyczne zarządzanie różnymi metodami wykonywania określonego zadania.

   Zastosowanie w projekcie: W projekcie wzorzec strategii zostanie wykorzystany **do zarządzania wyborem różnych metod przeszukiwania i oceny hiperparametrów**. Dzięki temu można łatwo implementować i zmieniać strategie przeszukiwania – np. losowe próbkowanie, grid search czy bardziej zaawansowane heurystyki – bez konieczności modyfikowania głównej logiki aplikacji.

2. **Command (Command Pattern)**
   Opis: Wzorzec komendy umożliwia enkapsulację wywołania operacji jako obiektu, co pozwala na zapisywanie, przekazywanie oraz uruchamianie komend w sposób niezależny od ich odbiorców. Ułatwia to zarządzanie żądaniami i zadaniami w skomplikowanych systemach.

   Zastosowanie w projekcie: W projekcie wzorzec komendy będzie użyty do zarządzania komunikacją między węzłami sieci P2P. Komendy (np. „dołącz do sieci", „powiadom o wynikach") będą enkapsulowane w obiektach i przekazywane do węzłów w sieci. Dzięki temu łatwo jest wysyłać konkretne polecenia do różnych komputerów i dynamicznie zarządzać procesem obliczeń rozproszonych. Każdy węzeł będzie mógł niezależnie wykonywać otrzymane komendy, co ułatwi synchronizację i skalowalność całego systemu.

3. **Singleton (Singleton Pattern)**
   Opis: Singleton zapewnia, że klasa ma tylko jedną instancję i dostarcza globalny punkt dostępu do tej instancji. Jest przydatny w przypadkach, gdy wymagane jest jedno źródło prawdy, np. do zarządzania stanem aplikacji.
   Zastosowanie w projekcie: W projekcie singleton może być użyty do przechowywania i zarządzania danymi, które muszą być dostępne globalnie dla całej sieci, np. wynikami częściowych obliczeń lub bieżącym stanem optymalizacji. Singleton zapewni, że do tych danych będzie miała dostęp tylko jedna instancja, co zminimalizuje ryzyko konfliktów lub niespójności. Dane przesyłane między węzłami będą spójne, co jest kluczowe w architekturze rozproszonej.

6 Komunikacja

W systemie peer-to-peer (P2P) komunikacja między węzłami odbywa się za pomocą socketów TCP, które umożliwiają niezawodną, dwukierunkową wymianę danych. Każdy węzeł pełni rolę zarówno nadawcy, jak i odbiorcy, otwierając odpowiednie gniazda (sockety) do komunikacji. Węzeł działa jako serwer, nasłuchując na określonym porcie na przychodzące połączenia, a także jako klient, łącząc się z innymi węzłami w celu wymiany danych. Sockety TCP zapewniają stabilność połączeń, gwarantując, że dane nie zostaną utracone w trakcie transmisji. Po zakończeniu obliczeń, węzeł wysyła wyniki do innych węzłów, które odbierają dane, aktualizując swoje lokalne logi. Na podstawie otrzymanych informacji, węzły mogą wybierać nowe konfiguracje hiperparametrów do testowania.