Univerzita Karlova v Praze

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

INTERPOLACE S VYUŽITÍM METODY IDW V JAZYCE PYTHON

Jáchym Slovák

2. B-FGG

v Chocni 5. 2. 2026

Anotace:

Tato práce se zabývá implementací interpolační metody Inverse Distance Weighted (metoda inverzního váženého průměru) v jazyce Python.

Zadání:

Z textového souboru načtěte vstupní data představovaná [x, y, z], kde hodnota „z“ odpovídá vizualizovanému jevu. Z druhého textového souboru načtěte hodnoty [X, Y], souřadnici „Z“ určete interpolační metodou IDW. Vstupním parametrem je „k“ představující počet nejbližších sousedů. Hodnotu váhy volte w(k)=1/s(k), kde „s“ je vzdálenost mezi interpolovaným bodem a k-tým nejbližším bodem. Interpolované hodnoty uložte do textového souboru.

Rozbor problému:

Pomocí interpolace se snažíme zjistit pravděpodobnou hodnotu hledaného bodu pomocí matematických operací s prostorově rozloženými vzorky, jejichž hodnotu známe. Metoda Inverse Distance Weighted (IDW) odhaduje hodnotu bodu na základě známých hodnot v jeho okolí. S narůstající vzdáleností známých bodů od hledaného bodu klesá jejich váha pro výpočet hodnoty tohoto bodu. Body, které se nacházejí v blízkém okolí tak budou mít větší vliv na výslednou hodnotu. Výsledná hodnota *„Z“* je počítána jako vážený aritmetický průměr okolních hodnot *„z_i“*.

Nejprve je pro každý pár bodů (pár hledaný bod – známý bod) vypočítána Euklidovská vzdálenost *„s“* podle vzorce:

$s = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Váha *„w_i“* každého z *„k“* sousedů je nepřímo úměrná vzdálenosti od hledaného bodu. Můžeme ji vypočíst podle vzorce: $w_i = \frac{1}{s_i^p}$. V implementovaném algoritmu je mocnina *„p“* rovna 1. Váha tedy s narůstající vzdáleností klesá lineárně.

Výsledná hodnota „Z“ je dána podílem sumy vážených hodnot a sumy vah podle vzorce: $Z = \frac{\sum(z_i \cdot w_i)}{\sum w_i}$.

Pro větší efektivitu výpočtu můžeme zvolit, kolik nejbližších bodů bude do výpočtu vstupovat, jelikož vzdálenější body budou hrát již marginální roli a spíše přispívají k lehkému zkreslení výpočtu.

Existující algoritmy:

Pro řešení stejného problému lze využít mnoho dalších interpolačních algoritmů. Mezi nejpoužívanější algoritmy patří například Kriging, který sleduje, jak se data mění se vzdáleností a počítá pravděpodobnou statistickou chybu. Na rozdíl od IDW je matematicky náročnější a využívá také statistických ukazatelů.

Další interpolační metoda může být Spline. Ten se matematicky snaží minimalizovat křivost povrchu, je tak vhodný spíše pro pozvolně se měnící veličiny. Pracuje s křivkami.

V geoinformatice se také často využívá metoda přirozeného souseda (Natural Neighbor). Jedná se o geometrické rozdělení plochy pomocí Voroného teselace a porovnávání váhy okolních bodů pro výpočet neznámého bodu dle plošného zastoupení v rámci jeho polygonu.

Použitý algoritmus:

Zvolený algoritmus vypočítá hodnotu hledaného bodu metodou IDW s nastavitelným počtem nejbližších sousedů, kteří do výpočtu vstupují.

1. Import knihoven

Pro výpočet vzdáleností potřebujeme odmocninu. Z knihovny *math* byla importována funkce *sqrt*. Z důvodu ochrany před chybou při absenci dat byla z knihovny *sys* importována funkce *exit*.

2. Třída Point

Třída *Point* reprezentuje datový model bodu. Za použití datového typu *float* vytváří prostředí pro souřadnice a hodnotu bodu.

3. Třída IDW

Třída *IDW* je hlavní výpočetní jednotkou.

V metodě *__init__* přijímá seznam dat. Metodou *interpolation* vyhledává *„k“* nejbližších sousedů, které pomocí metody *sort* s klíčem *lambda* seřadí podle vzdálenosti do nového seznamu. Singularita nulové vzdálenosti je zde ošetřena tak, že při výpočtu vzdálenosti *s = 0* metoda vrátí přímo hodnotu bodu, čímž zabrání dělení nulou. Pomocí iterátoru *for* pro každý z vybraných nejbližších sousedů vypočítá jejich váhu a váženou hodnotu, které jsou následně přičítány k vahám a váženým hodnotám dalších sousedů pro konečný výpočet hodnoty „Z“ dle výše uvedeného vzorce.

4. Třída File (Správce souborů)

Třída *File* zapouzdřuje veškeré operace se soubory. Pomocí metody *__init__* do této třídy vstupují názvy vstupních souborů a požadovaného výstupního souboru.

Metoda *load_known_points* validuje a připravuje vstupní data známých bodů pro výpočet. Byl vytvořen nový seznam vstupních dat, do kterého byly přidány instance třídy *Point*, které přejímají hodnoty souřadnic a hodnoty „z“ ze souboru. Pomocí metody *strip* došlo k odstranění neviditelných znaků a metodou *split* byly jednotlivé hodnoty od sebe odděleny středníkem. Logický operátor *if* zde slouží především pro validaci vstupních dat a k přeskakování problémových řádků. V případě, že se na řádku vyskytuje nějaká chyba (např. hodnoty nejsou odděleny

středníkem nebo je neplatný počet vstupních hodnot) dojde k vypsání varovné zprávy o přeskočení řádku. Celý načítací proces se nachází v bezpečném bloku *try-except*, který zajišťuje řešení výjimek, jako např. *FileNotFoundError*, vypsáním varovné zprávy.

Metoda *calculation_and_result* řídí zápis výsledků do výstupního souboru. Jelikož je tento soubor kombinací textového souboru s neznámými body a interpolačního výpočtu, načteme data ze souboru pomocí *with open.* V této části kódu probíhá validace obdobným způsobem, jako u načítání známých bodů. Po validaci vstupních dat přichází mnou vytvořená metoda *interpolation*, jejímž vstupními daty jsou souřadnice neznámých bodů a uživatelem zvolený parametr *„k“*. Metoda *write* zde slouží pro zapsání výsledků do textového souboru. Pro přehlednost byla interpolovaná hodnota *„Z“* zaokrouhlena na 4 desetinná místa. Pokud vše proběhlo v pořádku, program funkcí *print()* zahlásí úspěšné uložení do textového souboru. I zde jsou ošetřeny výjimky pomocí varovných zpráv. Případné výjimky jsou zde vyřešeny bezpečným blokem *try-except*.

5. Třída App

Třída *App* zde funguje jako hlavní řídící jednotka. Získává od uživatele parametr *„k“* a spouští celý proces.

Metodou *__init__* předává informace o vstupních a výstupních souborech třídě *File*. Pomocí metody *get_k* program umožňuje uživateli zvolit vstupní parametr *„k“*, tedy počet nejbližších sousedů, kteří budou vstupovat do výpočtu interpolace. Pomocí iterátoru *while True* program umožní v případě zadání neplatného vstupu ihned zadat nové číslo. Pomocí funkce *len* program zjišťuje počet známých (vstupních) bodů, aby bylo možné předejít chybě v podobě požadavku na větší množství vstupních bodů, než je k dispozici. I tento proces je umístěn v bezpečném bloku *try-except*, program tedy v případě zadání jiného znaku než čísel, vypíše varovnou zprávu. Metodou *launch* dojde ke spuštění celého procesu a předání parametrů. Pomocí logického operátoru *if* je zde ošetřen případ, kdyby vstupní data neobsahovala žádné body. V tomto případě by došlo k vypsání varovné zprávy a ukončení programu funkcí *exit*.

Vstupní a výstupní data:

Vstupem jsou dva textové soubory s názvy „zname_body_idw.txt“ a „nezname_body.txt“. U obou musí být dodržena stejná pravidla formátování. První soubor obsahuje řádky s hodnotami ve tvaru „X; Y; Z“, kde „X“ a „Y“ zastupují souřadnice v kartézské soustavě a „Z“ vyjadřuje naměřenou hodnotu. Soubor „nezname_body.txt“ obsahuje pouze řádky s „X; Y“, tedy bez hodnoty „Z“, která je interpolací vypočítána programem. Oba soubory mohou obsahovat libovolný počet řádků. Hodnoty musí být znázorněny čísly.

Výstupem je textový soubor s názvem „vystup_idw_slovak.txt“. Formát je obdobný, jako u souboru „zname_body_idw.txt“, tedy ve tvaru „X; Y; Z“. Výsledné hodnoty budou desetinná čísla datového typu *float*.

Problematická místa a možná vylepšení:

Překlep ve vstupních datech zaviní přeskočení řádku. V případě menšího počtu bodů to bude mít zásadní vliv na správnost interpolace. Program sice vypíše varovnou zprávu, ale interpolaci přesto provede, čímž dochází ke zkreslení. Uživatel musí zadat data v požadovaném formátu a s úplnou čistotou pro správné fungování programu.

Tento kód bude efektivní pouze pro menší množství vstupních bodů, jelikož pro seřazení a výběr „k“ nejbližších sousedů počítá vzdálenost pro všechny vstupní body, které podle ní posléze seřadí. Problém by se dal vyřešit implementací stromových datových struktur, které by umožnily prohledat pouze nejbližší body namísto všech existujících vstupních bodů.

Váha se počítá lineárně (1/s), což u interpolace metodou IDW může vést k fenoménu tzv. býčího oka (Bull’s Eye). To se projevuje vytvářením kruhovitých útvarů kolem známých hodnot. Tento problém se dá vylepšit přidáním parametru mocniny, který zajistí, že se zvyšující se vzdáleností klesá váha známých bodů rychleji. Výpočet váhy by tak mohl vypadat například takto: $w = 1/s^2$.

Zdroje a literatura:

How inverse distance weighted interpolation works. *ArcGIS Pro Documentation* [online]. Esri, [cit. 5. 2. 2026]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm

Inverse Distance Weighting (IDW). *Freie Universität Berlin – Statistics and Geodata Analysis using R (SOGA-R)* [online]. Freie Universität Berlin, Department of Earth Sciences [cit. 5. 2. 2026]. Dostupné z: https://www.geo.fu-berlin.de/en/v/soga-r/Advances-statistics/Geostatistics/Inverse-Distance-Weighting-IDW/index.html

11. Spatial Analysis (Interpolation). *QGIS Documentation – A Gentle GIS Introduction* [online]. QGIS Project, [cit. 5. 2. 2026]. Dostupné z: https://docs.qgis.org/3.40/en/docs/gentle_gis_introduction/spatial_analysis_interpolation.html