ÚLOHA 4
Lineární regrese více proměnných

Zadáno na cvičení: 5
Mezní termín: 10.11. 2022
Maximální počet bodů: 10-15
Povinná úloha

Zadání

Stáhněte si archiv *linRegMulti(NumPy)* ze stránky *Lineární regrese*.

Struktura kódu je následující:

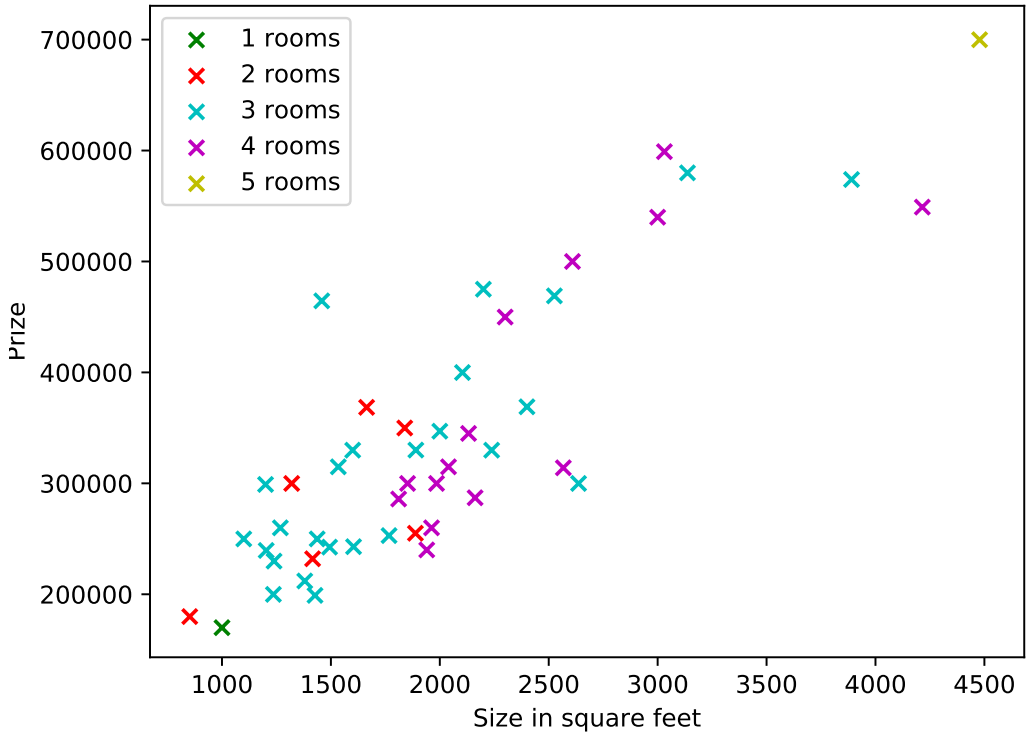
- *data1.txt* – vstupní data pro první část úlohy
- *data_machines.txt* – vstupní data pro druhou část úlohy
- *data_machines_readme.txt* – popis jednotlivých příznaků v datech pro druhou část úlohy.
- *model/LinearRegression*[1] – třída implementující celou funkcionalitu linearní regrese (hypotéza, pokutová funkce, gradient)
- *optimize/Optimizer* – generický optimalizační algoritmus
- *optimize/GradinetDescent*[1] – gradientní sestup
- *utils/normalize_features()*[1] – škálování příznaků
- *utils/build_dict()*[2] – vytváří slovník pro reprezentaci výčtových příznaků
- *utils/transform()*[2] – transformuje výčtové příznaky na one-hot vektory
- *utils/cross_vlidation()* – křížová validace pro využití stejných dat pro trénování i testování
- *visualize.py* – kontrolní vizualizace podobné jako v předchozí úloze
- *ex4.py*[1] – hlavní skript první části úlohy
- *ex4-2.py* – hlavní skript druhé části úlohy

Třídy/funkce označené [1] budete doplňovat v rámci první části, třídy/funkce označené [2] ve druhé části.

1 Vícerozměrná lineární regrese a škálování příznaků

Vstupní data

V této části budeme predikovat cenu domu podle jeho velikosti a počtu místností. Rozložení dat můžete vidět na obrázku 1.



Obrázek 1: Vizualizace dat.

Úkoly

V této části budete programovat lineární regresi o libovolném počtu proměnných. Před touto úlohou je doporučeno naprogramovat úlohu předchozí, protože většina úkolů je pouze drobnou modifikací úkolů z předchozí úlohy.

1. **Cenová funkce a hypotéza**

   Cenovou funkci a hypotézu naprogramujte pomocí maticových operací (bez cyklů).

2. **Gradientní sestup**

   Gradientní sestup musí umožňovat nastavení více ukončovacích podmínek:

   (a) Počet iterací *num_iters*

   (b) Minimální chyba *minCost*

   (c) Minimální rozdíl parametrů oproti předchozí iteraci *minThetaDiff*

   může být nastaveno 1-N ukončovacích podmínek. Všechny zadané ukončovací podmínky musí být kontrolovány současně.

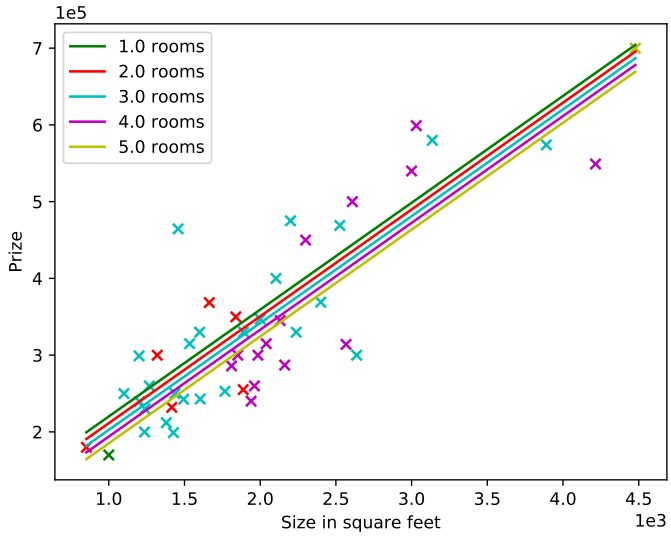
3. **Škálování příznaků**

   Normalizujte střední hodnotu a rozptyl příznaků .

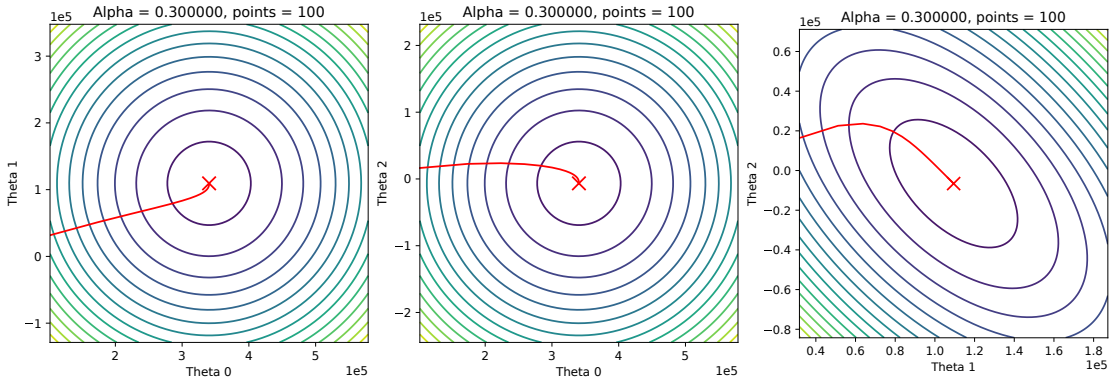
4. V souboru *ex4.py* **doplňte predikci ceny domu o 1650 čtverečních stopách a 3 místnostech**. Stejnou predikci udělejte pomocí **normální rovnice**.

5. **Vyladťe parametry gradientního sestupu** tak, aby konvergoval co nejrychleji.

Po škálování příznaků by vykreslené grafy měly vypadat zhruba tak, jak je vidět na Obrázku 3.



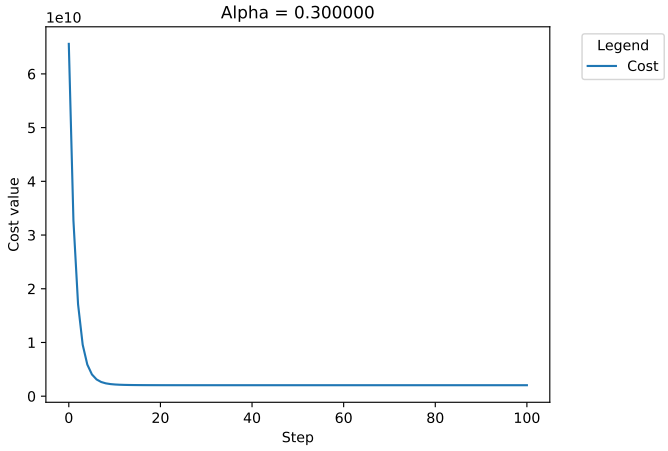
Obrázek 2: Regrese dat.



Obrázek 3: Vývoj chyby se změnou parametrů modelu.

2 Transformace příznaků (nepovinná část)

Cílem je predikovat skóre výkonu počítačů na základě některých jeho parametrů. Popis parametrů najdete v souboru *data_machines_readme.txt*. Vaším úkolem bude naprogramovat univerzální funkci pro reprezentaci textového řetězce jako příznaku výčtového charakteru. Pro tyto účely se využívá one-hot vektor, což je vektor o velikosti rovné počtu všech různých hodnot (plus jedna pro neznámou hodnotu). Řetězec pak reprezentujeme tímto vektorem, kde máme pouze jednu jedničku na pozici odpovídající danému řetězci a zbytek složek jsou nuly. Vaším úkolem je naprogramovat univerzální funkce pro vytvoření této reprezentace. Ve fázi trénování musíte vytvořit slovník. In-



Obrázek 4: Graf konvergence

dexy v tomto slovníku pak budou odpovídat nenulové složce one-hot vektoru. Budete doplňovat funkce *dictionaryFT_train.m* a *dictionaryFT_transform.m*.

Příklad

Ve fázi trénování dostaneme text:

$$\begin{pmatrix} 'first' \\ 'second' \\ 'first' \\ 'third' \\ 'first' \\ 'second' \end{pmatrix}$$

Slovník tedy vypadá následovně:

$$('first','second','third')$$

pokud vstupem funkce transform bude:

$$\begin{pmatrix} 'first' \\ 'second' \\ 'first' \\ 'fourth' \\ 'fifth' \end{pmatrix}$$

výstupem pak bude matice:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

První složka vektoru odpovídá všem neznámým slovům.
Na konkrétním pořadí prvků nezáleží, ale musí být pořád stejné.