ÚLOHA 3
Lineární regrese více proměnných

Zadáno na cvičení: 4
Mezní termín: 31.10. 2017
Maximální počet bodů: 10-15
Povinná úloha

Zadání

Stáhněte si archiv *linRegMulti.zip* ze stránky *2 Lineární regrese*. Archiv obsahuje tyto soubory:

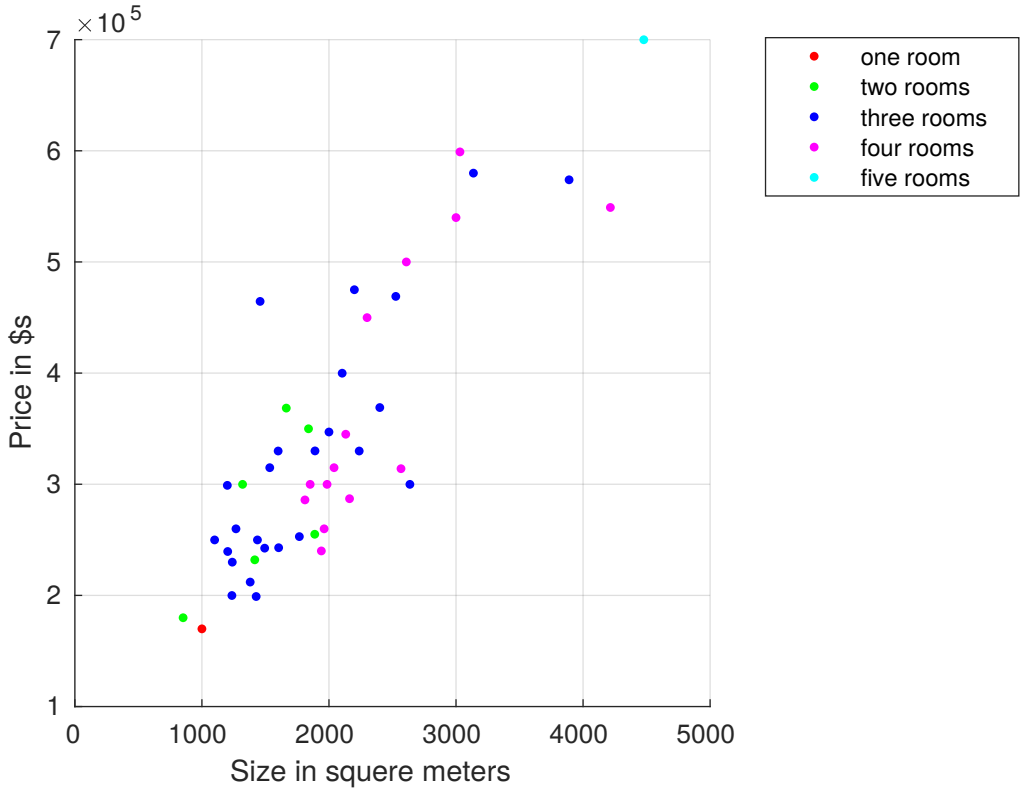
- *crossValidation.m* – křížová validace pro využití stejných dat pro trénování i testování
- *data1.txt* – vstupní data pro první část úlohy
- *data_machines.txt* – vstupní data pro druhou část úlohy
- *data_machines_readme.txt* – popis jednotlivých příznaků v datech pro druhou část úlohy.
- *dictionaryFT_train.m*[2] – trénovací funkce pro transformaci slovníkových příznaků
- *dictionaryFT_transform.m*[2] – funkce pro transformaci slovníkových příznaků
- *featureNormalize.m*[1] – normalizace střední hodnoty a škálování příznaků
- *featureTransform.m* – transformace všech příznaků
- *getLinearRegression.m* – vytváří strukturu klasifikátoru pro lineární regresi s implicitním nastavením parametrů
- *gradientDescent*[1] – gradientní sestup jako obecný optimalizační algoritmus
- *linRegCost.m*[1] – vypočítá cenovou funkci a její gradient podle vektoru parametrů.
- *linRegMulti.m*[1] – hlavní spouštěcí skript první části
- *linRegMulti.m* – hlavní spouštěcí skript druhé části
- *linRegPredict.m*[1] – hypotéza lineární regrese
- *plotData.m* – vizualizace dat pro první část
- *plotThetaJ.m* – vizualizace chyby v závislosti na parametrech theta
- *train.m*[1] – výpočet parametrů regresní přímky ze vstupních dat (pomocí gradientního sestupu).

Soubory označené [1] budete doplňovat v rámci první části, soubory 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**

   Soubory *linRegCost.m* a *predict.m*.

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

2. **Gradientní sestup**

   V této části je potřeba doplnit soubor *gradientDescent.m*.

   Gradientní sestup musí umožňovat nastavení více ukončovacích podmínek:

   (a) Počet iterací *iters*

   (b) Minimální chyba *minCost*

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

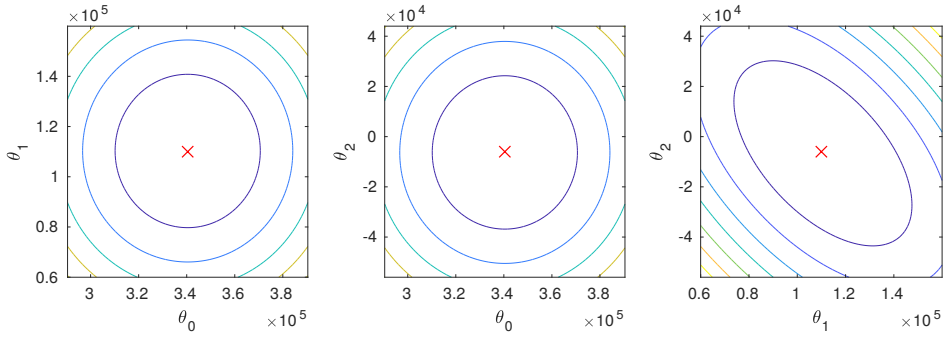
   Struktura *options* může obsahovat 1-N ukončovacích podmínek. Všechny zadané ukončovací podmínky musí být kontrolovány současně. Můžete předpokládat, že vždy bude zadaná alespoň jedna ukončovací podmínka. V souboru *train.m* si prohlédněte jakým způsobem se funkce gradientního sestupu volá.

3. **Škálování příznaků**

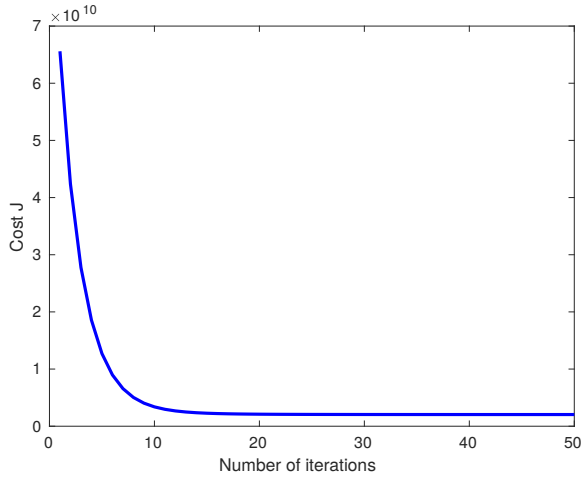
   Normalizujte střední hodnotu a rozptyl příznaků (soubor *featureNormalize.m*).

4. V souboru *linReg_multi.m* **doplňte predikci ceny domu o 1650 čtverečních stopách a 3 místnostech**. Stejnou predikci udělejte pomocí **normálních rovnic**,
5. **Vyladťe parametry gradientního sestupu** ve funkci *train.m* tak, aby konvergoval co nejrychleji.

Po škálování příznaků by vykreslené grafy měly vypadat zhruba následovně:



Obrázek 2: Vývoj chyby se změnou parametrů modelu.



Obrázek 3: Učící křivka

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 výčtového charakteru jako příznaku. 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. 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

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

První složka vektoru odpovídá všem neznámým slovům.