3. cikel: Linearna regresija

Namen:

Implementacija linearne regresije na podanih podatkih. Pri tem uporabite predpripravljene funkcije iz datoteke L_cikel3_linreg.zip (na voljo na spletni strani predmeta med gradivi):

- `ex3.m` – skripta, v kateri boste opravili prvi del vaj
- `ex3_multi` – skripta, v kateri boste opravili drugi del vaje
- `ex3data1.txt` – podatki za linearno regresijo z eno spremenljivko (prvi del vaje)
- `ex3data2.txt` – podatki za linearno regresijo z več spremenljivkami (drugi del vaje)
- `plotData.m` – funkcija za prikaz/izris podatkov
- `computeCost.m` – funkcija za računanje cene linearne regresije
- `gradientDescent.m` – funkcija za izvajanje gradientne metode
- `computeCostMulti.m` – funkcija za računanje cene v več dimenzijah
- `gradientDescentMulti.m` – funkcija za izvajanje gradientne metode v več dimenzijah
- `featureNormalize.m` – funkcija za normalizacijo značilk
- `normalEqn.m` – funkcija za izračun normalne enačbe

Skozi vajo boste uporabljali skripti `ex3.m` in `ex3_multi.m`, ki sta namenjeni pripravi podatkov in klicanju funkcij, ki jih boste napisali. Teh dveh funkcij vam ni potrebno spreminjati. Spreminjali boste le funkcije določene v teh navodilih. Pri vaji boste dopolnili ustrezne datoteke tako, da boste z njimi implementirali linearno regresijo v eni in več dimenzijah.

1. Naloga: Linearna regresija z eno spremenljivko

Vzemimo za primer podjetje, ki želi širiti obstoječo mrežo poslovalnic. Pri odločanju o smeri širitve si želi pomagati s podatki o obstoječi mreži: dobiček in število prebivalcev mesta, kjer je poslovalnica (v vsakem mestu je ena poslovalnica). Podatki obstoječih poslovalnic so zapisani v datoteki `ex3data1.txt` v obliki tabele z dvema stolpcema. V prvem stolpcu so podatki o številu prebivalcev posameznega mesta, v drugem pa so podatki o dobičku v tem mestu. Negativna vrednost dobička seveda pomeni izgubo. Skripta `ex3.m` je pripravljena tako, da podatke samodejno naloži v Matlab.

1. Grafična predstavitev podatkov

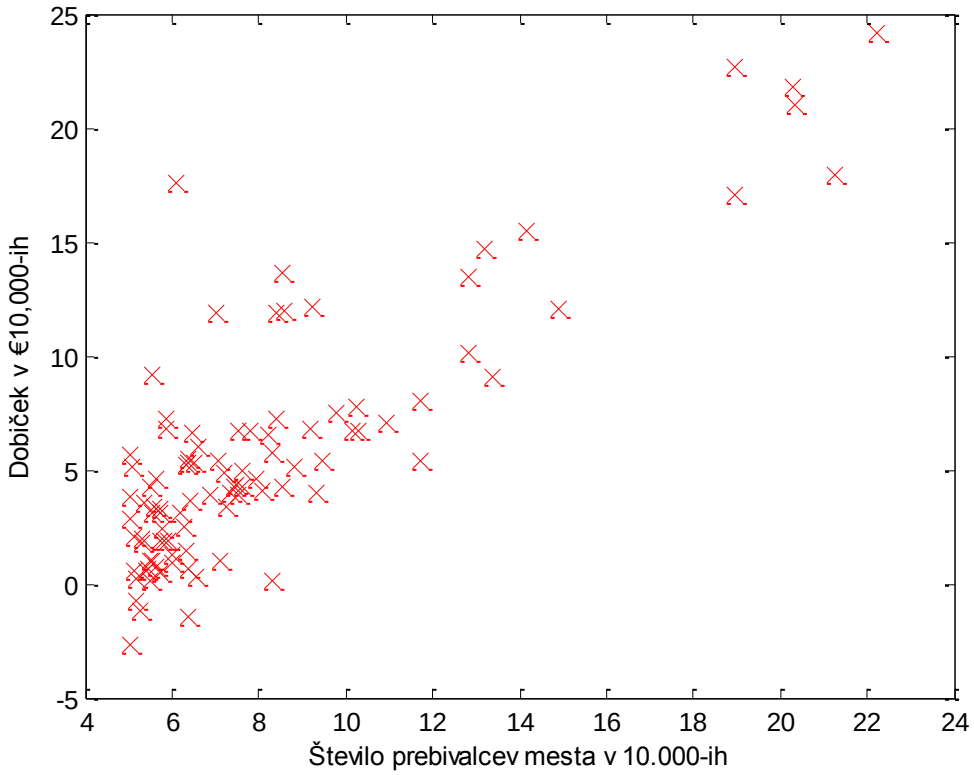
Preden začnemo z reševanjem problema, je običajno koristno, da si podatke vizualiziramo za boljše razumevanje. V tem primeru bomo za vizualizacijo uporabili 2D raztreseni graf (scatter plot). Podatki se naložijo v spremenljivki X (št. prebivalcev) in y (dobiček)

```
data = load('ex3data1.txt');          % branje podatkov
X = data(:, 1); y = data(:, 2);
m = length(y);                        % število učnih primerov
```

V naslednjem koraku kličemo funkcijo `plotData`, ki za dane podatke ustvari 2D raztreseni graf.

```
figure;                                       % odpri novo okno za graf
plot(x, y, 'rx', 'MarkerSize', 10);           % izris grafa
ylabel('Dobicek v €10,000-ih');               % oznaka y osi
xlabel('Stevilo prebivalcev v 10,000-ih');    % oznaka x osi
```

Rezultat vizualizacije podatkov je predstavljen na sliki 1.



Slika 1 – Vizualizacija podatkov

2. Gradientna metoda

V tem delu boste za dane podatke prilagodili parametre linearne regresije θ z uporabo gradientne metode.

Iterativne enačbe

Cilj linearne regresije je minimizacija stroškovne (kriterijske) funkcije

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2,$$

kjer je hipoteza $h_\theta(x)$ podana z linearnim modelom

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

Pri tem so parametri našega modela vrednosti θ_j, ki jih bomo prilagajali tako, da bo stroškovna funkcija $J(\theta)$ minimalna. Ena od možnih algoritmov je seveda iterativni gradientni spust

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Z vsakim korakom gradientnega spusta se parametri θ_j približajo optimalnim vrednostim, ki bodo dosegle minimalni strošek $J(\theta)$.

Opomba: Ker uporabljamo linearni model moramo pri računanju upoštevati tudi člen θ_0. Zato moramo v matriko *X* dodati še en stolpec samih enic na prvo mesto. Tako lahko parameter θ_0 obravnavamo kot značilko.

Implementacija

Potem, ko smo naložili podatke, jih moramo še pripraviti za nadaljnjo obdelavo tako, da matriki *X* dodamo na začetek vrinemo stolpec enic in s tem omogočimo obravnavo člena $\theta_0$pri računanju. Prav tako začetne vrednosti parmetrov θ postavimo na 0, učni parameter α na 0.01, število iteracij pa na 1500.

Izračun stroškovne (kriterijske) funkcije $J(\theta)$

Med izvajanjem gradientnega spusta za minimizacijo stroškovne funkcije $J(\theta)$ lahko konvergenco opazujete s sprotnim računanjem vrednosti stroškovne funkcije. V tej sekciji boste implementirali funkcijo za izračun vrednosti stroškovne funkcije, ki jo boste uporabljali za opazovanje/nadzor konvergence gradientne metode. V ta namen dopolnite kodo v datoteki `computeCost.m`. Pri tem upoštevajte, da sta *X* in *y* matriki, katerih vrstice predstavljajo primere (pare števila prebivalcev in dobička) iz učne množice. Če boste kodo pravilno dopolnili bo za začetne vrednosti parametrov strošek znašal 32.07.

Gradientni spust

V naslednjem koraku dopolnite datoteko `gradientDescent.m` tako, da bo implementiran algoritem gradientnega spusta. V sami datoteki je že pripravljena struktura zanke, ki jo dopolnite z ustreznimi iterativnimi enačbami za parameter θ.

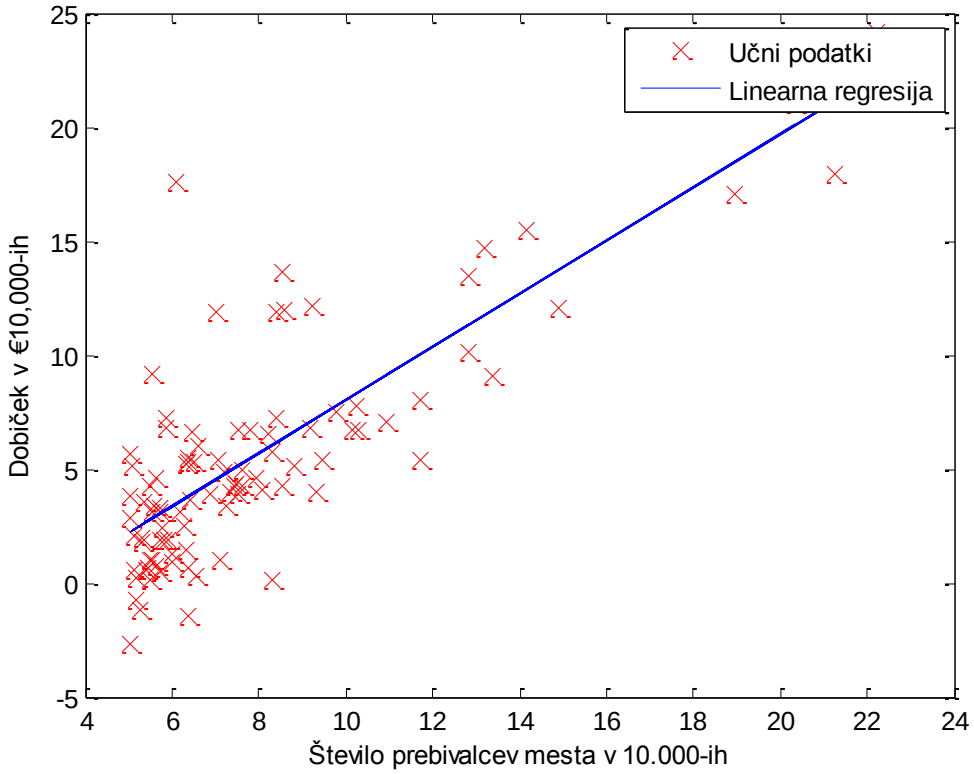
Pri pisanju kode bodite pozorni, kaj skušate optimizirati in kaj se posodablja z iterativnimi enačbami. Upoštevajte, da je strošek $J(\theta)$ odvisen od vektorja θ in ne *X* ter *y*. To pomeni, da se vrednost $J(\theta)$ optimizira s spreminjanjem vektorja θ in ne s spreminjanjem *X* ter *y*.

Primeren način za potrditev, ali gradientni spust deluje pravilno je, da spremljate vrednosti $J(\theta)$ in preverjate ali se v vsakem koraku zmanjšujejo, kot pričakovano. V ta namen v vsaki iteraciji uporabite

predhodno pripravljeno funkcijo `computeCost`. Pri pravilni implementaciji gradientnega spusta se vrednost $J(\theta)$ ne sme v nobeni iteraciji povečati v primerjavi s prejšnjo, temveč mora konvergirati k neki stabilni vrednosti.

Ko boste z gradientnim spustom določili optimalne parametre, si oglejte rezultat (linearizacijo podatkov) na sliki skupaj s podatki (slika 2). Končne vrednosti parametrov θ bodo uporabljene tudi za oceno dobička v dveh primerih mest z različnim številom prebivalcev (35,000 in 70,000).

```
Predict1 = [1, 3.5] * theta
Predict1 = [1, 3.5] * theta
```



Slika 2 – Linearna regresija

3. Vizualizacija stroškovne funkcije $J(\theta)$

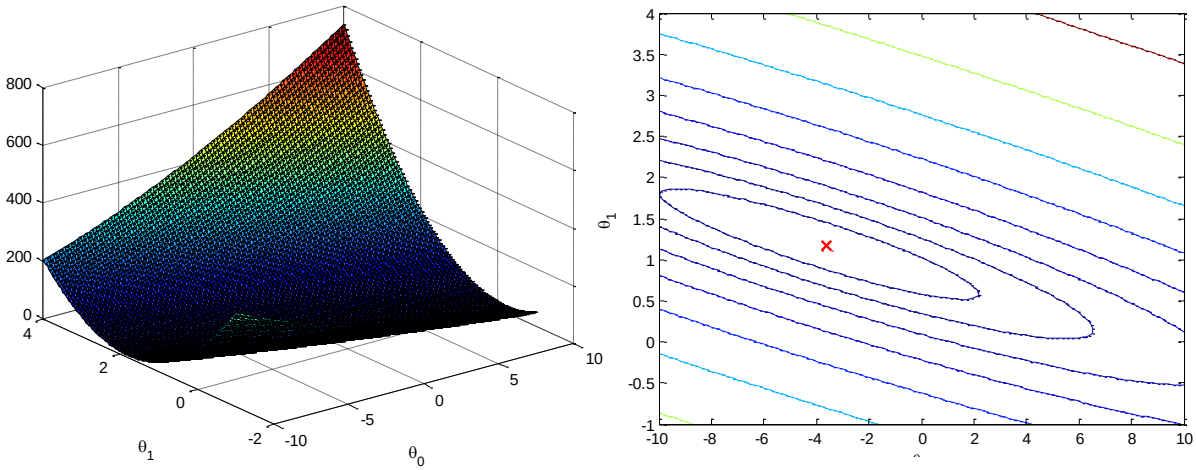
Za boljše razumevanje stroškovne funkcije $J(\theta)$ skripta na koncu tudi izriše strošek nad 2D mrežo vrednosti θ_0 in θ_1. V tem delu ni potrebno dodajati lastne kode; oglejte si zapisano za boljše razumevanje.

V prvem koraku se z uporabo funkcije `computeCost` določijo vrednosti stroškovne funkcije na določenem intervalu/mreži parametrov θ.

```
% inicializacija J_vals na matriko samih ničel
J_vals = zeros(length(theta0_vals), length(theta1_vals));

% Polnjenje matrike J_vals
for i = 1:length(theta0_vals)
    for j = 1:length(theta1_vals)
      t = [theta0_vals(i); theta1_vals(j)];
      J_vals(i,j) = computeCost(X, y, t);
    end
end
```

S tem dobimo 2D polje vrednosti $J(\theta)$, ki jih uporabimo za izris surface in contour grafov (slika3).



Slika 3 – Stroškovna funkcija

Namen teh dveh grafov je pokazati, kako se stroškovna funkcija spreminja s parametri θ. V tem primeru ima stroškovna funkcija obliko sklede in ima globalni minimum. To je lažje opaziti v contour grafu kot pa v 3D površinskem (surface) grafu. Ta minimum je tudi optimalna točka parametrov θ in z vsako iteracijo gradientnega spusta s tej točki bolj približamo.

2. Naloga: Linearna regresija v več dimenzijah

V tem delu boste implementirali linearno regresijo z več spremenljivkami na enakem primeru ocenjevanja vrednosti hiš. Podatki so pripravljeni v datoteki `ex3data2.txt`: v prvem stolpcu so navedene velikosti hiš (v m2), v drugem je navedeno število sob vsake hiše, v tretjem stolpcu pa je navedena cena hiše. Pri reševanju naloge uporabite skripto `ex3_multi.m`.

Normalizacija značilk

Ob zagonu bo skripta `ex3_multi.m` najprej naložila podatke in izpisala nekaj primerov podatkov. Ob pregledu podatkov lahko ugotovite, da so velikosti hiše približno 100-krat večje kot število sob. V takšnih primerih, ko se značilke med seboj po vrednostih razlikujejo za velikostni razred magnitud, lahko njihova normalizacija znatno pohitri algoritem gradientnega spusta. Tako je vaša naloga v tej sekciji, da dopolnite kodo datoteke `featureNormalize.m` tako, da:

- Odštejete srednjo (`mean`) vrednost od vsake značilke v podatkovni zbirki.
- Po odštevanju srednje vrednosti skalirate (delite) vrednosti značilk z njihovo pripadajočo standardno deviacijo.

Standardna deviacija je merilo, koliko variacije je v razponu vrednosti določene značilke (večina podatkovnih točk je zajeta znotraj +/- 2 standardne deviacije od srednje vrednosti). Standardna deviacija je alternativa uporabi razpona vrednosti (max-min). V Matlabu lahko za izračun standardne deviacije uporabite vgrajeno funkcijo `std`. Na primer, vektor $X(:,1)$ zajema vse vrednosti značilk x_1 (velikost hiš) učne množice. Tako ukaz `std`($X(:,1)$) izračuna standardno deviacijo velikosti hiš. V trenutku, ko v skripti kličemo funkcijo `featureNormalize` matriki X še ni bil dodan stolpec enic.

Normalizacijo izvedite za vse značilke in pripravite kodo tako, da bo delovala za podatkovne množice vseh velikosti (poljubno število značilk, primerov). Upoštevajte, da vsaki značilki pripada po en stolpec v X.

OPOMBA: pri normalizaciji značilk je pomembno, da shranimo vrednosti, ki jih potrebujemo za normalizacijo (srednja vrednost, standardna deviacija). Potem, ko zaključimo z učenjem parametrov modela, želimo naučeni model običajno uporabiti za oceno vrednosti hiš, ki jih še nismo videli in jih poznamo zgolj po značilkah podanih z X. Značilke X (velikost in št. sob) pa moramo najprej normalizirati z uporabo srednje vrednosti in standardne deviacije, ki smo ju izračunali prej iz učne množice.

Gradientni spust (gradient descent)

V prejšnji nalogi ste implementirali gradientni spust regresije v eni dimenziji. Edina razlika sedaj je, da matrika X vsebuje še eno dodatno značilko. Funkcija hipoteze ter iterativne enačbe gradientnega spusta ostajajo nespremenjene.

Vaša naloga je, da dopolnite kodo v datotekah `computeCostMulti.m` ter `gradientDescentMulti.m`, s katerima boste določili postopek izračuna stroškovne funkcije in

gradientnega spusta za linearno regresijo z več spremenljivkami. Če vaša programska koda iz prejšnje naloge (ena spremenljivka) že podpira več spremenljivk jo lahko uporabite tudi za to nalogo.

Bodite pozorni na to, da bo vaša programska koda podpirala poljubno število značilk in da je ustrezno (čim bolj) vektorizirana. Pri ugotavljanju števila prisotnih značilk v podatkovni zbirki lahko uporabite funkcijo `` `size(X, 2)' ``.

OPOMBA: V primeru več spremenljivk je stroškovno funkcijo mogoče zapisati v naslednji vektorizirani obliki:

$$J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

Kjer sta *X* in *y* podana kot

$$X = \begin{bmatrix} -(x^{(1)})^T - \\ -(x^{(2)})^T - \\ \vdots \\ -(x^{(m)})^T - \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

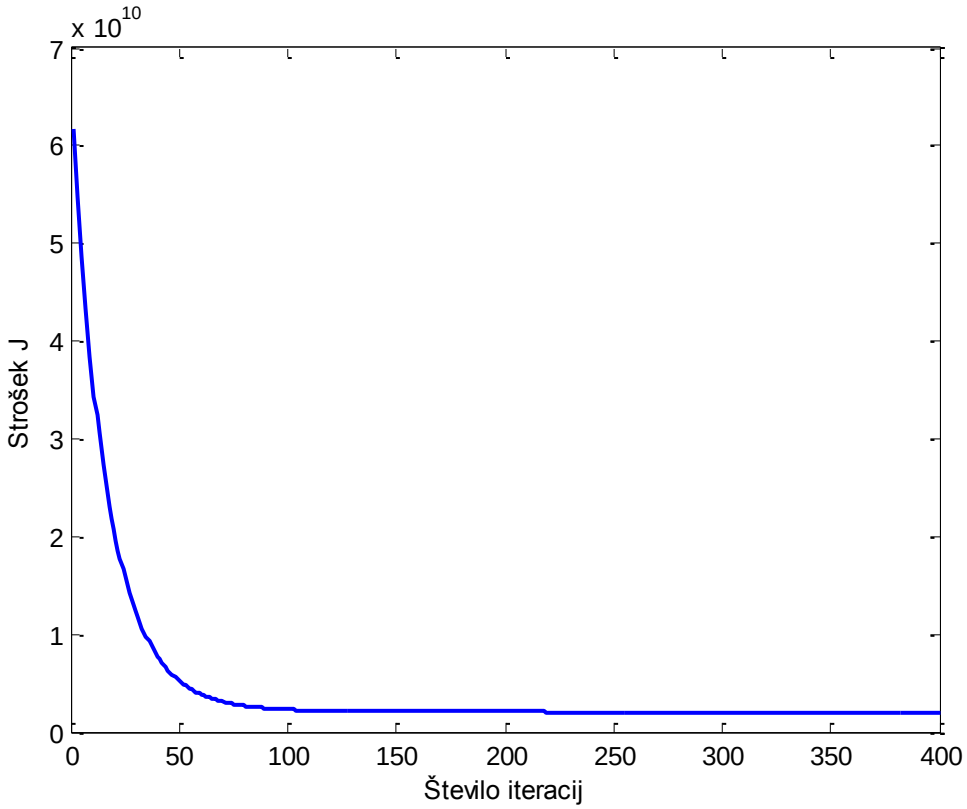
Vektorizirana oblika je učinkovita predvsem takrat, ko delamo z numeričnimi orodji kot je na primer Octave.

Izbira učne stopnje

V tem delu vaje boste preizkusili različne učne stopnje za izbrano učno množico in poiskali učno stopnjo, ki najhitreje konvergira. Učno stopnjo spreminjate v skripti `ex3_multi.m`, kjer ustrezno spremenite del kode, ki se nanaša na izbiro učne stopnje.

V naslednji fazi, skripta `ex3_multi.m`, kliče funkcijo `gradientDescent.m`, ki izvrši približno 50 iteracij gradientnega spusta za izbrano učno stopnjo. Funkcija prav tako vrne zgodovino vrednosti $J(\theta)$ podanih v vektorju J. Po zadnji iteraciji skripta `ex3_multi.m`, izriše vrednosti vektorja J v odvisnosti od števila iteracij.

V primeru, da izberete učno stopnjo znotraj primernega razpona, bo potek vrednosti vektorja J v odvisnosti od števila iteracij podoben poteku, ki je prikazan na sliki 4. Če vaš graf poteka močno odstopa od tistega, ki je prikazan na sliki 4 in se vrednost $J(\theta)$ poveča ali celo eksplodira, potem je potrebno spremenit učno stopnjo in ponovno pognati skripto. Priporočljive vrednosti učne stopnje (α) za testiranje so večkratniki trikratne prejšnje vrednosti (na primer: 0.3, 0.1, 0.03, 0.01, itn.). Prav tako lahko prilagodite število iteracij izvajanja v primeru, da se izkaže, da vam to koristi pri opazovanju trenda v grafu.



Slika 4 – Konvergenca gradientnega spusta s primerno učno stopnjo

OPOMBA: Če je učna stopnja prevelika, lahko to povzroči divergenco (eksplozijo) $J(\theta)$, pri čemer so vrednosti lahko previsoke za numerične kalkulacije z računalnikom. V tem primeru bo Matlab kot rezultat vrnil vrednosti NaN (not a number), ki večinoma pomenijo vrednosti +/- neskončno.

Za primerjavo vpliva različnih učnih stopenj na konvergenco je priporočljivo izrisati na istem grafu J pri več učnih stopnjah (ukaz hold on).

Opazne so spremembe v konvergenci ob spreminjanju učne stopnje. Pri majhni učni stopnji bi morali opaziti, da potrebuje gradientni spust zelo dolgo časa za konvergenco do optimalne vrednosti. Nasprotno pa se lahko ob velikih učnih stopnjah zgodi, da gradientni spust ne konvergira ali pa celo divergira.

Z najprimernejšo učno stopnjo (ki ste jo vi določili) poženite skripto `ex3_multi.m`, ki bo izvedla gradientni spust do konvergence in določila neke končne vrednosti θ. Te vrednosti nato uporabite za oceno vrednosti 183 m2 velike hiše s tremi sobami. S to vrednost boste kasneje preverili pravilnost implementacije normalnih enačb. Pri ocenjevanju vrednosti hiše ne pozabite uporabiti normaliziranih značilk.

Normalna enačba

Zaprta oblika rešitve linearne regresije je

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

Ob uporabi te enačbe ni zahtevano nobeno skaliranje značilk (npr. normalizacija), dobili pa bomo točno rešitev z eno samo kalkulacijo. Dopolnite kodo v normalEqn.m tako, da boste zgornjo enačbo uporabili za izračun θ. Normalizacija značilk v tem primeru ni potrebna, vendar pa morate tudi sedaj dodati stoplpec enic na začetek X

Potem, ko ste določili θ z uporabo normalne enačbe ponovno ocenite vrednost hiše (183 m2, 3 sobe) in primerjajte rezultat s tistim, ki ste ga dobili pri gradientnem spustu – seveda se morata ujemati.