

به نام خدا

**تمرین اول درس یادگیری ماشین**

**عنوان: رگرسیون خطی**

**ا**ستاد درس:

دکتر اکبری

موعد تحویل: ؟؟/09/1400

# 1. رگرسیون خطی

در این تمرین شما رگرسیون خطی چند متغیره[[1]](#footnote-1) را پیاده سازی خواهید کرد. این تمرین طولانی به نظر می رسد، اما در عمل شما تنها حدود 10 خط کد را پیاده سازی می کنید. این تمرین برای آشنایی شما با مکانیزم کتابخانه scikit-learn و پیاده سازی الگوریتم های یادگیری ماشین در پایتون طراحی شده است.

## فایل‎ها و توابع

در این تمرین یک سری فایل و توابع پایتون در اختیار شما قرار دارد:

:test\_linreg\_univariate.py اسکریپتی برای تست کردن رگرسیون خطی تک متغیره[[2]](#footnote-2)

* plotData1D: رسم نمودار نقطه‎ای(پراکندگی)[[3]](#footnote-3) دیتای یک بعدی
* :plotRegLine1D بوسیله تابع plotData1D نمودار نقطه‎ای دیتا را رسم می‎کند و بوسیله پارامترهای یادگرفته شده توسط الگوریتم، خط رگرشن را در همان نمودار رسم می‎کند.
* :visualizeObjective رسم نموار surface و contour تابع هزینه[[4]](#footnote-4) (شما این تابع را تغییر نمی‎دهید)

:test\_linreg\_multivariate.py اسکریپتی برای تست کردن رگرسیون خطی متغیره

:linreg.py اسکریپتی شامل کدهای رگرسیون خطی

:LinearRegression کلاس رگرسیون خطی چند متغیره

* \_\_init\_\_: کانستراکتور کلاس
* :fit تابعی برای یادگیری[[5]](#footnote-5) مدل رگرسیون خطی چند متغیره
* predict: تابعی برای پیش بینی ورودی جدید توسط مدل یاد گرفته شده
* computeCost: محاسبه مقدار تابع هزینه
* gradientDescent: بهینه سازی پارامترهای مدل توسط الگوریتم گرادیان کاهشی[[6]](#footnote-6)

## مجموعه داده‎ها[[7]](#footnote-7) (واقع در فولدر (data

* univariateData.dat: مجموعه داده‎ها برای مسئله رگرسیون تک متغیره
* multivariateData.dat: مجموعه داده‎ها برای مسئله رگرسیون چند متغیره

## بصرسازی داده‎ها[[8]](#footnote-8)

بصرسازی داده‎ها بینش ارزشمندی از مسئله به ما ارائه می دهد، اما اغلب به عنوان بخشی از فرآیند یادگیری ماشین نادیده گرفته می شود. ما با رسم مجموعه داده‎های تک متغیره با استفاده از نمودار پراکندگی 2 بعدی شروع خواهیم کرد. با این حال ما معمولاً با مجموعه داده‎های چند بعدی مواجه هستیم. هنگامی که از دو بعد فراتر می رویم، تجسم بسیار دشوارتر می شود. در چنین مواردی یا باید هر بعد را به طور جداگانه رسم کنیم، یا از تکنیک های کاهش ابعاد (مانند PCA) برای کاهش تعداد ویژگی‎ها[[9]](#footnote-9) استفاده کنیم. بعداً در کلاس در مورد چنین تکنیک هایی صحبت خواهیم کرد.

با اجرای دستورات زیر در مفسر پایتون از داخل دایرکتوریhw1 می توانید داده‎های تک متغیره را در متغیرهای X و y بصورت ماتریس بارگذاری کنید:

import numpy as np

filePath = “data/univariateData.dat”

file = open(filePath, ‘r’)

allData = np.loadtxt(file, delimiter=’,’)

X = np.matrix(allData[:,:-1])

y = np.matrix((allData[:,-1])).T

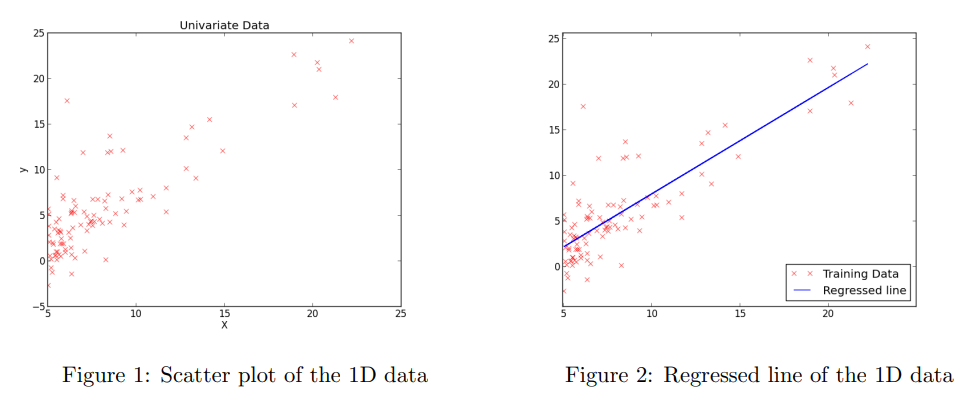
# get the number of instances (n) and number of features (d)

n, d = X.shape

سپس بوسیله‎ی تابع plotData1D نمودار پراکندگی را رسم کنید، خروجی شما باید مانند شکل 1 باشد:

from test\_linreg\_univariate import plotData1D

plotData1D(X, y)



## پیاده سازی[[10]](#footnote-10)

با تکمیل کلاس LinearRegression، رگرسیون خطی چند متغیره را از طریق گرادیان کاهشی اجرا کنید. اسم توابع (APIها) را ابدا تغییر ندهید. قسمت‎هایی از کد که باید تغییر دهید با کامنت‎ "TODO" مشخص شده اند.

الگوریتم رگرسیون خطی بعد از یادگیری پارامترهای مدل به کمک دیتا، آن‎ها را بر روی بردار[[11]](#footnote-11) θ ذخیره می‎کند[[12]](#footnote-12). در این تمرین از گرادیان کاهشی برای یافتن جواب بهینه استفاده می کنیم. دقت کنید که تابع هزینه رگرسیون خطی L2، محدب است، بنابراین الگوریتم گرادیان کاهشی مینیمم مطلق را پیدا می کند.

تابع در رگرسیون خطی تک متغیره (ماتریس x تنها یک ستون دارد( به این شکل است، که همان عرض از مبدا[[13]](#footnote-13) است. برای هندل کردن عرض از مبدا در قالب معادله‎ی (3) می توانیم یک ویژگی جدید به تمام سطرهای داده با مقدار 1 اضافه کنیم. در واقع را به عنوان ضریب در نظر می‎گیریم. برای اضافه کردن عرض از مبدا به کل مجموعه داده‎ها می توانیم ستونی از یک ها را به ماتریس X اضافه کنیم:

X = np.c\_[np.ones((n, 1)), X]

*الگوریتم گرادیان کاهشی برای یافتن مقدار مینیمم تابع فضای* θهای ممکن را جست و جو می‎کند. حلقه for اولیه گرادیان کاهشی برای شما پیاده سازی شده است. شما فقط باید معادله را به روز رسانی کنید. در هر مرحله[[14]](#footnote-14) از گرادیان کاهشی باید به کمک معادله زیر بصورت **همزمان[[15]](#footnote-15)** همه پارامترها را به روز رسانی کنید.

در هر مرحله از گرادیان کاهشی با به روز رسانی θ، به مقدار مینیمم تابع نزدیک‎تر می‎شویم. متغیر نرخ یادگیری[[16]](#footnote-16) است که آن را بسیار کوچک در نظر می‎گیریم (مثلا ). همیشه مقدار اولیه θ را یک مقدار کوچک تصادفی در حوالی صفر انتخاب می‎کنیم (می توان از توزیع نرمال با واریانسی کوچک و میانگین صفر استفاده کرد).

در نهایت تابع هزینه‎ (معادله‎ی 2) را در قالب تابع computeCost در کلاس LinearRegression پیاده سازی می‎کنیم.

**مشکلات متداول:**

* در هر مرحله از گرادیان کاهشی پارامترهای θ را بصورت همزمان آپدیت کنید. یعنی درهر مرحله بعد از به روز رسانی یکی از اعضای بردار θ، بردار θ را بصورت جزیی به روز رسانی نکنید و مقدار را محاسبه کنید. در پایان حلقه بردار θ را کاملا به روز رسانی کنید.
* به یاد داشته باشید در هر مرحله از گرادیان کاهشی شما تنها فضای θهای ممکن را جست و جو می‎کنید و نباید X و y را تغییر دهید.

**تست کردن کد:**

یک راه ساده برای تست کردن کد چاپ کردن مقدار در هر مرحله است. اگر مقدار آن با گذر زمان در هر مرحله بصورت یکنواخت در حال کاهش است یعنی کد شما درست کار می‎کند.

بعد از اتمام پیاده‎سازی کد، مدل خود را بر روی داده‎های univariateData آموزش دهید و سپس تابع plotRegLine1D در فایل test\_linreg\_univariate.py را اجرا کنید.

from test\_linreg\_univariate import plotRegLine1D

from linreg import LinearRegression

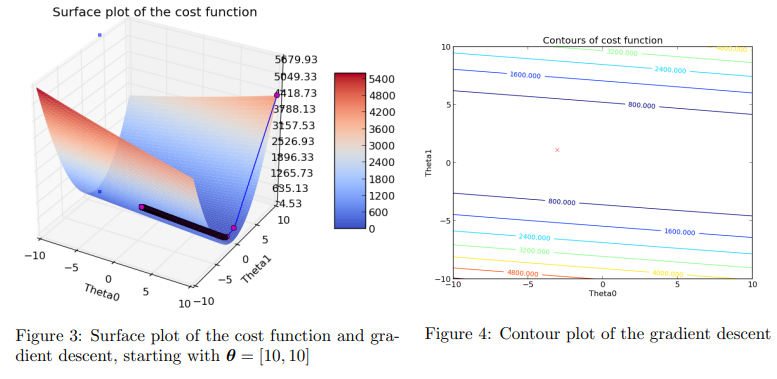
X = np.c\_[np.ones((n,1)), X] # if you didn’t this step already

lr\_model = LinearRegression(alpha = 0.01, n\_iter = 1500)

lr\_model.fit(X, y)

plotRegLine1D(lr\_model, X, y)

خروجی باید یک نمودار مانند شکل 2 باشد.



## درک بهتر گرادیان کاهشی

در این قسمت نیازی به پیاده سازی کد ندارید. برای درک بهتر کد پیاده سازی شده در مرحله قبل، تابع هزینه و مسیر رسیدن به مینیمم آن که توسط الگوریتم گرادیان کاهشی انتخاب شده را رسم می‎کنیم.

برای مجموعه داده‌های تک متغیره می‌توانیم تغییر تابع هدف را در فضای و به‌عنوان یک نمودار سطحی[[17]](#footnote-17) و یک نمودار کانتور[[18]](#footnote-18) ترسیم کنیم تا شکل محدب و نزولی آن را نشان دهیم. خط آبی در شکل 3 مسیر طی شده توسط گرادیان کاهشی را نشان می دهد و نقاط سرخابی مقدار را در هر مرحله نشان می دهد. برای مشاهده نمودار دستور زیر را در ترمینال اجرا کنید:

python test\_linreg\_univariate.py

بعد از رسم نمودار، آن را حرکت دهید تا مسیر طی شده توسط گرادیان کاهشی را به وضوح مشاهده کنید. با تغییر نقطه شروع (یعنی مقدار اولیه θ) نتایج مختلف را بررسی کنید.

اگر کد شما بر روی داده های تک متغیره درست کار می کند، با اجرای دستور زیر در ترمینال آن را روی مجموعه داده‎های چند متغیره آزمایش کنید:

python test\_linreg\_multivariate.py

## سریع‎تر کردن کد پیاده‎سازی شده

اغلب می‌توانیم با برداری کردن[[19]](#footnote-19) کد، الگوریتم‎های یادگیری ماشین را بسیار سریع‌تر و مختصرتر کنیم (اگرچه با مجموعه داده‌های کوچکی که در این مسئله استفاده کردیم تفاوت چندانی ایجاد نمی‌شود). پیاده سازی قبلی خود را برای محاسبه تابع هزینه و به روز رسانی گرادیان را با استفاده از عملیات ماتریسی (مثلاً بدون استفاده از حلقه for در معادله) تغییر دهید.

به عنوان مثال فرم ماتریسی تابع هزینه به شکل زیر است:

where and

# 2. سوال تشریحی

\*\* در حال تکمیل \*\*

**معیار ارزیابی شما:**

1. رگرسیون خطی **(80 نمره)**

* پیاده سازی رگرسیون چند متغیره **(40 نمره)**
* سریع‎تر کردن کد پیاده سازی شده **(20 نمره)**
* گزارش کار **(20 نمره)**

1. سوال تشریحی **(20 نمره)**

**نکات مهم:**

1. کد شما باید به درستی بر روی مجموعه داده‎ها عمل کند، در غیر این صورت نمره کامل نمی‎گیرید.
2. سعی کنید اصول clean code را رعایت کنید و در مواقع لازم کامنت گذاری کنید.
3. مشاهدات و استنباط خود از قسمت‎های بصرسازی داده‎ها، تست کردن کد و درک بهتر گرادیان کاهشی (که نیاز به پیاده سازی نداشت) را در قالب یک گزارش بنویسید. اپلود این قسمت اجباری است، در غیر این صورت نمره تمرین 1 را نمی‎گیرید. (اگر کد شما به توضیحات بیشتر نیاز داشت می توانید به گزارش ضمیمه کنید).
4. نکته مهم در گزارش نویسی و سوال تشریحی روشن بودن پاسخ می‎باشد نه حجم زیاد، اگر فرضی برای حل سوال استفاده می‎کنید حتما آن را ذکر کنید، اگر جواب نهایی عددی است به صورت واضح آن را بیان کنید.
5. هرگونه شباهت در گزارش به منزله تقلب می‎باشد و کل نمره تمرین صفر می‎باشد.
6. گزارش کد و پاسخ تشریحی بصورت فایل pdf باشد.
7. فایل‎ pdf گزارش، تمرین 2 و کدها را بصورت یکجا در قالب یک فایلzip آپلود کنید (نام فایل = شماره دانشجویی).

1. Multivariate linear regression [↑](#footnote-ref-1)
2. Univariate linear regression [↑](#footnote-ref-2)
3. Scatter plot [↑](#footnote-ref-3)
4. Cost function (objective function) [↑](#footnote-ref-4)
5. Train [↑](#footnote-ref-5)
6. Gradient descent [↑](#footnote-ref-6)
7. Data sets [↑](#footnote-ref-7)
8. Visualizing data [↑](#footnote-ref-8)
9. Features [↑](#footnote-ref-9)
10. Implementation [↑](#footnote-ref-10)
11. Vector [↑](#footnote-ref-11)
12. Fit [↑](#footnote-ref-12)
13. Bias term [↑](#footnote-ref-13)
14. Iteration [↑](#footnote-ref-14)
15. Simultaneously [↑](#footnote-ref-15)
16. Learning rate [↑](#footnote-ref-16)
17. Surface plot [↑](#footnote-ref-17)
18. Contour plot [↑](#footnote-ref-18)
19. Vectorizing [↑](#footnote-ref-19)