دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق- گروه مهندسی کنترل

# درس یادگیری ماشین

# پاسخ تمرین امتیازی

| | |
|---|---|
| نام و نام خانوادگی | شیما سادات ناصری |
| شمارهٔ دانشجویی | ۴۰۱۱۲۸۱۴ |
| تاریخ | اسفند ۱۴۰۲ |

# فهرست مطالب

# ۱ سوال اول

برای اثبات اینکه اگر یک ماتریس قطعی مثبت باشد، تمام مقادیر ویژه آن مثبت هستند و بالعکس، می‌توان از تجزیه چالسکی استفاده کرد. تجزیه چالسکی یک روش فاکتورسازی برای ماتریس های هرمیتی، مثبت–معین به حاصلضرب یک ماتریس مثلثی پایین تر و انتقال مزدوج آن است. آندره لوئیس چالسکی دریافت که اگر ماتریس متقارن و قطعی مثبت باشد، کارایی محاسباتی می تواند به طور چشمگیری در حل معادلات خطی بهبود یابد.

## ۱.۱ اثبات بخش اول: اگر یک ماتریس مثبت معین باشد، تمام مقادیر ویژه آن مثبت هستند

- ابتدا باید ماتریس مثبت معین را تعریف کنیم: اگر برای هر بردار غیر صفر $x$، فرم کوادراتیک آن $x^TAx > 0$ باشد، ماتریس مربعی $A$ مثبت معین است.
- تجزیه چالسکی نیز به این صورت تعریف می‌شود: برای هر ماتریس قطعی واقعی، متقارن و مثبت $A$، یک ماتریس مثلثی پایین تر $L$ با ورودی های مورب واقعی و مثبت وجود دارد به طوری که $A = LL^T$.
- فرض می‌کنیم $\lambda$ یک مقدار ویژه از $A$ با بردار ویژه مربوطه $v$ است ($Av = \lambda v$)، جایی که $v \neq 0$. حال اگر عبارت $v^TAv$ را در نظر بگیریم، با جایگزین کردن $Av = \lambda v$، عبارت $v^TAv = v^T\lambda v = \lambda v^Tv$ بدست می‌آید.

  از آنجایی که $v \neq 0$، $v^Tv > 0$، که مجموع مربعات اجزای $v$ است. از تعریف مثبت معین بودن یک ماتریس داریم: $v^TAv > 0$. درنتیجه خواهیم داشت: $\lambda v^Tv > 0$. این به معنای $\lambda > 0$ است زیرا $v^Tv > 0$. بنابراین، هر مقدار ویژه $\lambda$ از $A$ مثبت است.

## ۲.۱ اثبات بخش دوم: اگر همه مقادیر ویژه یک ماتریس مثبت باشند، ماتریس مثبت معین است.

- برای اثبات این بخش نیاز به استفاده از یک قضیه به نام قضیه طیفی یا $SpectralTheorem$ داریم که به این صورت تعریف می‌شود: برای یک ماتریس متقارن $A$، یک ماتریس متعامد $Q$ و یک ماتریس مورب $D$ وجود دارد به طوری که $A = QDQ^T$، که در آن عناصر قطری $D$ مقادیر ویژه $A$ هستند.
- حال فرض می‌کنیم همه مقادیر ویژه $A$ مثبت هستند، به این معنی که تمام ورودی های مورب $D$ مثبت هستند. برای هر بردار غیر صفر $x$ ، با جایگزینی عبارت $A = QDQ^T$ در عبارت اصلی $x^TAx$، عبارت $x^TAx = x^TQDQ^Tx$ بدست می‌آید. حال اگر داشته باشیم:$y = Q^Tx$؛ درنتیجه عبارت $x^TAx = y^TDy$ حاصل می‌شود. از آنجایی که $D$ مورب با ورودی های مثبت است، خواهیم داشت: $y^TDy = \sum \lambda_i y_i^2 > 0$؛ زیرا هر $\lambda_i > 0$ بوده و $y$ یک بردار صفر نیستند. ($x$ صفر نیست و $Q$ معکوس است). بنابراین، $x^TAx > 0$ برای هر بردار غیر صفر $x$صدق کرده و ماتریس $A$ مثبت معین است.

در نتیجه ثابت شد که اگر یک ماتریس مثبت معین باشد، تمامی مقادیر ویژهٔ آن نیز مثبت است و بلعکس. ■

## ۲ سوال دوم

ماتریس $A$ را در نظر می‌گیریم که نشان دهنده یک دوران در صفحه است، که می تواند با یک فرآیند تناوبی مرتبط باشد زمانی که زاویه دوران با میزان تناوب مطابقت دارد. . یک مثال رایج از چنین ماتریسی، ماتریس دوران برای یک چرخش با یک زاویه $\theta$ در صفحه است:

$$A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

این ماتریس ماهیت تناوبی دارد زیرا دوران، سیستم را پس از افزایش $2\pi$ در $\theta$ به جهت اولیه خود بازمی گرداند.
مقادیر ویژه $\lambda$ ماتریس $A$ به صورت زیر بدست می‌آیند:

$$\det(A - \lambda I) = 0$$

درنتیجه پس از جایگذاری خواهیم داشت:

$$\det \begin{pmatrix} \cos(\theta) - \lambda & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) - \lambda \end{pmatrix} = 0$$

حال برای بدست آوردن بردارهای ویژه $v$ از معادله زیر استفاده می‌کنیم:

$$(A - \lambda I)v = 0$$

مقادیر ویژه ماتریس $A$ به صورت $e^{-i\theta}$ و $e^{i\theta}$ بدست می‌آیند که ماهیت مختلط مقادیر ویژه را برای ماتریس‌های دورانی در دو بعد نشان می‌دهند. این مقادیر ویژه، مختلط‌های مزدوجی هستند که ماهیت دوره‌ای$transform$ در ماتریس‌های دورانی را نشان می دهد.
حال اگر به صورت زیر مقادیر ویژه را بازنویسی کنیم:

$$\lambda_1 = \cos(\theta) + i\sin(\theta) \quad \lambda_2 = \cos(\theta) - i\sin(\theta)$$

بردارهای ویژه به صورت زیر بدست می‌آیند:

$$v_1 = \begin{bmatrix} 1 \\ i \end{bmatrix} \quad v_2 = \begin{bmatrix} 1 \\ -i \end{bmatrix}$$

برای یک ماتریس دوران، بردارهای ویژه به معنای متعارف بردارهایی در صفحه حقیقی نیستند بلکه ماهیت تبدیل دورانی را منعکس می کنند.

## ۳ سوال سوم

با توجه به یک ماتریس مربع $A$ به اندازه $n \times n$، ماتریس مودال آن $M$ از بردارهای ویژه آن به عنوان ستون تشکیل شده است. اگر $A$ دارای $n$ بردار ویژه مستقل خطی باشد، آنگاه $M$ غیر مفرد است و دترمینان آن غیر صفر است.

دترمینان یک ماتریس حاصل ضرب مقادیر ویژه آن است و برای $M$، هر مقدار ویژه با یک بردار ویژه مطابقت دارد. بنابراین، اگر همه مقادیر ویژه غیر صفر باشند، دترمینان ماتریس مودال نیز غیرصفر است، که نشان دهنده معکوس پذیری ماتریس است.

اگر $\lambda_1\lambda_2...\lambda_n$ مقادیر ویژه باشند، دترمینان $M$ برابر است با:

$$\det(M) = \lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_n$$

هرچند ماتریس مودال $M$ لزوماً مقادیر ویژه را به عنوان ورودی‌های قطری خود ندارد و دترمینان آن به بردارهای ویژه انتخاب شده بستگی دارد. با این وجود، اگر ماتریس $A$ قابلیت قطری شدن داشته باشد، دترمینان $M$ حاصلضرب مقادیر ویژه غیر صفر $A$ خواهند بود، اگر بردارهای ویژه نرمال شده باشند.

## ۴ سوال چهارم

از میان روش‌های مخلتف، روش تجزیه SVD یک رویکرد قوی برای یافتن راه حل حداقل مربعات ارائه می دهد. این فرایند به شرح زیر است:

- برای سیستم بیش از حد تعیین شده که با یک معادله ماتریسی $Ax = b$ تعریف شده که در آن $A$ یک ماتریس m$\times n$ با $m > n$ است، $A$ با استفاده از SVD تجزیه می‌شود و خواهیم داشت:

$$A = U\Sigma V^T.$$

در اینجا، $U$ یک ماتریس متعامد $m \times m$، $\Sigma$ یک ماتریس $m \times n$ مورب با اعداد واقعی غیر منفی روی مورب (مقادیر منفرد) است و $V$ یک ماتریس متعامد $n \times n$ است.

درنتیجه سیستم اصلی $Ax = b$ به سیستم $\Sigma y = U^T b$ تبدیل می‌شود، به طوری که $y = V^T x$. این از ویژگی های متعامد $U$ و $V$ استفاده می‌کند، که در آن $U^T U = I$ و $V^T V = I$ است.

- حال پس از تجزیه، باید شبه معکوس ماتریس پیدا شود که با $A^+ = V\Sigma^+ U^T$ محاسبه می‌گردد، که در آن $\Sigma^+$ شبه معکوس $\Sigma$ است. حال می‌توان با کمک حل حداقل مربعات که به صورت $x = A^+ b$ تعریف می‌شود، نرم اقلیدسی ($||Ax - b||_2$) را به حداقل رساند .

# ۵ سوال پنجم

ماتریس های مثلثی، هر دو بالا و پایین، غیر منفرد (معکوس) هستند اگر همه عناصر مورب غیر صفر باشند. دلیل این امر این است که دترمینان یک ماتریس مثلثی (بالا یا پایین) برابر است با حاصل ضرب ورودی های مورب آن. اگر هیچ یک از این ورودی ها صفر نباشد، دترمینان غیر صفر است و بنابراین ماتریس غیر مفرد است.

برای نشان دادن این موضوع در پایتون، از دو ماتریس زیر استفاده می‌کنیم:

$$\text{ماتریس بالا مثلثی :} \begin{bmatrix} 1.255 & 1.809 & 1.483 & 1.980 & 1.647 \\ 0 & 1.661 & 1.949 & 1.450 & 1.994 \\ 0 & 0 & 1.346 & 1.909 & 1.278 \\ 0 & 0 & 0 & 1.784 & 1.186 \\ 0 & 0 & 0 & 0 & 1.366 \end{bmatrix}$$

$$\text{ماتریس پایین مثلثی :} \begin{bmatrix} 1.958 & 0 & 0 & 0 & 0 \\ 1.563 & 1.423 & 0 & 0 & 0 \\ 1.448 & 1.880 & 1.791 & 0 & 0 \\ 1.596 & 1.560 & 1.719 & 1.193 & 0 \\ 1.720 & 1.324 & 1.578 & 1.848 & 1.994 \end{bmatrix}$$

سپس دترمینان آن را محاسبه می‌کنیم تا بررسی کند که آیا غیرصفر است یا خیر. اگر دترمینان واقعاً غیر صفر باشد، تأیید می کند که ماتریس غیر مفرد است.

```python
# -*- coding: utf-8 -*-
"""ML_HW_Extra_Q5.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1orKdc8CQiNpdYTEGEpfmumM03-PmPjib
"""

import numpy as np

# Function to generate a random triangular matrix and check if it's non-singular
def is_triangular_matrix_non_singular(size, matrix_type):
    # Generate a random triangular matrix with non-zero diagonal elements
    if matrix_type == 'upper':
```

```python
        A = np.triu(np.random.rand(size, size) + 1, 0)  # Shift by +1 to avoid zero diagonal entries
    elif matrix_type == 'lower':
        A = np.tril(np.random.rand(size, size) + 1, 0)  # Shift by +1 to avoid zero diagonal entries
    else:
        raise ValueError("matrix_type must be 'upper' or 'lower'")

    # Calculate determinant
    det = np.linalg.det(A)

    # Check if the determinant is non-zero
    is_non_singular = det != 0

    return is_non_singular, det, A

# Check for an upper triangular matrix
upper_non_singular, upper_det, upper_matrix = is_triangular_matrix_non_singular(5, 'upper')

# Check for a lower triangular matrix
lower_non_singular, lower_det, lower_matrix = is_triangular_matrix_non_singular(5, 'lower')

upper_non_singular, upper_det, upper_matrix, lower_non_singular, lower_det, lower_matrix
```

Code 1: کد سوال 5

کد پایتون تأیید کرده است که هر دو ماتریس مثلثی بالا و پایین تولید شده غیرمفرد هستند. دترمینان های این ماتریس ها غیر صفر هستند، به طوری که ماتریس مثلثی بالایی دارای دترمینان تقریباً ۸۴.۶ و ماتریس مثلثی پایینی دارای دترمینان تقریباً ۸۶.۱۱ است. از آنجایی که هیچ یک از این دترمینان ها صفر نیستند، تأیید می کند که هر دو ماتریس در واقع غیر مفرد هستند.

## ۶ سوال ششم

کد زیر که برای بررسی این موضوع نوشته شده، دو روش برای انجام ضرب ماتریس-بردار با استفاده از کتابخانه $NumPy$ پایتون را مقایسه می‌کند: یک روش کندتر و غیربردار با استفاده از حلقه for، و یک روش سریع‌تر و بردار با استفاده از توابع بهینه‌سازی شده $NumPy$.
ماتریس و بردار به ترتیب با ابعاد $20 \times 10$ به صورت تصادفی تولید می شوند.

```python
# -*- coding: utf-8 -*-
"""ML_HW_Extra.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1HF3-ZK3uzywZwTlD4XYiOtNNkRv4bizt
"""

import numpy as np

m, n = 20, 10
A = np.random.rand(m, n)
k = np.random.rand(n)
# Vectorized matrix-vector multiplication
p_vectorized = np.dot(A, k)

p_vectorized

import timeit

# Original for-loop implementation
def original_implementation():
    p = np.zeros(m)
    for i in range(n):
        p += A[:, i] * k[i]
    return p

# Vectorized implementation
def vectorized_implementation():
```

```python
    return np.dot(A, k)

p_original = original_implementation()
p_vectorized = vectorized_implementation()
results_are_close = np.allclose(p_original, p_vectorized)

original_time = timeit.timeit(original_implementation, number=1000)
vectorized_time = timeit.timeit(vectorized_implementation, number=1000)

print(f"Looped time: {original_time}")
print(f"Vectorized time: {vectorized_time}")
print(f"Are the results close? {results_are_close}")
```

Code 2: کد سوال 6

$$Looped time : 0.028395261000000005$$

$$Vectorized time : 0.0011631039999997483$$

$$Are the results close? True$$

متغیر بولی `results_are_close` تأیید می‌کند که هر دو پیاده‌سازی نتایج یکسانی را در یک تلورانس مشخص محاسبه می‌کنند. زمان‌های اجرا، که برای ۱۰۰۰ تکرار با استفاده از `timeit.timeit()` اندازه‌گیری می‌شوند، نشان می‌دهند که پیاده‌سازی برداری شده به‌طور قابل‌توجهی سریع‌تر از پیاده‌سازی اصلی برای حلقه است.