بسمه تعالی

دانشکده مهندسی کامپیوتر

آذر ۱۴۰۰

مبانی هوش محاسباتی

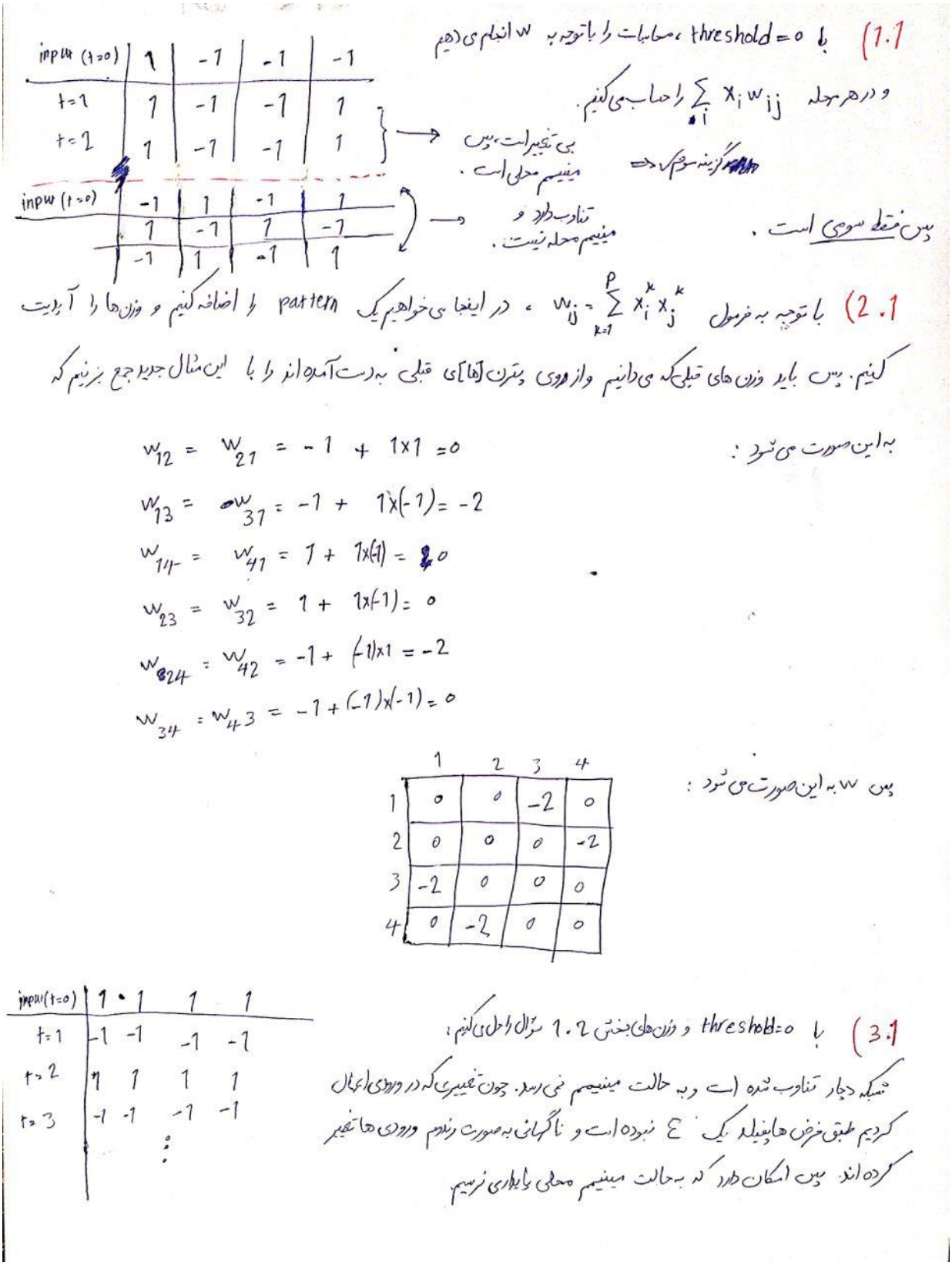
نام استاد: دکتر مزینی

تمرین سوم

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

۱. پاسخ سوال اول



1.1) با threshold = 0 ، محاسبات را با توجه به w انجام می‌دهیم

و در هر مرحله $\sum_i x_i w_{ij}$ را حساب می‌کنیم.

input (t=0)	1	-1	-1	-1
t=1	1	-1	-1	1
t=2	1	-1	-1	1

→ بی تغییرات، پس مینیمم محلی است.

⇒ گزینه سوم درست

input (t=0)	-1	1	-1	1
	1	-1	1	-1
	-1	1	-1	1

→ تناوب دارد و مینیمم محلی نیست.

پس فقط سومی است.

2.1) با توجه به فرمول $w_{ij} = \sum_{k=1}^{p} x_i^k x_j^k$ ، در اینجا می‌خواهیم یک pattern را اضافه کنیم و وزن‌ها را آپدیت کنیم. پس باید وزن‌های قبلی که می‌دانیم و از روی پترن‌های قبلی به دست آمده‌اند را با این مثال جدید جمع بزنیم که به این صورت می‌شود:

$$w_{12} = w_{21} = -1 + 1\times 1 = 0$$

$$w_{13} = w_{31} = -1 + 1\times(-1) = -2$$

$$w_{14} = w_{41} = 1 + 1\times(-1) = 0$$

$$w_{23} = w_{32} = 1 + 1\times(-1) = 0$$

$$w_{24} = w_{42} = -1 + (-1)\times 1 = -2$$

$$w_{34} = w_{43} = -1 + (-1)\times(-1) = 0$$

پس w به این صورت می‌شود:

	1	2	3	4
1	0	0	-2	0
2	0	0	0	-2
3	-2	0	0	0
4	0	-2	0	0

3.1) با threshold = 0 و وزن‌های بخش 2.1 سوال را حل می‌کنیم:

input (t=0)	1	-1	1	1	1
t=1	-1	-1	-1	-1	
t=2	1	1	1	1	
t=3	-1	-1	-1	-1	
	⋮				

همینکه دچار تناوب شده است و به حالت مینیمم نمی‌رسد. چون تغییری که در ورودی‌ها ایجاد کردیم طبق فرض‌های فیلد یک ε نبوده است و ناگهانی به صورت رندوم ورودی‌ها تغییر کرده‌اند. پس امکان دارد که به حالت مینیمم محلی پایداری نرسیم

۲. پاسخ سوال دوم

۲.۱) کد مربوط به این بخش در نوتبوک زده شده است و با الگوریتم hebbian، بر اساس سه ورودی داده شده وزنهایی را پیدا کرده و ذخیره میکند.

۲.۲) از np.sign برای اعمال این تابع فعالسازی استفاده میکنیم. چون برای ورودی های کمتر از صفر -۱ و برای ورودی های مثبت، ۱ را بر میگرداند. برای محاسبه پایداری شبکه در ورودی خاصی، باید وقتی آن را به شبکه میدهیم، خروجی دقیقا با ورودی برابر باشد گویا تحریکی صورت نگرفته است. که در این قسمت با چک کردن برابر بودن خروجی ها و ورودی ها (np.array_equal) این کار را انجام دادیم و دیدیم که شبکه به درستی عمل کرده است.

۲.۳) بله همه ی آن ها در شبکه ذخیره شده اند و مطابق قسمت قبل، پایدار هستند. البته برای این قسمت صرفا به ضرب و استفاده از sign اکتفا نمیکنیم و تابعی برای چک کردن ورودی ها به صورت مرحله به مرحله و چک کردن ثابت ماندن آن در حالت پایداری می نویسیم. که در بخش بعد هم که ورودی جدیدی داریم میتوانیم از این تابع استفاده کنیم.

دلیل آن که این سه ورودی هم در شبکه ذخیره شده اند و پایدار هستند، این است که آن ها قرینه‌ی (منفیِ) ورودی های قبلی هستند. و میدانیم که یکی از ویژگی/ضعف های شبکه هاپفیلد این است که در آن تمام وزن ها قرینه هستند و لذا هر ورودی که ذخیره میکند، قرینه آن هم ذخیره می شود.

۲.۴) با دادن این ورودی به شبکه میبینیم که شبکه به حالت پایداری نمیرسد. درواقع بین دو حالت همواره تناوب میکند. چون چندین ورودی نورون را اگر باهم به صورت رندوم تغییر دهیم (synchronous)، ممکن است این اتفاق بیافتد. هاپفیلد ادعا میکند که درصورت تغییرات دیفرانسیلی در ورودی ها میتوانیم آن ها را حتما به حالات پایدار برسانیم. و لزوما تمام حالات پایدار، نقاط ثابتی نیستند.

۳. پاسخ سوال سوم

مطابق الگوریتم هاپفیلد شبکه را آموزش داده (وزن ها را می یابیم) و تست ها را به آن می دهیم. تصاویر خروجی بسیار با کیفیت هستند و دقت هم در هر خروجی ۱۰۰ به دست آمد. در نوتبوک این موارد مشخص است و کارکرد توابع توضیح داده شده اند. دقت های خروجی :

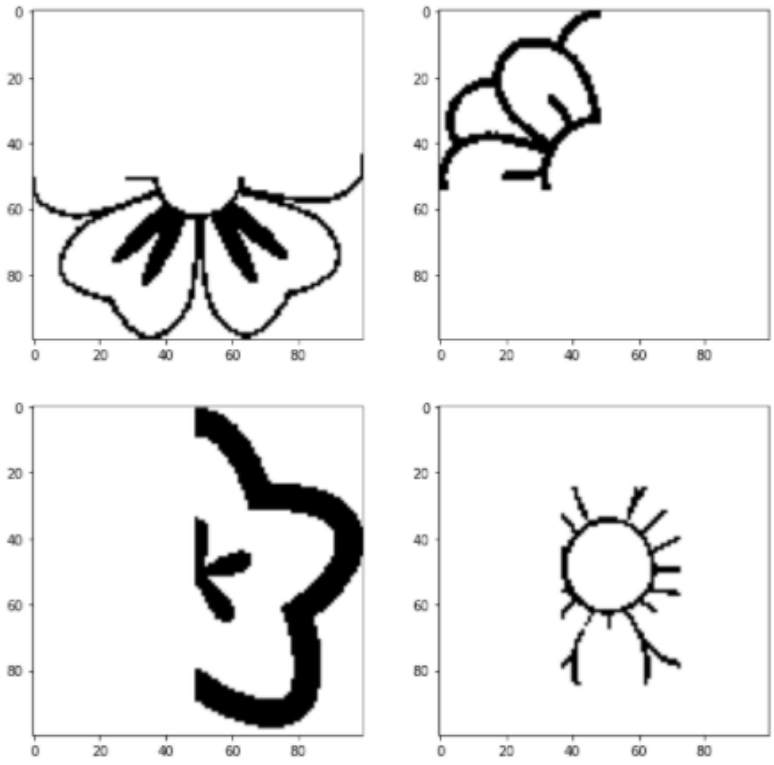
```
similarity of updated test1.jpg with 1.jpg is 100.0 %
similarity of updated test1.jpg with 2.jpg is 71.50000000000001 %
similarity of updated test1.jpg with 3.png is 58.29 %
similarity of updated test1.jpg with 4.jpg is 75.84 %
---------------------------------
similarity of updated test2.jpg with 1.jpg is 71.50000000000001 %
similarity of updated test2.jpg with 2.jpg is 100.0 %
similarity of updated test2.jpg with 3.png is 60.830000000000005 %
similarity of updated test2.jpg with 4.jpg is 72.82 %
---------------------------------
similarity of updated test3.png with 1.jpg is 58.29 %
similarity of updated test3.png with 2.jpg is 60.830000000000005 %
similarity of updated test3.png with 3.png is 100.0 %
similarity of updated test3.png with 4.jpg is 67.17 %
---------------------------------
similarity of updated test4.jpg with 1.jpg is 75.84 %
similarity of updated test4.jpg with 2.jpg is 72.82 %
similarity of updated test4.jpg with 3.png is 67.17 %
similarity of updated test4.jpg with 4.jpg is 100.0 %
---------------------------------
```

به علت هم سایز نبودن داده های تست، آن ها را به صورت دستی با حاشیه سفید پر کردم.

دلیل این که دقت مقایسه با فایل های بازنده هم بالاست، سفید بودن بخش های زیادی از عکس هاست و مشکل خاصی وجود ندارد. دقت ۱۰۰ هم غیر منطقی نیست چون تعداد نورون ها بسیار بالاست و نسبت تعداد پترن به نورون خیلی کمتر از ۰.۱۳۸ است و این باعث می شود شبکه هاپفیلد عالی عمل کند.

برای بهتر عمل کردن شبکه، هنگام آموزش و همچنین هنگام ورودی دادن، پیکسل های از یک حدی سیاه تر (rgb<200) را با −۱ و پیکسل های سفید را با ۱ مدل کردم.

عکس های تست پیش از دادن به شبکه:



عکس های تست پس از دادن به شبکه: