بسمه تعالی



دانشکده مهندسی کامپیوتر

آذر **1400**

**مبانی هوش محاسباتی**

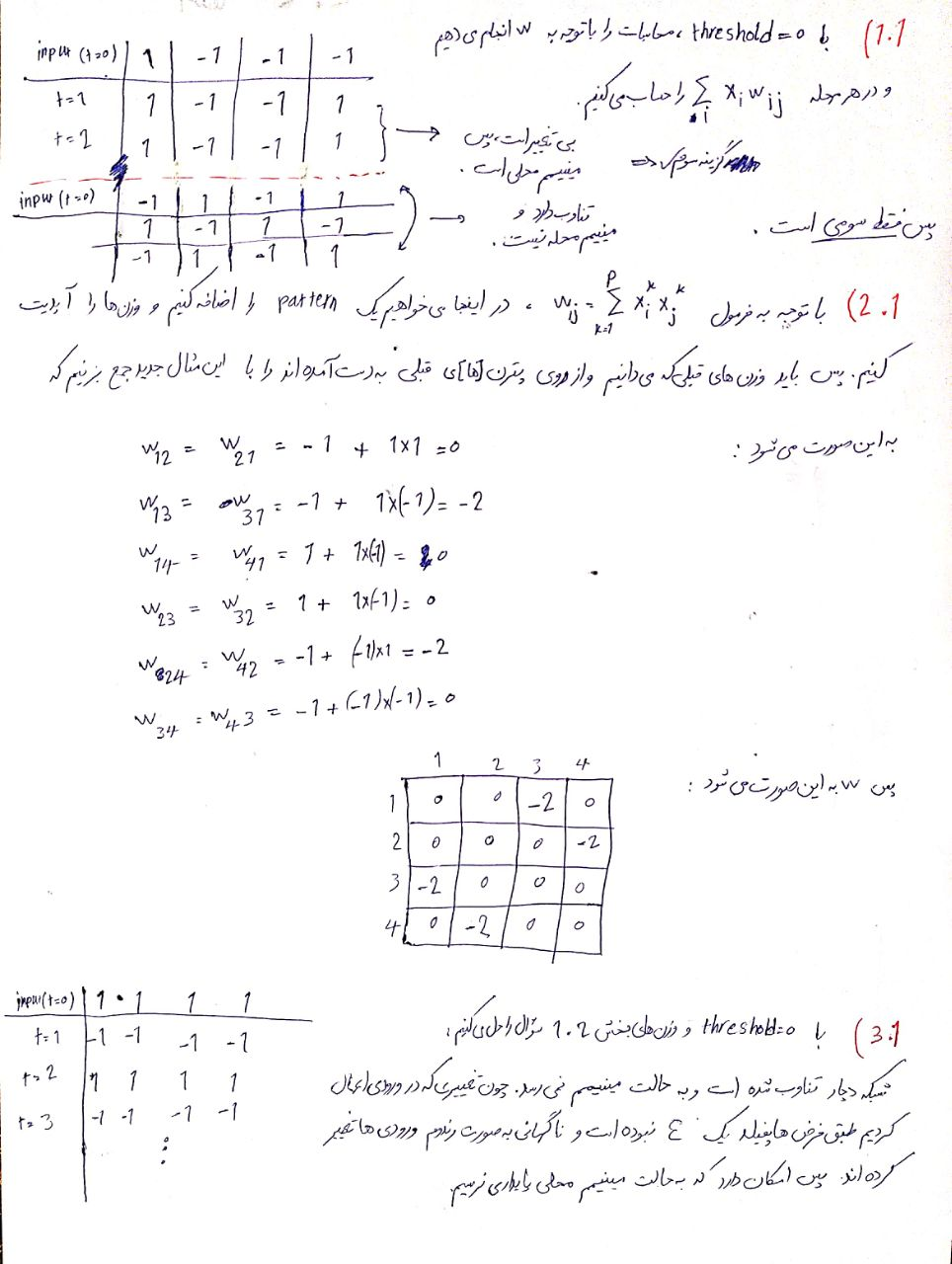
نام استاد: دکتر مزینی

تمرین سوم

آرمان حیدری

شماره دانشجویی: **97521252**

# پاسخ سوال اول



# پاسخ سوال دوم

2.1) 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.

2.2) از np.sign برای اعمال این تابع فعالسازی استفاده میکنیم. چون برای ورودی های کمتر از صفر -1 و برای ورودی های مثبت، 1 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.

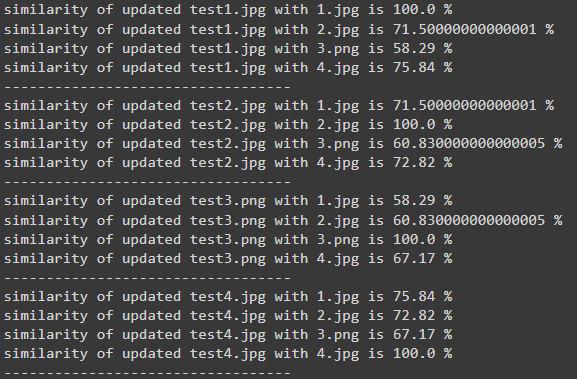
2.3) بله همه ی آن ها در شبکه ذخیره شده اند و مطابق قسمت قبل، پایدار هستند. البته برای این قسمت صرفا به ضرب و استفاده از sign اکتفا نمیکنیم و تابعی برای چک کردن ورودی ها به صورت مرحله به مرحله و چک کردن ثابت ماندن آن در حالت پایداری می نویسیم. که در بخش بعد هم که ورودی جدیدی داریم میتوانیم از این تابع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

2.4) 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.

# 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 100 به دست آمد. در نوتبوک این موارد مشخص است و کارکرد توابع توضیح داده شده اند. دقت های خروجی :

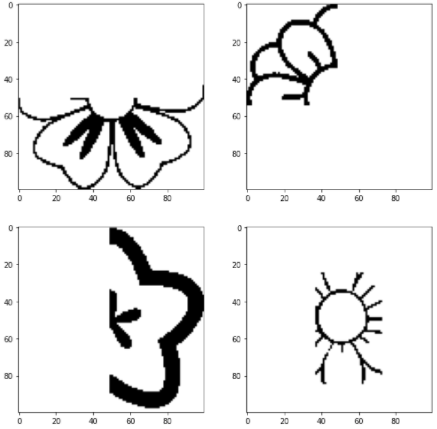


به علت هم سایز نبودن داده های تست، آن ها را به صورت دستی با حاشیه سفید پر کردم.

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 100 هم غیر منطقی نیست چون تعداد نورون ها بسیار بالاست و نسبت تعداد پترن به نورون خیلی کمتر از 0.138 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 -1 و پیکسل های سفید را با 1 مدل کردم.

عکس های تست پیش از دادن به شبکه:



عکس های تست پس از دادن به شبکه:

