Sayed Mohammad Fatemi
Student number: 40127793
HW2
Computational Intelligence

سوال 1)

(1.1
لزوما خیر، چون بر اساس توزیع داده ممکن است داده های بیشتر هم که بدهیم توزیع درستی نداشته باشند. یک مثال ساده مثلا اگر در تشخیص سگ و گربه تعداد زیادی سگ داشته باشیم هرچقدر هم که دیتا زیاد باشد مدل به سمت تشخیص سگ اورفیت میکند و در تشخیص گربه عاجز است. دلیل دیگر اصلا پیچیدگی خود مدل است که اگر خیلی پیچیده باشد نسبت به دیتای ورودی آن ، هرچقدر هم دیتا بیاوریم اورفیت میکند. در ضمن اگر آموزش زیاد طولانی شود هم همچنان مشکل اورفیت ممکن است به وجود بیاید.

(1.2
درست است، زیرا طبق تعریف اورفیت این مشکل در زمانی پیدا میشود که مدل روی دیتایی که ترین شده خیلی دقیق شده باشد ولی وقتی به سراغ دیتای دنیای واقعی می رویم(مثلا اعتبارسنجی یا تست) مذل دیگر خوب عمل نمی کند و قابلیت generalization ندارد.

(1.3
درست است زیرا L1 باعث می‌شود برخی از وزن‌ها دقیقاً به صفر برسند و در نتیجه برخی ویژگی‌ها حذف شوند، که این به معنای کاهش ابعاد مدل و انتخاب ویژگی است. این خاصیت باعث می‌شود مدل به صورت اسپارس (sparse) شود و فقط ویژگی‌های مهم‌تر باقی بمانند. اما **L2** وزن‌ها را به سمت صفر کوچک می‌کند اما هیچ‌کدام را دقیقاً صفر نمی‌کند، بنابراین همه ویژگی‌ها در مدل باقی می‌مانند ولی با وزن‌های کوچکتر. این باعث کاهش پیچیدگی مدل و جلوگیری از بیش‌برازش می‌شود اما حذف ویژگی انجام نمی‌دهد.

(1.4
غلط است. زیرا در زمان استنتاج اگر بخواهیم دراپ اوت بزنیم ممکن است نتایج مختلفی بگیریم که اصلا مطلوب نیست، بلکه فقط در زمان آموزش برای دلایلی مثل اینکه وزن همه نورن ها بهتر آپدیت شوند و فیچر های بهتری پیدا شوند و حتی گاهی مدل زیاد پیچیده نشود استفاده می شود.

(1.5
غلط. یک روش regularization هست منتها برای جلوگیری از overfit است.

(1.6
درست است. زیرا همه ی داده ها را باهم یا به صورت دسته ای نمی بینیم و لزوما داریم هر دفعه یک نقطه را در نظر میگیریم که همان روش گرادیان نزولی تصادفی می شود.

(1.7
غلط. نرمال سازی اتفاقا در activation function هایی مثل sigmoid یا tanh به آموزش سریع تر کمک میکند.

(1.8
منظور سوال را دقیق متوجه نشدم ولی پاسخ آن را توضیح میدهم. برای تقسیم بندی داده اگر داده ها خیلی زیاد باشند درصد کمی از آن برای تست می رود ولی اگر داده تعداد معمولی داشته باشد حدود 20 درصد آن مثلا برای تست میرود. از طرفی دیگر برای

مجموعه ی validation یا همان dev روشی داریم به نام cross validation که هی زیر مجموعه های مختلفی استفاده میکنیم تا مطمن شویم روی قسمتی از داده متمرکز نشده ایم و مذل واقعا نتیجه خوبی داشته یا نه.

(1.9
غلط است. به گرادیان‌های اخیر وزن بیشتری داده می‌شود و وزن گرادیان‌های قدیمی‌تر به صورت نمایی کاهش می‌یابد.

سوال 2)

(2.1
چند دلیل دارد که بعضی از آن ها به این شرح اند: خروجی آن zero-centered است. در حول نقطه ی صفر مشکل vanishing gradient آن نسبت به sigmoid کمتر است(در بقیه نقاط لزوما نه) بنابراین برای داده نرمال میتواند بهتر باشد . همگرایی سریع تری دارد. مشتق آن حداکثر 1 میباشد در صورتی که در در sigmoid برابر ¼ است.

(2.2
زیرا در هر مرحله آموزش به صورت تصادفی برخی نورون‌ها را خاموش می‌کند، بنابراین ساختار شبکه در هر اپوک کمی متفاوت است. به دلیل حذف تصادفی نورون‌ها، مقدار هزینه محاسبه‌شده در هر اپوک شامل نویز بیشتری است و ممکن است کاهش یا افزایش‌های موقتی نشان دهد، درواقع تابع هزینه فرمول مشخصی ندارد در هر مرحله (حداقل از چشم ما).

(2.3
زیرا به داده‌های جدید وزن بیشتری داده می‌شود و داده‌های قدیمی‌تر به صورت نمایی وزن کمتری دارند. این باعث می‌شود تغییرات ناگهانی و نویزهای کوتاه‌مدت که در داده‌های اخیر دیده می‌شوند، کمتر تأثیرگذار باشند و روند کلی داده‌ها بهتر نمایان شود، این وزن‌دهی نمایی باعث می‌شود که فیلتر به عنوان یک فیلتر پایین‌گذر (Low-pass filter) عمل کند و نوسانات سریع و نویزهای با فرکانس بالا را کاهش دهد.

(2.4
میتواند از vanishing gradient و نیز exploding gradient جلوگیری کند. حتی میتواند به همگرایی سریعتر منجر شود اگر وزن ها درست انتخاب شوند. همچنین با ایجاد عدم تقارن بین نورون ها ، مقداردهی تصادفی و مناسب وزن‌ها باعث می‌شود نورون‌ها در لایه‌های مختلف متفاوت عمل کنند و یادگیری بهتر انجام شود.

سوال 3)

سوال 4)

$$v_i = betha * v_{i-1} + (1 - betha) * theta_i$$

$$v_1 = 0.9 * 22 + 0.1 * 20 = 21.8$$
$$v_2 = 0.9 * 24 + 0.1 * 21.8 = 23.78$$
$$v_3 = 0.9 * 20 + 0.1 * 23.78 = 20.378$$

$v_4 = 0.9 * 19 + 0.1 * 20.378 = 19.1378$

$v_5 = 0.9 * 21 + 0.1 * 19.1378 = 20.81378$

حساسیت به این صورت است که به صررت تخمینی ما داریم در هر مرحله حدود $1/(1 - betha)$ داده را در نظر میگیریم و در نتیجه با افزایش آن به سمت 1 درواقع تعداد بیشتری از داده های اخیر را در نظر میگیریم و برای همین منحنی نرم تر و به نویز کم حساسیت تر میشود و برعکس اگر بتا به سمت صفر برود خیلی به نویز حساس میشود و تعداد داده های اخیر کمتری را در نظر میگیرد. بنابراین مثلا بتای 0.5 به نویز حساس تر از 0.9 است.