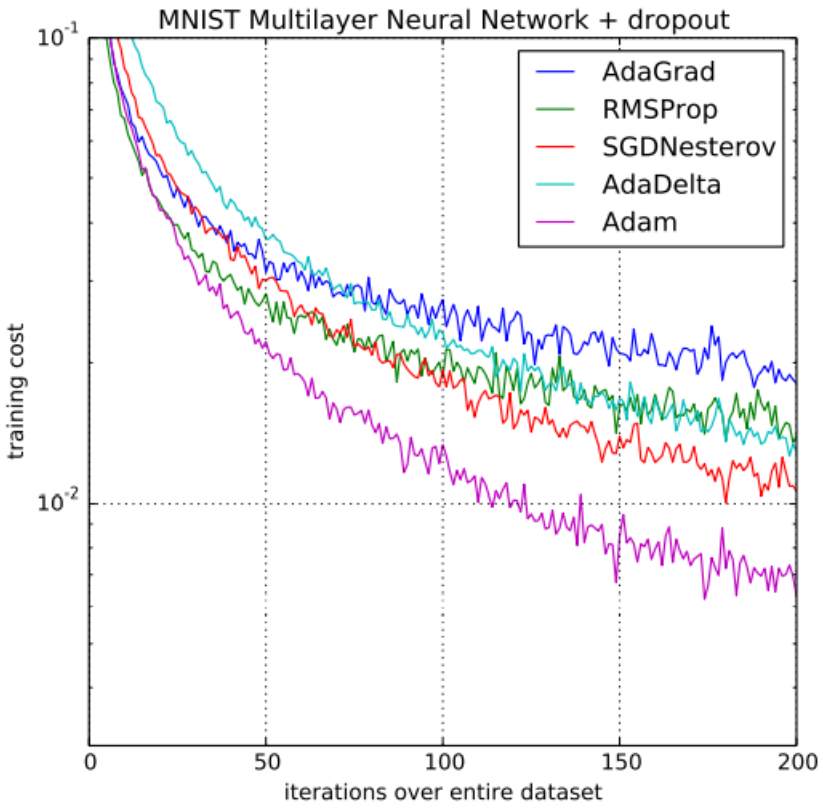
۱. نمودار داده شده، ۵ نوع از Optimizerهای مختلف را با هم از لحاظ هزینه محاسباتی که دارند، مقایسه کرد و همین‌طور برای درک بهتر باید به نوع کارکرد آنها توجه کرد.



نمودار بالا نشان‌دهنده این است Optimizerهای مختلف، به دلیل داشتند فرمول‌ها و همین‌طور نحوه کار متفاوت، باعث متفاوت بودن سرعت آموزش ما می‌شوند. به طور مثال در epochهای اولیه، بار محاسباتی ما برای optimizer نمودار بالا مطابق با زیر است:

RMSProp < Adam < AdaGead < SGDNesterov < AdaDelta

حال با افزایش کمی epoch ها، بار محاسباتی مطابق زیر است:

Adam < RMSProp < SGDNesterov < AdaGead < AdaDelta

و در نهایت با داشتن epochهای بیشتر در نهایت به حالت زیر می‌رسیم:

Adam < SGDNesterov < AdaDelta < RMSProp < AdaGead

عملکرد optimizer در آموزش به موارد زیادی بستگی دارد.

SGDیک الگوریتم بسیار ابتدایی است و به دلیل سرعت کم محاسباتی که دارد در حال حاضر به‌ندرت در برنامه‌های کاربردی استفاده می‌شود. یکی دیگر از مشکلات آن الگوریتم نرخ یادگیری ثابت برای هر دوره است. علاوه بر این، نمی‌تواند به‌خوبی نقاط زین را اداره کند.

Adagrad معمولاً به دلیل به‌روزرسانی‌های مکرر در نرخ یادگیری، بهتر از شیب نزولی تصادفی عمل می‌کند. زمانی که برای برخورد با داده‌های پراکنده استفاده می‌شود بهترین کار است .

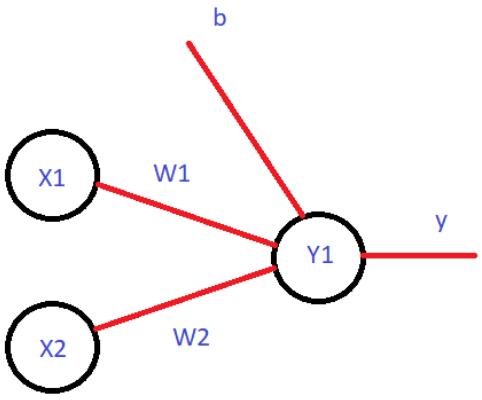
RMSProp نتایج مشابهی با الگوریتم نزول گرادیان با تکانه نشان می‌دهد، فقط نحوه محاسبه گرادیان‌ها را متفاوت می‌کند.

در نهایت بهینه‌ساز Adam می‌آید که ویژگی‌های خوب RMSProp و سایر الگوریتم‌ها را به ارث برده است. نتایج بهینه‌ساز Adam عموماً بهتر از هر الگوریتم بهینه‌سازی دیگر است، زمان محاسبات سریع‌تری دارد و به پارامترهای کمتری برای تنظیم نیاز دارد. به همین دلیل، Adamبه‌عنوان بهینه‌ساز پیش‌فرض برای اکثر برنامه‌ها توصیه می‌شود.

انتخاب بهینه‌ساز Adam برای برنامه شما ممکن است بهترین احتمال را برای گرفتن بهترین نتایج به شما بدهد .در نهایت، ما متوجه شدیم که حتی بهینه‌ساز Adam دارای برخی نقاط ضعف است. همچنین، مواردی وجود دارد که الگوریتم‌هایی مانند SGD ممکن است سودمند باشند و بهتر از بهینه‌ساز Adam عمل کنند.

بنابراین، برای انتخاب بهترین الگوریتم بهینه‌سازی و دستیابی به نتایج برجسته، دانستن نیازهای خود و نوع داده‌هایی که با آنها سروکار دارید، نوع مدلی که می‌خواهیم آن را آموزش دهیم و روند عملیاتی آن، بسیار مهم است.

۲. مدل ما به شکل زیر است:



طبق گفته سؤال، آموزش را با مقادیر زیر شروع می‌کنیم:

$$Y = 1, \quad X_1 = 3, \quad W_1 = 2, \quad X_2 = -1, \quad W_2 = 1, \quad b = 2$$

$$Y_1 = X_1 * W_1 + X_2 * W_2 + b = 3 * 2 + -1 * 1 + 2 = 7$$

حال با داشتن مقدارهای خروجی لایه مخفی می‌توانیم برای لایه خروجی هم محاسبات بالا را انجام دهیم البته تابع فعال‌ساز ما sigmoid است:

$$Sigmoid\ (Y_1) = Y_{pred} = \ 0.99908$$

حال بیایم مقدار ارور را باتوجه‌به فرمول زیر حساب کنیم:

$$Ei = \frac{(Y - Y_{pred})^2}{2}$$

که در فرمول بالا مقدار ارور برای y، i ام حساب شده است.out(yi) برابر با مقدار خروجی yi است و همچنین در فرمول بالا Yi مقداری است که ما انتظار داشتیم به‌عنوان خروجی بگیریم که در این مدل ما ۰ یا ۱ باید باشد.

$$E = \frac{(Y - Y_{pred})^2}{2} = \ \frac{(1 - 0.99908)^2}{2} = \ 4.15e - 7$$

حال در مرحله بعدی نیاز داریم به‌صورت backward حرکت کنیم و وزن‌ها را اصلاح کنیم.

$$\frac{\partial Error}{\partial w1} = \frac{\partial Error}{\partial Y} * \frac{\partial Y}{\partial Y_{pred}} * \frac{\partial Y_{pred}}{\partial w1}$$

$$\frac{\partial Error}{\partial Y} = \ 2 * \frac{1}{2} * \left(Y - Y_{pred}\right) * (-1) = (-1 + 0.99908) = \ 0.0009$$

$$\frac{\partial Y}{\partial Y_{pred}} = \ \frac{\partial S(Y_{pred})}{\partial Y_{pred}} = S\left(\partial Y_{pred}\right) * \left(1 - S\left(\partial Y_{pred}\right)\right) = \ 0.99908 * (1 - 0.99908)$$
$$= \ 9.1022\text{e} - 4$$

$$\frac{\partial Y_{pred}}{\partial w1} = \ X_1 = 3$$

$$\frac{\partial Error}{\partial w2} = \ 0.0009 * (9.1022\text{e} - 4) * 3 = \ 2.49\text{e} - 6$$

$$New\ w1 \ = \ \mathbf{3} \ - \ \eta \ * \ 2.49\text{e} - 6$$

$$\eta\ (learning\ rete) \ = \ 0.5$$

$$\boldsymbol{New\ w1 = 2.999998}$$

$$\frac{\partial Error}{\partial w2} = \frac{\partial Error}{\partial Y} * \frac{\partial Y}{\partial Y_{pred}} * \frac{\partial Y_{pred}}{\partial w2}$$

$$\frac{\partial Error}{\partial Y} = 2 * \frac{1}{2} * \left(Y - Y_{pred}\right) * (-1) = (-1 + 0.99908) = 0.0009$$

$$\frac{\partial Y}{\partial Y_{pred}} = \frac{\partial S(Y_{pred})}{\partial Y_{pred}} = S(\partial Y_{pred}) * \left(1 - S(\partial Y_{pred})\right) = 0.99908 * (1 - 0.99908)$$
$$= 9.1022\text{e} - 4$$

$$\frac{\partial Y_{pred}}{\partial w2} = X_1 = -1$$

$$\frac{\partial Error}{\partial w2} = 0.0009 * (9.1022\text{e} - 4) * 3 = 2.49\text{e} - 6$$

$$New\ w2 = \mathbf{-1} - \eta * 2.49\text{e} - 6$$

$$\eta\ (learning\ rete) = 0.5$$

$$\boldsymbol{New\ w2 = -1.000001}$$

حال با این وزن‌های جدید دوباره آموزش را شروع می‌کنیم:

برای اینکه این موارد محاسباتی زمان‌گیر است، کد backpropagation را درست کردیم و نتایج رو در آنجا حساب کردیم.

```
>epoch=0, lrate=0.500, error=1.231
>epoch=1, lrate=0.500, error=1.203
>epoch=2, lrate=0.500, error=1.177
>epoch=3, lrate=0.500, error=1.155
>epoch=10, lrate=0.500, error=1.067
>epoch=11, lrate=0.500, error=1.061
>epoch=12, lrate=0.500, error=1.056
>epoch=13, lrate=0.500, error=1.051
>epoch=14, lrate=0.500, error=1.047
>epoch=15, lrate=0.500, error=1.043
>epoch=16, lrate=0.500, error=1.040
>epoch=17, lrate=0.500, error=1.037
>epoch=18, lrate=0.500, error=1.034
>epoch=19, lrate=0.500, error=1.032
[{'weights': [-0.24721764529888557, 0.8317348282246627, 0.6937378045793847], 'output': 0.22490152138520883, 'delta': -0.0067926565437739l9}, {'weights': [0.05028838549412932б, 0.7533322294716943, 0.3053760797877784], 'output': 0.2465603002218892, 'delta': 0.011319198282635607}]
[{'weights': [0.3529054850134387, 0.4260924495871123, -0.16415032841907762], 'output': 0.4872762124725615, 'delta': -0.1280979395826948}, {'weights': [0.01751521771567937, 0.826573904132544, -0.11301976169792204], 'output': 0.5422776110104924, 'delta': 0.1346001377068314}]
```

۴. داده‌های ما به‌صورت zip شده دانلود شده‌اند و سپس به فیلتر train و validation آن‌ها را از همدیگر جدا می‌کردیم و همین‌طور در ادامه یک فیلتر cats و dogs را روی آنها اعمال می‌کنیم تا labelهای متفاوت را در آنها جدا می‌کنیم.

در لایبرری keras از ImageDataGenerator استفاده کردیم که augmentation را برای ما فراهم می‌کند. در ادامه به‌وسیله flow_from_directory داده‌های آموزش و ارزیابی را در batchهایی تقسیم کردیم.

۴ لایه برای مدل تدبیر شده:

- لایه کانولووشن همراه با ۳۲ نورون به همراه کرنل 3*3 و تابع فعال‌ساز relu و MaxPooling2D با سایز 2
- لایه کانولووشن همراه با ۶۴ نورون به همراه کرنل 3*3 و تابع فعال‌ساز relu و MaxPooling2D با سایز 2
- لایه کانولووشن همراه با ۱۲۸ نورون به همراه کرنل 3*3 و تابع فعال ساز relu و MaxPooling2D با سایز 2
- یک‌لایه dense همراه با 512 نورون و تابع فعال‌سازی که باید با مقادیر مختلف امتحان کرد.

داده‌های آموزش ما در ۲ دسته "سگ" و "گربه" تقسیم بنده شده‌اند و بنا بر همین موضوع تعداد لایه آخر ما باید ۲ باشد. نورون A در لایه آخر نشان‌دهنده "سگ بودن" و نورون B در لایه آخر نشان‌دهنده "گربه بودن" است. افزایش تعداد این نورون‌ها به دلیل ۲ دسته بودن مسئله ما بی‌معنی است و همچنین با یک نورون نیز می‌توان مدل را برای یک مسئله ۲ کلاس در نظر گرفت.

اصولاً برای مسائل binary classifactin از تابع فعال‌سازی sigmoid و تابع ضرر binary crossentropy استفاده می‌کنیم، ولی طبق گفته سؤال، حالت‌های زیر را در نظر می‌گیریم. (برای لایه آخر یک مسئله کلاس‌بندی، از relu استفاده نمی‌کنیم.)

	Loss	*Acc*	*Val Loss*	*Val Acc*
CategoricalCrossentropy softmax*(2- neurons)*	۳۹/۱	۸۳/۴۵	۵۵/۹	۸۰/۹
CategoricalCrossentropy *sigmoid(2- neurons)*	۴۰/۷	۸۲/۲۵	۶۲/۸۹	۷۸/۶
BinaryCrossentropy Sigmoid(1- neurons)	۳۷/۹۴	۸۴/۶	۵۹/۰۲	۷۴/۶
BinaryCrossentropy softmax*(1- neurons)*	۴۱/۶۹	۵۰	۵۹/۷۰	۵۰
BinaryCrossentropy softmax*(2- neurons)*	۴۰/۲۴	۸۳/۴۵	۴۵	۸۱/۳۰
BinaryCrossentropy Sigmoid (2- neurons)	۴۰/۳۲	۸۲/۸	۶۴/۴۹	۷۴/۵
CategoricalCrossentropy *sigmoid(1- neurons)*	۰	۵۰	۰	۵۰
CategoricalCrossentropy softmax*(1- neurons)*	۰	۵۰	۰	۵۰

نتایج بالا نشان‌دهنده این است برای این مسئله ۲ کلاسه بهترین حالت برای ردیف ۵ است و همچنین اگر در لایه آخر ما یک نورون داشته باشیم و از softmax استفاده کرده باشیم، مدل ما آموزش نمی‌بیند.

A Comprehensive Guide on Deep Learning Optimizers (analyticsvidhya.com)