Algorithm 6 Adam

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\mu^t}$$
$$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t + \varepsilon}}$$

Adam، RMSProp را با momentum کلاسیک ترکیب می کند. اما نشان داده شده است که در حالت کلی NAG (Nesterov's accelerated gradient) از momentum کلاسیک بهتر است .الگوریتم NAG را بازنویسی می کنیم تا ساده تر و کارآمدتر باشد.

Algorithm 3 Nesterov's accelerated gradient

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1} - \eta\mu\mathbf{m}_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \mathbf{m}_t$$

→

Algorithm 7 NAG rewritten

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\bar{\mathbf{m}}_t \leftarrow \mathbf{g}_t + \mu_{t+1}\mathbf{m}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \bar{\mathbf{m}}_t$$

بردار $\tilde{m}$ علاوه بر آپدیت بردار momentum برای timestep بعدی $\mu_{t+1}m_t$ ، حاوی آپدیت گرادیان (مشتق) برای timestep فعلی g_t است که باید قبل از گرفتن گرادیان در timestep بعدی اعمال شود. دیگر نیازی به اعمال بردار momentum برای timestep فعلی نیستیم زیرا قبلاً آن را در آخرین آپدیت پارامترها ، در timestep t-1 اعمال کرده ایم.

قانون آپدیت Adam را می توان با توجه به بردارهای momentum/norm قبلی و آپدیت گرادیان فعلی مانند زیر نوشت.

$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\mu \mathbf{m}_{t-1}}{\sqrt{\nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2 + \varepsilon}} - \eta \frac{(1-\mu)\mathbf{g}_t}{\sqrt{\nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2 + \varepsilon}}$$

در بازنویسی NAG ، ما قسمت اول مرحله را بر می داریم و آن را قبل از گرفتن گرادیان (مشتق) تابع هزینه f اعمال می کنیم. با این حال ، مخرج به g_t بستگی دارد ، بنابراین ما نمی توانیم از ترفند مورد استفاده در NAG برای این معادله استفاده کنیم . هر چند ، ν به طور کلی بسیار بزرگ انتخاب می شود (به طور معمول> 0.9) ، بنابراین تفاوت بین n_{t-1} و

n_t به طور کلی بسیار کم خواهد بود. پس می توانیم n_{t-1} را با n_t بدون آنکه دقت زیادی را از دست بدهیم جایگزین می‌کنیم:

$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\mu \mathbf{m}_{t-1}}{\sqrt{\mathbf{n}_{t-1}} + \varepsilon} - \eta \frac{(1-\mu)\mathbf{g}_t}{\sqrt{\nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2} + \varepsilon}$$

ترم اول در عبارت فوق دیگر به g_t بستگی ندارد ، به این معنی که در اینجا می توانیم از ترفند Nesterov استفاده کنیم. این، عبارات زیر را برای $\tilde{m}_t$ و θ_t به ما می دهد:

$$\bar{\mathbf{m}}_t \leftarrow (1-\mu_t)\mathbf{g}_t + \mu_{t+1}\mathbf{m}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\bar{\mathbf{m}}_t}{\sqrt{\mathbf{v}_t} + \varepsilon}$$

تنها چیزی که باقی مانده این است که تعیین کنیم که چگونه ترم‌های تصحیح bias مقداردهی اولیه را بگنجانیم ، با توجه به اینکه g_t از timestep فعلی می آید اما m_t از timestep بعدی است. این به ما فرم نهایی الگوریتم Nadam را می دهد.

Algorithm 8 Nesterov-accelerated adaptive moment estimation

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\hat{\mathbf{g}} \leftarrow \frac{\mathbf{g}_t}{1 - \prod_{i=1}^{t} \mu_i}$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1 - \prod_{i=1}^{t+1} \mu_i}$$
$$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$$
$$\bar{\mathbf{m}}_t \leftarrow (1-\mu_t)\hat{\mathbf{g}}_t + \mu_{t+1}\hat{\mathbf{m}}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\bar{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t} + \varepsilon}$$

خط اول کد فوق مربوط به گرادیان است. خط سوم و چهارم کد مربوط به first moment و first unbias هست. μ همان beta1 است. خط پنج و شش مربوط به second moment و second unbias می‌باشد و v همان پارامتر beta2 هست. خط آخر همان خط آخر adam هست با این تفاوت که به جای mhat از $\tilde{m}_t$ استفاده شده است. خط هفتم از الگوریتم NAG آمده است.

Algorithm 3 Nesterov's accelerated gradient

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1} - \eta\mu\mathbf{m}_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu\mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta\mathbf{m}_t$$

Algorithm 7 NAG rewritten

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu_t\mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\bar{\mathbf{m}}_t \leftarrow \mathbf{g}_t + \mu_{t+1}\mathbf{m}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta\bar{\mathbf{m}}_t$$

Algorithm 6 Adam

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu\mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\mu^t}$$
$$\mathbf{n}_t \leftarrow \nu\mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$$
$$\theta_t \leftarrow \theta_{t-1} - \eta\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t}+\varepsilon}$$

Algorithm 8 Nesterov-accelerated adaptive moment estimation

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\hat{\mathbf{g}} \leftarrow \frac{\mathbf{g}_t}{1-\prod_{i=1}^{t}\mu_i}$$
$$\mathbf{m}_t \leftarrow \mu\mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\prod_{i=1}^{t+1}\mu_i}$$
$$\mathbf{n}_t \leftarrow \nu\mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$$
$$\bar{\mathbf{m}}_t \leftarrow (1-\mu_t)\hat{\mathbf{g}}_t + \mu_{t+1}\hat{\mathbf{m}}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta\frac{\bar{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t}+\varepsilon}$$

همان‌طور که باتوجه به expriment‌ها روی taskهای گفته شده در مقاله می‌توان فهمید loss در الگوریتم Nadam در training set و validation set کمتر از adam و performance آن بهتر می‌باشد.

به نظر می‌آید دلیل عملکرد متفاوت optimizerها در مسئله‌های متفاوت مدل‌های متفاوت که برای حل مسائل به کار گرفته می‌شود و همچنین شکل تابع ضرر که ناشی از همین انتخاب مدل و همچنین و ویژگی‌های داده می‌باشد هست.

با توجه به شکل مربوط به شبکه عصبی داریم

$$h = w_0 x_1 + w_1 x_2 + w_2 \, bias$$

که با w_2 و با bias یک جمع شود و دیگری با استفاده از شبکه train شود ما bias=1 قرار می دهیم و w_2 را با استفاده از شبکه train می کنیم. بنابراین داریم

$$h = w_0 x_1 + w_1 x_2 + w_2$$

$$z = \sigma(h)$$

برای optimizer داریم SGD :

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i(z_i, y_i)$$

$$\nabla_w L = \frac{1}{N} \sum_{i=1} \nabla_w L_i(z_i, y_i)$$

$$w = w - \eta \nabla_w L$$

مقدار η توسط سئوال تعیین شده است بنابراین فرض می کنیم $\eta = 0.1$

تابع ضرر از نوع binary cross entropy هست یعنی:

$$L_i(w) = -\left(y_i \log z_i + (1-y_i) \log(1-z_i) \right)$$

برای آپدیت وزن های شبکه باید مشتق تابع ضرر را نسبت به وزن های شبکه به دست آوریم:

$$\text{chain rule: } \frac{\partial L_i}{\partial w} = \frac{\partial L_i}{\partial z_i} \times \frac{\partial z_i}{\partial h_i} \times \frac{\partial h_i}{\partial w}$$

$$\frac{\partial L_i}{\partial z_i} = -\left(\frac{y_i}{z_i} - \frac{1-y_i}{1-z_i} \right) = \frac{z_i - y_i}{z_i(1-z_i)}$$

$$\frac{\partial z_i}{\partial h_i} = z_i(1-z_i)$$

شماره ورودی

$$\frac{\partial h_i}{\partial w_0} = x_1^i \qquad \frac{\partial h_i}{\partial w_1} = x_2^i \qquad \frac{\partial h_i}{\partial w_2} = 1$$

$$\frac{\partial L_i}{\partial w_0} = \frac{(z_i - y_i)}{z_i(1-z_i)} z_i(1-z_i) x_1^i = (z_i - y_i) x_1^i$$

$$\frac{\partial L_i}{\partial w_1} = (z_i - y_i) x_2^i$$

$$\frac{\partial L_i}{\partial w_2} = (z_i - y_i)$$

حال وزن های اولیه را به صورت تصادفی به

$w_0 = w_1 = w_2 = 0$

مقدار دهی کرده و شروع می کنیم به train شبکه:

شماره ورودی

$h_i = w_0 x_1^i + w_1 x_2^i + w_2 \quad , z_i = \frac{1}{1+e^{-h_i}}$

epoch اول:

batch اول:

داده شماره 1: $h_1 = 0 \times 121 + 0 \times 16.8 + 0 = 0$

$$z_1 = \frac{1}{1+e^0} = \frac{1}{2}$$

از آن جایی که در ابتدا $z_1 = z_2 = \frac{1}{2}$ یعنی

داده شماره 2: $h_2 = 0 \times 114 + 0 \times 15.2 + 0 = 0$

شبکه برای این داده نمی تواند به سادگی

$$z_2 = \frac{1}{1+e^0} = \frac{1}{2}$$

مقدار y را تشخیص دهد

$$\frac{\partial L_1}{\partial w_0} = (z_1 - y_1) x_1^1 = \left(\frac{1}{2} - 0\right) 121 = 60.5$$

$$\frac{\partial L_2}{\partial w_0} = (z_2 - y_2) x_1^2 = \left(\frac{1}{2} - 0\right) 114 = 57$$

$$\frac{\partial L}{\partial w_0} = \frac{60.5 + 57}{2} = 58.75$$

$$\frac{\partial L_1}{\partial w_1} = (z_1 - y_1) x_2^1 = \left(\frac{1}{2} - 0\right) 16.8 = 8.4$$

$$\frac{\partial L_2}{\partial w_2} = \left(\frac{1}{2} - 0\right) 15.2 = 7.6$$

$$\frac{\partial L}{\partial w_1} = \frac{8.4 + 7.6}{2} = 8$$

$$\frac{\partial L_1}{\partial w_2} = (z_1 - y_1) = \frac{1}{2} \qquad \frac{\partial L_2}{\partial w_2} = \left(\frac{1}{2} - 0\right) = \frac{1}{2}$$

$$\frac{\partial L}{\partial w_2} = \frac{\frac{1}{2} + \frac{1}{2}}{2} = \frac{1}{2} = 0.5$$

$$w = w - \eta \frac{\partial L}{\partial w}$$

$$w_0 = 0 - \frac{1}{10} \times 58.75 = -5.875$$

$$w_1 = 0 - \frac{1}{10} \times 8 = -0.8$$

$$w_2 = 0 - \frac{1}{10} \times 0.5 = -0.05$$

3 بعدی batch : پس

$$h_3 = -5.875 \times 210 - 0.8 \times 9.4 - 0.05 = -1241.32$$

$$z_3 = \frac{1}{1 + e^{1241.32}} = 0$$

داده شماره ۴: $h_4 = -5.875 \times 195 - 0.8 \times 8.1 - 0.05 = -1152.155$

$$z_4 = \frac{1}{1 + e^{1152.155}} \simeq 0$$

$$\frac{\partial L_3}{\partial w_0} = (0-1)\,210 = -210$$

$$\frac{\partial L_4}{\partial w_0} = (0-1)\,195 = -195$$

$$\longrightarrow \frac{\partial L}{\partial w_0} = -202.5$$

$$\frac{\partial L_3}{\partial w_1} = (0-1)\,9.4 = -9.4$$

$$\frac{\partial L_4}{\partial w_1} = (0-1)\,8.1 = -8.1$$

$$\longrightarrow \frac{\partial L}{\partial w_1} = -\frac{9.4+8.1}{2} = -8.75$$

$$\frac{\partial L_3}{\partial w_2} = (0-1) = -1$$

$$\frac{\partial L_4}{\partial w_2} = (0-1) = -1$$

$$\longrightarrow \frac{\partial L}{\partial w_2} = -\frac{1+1}{2} = -1$$

$$w_0 = -5.875 - \frac{1}{10} \times (-202.5) = 14.375$$

$$w_1 = -0.8 - \frac{1}{10}(-8.75) = 0.075$$

$$w_2 = -0.05 - \frac{1}{10}(-1) = 0.05$$

epoch دوم، batch اول

داده شماره ۱: $h_1 = 14.375 \times 121 + 0.075 \times 16.8 + 0.05$

$= 1790.685 \rightarrow z_1 = \frac{1}{1+e^{-1790.685}} \simeq 1$

داده شماره ۴ $h_4 = 14.375 \times 195 + 0.075 \times 8.1 + 0.05$

$= 2803.7825 \rightarrow z_4 = \frac{1}{1+e^{-2803.7825}} \simeq 1$

داده ی شماره ۱ غلط تخمین (حدس) زده شده (پیشبینی شده) بنابراین گرادیان (مشتق) برای آن باید به اندازه کافی بزرگ باشد تا بتواند آن را تصحیح کند

داده شماره ۴ درست پیشبینی شده بنابراین گرادیان آن باید نزدیک به صفر باشد

$\frac{\partial L_1}{\partial w_0} = (1-0)\,121 = 121$

$\frac{\partial L_4}{\partial w_0} = (1-1)\,210 = 0$

$\rightarrow \frac{\partial L}{\partial w_0} = \frac{121+0}{2} = 60.5$

$\frac{\partial L_1}{\partial w_1} = (1-0)\,16.8 = 16.8$

$\frac{\partial L_4}{\partial w_1} = (1-1)\,8.1 = 0$

$\rightarrow \frac{\partial L}{\partial w_1} = \frac{16.8+0}{2} = 8.4$

$\frac{\partial L_1}{\partial w_2} = (1-0) = 1$

$\frac{\partial L_4}{\partial w_2} = (1-1) = 0$

$\frac{\partial L}{\partial w} = \frac{1+0}{2} = 0.5$

$w_0 = 14.375 - \frac{1}{10} \times 60.5 = 8.325$

$w_1 = 0.075 - \frac{1}{10} \times 8.4 = -0.765$

$w_2 = 0.05 - \frac{1}{10} \times 0.5 = 0$

دوم batch:

داده شماره 2: $h_2 = 8.325 \times 114 - 0.765 \times 15.2 + 0$

$= 937.422 \Rightarrow z_2 = \frac{1}{1+e^{-937.422}} = 1$

داده شماره 3: $h_3 = 8.325 \times 210 - 0.765 \times 9.4 + 0$

$= 1,741.059 \rightarrow z_3 = 1$

در این جا نیز داده 2 اشتباه پیشبینی شده و باید برای آن گرادیان به اندازه کافی بزرگ باشد و داده 3 درست پیشبینی شده و گرادیان باید برای آن نزدیک صفر باشد

$\frac{\partial L_2}{\partial w_0} = (1-0)\,114 = 114$

$\rightarrow \frac{\partial L}{\partial w_0} = \frac{114+0}{2} = 57$

$\frac{\partial L_3}{\partial w_0} = (1-1)\,210 = 0$

$\frac{\partial L_2}{\partial w_1} = (1-0)\,15.2 = 15.2$

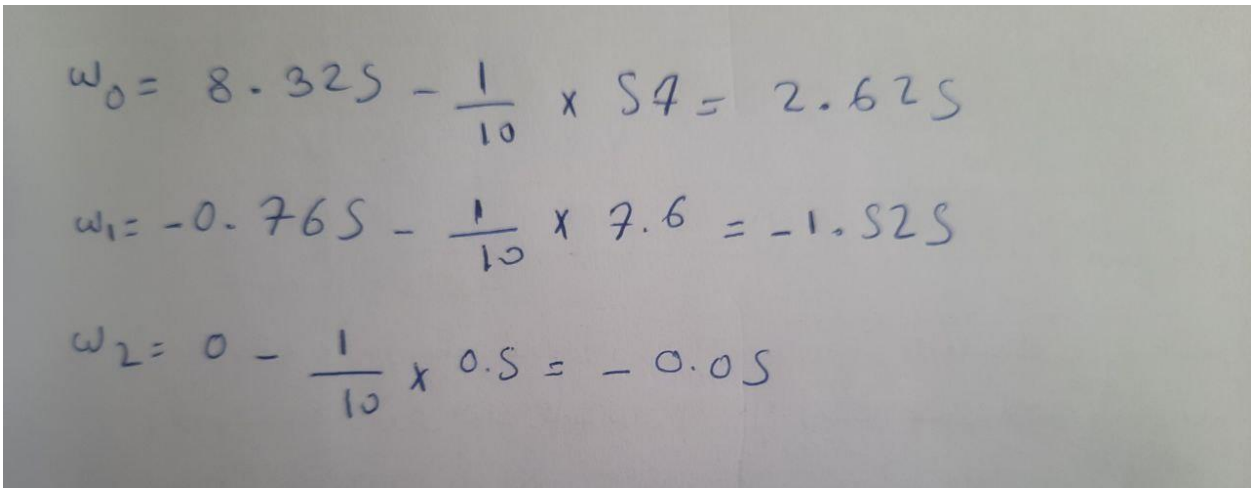
$\rightarrow \frac{\partial L}{\partial w_1} = \frac{15.2+0}{2} = 7.6$

$\frac{\partial L_3}{\partial w_1} = (1-1)\,9.4 = 0$

$\frac{\partial L_2}{\partial w_2} = (1-0) = 1$

$\rightarrow \frac{\partial L}{\partial w_2} = \frac{1+0}{2} = 0.5$

$\frac{\partial L_3}{\partial w_2} = (1-1) = 0$



به نظر می‌رسد در این مسئله هر چه تعداد epochها بالاتر برود نتیجه بهتر می‌شود ولی شاید اگر learning rate کوچک باشد این نتیجه زودتر بهتر شود و مسئله زودتر همگرا شود چون اگر learning rate کوچک شود مقدار h کوچک می‌شود و شاید این به داده‌های یک و دو کمک کند تا زودتر همگرا شود.

https://math.stackexchange.com/questions/2503428/derivative-of-binary-cross-entropy-why-are-my-signs-not-right

-3

امکان اینکه پارامترهای beta1 و beta2 طوری مقداردهی کرد که الگوریتم adam دقیقا معادل الگوریتم SGD شود وجود ندارد اما اگر beta1 = 0 و beta2 را برابر مقداری بسیار نزدیک به 1 که مساوی با آن نباشد قرار دهیم می‌توان این دو الگوریتم را به یکدیگر تقریب زد.

برای قسمت دو این سوال beta1 هر چه نزدیکتر به صفر و نه برابر با آن باشد نتیجه بهتر است و در مورد beta2 هر چه نزدیک به یک و نه برابر با آن باشد نتیجه بهتر است.

-4

Learning rate = 0.001

```
Epoch 1/10
300/300 - 2s - loss: 2.2066 - accuracy: 0.1816 - val_loss: 2.0583 - val_accuracy: 0.3687
Epoch 2/10
300/300 - 2s - loss: 1.9496 - accuracy: 0.4805 - val_loss: 1.8231 - val_accuracy: 0.5946
Epoch 3/10
300/300 - 1s - loss: 1.7321 - accuracy: 0.6219 - val_loss: 1.6130 - val_accuracy: 0.6775
Epoch 4/10
300/300 - 1s - loss: 1.5389 - accuracy: 0.6872 - val_loss: 1.4298 - val_accuracy: 0.7248
Epoch 5/10
300/300 - 2s - loss: 1.3732 - accuracy: 0.7292 - val_loss: 1.2760 - val_accuracy: 0.7612
Epoch 6/10
300/300 - 2s - loss: 1.2355 - accuracy: 0.7582 - val_loss: 1.1497 - val_accuracy: 0.7835
Epoch 7/10
300/300 - 2s - loss: 1.1226 - accuracy: 0.7780 - val_loss: 1.0469 - val_accuracy: 0.8002
Epoch 8/10
300/300 - 2s - loss: 1.0303 - accuracy: 0.7932 - val_loss: 0.9627 - val_accuracy: 0.8129
Epoch 9/10
300/300 - 2s - loss: 0.9544 - accuracy: 0.8052 - val_loss: 0.8935 - val_accuracy: 0.8218
Epoch 10/10
300/300 - 1s - loss: 0.8915 - accuracy: 0.8144 - val_loss: 0.8360 - val_accuracy: 0.8283
Model Error: 17.17%
```

در این حالت Learning rate کوچک است بنابراین مدل دیر همگرا می‌شود و پس از 10 epoch دقت مدل به 80 رسیده که نسبت به چیزی که می‌توان به آن رسید کم می‌باشد و خطای مدل روی داده تست حدود 17 درصد می‌باشد.

Learning rate = 0.01

```
Epoch 1/10
300/300 - 2s - loss: 1.4424 - accuracy: 0.6395 - val_loss: 0.8499 - val_accuracy: 0.8147
Epoch 2/10
300/300 - 1s - loss: 0.6997 - accuracy: 0.8400 - val_loss: 0.5591 - val_accuracy: 0.8678
Epoch 3/10
300/300 - 1s - loss: 0.5277 - accuracy: 0.8697 - val_loss: 0.4610 - val_accuracy: 0.8845
Epoch 4/10
300/300 - 1s - loss: 0.4560 - accuracy: 0.8830 - val_loss: 0.4114 - val_accuracy: 0.8932
Epoch 5/10
300/300 - 1s - loss: 0.4152 - accuracy: 0.8910 - val_loss: 0.3804 - val_accuracy: 0.8995
Epoch 6/10
300/300 - 1s - loss: 0.3881 - accuracy: 0.8960 - val_loss: 0.3587 - val_accuracy: 0.9037
Epoch 7/10
300/300 - 1s - loss: 0.3680 - accuracy: 0.9004 - val_loss: 0.3430 - val_accuracy: 0.9065
Epoch 8/10
300/300 - 1s - loss: 0.3524 - accuracy: 0.9033 - val_loss: 0.3295 - val_accuracy: 0.9099
Epoch 9/10
300/300 - 1s - loss: 0.3396 - accuracy: 0.9060 - val_loss: 0.3195 - val_accuracy: 0.9121
Epoch 10/10
300/300 - 1s - loss: 0.3288 - accuracy: 0.9085 - val_loss: 0.3103 - val_accuracy: 0.9147
Model Error: 8.53%
```

در حالت قبل پس از 10 epoch به دقت 82 رسیدیم اما با Learning rate = 0.01 پس از یکی دو epoch. بنابراین همگرایی مدل بهتر شده، یادگیری بهتر شده و خطای مدل هم به 8.5 درصد رسیده که نصف حالت قبل می‌باشد.

Learning rate = 0.1

```
Epoch 1/10
300/300 - 2s - loss: 0.5396 - accuracy: 0.8593 - val_loss: 0.3123 - val_accuracy: 0.9118
Epoch 2/10
300/300 - 1s - loss: 0.2953 - accuracy: 0.9166 - val_loss: 0.2566 - val_accuracy: 0.9285
Epoch 3/10
300/300 - 1s - loss: 0.2507 - accuracy: 0.9294 - val_loss: 0.2294 - val_accuracy: 0.9373
Epoch 4/10
300/300 - 1s - loss: 0.2198 - accuracy: 0.9386 - val_loss: 0.2030 - val_accuracy: 0.9417
Epoch 5/10
300/300 - 1s - loss: 0.1966 - accuracy: 0.9450 - val_loss: 0.1820 - val_accuracy: 0.9487
Epoch 6/10
300/300 - 1s - loss: 0.1775 - accuracy: 0.9504 - val_loss: 0.1677 - val_accuracy: 0.9514
Epoch 7/10
300/300 - 1s - loss: 0.1619 - accuracy: 0.9555 - val_loss: 0.1551 - val_accuracy: 0.9548
Epoch 8/10
300/300 - 1s - loss: 0.1482 - accuracy: 0.9589 - val_loss: 0.1474 - val_accuracy: 0.9581
Epoch 9/10
300/300 - 1s - loss: 0.1372 - accuracy: 0.9622 - val_loss: 0.1375 - val_accuracy: 0.9603
Epoch 10/10
300/300 - 1s - loss: 0.1273 - accuracy: 0.9648 - val_loss: 0.1289 - val_accuracy: 0.9632
Model Error: 3.68%
```

در حالت قبل پس از 10 epoch به دقت 92 رسیدیم اما با Learning rate = 0.1 پس از یکی دو epoch. بنابراین همگرایی مدل بهتر شده، یادگیری بهتر شده و خطای مدل هم به 3.68 درصد رسیده که کمتر از نصف حالت قبل می‌باشد. دقت هم به 96 درصد رسیده که دقت بالایی است و همچنین مدل هم overfit نشده است.

Learning rate = 0.5

```
Epoch 1/10
300/300 - 2s - loss: 0.3439 - accuracy: 0.8978 - val_loss: 0.1730 - val_accuracy: 0.9494
Epoch 2/10
300/300 - 1s - loss: 0.1534 - accuracy: 0.9559 - val_loss: 0.1362 - val_accuracy: 0.9572
Epoch 3/10
300/300 - 1s - loss: 0.1139 - accuracy: 0.9668 - val_loss: 0.1112 - val_accuracy: 0.9659
Epoch 4/10
300/300 - 1s - loss: 0.0902 - accuracy: 0.9746 - val_loss: 0.0903 - val_accuracy: 0.9721
Epoch 5/10
300/300 - 1s - loss: 0.0748 - accuracy: 0.9784 - val_loss: 0.0948 - val_accuracy: 0.9698
Epoch 6/10
300/300 - 1s - loss: 0.0639 - accuracy: 0.9812 - val_loss: 0.0792 - val_accuracy: 0.9758
Epoch 7/10
300/300 - 1s - loss: 0.0551 - accuracy: 0.9840 - val_loss: 0.0800 - val_accuracy: 0.9759
Epoch 8/10
300/300 - 1s - loss: 0.0484 - accuracy: 0.9857 - val_loss: 0.0756 - val_accuracy: 0.9765
Epoch 9/10
300/300 - 1s - loss: 0.0427 - accuracy: 0.9883 - val_loss: 0.0706 - val_accuracy: 0.9776
Epoch 10/10
300/300 - 1s - loss: 0.0373 - accuracy: 0.9896 - val_loss: 0.0708 - val_accuracy: 0.9778
Model Error: 2.22%
```

خطای مدل هم به 2.22 درصد رسیده که بسیار کم هست. دقت هم به حدود 98 درصد رسیده که دقت بالایی است و همچنین مدل هم overfit نشده است.

Learning rate = 1

```
Epoch 1/10
300/300 - 2s - loss: 0.3883 - accuracy: 0.8789 - val_loss: 0.1591 - val_accuracy: 0.9521
Epoch 2/10
300/300 - 1s - loss: 0.1358 - accuracy: 0.9591 - val_loss: 0.1145 - val_accuracy: 0.9648
Epoch 3/10
300/300 - 1s - loss: 0.0979 - accuracy: 0.9703 - val_loss: 0.0991 - val_accuracy: 0.9681
Epoch 4/10
300/300 - 1s - loss: 0.0790 - accuracy: 0.9753 - val_loss: 0.0875 - val_accuracy: 0.9723
Epoch 5/10
300/300 - 1s - loss: 0.0654 - accuracy: 0.9800 - val_loss: 0.0861 - val_accuracy: 0.9739
Epoch 6/10
300/300 - 1s - loss: 0.0558 - accuracy: 0.9827 - val_loss: 0.0912 - val_accuracy: 0.9724
Epoch 7/10
300/300 - 1s - loss: 0.0484 - accuracy: 0.9850 - val_loss: 0.0883 - val_accuracy: 0.9741
Epoch 8/10
300/300 - 1s - loss: 0.0427 - accuracy: 0.9866 - val_loss: 0.0823 - val_accuracy: 0.9746
Epoch 9/10
300/300 - 1s - loss: 0.0365 - accuracy: 0.9892 - val_loss: 0.0794 - val_accuracy: 0.9760
Epoch 10/10
300/300 - 1s - loss: 0.0322 - accuracy: 0.9901 - val_loss: 0.0802 - val_accuracy: 0.9758
Model Error: 2.42%
```

مثل حالت قبل می‌باشد و فرق چندانی نکرده است.