حمید رضا کاشانی   پروژه یک   810100441

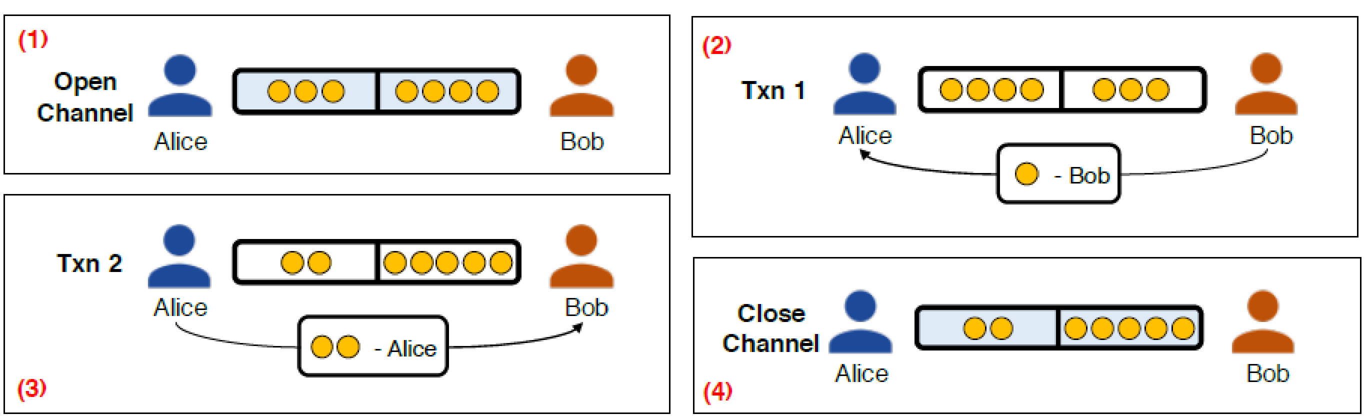
به دلیل وقت نا کافی و پیچیدگی پیاده سازی مسئله خود موفق نشدم کلیه کار های لازم تصویب شده در پروپوزال پروژه خود را انجام دهم و به جای کار های انجامنشده پروژه ی فدریتد لرنینگی را اضافه بر کار خود انجام داده ام و آن را به پروژه اول خود اضافه کردم تا جبران کار های انجام نشده شود.

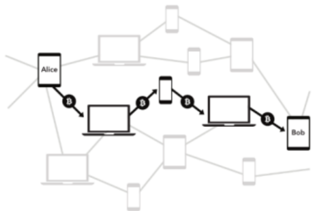
توضیحات ابتدایی برای اشنایی با موضوع پروژه

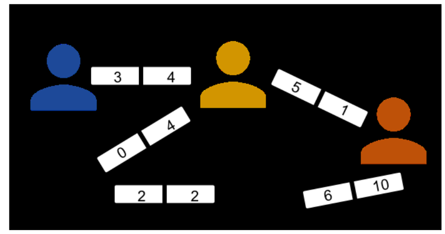
شبکه های کانال پرداخت یا PCN ها شبکه ای از نود هایی هستند که دو به دو با یکدیگر کانال پرداخت ایجاد می کنن و بدون تکرانش در زنجیره بلاک قابلیت پرداخت را در شرایطی داخل نود های شبکه می دهد. این شبکه به علت مشکل مقیاس پذیری در برخی رمز ارز ها مانند بیتکون ایجاد شده است زیرا هر بلاک درزمانی با میانگین ثابت و حجم محدود رمز ارز ایجاد می شود و با زیاد شدن تقاضا بلاکچین نمی تاند اسکیل پیدا کنند و باید مشترکین برای پرداخت در صف قرار بگیرن .

این شبکه به این صورت کار می کند که هر نود با چند نود دیگر در شبکه کانال ایجاد کرده است و در این شبکه هر دو نودی که بخواهند با هم تراکنش داشته باشند باید ارز های خود را با انتقال ار بین کانال های موجود به نود مورد نظر برساند و هر نود میانی هزینه انتقال ارز از خود را که قبلا اعلام کرده است از ارز های انتقالی بر می دارد و به نود بعدی می دهد .

نحوه ی ایجاد کانال اینگونه است که هر دو نود مقداری ارز را در حسابی چند امضا در زنجیره بلاک چین قرار می دهند برای مثال هر نود ده ارز را در حساب کانال قرار می دهد و یک کانال با ظرفیت 20 را تشکیل می دهد که هر نود می تواند ده ارز را هنگام تراز بودن کانال از خود انتقال دهد و به طور مثال با عبور دو ارز از یک طرف تعادل کانال به هم ریخته و به 8 و 12 تبدیل می شود و در صورتی که یک طرف کانال 0 شود دیگر نمیتوان از آن طرف کانال ارز انتقال داد مگر این که کانال جدید بین دو نود در بلاکچین ایجاد شود و کانال قبلی بسته شود که این عمل برای دو نود هزیته ایجاد می کند .







پس مسیریابی ها در شبکه باید به گونه ای باشد که تعادل کانال ها حفظ شود که ابن به این معنی است که مقدار ارز عبوری از کانال از دو نود برابر شود . همینطور این را هم باید در نظر گرفت که هر کانال محدودیتی دارد در انتقال ارز از خود که متناسب با ظرفیت آن است.

مدل کردن مسئله:

شبکه خود را به صورت گراف جهت دار وزن دار در نظر میگریم که نود های آن افراد در شبکه لایتنیگ هستن و هر یال یک طرف از کانال موجود بین دو نفر را نشان می دهد. وزن هر یال مقدار ارزی است که می شود از هر نود به نوددیگر انتقال داد و توجه شود شود که به علت وجود بالانس در موجودی هر کانال با کم شدن مقدار وزن یا ظرفیت هر یال به جهت مخالف آن همان مقدار اضافه می شود.مسیر انتخاب شده بین دو نود را مجموعه ای از یال های سری شده تشکیل می دهد. هر نود می خواهد مقداری ارز را به نود دیگری بدهد که به آن تقاضا می گوییم و با dij نمایش می دهیم. در شبکه از نود به نود دیگر مسیر هایی وجود دارد که هر کدام از این مسیر ها مقداری فلو ارز در آن ها وجود دارد که این مقدار را با Xp نمایش می دهیم. در هر نود برای جابه جایی ارز مقداری طول می کشد تا این تراکنش انجام شود و این پول باید تا زمان برگشت رسید قفل بماند که میانگین آن را با دلتا نمایش می دهیم و این دلتا باعث کم شدن ظرفیت واقعی کانال ها می شود به صورت میانگین. مقدار کل ظرفیت یک کانال یعنی مجموع ارزی که دو نود در کانال گذاشته اند برای تشکیل کانال را ظرفیت کانال می گویند و با Cuv نمایش داده شده است.xuv هم مقدار مجموع فلویی است که از نود u به نودv وجود دارد دقت شود که نود uوv متصل اند.

جدول توضیحات هر پارامتر

$G(V,E)$	Graph of the PCN with a set of V routers and E payment channels
$\mathcal{P}_{ij}$	Set of paths that sender i uses to receiver j
$\mathcal{P}$	$\bigcup_{i,j \in V} \mathcal{P}_{ij}$
x_p	Average rate of transaction-units on path p between sender i and receiver j
x_{uv}	$\sum_{p \in \mathcal{P}:(u,v) \in p} x_p$
d_{ij}	Demand from sender i to receiver j
c_{uv}	Total amount of tokens escrowed into payment channel (denotes channel size)(u,v)
Δ	Average time (s) over which tokens sent across a payment channel are unusable

Table 4.1: Notation for routing problem

حمید رضا کاشانی   پروژه یک   810100441

هدف ما این است که مجموع کل فلو های عبوری از شبکه را به ماکسیموم برسانیم (به صورت منصافانه یا غیر منصافنه که آن را با مشخص کردن تابع U می شود مدل کرد) و قید هایی در مسئله وجود دارد که باید این قید ها هم در شبکه اعمال شود:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i,j\in V} U\Big(\sum_{p\in\mathcal{P}_{ij}} x_p\Big) \\
\text{s.t.} \quad & \sum_{p\in\mathcal{P}_{ij}} x_p \le d_{ij} \quad \forall i,j \in V \\
& x_{uv} + x_{vu} \le \frac{c_{uv}}{\Delta} \quad \forall (u,v)\in E \\
& x_{uv} = x_{vu} \quad \forall (u,v)\in E \\
& x_p \ge 0 \quad \forall p \in \mathcal{P}.
\end{aligned}
$$

قید اول برای این است که مجموع فلو های دو نفر بیشتر از تقاضایشان نشود.

قید دوم محدودیت ظرفیت هر کانال را مشخص می کند که مجموع فلو عبور از هر دو طرف کانال باید از ظرفیت حقیقی آن کانال که وابسته به میانگین زمان تراکنش است کوچکتر باشد.

قید سوم برای پایداری بیشتر شبکه در نظر گرفته شده است که مقدار فلو عبور از هر دو طرف کانال با هم برابر باشد.

قید چهارم مقدار فلو ها نمی تواند منفی شود.

توجه شود که این چهار قید برای هر دو نود باید برقرار باشد

حل مسئله:

Primal: ابتدا بازنویسی مسئله و تغییراتی در فرم شروط

U را برای مسئله ماکسیمم thoughput برابر $U(x)=x$ گرفتیم $\Leftarrow$

$$\min_{x_p} -\sum_{i,j\in V}\sum_{p\in P_{ij}} x_{p_{ij}}^{K}$$

s.t.

$$\sum_{p\in P_{ij}} x_p \le d_{ij} \rightarrow \sum_{K=1}^{\overbrace{num(path\ ij)}^{\text{تعداد مسیرها بین i و j}}} x_{p_{ij}}^{K} - d_{ij} \le 0 \quad \forall i,j\in V$$

$$x_{uv}+x_{vu} \le \frac{c_{uv}}{\Delta} \quad \forall (u,v)\in E \rightarrow \sum_{p\in(u,v)} x_{p_{ij}}^{K} + \sum_{p'\in(v,u)} x_{p'_{ij}}^{K} - \frac{c_{uv}}{\Delta} \le 0 \rightarrow \underbrace{\sum_{p\in u,v}}_{\text{رفت و برگشت}} x_{p_{ij}}^{K} - \frac{c_{uv}}{\Delta} \le 0$$

$$x_{uv} = x_{vu} \quad \forall (u,v)\in E \rightarrow \sum_{p\in(u,v)} x_{p_{ij}}^{K} - \sum_{p'\in(v,u)} x_{p'_{ij}}^{K} = 0 \quad \text{و} \quad x_{p_{ij}}^{K} \ge 0$$

تجزیه شروط به فلو عبوری

$$t_{ij}^{K} \rightarrow \sum_{K=1}^{num(path\,ij)} t_{ij}^{K} = 0 \qquad t_{ijK}^{\mu_{uv}} \rightarrow \sum^{num(path \in u,v)} t_{ijK}^{\mu_{uv}} = 0$$

$$t_{ijK}^{\lambda_{uv}} \rightarrow \sum^{num(path \in u,v)} t_{ijK}^{\lambda_{uv}} = 0$$

$$\begin{cases} \min\limits_{x_{p_{ij}}^{K}} \; x_{p_{ij}}^{K} \\ \forall ij \rightarrow x_{p_{ij}}^{K} - \dfrac{d_{ij}}{num(path\,ij)} \leq t_{ij}^{K} \\ \forall e \text{ if } e \in x_{p_{ij}}^{K} \rightarrow x_{p_{ij}}^{K} - \dfrac{C_{vu}}{num(p \in u,v)} \leq t_{ijK}^{\mu_{uv}} \\ \forall e \text{ if } e \in x_{p_{ij}}^{K} \longrightarrow x_{p_{ij}}^{K} = t_{ijK}^{\lambda_{uv}} \quad \text{و} \quad x_{ijp}^{K} \geq 0 \end{cases}$$

تجزیه مسئله ⟸

برای هر (i,j و K) مسئله

برای هر flow

نوشتن الگوریتم:

حل جواب بهینه هر عامل $\phi_{ij}^{K}(t)$ است که t برداری شامل t_{ij}^{K} و $t_{ijK}^{\mu_{uv}}$ و $t_{ijK}^{\lambda_{uv}}$ است.

$$\min_{t} \sum_{i,j \in V} \sum_{p \in P_{ij}}^{K \in P} \phi_{ij}^{K}(t)$$

مسیرهایی که از دو نود i و j می گذرد ⟸ CO با این مسئله را حل کند

$$S.t \quad \sum_{K=1}^{num(path\,ij)} t_{ij}^{K} = 0 \;\; \forall ij \in V, \quad \sum^{path \in (u,v)} t_{ijK}^{\mu_{uv}} = 0 \;\; \forall (u,v) \in E, \quad \sum^{(path \in u,v)} t_{ijK}^{\lambda_{uv}} = 0 \;\; \forall (u,v) \in E$$

برای آپدیت t توسط CO عامل ها در هر مرحله با زیرگرادیان $\phi_{ij}^{K}(t)$ خود را به CO بر میگردانند که تابعی از λ^* و μ^* است که از t لحظه قبل خود به دست آورده اند.

الگوریتم:

۱- مقدار دهی اولیه t_0 ۲- محاسبه $d(\phi_{ij}^{K}(t(K)))$ ها یعنی زیرگرادیان در هر $t(K)$ برای تمام عامل ها ۳- به روزرسانی $t(K)$ با توجه به زیرگرادیان ها و $\alpha(K)$ مشخص شده ۴- $K = K+1$ ۵- رفتن به مرحله ۲ تا شرط توقف

استفاده از این روش برای حل غیر منطقی و دشوار به نظر می رسد زیرا در بهینه سازی دوم خود شرط داریم و کوردینیتور بایستی یک مسئله با شروط نه چندان آسان را حل کند و آپدیت ها باید در فضای شدنی t تصویر شوند که تصویر کردن معمولا کار دشواری به حساب می آید

Dual:

برای راحتی در محاسبات و مشتق گیری تابع را برای هر سوال f_i در نظر می‌گیریم و تابع را $\log(x)$ و $U(x)$ در نظر می‌گیریم.

تابع لاگرانژین:

$$L(x_p, \lambda, \mu) = \sum_{i,j\in V}\sum_{p\in P_{ij}} -\log(x_{p_{ij}}) + \sum_{ij\in V}\lambda_{ij}\Big(\sum_{p\in P_{ij}} x_{p_{ij}} - d_{ij}\Big) + \sum_{i,j\in V}\sum_{p\in P_{ij}}\lambda_{ijp}(-x_p)$$
$$+ \sum_{(u,v)\in E}\lambda_{uv}\Big[\sum_{\substack{p\in P_{ij}\\(u,v)}} x_p + \sum_{\substack{p'\in P_{ij}\\(v,u)}} x_{p'} - \frac{C_{uv}}{\Delta}\Big] + \sum_{(u,v)\in E}\mu_{uv}\Big[\sum_{\substack{p\in P_{ij}\\(u,v)}} x_p - \sum_{\substack{p'\in P_{ij}\\(v,u)}} x_{p'}\Big]$$

بازنویسی معادله بالا به طوری که بتوان نسبت به x_p جدا کرد:

$$\sum_{i,j\in V}\sum_{p\in P_{ij}}\Big(\underbrace{-\log(x_p) + \lambda_{ij}x_p + \lambda_{ijp}(-x_p) + \sum_{(u,v)\in p}\lambda_{uv}x_p + \sum_{(u,v)\in p}(\mu_{uv}-\mu_{vu})x_p}_{g(x_p,\mu,\lambda)}\Big) - \sum_{(u,v)\in E}\lambda_{uv}\frac{C_{uv}}{\Delta}$$
$$- \sum_{i,j\in V}\lambda_{ij}d_{ij}$$

برای پیدا کردن min تابع بالا نسبت به x_p از $g(x_p,\mu,\lambda)$ مشتق و مساوی صفر قرار می‌دهیم: چون نسبت به x_p جدا پذیر است هر کدام را جدا بهینه می‌کنیم:

$$\frac{-1}{x_p \ln(10)} + \lambda_{ij} - \lambda_{ijp} + \sum_{(u,v)\in p}\lambda_{uv} + \sum_{(u,v)\in p}(\mu_{uv}-\mu_{vu}) = 0 \longrightarrow$$ برای تمام path ها

$$\Rightarrow x_p^* = 1\Big/\Big(\lambda_{ij} - \lambda_{ijp} + \sum_{(u,v)\in p}\lambda_{uv} + \sum_{(u,v)\in p}(\mu_{uv}-\mu_{vu})\Big) \longrightarrow$$ مقدار بهینه هر عامل برای هر μ و λ ای

مشتق تابع لاگرانژین را برحسب λ و μ موارد نظر می‌سنجیم و محاسبه می‌کنیم: یک منفی به خاطر max گرفتن در آن ضرب شده است.

λ را بردار تمام ضرایب μ و λ در نظر گرفتیم

$$d^{ij} = \sum_{ij\in V} d_{ij} - \sum_{i,j\in V}\sum_{p\in P_{ij}} x_p^*(\lambda)$$

$$d^{\mu_{uv}} = -\sum_{(u,v)\in E}\Big(\sum_{\substack{p\in P_{ij}\\(u,v)}} x_p^*(\lambda) - \sum_{\substack{p'\in P_{ij}\\(v,u)}} x_p^*(\lambda)\Big)$$

$$d^{\lambda_{ijp}} = \sum_{i,j\in V}\sum_{p\in P_{ij}} x_p^*(\lambda)$$

$$d^{\lambda_{uv}} = \sum_{(u,v)\in E}\frac{C_{uv}}{\Delta} - \sum_{i,j\in V}\sum_{p\in P_{ij}}\sum_{(u,v)\in p} x_p^*(\lambda) \qquad \lambda = \begin{bmatrix}\lambda_{ij}\\ \lambda_{ijp}\\ \lambda_{uv}\\ \mu_{uv}\end{bmatrix} \quad d = \begin{bmatrix}d^{\lambda_{ij}}\\ d^{\lambda_{ijp}}\\ d^{\lambda_{uv}}\\ d^{\mu_{uv}}\end{bmatrix}$$

الگوریتم توزیع شده: 1- بردار λ را در $K=0$ مقداردهی اولیه کنید

2- هر عامل زیرگرادیان خود را محاسبه کند:

$$d_{\lambda_{ij}p}^{(K)} = x_p^*(\lambda), \quad d_{\lambda_{ij}pp}(K) = x_p^*(\lambda), \quad d_{\lambda_{uv}p}(K) = \sum_{(u,v)\in p} x_p^*(\lambda)$$

$$d_{\mu_{uv}p}^{(K)} = \sum_{(u,v)\in p} x_p^*(\lambda) - \sum_{(v,u)\in p} x_p^*(\lambda)$$

3- بروزرسانی λ در C_0:

$$d_{\lambda_{ij}}(K) = \sum_{i,j\in V} d_{ij} - \sum_{i,j\in V} \sum_{p\in \mathcal{P}_{ij}} d_{\lambda_{ij}p}^{(K)} \qquad d_{\lambda_{ij}p}^{(K)} = +\sum_{i,j\in V} \sum_{p\in \mathcal{P}_{ij}} d_{\lambda_{ij}p}^{(K)}$$

$$d_{\lambda_{uv}}(K) = \sum_{(u,v)\in E} \frac{c_{uv}}{\Delta} - \sum_{i,j\in V} \sum_{p\in \mathcal{P}_{ij}} d_{\lambda_{uv}p}^{(K)} \qquad d_{\mu_{uv}}(K) = \sum_{i,j\in V} \sum_{p\in \mathcal{P}_{ij}} d_{\mu_{uv}p}^{(K)}$$

$$\lambda = \left[\lambda + \alpha(K)\, d(K)\right]_+$$ μ_{uv} نباید + بشود

4- $K=K+1$ 5- برو به 2 تا شرط توقف

Dual-primal:

تابع لاگرانژ را با حذف قیود مربوط به فلو ها و مثبت کردن ضریب قید تعادل با اضافه کردن دو قید تا تساوی به جای یک قید تساوی به صورت زیر تعریف می کنیم.

$U(x) = x$. Consider the partial Lagrangian of the LP:

$$\begin{aligned}
L(\mathbf{x}, \lambda, \mu) = & \sum_{i,j\in V} \sum_{p\in\mathcal{P}_{i,j}} x_p \\
& - \sum_{(u,v)\in E} \lambda_{uv} \left[\sum_{\substack{p\in\mathcal{P}: \\ (u,v)\in p}} x_p + \sum_{\substack{p'\in\mathcal{P}: \\ (v,u)\in p'}} x_{p'} - \frac{c_{u,v}}{\Delta} \right] \\
& - \sum_{(u,v)\in E} \mu_{uv} \left[\sum_{\substack{p\in\mathcal{P}: \\ (u,v)\in p}} x_p - \sum_{\substack{p'\in\mathcal{P}: \\ (v,u)\in p'}} x_{p'} \right] \\
& - \sum_{(u,v)\in E} \mu_{vu} \left[\sum_{\substack{p\in\mathcal{P}: \\ (v,u)\in p}} x_p - \sum_{\substack{p'\in\mathcal{P}: \\ (u,v)\in p'}} x_{p'} \right],
\end{aligned}$$

ضریب مربوط به ظرفیت لینک ها به جهت کانال مرتبط نیست ولی ضریب تعادل متناسب با جهت کانال متفاوت است.

معادله بالا را به صورت زیر بازنویسی می کنیم:

$$L(\mathbf{x}, \lambda, \mu) = \sum_{i,j \in V} \sum_{p \in \mathcal{P}_{i,j}} x_p \left(1 - \sum_{(u,v) \in p} \lambda_{uv} - \sum_{(u,v) \in p} \mu_{uv} + \sum_{(v,u) \in p} \mu_{vu}\right) + \sum_{(u,v) \in E} \lambda_{uv} \frac{c_{uv}}{\Delta}.$$

تا بتوان آن را به فلو های عبوری تجزیه کنیم و با توجه به تغییر معادله بالا ضریب زیر را تعریف می کنیم:

$$z_p = \sum_{(u,v):(u,v) \in p} \left(\lambda_{uv} + \mu_{uv} - \mu_{(v,u)}\right)$$

ابدیت پرایمال:

$$x_p(t+1) = x_p(t) + \alpha(1 - z_p(t)) \quad (9.3)$$

$$x_p(t+1) = \mathrm{Proj}_{\chi_{i,j}}(x_p(t+1)), \quad (9.4)$$

where Proj is a projection operation on to the convex set $\{x_p : \sum_{p:p \in \mathcal{P}_{i,j}} x_p \leq d_{i,j}, x_p \geq 0 \ \forall p\}$, to ensure the rates are feasible.

نود های ابتدایی و انتهایی مسیر مقدار فلو فرستاده خود را متناسب با شرایط تغییر کرده در شبکه اپدیت می شنود.

اپدیت دوال:

مقدار تخمین هر نود میانی از شرایط کانال به صورت زیر انجام می شود

$$w_{uv}(t) = \sum_{\substack{p \in \mathcal{P}: \\ (u,v) \in p}} x_p(t) + \sum_{\substack{p' \in \mathcal{P}: \\ (v,u) \in p'}} x_{p'}(t) - \frac{c_{uv}}{\Delta}$$

$$y_{uv}(t) = \sum_{\substack{p \in \mathcal{P}: \\ (u,v) \in p}} x_p(t) - \sum_{\substack{p' \in \mathcal{P}: \\ (v,u) \in p'}} x_{p'}(t)$$

نود های میانی پارامتز های دوال شبکه را آپدیت می کنند تا شرایط شبکه را به سمت هدف خود تغییر دهند و ایم تغییر در اپدیت فلو نود های ابتدایی ظاهر شود:

حمید رضا کاشانی **پروژه یک** 810100441

$$\lambda_{uv}(t+1) = \left[\lambda_{uv}(t) + \eta w_{uv}(t)\right]_{+}$$

$$\mu_{uv}(t+1) = \left[\mu_{uv}(t) + \kappa y_{uv}(t)\right]_{+}$$

$$\mu_{vu}(t+1) = \left[\mu_{vu}(t) - \kappa y_{uv}(t)\right]_{+}.$$

این روش سیستم ما رو به سمت نقطه زیتی تابع لاگرانژ حرکت می دهد و برای استفاده عملی کارا تر است .

حل سنترالایز مسئله:

مسئله از نظر امکان حل بسته داری حل هست ولی به علت تعداد بالای قیود و ترکیب شدن پارامترها در معادلات حل دستگاه معادلاتی که در آخر به دست می آید نا ممکن یا بسیار دشوار است.

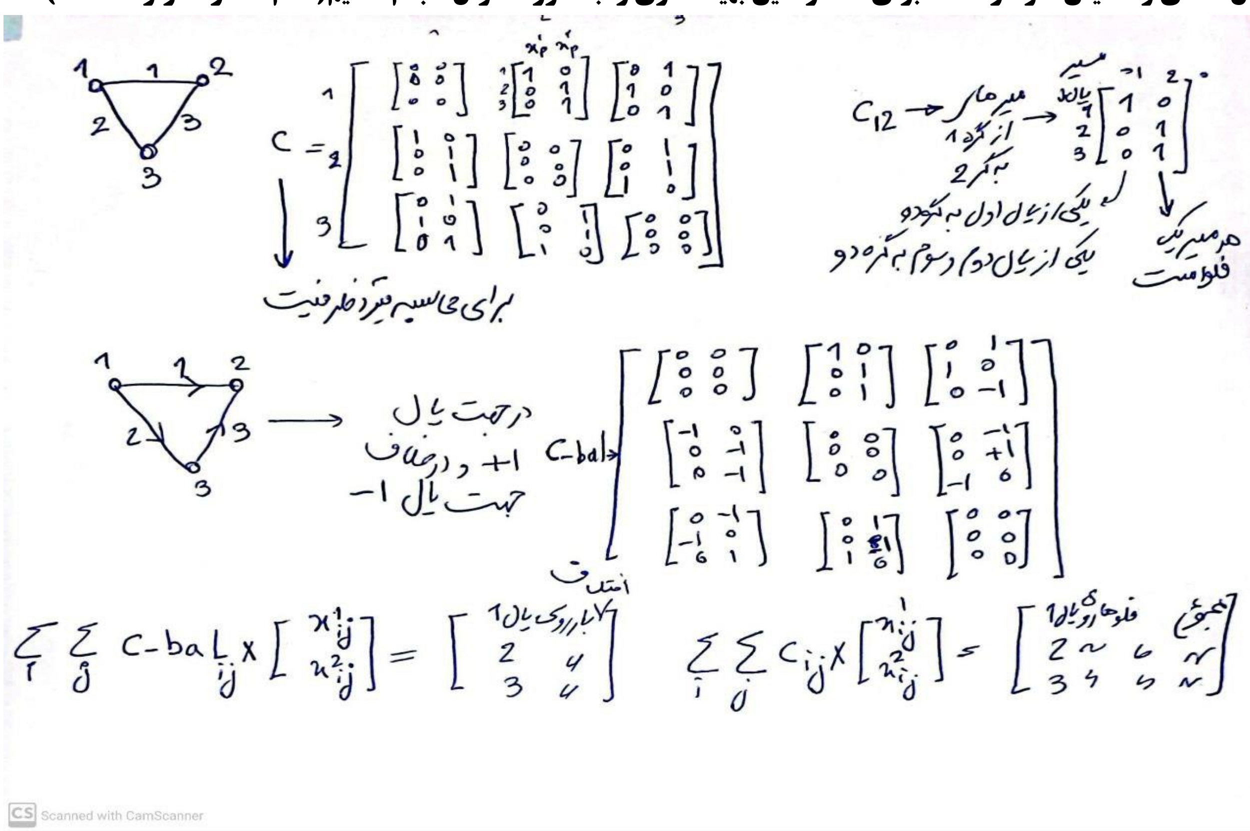
حل به صورت پرایمال و با روش های عددی به علت نیاز به تصویر سازی روی اشتراک قیود و زیاد بودن قیود کار نامعقولی است.

حل به صورت دوال هم بسیار شبیه حل دوال گسسته است با این تفاوت که دیگر عاملی وجود ندارد و به صورت مرکزی زیر گرادیان ها محاسبه می شود که حل برای سه نود و با ارتباط کامل و تابع log را انجام شده که درادامه در کنار شبیه سازی دیستریبیوتد ارائه می شود.

حل ADMM:

مسئله دارای قیود بسبار زیادی است و پیچیدگی بالایی دارد ، استفاده از روش ADMM هم برای مسائلی که عامل در قید باهم وابستگی دارند ونه در متغییر عمومی دشوار و پیچیدگی بالا تر پیدا میکند و در کلاس هم حل آن به مقالات ارجاع داده شد پس حل admm معقول به نظر نمی رسد.

حل عددی و نمایش نمودار ها: ما برای سه گره این بهینه سازی را به صورت دوال انجام دادیم(تمام کد از صفرنوشته شده)



حمید رضا کاشانی **پروژه یک** 810100441

توابع نوشته شده:

بهینه مقداری که هر عامل می فرستد:

```
def f_xstar_agent(i,j,k,lij,lpij,luv,muv,mvu,d):
  if lij[i,j] - lpij[i,j,k]+ (c[i,j,:,k])@luv + c_bal[i,j,:,k]@(muv-
mvu)==0:
    xstar=d[i,j]
  else:
    xstar=((1)/((lij[i,j] - lpij[i,j,k]+ (c[i,j,:,k])@luv + c_bal[i,j,:,k]
@(muv-mvu) )*np.log2(10))).reshape([1])
  if xstar>d[i,j]:
    xstar=d[i,j]
  if xstar<=0 :
    xstar=0.0000000000001
  return xstar
```

ابدیت در کوردینیتور:

```
def co_update(xstar,lij,lpij,luv,muv,mvu,n):
  alpha=1/(n+1)
  for i in range(3):
    for j in range(3):
      if i!=j:
        lij[i,j]=lij[i,j]+alpha *(np.sum(xstar[i,j])-d[i,j])
        if lij[i,j]<0:
          lij[i,j]=0.0
  for i in range(3):
    for j in range(3):
      if i!=j:
        for k in range(2):
          lpij[i,j,k]=lpij[i,j,k]-alpha*xstar[i,j,k]
          if lpij[i,j,k]<0:
            lpij[i,j,k]=0.0
  s =c@xstar +(c@xstar).transpose(1,0,2,3)
  sn =c_bal@xstar +(c_bal@xstar).transpose(1,0,2,3)
  sn_ =-c_bal@xstar - (c_bal@xstar).transpose(1,0,2,3)
  gluv =np.array([[0.],[0.],[0.]])
  gmuv =np.array([[0.],[0.],[0.]])
  gmvu =np.array([[0.],[0.],[0.]])
  for i in range(3):
    for j in range(3):
      if j>i :
        for k in range(ne):
          gluv[k]+=s[i,j,k]
          gmuv[k]+=sn[i,j,k]
          gmvu[k]+=sn_[i,j,k]
```

```
  for k in range(ne):
    luv[0,0,k]= luv[0,0,k] +alpha*(gluv[k]-CA[k])
    if luv[0,0,k]<0:
      luv[0,0,k]=0
    muv[0,0,k]= muv[0,0,k] + alpha*(gmuv[k])
    #print(muv,"\n-----------")
    mvu[0,0,k]= mvu[0,0,k] - alpha*(gmvu[k])
    if muv[0,0,k]<0 :
      muv[0,0,k]=0
    if mvu[0,0,k]<0 :
      mvu[0,0,k]=0
  return lij,lpij,luv,muv,mvu
```

و تابع محاسبه مجموع فلو :

```
def sum_log(x):
  yy=[]
  for i in range(m):
    for j in range(m):
      if i!=j :
        yy+=[[[True],[True]]]
      else:
        x[i,j]=np.array([[np.nan],[np.nan]])
        yy+=[[[False],[False]]]
  yy=np.array(yy).reshape([m,m,2,1])
  for i in range(3):
    for j in range(3):
      for k in range(2):
        if i!=j :
          if x[i,j,k]==0:
            x[i,j]=0.00001
  ret =(np.sum(np.log10(x),where=yy))
  return ret
```

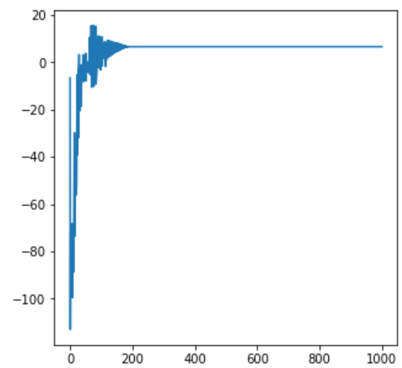
حال پاسخ مسئله برای alpha برابر 1/n به صورت گسسته و سنترال به صورت زیر است:

نمودار بلا نمودار مجموع لگاریتم تمام فلو هاست و تمودار های پایینی نمودار همگرایی دوازده فلو موجود در شبکه سه گره ای است

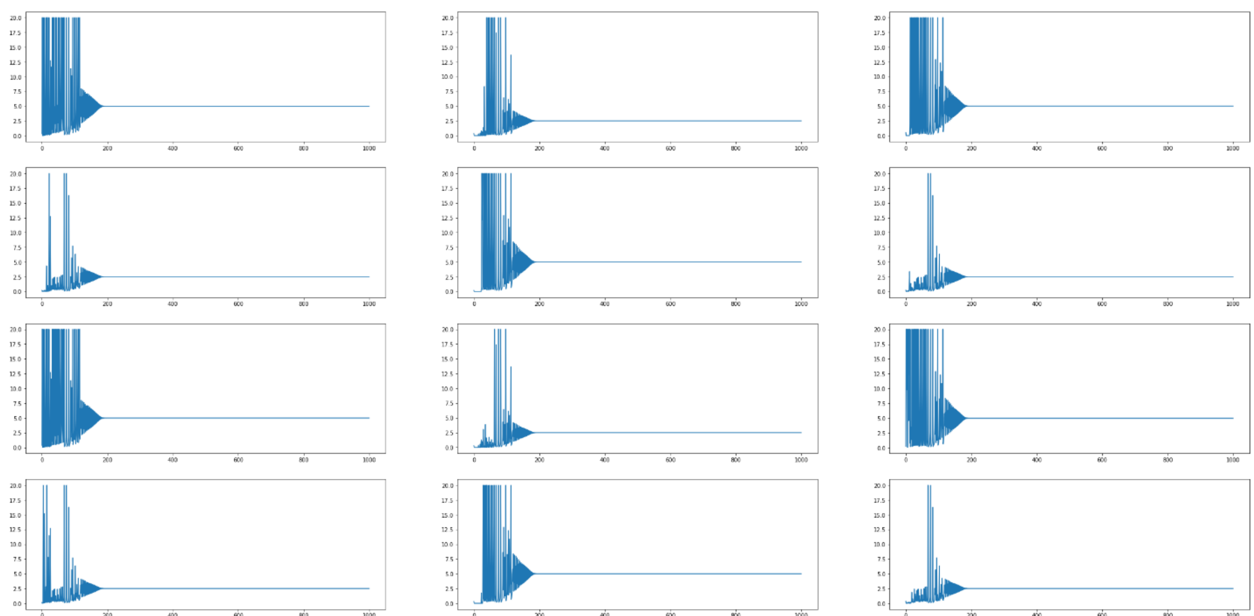
علت این که مقدار تابع ما در زمان هایی از مقدار همگرایی همگرایی بالا تر است این است که قبل از همگرایی شرایط شدنی براقرار نیست و در مواردی که شرایط ما برقرار نباشد می تواند مقدار آن بالا تر از مقدار بهینه بشود.

ابتدا برای سنترالایز نمودار را می دهم و سپس برای دیسنترالایز اما حل دوال ما برای هر دو یکی است و جواب ها کاملا شبیه به هم و تفاوت آنها فقط در شکل رسیدن به آن و توابع نوشته شده آنها است . و عامل ها در دیسنترالایز باید مقدار بهینه فلو خود را در هر مرحله محاسبه کرده و به کوردینیتور فرستاده می شود برای برای آپدیت. اما برای سنترالایز همه در یک جا به صورت مرکزی محاسبه می شود.

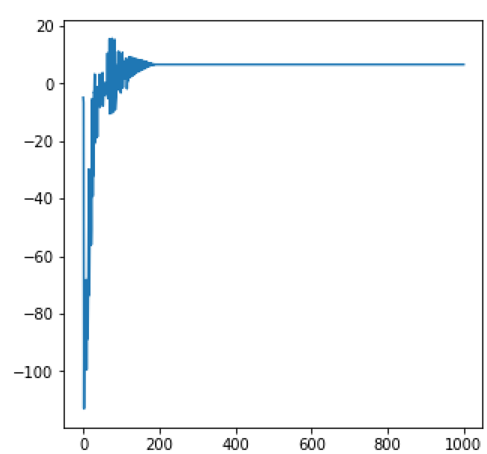
حل سنترالایز:



همگرایی هر کدام از فلو ها: که به ترتیب ماتریس آن است از چپ به راست فلو 12 اولی و دومی فلو 13 اولی و دومی فلو 23 اولی و دومی و ترانهاده آنها به ترتیب ادامه پیدا کرده است:

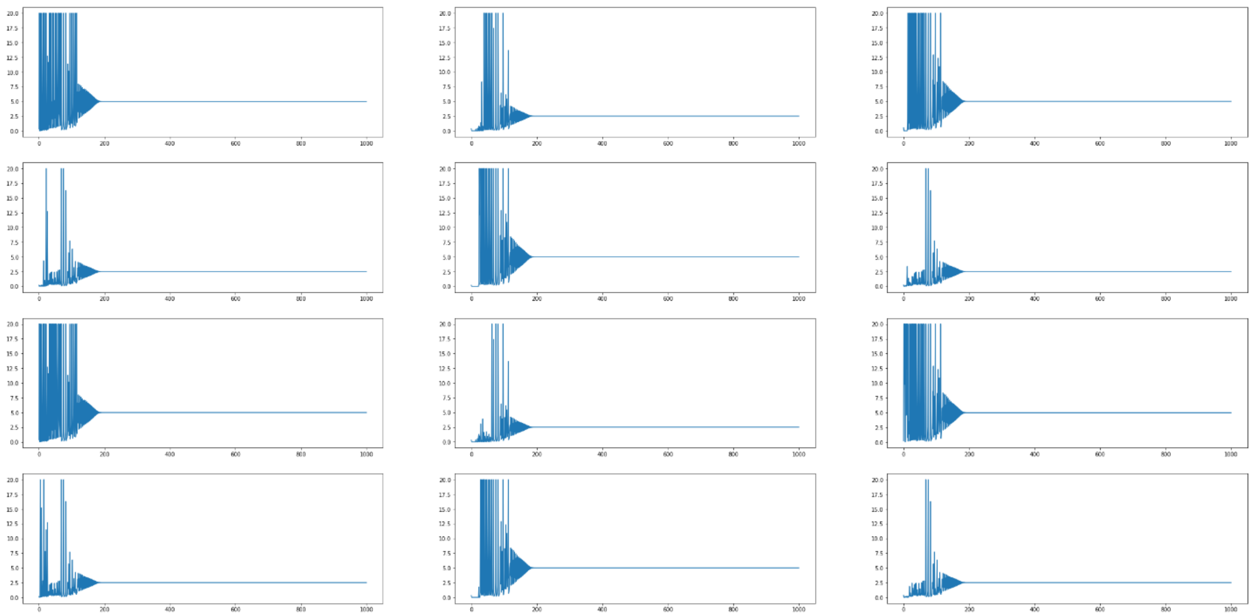


حل دیستریبیوتد:

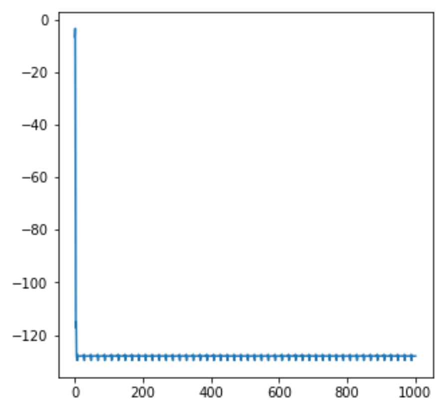


حمید رضا کاشانی **پروژه یک** 810100441

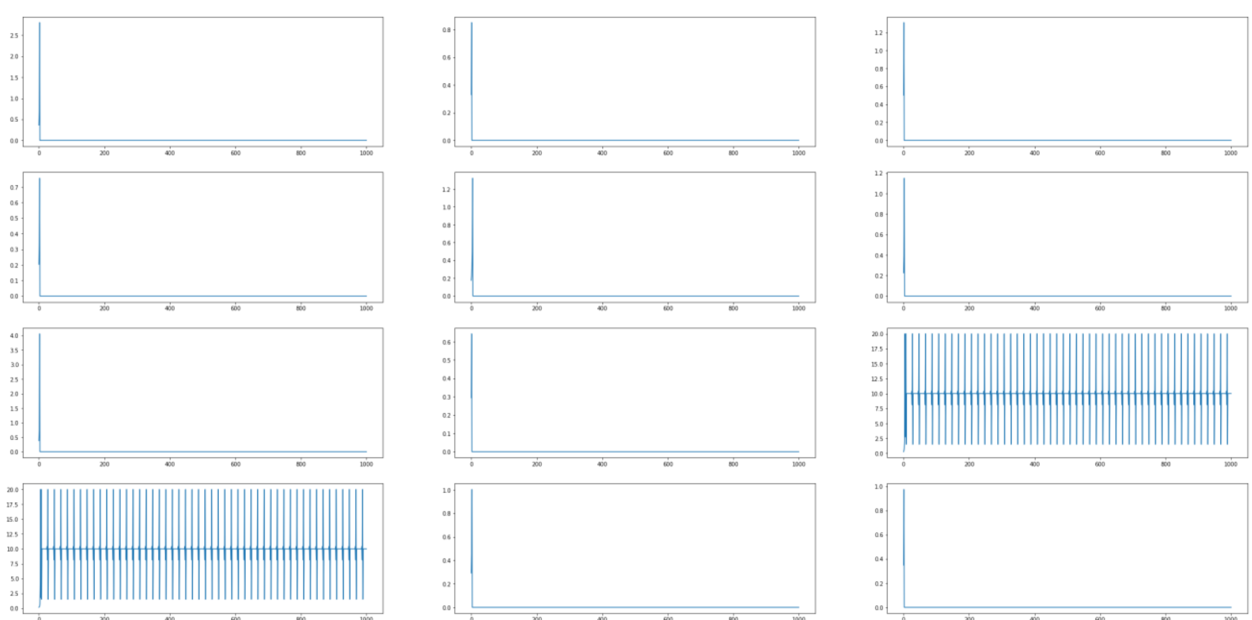
فلو:



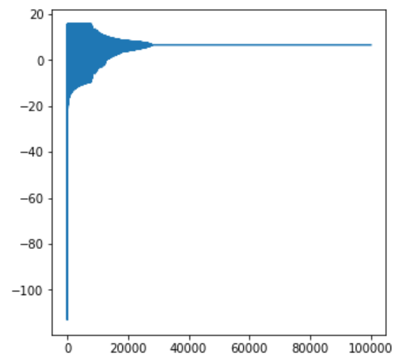
حال حل برای alpha ثابت 0.01:



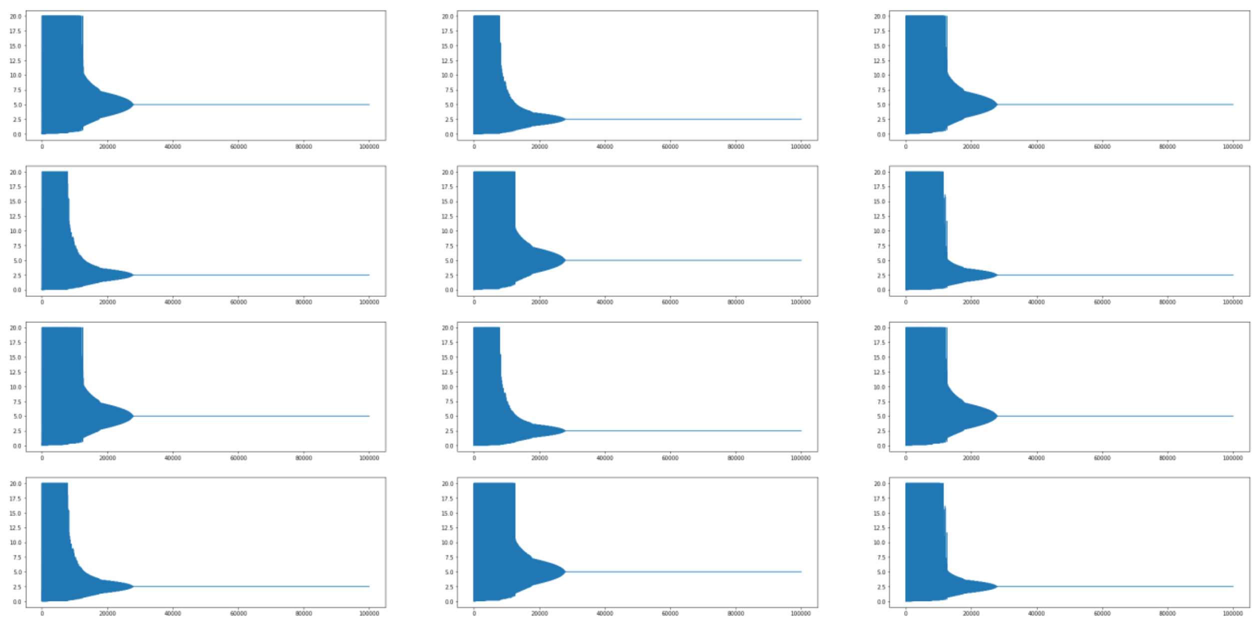
فلو ها:



حال برای alpha برابر(1/sqrt(n : مقدار تکرار را از 1000 به 100000 افزایش دادیم.



فلو ها:



همانطور که مشاهده کردید این اخری در تکرار بسیار بیشتربه همگرایی رسید و برای alpha ثابت همگرای جواب بهینه نبود.

Federated learning

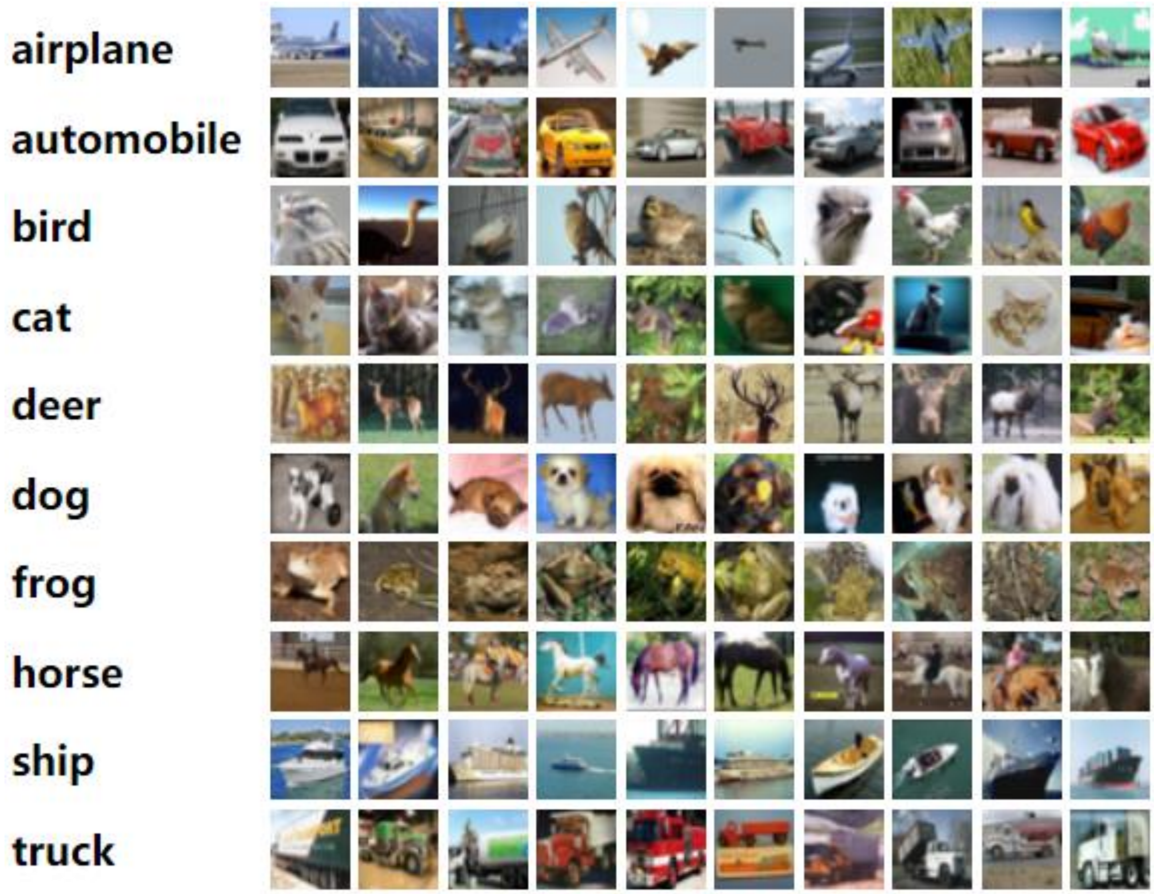
برای جبران کاستی های پروژه تصویب شده برای پروپوزال تصمیم گرفتن یک قسمت نسبتا مفصلی روی فدریتد لرنینگ به آن اضافه کنم .

در این قسمت قصد هست روی داده های تصویری و در مدل ساده شبکه عصبی کانولوشنری برای ده عامل فدریتد لرنینگ با استفاده از روش های fedavg و fedprox در حالتی که داده ها مساوی و نا مساوی بین عامل ها تقسیم شده اند لرنیگ انجام دهیم.

و همینطور این فدریتد لرنینک را برای learning rate های امده در درس امتحان می کنیم تا حاصل را نشاهده کنیم.

حمید رضا کاشانی **پروژه یک** 810100441

داده های ما 50000 داده تصوری با اسم cifar10 که مجموعه ای 10 گروه مختلف حیوان در آن لیبل گذاری شده است.



حال داده های خود را به صورت نا متقارن تقسیم بندی می کنیم:

```
XX=np.split(x_train,[8000,19000,20000,23000,24500,28000,30000,35000,44400])
YY=np.split(y_train,[8000,19000,20000,23000,24500,28000,30000,35000,44400])
```

و اماده سازی های اولیه را روی داده انجام می دهیم.

حال مدل شبکه عصبی کانولوشنری خود را تعریف می کنیم:

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 32)	416
conv2d_5 (Conv2D)	(None, 32, 32, 16)	2064
flatten_2 (Flatten)	(None, 16384)	0
dense_6 (Dense)	(None, 256)	4194560
dense_7 (Dense)	(None, 256)	65792
dense_8 (Dense)	(None, 10)	2570

Total params: 4,265,402

Trainable params: 4,265,402

Non-trainable params: 0

پیاده سازی اول پیاده سازی fedavg با داده های نا متقارن:

پارامتر های لرنینگی مدل:

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

تابع کوردینیتور که عمل میانگین وزندار روی وزن های شبکه عصبی انجام می دهد:

```
def avg_w_co(a,XXX,n):
  w=np.zeros(np.array(a[0]).shape)
  for i in range(10):
    wi=XXX[i].shape[0]/(50000)
    w= w+(wi)*np.array(a[i])
  w=list(w)
  print(n+1,'\n','_________________________')
  return w
```

هر ده مدل را در یک لوپ قرار می دهیم و در اخر وزن هر ده مدل را با یکدیگر میانگین خواهیم گرفت.طبق الگوریتم زیر:

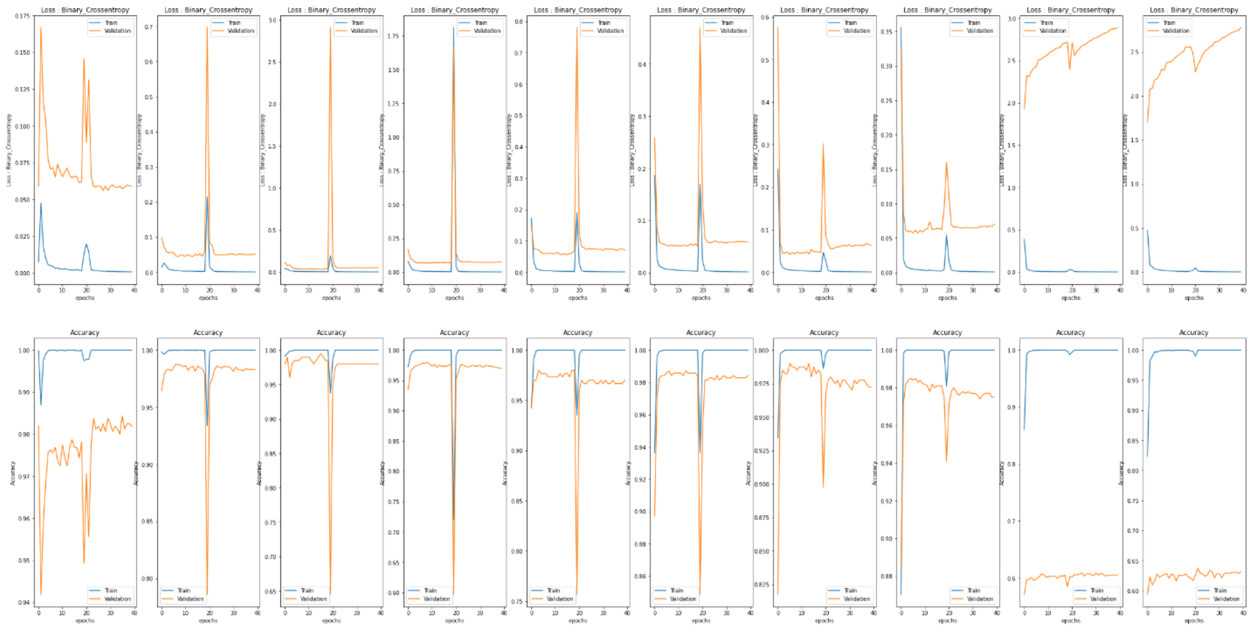
Algorithm 1 FederatedAveraging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): // *Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

اجرای کامل این الگوریتم برای 40 ایتریشن حدود یک ساعت زمان می برد.

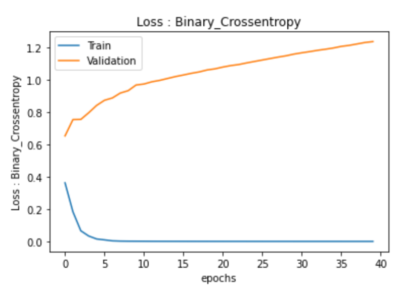
و ما حاصل را به صورت بیست نمودار نشان داده ایم که هر جفت زیر هم دقت و خطای هر عامل را نمایش می دهند:

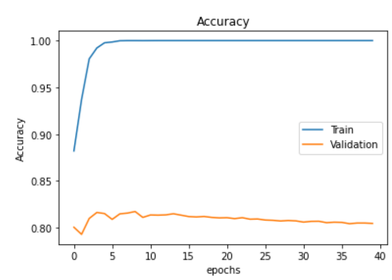


دقت و خطا برای داده های تست ما هم برای یک عامل به صورت نمونه این است:

```
loss: 2.7987 - accuracy: 0.6203
```

برای اینکه بتوان مقایسه ای انجام داد ما باید با همین شبکه کل داده ها را به صورت یه جا لرن بکنیم و حاصل را مقایسه کنیم که زمان انجام این کار هم کمی کمتر از قبلی و در همان اوردر است:





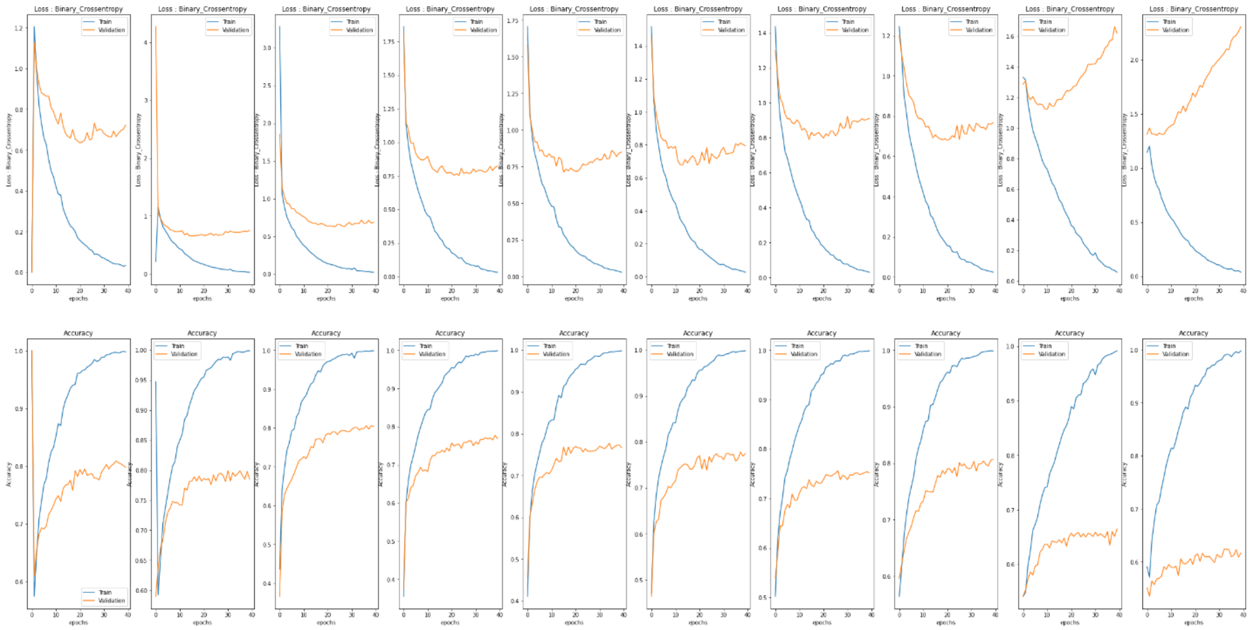
دقت و خطا برای داده تست در حالت کلی:

```
loss: 4.0433 - accuracy: 0.6072
```

نتیجه گیری :

همینطور که مشاهده می کنیم برای fedavg مقدار یاد گیری ها خیلی از هم فاصله دارند و این به خاطر نا متقارن بودن داده هایمان از هم است به طوری که در عامل هایی بیشتر از شبکه کلی و در عامل هایی کمتر از شبکه کلی دقت گرفتیم. اما نکته جالب اینجا است که عامل اخرما با این که دقت کمتری روی داده های ولیدشن داشته است اما دقت بالا تری در داده های تست نسبت به شبکه اصلی داشته است و این می تواند نشان دهد که فدریتد لرنیگ حتی عملکذد بهتری در عمومی سازی یادگیری دارد.

حال دوباره همین عمل یادگیری داده ها برای عانل هارا با داده های مساوی و fedavg انجام می دهیم:



دقت و خطای تست عامل آخر:

```
loss: 2.4712 - accuracy: 0.5740
```

همانطور که مشاهده می کنید عمکرد یادگیری ما تا حد خوبی به یکدیگر شبیه شدند اما مقدار دقت برای داده های تست ما کم شدند .

پیاده سازی الگوریتم fedprox روی دادههای نا متقارن:

تفاوت این روش با روش قبل در تابع loss استفاده شده برای یادگیری شبکه های عصبی است و ما در این تابع لاس یه توان دو خطای وزن هر عامل از میانگین عامل ها را هم در نظر می گیرم با اعمال ضریبی بع آن می توانیم مشخص کنیم که تاثیر بیشتر یا کمتر ی داشته باشد به این صورت که این عامل اضافه شده فاصله وزن ها و شبکه ها را به هم کم می کنه ولی باد گیری را سخت تر مب کند . الگوریتم آن:

Algorithm 2 `FedProx` (Proposed Framework)

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$

for $t = 0, \cdots, T-1$ **do**

Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)

Server sends w^t to all chosen devices

Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w \; h_k(w;\; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$

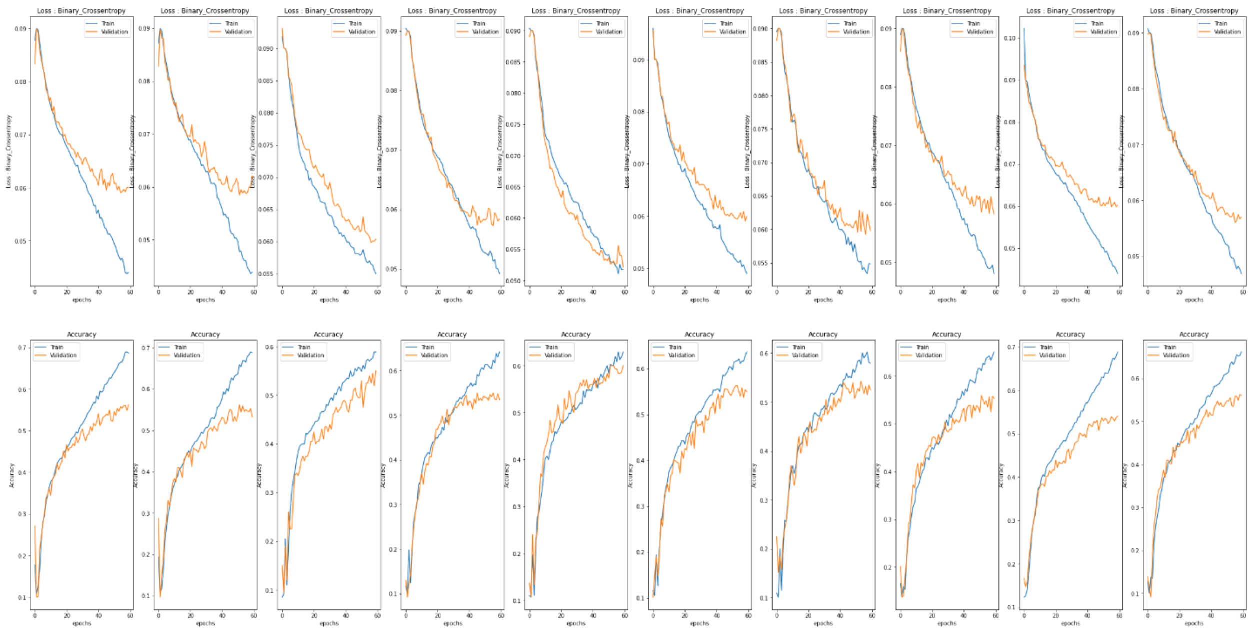
Each device $k \in S_t$ sends w_k^{t+1} back to the server

Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$

end for

پیاده سازی این الگوریتم با کتابخانه keras کمی مشکل به همین علت کد نشوته شده بند در این قسمت کمی کثیف است اما کاراست.

انجام این یادگیری برای 60 ایتریشن برای هر بار یک و نیم ساعت زمان لازم دارد:



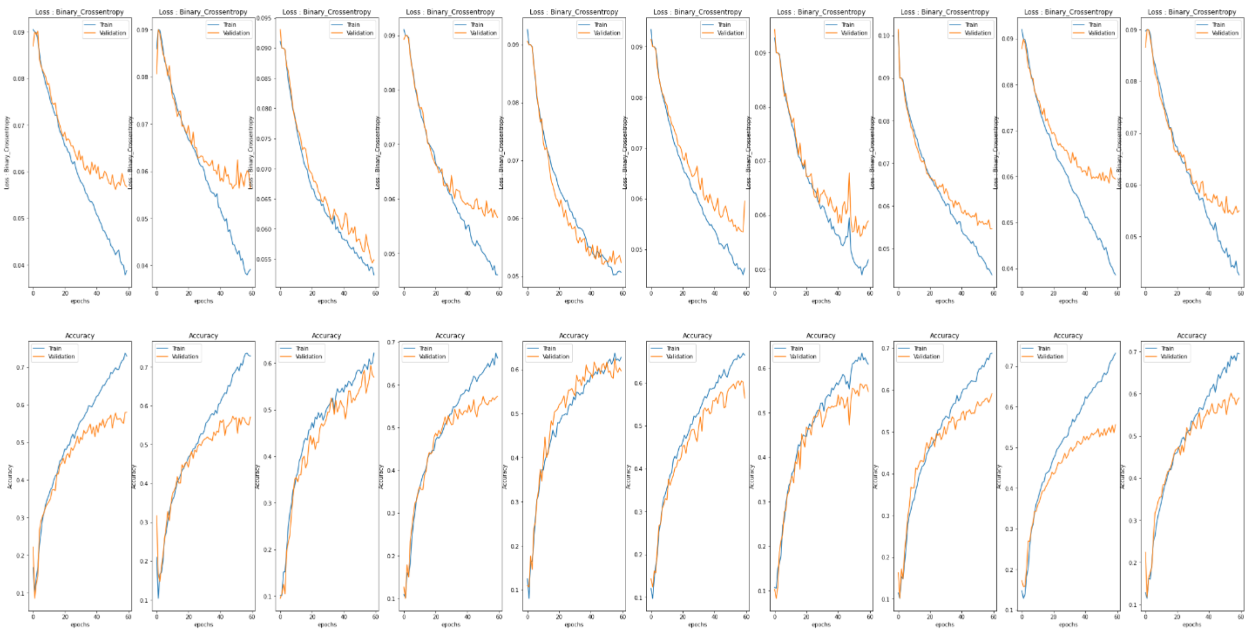
دقت و خطا برای داده های تست برای هر ده عامل:

```
loss: 0.0596 - accuracy: 0.5530
loss: 0.0624 - accuracy: 0.5226
loss: 0.0601 - accuracy: 0.5419
loss: 0.0590 - accuracy: 0.5525
loss: 0.0588 - accuracy: 0.5509
loss: 0.0579 - accuracy: 0.5600
loss: 0.0596 - accuracy: 0.5470
loss: 0.0578 - accuracy: 0.5615
loss: 0.0594 - accuracy: 0.5568
loss: 0.0587 - accuracy: 0.5570
```

همیطور که مشاهده می کنید شباهت عامل بسیار بهبود پیدا کرد و همینطور یادگیری داده های وبیدیشن مت هم بهبود پیدا کرده است و همچنین پس از 60 ایتریشن هنوز ثابت نشده اند که به معنی دقت بالا تر از آن چیزی است که نوشته شده.

تغییر Miu:

حال با miu ضریب جمله وزن های لاس باز یکرده ایم و آن را به ¼ کاهش داده ایم که قاعدتا باید شباهت عامل ها را کمتر ولی یادگیری را بهتر کند:

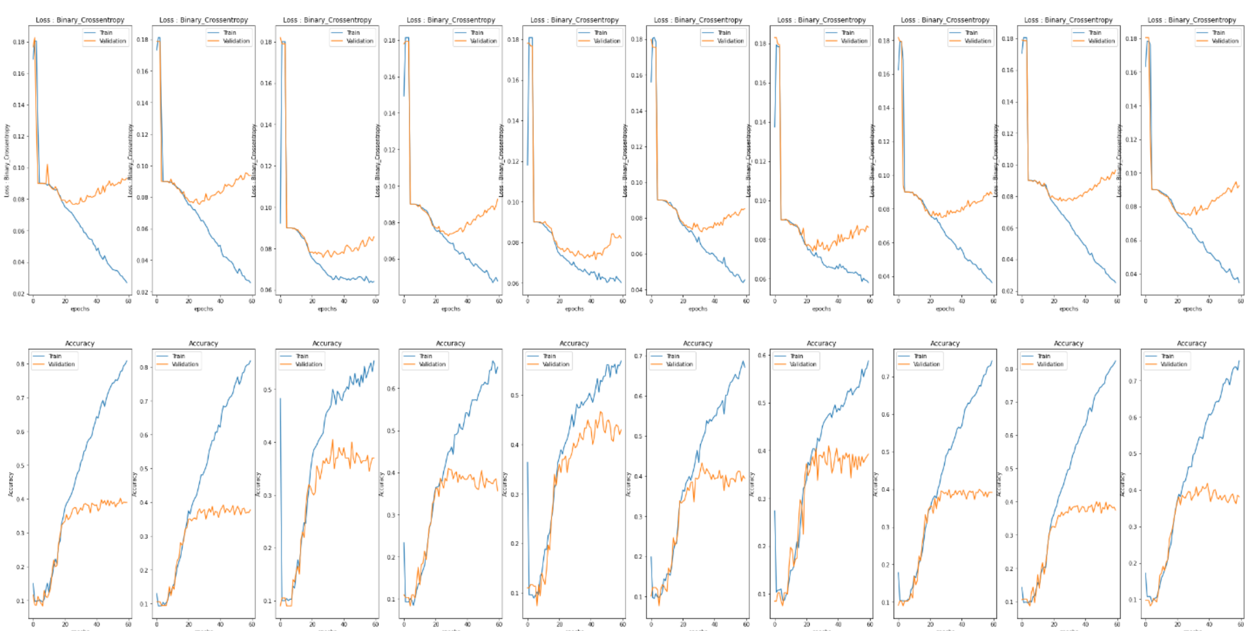


دقت و خطا تست:

```
loss: 0.0569 - accuracy: 0.5748
loss: 0.0563 - accuracy: 0.5779
loss: 0.0561 - accuracy: 0.5774
loss: 0.0558 - accuracy: 0.5824
loss: 0.0557 - accuracy: 0.5809
loss: 0.0609 - accuracy: 0.5395
loss: 0.0585 - accuracy: 0.5571
loss: 0.0550 - accuracy: 0.5899
loss: 0.0568 - accuracy: 0.5795
loss: 0.0565 - accuracy: 0.5757
```

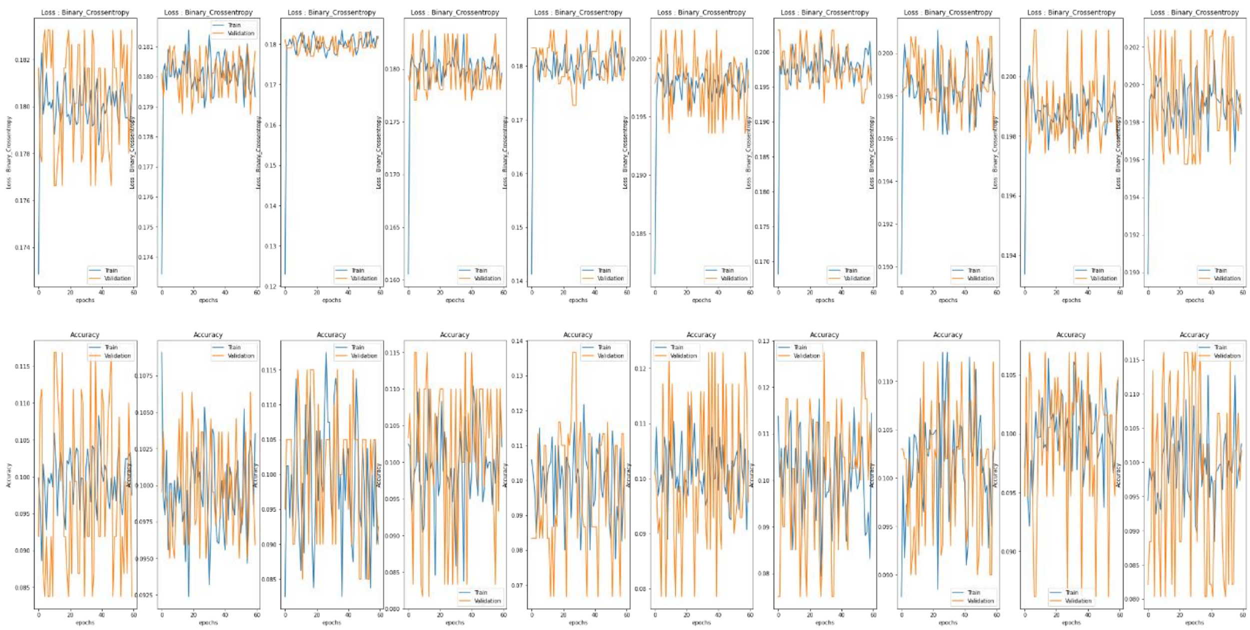
نتایج گفته ها را تایید کرد .

تغییر learning rate به 1/n: با اینکه کار غیر معمولیست ولی برای دریافت نتیجه انجام شده است



وضعیت یادگیری را به شدت ضعیف کرد.

تغییر learning rate به 1/sqrt(n):



به کلی یادگیری رت مختل کرد.

تغییر learning rate از 0.00101 به 0.05: