**مسئله ۱ –**

**سوال آ)** ماتریس ورودی(X) را به شکل زیر داریم(بایاس نیز اعمال شده است):

$$\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$

چون ورودی ها d بعدی هستند پس ماتریس وزن برابر است با:

$$\begin{bmatrix} w^{(0)} \\ \vdots \\ w^{(d)} \end{bmatrix}$$

تابع هزینه بر اساس وزن ها برابر است با:

$$J(w) = \|y - Xw\|^2 \xrightarrow{\text{گرادیان تابع هزینه}} \nabla_w J(w) = -2X^T(y - Xw) \rightarrow X^T X w^* = X^T y$$
$$\rightarrow w^* = (X^T X)^{-1} X^T y$$

**سوال ب)** مشکلات زیادی میتواند اتفاق بیفتند از جمله: در این رابطه نیازمند هزینه محاسباتی بالا و تشکیل ماتریس بزرگ طبق اندازه ورودی است که خب راه حل آن استفاده از روابط دیگر با کاربرد یکسان است – آخر رابطه بالا، فرض کردیم ماتریس $X^T X$ معکوس پذیر است و از روابط جبر خطی برای بدست آوردن وزن استفاده کردیم. اما ممکن است این ماتریس معکوس پذیر نباشد که خب راه حل آن اضافه کردن یه ماتریس همانی با مقدار کوچک است – مشکل بعدی این است که برای تشکیل ماتریس X نیاز به تمام ورودی ها به صورت یکجا داریم که ممکن است بنا به مسئله در دسترس نباشد که در این حالت باید از گرادیان تصادفی استفاده کنیم.

**سوال ج)** طبق قسمت آ برای تابع هزینه داریم:

$$J(w) = \|y - Xw\|^2 + \|w\|^2 \rightarrow w = (X^T X + \lambda I)^{-1} X^T y$$

**سوال د)** اگر ماتریس F را یک ماتریس قطری که هر درایه آن $\sqrt{f_i}$ تعریف کنیم، تابع هزینه برابر است با:

$$J(w) = \|F(y - Xw)\|^2 \rightarrow \nabla_w J(w) = 0 = -2X^T F^T F(y - Xw) \rightarrow X^T F^T Fy = X^T F^T FXw$$
$$\rightarrow w = (X^T F^T FX)^{-1} X^T F^T Fy = \left(X^T \hat{F} X\right)^{-1} X^T \hat{F} y$$

در آخرین مساوی از این فرض استفاده کردیم که ماتریس $\hat{F}$ یا همان $F^T F$ ماتریس قطری با درایه های $f_i$ است.

**سوال ه)** گرادیان $\widehat{w}$ برابر است با :

$$\nabla_w \widehat{w} = \nabla_w\left(E_{x,y}\left[(y - w^T X)^2\right]\right) = E_{x,y}\left[\nabla_w[(y - w^T X)^2]\right]$$
$$= E_{x,y}\left[-2X(y - X^T \widehat{w})\right] = 0 \rightarrow E_{x,y}(Xy) - E_{x,y}(XX^T)\widehat{w} = 0 \rightarrow \widehat{w} = \frac{C}{R}$$

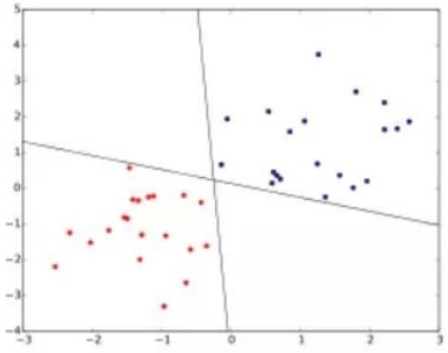
حال با توجه به رابطه مفروض در سوال داریم:

$$E_{x,y}[(y - \widehat{w}^T x)^2] = E_{x,y}\left[\left(y + w^{*T} x - w^{*T} x - \widehat{w}^T x\right)^2\right]$$
$$= E_{x,y}\left[\left(y - w^{*T} x\right)^2\right] + E_x\left[\left(w^{*T} x - \widehat{w}^T x\right)^2\right] + 2E_{x,y}\left[\left(y - w^{*T} x\right)\left(w^{*T} x - \widehat{w}^T x\right)\right]$$
$$= \underbrace{E_{x,y}\left[\left(y - w^{*T} x\right)^2\right]}_{\text{خطای ساختاری}} + \underbrace{E_x\left[\left(w^{*T} x - \widehat{w}^T x\right)^2\right]}_{\text{خطای تقریب}}$$

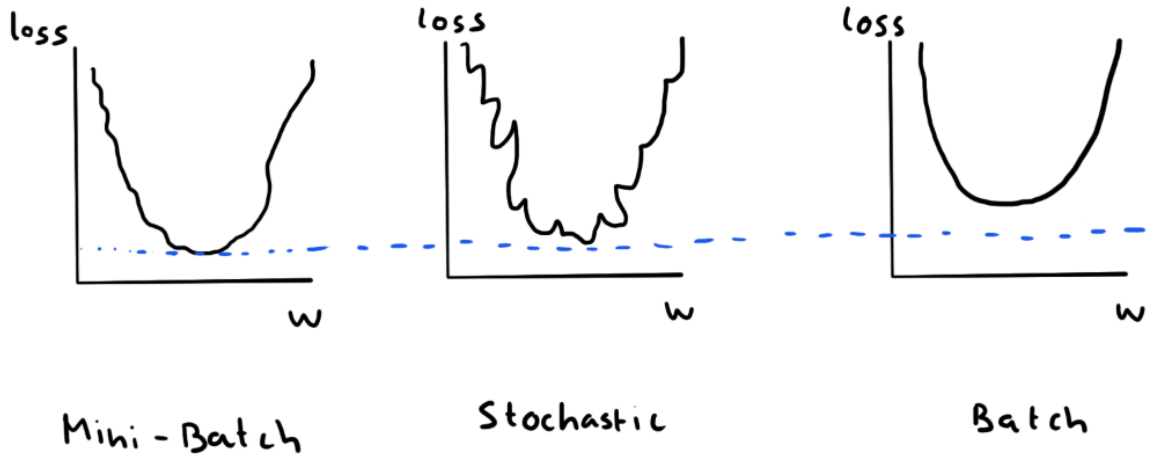
---

**مسئله ۲ –**

**سوال الف)** چون که داده های آموزشی تک تک به شبکه داده میشوند، بردار وزن میتواند در ترتیب مختلف داده ها متفاوت همگرا شود زیرا الگوریتم یادگیری وقتی شبکه دیگر خطا نکند، یادگیری را متوقف میکند، لذا بردار های وزن مختلفی میتوانند این شرایط را مهیا کنند. برای مثال اگر داده های آموزشی زیر را به شبکه بدهیم، میتوانیم بردار های وزن مختلفی داشته باشیم که به وسیله آن ها نقطه های قرمز و آبی را از هم جدا کنیم.



**سوال ب)** در ابتدا انواع یادگیری پرسپترون که در اصل انواع gradient descent محسوب میشوند را در نظر گرفته و تابع هزینه آن را نمایش میدهیم:




با توجه به نمودار های بالا، در مقایسه stochastic GD و batch GD به نکات زیر اشاره میکنیم:

۱- اگر مجموعه داده بزرگی داشته باشیم، بخاطر اینکه SGD با سرعت بیشتری(با فاصله زمانی کمتر) پارامتر هارا آپدیت

۲- میکند، لذا همگرایی سریع تر است اما در حالت کلی، همگرایی BGD سریع تر است چون مستقیم به سمت کمینه حرکت میکند.

۳- بخاطر حالت زیگزاگی که SGD دارد، در مواجه با کمینه های محلی، میتواند از آن ها برای فرار از کمینه های محلی استفاده کند اما BGD بخاطر حرکت مستقیمی که دارد، ممکن است بیشتر در کمینه محلی گرفتار شود.

۴- بخاطر تعدد آپدیت پارامتر توسط SGD ، حرکت نویزی ای(زیگزاگی) داریم و زمان را تلف میکند.

۵- آپدیت پارامتر ها توسط BGD بسیار پایدار تر است و آسان تر به سمت کمینه سراسری حرکت میکند.

**سوال ج)** اگر تعداد بروزرسانی های الگوریتم پرسپترون را k در نظر بگیریم، فرض میکنیم که برای یک عدد ثابت N داریم $Nk \leq \|w_{(k+1)}\|$ . حال با توجه به فرض هایی که در صورت سوال گفته شده:

$$w_{(k+1)} \cdot w^* = \left( w_{(k)} + y^{(i)}x^{(i)} \right)w^* = w_{(k)} \cdot w^* + y^{(i)}\left(w^* \cdot x^{(i)}\right) \geq w_{(k)} \cdot w^* + \rho$$
$$\rightarrow w_{(k+1)} \cdot w^* \geq k\rho$$

همچنین:

$$\|w_{(k+1)}\| \times \|w^*\| \geq w_{(k+1)} \cdot w^*$$

لذا با توجه به نامساوی های بالا نتیجه میشود:

$$\|w_{(k+1)}\| \times \|w^*\| \geq k\rho$$

حال فرض میکنیم برای یک عدد ثابت دیگر به نام $M$ داریم $\|w_{(k+1)}\| \leq M\sqrt{k}$ . حال با توجه به دانش قبلی و فرض سوال داریم:

$$\|w_{(k+1)}\|^2 = \|w_{(k)} + y^{(i)}x^{(i)}\|^2 = \|w_{(k)}\|^2 + (y^{(i)})^2\|x^{(i)}\|^2 + 2y^{(i)}\left(w_{(k)} \cdot x^{(i)}\right)$$
$$\leq \|w_{(k)}\|^2 + r^2$$

نامساوی بالا اتفاق افتاد چون که قسمت هایلایت شده در k امین بروزرسانی، عددی صفر یا منفی است. حال نتیجه میشود که :

$$\|w_{(k+1)}\|^2 \leq k\,r^2 \rightarrow \|w_{(k+1)}\| \leq \sqrt{k}\,r$$

حال با ترکیب دو نامساوی نتیجه شده داریم:

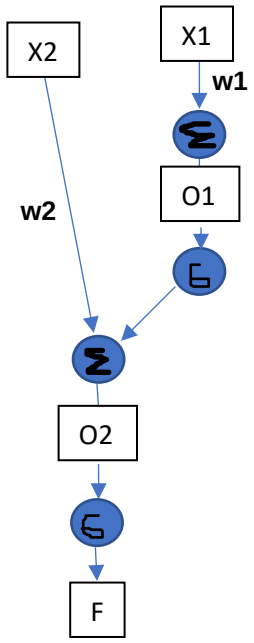
$$k\rho \leq \left\|w_{(k+1)}\right\|^2 \|w^*\| \leq \sqrt{k}\, r \,\|w^*\| \rightarrow k\rho \leq \sqrt{k}\, r$$

$$\rightarrow k^2\rho^2 \leq kr^2\|w^*\|^2 \rightarrow k \leq \left(\frac{r\,\|w^*\|}{\rho}\right)^2$$

---

**مسئله ۳ –**

**سوال الف)** در قسمت اول این سوال، ابتدا طبق رابطه شبکه عصبی، معماری آن را رسم میکنیم:




$$\frac{\partial f}{\partial w2} = \frac{\partial f}{\partial O2} * \frac{\partial O2}{\partial w2} \overset{*}{\Rightarrow} \frac{1}{4} * \frac{1}{2} = \frac{1}{8}$$

$$\overset{*}{\rightarrow} \frac{\partial f}{\partial O2} = \sigma(O2)\big(1 - \sigma(O2)\big) = \frac{1}{4}$$

$$\overset{*}{\rightarrow} \frac{\partial O2}{\partial w2} = \sigma(O1) = \frac{1}{2}$$

**سوال ب)**

$$\mathcal{L} = (y - f)^2 , \frac{\partial \mathcal{L}}{\partial w2} = \frac{\partial \mathcal{L}}{\partial f} * \frac{\partial f}{\partial w2} = -2(y - f) * \frac{\partial f}{\partial w2} = -2 * \left(5 - \frac{1}{2}\right) * \frac{1}{8} = -\frac{9}{8}$$

$$w2 = w2 - \alpha \frac{\partial \mathcal{L}}{\partial w2} = 0 - \frac{1}{2} * -\frac{9}{8} = \frac{9}{18}$$

---

**مسئله ۴ –**

**سوال ۱ – الف)** با فرض اینکه بردار ورودی لایه $z_i = \{z_1 , z_2 , \dots , z_n\}$ باشد لذا بردار خروجی میشود:

$$\tanh(z_i) = \{y_1 , y_2 , \dots , y_n\}$$

ماتریس ژاکوبین مشتق جزئی خروجی به ورودی را نشان میدهد که در این مسئله سایز اش n*n است و به شکل زیر است:

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_n}{\partial z_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial z_n} & \cdots & \frac{\partial y_n}{\partial z_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial y_2}{\partial z_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial y_n}{\partial z_n} \end{bmatrix}$$

چون خروجی $y_i$ به ورودی $z_j$ به شرطی که $i \neq j$ باشد، مستقل است، لذا مشتق جزئی آن ها صفر میشود و ماتریس ژاکوبین یک ماتریس sparse قطری میشود. پس داریم:

$$\frac{\partial y}{\partial z} = \frac{\partial \frac{e^z - e^{-z}}{e^z + e^{-z}}}{\partial z} = \frac{\left(e^z - (-e^{-z})\right)(e^z + e^{-z}) - \left(e^z + (-e^{-z})\right)(e^z - e^{-z})}{(e^z + e^{-z})^2}$$
$$= 1 - (e^z - e^{-z})^2 = 1 - (tanh)^2$$

لذا ماتریس ژاکوبین $tanh$ برابر میشود با:

$$\begin{bmatrix} 1-(tanh)^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1-(tanh)^2 \end{bmatrix}$$

**سوال ۱ – ب)** برای توضیح این سوال، ابتدا تعریف ریاضی اشباع یک تابع فعالساز را بررسی میکنیم. میگوییم یک تابع فعالساز غیر اشباع است اگر و تنها اگر:

$$f\ is\ non-saturating\ iff\ \left(\left|\lim_{z\to-\infty} f(z)\right| = +\infty\right) \bigvee \left(\left|\lim_{z\to+\infty} f(z)\right| = +\infty\right)$$

و بالطبع، اگر تابعی غیر اشباع کننده نباشد، اشباع کننده است. تابع ***sigmoid*** اشباع کننده است چون همه اعداد حقیقی را به بازه ***[0,1]*** محدود میکند. تابع ***tanh*** هم اشباع کننده است زیرا کل بازه اعداد حقیقی را به ***[-1,1]*** محدود میکند. اشباع کردن هم چیز خوبی نیست زیرا اعداد کوچک و بزرگ تفاوت چندانی پس از گذشتن از تابع فعالساز نمیکنند. پس از نظر اشباع، ***tanh*** مزیتی نسبت به ***sigmoid*** ندارد.

**سوال ۱ – ج)** مزیت ***tanh*** بر ***sigmoid*** اولا این است که ***tanh*** حول نقطه صفر مرکزیت دارد یعنی ***zero centered*** و متقارن نسبت به نقطه صفر است اما ***sigmoid*** حول نقطه ***0.5*** مرکزیت دارد. این ***zero centered*** بودن سرعت و آسان بودن یادگیری لایه بعدی را آسان تر میکند زیرا همگرایی افزایش می یابد.

از طرف دیگر، مزیت دیگری که ***tanh*** بر ***sigmoid*** دارد، این است که شیب گرادیان ***tanh*** بزرگ تر از ***sigmoid*** است لذا در مواقعی که به سرعت میخواهیم تابع هزینه را کمینه کنیم، این شیب(مشتق) به درد میخورد.

**سوال ۲ –** در یادگیری ماشین، وقتی که بخواهیم یک شبکه عصبی مصنوعی را با روش های مبتنی بر گرادیان و انتشار رو به عقب، آموزش دهیم، ممکن است با مشکل محو شدن گرادیان روبرو شویم. مشکل از جایی شروع میشود که گرادیان تابع زیان، به سمت صفر میل میکند و این باعث میشود که وزن ها به خوبی تغییر نکنند(اطلاعات خوبی از گرادیان به عقب برگردانده نمیشود) و آموزش لایه ها را کند و در بدترین حالت کاملا متوقف کند. دلیل این امر هم توابع فعالساز مثل ***sigmoid*** هستند

که بخاطر مشکل اشباعی که دارند، تغییرات زیاد در ورودی باعث تغییرات اندک در خروجی میشوند. لذا مشتق کوچک شده و به سمت صفر میل میکند و این مشکل پدید می آید.

راه حل های زیادی برای این مشکل وجود دارد: ۱- استفاده از تابع فعالساز دیگری که مشکل اشباع را نداشته باشند مانند ***RELU*** ۲- استفاده از شبکه های ***residual*** ۳- استفاده از ***batch normalization*** و نرمال کردن ورودی

**سوال ۳ –** در رابطه تابع ***softmax*** که در زیر به آن اشاره شده است، اگر مقدار ورودی ها بسیار بزرگ باشند، مقدار ***exp(x)*** سرریز میکند و کل رابطه تعریف نشده میشود.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

راه حل این است که تمامی درایه های بردار ورودی را منهای ماکسیمم این بردار کنیم یعنی: $z_i = x_i - \max\{X\}$

با اینکار بزرگ ترین ورودی تبدیل به صفر و مابقی ورودی ها منفی میشوند. از مفاهیم جبر میدانیم که وقتی از یک بردار یک مقدار ثابت را کم کنیم، مقدار تابع ***softmax*** از لحاظ تحلیلی تغییری نمیکند و مشکلی ایجاد نمیشود.

---

**مسئله ۵ –**

**سوال ۱ )** میدانیم که یکی از مشکلاتی که افزایش ضریب یادگیری به همراه خود می آورد، مشکل در بزرگی مقدار گرادیان است زیرا هر چه لایه هارا به سمت خروجی جلو میرویم، شیب گرادیان متوالیا ضرب میشود و اگر مقدار بزرگی آن بسیار کوچک باشد، مشکل محو شدن گرادیان پیش می آید و اگر مقدار بزرگی داشته باشیم، مشکل انفجار گرادیان رخ میدهد. پس یک جوری باید مقدار گرادیان یا همان شیب هارا کنترل کنیم که روش ***batch normalization*** همین کار را میکند و مقدار خروجی ***activation function*** هارا در محدوده خاصی نرمال میکند پس دستمان باز میشود که ضریب یادگیری را افزایش دهیم.

**سوال ۲ – الف )** رگولاریزاسیون با اضافه کردن یک پنالتی به ***cost function*** از ***overfit*** شدن مدل جلوگیری میکند. این باعث میشود که وزن های ***feature*** های وابسته بروز نشوند و یادگیری تکراری اتفاق نیفتد. حال ***dropout*** با هرس کردن شبکه عصبی در حین آموزش، نیز باعث جلوگیری از یادگیری تکراری و در نتیجه جلوگیری از ***overfit*** شدن مدل میشود.

**سوال ۲ – ب** ) تفاوتی که روش *ensemble learning* و *multi-task learning* است این است که در *ensemble learning* برای یادگیری یک شبکه واحد از چند الگوریتم یادگیری استفاده میکنیم اما در *multi-task learning* چند شبکه مختلف را آموزش میدهیم. حال در رابطه با *ensemble learning* صحبت میکنیم. در این روش، یک شبکه با چند الگوریتم آموزش داده میشود و سپس توسط رویکرد *voting* یا میانگین گیری یا هر روش تلفیقی مشابه، خروجی نهایی ایجاد میشود. در *dropout* می آییم برخی نورون های شبکه را هرس میکنیم یعنی یه جورایی داریم شبکه های مختلفی را روی داده هایمان اعمال میکنیم که تقریبا ایده آن شبیه روش *ensemble learning* است که شبکه های مختلفی را داریم(یک شبکه با آموزش های مختلف) و آن هارا در آخر تجمیع میکنیم.

**سوال ۲ – ج** ) در روال آموزش، در هر *iteration* به ازای هر نمونه آموزشی، در یک لایه مخفی $p$ درصد نورون هارا غیرفعال میکنیم و از یادگیری آن ها جلوگیری میشود. در روال تست، نورونی حذف نمیشود بلکه با یک اندازه ای خروجی آن(*activation* آن) کاهش پیدا میکند.

**سوال ۳ –** اصطلاح *multi-task learning* یعنی آموزش همزمان چندین مدل و انتقال دانش بین آن ها برای یادگیری بهتر همه مدل ها. یکی از متد های این نوع یادگیری ، *parameter sharing* است که دو حالت میتواند اتفاق بیفتد: ۱- اشتراک واقعی لایه های ابتدایی(وزن) مدل ها و ۲- گذاشتن محدودیت و قید روی وزن های ابتدایی مدل ها.

این باعث میشود مدل ها مثل هم عمل کنند و عملا برای آپدیت های لایه های ابتدایی به هم کمک کنند و به عبارتی انتقال دانش انجام دهند.این مثل هم عمل کردنی که گفته شد، در اصل همان *generalization* و عام بودن مدل را بیان میکند و به عبارتی انتقال دانش و مثل هم کار کردن، به عنوان *inductive bias* کار میکند که خود باعث جامعیت میشود.

**سوال ۴ –** ایده پشت *regularization* این است که وزن ها را محدود کنیم تا از *overfitting* یعنی حساسیت زیاد خروجی به ورودی را کنترل کنیم. از آن جایی که بایاس یک ورودی مقدار ثابت است کنترل وزن آن مهم نیست و نیازی نداریم برای آن از *regularization* استفاده کنیم.

**سوال ۵ –** در روش بیشینه-نرم ، قید *Strict* ای روی وزن میگذاریم تا در آن محدوده بماند اما روش اول ، بخاطر پنالتی ای که اضافه کرده ایم، وزن ها را ترغیب میکنیم محدود بمانند اما *strict* نیست و میتوانند هم مطیع نباشند. منظور اینکه اگر نیاز داشتیم حتما وزن محدود بماند به هر قیمتی، از روش دوم استفاده میکنیم(مثلا شبکه ای که تمایل خیلی زیادی به افزایش وزن دارد به هردلیلی). در روش دوم یک هایپرپارامتر به نام $c$ داریم که عملا ضریب تقسیم وزن بعد از فراتر رفتن از محدوده مجاز است.

**مسئله ۶ –** با توجه به ریاضیات پایه میدانیم:

**سوال ۱ )**

$$L(x2) = L(x1) + \frac{1}{2} * (x2 - x1)^T \, H \, (x2 - x1) + (x2 - x1)^T \, g$$

وقتی میخواهیم ***L(x2)*** را کمینه کنیم، پس از آن نسبت به ***x2*** مشتق جزئی میگیریم:

$$\frac{\partial L(x2)}{\partial x2} = \; g^T + \frac{1}{2}(x2 - x1)^T (H + \; H^T) = \; g^T + \; (x2 - x1)^T \, H = \; 0$$
$$\rightarrow \; g^T H^{-1} = (x1 - x2)^T \; \rightarrow \; H^{-1} \, g = x1 - x2 \; \rightarrow x2 = x1 - \; H^{-1} \, g$$

پس در ***gradient descent*** عبارت $\eta^* g = \; H^{-1} g$ که با ضرب کردن طرفین در ***H*** داریم : $\eta^* H g = g$ حال طرفین را در ترانهاده ***g*** ضرب میکنیم که داریم: $\eta^* g^T H g = g^T g$ لذا داریم:

$$\eta^* = \frac{g^T g}{g^T H g}$$

**سوال ۲ )** اولا که چون نقاط ***saddle point*** ترکیبی از ***local minima , local maxima*** است، طبیعتا از نقاط ***local optimum*** که یکی از ***local*** ها میتواند باشد، فرکانس بیشتری دارد. ثانیا گرادیان نقاط ***saddle point*** صفر است و با استفاده از ***gradient descent*** در آن گیر میکنیم و یادگیری متوقف میشود.

**سوال ۳ )** روش ***gradient descent*** بهینه سازی تابع هزینه(کمینه کردن) با حرکت بر خلاف جهت گرادیان با یک نرخ یادگیری است.مشکلات این روش ۱-انتخاب نرخ یادگیری مناسب که هم از کاهش سرعت یادگیری هم از همگرا نشدن جلوگیری شود ۲- اعمال همزمان نرخ یادگیری روی تمام پارامتر ها و منعطف نبودن آن ۳- گیر افتادن در نقاط بهینه محلی و نقاط ***saddle***

روش *momentum* مشکل یکی از مشتقات *GD* یعنی *SGD* را حل میکند و آن حرکت مستقیم به سمت شیب زیادتر و بعد بهتر است. ایده این روش، حفظ اثر جهت آپدیت قبلی در تکرار فعلی است. یعنی سرعت کم مرحله قبل منجر به گام کوچک در مرحله فعلی میشود و برعکس. در این روش احتمال فرار از بهینه محلی و *saddle* ها افزایش می یابد.

روش *NAG* به جای حرکت کورکورانه از یک روش پیشگویی استفاده میکند یعنی از نقطه بعدی برای حرکت فعلی استفاده میشود. یعنی گرادیان را بر حسب نقطه تقریبی بعدی حساب میکند. با این روش نرخ همگرایی افزایش می یابد.

روش *RMSprobe* بر کاستی های روش *adagrad* در گرادیان تجمعی غلبه میکند و آن با استفاده کردن *RMSprobe* از *moving average gradient* اِ وزن نمایی است. به جای اینکه تمام گرادیان های قبلی ذخیره شود، از یک رابطه بازگشتی برای گرادیان های قبلی استفاده میشود. همگرایی سریع تر و آسان تر میشود.

روش *Adam* از نرخ *adaptive* برای هر پارامتر استفاده میکند و بسیار جامعیت و قدرت بالایی دارد. خوبی های روش های قبلی، در این الگوریتم جمع شده. هم سرعت بالایی دارد هم از روش *adaptive* برای پارامتر ها عمل میکند. حول و حوش بهنیه محلی سرعت آن کم میشود.

---