به نام خدا

دانشگاه امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه دوم

شبکه های عصبی

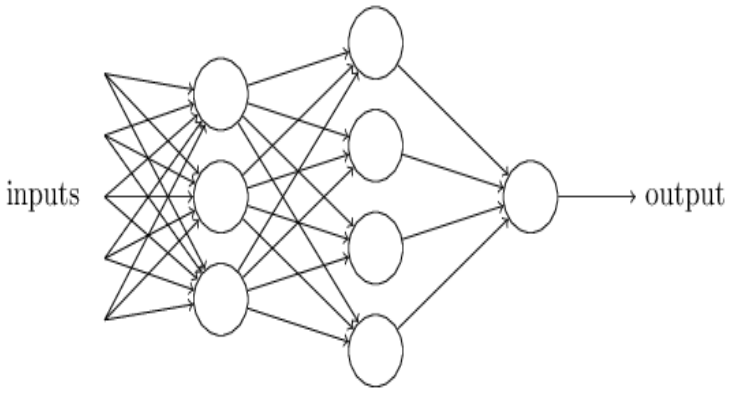
خدیجه ساعدنیا

۹۴۱۳۱۰۵۹

فهرست

تعریف MLP :

شبکه پرسپترون چند لایه حداقل شامل یک لایه مخفی می باشدپس می توان گفت که حداقل سه لایه دارد معماری آن به صورت زیر می باشد:



برای آموزش این شبکه سه مرحله اصلی وجود دارد:

۱) ابتدا شبکه به روش پیش رو باید با اعمال ورودی ها به لایه ورودی تمامی خروجی ها را بدست آور

۲) در این مرحله خروجی بدست آمده با خروجی مورد انتظار مقایسه می شود و خطا به وجود آمده محاسبه میشود

۳) در این مرحله خطا لایه آخر به روش عقبگرد در کل شبکه منتقل می شود و مقادیر وزن در صورت نیاز اصلاح می شوند.

الگوریتم انجام این آموزش به صورت زیر می باشد:

۰- برای وزن ها مقادیر اولیه رندم و کوچک انتخاب می کنیم.

۱- تا زماتی که به شرط پایان نرسیده ایم مراحل ۲ تا ۷ را تکرار می کنیم

۲- برای هر نمونه ی آموزشی که به صورت (S_p , T_p) می باشد، مراحل ۳ تا ۶ را تکرار می کنیم:

۳- خروجی تمام لایه ها را از لایه دوم تا آخر با استفاده از روابط زیر محاسبه می کنیم:

$$y_{pj}^{(1)} = f_j(\sum_{i=0}^{n} w_{ij} s_{pi})$$

$$y_{pj}^{(l)} = f_j\left(\sum_{i=0}^{n} {w_{ij}}^{l} {y_{pi}}^{l-1}\right), l > 1$$

۴- مقدار دلتا را برای لایه خروجی با استفاده از رابطه زیر محاسبه می کنیم:

$$\delta_{pj} = (t_{pj} - y_{pj}) f_j'(I_{pj})$$

۵- مقدار دلتا را برای لایه های مخفی از رابطه زیر محاسبه می کنیم:

$$\delta_{pj}{}^{(l)} = \left(\sum_k \delta_{pk}{}^{(l+1)} w_{jk}^{(l+1)} \right) f_j'(I_{pj}{}^{(l)})$$

۶- مقادیر وزن ها را به روز رسانی می کنیم:

$$w_{ij}^{new(l)} = w_{ij}^{old(l)} + \eta \delta_{pj}{}^{(l)} y_{pj}^{(l-1)}$$

۷- شرط پایان را تست می کنیم.

روش پیاده سازی پروژه:

در این پروژه مجموعه داده ها در یک فایل csv می باشد که شامل ۳۸۵ ویژگی هستند و باید ویژگی ۳۸۶ ام تخمین زده شود. در فایل readInput پس از خواندن داده ها در یک آرایه قرار می گیرند. با توجه به اینکه نیاز است در آزمایشی داده ها مقیاس گذاری شوند، در ورودی تابع readData() دو مقدار گرفته می شود، در صورتی که آرگومان اول مقدارش برابر True شود، داده ها مقیاس گذاری می شوند با استفاده از رابطه زیر:

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

تابع feedForward() برای محاسبه خروجی هر لایه استفاده می شود.که با گرفتن هر کدام از ورودی ها با استفاده از مقدار بایاس و وزن مربوطه و همچنین تابع sigmoid خروجی را محاسبه می کند.

کار آموزش ما با تابع SGD که از روش کاهش گرادیان استفاده می کند تکمیل می شود.

در این تابع ابتدا پارامترهای مورد نیاز را دریافت می کنیم.Training _set شامل زوجهای ورودی و خروجی مورد انتظار می باشد. تعداد ایپک های آموزشی توسط epochs مشخص می شود و برای سرعت بخشیدن به آموزش از روش batch استفاده شده، به این صورت که داده ها را به دسته هایی با اندازه مشخص که توسط mini_batch_size مشخص شده است تقسیم کرده و هر بار الگوریتم بر روی یکی از دسته ها تغییر ایجاد می کند به جای آنکه بر روی هر داده تک تک اعمال شود.

در صورتی که اندازه دسته یک داده شود، داده ها بدون دسته بتدی استفاده می شوند.

در هر ایپک آموزشی ابتدا ترتیب قرارگیری داده ها را تغییر می دهیم و سپس آنها را دسته بندی می کنیم تا توزیع مناسبی در هر یک از دسته ها داشته باشیم.

حال به ازای تک تک داده ها در هر یک از دسته ها مقادیر را به روز رسانی می کنیم، که این کار توسط تابع update_mini_batches() انجام می شود.

در این تابع از روش backpropagation استفاده شده است، همانطور که در بخش قبلی توضیح داده شد، در تابع backprop() ابتدا به روش پیشرو خروجی را محاسبه می کنیم و سپس با استفاده از روابطی که بیان شد خطا را محاسبه می کنیم و در نهایت به میزانی که باید وزن ها و بایاس ها تغییر کند در تابع update_mini_batches() روی آنها تغییر ایجاد می شود.

برای پیاده سازی ممنتم نیاز است که وزنها تا دو مرحله نگهداری شود که این کار با استفاده از آرایه weight_history انجام می شود.

برای پیاده سازی قاعده ویدرو، از روابط زیر استفاده می شود، که در کد پیاده سازی شده است:

$$s = \sqrt[n]{p}$$

$$w_{ij} = s\, w_{ij}(old) / \| W_j(old) \|$$

همانند پروژه قبلی شرط هایی که برای پایان آموزش در نظر گرفته شده است، به صورت زیر می باشد:

۱- رسیدن به خطایی که در واسط کاربری تعیین شده است.
۲- رخداد OverFlow
۳- رسیدن به تعداد مشخصی ایپک آموزشی
۴- برای اینکه آموزش بیهوده پیش نرود شرطی در برنامه قرار دارد که در صورتی که در ۱۰ ایپک آموزشی نتیجه بهتری حاصل نشود آموزش خاتمه یابد.

نمایش داده ها و واسط کاربری:

برای سرعت بخشیدن به اجرای برنامه و همچنین راحت تر کار کردن با برنامه، برنامه در دو پروسس به صورت موازی اجرا می شود که در یکی از آنها به آموزش و محاسبه خطا ها می پردازد و در دیگری به رسم نمودار خطاها و نمودار تغییرات وزن می پردازد.

پارامترهای خواسته شده در واسط کاربری می توانند همگی تنظیم شوند و با زدن دکمه train آموزش آغاز میشود. همچنین امکان ذخیره سازی و بازیابی یک شبکه هم در نظر گرفته شده است که در صورتی که آدرس وارد شده برای وزنها صحت داشته باشد، شبکه از روی آن لود می شود و در غیر این صورت شروع به آموزش شبکه با پارامترهای تنظیم شده می کند.

در واسط کاربری امکان تنظیم تمامی پارامترها و شرایط آزمایش در نظر گرفته شده است.

نتایج تست های انجام شده روی شبکه:

در ابتدا یک آزمایش پایه صورت گرفته است که شرایط و نتایج آن در زیر آمده است، سپس در هر مرحله به تاثیر عوامل مختلف بر روی شبکه پرداخته شده است.

با توجه به اینکه تعداد نورون ها در لایه ورودی ۳۸۵ می باشد، نمودار تغییرات وزن به شدت متراکم می شود، و اطلاعات مفیدی نمی توان از آن بدست آورد، به همین خاطر از آوردن نمودارها در گزارش خودداری شده است، اما در زمان اجرا برنامه می توان آن را مشاهده کرد.

در تمامی نمودارها خط آبی نشان دهنده خطای آموزشی و خط سبز نشان دهنده خطای validation می باشد.

آزمایش پایه:

مشخصات شبکه

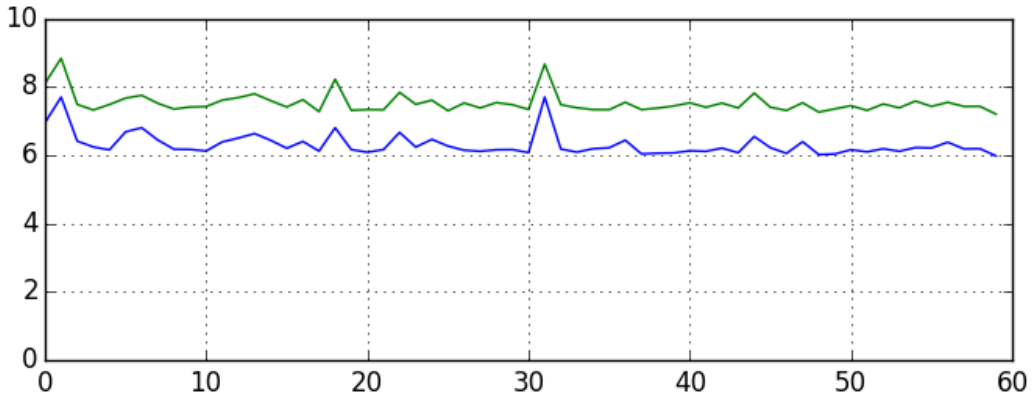
معماری شبکه: $784, 30, 10$

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: خطی

نحوه اتمام آموزش: رسیدن به تعداد ایپک مشخص شده



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار5.9 می باشد و کمترین خطا برای validation مقدار 7.2 می باشد.

بررسی اثر تابع فعالیت:

آزمایش اول:

مشخصات شبکه

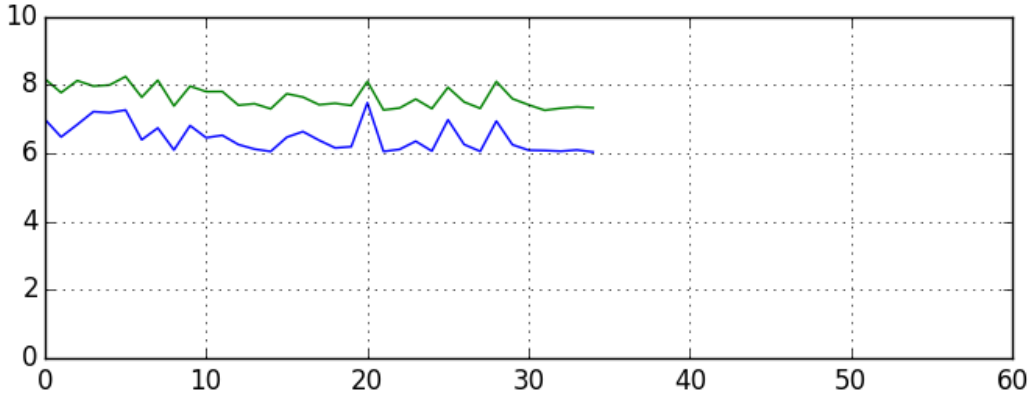
معماری شبکه: 784, 50, 10

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

نحوه اتمام آموزش: رخداد OverFiting



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار6.1 می باشد و کمترین خطا برای validation مقدار 7.3 مـی باشد.

آزمایش دوم:

مشخصات شبکه

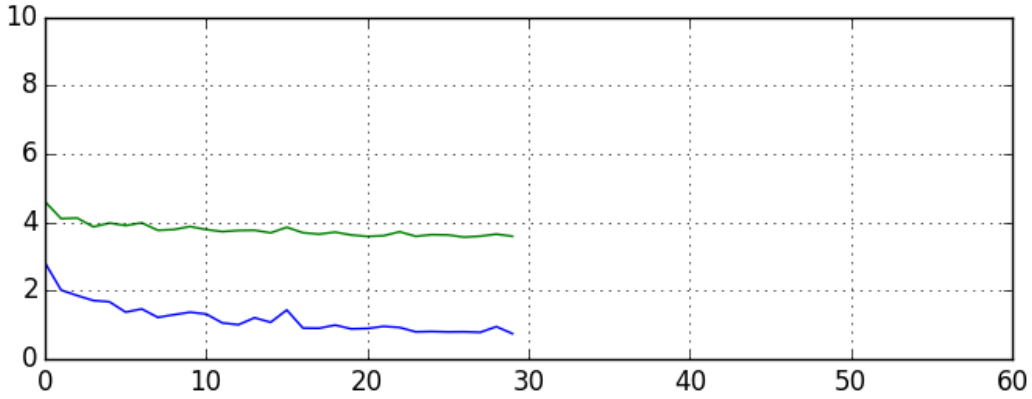
معماری شبکه: 784, 50, 10

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Piece Wise Linear

نحوه اتمام آموزش: OverFiting



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار0.74 می باشد و کمترین خطـا بـرای validation مقـدار 3.5 می باشد.

آزمایش سوم:

مشخصات شبکه

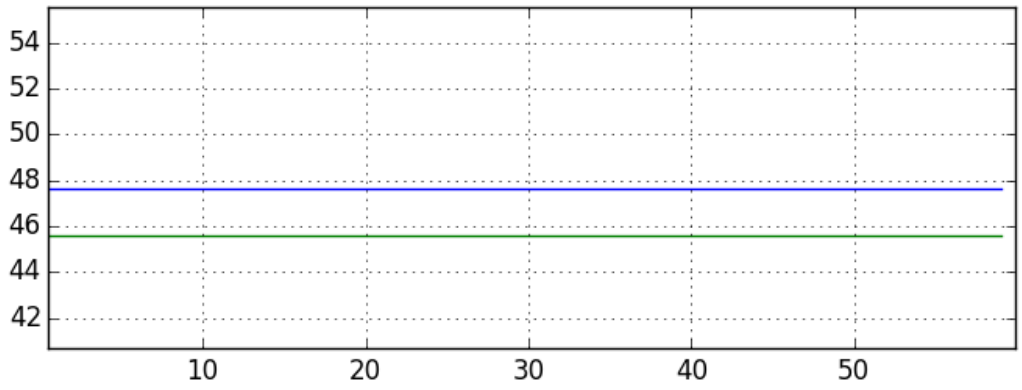
معماری شبکه: 784, 50, 10

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Logarithm

نحوه اتمام آموزش: رسیدن به تعداد ایپک مشخص شده



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار 47.6 می باشد و کمترین خطا برای validation مقدار 45.6 می باشد.

نتیجه آزمایش:

همانطور که نتایج آزمایش ها نشان می دهد، تغییر تابع فعالیت تاثیر زیادی در سرعت و دقت آموزش دارد. با استفاده از تابع لگاریتم سرعت آموزش به شدت کاهش می یابد و خطا نیز بسیار زیاد می شود.

با استفاده از تابع خطی می بینیم که سرعت آموزش افزایش می سابد و خطا نیز کاهش می یابد.

بهترین تابع فعالیتی که در این پروژه آزمایش شده تابع خطی تکه ای می باشد، که هم از نظر سرعت و هم از نظر خطا بهترین می باشد.

بررسی اثر پس انتشار خطا دسته ای(Batch):

آزمایش اول:

مشخصات شبکه

معماری شبکه: 784, 50, 10

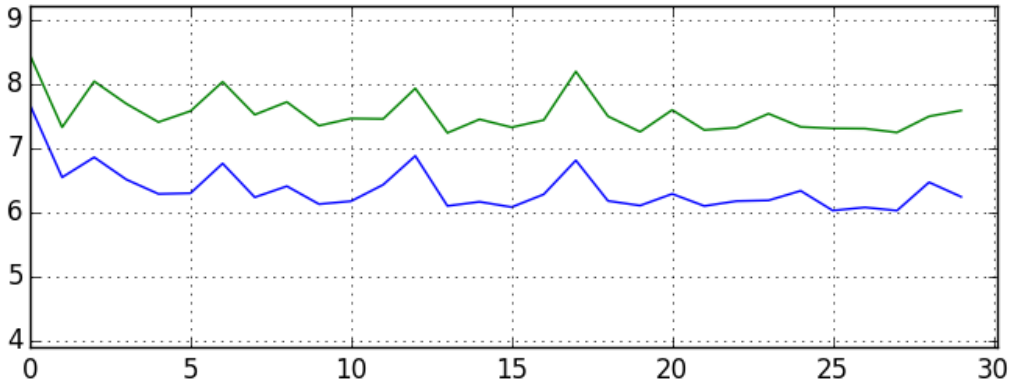
تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

اندازه دسته ها: 100

نحوه اتمام آموزش: OverFitting



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار6.2 می باشد و کمترین خطا برای validation مقدار 7.6 می باشد.

نتیجه آزمایش:

همانطور که از آزمایش مشخص می شود استفاده از انتشار خطا باعث می شود سرعت افزایش یابد، اما خطا بهبود کمتری پیدا می کند و آموزش با خطای بیشتری خاتمه می یابد.

بررسی اثر مقادیر اولیه وزن ها و روش پیشنهادی ویدرو:

آزمایش اول:

مشخصات شبکه

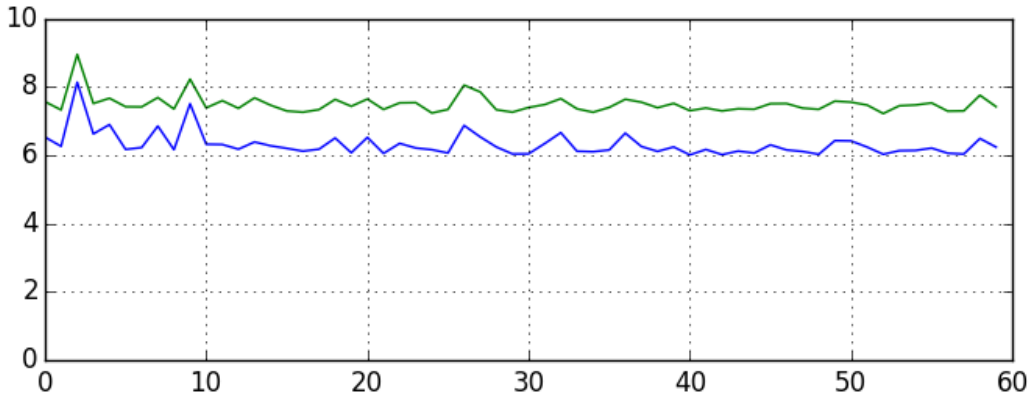
معماری شبکه: $784, 50, 10$

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

نحوه اتمام آموزش: رسیدن به تعداد ایپک مشخص شده



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار6.2 می باشد و کمترین خطا برای validation مقدار 7.4 مـی باشد.

نتیجه آزمایش:

با استفاده از روش ویدرو به دلیل اینکه وزن های اولیه توزیع مناسبی دارند، خطای شبکه از ابتدا مقدار کمی دارد و این مسئله باعث می شود که سرعت همگرا شدن خروجی افزایش یابد.

بررسی اثر پس انتشار خطا با استفاده از ممنتم:

آزمایش اول:

مشخصات شبکه

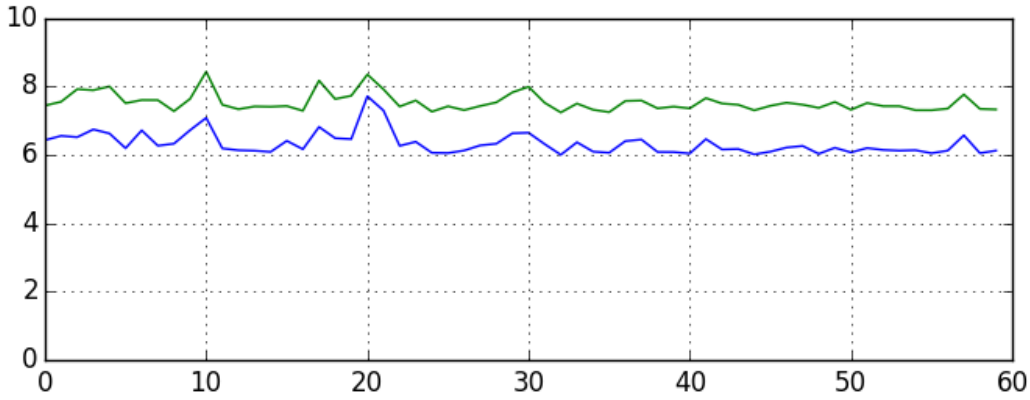
معماری شبکه: $784, 50, 10$

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

نحوه اتمام آموزش: رسیدن به تعداد ایپک مشخص شده



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار6.05 می باشد و کمترین خطـا بـرای validation مقـدار 7.3 می باشد.

نتیجه آزمایش:

با استفاده از روش ممنتم با توجه به اینکه در هر مرحله آپدیت وزنها در جهت کاهش خطا صورت می گیرد، سرعت آموزش افزایش می یابد و خطای نهایی زیر بهبود می یابد.

بررسی اثر مقیاس گذاری:

آزمایش اول:

مشخصات شبکه

معماری شبکه: 784, 50, 10

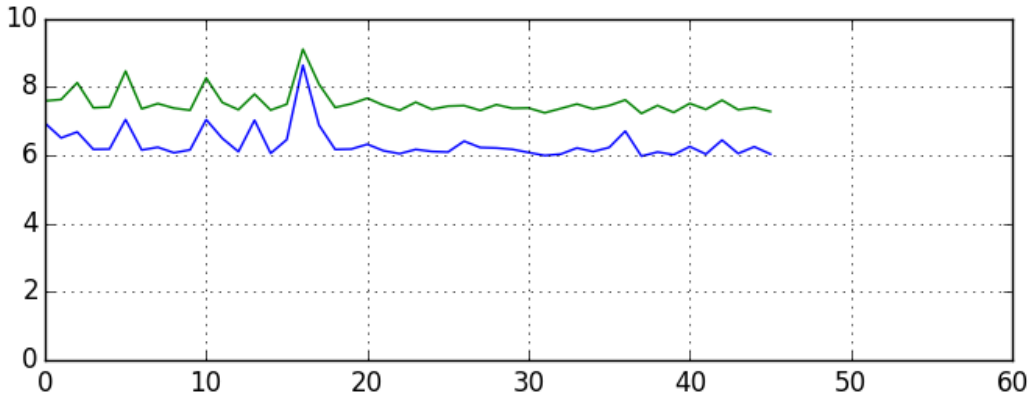
تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

مقیاس ورودی: داده ها همگی بین 0 و1 هستند.

نحوه اتمام آموزش: OverFiting



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار6.03 می باشد و کمترین خطا برای validation مقدار 7.2 می باشد.

آزمایش دوم:

مشخصات شبکه

معماری شبکه: 784, 50, 10

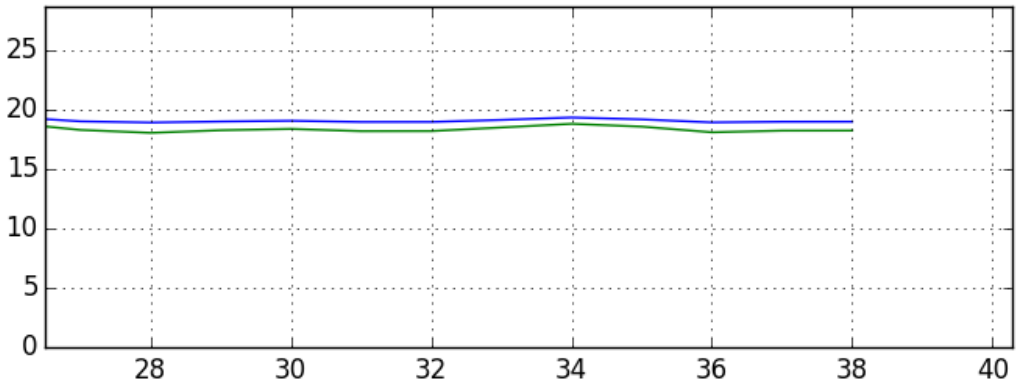
تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

مقیاس ورودی: تمامی داده ها پس از نرمال شدن در مقدار 0.0001 ضرب شده اند.

نحوه اتمام آموزش: OverFitting



نمودار تغییرات خطا بر حسب شماره ایپک

در این آزمایش بهترین خطا برای آموزش مقدار18.9 می باشد و کمترین خطا برای validation مقـدار 18.2 می باشد.

آزمایش سوم:

مشخصات شبکه

معماری شبکه: 784, 50, 10

تعداد ایپک های آموزشی: 60

نرخ آموزش: 0.003

تابع فعالیت: Linear

مقیاس ورودی: تمامی داده ها پس از نرمال شدن در مقدار 100 ضرب شده اند.

نحوه اتمام آموزش: ---------

در این شبکه به علت بزرگ بودن مقادیر ورودی شبکه توانایی آموزش نداشته، داده ها بزرگتر از تایپی بوده اند که در برنامه استفاده شده، در صورتی که نوع داده ها بزرگتر در نظر گرفته می شد رم کامپیوتر توان اجرای برنامه را نداشت.

نتیجه آزمایش:

با استفاده از مقیاس گذاری داده ها در آزمایش اول به علت اینکه داده ها نرمال شده اند، تاثیر آنها نسبتا برابر است و باعث می شود که خطا خروجی کاهش یابد، اما در آزمایش دوم به علت ضریب کوچکی که در آنها ضرب شده است، اثر آنها به شدت کاهش می یابد و سرعت آموزش نیز کاهش می یابد.

مراجع

[1] Fundamentals Of Neural Networks by Laurene Fausett
[2] Neural_Networks_-_A_Comprehensive_Foundation_-_Simon_Haykin