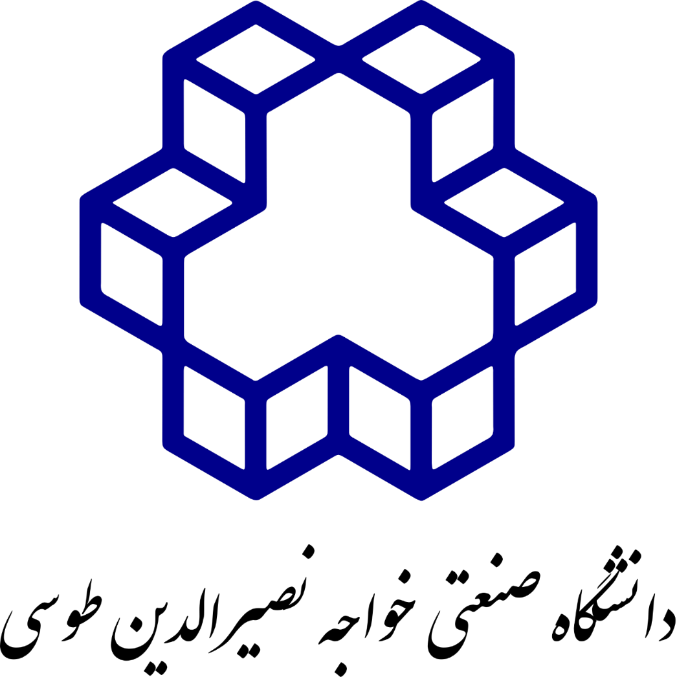
به نام خدای رنگین کمان



|  |  |
| --- | --- |
| **نام و نام خانوادگی:** | سجاد فودازی |
| **شماره دانشجویی:** | 40007903 |
| **استاد:** | دکتر مهدی علیاری |
| **درس:** | مبانی سیستم های هوشمند |
| **موضوع:** | گزارشکار مینی پروژه شماره 2 |

فهرست مطالب

[فهرست تصاویر 3](#_Toc187011606)

[فهرست جداول 4](#_Toc187011607)

[پرسش یک 5](#_Toc187011608)

[سوال 1.1 5](#_Toc187011609)

[سوال 1.2 6](#_Toc187011610)

[سوال 1.3 7](#_Toc187011611)

[کد پرسش یک 8](#_Toc187011612)

[پرسش دوم 9](#_Toc187011613)

[سوال 2.1 9](#_Toc187011614)

[سوال 2.2 10](#_Toc187011615)

[سوال 2.3 11](#_Toc187011616)

[سوال 2.4 12](#_Toc187011617)

[سوال 2.5 23](#_Toc187011618)

[کد پرسش دوم 25](#_Toc187011619)

[پرسش سوم 26](#_Toc187011620)

[سوال 3.1 26](#_Toc187011621)

[سوال 3.2 28](#_Toc187011622)

[سوال 3.3 31](#_Toc187011623)

[کد پرسش سوم 34](#_Toc187011624)

[پرسش چهارم 35](#_Toc187011625)

[کد پرسش چهارم 37](#_Toc187011626)

# فهرست تصاویر

[تصویر 1 - تابع ReLU 4](#_Toc187011473)

[تصویر 2 - تابع sigmoid 5](#_Toc187011474)

[تصویر 3 - تابع ELU 5](#_Toc187011475)

[تصویر 4 - جداسازی دیتا بصورت مثلث 7](#_Toc187011476)

[تصویر 5 - نقشه حرارتی دیتا 9](#_Toc187011477)

[تصویر 6 – هیستوگرام custcat با تحصیلات 10](#_Toc187011478)

[تصویر 7 - هیستوگرام custcat با مدت عضویت 10](#_Toc187011479)

[تصویر 8 - دقت به ازای افزایش epoch در 16 نورون 12](#_Toc187011480)

[تصویر 9 - دقت به ازای افزایش epoch در 32 نورون 12](#_Toc187011481)

[تصویر 10 - دقت به ازای افزایش epoch در (16, 8) نورون 12](#_Toc187011482)

[تصویر 11 - دقت به ازای افزایش epoch در (32, 16) نورون 13](#_Toc187011483)

[تصویر 12 - دقت به ازای افزایش epoch با Batch Normalization و 16 نورون 14](#_Toc187011484)

[تصویر 13 - دقت به ازای افزایش epoch با Batch Normalization و 32 نورون 15](#_Toc187011485)

[تصویر 14 - - دقت به ازای افزایش epoch با Batch Normalization و [16, 8] نورون 15](#_Toc187011486)

[تصویر 15 - دقت به ازای افزایش epoch با Batch Normalization و [32, 16] نورون 15](#_Toc187011487)

[تصویر 16 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با Batch Normalization 16](#_Toc187011488)

[تصویر 17 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با Batch Normalization 17](#_Toc187011489)

[تصویر 18 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با Batch Normalization 17](#_Toc187011490)

[تصویر 19 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با L2-Regularization 18](#_Toc187011491)

[تصویر 20 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با L2-Regularization 19](#_Toc187011492)

[تصویر 21 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با L2-Regularization 19](#_Toc187011493)

[تصویر 22 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با adam 21](#_Toc187011494)

[تصویر 23 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با adam 21](#_Toc187011495)

[تصویر 24 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با adam 21](#_Toc187011496)

[تصویر 25 - اولین حرف اشتباه تشخیص داده شده 29](#_Toc187011497)

[تصویر 26 - اولین تشخیص اشتباه در Missing point 31](#_Toc187011498)

[تصویر 27 - مقایسه MSE چهار حالت مختلف 35](#_Toc187011499)

# فهرست جداول

[جدول 1 - مقایسه دقت تعداد نورون های متفاوت 11](#_Toc187011568)

[جدول 2 - مقایسه عملکرد Batch Normalization در حالات مختلف 14](#_Toc187011569)

[جدول 3 - مقایسه عملکرد Dropout در بهترین سه مدل 16](#_Toc187011570)

[جدول 4 - مقایسه عملکرد L2-Regularization در بهترین سه مدل 18](#_Toc187011571)

[جدول 5 - - مقایسه عملکرد adam در بهترین سه مدل 20](#_Toc187011572)

[جدول 6 - پیشبینی بهترین سه مدل در 10 داده تست رندوم 22](#_Toc187011573)

[جدول 7 - پسشبینی مدل ترکیب شده برای همان 10 داده تست قبلی 23](#_Toc187011574)

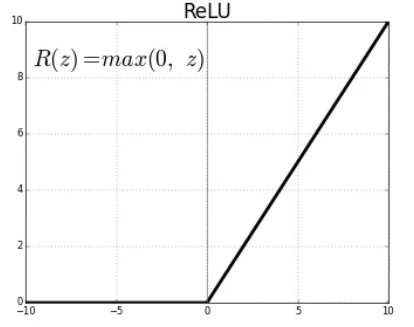
# پرسش یک

## سوال 1.1

فرض کنید در یک مسأله طبقه بندی دوکلاسه، دو لایه انتهایی شبکه شما فعال ساز ReLU و سیگموید است. چه اتفاقی می افتد.

ReLU: یکی از رایج‌ترین توابع فعال‌ساز در شبکه‌های عصبی است که به شکل زیر تعریف می‌شود:

به این معنی که هر مقدار منفی به صفر تبدیل می‌شود و مقادیر مثبت بدون تغییر عبور داده می‌شوند.



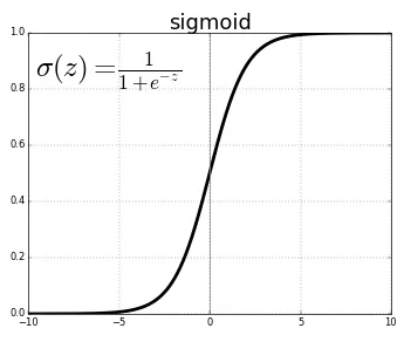
تصویر 1 - تابع ReLU

**مزایا:** محاسبات بسیار سریع و ساده‌ای دارد که باعث افزایش سرعت یادگیری شبکه می‌شود. برخلاف سیگموید، گرادیان ReLU در اکثر نقاط صفر نمی‌شود، به‌جز مقادیر منفی، و این باعث یادگیری بهتر شبکه‌های عمیق می‌شود. همچنین عملکرد بسیار خوبی در شبکه‌های عمیق دارد و در اکثر معماری‌های مدرن استفاده می‌شود.

**معایب:** اگر مقدار ورودی به ReLU همیشه منفی باشد، خروجی آن همیشه صفر خواهد بود و این نرون عملاً غیرفعال می‌شود.همچنین خروجی ReLU می‌تواند بسیار بزرگ شود که ممکن است باعث بی‌ثباتی در برخی موارد شود.

Sigmoid: یک تابع غیرخطی است که خروجی آن به صورت زیر تعریف می‌شود:

این تابع خروجی را به بازه 0 تا 1 نگاشت می‌کند و معمولاً برای مدل‌سازی احتمال استفاده می‌شود.



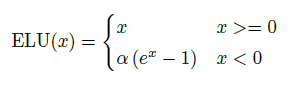
تصویر 2 - تابع sigmoid

**مزایا:** چون خروجی سیگموید بین 0 تا 1 است، برای مسائلی مثل طبقه‌بندی دوتایی مناسب است. چون یک تابع غیرخطی است می‌تواند وابستگی‌های پیچیده بین ورودی و خروجی را مدل کند.

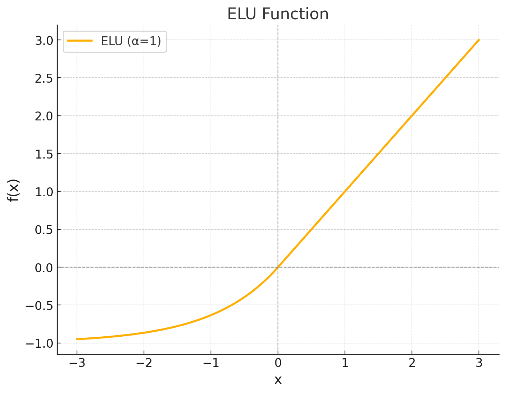
**معایب:** در مقادیر بسیار کوچک یا بزرگ ورودی، گرادیان تابع سیگموید بسیار کوچک می‌شود، که باعث کند شدن یادگیری در شبکه‌های عمیق می‌شود. همچنین به دلیل محاسبه نمایی زمان بیشتری برای پردازش نیاز دارد و اگر ورودی خیلی بزرگ یا خیلی کوچک باشد، خروجی به صفر یا یک نزدیک می‌شود و باعث کاهش حساسیت شبکه به تغییرات ورودی می‌گردد.

## سوال 1.2

یک جایگزین برای ReLU در معادله زیر آورده شده است. ضمن محاسبه گرادیان آن، حداقل یک مزیت آن نسبت به ReLU را بیان کنید.



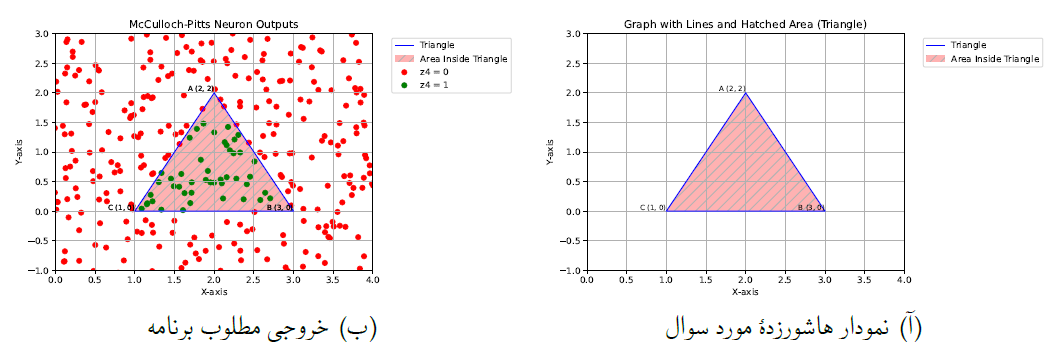
ELU یک نسخه پیشرفته‌تر از ReLU است که مشکل نرون‌های مرده (خروجی صفر به ازای مقدار منفی) را کاهش می‌دهد.



تصویر 3 - تابع ELU

## سوال 1.3

به کمک یک نرون ساده یا پرسپترون یا نرون McCulloch-Pitts، شبکه‌ای طراحی کنید که بتواند ناحیه هاشورزدهٔ داخل مثلثی که در نمودار شکل (۱) نشان داده شده را از سایر نواحی تفکیک کند. پس از انجام مرحلهٔ طراحی شبکه (که می‌تواند به‌صورت دستی انجام شود)، برنامه‌ای که در این [دفترچهٔ کار](https://colab.research.google.com/drive/1Jhlt-gYzQSU4PHNkoRk7zM89JVZzVz_2?usp=sharing) و کلاس برای نرون McCulloch-Pitts آموخته‌اید، را به‌گونه‌ای توسعه دهید که ۲۰۰۰ نقطهٔ رندوم تولید کند و آن‌ها را به‌عنوان ورودی به شبکهٔ طراحی‌شدهٔ شما بدهد و نقاطی که خروجی «۱» تولید می‌کنند را با رنگ سبز و نقاطی که خروجی «۰» تولید می‌کنند را با رنگ قرمز نشان دهد. خروجی تولیدشده توسط برنامهٔ شما باید به‌صورتی که در شکل (ب) نشان داده شده است باشد (به محدودهٔ عددی محورهای x و y هم دقت کنید). اثر اضافه‌کردن دو تابع فعال‌ساز مختلف به فرآیند تصمیم‌گیری را هم بررسی کنید.



تعریف نرون مک‌کالوک-پیتس: یک کلاس تعریف شده که به‌عنوان یک نرون ساده عمل می‌کند. این نرون دو ویژگی اصلی دارد: وزن‌ها و آستانه. وزن‌ها تأثیر هر ورودی را مشخص می‌کنند و آستانه تعیین می‌کند که آیا نرون فعال (روشن) باشد یا غیرفعال (خاموش).

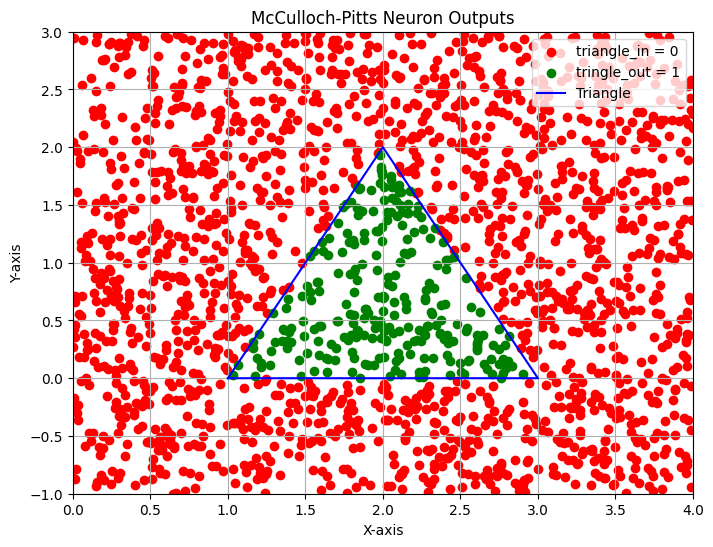
اگر حاصل ضرب وزن‌ها در ورودی‌ها از مقدار آستانه بیشتر یا مساوی باشد، خروجی نرون 1 است (فعال). اگر کمتر باشد، خروجی 0 خواهد بود (غیرفعال). این نرون یک ابزار ساده اما مؤثر برای تصمیم‌گیری خطی است.

ایجاد نرون‌های مربوط به مثلث: در این مرحله، چند نرون تعریف شده که هرکدام یک لبه از مثلث را مدل‌سازی می‌کنند:

هرکدام از این نرون‌ها یک مرز مثلث را شناسایی می‌کنند:

* نرون اول مرز بین نقاط A و C را مشخص می‌کند.
* نرون دوم مرز بین نقاط A و B را بررسی می‌کند.
* نرون سوم مرز بین نقاط B و C را تعیین می‌کند.

نرون چهارم: این نرون به‌عنوان یک ترکیب‌کننده عمل می‌کند. خروجی سه نرون اولیه را دریافت کرده و تصمیم می‌گیرد که آیا نقطه ورودی داخل مثلث است یا خارج از آن. در مجموع، این نرون‌ها مرزهای مثلث را شناسایی می‌کنند و امکان طبقه‌بندی نقاط به داخل یا خارج مثلث را فراهم می‌کنند.



تصویر 4 - جداسازی دیتا بصورت مثلث

استفاده از تابع های فعال ساز برای این نوع نرون ممکن نیست پس بخش آخر انجام نشده است.

## [کد پرسش یک](https://colab.research.google.com/drive/13UQ3bMxLwhsr280OJhZWxz2T5nhDWNJM?usp=sharing)

# پرسش دوم

تصور کنید یک شرکت مخابراتی مشتریان خود را بر اساس الگوهای استفاده از خدمات، به چهار گروه تقسیم کرده است. اگر بتوان با استفاده از داده های جمعیت شناختی عضویت در گروه ها را پیش بینی کرد، شرکت می تواند پیشنهادهای ویژه ای برای مشتریان احتمالی ارائه دهد. این مسئله یک مسئله طبقه بندی است. یعنی با داشتن مجموعه داده ای با برچسب های از پیش تعیین شده، می توان مدلی ارائه کرد که بتواند کلاس مورد نظر یک نمونه جدید یا ناشناخته را پیش بینی کند. در دنیای مدرن یادگیری عمیق، طراحی و آموزش شبکه های عصبی به یکی از مهمترین چالش ها در حوزه ی یادگیری ماشین تبدیل شده است. در این سوال از داده های جمعیت شناختی مانند منطقه جغرافیایی، سن و وضعیت تأهل برای پیش بینی الگوهای مصرف استفاده میکنیم. به این منظور مجموعه داده "telecust 1000t" بهره میبریم. در این مجموعه فیلد هدف که custcat نام دارد دارای چهارمقدار ممکن است که به چهارگروه مشتریان مربوط میشود:

* خدمات پایه
* خدمات الکترونیکی
* خدمات پیشرفته
* خدمات کامل

هدف ما ساخت یک طبقه بند برای پیش بینی کلاس نمونه های ناشناخته است. در این پروژه از یک شبکه عصبی چندلایه MLP استفاده خواهیم کرد. در این راستا، پارامترهایی همچون تعداد لایه های مخفی، تعداد نورون ها، استفاده از تکنیک هایی مانند Dropout، L2-Regularization و انتخاب بهترین روش بهینه سازی، نقش مهمی در دستیابی به عملکرد بهینه مدل دارند. در این سوالات، طراحی و آموزش مدل های شبکه عصبی با تنظیمات مختلف بررسی شده و تأثیر تنظیمات گوناگون هایپر پارامتر ها، منظم کننده ها و روش های بهینه سازی بر روی مجموعه داده Telecust 1000t تحلیل میشود.

حال با توجه به این توضیحات به سوالات زیر پاسخ دهید.

## سوال 2.1

فایل csv مجموعه داده را با استفاده از کتابخانه pandas بخوانید.

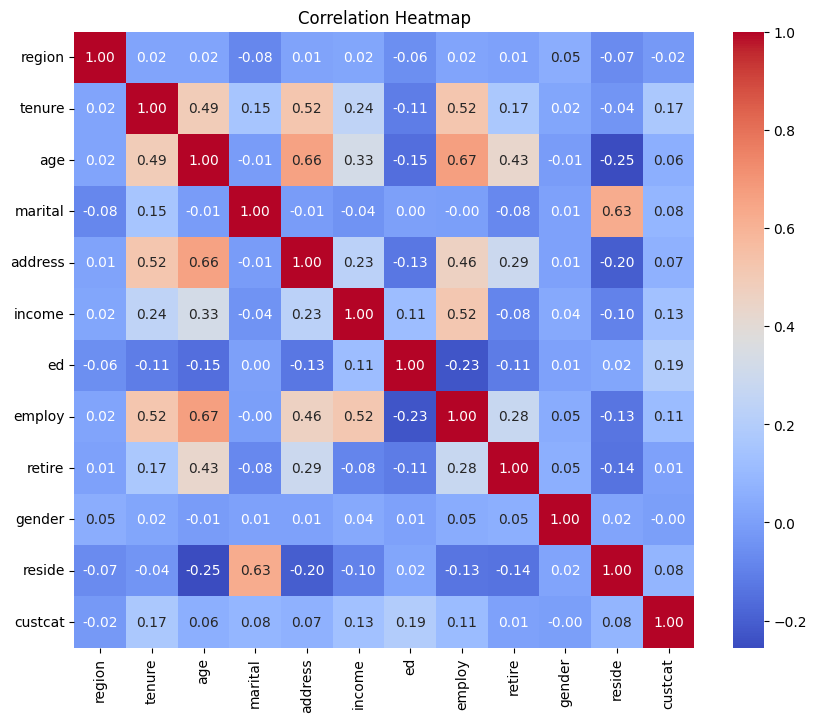
داده ها شامل ستون های زیر است:

* region: منطقه
* tenure: مدت عضویت
* age: سن
* marital: وضعیت تأهل
* address: آدرس
* income: درآمد
* ed: تحصیلات
* employ: اشتغال
* retire: بازنشستگی
* gender: جنسیت
* reside: محل سکونت
* custcat: دسته‌بندی مشتری (4 نوع)

هیچ دیتای NULL در داده ها وجود ندارد. دو دیتا(بازنشستگی و درآمد) از تایپ float و بقیه همه int هستند.

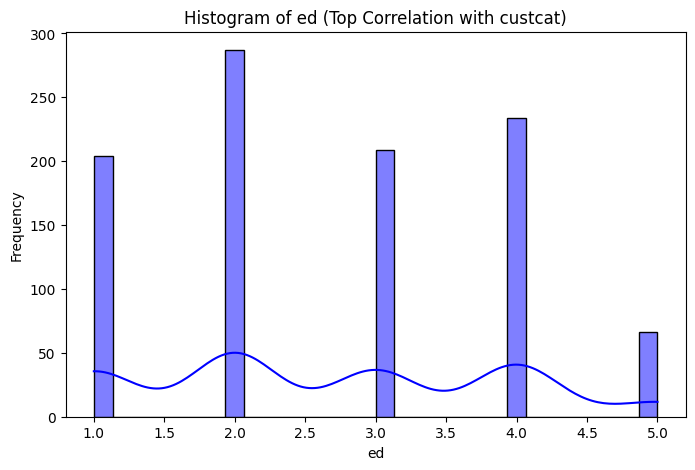
## سوال 2.2

با استفاده از کتابخانه seaborn یا مشابه هیت مپ مربوط به این مجموعه داده را تولید و تحلیل کرده و سپس هیستوگرام دو ویژگی که بیشترین همبستگی را با فیلد هدف دارند رسم کنید.

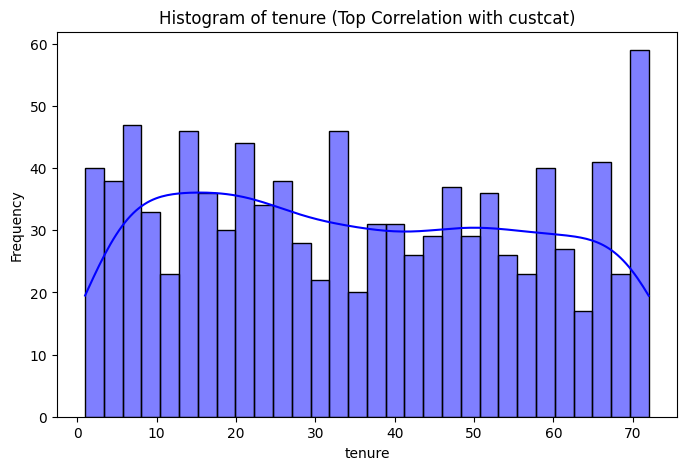


تصویر 5 - نقشه حرارتی دیتا

همانطور که در دیتا دیده میشود دو ویژگی تحصیلات (0.19) و مدت عضویت (0.17) بیشترین همبستگی را با داده هدف دارند.



تصویر 6 – هیستوگرام custcat با تحصیلات



تصویر 7 - هیستوگرام custcat با مدت عضویت

## سوال 2.3

داده ها را با استفاده از MinMaxScaler نرمالایز کنید و به سه دسته validation test train تقسیم کنید.

* داده های آموزش: 80 درصد
* داده های ولیدیشن: 5 درصد
* داده های تست: 15 درصد

## سوال 2.4

دو شبکه عصبی چندلایه (MLP) برای مجموعه داده طراحی کنید که به ترتیب مدل اول شامل یک لایه مخفی و مدل دوم شامل دو لایه مخفی باشد و از بهینه ساز SGD برای آموزش شبکه ها استفاده کنید. سپس با توجه به شرایط زیر و با استفاده از مجموعه داده های train و validation عمل کنید:

**بررسی تعداد نورونها بر عملکرد مدل**

برای تطبیق برچسب‌های هدف با خروجی مدل، مقدار همه‌ی برچسب‌ها یک واحد کاهش داده شده تا در بازه [0 3] قرار گیرند.

**مدل یک لایه‌ی مخفی:** این مدل شامل یک لایه‌ی مخفی با تعداد نورون‌های مشخص و یک لایه‌ی خروجی با 4 نورون و تابع فعال‌سازی Softmax است. دو حالت 16 یا 32 نورون.

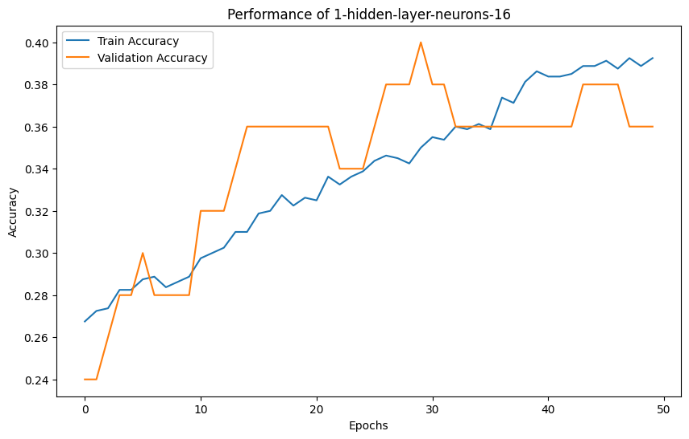
**مدل دو لایه‌ی مخفی:** این مدل دارای دو لایه‌ی مخفی است. دو حالت ترکیبات (16, 8) و (32, 16).

سافت مکس: در لایه‌ی خروجی، تابع Softmax برای فعال‌سازی نورون‌های خروجی استفاده شده است.

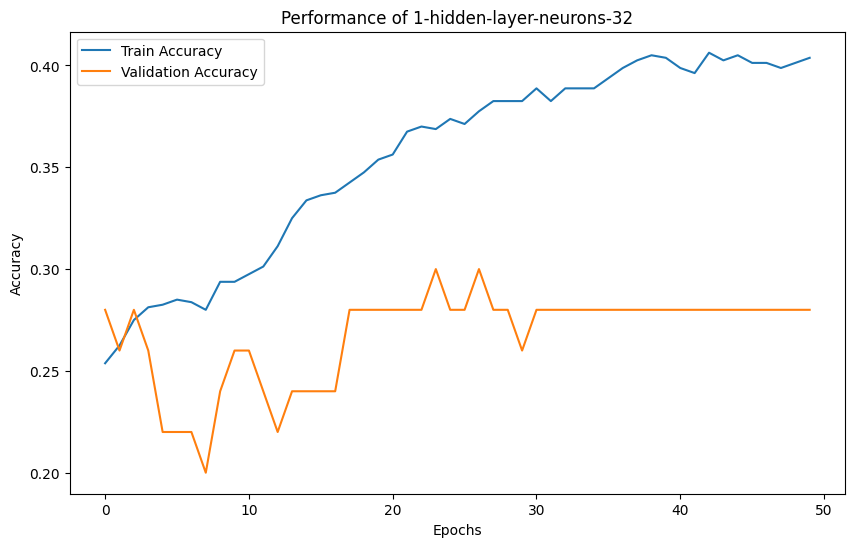
این تابع احتمال هر کلاس را محاسبه می‌کند و مجموع احتمالات برابر 1 خواهد بود. مناسب برای مسائل دسته‌بندی چندکلاسه است. خروجی نورون با بالاترین مقدار احتمال به عنوان پیش‌بینی مدل انتخاب می‌شود.

جدول 1 - مقایسه دقت تعداد نورون های متفاوت

|  |  |
| --- | --- |
| Validation accuracy | Model |
| 0.36 | 1-hidden-layer-neurons-32 |
| 0.34 | 2-hidden-layers-neurons-(32, 16) |
| 0.28 | 1-hidden-layer-neurons-16 |
| 0.26 | 2-hidden-layers-neurons-(16, 8) |



تصویر 8 - دقت به ازای افزایش epoch در 16 نورون



تصویر 9 - دقت به ازای افزایش epoch در 32 نورون

A graph of a line

Description automatically generated with medium confidence

تصویر 10 - دقت به ازای افزایش epoch در (16, 8) نورون

A graph of a line

Description automatically generated with medium confidence

تصویر 11 - دقت به ازای افزایش epoch در (32, 16) نورون

همانطور که مشخص است دقت مدل در همه حالات اصلا خوب نیست ولی میتوان نتیجه گرفت که 32 نورون چه در یک لایه چه در دولایه بهتر از 16 نورون عمل کرده است.

**تأثیر اضافه کردن لایه نرمال سازی دسته**

Batch Normalization در هر لایه از مدل، مقادیر ورودی آن لایه را نرمال‌سازی می‌کند. این فرآیند شامل مراحل زیر است:

1. محاسبه میانگین و انحراف معیار برای هر دسته (Batch): فرض کنید ورودی به لایه x باشد. ابتدا میانگین () و انحراف معیار () روی دسته محاسبه می‌شود.
2. مقادیر ورودی را به یک توزیع با میانگین صفر و واریانس یک تبدیل می‌کند.
3. برای اطمینان از اینکه لایه همچنان می‌تواند مقادیر دلخواه را یاد بگیرد، دو پارامتر قابل یادگیری معرفی می‌شوند:

* : مقیاس
* : انتقال

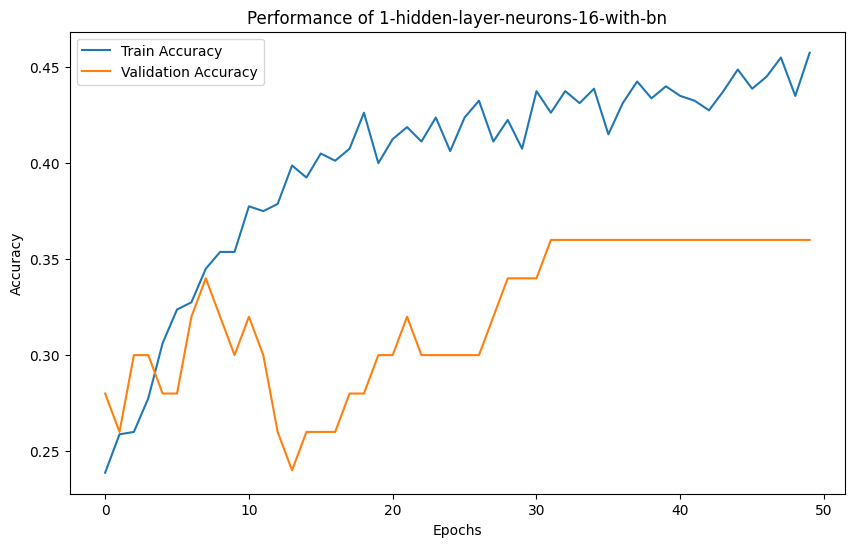
1. خروجی:

**مزایا:**

* پایداری آموزش (Stability): با نرمال‌سازی ورودی هر لایه، تغییرات شدید در مقادیر ورودی به لایه‌ها کاهش می‌یابد و فرآیند یادگیری پایدارتر می‌شود.
* سرعت بخشیدن به آموزش: کاهش نیاز به یادگیری ویژگی‌های اولیه از داده‌ها، باعث همگرایی سریع‌تر مدل می‌شود.
* کاهش حساسیت به مقادیر اولیه وزن‌ها: مدل کمتر به تنظیم دقیق وزن‌های اولیه وابسته است.
* جلوگیری از بیش‌برازش (Overfitting): نرمال‌سازی دسته‌ای به عنوان یک مکانیزم تنظیم‌کننده (Regularization) عمل می‌کند، هرچند که کاملاً جایگزین Dropout نمی‌شود.
* انعطاف‌پذیری در انتخاب نرخ یادگیری (Learning Rate): مدل‌ها با Batch Normalization می‌توانند با نرخ یادگیری بالاتر آموزش ببینند.

جدول 2 - مقایسه عملکرد Batch Normalization در حالات مختلف

| Model | Validation Accuracy |
| --- | --- |
| 2-hidden-layers-neurons-(32, 16)-with-bn | 0.42 |
| 2-hidden-layers-neurons-(16, 8)-with-bn | 0.38 |
| 1-hidden-layer-neurons-32-with-bn | 0.32 |
| 1-hidden-layer-neurons-16-with-bn | 0.36 |



تصویر 12 - دقت به ازای افزایش epoch با Batch Normalization و 16 نورون

A graph of a number of blue and orange lines

Description automatically generated

تصویر 13 - دقت به ازای افزایش epoch با Batch Normalization و 32 نورون

A graph of blue and orange lines

Description automatically generated

تصویر 14 - - دقت به ازای افزایش epoch با Batch Normalization و [16, 8] نورون

A graph of a line graph

Description automatically generated with medium confidence

تصویر 15 - دقت به ازای افزایش epoch با Batch Normalization و [32, 16] نورون

همانطور که در جدول قابل مشاهده است استفاده از Batch Normalization باعث شده است که دقت در تمام ترکیب ها بیشتر شود اما همچنان دقت بشدت پایین است و نمیتوان به آن اعتماد کرد.

**تاثیر Dropout**

Dropout یکی از تکنیک‌های معروف برای تنظیم‌سازی (Regularization) در شبکه‌های عصبی است که با کاهش احتمال بیش‌برازش (Overfitting) به بهبود عملکرد مدل کمک می‌کند.

Dropout در طی آموزش، به طور تصادفی تعدادی از نورون‌های یک لایه را غیرفعال (Drop) می‌کند. غیرفعال کردن به این معناست که:

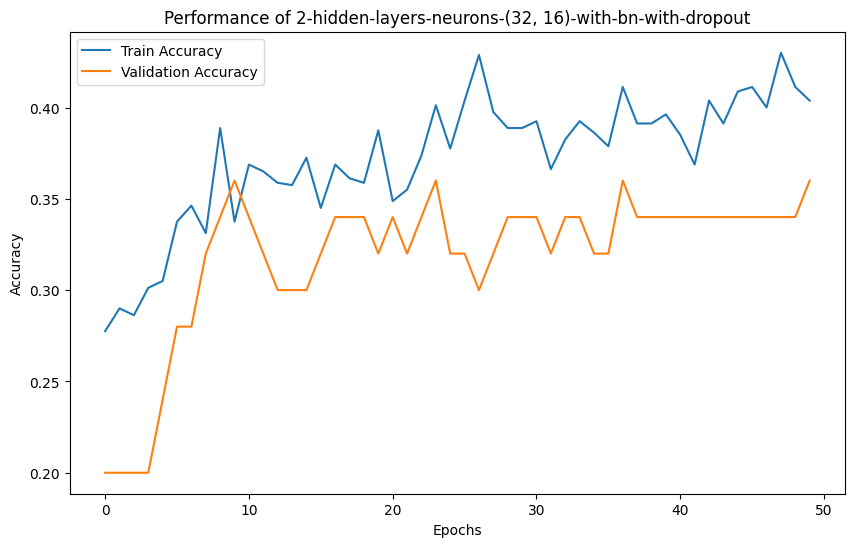
* خروجی نورون برابر صفر می‌شود.
* وزن‌های مربوط به نورون‌های غیرفعال در آن گام به‌روزرسانی نمی‌شوند.

**مزایا:**

* جلوگیری از بیش‌برازش
* پایداری مدل: مدل به جای یادگیری ویژگی‌های خاص از داده‌ها، ویژگی‌های عمومی‌تری را یاد می‌گیرد.
* افزایش قدرت تعمیم‌پذیری (Generalization): عملکرد مدل روی داده‌های دیده‌نشده (تست) بهبود می‌یابد.

جدول 3 - مقایسه عملکرد Dropout در بهترین سه مدل

|  |  |
| --- | --- |
| Validation Accuracy | Model |
| 0.36 | 2-hidden-layers-neurons-(32, 16)-with-bn-with-dropout |
| 0.30 | 2-hidden-layers-neurons-(16, 8)-with-bn-with-dropout |
| 0.34 | 1-hidden-layer-neurons-32-with-bn-with-dropout |



تصویر 16 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با Batch Normalization

A graph of a graph

Description automatically generated with medium confidence

تصویر 17 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با Batch Normalization

A graph of blue and orange lines

Description automatically generated

تصویر 18 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با Batch Normalization

از مزایای Dropout گفتیم اما چیزی که به آن کمتر اشاره شد معایب آن است. در این دیتا که فقط 1000 نمونه داریم و 10 ویژگی داده شده همبستگی بسیار پایینی با خروجی دارند(یعنی با مسئله پیچیده ای روبرو هستیم) این تکنیک نمیتواند خیلی موثر باشد و همانطور که قابل مشاهده است دقت ما نسبت به مرحله قبل خیلی کمتر شده است پس ازین سه مدل در ادامه استفاده نخواهیم کرد.

**L2-Regularization**

در مسائل یادگیری ماشین، اگر مدل پیچیدگی زیادی داشته باشد یا داده‌های آموزشی نویزی باشند، ممکن است مدل به جای یادگیری الگوهای کلی، بیش‌ازحد داده‌های آموزشی را به خاطر بسپارد. این مسئله باعث می‌شود که مدل روی داده‌های جدید (داده‌های آزمون) عملکرد ضعیفی داشته باشد.

L2-Regularization با محدود کردن وزن‌های مدل به کوچکتر شدن آن‌ها کمک می‌کند و از این طریق از پیچیدگی مدل می‌کاهد.

در روش L2-Regularization، به تابع هزینه (Loss Function) مدل، جمله‌ای اضافه می‌شود که مجموع مربعات وزن‌ها را شامل می‌شود. فرمول کلی تابع هزینه جدید به صورت زیر است:

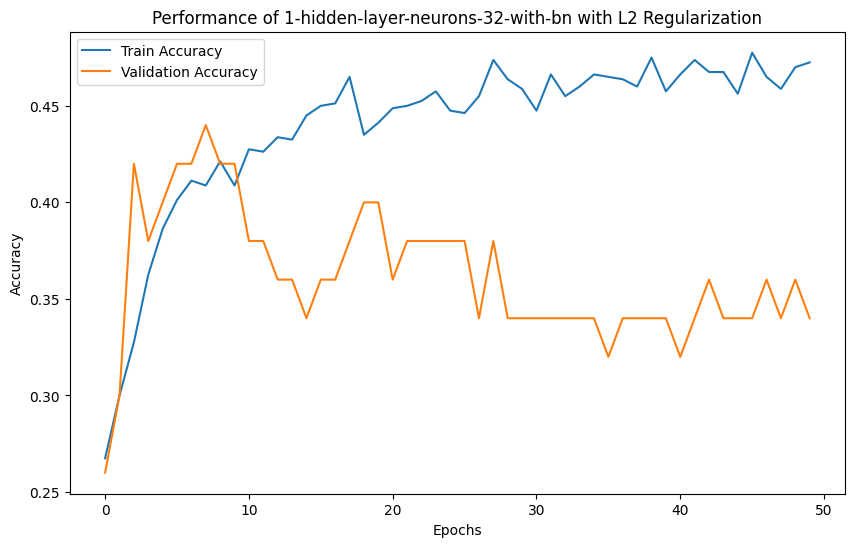
*: ضریب* regularization *که کنترل می‌کند جریمه وزن‌ها چقدر باشد.*

*ویژگی:*

* *کوچک‌سازی وزن‌ها*
* *عدم حذف کامل وزن‌ها*
* *کنترل پیچیدگی مدل*

جدول 4 - مقایسه عملکرد L2-Regularization در بهترین سه مدل

| Model | Validation Accuracy |
| --- | --- |
| 2-hidden-layers-neurons-(32, 16)-with-bn | 0.38 |
| 1-hidden-layer-neurons-32-with-bn | 0.34 |
| 2-hidden-layers-neurons-(16, 8)-with-bn | 0.32 |



تصویر 19 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با L2-Regularization

A graph of a line graph

Description automatically generated with medium confidence

تصویر 20 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با L2-Regularization

A graph of blue and orange lines

Description automatically generated

تصویر 21 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با L2-Regularization

همانطور که مشاهده میشود مدل ها دقت بیشتری نسبت به حالت قبل ارائه داده اند اما باز هم مشاهده میشود که دقت به شدت کم است.

**تاثیر مدل Adam**

Adam ترکیبی از دو روش Momentum و RMSProp است:

* Momentum: جهت حرکت میانگین‌گیری می‌شود تا نوسانات کاهش یابد.
* RMSProp: نرخ یادگیری را به صورت تطبیقی برای هر پارامتر تنظیم می‌کند.

Adam از دو میانگین متحرک استفاده می‌کند:

* میانگین گرادیان‌ها (​): جهت کلی حرکت.
* میانگین مربع گرادیان‌ها (​): تنظیم نرخ یادگیری.
* اصلاح انحراف در ابتدا برای دقیق‌تر کردن تخمین‌ها انجام می‌شود.
* وزن‌ها با استفاده از این مقادیر به‌روزرسانی می‌شوند.

میانگین متحرک گرادیان:

میانگین متحرک مربع گرادیان:

*اصلاح انحراف:*

*به‌روزرسانی وزن:*

: نرخ یادگیری

​: نرخ کاهش میانگین گرادیان

​: نرخ کاهش میانگین مربع گرادیان

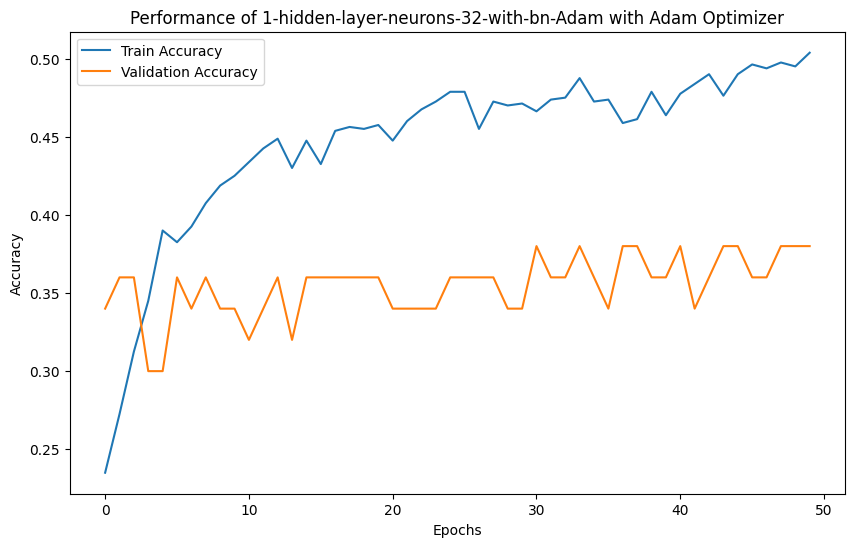
: مقدار کوچک برای جلوگیری از تقسیم بر صفر

**مزایا:**

* نرخ یادگیری تطبیقی برای هر پارامتر.
* همگرایی سریع‌تر و پایدارتر.
* مناسب برای داده‌های noisy و مسائل پیچیده.

جدول 5 - - مقایسه عملکرد adam در بهترین سه مدل

| Model | Validation Accuracy |
| --- | --- |
| 1-hidden-layer-neurons-32-with-bn-Adam | 0.38 |
| 2-hidden-layers-neurons-(32, 16)-with-bn-Adam | 0.38 |
| 2-hidden-layers-neurons-(16, 8)-with-bn-Adam | 0.20 |



تصویر 22 - دقت به ازای افزایش epoch با Dropout در مدل 32 نورون با adam

A graph with orange lines and white text

Description automatically generated

تصویر 23 - دقت به ازای افزایش epoch با Dropout در مدل [32, 16] با adam

A graph of a number of people

Description automatically generated with medium confidence

تصویر 24 - دقت به ازای افزایش epoch با Dropout در مدل [16, 8] با adam

با دیدن نتایج نهایی در این متد متوجه میشویم که برای دوتا از حالات به دقت بهتری نسبت به تمام حالات قبل رسیده ایم که یعنی برای این مسئله این متد عملکرد به مراتب بهتری دارد. البته که همچنان دقت پایین است و همچنین ممکن است که دقت بالا به دلیل اورفیت شدن مدل باشد و فقط همه را یک مقدار پیشبینی میکند که این موضوع را در قسمت ها بعد بررسی میکنیم.

## سوال 2.5

ارزیابی بهترین مدلهای بخش قبل را بر روی دادههای test انجام داده و تحلیل کنید سپس 10 نمونه را از داده ای test بصورت تصادفی انتخاب کرده و خروجیهای شبکه ها را با مقادیر واقعی مربوطه گزارش کنید.

جدول 6 - پیشبینی بهترین سه مدل در 10 داده تست رندوم

| Sample Index | Actual Value | 1-hidden-layer-neurons-32-with-bn-Adam | 2-hidden-layers-neurons-(32, 16)-with-bn | 2-hidden-layers-neurons-(32, 16)-with-bn-Adam |
| --- | --- | --- | --- | --- |
| 30 | 0 | 1 | 0 | 2 |
| 126 | 1 | 1 | 0 | 3 |
| 108 | 1 | 2 | 0 | 1 |
| 47 | 0 | 1 | 0 | 3 |
| 45 | 1 | 1 | 0 | 3 |
| 54 | 3 | 1 | 0 | 3 |
| 43 | 2 | 2 | 0 | 3 |
| 144 | 3 | 2 | 0 | 0 |
| 32 | 1 | 2 | 0 | 2 |
| 10 | 1 | 1 | 0 | 3 |

با مشاهده جدول به وضوح مشاهده میشود که مدل دولایه با لایه نرمال سازی کاملا اورفیت شده است و فقط مقدار 0 را نمایش میدهد و برای همین موضوع است که دقت بالایی نشان میداد. مدل یک لایه با آدام دقت 40 درصد در این 10 نمونه نتیجه میدهد. مدل دو لایه با آدام 20 درصد دقت در این 10 نمونه تصادفی نشان میدهد.

6. بهترین مدلهای به دست آمده از دو شبکه طراحی شده در بندهای قبل را به یکی از روشهایی که آموخته اید ترکیب کنید و نتایج تست را برای این حالت هم محاسبه کرده و نمایش دهید آیا نتایج بهبود پیدا کردند؟ چرا؟

دو مدل بهتر آدام را ترکیب میکنیم و برای نتیجه معیار را Averaging قرار میدهیم. به این معنی است که خروجی دو مدل (احتمالات پیش‌بینی شده برای هر کلاس) ترکیب شده و میانگین آن‌ها گرفته می‌شود. سپس از این میانگین برای تعیین کلاس نهایی استفاده می‌شود.

**مزایای این روش:**

* کاهش نویز
* افزایش دقت
* استفاده از قدرت چند مدل

جدول 7 - پسشبینی مدل ترکیب شده برای همان 10 داده تست قبلی

| Index | Actual Value | Combined Prediction |
| --- | --- | --- |
| 30 | 0 | 2 |
| 126 | 1 | 1 |
| 108 | 1 | 2 |
| 47 | 0 | 1 |
| 45 | 1 | 1 |
| 54 | 3 | 1 |
| 43 | 2 | 1 |
| 144 | 3 | 2 |
| 32 | 1 | 2 |
| 10 | 1 | 1 |

همانطور که مشاهده میشود این کار دقت را نه تنها افزایش نداد برای همان 10 داده قبلی به 20 درصد کاهش داد که اصلا نمیتواند نتیجه خوبی باشد. این اتفاق میتواند دلایل متنوعی داشته باشد که احتمالا مهمترین دلیل آن کیفیت پایین مدل‌های اولیه ما است.

دقت پایین مدل می‌تواند به دلایل متعددی مرتبط باشد. در زیر به چند دلیل اصلی اشاره می‌کنیم:

1. داده‌ها

* نویز در داده‌ها
* حجم ناکافی داده
* کم بودن همبستگی میان داده هدف و ویژگی ها

1. مدل

* مدل بیش از حد ساده است(underfit)
* مدل بیش از حد پیچیده است(overfit)
* Learning Rate نامناسب
* تعداد epoch‌ها
* Regularization ضعیف

1. بهینه‌سازی و الگوریتم‌ها

* Optimizer نامناسب
* عدم تغییر وزن مناسب

1. مشکلات پیاده‌سازی

* لایه‌های کم
* نورون های کم

## [کد پرسش دوم](https://colab.research.google.com/drive/12UCB8Hx4P11jN8sLbfSgQy2iJmPntNVJ?usp=sharing)

# پرسش سوم

به این [دفترچه کد](https://colab.research.google.com/drive/10dECB187apZS7bqGSGPnEIeUlupte8Fr?usp=sharing) مراجعه کنید و با اجرای سلول اول، ۵ داده تصویری مربوط به حروف الفبای فارسی که در شکل ۲ نشان داده شده است را دریافت کنید و سپس به سوالات زیر پاسخ دهید. دقت داشته باشید که در هر مرحله ارائه توضیحات متنی و دیداری مناسب لازم است. مثلا می توانید ورودی نویزی و خروجی پیش بینی شده را در یک تصویر در کنار هم قرار دهید.



## سوال 3.1

دو تابع پایتونی در سلول های دوم و سوم این [دفترچه کد](https://colab.research.google.com/drive/10dECB187apZS7bqGSGPnEIeUlupte8Fr?usp=sharing) نوشته شده اند. اولین تابع تصویر را در ورودی خود دریافت و به صورت نمایش باینری درمی آورد و دومین تابع با افزودن نویز به داده ها، داده های جدید نویزی تولید می کند. در مورد نحوه عملکرد هریک از این توابع توضیح دهید. هم چنین، می توانید این دستورات را به صورتی بهتر و کارآمدتر بازنویسی کنید.

این کد شامل دو بخش اصلی است: تبدیل تصویر به نمایش باینری و ایجاد تصاویر نویزی.

**convertImageToBinary:**

این فانکشن تصویری را به نمایش باینری تبدیل می‌کند. در این نمایش، پیکسل‌های سفید با -1 و پیکسل‌های سیاه با 1 نشان داده می‌شوند.

مراحل:

1. بارگذاری تصویر
2. ایجاد ابزار رسم: با استفاده از `ImageDraw.Draw` ابزار رسم ایجاد می‌شود که امکان تغییر مقادیر پیکسل‌ها را فراهم می‌کند.
3. اندازه‌گیری ابعاد تصویر: عرض و ارتفاع تصویر با استفاده از ویژگی `size` تعیین می‌شود.
4. پیمایش تمام پیکسل‌ها: با استفاده از دو حلقه تودرتو، تمام پیکسل‌های تصویر پیمایش می‌شوند:

* مقدار رنگ‌های قرمز، سبز و آبی هر پیکسل (RGB) استخراج می‌شود.
* شدت کل پیکسل با جمع مقادیر `R+G+B` محاسبه می‌شود.
* اگر شدت کل بزرگ‌تر از مقدار آستانه تعیین‌شده باشد (در اینجا میانگین مقدار روشنایی پیکسل‌ها)، پیکسل سفید می‌شود (`255, 255, 255`) و مقدار -1 به نمایش باینری اضافه می‌شود.
* در غیر این صورت، پیکسل سیاه می‌شود (`0, 0, 0`) و مقدار 1 به نمایش باینری اضافه می‌شود.

1. بازگشت نمایش باینری: لیستی شامل مقادیر -1 و 1 بازگردانده می‌شود.

**generateNoisyImages:**

این فانکشن تصاویر نویزی را بر اساس یک لیست از مسیرهای تصاویر ورودی تولید کرده و در مسیرهای جدید ذخیره می‌کند.

مراحل:

1. تعریف لیست مسیرها:
2. پیمایش مسیر تصاویر: برای هر تصویر در لیست، فانکشن `getNoisyBinaryImage` فراخوانی شده و تصویر نویزی تولید می‌شود. مسیر تصویر نویزی تولید شده نیز در کنسول چاپ می‌شود.

**getNoisyBinaryImage:**

این فانکشن به هر پیکسل یک نویز تصادفی اضافه کرده و تصویر نویزی را ذخیره می‌کند.

مراحل:

1. بارگذاری تصویر ورودی
2. ایجاد ابزار رسم: ابزار رسم برای تغییر پیکسل‌های تصویر ایجاد می‌شود.
3. اندازه‌گیری ابعاد تصویر: عرض و ارتفاع تصویر تعیین می‌شود.
4. پیمایش تمام پیکسل‌ها: با استفاده از دو حلقه تودرتو، تمام پیکسل‌ها پیمایش شده و نویز اضافه می‌شود:

* یک مقدار نویز تصادفی بین -noise\_factor و +noise\_factor تولید می‌شود.
* نویز به مقادیر قرمز، سبز و آبی (RGB) هر پیکسل اضافه می‌شود.
* مقادیر نهایی رنگ‌ها با استفاده از شرط‌هایی بین `0` و `255` محدود می‌شوند تا از دامنه مجاز خارج نشوند.
* رنگ جدید به پیکسل اعمال می‌شود.

1. ذخیره تصویر نویزی: تصویر نویزی در مسیر مشخص‌شده ذخیره می‌شود.
2. پاک‌سازی ابزار رسم: ابزار رسم حذف می‌شود تا منابع آزاد شوند.

**بهبود توابع:**

مدیریت منابع بهتر: از ابزار اضافی مثل ImageDraw.Draw استفاده نمی‌کند و پردازش مستقیم پیکسل‌ها سریع‌تر است.

پیاده‌سازی ساده‌تر: ساختار واضح‌تر و خطایابی راحت‌تری دارد.

خروجی کاربردی‌تر: مستقیماً لیست باینری تولید می‌کند، در حالی که کد دوم فقط تصویر نویزی می‌سازد.

کنترل کیفیت بهتر: از آستانه شدت منطقی و نویز مناسب استفاده می‌کند.

مصرف حافظه کمتر: کد دوم ابزارها و حلقه‌های پیچیده‌تری دارد که منابع بیشتری مصرف می‌کند.

## سوال 3.2

یک شبکه عصبی (همینگ یا هاپفیلد) طراحی کنید که با اِعمال ورودی دارای میزان مشخصی نویز برای هر یک از داده ها، خروجی متناسب با آن داده نویزی را بیابد. میزان نویز را تا حدی که شبکه شما ناموفق عمل کند افزایش دهید و نتایج را مقایسه و تحلیل کنید.

از شبکه عصبی همینگ استفاده میکنیم. شبکه عصبی Hamming (یا شبکه‌های تطابق الگو بر پایه همینگ) یکی از معماری‌های شبکه‌های عصبی مصنوعی است که برای تشخیص الگو و بازیابی حافظه تداعی‌گر طراحی شده است. این شبکه معمولاً در مسائلی که به دنبال یافتن تطابق یک ورودی با یکی از الگوهای ذخیره‌شده در حافظه هستیم، استفاده می‌شود. در مقایسه با شبکه‌های Hopfield، شبکه همینگ ساده‌تر است و فقط برای تطابق و بازشناسی استفاده می‌شود.

کاربرد های این شبکه:

* تشخیص الگو (Pattern Recognition)
* بازشناسی کاراکترها (Character Recognition)
* تطابق رشته‌ها
* ذخیره و بازیابی اطلاعات در سیستم‌های حافظه تداعی‌گر

معماری شبکه همینگ

شبکه همینگ معمولاً دو لایه دارد:

1. لایه تطابق الگو (Matching Layer):

* محاسبه فاصله همینگ بین ورودی و هر الگوی ذخیره‌شده.
* فاصله همینگ نشان‌دهنده تعداد بیت‌های متفاوت بین دو بردار دودویی است.

1. لایه رقابتی (Competitive Layer):

* انتخاب الگوی ذخیره‌شده که بیشترین شباهت را با ورودی دارد.
* به صورت یک شبکه رقابتی عمل می‌کند، جایی که یک نرون فعال می‌شود و دیگر نرون‌ها خاموش می‌مانند (قانون برنده-همه‌چیز).

مراحل عملکرد:

* ورودی به شبکه: برداری دودویی یا عددی به شبکه وارد می‌شود.
* محاسبه فاصله همینگ: فاصله همینگ بین بردار ورودی و الگوهای ذخیره‌شده محاسبه می‌شود.
* انتخاب نزدیک‌ترین الگو: الگویی که کمترین فاصله همینگ را با ورودی دارد، به عنوان خروجی انتخاب می‌شود.
* تصحیح خطا: اگر الگوی ورودی مقداری نویز داشته باشد، شبکه می‌تواند الگوی صحیح را بازیابی کند.

**مزایا**

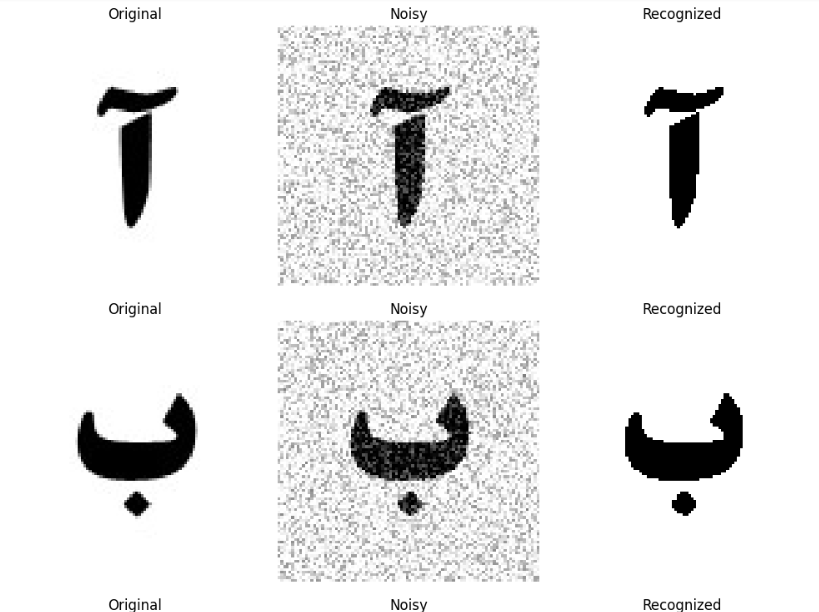
* سرعت بالا در تطابق الگوها به دلیل طراحی ساده.
* مقاوم در برابر نویز و تغییرات جزئی در ورودی.
* معماری ساده و مناسب برای پیاده‌سازی سخت‌افزاری.

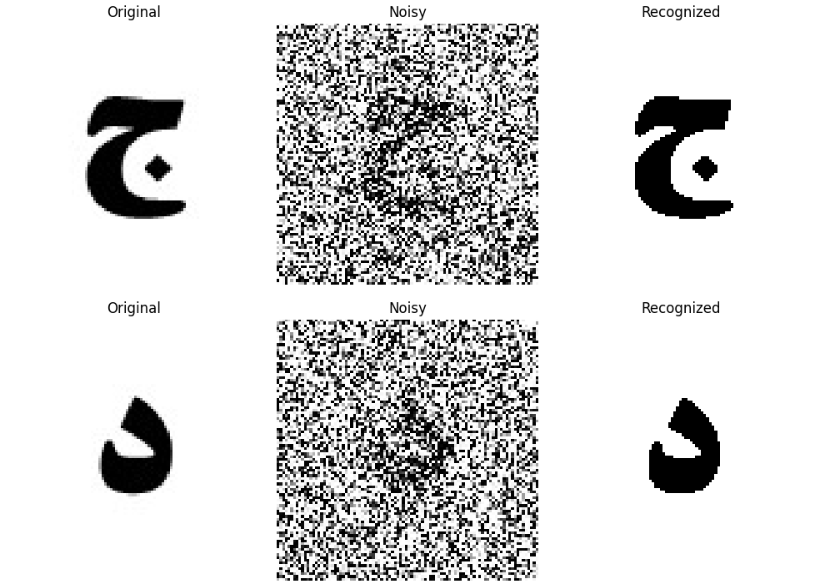
**معایب**

* محدودیت در تعداد الگوهای ذخیره‌شده (به دلیل کاهش تفکیک‌پذیری با افزایش الگوها).
* مناسب بودن فقط برای بردارهای دودویی یا فاصله‌های ساده (مثل فاصله همینگ).
* عدم امکان یادگیری پویا (الگوها از ابتدا مشخص هستند).

برای این بخش نویز از سطح 50 شروع میشود و تصاویر نویزی تولید می‌شوند و شبکه تا زمانی که بتواند تصاویر را با نویز مشخص شناسایی کند، سطح نویز را افزایش می‌دهد.

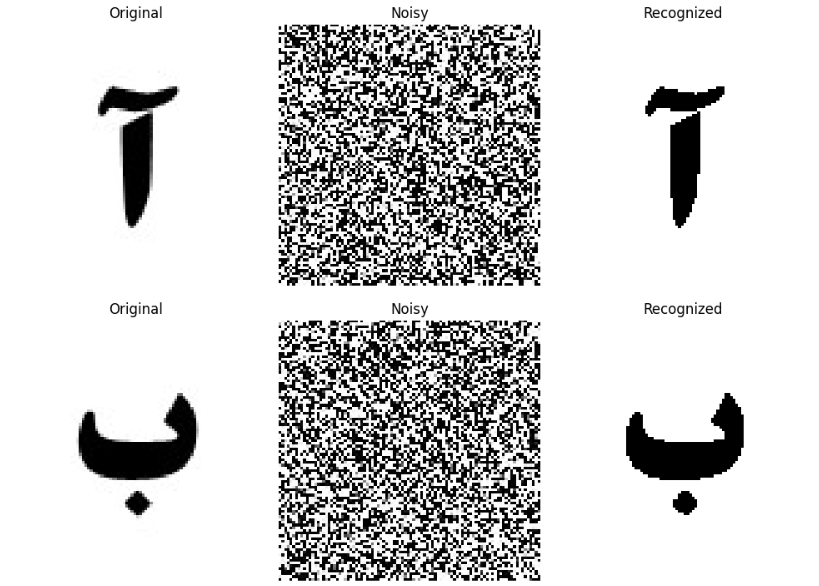
نمونه هایی از روند زیاد شدن نویز و تشخیص تصاویر:





A black and white image of a square

Description automatically generated



A group of black and white symbols

Description automatically generated

و اما در نویز لول 1900 تصویر پنجم به اشتباه تشخیص داده میشود برای اولین بار.

A black and white image of a square

Description automatically generated

تصویر 25 - اولین حرف اشتباه تشخیص داده شده

## سوال 3.3

با الهام گرفتن از تابع نوشته شده برای تولید داده های نویزی، یک تابع بنویسید که از داده های ورودی، خروجی های دارای Missing points تولید کند. سپس عملکرد شبکه خود را با مقدار مشخصی Missing point آزمایش و تحلیل کنید. اگر میزان Missing point از چه حدی بیش تر شود عملکرد شبکه طراحی شده شما دچار اختلال می شود؟ راه حل چیست؟ (راهنمایی: نمونه داده دارای Missing point در شکل ۳ نشان داده شده است.)

A black letter with white text

Description automatically generated

مراحل:

* تصاویر اصلی به بردارهای باینری تبدیل می‌شوند.
* ماتریس وزنی W بر اساس بردارهای آموزشی ساخته می‌شود.
* درصدی از پیکسل‌ها حذف می‌شوند (سفید می‌شوند).
* با هر تکرار، درصد نقص افزایش می‌یابد.
* بردار تصویر ناقص با الگوهای ذخیره‌شده تطبیق داده می‌شود.
* نزدیک‌ترین الگو بر اساس فاصله همینگ شناسایی می‌شود.
* تست از درصد نقص 10% شروع می‌شود.
* در هر مرحله، درصد نقص به میزان 10% افزایش می‌یابد.
* اگر شبکه نتواند یک تصویر را به درستی شناسایی کند، فرآیند متوقف شده و سطح تحمل شبکه گزارش می‌شود.

نمونه هایی از طی شدن روند:

A black and white image of letters

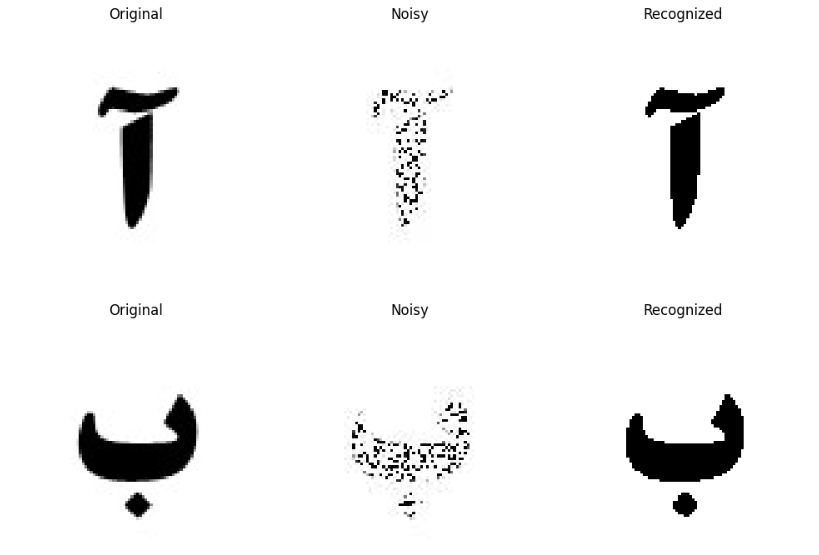
Description automatically generated

A group of black symbols

Description automatically generated

A black and white image of a person's head

Description automatically generated



در نهایت در 130% اولین اشتباه رخ میدهد:

A black and white image of a letter c

Description automatically generated

تصویر 26 - اولین تشخیص اشتباه در Missing point

راه‌حل‌های پیشنهادی برای بهبود عملکرد شبکه در مواجهه با داده‌های ناقص:

1. استفاده از شبکه‌های بازسازی تصویر (Autoencoders):

* اتواینکودرها به طور خاص برای یادگیری ویژگی‌های اصلی تصاویر طراحی شده‌اند.
* می‌توانید تصاویر نویزی را به عنوان ورودی به این شبکه‌ها داده و نسخه اصلی بازسازی‌شده را به شبکه اصلی ارائه کنید.
* این فرآیند به عنوان یک پیش‌پردازش عمل کرده و نویز موجود در تصاویر را کاهش می‌دهد.

1. افزایش تنوع داده‌های آموزشی:

* یکی از دلایل حساسیت شبکه به نویز، عدم تنوع کافی در داده‌های آموزشی است. برای حل این مشکل:
* در مرحله آموزش، عمداً تصاویر را با انواع نویزها مانند لکه‌های سفید، نویز گوسی، و اختلالات دیگر خراب کنید.
* این کار شبکه را تشویق می‌کند به جای جزئیات کوچک، روی ویژگی‌های اصلی و کلیدی تصویر تمرکز کند.

1. طراحی معماری مقاوم در برابر نویز:

* استفاده از معماری‌هایی که نسبت به نویز حساسیت کمتری دارند، می‌تواند عملکرد شبکه را بهبود بخشد:
* لایه‌های Dropout در حین آموزش، شبکه را مقاوم‌تر در برابر تغییرات تصادفی می‌سازند.
* معماری‌هایی مانند ResNet یا DenseNet، به دلیل وجود مسیرهای متعدد برای پردازش اطلاعات، به کاهش حساسیت شبکه به نویز کمک می‌کنند.

1. تکنیک‌های افزایش داده (Data Augmentation):

* اعمال تغییرات تصادفی روی تصاویر اصلی مانند:
* چرخش، برش، تغییر روشنایی، بزرگ‌نمایی، و دیگر تغییرات.
* این تکنیک‌ها شبکه را آموزش می‌دهند تا ویژگی‌های اصلی تصویر را حتی در مواجهه با تغییرات متنوع حفظ کند.

1. استفاده از تبدیل‌ها برای حذف نویز:

* برخی از نویزها در حوزه زمان یا مکان قابل شناسایی و حذف هستند. از تبدیل‌های مانند Wavelet یا Fourier می‌توان استفاده کرد:
* در تبدیل موجک، نویزها اغلب در مقیاس‌های خاصی ظاهر می‌شوند که قابل حذف هستند.
* این روش به خصوص برای تصاویر با نویز متناوب یا خاص مؤثر است.

1. استفاده از داده‌های مصنوعی با نویز زیاد:

* تولید تصاویر مصنوعی با سطوح مختلف نویز و استفاده از آن‌ها در مراحل آموزش:
* این کار باعث می‌شود شبکه در مواجهه با نویزهای مشابه، عملکرد بهتری داشته باشد.

با استفاده از این روش‌ها می‌توانید مقاومت شبکه را در برابر نویز افزایش داده و دقت آن را در شرایط داده‌های ناقص بهبود دهید.

## [کد پرسش سوم](https://colab.research.google.com/drive/11cXH17cKil4u3Uhg_imdpnayG6ZzEAGC?usp=sharing)

# پرسش چهارم

در این سوال به ساخت و آموزش دو مدل رگرسیون با یک لایه پنهان برای پیش‌بینی قیمت خانه‌ها با استفاده از دیتاست California Housing می‌پردازیم. هدف مقایسه عملکرد دو مدل مختلف است: یکی با لایه RBF و دیگری با لایه‌های کاملاً متصل (Dense). این دیتاست شامل ویژگی‌هایی مانند درآمد میانگین، سن خانه‌ها و تعداد اتاق‌ها در هر خانه است و متغیر هدف قیمت میانگین خانه‌ها است.

1. یک شبکه عصبی پیاده‌سازی کنید که در آن لایه‌ی RBF به عنوان لایه پنهان قرار باشد.

2. از Mean Squared Error (MSE) به عنوان تابع از دست دادن و از Adam به عنوان بهینه‌ساز برای آموزش استفاده کنید.

3. یک شبکه عصبی ساده با استفاده از لایه‌های کاملاً متصل (Dense) پیاده‌سازی کنید.

4. از Mean Squared Error (MSE) به عنوان تابع از دست دادن و از Adam به عنوان بهینه‌ساز برای آموزش استفاده کنید.

5. پس از آموزش هر دو مدل، آنها را روی مجموعه تست ارزیابی کرده و میزان از loss آنها را نمایش دهید.

6. عملکرد دو مدل را مقایسه کنید. کدام مدل بهتر عمل کرده‌است؟ پاسخ خود را توجیه کنید.

راهنمایی:

* از دیتاست موجود در sklearn.datasets استفاده کنید.
* از StandardScaler برای نرمال‌سازی ویژگی‌های ورودی استفاده کنید.
* دیتاست را به مجموعه‌های آموزشی و آزمایشی تقسیم کنید.

این داده شامل ویژگی‌هایی مانند سن خانه‌ها، جمعیت منطقه، تعداد اتاق‌ها و غیره است. با استفاده از `StandardScaler` داده‌ها را نرمالایز می‌کنیم (همه ویژگی‌ها را به یک بازه مشابه می‌بریم) و آن را به دو بخش تقسیم می‌کنیم:

داده‌های آموزشی (Train): 80 درصد

داده‌های تست (Test): 20 درصد

شبکه عصبی RBF-like با MLPRegressor

ساختار شبکه:

* یک لایه مخفی (Hidden Layer) با 100 نورون.
* از فعال‌سازی ReLU استفاده می‌کنیم که رفتار شبیه RBF را تا حدی شبیه‌سازی می‌کند.
* آپتیمایزر Adam وزن‌ها و بایاس‌ها را بهینه می‌کند.

این مدل RBF واقعی نیست، اما با استفاده از این لایه و MLPRegressor، رفتار نزدیک به آن را ایجاد می‌کنیم.

شبکه عصبی Dense با MLPRegressor

ساختار شبکه:

* دو لایه مخفی: لایه اول با 100 نورون. لایه دوم با 50 نورون.
* از فعال‌سازی ReLU استفاده می‌کنیم که برای مدل‌سازی روابط غیرخطی مناسب است.
* این مدل کامل‌ترین شبکه تمام‌متصل (Dense) را شبیه‌سازی می‌کند.

شبکه عصبی RBF با TensorFlow

یک لایه سفارشی برای محاسبه RBF طراحی کردیم:

* هر ورودی فاصله‌اش را با مراکز (Centers) حساب می‌کند.
* سپس خروجی گوسی برای هر نورون محاسبه می‌شود.

ساختار شبکه:

* یک لایه RBF با 100 نورون.
* لایه خروجی (Dense) برای پیش‌بینی مقدار نهایی.

شبکه عصبی Dense با TensorFlow

ساختار شبکه:

* لایه اول: 100 نورون با فعال‌سازی ReLU.
* لایه دوم: 50 نورون با فعال‌سازی ReLU.
* لایه خروجی: 1 نورون برای پیش‌بینی مقدار نهایی.

A graph of a bar graph

Description automatically generated

تصویر 27 - مقایسه MSE چهار حالت مختلف

همانطور که قابل مشاهده است بجز مدل RBF با tensorflow هر 3 مدل دیگر تقریبا MSE بسیار نزدیک و خوبی دارند ولی بهترین عملکرد را مدل dense با MLP داشته است که این میتواند دلایل متفاوتی داشته باشد که به چندتای آنها در ادامه اشاره میکنیم:

* لایه‌های Dense قادرند تعاملات پیچیده و غیرخطی بین ویژگی‌ها را یاد بگیرند. این در حالی است که RBF بر پایه یک مفهوم هسته (kernel) ساخته شده که در بسیاری از موارد توانایی کمتری برای مدلسازی چنین تعاملاتی دارد.
* معماری Dense می‌تواند لایه‌های متعددی داشته باشد(همانطور که در این مسئله یک لایه بیشتر از RBF دارد) که در هر لایه ویژگی‌های سطح بالا را از داده‌ها استخراج می‌کند. RBF معمولاً این عمق را ندارد و در بسیاری از موارد تنها یک لایه با هسته‌های گوسی یا مشابه دارد.
* واحدهای RBF به شدت متکی به فاصله (norm) از مراکز خاصی هستند و اغلب در یادگیری الگوهای غیرخطی پیچیده در ابعاد بالا مشکل دارند. این موضوع زمانی که داده‌ها به طور کلی پراکنده یا دارای تعاملات غیرمحلی هستند، محدودیت بزرگی ایجاد می‌کند.
* لایه‌های Dense ویژگی‌های جهانی را از داده‌ها استخراج می‌کنند و کمتر به فاصله یا موقعیت خاص در فضای ورودی محدود می‌شوند.
* Dense شبکه‌ها با داشتن توابع فعال‌سازی غیرخطی مانند ReLU، Sigmoid یا Tanh می‌توانند پیچیدگی بیشتری در یادگیری الگوهای داده‌ها داشته باشند.
* تعیین تعداد مناسب گره‌های RBF (مراکز هسته‌ها) بسیار مهم است و انتخاب نادرست می‌تواند به Underfitting یا Overfitting منجر شود.
* شبکه‌های Dense مقیاس‌پذیری بیشتری دارند و می‌توان به راحتی آن‌ها را با تغییر معماری برای داده‌های متفاوت تنظیم کرد.

## [کد پرسش چهارم](https://colab.research.google.com/drive/1DJLdcVGMyO4_Bo0DiP9ItwC-eCaY-euP?usp=sharing)