گزارش تمرین یک

مهرانه مقتدائی فر 97222086

برای پیاده سازی شبکه عصبی مورد نظر در ابتدا باید داده ها را پردازش کرده به طوریکه برای شبکه قابل فهم باشد و سپس به سراغ آموزش دادن شبکه توسط داده های train رفته و سپس ارزیابی آن را بر روی داده های test امتحان میکنیم.

1. پردازش داده ها:

در این مرحله ابتدا فایل csv مورد نظر را باز کرده و به داده ها یک نگاه کلی می اندازیم، سپس با دستور مورد نظر میبینیم که 10000 سطر و 13 ستون داریم، ازین 13 ستون دو داده به نام های CreditId , surname داریم که در نتیجه ما تاثیری ندارند پس آنها را از جدول حذف میکنیم(11 ستون داریم)

حال با توجه به ویژگی های کلی جدول متوجه میشویم که 2 ستون یعنی Geography, Gender داده های حروفی (Categorical) هستند و بقیه داده ها عددی (Numerical) هستند.

همچنین از این 13 ستون، یک ستون نهایی ما (Exited) داده خروجی (Label) است و بقیه ستون ها را به عنوان داده های ورودی و یا همان ویژگی ها (Features) در نظر میگیریم.

حال باید داده های Categorical را طوری تغییر دهیم که شبکه متوجه آنها بشود، در اینجا از one hot encoding استفاده کردیم به گونه ای که ستون های Gender , Geography به 5 ستون تبدیل می شوند(3 تا برای Geography و 2تا برای Gender) حال داده هایی که در این 5 ستون هستند را به 0 و 1 تبدیل میکند (0 به معنای False و 1 به معنای True)
و تعداد ستون های ما از 11 تا به 14تا تغییر میکند (2 ستون حذف و 5 ستون اضافه شده)

حال وقت آن است که داده های ویژگی را به دو بخش train , test تقسیم کنیم. 0.2 داده ها را test و بقیه را train میگیریم.

سپس برای آنکه شبکه بهتر عمل کند و سرعت همگرایی ما بیشتر باشد، داده های موجود را scale میکنیم. در اینجا از Standard Scaler استفاده کردیم و با دستور نوشته شده داده های train, test به مقیاس مورد نظر تبدیل میشوند.

2. ساختن و آموزش دادن شبکه عصبی:

در این مرحله برای ساختن شبکه ابتدا لایه ها را چیده و سپس مدل را میسازیم.
ابتدا یک نقطه شروع برای وزن ها در نظر گرفتم که شبکه تکرار پذیر باشد.
سپس هایپر پارامترها را قرار میدهیم و شبکه را آموزش میدهیم.
در چندین مرحله با تغییر هایپر پارامتر ها، نتایج مختلف گرفته تا به بهترین نتیجه برسیم.
در چندین مدل این نتایج را باهم میبینیم.

مدل 1:

شبکه با سه لایه

Dense: 100, 20, 2

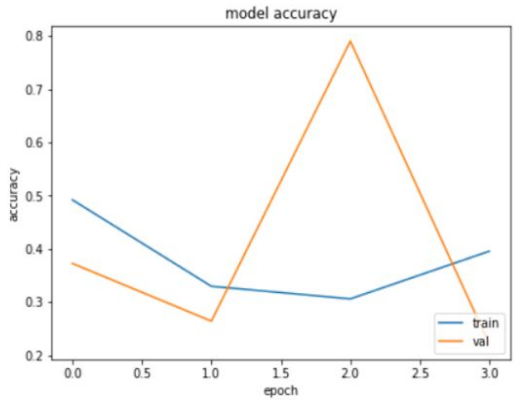
Activation function: relu,relu, softmax

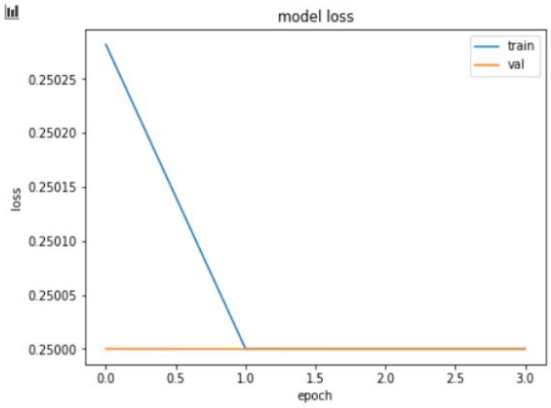
Without Dropout

Optimizer: Adam, Learning Rate: 0.01

Loss: Mean Squared Error

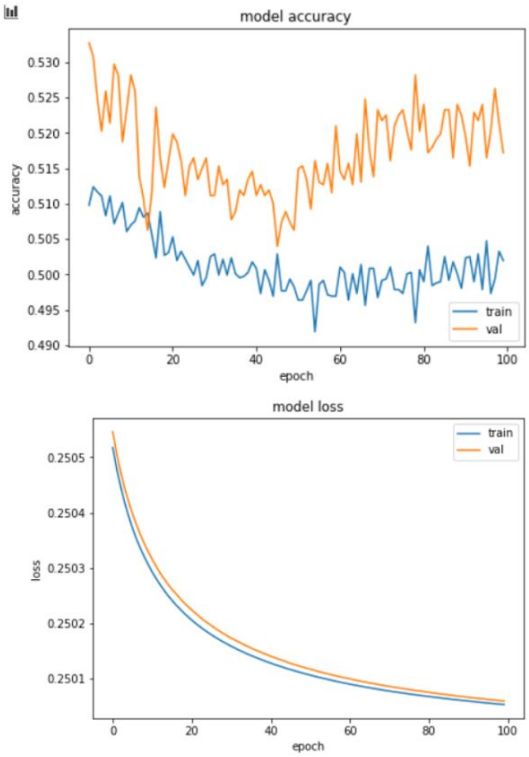
Batch_size = 60, epochs = 100, validation_split = 0.33





همانطور که میبینیم مدل مورد نظر اصلا نتیجه خوبی ندارد و به overfitting خوردیم که برای جلوگیری از آن از early stopping استفاده کرده ام این نتیجه به دلیل استفاده از Adam رخ داده و در مراحل بعدی آن را تغییر میدهیم.

مدل 2:



شبکه با سه لایه

Dense: 100, 20, 2

Activation function: relu,relu, softmax

Without Dropout

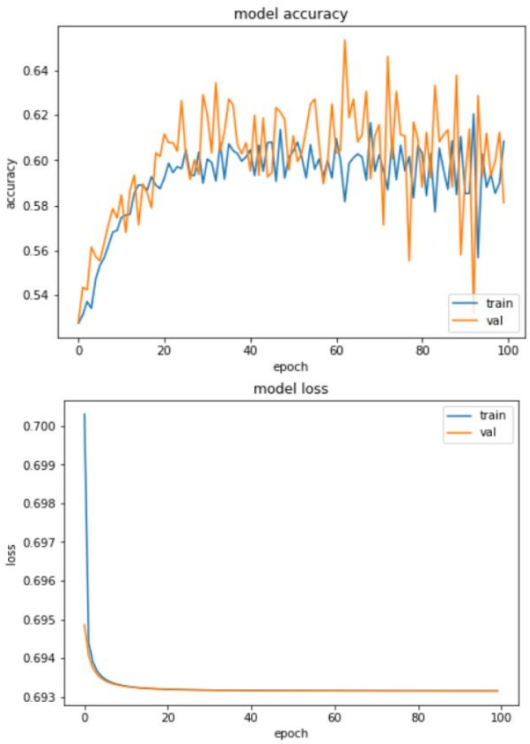
Optimizer: SGD , Learning Rate: 0.01

Loss: Mean Squared Error

Batch_size = 60, epochs = 100, validation_split = 0.33

دقیقا همان مشخصات قبلی را داریم با این تفاوت که این بار از SGD استفاده کرده ایم. مشاهده میشود که دیگر overfitting رخ نداده و حتی دقت بر روی داده های تست از 20% به 50 % رسیده است.

مدل 3:



شبکه با سه لایه

Dense: 100, 10, 2

Activation function: relu,relu, softmax

Without Dropout

Optimizer: SGD , Learning Rate: 0.01

Loss: binary cross entropy

Batch_size = 60, epochs = 100, validation_split = 0.33

همانطور که مشاهده میشود با تغییر برخی ویژگی ها روند یادگیری شبکه بهتر شده و مقدار accuracy افزایش و مقدار loss کاهش یافته است.

دقت در داده های تست: 57%

مدل 4:

شبکه با چهار لایه

Dense: 100, 10, 8, 2

Activation function: relu,relu,relu, softmax

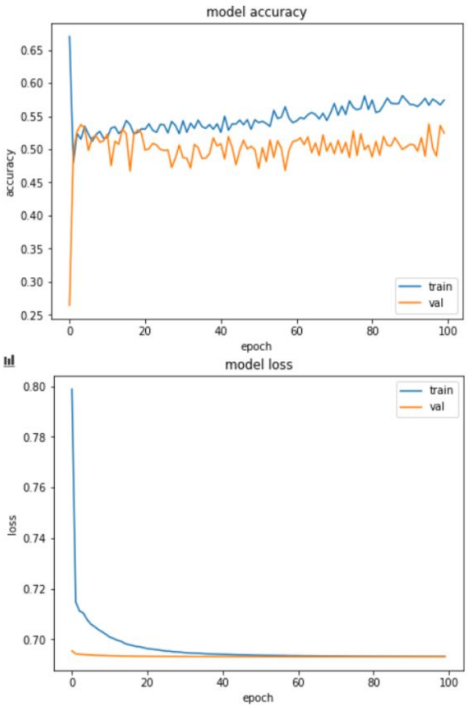
With dropout 0.2

Optimizer: SGD , Learning Rate: 0.001, Momentum = 0.95

Loss: binary cross entropy

Batch_size = 60, epochs = 100, validation_split = 0.33

در این مرحله یک لایه به شبکه اضافه کرده ایم، تغییر زیادی در خروجی مشاهده نمیشود اما بهتر است هما سه لایه را حفظ کنیم و در عوض عمق لایه ها را زیادتر کنیم تا ببینیم نتیجه چه میشود.
دقت در داده های تست: 51%



مدل 5:

شبکه با سه لایه

Dense: 200, 50, 2

Activation function: relu,relu و softmax

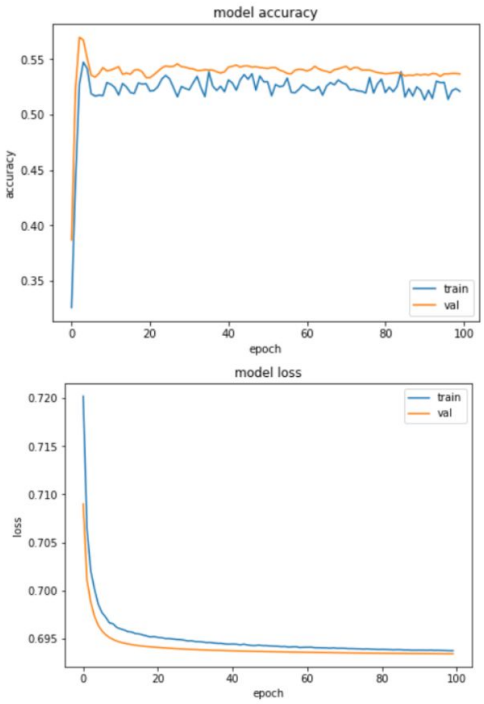
With dropout 0.2

Optimizer: SGD , Learning Rate: 0.001, Momentum = 0.95

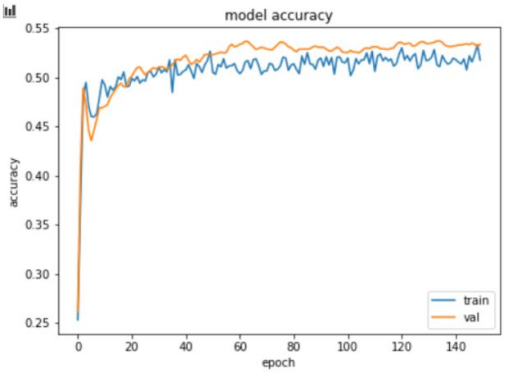
Loss: binary cross entropy

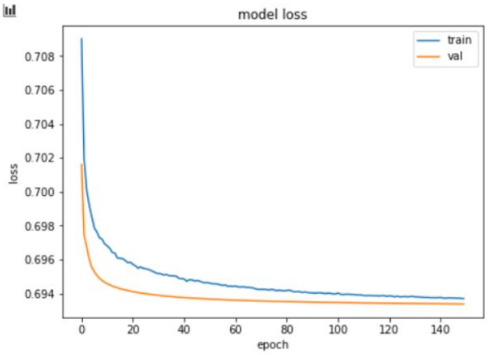
Batch_size = 200, epochs = 100, validation_split = 0.33

دوباره تعداده لایه ها را به سه کاهش دادیم، با تغییر عمق لایه ها و تعداد batch ها مشاهده میشود که نتیجه نهایی بهتر بوده و عملکرد شبکه بهتر شده است. اما دقت در داده های تست تغییر نیافته: 51%



مدل 6:





شبکه با سه لایه

Dense: 350, 70, 2

Activation function: relu,relu و softmax

With dropout 0.2

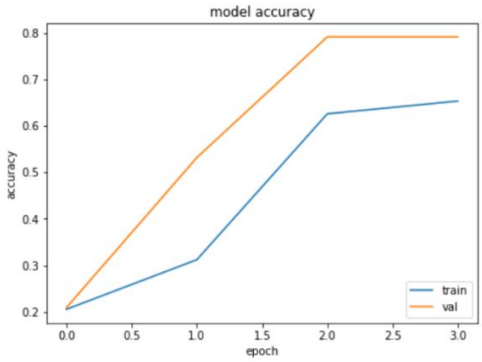
Optimizer: SGD , Learning Rate: 0.001, Momentum = 0.95

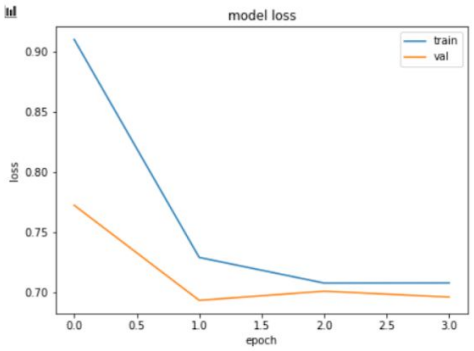
Loss: binary cross entropy

Batch_size = 250, epochs = 150, validation_split = 0.33

با زیاد کردن عمق لایه ها و همچنین زیاد کردن سایز batch ها و تعداد epoch ها مشاهده میکنیم که عملکرد شبکه باز هم بهتر شده و حتی بر روی داده های تست نیز بهتر کار میکند.

مدل 7:





شبکه با سه لایه

Dense: 150, 10, 2

Activation function: sigmoid,sigmoid, softmax

With dropout 0.2

Optimizer: SGD , Learning Rate: 0.001, Momentum = 0.95

Loss: binary cross entropy

Batch_size = 250, epochs = 150, validation_split = 0.33

با تغییر activation function به sigmoid نتایج جالبی بدست می آوریم.اول که تابع به سرعت overfit میشود که این مساله با early stopping کنترل شده است.(در epoch 4 متوقف میشود) همانطور که مشاهده میشود مقدار accuracy , loss نیز به نظر خوب می آید و همچنین دقت در داده های تست به 80% رسیده!!

مدل 8:

شبکه با سه لایه

Dense: 100, 10, 2

Activation function: relu,relu, softmax

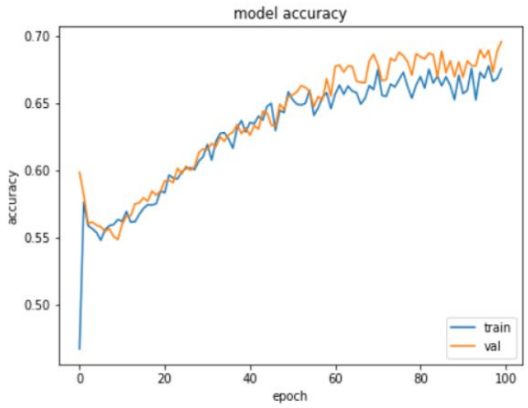
With dropout 0.2

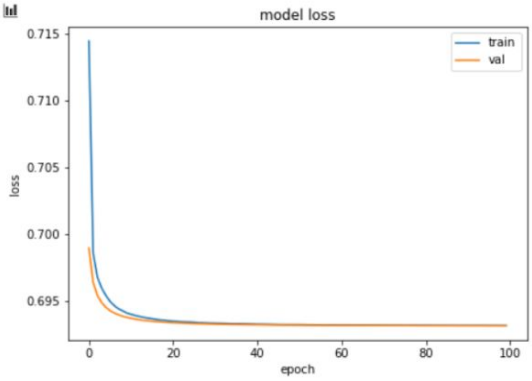
Optimizer: SGD , Learning Rate: 0.001,
Momentum = 0.95

Loss: binary cross entropy

Batch_size = 60, epochs = 100, validation_split = 0.33

با اضافه کردن لایه dropout و اضافه کردن momentum به مشخصات SGD مشاهده میشود که عملکرد شبکه بر روی داده های train بیشتر شده، همچنین Overfitting رخ نداده و دقت بر روی داده های تست به 69% رسیده است.





3.نتیجه گیری:

مدل 8 بهترین مدلی است که برای شبکه مورد نظر بدست آوردم.(وزن های این مدل در فایل hdf5 ذخیره شده و ارسال شده است) دقت بر روی داده های تست در این حالت نزدیک به 70% است

در این حالت overfitting رخ نداده و شبکه زود متوقف نمیشود.

برای داده ها مورد نظر مشاهده کردم که Adam optimizer به خوبی عمل نمیکند و در برخی موارد حتی شبکه آموزش نمیبیند شاید به دلیل نوع encode و scale کردن داده ها باشد.

همچنین با زیاد کردن عمق شبکه در برخی موارد نتایج خوب و در برخی موارد نتایج بدی بدست آمد اما در بهترین حالت بدست آمده عمق شبکه زیاد نیست.

هرچه learning rate کمتر و momentum بیشتر باشد مشاهده میشود که عملکرد بر روی داده ها بهتر میشود

اضافه کردن Dropout به لایه ها باعث جلوگیری از overfitting و همچنین بهبود عملکرد شبکه میشود.

همچنین استفاده از تابع فعالسازی sigmoid دقت بر روی داده های تست را بسیار زیاد کرد اما شبکه زود متوقف شد و خروجی های پیشبینی شده بر روی داده های تست تمامی 0 بود و به نظر این روش درست نمی آید.

طبق مشاهدات تعداد batch ها در بین 50 تا 100 نتیجه خوبی داشت و همچنین تعداد epoch ها بهتر است بین 100 تا 200 بماند.