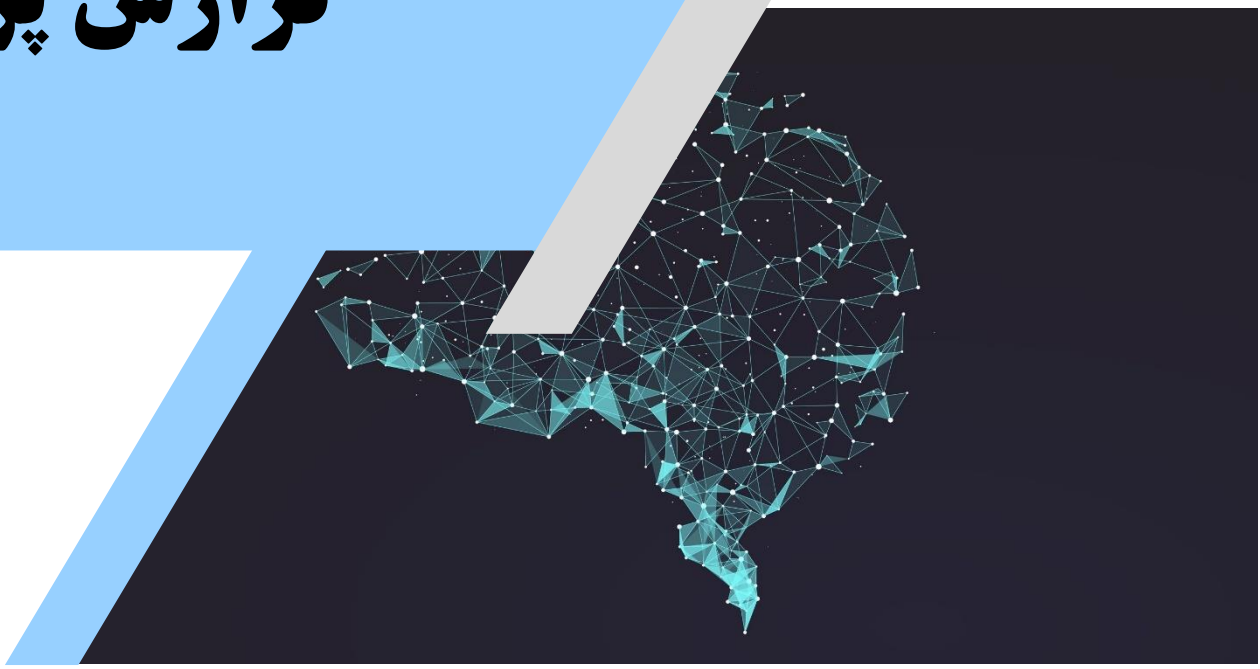
هوش محاسباتی

شبکه عصبی

گزارش پروژه



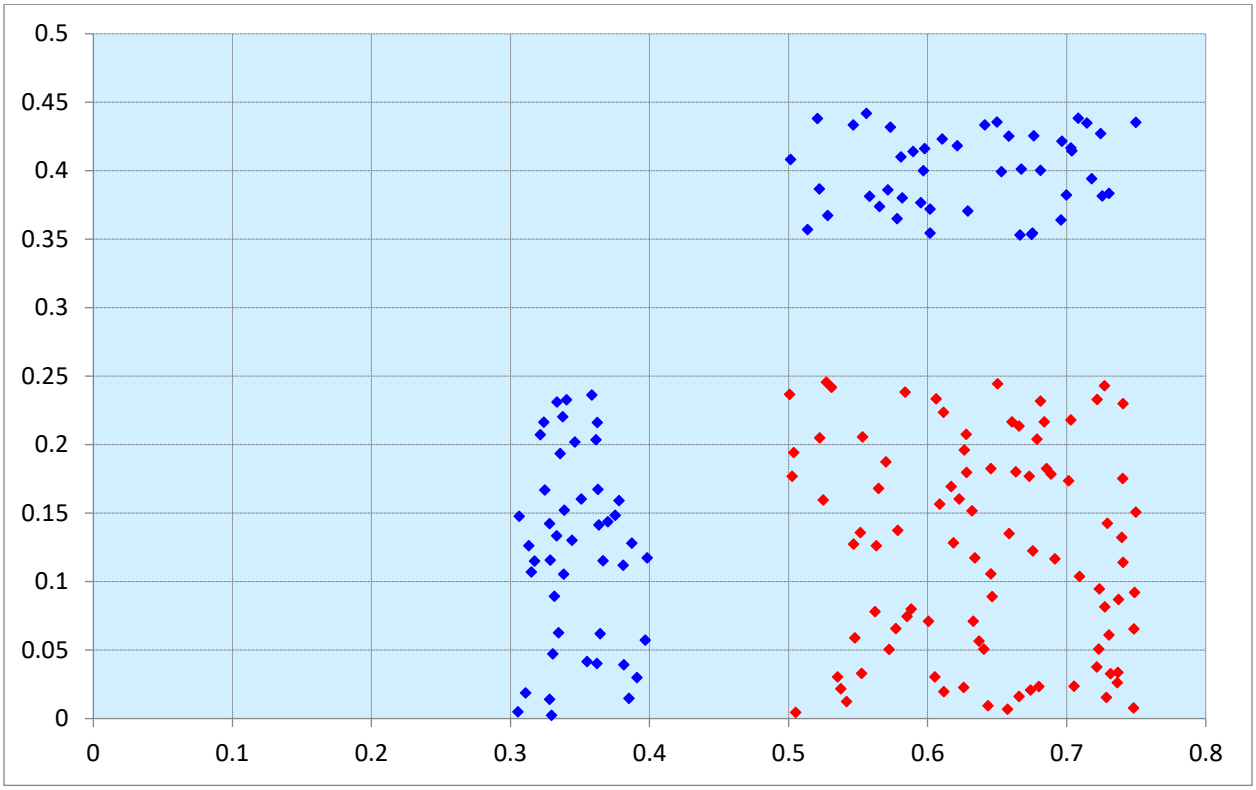
محمدرضا اخگری زیری

۹۶۳۱۰۰۱

سوال یک:

برای کشیدن نموار scatter از نرم‌افزار excel استفاده کردیم. برای ایجاد نمودار داده‌ها را انتخاب می‌کنیم و از تب insert، گزینه scatter را انتخاب می‌کنیم. برای رنگ‌آمیزی نمودار، ماکرویی در برنامه excel نوشتیم که فایل آن همراه تمرین آپلود شده است(Scatterplot.bas) که می‌توانید با texteditor یا ide باز کنید. (برای نوشتن macro در excel لازم است که تب developer را اضافه کنیم)

نمودار کشیده شده برای داده‌ها به شکل زیر است:



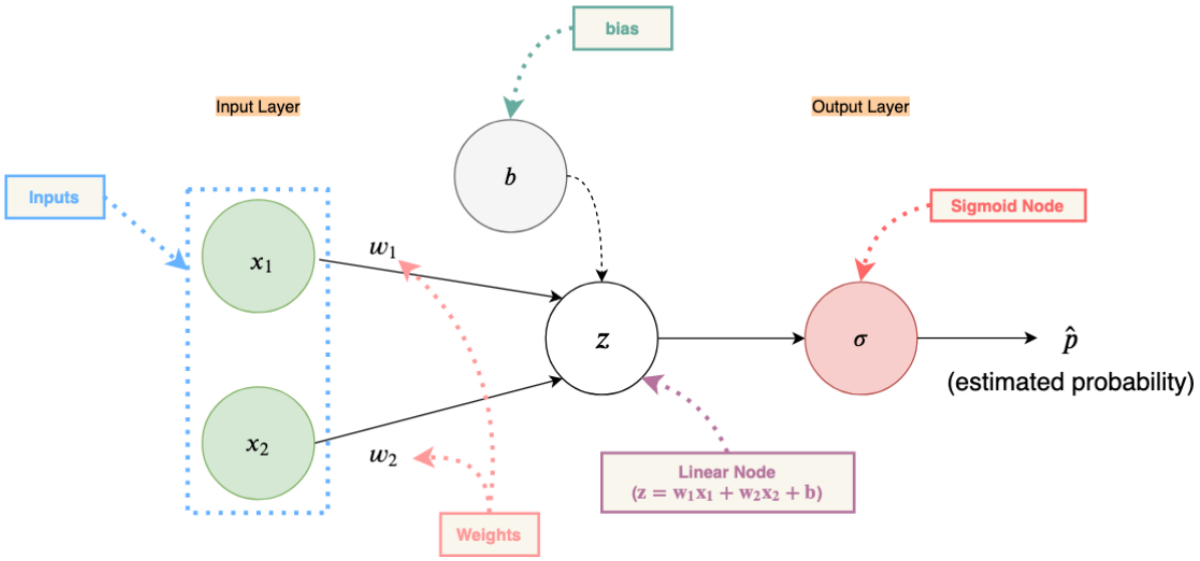
محور افقی داده‌ها برروی x1 هستند و محور عمودی مربوط به x2 است.

داده‌های به رنگ آبی دارای برچسب ۱ و باقی داده‌ها (قرمز) دارای برچسب ۰ می‌باشند.

داده‌ها را با نسبت ۳ به ۷ تقسیم می‌کنیم به دو دسته آزمایش و آموزش.

سوال یک:

برای فهم بهتر سوال و راحتی در پیاده‌سازی، شبکه داده‌شده در سوال را باز می‌کنیم:



ورودی ها: x_1 و x_2 گره های ورودی برای دو ویژگی هستند که نمونه ای از آن هستند که ما می خواهیم شبکه عصبی ما از آن یاد بگیرد. از آنجا که گره های ورودی اولین لایه شبکه را تشکیل می دهند ، در مجموع به "لایه ورودی"[1] گفته می شود.

وزن: w_1 و w_2 به ترتیب مقادیر وزنی را که ما به ترتیب با ورودی های x_1 و x_2 نشان می دهیم نشان می دهد. وزنها تأثیر هر ورودی را در محاسبه گره بعدی کنترل می کنند. یک شبکه عصبی برای تهیه پیش بینی های دقیق ، این وزن ها را "یاد می گیرد". در ابتدا ، وزنها به طور تصادفی تعیین می شوند.

گره خطی (z): گره "z" یک تابع خطی از تمام ورودی های ورودی به آن ایجاد می کند یعنی:

$$z = w_1 x_1 + w_2 x_2 + b$$

بایاس: "b" گره بایاس را نشان می دهد. گره بایاس مقدار افزودنی را به گره تابع خطی (z) وارد می کند. بایاس خروجی را به گونه‌ای نشان می دهد که ممکن است بهتر با خروجی مورد نظر ما هماهنگ شود. مقدار بایاس با b = 0 آغاز می شود و در مرحله آموزش نیز آموخته می شود.

Sigmoid Node: این گره σ ، با نام گره Sigmoid ، ورودی را از یک گره خطی قبلی (z) می گیرد و آن را از طریق عملکرد فعال سازی عبور می دهد ، به نام تابع Sigmoid (به دلیل منحنی S شکل آن) .

[1] Input layer

مقادیر مشتق:

از گراف باز شده بالا حساب می‌کنیم:

$$\frac{dCost}{db} = \frac{dCost}{dSigm} * \frac{dSigm}{dZ} * \frac{dZ}{db} = \left(-1 * \left(\frac{\hat{Y}}{y}\right) + \left(\frac{\left(1-\hat{Y}\right)}{1-y}\right)\right) * S(z)\left(1 - S(z)\right) * 1$$

$$= \Sigma_{\forall_x}\left[y - \hat{Y}\right]$$

$$\frac{dCost}{dW} = \frac{dCost}{dsigm} * \frac{dSigm}{dZ} * \frac{dZ}{dW} = \Sigma_{\forall_x}\left[y - \hat{Y}\right] * X$$

سوال سه:

برای پیدا کردن نرخ یادگیری مناسب، عملیات را برای گام‌های تصادفی اجرا می‌کنیم و بازه‌ی نرخ یادگیری را محدود می‌کنیم.

برای این کار در ابتدا، عبارت را اجرا می‌کنیم:

```
lr = 10 ** random.uniform(-3,3)
```

مشاهده می‌کنیم که بهترین نتیجه در بازی ۰ تا ۱۰ اتفاق می‌افتد، پس دوباه کد را با مقادیر جدید اجرا می‌کنیم.

پس از اجرا مقدار نرخ یادگیری ۱.۱ — ۱.۲ بهترین نتیجه را داشت.

برای تعریف بهترین نتیجه می‌توان معیار دقت را در نظر گرفت، از آنجایی که داده‌ها بالانس هستند (تعداد یک‌ها با صفرها برابر بود)

برای تعداد گام هم برنامه را برای تعدادگام‌های متفاوت اجرا کردم (با نرخ یادگیری مورد قبلی)

گام ۵ : دقت ۶۶ درصد

گام ۷: دقت ۸۱ درصد

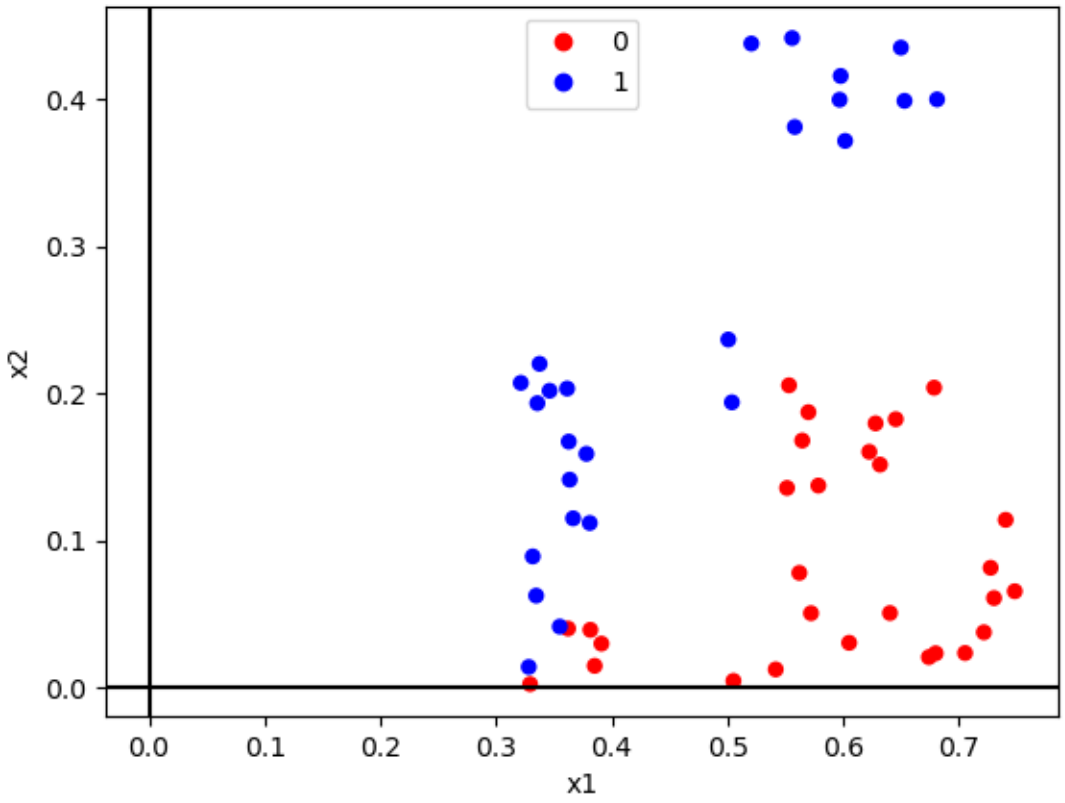
گام ۱۰: ۸۳ درصد

گام‌های بیشتر، دور همین عدد میگردند، که نشان می‌دهد به گام خوبی رسیدیم.

بدیهی است که برای مقادیر تصادفی این اعداد فرق خواهند کرد.

سوال چهار:

```
index is: 1 with cost 19.000020035221162 and accuracy_rate 0.8703703703703703 -> lr 1.2
```



سوال پنج:

مقادیر را برای این سوال نیز حساب می‌کنیم:

$$\frac{dCost}{dW} = \frac{dCost}{dy} \times \frac{dy}{dW} = 2\left(y - \hat{Y}\right) \times y(1-y)u_0 z_0(1-z_0)X$$

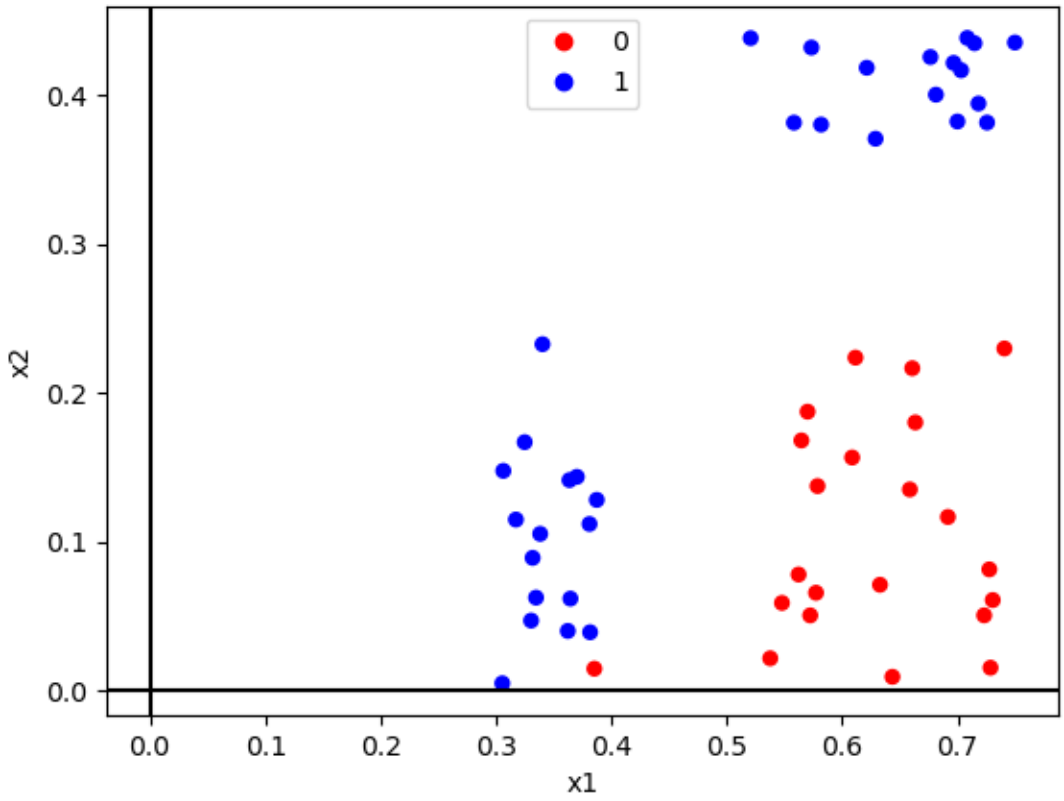
$$\frac{dCost}{dV} = \frac{dCost}{dy} \times \frac{dy}{dV} = 2\left(y - \hat{Y}\right)y(1-y)u_1 z_1(1-z_1)X$$

$$\frac{dCost}{dU} = \frac{dCost}{dy} \times \frac{dy}{dU} = 2\left(y - \hat{Y}\right)y(1-y)Z$$

$$\frac{dCost}{db_0} = \frac{dCost}{dy} \times \frac{dy}{db_0} = 2\left(y - \hat{Y}\right)y(1-y)u_0 z_0(1-z_0)$$

$$\frac{dCost}{db_1} = \frac{dCost}{dy} \times \frac{dy}{db_1} = 2\left(y - \hat{Y}\right)y(1-y)u_1 z_1(1-z_1)$$

$$\frac{dCost}{db_2} = \frac{dCost}{dy} \times \frac{dy}{db_2} = 2\left(y - \hat{Y}\right)y$$




```
index is: 1 with cost 6.99535215510283 and accuracy_rate 0.9814814814814815 -> lr 3
```

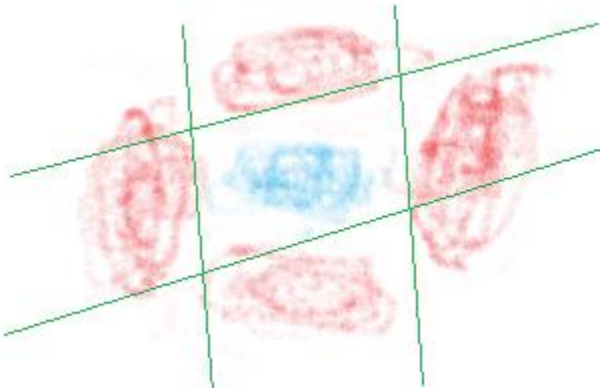
برای این سوال نرخ را بر روی ۳ و تعداد گام ها را ۴۰ قرار می‌دهیم.

سوال شش:

کارای این شبکه نسبت به حالت قبل بهتر شده و دقت بالاتری به دست آمده زیرا با استفاده از چند لایه و تعداد نرون بیشتر مدل غیرخطی توانستیم طراحی کنیم در حالی که در حالت قبل با یک نرون یک مرز تصمیم گیری خطی می توانستیم طراحی کنیم.

سوال هفت:

میتوان شکل سوال را اینگونه با خط مشخص کرد.



به وضوح مشخص است که شکل بالا خطی نیست، می‌توان هرخط را به یک نورون مربوط دانست و هر نورون تصمیم می‌گیرد، که این نقطه در کدام جهت خط خودش است، و نتیجه این چهار نورون را می‌توان به نورون دیگری داد و از نتیجه آن فهمید در کدام ناحیه است. شکل کلی به صورت زیر است: