مریم رضوانی
شماره دانشجویی: ۹۹۲۱۱۶۰۰۱۹

سؤال دوم پیاده سازی:

موضوع: طراحی الگوریتم پرسپترون برای حل مسأله OR

مقدمه: الگوریتم پرسپترون یک الگوریتم شبکه عصبی است.این الگوریتم با تغییر وزن ها و بایاس شبکه، سعی می‌کند که خطای پیش بینی شبکه(loss function) را کمتر کند. الگوریتم پرسپترون به این صورت عمل می‌کند:

- ابتدا، وزن ها و بایاس شبکه را به صورت تصادفی مقداردهی می کند.
- سپس، برای هر نقطه داده در مجموعه آموزش، شبکه را با ورودی های آن فعال می کند و خروجی پیش بینی شده را با استفاده از تابع فعال سازی (مثلا تابع قدم) محاسبه می کند.
- بعد، خروجی پیش بینی شده را با خروجی صحیح (برچسب) مقایسه می کند و خطای پیش بینی را به دست می آورد.
- در نهایت، وزن ها و بایاس شبکه را به مقدار کوچکی که با نرخ یادگیری و خطا ضرب شده است، تغییر می دهد.

این فرآیند را برای تمام نقاط داده در مجموعه آموزش تکرار می کند و این را به عنوان یک دوره (epoch) می نامد. با تکرار چندین دوره، الگوریتم پرسپترون قادر می شود که شبکه را به گونه ای آموزش دهد که خطای پیش بینی را کمینه کند.

حل مسأله طراحی تابع OR با استفاده از الگوریتم پرسپترون:

الگوریتم پرسپترون تابع or را با استفاده از یک فرآیند تکراری یاد می‌گیرد. در هر مرحله، شبکه پرسپترون یک ورودی را دریافت می‌کند و با ضرب داخلی آن در وزن‌ها و اضافه کردن بایاس، یک خروجی پیش‌بینی می‌کند. سپس، با مقایسه این خروجی با برچسب صحیح، خطای پیش بینی را محاسبه می‌کند. اگر خطا صفر باشد، یعنی شبکه پاسخ درست را داده است و نیاز به تغییر وزن ها و بایاس ندارد. اما اگر خطا غیر صفر باشد، یعنی شبکه پاسخ اشتباه را داده است و باید وزن‌ها و بایاس را به مقدار کوچکی که با نرخ یادگیری (learning_rate) و خطا(error) ضرب شده است، تغییر دهد. این تغییرات باعث می شوند که شبکه در دفعات بعدی به سمت پاسخ صحیح حرکت کند. الگوریتم پرسپترون این فرآیند را برای تمام نقاط داده در مجموعه آموزش تکرار می کند و این را به عنوان یک دوره (epoch) می نامد. با تکرار چندین دوره، شبکه پرسپترون قادر می شود که تابع or را با دقت بالایی یاد بگیرد.

پیاده‌سازی طراحی تابع OR با روش پرسپترون:

پیاده‌سازی تابع OR با روش پرسپترون به این صورت است که یک شبکه عصبی ساده با یک لایه و یک نورون را ایجاد می کنیم. این شبکه دو ورودی بولین (صفر یا یک) را دریافت می‌کند و خروجی آن را بر اساس تابع OR مشخص می‌کند.
برای مثال، اگر ورودی ها 0 و 1 باشند، خروجی باید 1 باشد. برای یادگیری پرسپترون، ما از چهار جفت ورودی و خروجی ممکن برای تابع OR استفاده می کنیم.

```
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 1, 1, 1]
```

این جفت ها را به عنوان داده های آموزش به شبکه می‌دهیم و از روش پرسپترون برای آپدیت وزن ها و بایاس شبکه استفاده می‌کنیم. روش پرسپترون بر اساس تفاضل بین خروجی پیش بینی شده و خروجی صحیح، وزن ها و بایاس را تغییر می‌دهد تا خطای پیش بینی را کمتر کند. این فرآیند را برای تعداد مشخصی از دوره ها (epoch) تکرار می‌کنیم تا شبکه قادر باشد که تابع OR را با دقت بالایی شبیه سازی کند.

توضیح برنامه پیاده‌سازی این تابع:

کلاسPerceptron یک شبکه عصبی ساده با یک لایه و یک نورون را پیاده سازی می کند. این کلاس شامل توابع زیر است:

- __init__(self, input_size, learning_rate=0.1): این تابع سازنده کلاس است که ورودی های زیر را می گیرد
- input_size: تعداد ورودی های پرسپترون، که در اینجا دو است.
- learning_rate: نرخ یادگیری پرسپترون، که به صورت پیش فرض 0.1مقدار دادیم.

این تابع وزن‌ها و بایاس شبکه را به صورت رندوم مقداردهی می‌کند و آن‌ها را به عنوان خصوصیات کلاس ذخیره می‌کند.

- activation(self, x): این تابع فعال سازی است که تابع ***step function*** را پیاده سازی می کند. این تابع یک عدد x را دریافت می کند و اگر x بزرگتر یا مساوی صفر باشد، 1 را برمیگرداند و در غیر این صورت، 0 را برمی گرداند.
- predict(self, inputs): این تابع خروجی پیش بینی شده پرسپترون را برای یک ورودی داده شده محاسبه می‌کند. این تابع آرایه ای از ورودی های شبکه را دریافت می‌کند و با ضرب داخلی آن در وزن‌ها و اضافه کردن بایاس، یک عدد z را به دست می آورد. سپس، با استفاده از تابع فعال سازی، 1 یا 0 را به عنوان خروجی برمی‌گرداند.
- train(self, training_inputs, labels, epochs): این تابع پرسپترون را با استفاده از الگوریتم پرسپترون آموزش می‌دهد. این تابع ورودی های زیر را می‌گیرد:
  - training_inputs: لیست آرایه های ورودی های شبکه برای مجموعه آموزش.
  - labels: لیست خروجی های صحیح (برچسب ها) برای مجموعه آموزش.
  - epochs: تعداد دوره های (epoch) آموزش.

این تابع برای هر دوره، برای هر جفت ورودی و خروجي، خروجي پيش بيني شده شبکه را محاسبه می‌کند و با مقایسه آن با خروجي واقعی ، خطای پیش بینی را به دست می‌آورد. سپس، با استفاده از قانون یادگیری پرسپترون، وزن‌ها و بایاس شبکه را به مقدار کوچکي که با نرخ یادگیری و خطا ضرب شده است، تغییر می‌دهد.

نتیجه: این تغییرات باعث می‌شوند که شبکه در دفعات بعدی به سمت پاسخ واقعی نزدیک‌تر شود. این تابع همچنین خطا و وزن ها و بایاس شبکه را در هر مرحله چاپ می‌کند.