دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر

مبانی هوش محاسباتی
تمرین پیاده‌سازی شبکه‌های عصبی
(Fruits Classification)

استاد درس: دکتر عبادزاده

پاییز ۱۴۰۰

فهرست

مقدمه

یکی از کاربردهای شبکه‌های عصبی، Classification می‌باشد. در این پروژه قصد داریم سراغ مسئله کلاس‌بندی میوه‌ها برویم. ما در اینجا می‌خواهیم به کمک شبکه‌های عصبی Feedforward Fully Connected که در درس با آن آشنا شدیم، این مسئله رو حل کنیم.

شرح مسئله

در این مسئله، ورودی ما چهار میوه‌ای که در شکل زیر مشاهده می‌کنید، می‌باشد، و مدل ما باید این میوه ها را تشخیص دهد.

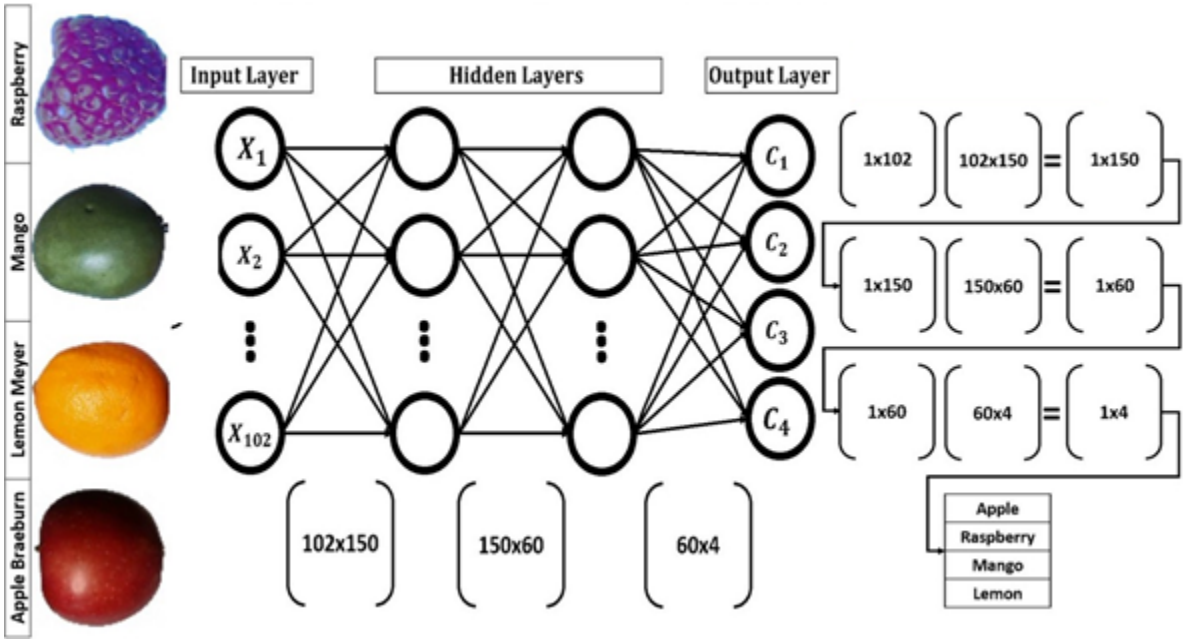
دیتاست‌ای که در این پروژه از آن استفاده می‌کنیم، دیتاست Fruit-360 می‌باشد. این دیتاست شامل تعداد زیادی از انواع میوه‌ها می‌باشد، که تصاویر هر نوع بصورت ۳۶۰ درجه تهیه شده است. اما در این پروژه برای تسهیل کار و افزایش دقت الگوریتم، تنها چهار کلاس میوه بالا را انتخاب و در ادامه بر روی آن‌ها کار خواهیم کرد.

در این دیتاست، هر کلاس حدود ۴۹۱ تصویر در ابعاد ۱۰۰ در ۱۰۰ دارد. بنابراین، اگر بخواهیم پیکسل های تصاویر را بصورت مستقیم ورودی شبکه عصبی خود در نظر بگیریم، لایه ورودی ما ۱۰۰۰۰ نورون خواهد داشت که شبکه را بسیار سنگین می‌کند. برای رفع این مشکل، در ابتدا با استفاده از تکنیک های استخراج ویژگی، تعداد ۳۶۰ ویژگی استخراج شده است. سپس با استفاده از تکنیک های کاهش سایز بردار ویژگی، این سایز به ۱۰۲ کاهش پیدا کرده که آن ها را به عنوان ورودی شبکه عصبی درنظر می‌گیریم.

با توجه به اینکه مدل ما قرار است در نهایت یکی از این چهار نوع میوه را تشخیص دهد، لایه خروجی ما دارای ۴ نورون است که به ترتیب نمایانگر میوه‌ها هستند و آن نورونی که بیشتری مقدار Activation را دارد، به عنوان میوه تشخیص داده شده، انتخاب می‌شود.

همچنین در این مسئله از دو لایه پنهان (Hidden Layer)، لایه اول با ۱۵۰ نورون و لایه دوم با ۶۰ نورون (عملکرد این مقادیر تست شده اند) و تابع سیگموئید(sigmoid) به عنوان Activation Function استفاده می‌کنیم.

بنابراین، ساختار شبکه عصبی به صورت زیر خواهد بود:



شبه‌کد

شبه‌کد فرآیند یادگیری شبکه عصبی ما طبق روش Stochastic Gradient Descent، به شکل زیر می‌باشد:

```
Allocate W matrix and vector b for each layer.
Initialize W from standard normal distribution, and b = 0, for each layer.
Set learning_rate, number_of_epochs, and batch_size.
for i from 0 to number_of_epochs:
    Shuffle the train set.
    for each batch in train set:
        Allocate grad_W matrix and vector grad_b for each layer and initialize to 0.
        for each image in batch:
            Compute the output for this image.
            grad_W += dcost/dW for each layer (using backpropagation)
            grad_b += dcost/db for each layer (using backpropagation)
        W = W - (learning_rate × (grad_W / batch_size))
        b = b - (learning_rate × (grad_b / batch_size))
```

ایده‌ی این روش اینه که به جای اینکه در هر مرحله از یادگیری مدل، بیایم و با کل داده‌های مجموعه Train کار کنیم، می‌تونیم در هر پیمایش، داده‌ها رو به بخش‌هایی تحت عنوان mini-batch تقسیم کنیم، گرادیان مربوط به هر سمپلِ اون mini-batch بدست بیاریم، و در نهایت، میانگین اونها رو بدست بیاریم، و بعد تغییرات رو اعمال کنیم. این کار باعث میشه که محاسبات در هر پیمایش کمتر بشه و زمان یادگیری مدل ما، کاهش پیدا کنه.

تعداد سمپل‌هایی که هر مرحله باهاشون کار می‌کنیم رو بهش میگن batch size. همچنین، به هر دور که تمامی mini-batch ها (و در نتیجه تمامی سمپل‌ها) پیمایش میشن، میگن epoch (بخوانید ای‌پاک!)

قدم اول: دریافت دیتاست

در ابتدا نیاز است دیتاست هایی را که در اختیار شما قرار دادیم، لود کنید. دیتاست ها به فرمت ".pkl" هستند که می‌توانید با استفاده از کتابخانه "Pickle" آن ها را بخوانید.

چهار فایل زیر دیتاست هایی هستند که برای انجام این پروژه (بجز بخش امتیازی) به آن‌ها نیاز دارید.

- فایل "train_set_features.pkl" شامل یک آرایه دو بعدی به طول ۱۹۶۲ (۴۹۲ آرایه سیب و ۴۹۰ تا برای هر یک از سه میوه دیگر که به ترتیب در آرایه قرار گرفته اند) می‌باشد، که در هر آرایه ۳۶۰ داده قرار دارد.
- فایل "test_set_features.pkl" شامل یک آرایه دو بعدی به طول ۶۶۲ (۱۶۴ آرایه سیب و ۱۶۶ تا برای هر یک از سه میوه دیگر که به ترتیب در آرایه قرار گرفته اند) می‌باشد، که در هر آرایه ۳۶۰ داده قرار دارد.
- فایل "train_set_labels.pkl" شامل آرایه‌ای به طول ۱۹۶۲ می‌باشد، که به ترتیب داده های آموزشی با لیبل ۰ تا ۳ (apple، lemon، mango، raspberry) در آن قرار گرفته اند.
- فایل "test_set_labels.pkl" شامل آرایه‌ای به طول ۶۶۲ می‌باشد، که به ترتیب داده های تست با لیبل ۰ تا ۳ (apple، lemon، mango، raspberry) در آن قرار گرفته اند.

نیاز است که فایل های مربوط به ویژگی ها که داده های ورودی ما هستند خوانده و سپس کاهش سایز داده شوند. همچنین باید توجه داشته باشید که این مقادیر باید تقسیم شوند تا مقدار Activation ورودی بین ۰ و ۱ قرار بگیرد.

برای تسهیل در روند لود کردن داده ها، فایل "Loading_Datasets.py" حاوی کد خواندن و آماده سازی دیتاست های آموزش و تست، قرار داده شده است که از آن می‌توانید استفاده کنید.

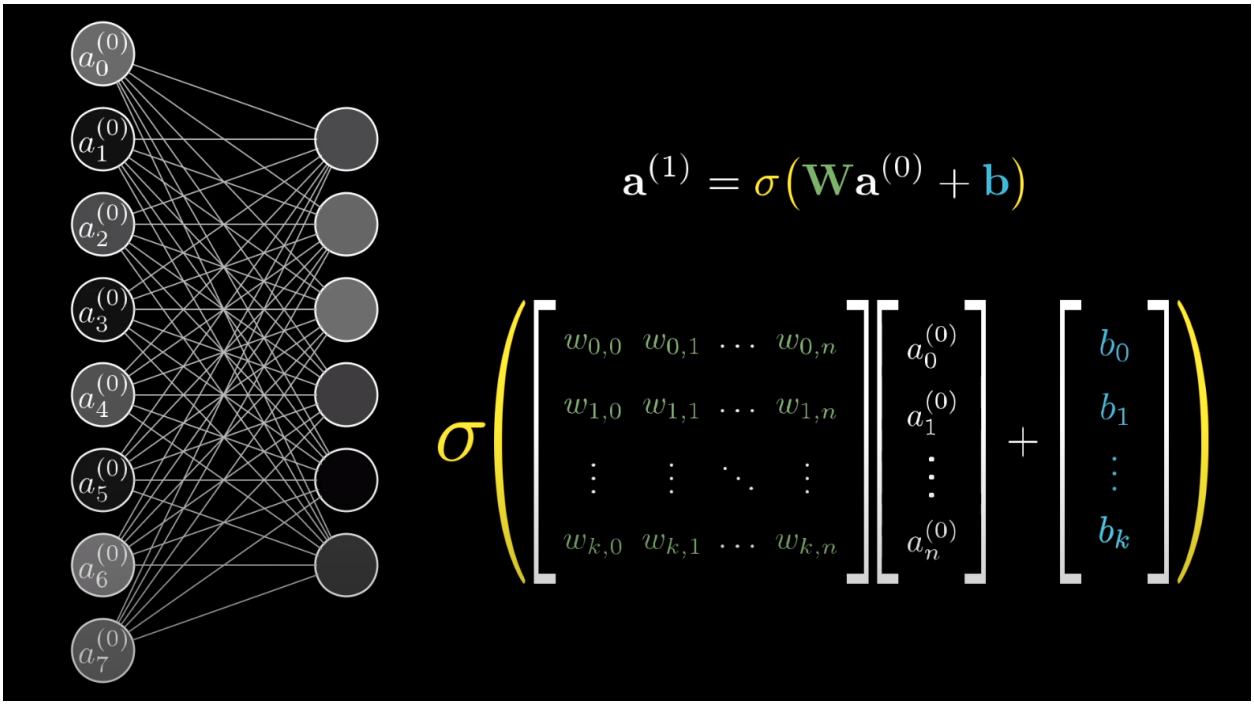
همچنین تصاویر و کد های مربوط به استخراج ویژگی نیز قرار داده شده که برای بخش امتیازی ممکن است به آن‌ها نیاز پیدا کنید.

قدم دوم: محاسبه خروجی (Feedforward)

همونطور که می‌دونید، برای محاسبه‌ی خروجی از روی ورودی در شبکه‌های عصبی، در هر لایه عملیات زیر انجام میشه:

$$a^{(L+1)} = \sigma(W^{(L+1)} \times a^{(L)} + b^{(L+1)})$$

در نتیجه، توی پیاده‌سازی شبکه عصبی، برای وزن‌های بین هر دو لایه، یک ماتریس k در n در نظر می‌گیریم که k، تعداد نورون‌های لایه‌ی بعدی و n، تعداد نورون‌های لایه‌ی فعلی. در نتیجه، هر سطر ماتریس W، وزن‌های مربوط به یک نورون خاص در لایه‌ی بعدی هستش. همچنین، برای بایاس‌های بین هر دو لایه هم، یک بردار جداگانه در نظر می‌گیریم که ابعادش برابر با تعداد نورون‌های لایه بعدی هستش.



در این قدم از پروژه، ۲۰۰ داده (داده های ۲۰۰تا عکس) مجموعه Train رو جدا کنید و پس از مقدار دهی اولیه‌ی ماتریس وزن‌ها با اعداد تصادفی نرمال و بایاس‌ها به صورت بردارهای تماما صفر، خروجی مربوط به این ۲۰۰ داده رو محاسبه کنید. محاسبه خروجی رو باید به طریقی که بالاتر گفتیم (یعنی به صورت ضرب و جمع ماتریسی/برداری و اعمال تابع سیگموئید) انجام بدید. در انتها در لایه آخر، نورونی که بیشترین مقدار را دارد به عنوان تشخیص مدل درنظر گرفته می‌شود که در واقع معادل میوه مربوط به آن نورون می‌باشد.

سپس دقت (Accuracy) مدل؛ یعنی، تعداد عکس‌هایی که به درستی تشخیص داده شده تقسیم بر تعداد کل عکس‌ها، را گزارش کنید. با توجه به اینکه هنوز فرآیند یادگیری طی نشده و مقدارهی‌ها رندوم بوده، انتظار میره دقت در این حالت، به طور میانگین به ۲۵ درصد نزدیک باشد.

نکته: اگر پروژه رو با پایتون انجام می‌دید، حتما برای کار با ماتریس‌ها، از **NumPy** استفاده کنید.

قدم سوم: پیاده‌سازی Backpropagation

همونطور که می‌دونید، فرآیند یادگیری شبکه‌ی عصبی به معنی مینیمم کردن تابع Cost هستش:

$$Cost = \sum_{j=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

که این‌کار به کمک روش Gradient Descent انجام میشه که در اون با بدست آوردن مشتقات جزئی تابع Cost نسبت به تمامی پارامترها (یعنی همان گرادیان)، تغییرات مورد نظر بر روی پارامترها رو انجام می‌دیم:

$$(W,\ b) = (W,\ b) - \alpha \nabla Cost$$

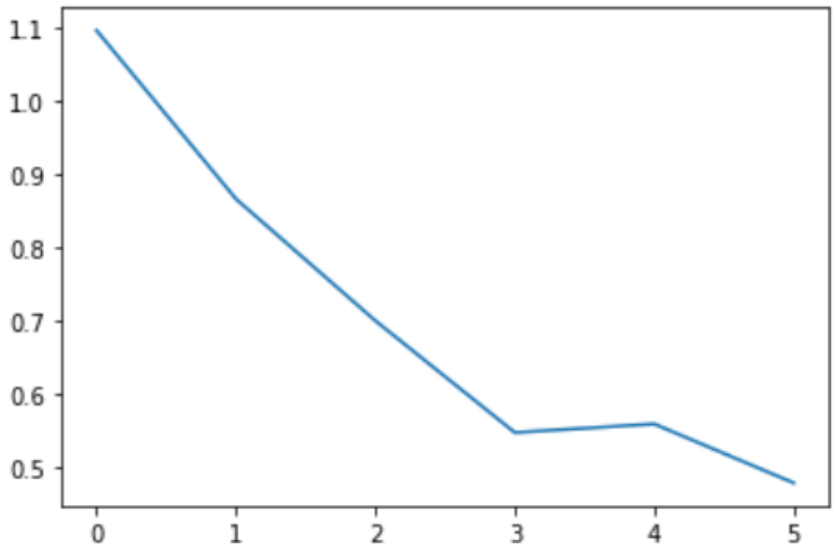
بدست آوردن این مشتق‌ها، به کمک Backpropagation انجام میشه.

در این قدم از پروژه، شبه‌کدی که بالاتر گفته شد رو به طور کامل پیاده‌سازی کنید. مجموعه Train رو، همون ۲۰۰ داده که در مرحله‌ی قبل گفته شد، در نظر بگیرید. Hyperparameter ها رو هم بدین شکل ست کنید: مقدار batch_size برابر با ۱۰، ضریب یادگیری برابر با ۱ و تعداد epoch ها برابر با ۵.

برای بدست آوردن گرادیان‌ها، ماتریس‌هایی و بردارهایی به ابعاد همان W و b و a ها در نظر بگیرید و با for زدن روی درایه‌ها، مشتق جزئی Cost نسبت به اون عنصر را بدست آورید.

در پایان این مرحله، دقت مدل رو برای همان ۲۰۰ داده گزارش کنید. با توجه به اینکه تعداد epoch ها و داده های آموزشی کم هستش، انتظار میره در پایان فرآیند یادگیری، دقت مدل در این حالت، بطور میانگین کمتر از ۷۰ درصد باشد. اگر زمان اجرا معقول بود (حدودا ۲-۳ دقیقه) می‌تونید به ازای تعداد epoch بیشتر هم، کدتون رو تست کنید.

همچنین میانگین Cost نمونه‌ها را در هر epoch محاسبه کنید و در آخر پلات کنید. انتظار میره که این میانگین‌ها، در هر epoch کاهش پیدا کنه و در نتیجه نمودار نهایی شبیه نمودار زیر بشه:



اگر این سیر نزولی در Cost ها دیده نشه، حتما مشکلی توی پیاده‌سازی الگوریتم وجود داره.

در آخر، زمان اجرای فرآیند یادگیری رو هم گزارش کنید.

قدم چهارم: Vectorization

دلیل اینکه تا اینجا فقط با ۲۰۰ داده اول دیتاست کار کردیم اینه که زمان اجرای پیاده‌سازی فعلی‌مون خیلی زیاد هستش. برای اینکه این مشکل رو برطرف کنیم، از مفهومی تحت عنوان **Vectorization** استفاده می‌کنیم. این مفهوم به این معنیه که به جای اینکه بیایم و توی کار با دیتامون، for بزنیم روی درایه‌ها، سعی کنیم عملیاتی که می‌خوایم انجام بدیم رو به شکل عملیات ماتریسی (ضرب و جمع ماتریسی و برداری، ضرب داخلی، ترانهاده کردن و اعمال توابع روی تک‌تک عناصر ماتریس‌ها) پیاده‌سازی کنیم.

این کار باعث میشه که زمان اجرای کد خیلی کمتر بشه. دلیلش اینه که عملیات‌های ماتریسی خیلی خوب می‌تونن موازی‌سازی بشن و به صورت چندهسته‌ای اجرا شن روی CPU، و همچنین پردازنده‌ها Instruction هایی مخصوص کار کردن با داده‌های بزرگ و برداری دارن که خیلی efficient تر اجرا می‌شن.

مرحله Feedforward الگوریتم رو، از همون اول به صورت Vectorized پیاده‌سازی کردیم. حالا توی این مرحله، باید Backpropagation رو هم Vectorized کنید. در پایان این مرحله انتظار میره که محاسبه‌ی مشتقات جزئی هر لایه (یعنی مشتقات نسبت به W و b و a ها) بدون for زدن انجام بشه.

برای مثال، کد پایین، برای محاسبه گرادیان برای وزن‌های لایه آخر:

```
for j in range(4):
    for k in range(60):
        grad_W3[j, k] += a2[k, 0] * sigmoid_deriv(z3[j, 0]) * (2 * a3[j, 0] - 2 * y[j, 0])
```

رو می‌تونید به صورت زیر بنویسید:

```
grad_W3 += (2 * sigmoid_deriv(z3) * (a3 - y)) @ (np.transpose(a2))
```

(علامت @ برای ضرب ماتریسی هستش).

یا محاسبه گرادیان برای نورون‌های لایه یکی‌مونده به آخر به شکل زیر هست:

```
grad_a2 = np.zeros((60,1))
for k in range(60):
    for j in range(4):
        grad_a2[k, 0] += W3[j, k] * sigmoid_deriv(z3[j, 0]) * (2 * a3[j, 0] - 2 * y[j, 0])
```

که میشه به صورت زیر Vectorized اش کرد:

```
grad_a2 =  np.transpose(W3) @ (2 * sigmoid_deriv(z3) * (a3 - y))
```

سایر عبارات رو هم مشابه همین Vectorized کنید.

در پایان این مرحله، انتظار میره که کدتون در زمان خیلی کمتری نسبت به مرحله‌ی قبل اجرا بشه. در نتیجه تعداد epoch ها رو افزایش بدید به عدد ۲۰ و دقت مدل نهایی و همچنین پلات Cost در طی زمان را گزارش کنید. با توجه به اینکه سرعت اجرا کد شما با توجه به ماتریسی شدن افزایش یافته است، برای این بخش کدتان را به نحوه ای طراحی کنید که ۱۰ بار از اول کدتان اجرا شود و نتیجه نهایی را به صورت میانگین کل نتایج، خروجی دهد.

قدم پنجم: تست کردن مدل

حالا که الگوریتم رو تا حد خوبی بهینه کردیم، می‌تونیم بریم و روی کل ۱۹۶۲ داده مجموعه Train، فرآیند یادگیری رو انجام بدیم. مقدار batch_size رو برابر با ۱۰، ضریب یادگیری رو برابر با ۱ و همچنین تعداد epoch ها رو ۱۰ در نظر بگیرید.

در پایان این قدم، دقت مدل رو برای مجموعه‌ی Train و همچنین برای مجموعه‌ی Test گزارش کنید. همچنین، همانند قبل میانگین Cost رو نیز پلات کنید. برای این قدم نیز مشابه حالت قبل کدتان بیاد نتایج ۱۰ اجرا را به صورت میانگین خروجی دهد.

اگر پیاده‌سازی‌ها درست انجام شده باشه، انتظار میره که دقت مدل برای Train و Test، حدود ۹۰ درصد باشه.

نکته: بسته به زبانی که باهاش پروژه رو پیاده‌سازی کردید، و قدرت سیستم‌تون، زمان اجرای فرآیند یادگیری متفاوت هستش. برای مثال با زبان پایتون و استفاده از NumPy، روی پردازنده Intel 7700HQ، با مقدار batch_size و epoch برابر ۱۰، برای هر بار اجرا بر روی کل دیتاست، حدود ۸ ثانیه زمان اجرای فرآیند یادگیری می‌باشد.

امتیازی‌ها

۱- همانطور که می‌دانید در روش Stochastic Gradient Descent ممکن است که مدل در مینیمم محلی گیر کند. برای کاهش این مشکل نیاز است که ضریب یادگیری مناسب انتخاب شود. مقادیر مختلف را برای learning rate و همچنین تعداد epoch و سایز batch تست کنید و در آخر تحلیل شخصی و مقادیر پیشنهادی خود را گزارش کنید.

۲- با انتخاب یک مقدار ضریب یادگیری مناسب همچنان امکان گیر کردن در مینیمم های محلی وجود دارد. برای بهبود این مسئله از ورژن های پیشرفته‌تر Stochastic Gradient Descent و یا الگوریتم های تکاملی و یا هر ایده دیگری که کمک‌کننده باشد، استفاده کنید. ایده و نتایج خود را گزارش کنید.

۳- ما در این مسئله تنها با چهار کلاس کار کردیم. حال که الگوریتم های خود را پیاده سازی کردید، می‌توانید آن را برای تعداد کلاس های بیشتر تست کنید. از دیتاست Fruits 360 یک یا چند میوه دیگر به دلخواه انتخاب کرده و به دیتاست اضافه کنید. سپس مدل خود را با کل دیتاست آموزش داده و تست کنید. برای افزایش سرعت تنها کار با الگوریتم ماتریسی خود کافی است. همچنین ممکن است برای افزایش دقت علاوه بر سایز لایه خروجی، سایز لایه های پنهان و hyperparameter های دیگر نیاز به تغییر داشته باشند. برای استخراج ویژگی نیز می‌توانید از کد داده شده و یا روش های دیگر استفاده کنید و همینطور می‌توانید تعداد نورون های ورودی را نیز تغییر دهید. در انتها تحلیل و نتایج خود را گزارش کنید.

۴- همانطور که مطلع هستید، از تابع softmax در مسائل classification برای نرمال سازی لایه خروجی شبکه عصبی و تبدیل آن به توزیع احتمالاتی استفاده می‌شود. از این تابع در مدل خود استفاده کرده و تحلیل و نتایج خود را گزارش کنید. برای مطالعه بیشتر در این مورد توصیه می شود به ویدیوهای آموزشی کانال درس مراجعه کنید.