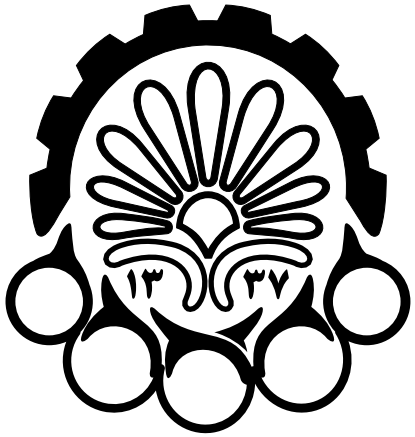

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

نام و نام خانوادگی : شروین غفاری

موضوع : Hybrid MPSO-CNN

استاد : دکتر قطعی

تدریسیار : دکتر یوسفی مهر

گزارش : هفتم

کلمات کلیدی :

1. شبکه عصبی نگاشتی
2. Multiple swarm
3. بهینه سازی Particle swarm
4. MPSO-CNN

چکیده :

الگوریتم PSO برای مسائل بهینه سازی مبتنی بر احتمال در یک فضای جستجو استفاده میشود و این الگوریتم که از طبیعت الهام گرفته شده کاربرد گسترده ای در حل مسائل بهینه سازی در دنیای حقیقی دارد.

با توجه به نوع مسائل میتوان با تغییراتی در این الگوریتم از ویژگی های آن که میتوان به تعداد پارامتر ها و پیچیدگی محاسباتی کم آن اشاره کرد استفاده نمود.

الگوریتم ارائه شده که MPSO نام دارد برای معماری شبکه عصبی نگاشتی (CNN) به منظور بهینه سازی hyperparameter های آن استفاده شده است.

این الگوریتم در مقایسه با یافتن پارامتر های بهینه به شکل manual توسط انسان ها سریعتر بوده و هزینه محاساباتی و overhead کمتری دارد.

الگوریتم MPSO همان PSO اما شامل 2 سطح است که سطح اول آن مربوط ایجاد مجموعه ای از لایه ها و بهینه سازی معماری شبکه است اما سطح دوم آن مربوط به بهینه سازی hyperparameters های لایه هایی است که در سطح اول ایجاد شده اند.

همچنین عملکرد و پیچیدگی الگوریتم MPSO-CNN به بعد فضای جستجو نیز وابسته است.

مقدمه :

برای حل اکثر مسائل در نیای حقیقی غالبا به شبکه های عصبی با بیش از 3 لایه نیاز است که پیچیدگی بالایی دارند و انتخاب hyperparameters به نسبت با افزایش لایه ها دشوار تر میشود .

شبکه های CNN جزو مورد استفاده ترین شبکه ها در deep network ها هستند . الگوریتم هایی که از هوش جمعی استفاده میکنند مانند PSO-ACO و یا الگوریتم ژنتیک کاربرد گشترده ای در چنین شبکه ها دارند اما الگوریتم PSO بیشترین استفاده را به دلیل پارمتر های کم محسابات ساده و فرموله کردن ساده مسائل دارد.

یکی از نقاط ضعف این الگوریتم همگرایی سریع به جواب های sub-optimal است که با ایجاد تنوع در initialization و یا تنظیم پارمتر ها میتوان این مشکل را برطرف کرد. مدل تغییر یافته آن که در حال حاضر در شبکه CNN به کار برده میشود یک تعادل بین یافتن جواب های جدید و استفاده از جواب های بدست آمده ایجاد کرده و همچنین با اعمال جهش در برخی از particle ها در مراحل میانی این الگوریتم میتوان به همگرایی بهتری دست یافت.

هدف این مقاله استفاده از مدلی از PSO که توانایی یادگیری معماری شبکه را داشته و پارمتر های بهینه مربوط به لایه های آن را به شکل خودکار و بدون دخالت انسان ها پیدا کند.

در گذشته انتخاب پیکربندی مناسب برای شبکه های عصبی نگاشتی بسته به نوع دیتاست نیازمند مهارت گسترده ای بود اما با الگوریتم ارائه شده حتی افراد غیر متخصص نیز میتوانند از این شبکه استفاده کنند.

MPSO **سطح اول در فضای جستجو به دنبال مجموعه ای لایه ها برای شبکه و در سطح دوم به دنبال پارمتر هایی با بهترین عملکرد برای هر لایه در سطح اول است. برای ارزیابی هر** particle **از تابع** fitness evaluation **که وابسته به** CNN **بوده استفاده شده اما به دلیل عدم وابستگی** particle **ها به هم میتوان در مدت زمان مشخص با موازی سازی پارامتر های هر** particle **را یافت که البته نیازمند سخت افزار قدرتمندی است.**

: Convolutional-Neural-Network

یک شبکه عصبی چندلایه بوده که پیچیدگی محاسباتی شبکه های عصبی قدیمی تر را ندارد همچنین به دلیل اینکه بعد فیلتر هایی که روی تصویر ورودی اعمال میشوند نسبت به تصویر ورودی بسیار کمتر است تعداد پارمتر های مورد نیاز برای ارتباط بین خروجی لایه قبل و ورودی لایه بعدی کاهش میابد.

این شبکه شامل 3 نوع لایه بوده که به ترتیب :

1. Convolutional-layer
2. Pooling-layer
3. Fully connected-layer(Dense)

لایه convolutional **شامل فیلتر هایی با بعد تعریف شده است که بر روی تصویر ورودی اعملا شده تا** feature **های آن استخراج شود . اگر چندین مرتبه فیلتر روی تصویر اعمال شود یک** feature map **ساخته میشود.**

لایه Pooling **با کاهش دادن بعد بردار های ویژگی خروجی لایه convolutional و حفظ اطلاعات مهم آن محسابات را کاهش میدهد و** Max-Pooling **یکی از آنها بوده که بین دو لایه** Convolutional **قرار میگیرد و برای هر کانال رنگی** (color-channel) **به طور مستقل عمل میکند.**

لایه fully connected **شامل دو بخش بوده که بخش اول آن برای کاهش بعد بردار خروجی لایه قبل یعنی** Pooling layer **بوده و بخش دوم آن مربوط به کلاس بندی و** label **گذاری با تابع** softmax **می باشد.**

مدل پیشنهادی MPSO-CNN :

الگوریتم MPSO **شامل چندین** swarm **بوده که برای بهینه سازی پیکربندی و** hyperparameters **های شبکه** CNN **استفاده میشود.**

در swarm **سطح اول هر** particle **نشان دهنده یک پیکربندی برای لایه های** convolutional,pooling,fully connected **بوده و سطح دوم شامل** swarm m **که هر کدام از** particle n **تکشیل شده اند . در این سطح هر** swarm **متناظر با یک** particle **در سطح اول است.**

Particle **های هر** swarm **در سطح دوم شامل** filter size,padding size,stride size **برای هر کدام از لایه های سطح اول است و این مقادیر ابتدا به صورت تصادفی مقدار دهی شده اند.**

در نهایت لایه convolutional **بردار ویژگی ها را ستخراج کرده و در لایه** softmax **مقدار** accuracy **را با تابع** fitness **که یک** particle **را به عنوان ورودی دریافت میکند را بر میگرداند.**

مقـادیر velocity **و** position **برای هر** particle **در ســـطح اول و دوم در هر** iteration **با توجه به** p_best **و** g_best **محاســبه شــده و درنهایت** [Pi,Pij] **که** Pi **نشـــان دهنده بهینه ترین مقدار برای تعداد لایه های هر کدام از ســـه نوع لایه گفته شـــده و** Pij **نشـــان دهنده بهینه ترین** hyperparameter **های مربوط به** Pi **است.**

پیاده سازی الگوریتم :

برای ارزیابی عملکرد الگوریتم از دیتاســـت MNIST **از کتابخانه** keras **اســـتفاده شـــده ســـپس با اســـتفاده از تابع** load_data() **داده ها را به دو بخش** training test**, تقسیم کرده و** preprocess **را بر روی داده ها انجام میدهیم .**

ســـپس در تابع evaluate_fitness **پیکربندی** CNN **را که شـــامل تعداد لایه های** convolutional **و** pooling **و** fullyconnected **به عنوان ورودی داده**

و باتوجه به اینکه اعداد به صورت تصادفی بین صفر و یک تولید شده اند آنها را به بازه تعیین شده scale **میکنیم.**

با دســـتور Sequential() **یک مدل** cnn **ایجاد کرده و با توجه به تعداد لایه ها با دستور** model.add() **میتوان لایه مورد نظر را به این شبکه اضافه کرد .**

به تعداد لایه های convolutional **با دستور** conv2D() **که تعداد فیلتر ها و سایز فیلتر را به عنوان ورودی دریافت میکند لایه جدید در شبکه قرار میدهیم.**

Input_shape() **با توجه به** width,height,color_channel **به صـــورت پیشـــفرض** (28,28,3) **در نظر گرفته شده است.**

برای ایجاد لایه Maxpooling **که بعد** feature map **را کاهش میدهد از دستور** Maxpooling2D **که ورودی آن سایز فیلتر برای** pooling **را نشان میدهد مقدار دهی میکنیم.**

برای تبدیل feature map **به یک بردار تک بعدی از** Flatten() **استفاده میکنیم.**

درنهایت برای اضافه کردن لایه fullyconnected **که بخش اول آن برای کاهش بعد خروجی لایه قبل استفاده شده با دستور** Dense **که ورودی آن تعداد** neuron **های این لایه را مشخص میکند و لایه دوم آن که مربوط به کلاس بندی و** label **زدن است تعداد کلاس های مورد نیاز را به عنوان ورودی دریافت کرده و** activation **آن را تابع** softmax **برای انتخاب کلاس ها به شکل احتمالی استفاده میشود.**

سپس مدل compile **شده و با دستور** fit **آنرا** train **میکنیم.**

برای بدست آوردن دقت بدست آمده با مقادیر داده شده evaluate **کرده و** accuracy **بدست می آید.**

: Particle Swarm

برای ایجاد particle **یک کلاس ایجاد کرده و به** constructor **آن محل فعلی** particle **را داده که** pbest **نیز به صورت پیشفرض همان در نظر گرفته میشود اما مقدار سرعت را 0 در نظر میگیریم اما** fitness **در ابتدا مقدار ندارد.**

ابتدا یک population **ایجاد میکنیم که مقادیر ورودی آن** Particle **با ورودی یک پیکربندی پیشفرض برای** CNN **است و با تعداد** generation **های مشخص شده** pbest **را بر اساس** evaluation fitness **انتخاب میکنیم که همان دقت** cnn **برای مقادیر این** particle **است.**

در مرحله بعد gbest **انتخاب شده و** velocity **و** position **بروزرسانی میشوند.**
در نهایت میتوان با اســـتفاده از مقادیر gbest **بهینه ترین پارمتر های شـــبکه** CNN **را انتخاب کرد.**

نتایج :

انتخاب پارمتر های بهینه با اسـتفاده از try and error **بسـیار وقت گیر و نیازمند محاسبات پیچیده هستند اما استفاده از** pso **که به صورت تصادفی اما هدایت شده عمل میکند میتواند این مسئله را ساده تر کند اما در ابتدا نیازمند تعریف فضای جستجو توسط کاربر می باشد.**
محدوده متغیر ها به صـورت دسـتی انتخاب شـده و محدوده مجاز برای تغییر مکان particle **ها را مشخص میکند.**
در ابتدای الگوریتم یک جمعیت اولیه از پیکربندی های ممکن برای شـــبکه CNN **ایجاد کرده و با** iteration **هـای بعـدی آنرا بهبود میـدهیم و الگوریتم بـه جواب بهینه پس از تعداد متناهی بار همگرا میشود.**
با افزایش محدوده جستجو پارمتر ها میتوان جواب های بهتری را ایجاد کرد.

نتیجه گیری:

الگوریتم MPSO **معرفی شـده به دلیل پیاده سـازی سـاده و تعداد پارامتر های کم آن نسـبت به دیگر الگوریتم های تکاملی برای شـبکه های عصـبی عملکرد قابل قبولی دارد و همانطور که گفته شـد این الگوریتم شـامل دو سـطح بود که سطح اول شامل یک** swarm **بود که مربوط به پیکربندی شبکه و سطح دوم آن مربوط به انتخاب** hyperparameter **های بهینه برای لایه های انتخاب شده بود.**

نتایج بدست آمده از 5 دیتاست استفاده شده تائیدی بر عملکرد موفق این الگوریتم در انتخاب پارمتر های شبکه CNN **بود که به طور خودکار و تصادفی عمل میکرد.**

در نهایت میتوان این الگوریتم را با ایجاد تغییراتی در تعداد iteration **ها یا اجرای موازی هر کدام از** particle **ها با توجه به ماهیت مستقل آنها بهبود داد و حتی برای مسائل دیگر به کار گرفت.**

منابع :

https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python/

https://towardsdatascience.com/swarm-intelligence-coding-and-visualising-particle-swarm-optimisation-in-python-253e1bd00772

https://www.datacamp.com/tutorial/convolutional-neural-networks-python