Sharif University of Technology
Computer Engineering

Bachelor's Thesis

Computer Engineering

Topic

Transfer Learning Applications in Reinforcement Learning

By
Amir-Hossein Shahidzadeh

Supervisor
Prof. Hamid Reza Rabiee

Fall 2020

Abstract

Throughout Autonomous History, reinforcement learning algorithms' heavily rely on expensive sensors like LiDAR, multi-camera, and 3D-camera, etc. which is conducive to the high price of such vehicles these days. Therefore, we were inclined to avoid the collisions only with the help of a single camera (monocular image) and high-level feature extractor.

Moreover, in this thesis, I have tried to train in a simulator to minimize real-world collision due to such systems' criticality and deployed transfer learning methods to handle distribution shifts between simulation and real-world images.

Keywords: *Deep Learning, Transfer Learning, Domain Adaptation, Reinforcement Learning*

Contents

چکیده

امروزه به کمک یادگیری کمکی سیستم‌های خودران پیشرفت چشمگیری داشته‌اند ولی در عمل داشتن چنین سیستم‌های مفیدی برای همگان به دلیل استفاده از حسگرهای گران فراوان همچنان ممکن نیست. ما قصد داریم در این مقاله راهی را برای کاهش این حسگرها پیشنهاد دهیم. برای این منظور ما می‌خواهیم همه‌ی حسگرهایی که به حرکت کردن و ناوبری کمک می‌کنند را با یک دوربین ساده جایگزین کنیم. به این ترتیب که تنها با تصویر یک دوربین بتوانیم بدون برخورد حرکت کنیم اما در عمل به دلیل بنیان بحرانی و گرانی سیستم‌های خودران، آموزش آن‌ها در محیط یادگیری هزینه‌ی گزافی دارد و برای حل این مشکل روش جایگزین می‌تواند استفاده از روش‌های پردازش تصویر برای افزایش شباهت توزیع آماری در زمان یادگیری و آزمایش باشد. به این ترتیب ما می‌توانیم با هزینه‌ی کمتر و سرعت بیشتر در محیط‌های شبیه‌ساز مدل خود را آموزش دهیم. در ادامه به تعریف این مشکل و بررسی روش‌های تعمیم یادگیری در توزیع‌های مختلف می‌پردازیم.

کلمات کلیدی: یادگیری عمیق، انتقال یادگیری، یادگیری کمکی، تطبیق دامنه

فهرست مطالب

فصل ۱

مقدمه

مسیریابی و عدم برخورد یکی از اولین نیازهای سیستم‌های رباتیک است. تاکنون بسیاری از این سیستم‌ها از حسگرهای عمق، بازسازی سه بعدی، دوربین دوگانه برای تشخیص عمق و سایر روش‌های هزینه بر استفاده می‌کردند که امروزه با توجه به پیشرفت یادگیری عمیق و شبکه‌های عصبی از روی یک عکس می‌شود با دقت خوبی عمق تصویر را به دست آورد. از طرف دیگر روش‌های یادگیری کمکی نیز پیشرفت چشم‌گیری داشته‌اند و اکنون به کمک شبکه‌های عصبی می‌توانند مسائل فراتر از بازی‌های آتاری را حل کنند.

۱.۱ تعریف مسئله و اهداف تحقیق

یادگیری کمکی عمیق امروزه به عنوان یک روش بسیار کارآمد در ماشین‌های خودران شناخته می‌شود اما به دلیل بحرانی بودن این سیستم‌ها اکثرا امکان یادگیری در دنیای واقعی از طریق آزمون و خطا وجود ندارد. در این تحقیق ما می‌خواهیم روش‌هایی را که ممکن است برای یادگیری کامل در محیط شبیه ساز و آزمون در دنیای واقعی باشد را بررسی کنیم.

۲.۱ ساختار مقاله

این مقاله از ۴ فصل تشکیل شده است که در فصل دوم به بررسی روش‌های یادگیری کمکی در حوزه‌ی گرادیان سیاست به دقت پرداخته می‌شود و در فصل سوم از مشکلات ناشی از یادگیری و آزمون در دو محیط مختلف صحبت می‌کنیم و در نهایت راه‌های مختلف غلبه بر مشکلات یادگیری در دو محیط مختلف را بررسی می‌کنیم.

فصل ۲

یادگیری کمکی

یادگیری کمکی[۲] یک زیرشاخه بزرگ از یادگیری ماشین است که هدف در بیشینه کردن بهره‌وری در یک هدف بلند مدت دارد و اخیرا توانسته نتایج خیره‌کننده‌ای کسب کند تا جایی که حتی قادر بر شکست دادن قهرمانان بازی‌هایی مثل بازی‌های آتاری شده.

نحوه‌ی یادگیری در این روش شبیه به یادگیری برخی مهارت‌هایی مثل راه رفتن است که انسان براساس عمل و عکس‌العمل از محیط یاد می‌گیرد و به طور خلاصه به این صورت است که یک یا چند عامل که دارای وضعیت[۳] مشخصی هستند، در محیط عملی را انجام می‌دهند و با توجه به عکس‌العملی که از محیط مشاهده می‌کنند ممکن است به یک وضعیت جدید روند و تجربه بدست می‌آورند.

در ادامه به روش‌های مبتنی بر گرادیان سیاست[۴] می‌پردازیم. فرض بر آن است که خواننده با تعاریف و مفاهیم ابتدایی یادگیری کمکی از جمله فرآیند تصمیم‌گیری مارکوف[۵] تسلط دارد.

۱.۲ گرادیان سیاست

روش گرادیان گیری از سیاست یکی از روش‌های حل یادگیری کمکی است که به دنبال سیاست بهینه‌ای است که بتواند بیشترین پاداش را از محیط دریافت کند.

برخلاف $Q - learning$ که سعی داشت به کمک یک شبکه تابع Q را تخمین بزند، در این روش شبکه‌ی عصبی‌ای روی توزیع حرکت‌ها نسبت به وضعیت ($State \rightarrow action$) داریم که می‌خواهیم امیدریاضی پاداش

---

[۲] Reinforcement Learning

[۳] State

[۴] Policy Gradients

[۵] Markov Decision Process

تخفیف‌یافته را بیشینه کند. پس در واقع یک شبکه‌ی سیاست $\pi(a|s;\theta)$ داریم که اگر مسیرهای حرکت‌مان[۱] را به صورت $\tau = \{(s_۱, a_۱, r_۱), (s_۲, a_۲, r_۲), ..., (s_T, a_T, r_T)\}$ تعریف کنیم، هدف ما پیدا کردن θای است که $J_\theta = E_{\tau\sim p(\tau;\theta)}[\sum_{t\in\tau}\gamma^t r_t]$ را بیشینه کند. ضمنا دقت کنید که احتمال یک مسیر حرکت هم به θ وابسته است که به صورت زیر تعریف می‌شود:

$$p(\tau;\theta) = p(s_۱)\prod_{t}^{T-۱}\pi_\theta(a_t|s_t)p(s_{t+۱}|s_t,a_t)$$

حال برای بهینه کردن هدف طبق روش افزایش گرادیان[۲] $\theta = \theta + \nabla_\theta J(\theta)$ نیاز به گرادیان گرفتن از J_θ داریم که به صورت زیر است:

$$\begin{aligned}
R(\tau) &= \sum_{t\in\tau}\gamma^t r_t \\
\nabla_\theta J(\theta) &= \sum_t \nabla_\theta p_\tau(\tau;\theta)R(\tau) \\
&= \sum_t p_\tau(\tau;\theta)\nabla_\theta \log p_\tau(\tau;\theta)R(\tau) \\
&= E_{\tau\sim p_\tau(\tau;\theta)}[\nabla_\theta \log p_\tau(\tau;\theta)R(\tau)] \quad (۱)
\end{aligned}$$

که برای $\nabla_\theta \log p_\tau(\tau;\theta)$ نیز طبق تعریفی که برای $p_\tau(\tau;\theta)$ داشتیم:

$$\nabla_\theta \log p_\tau(\tau,\theta) = \cancel{\sum_t^{T-۱}\nabla_\theta \log p(s_{t+۱}|s_t)} + \sum_t^{T-۱}\nabla_\theta \log \pi(a_t|s_t;\theta) \quad (۲)$$

و در نهایت طبق رابطه‌ی (۱) ، (۲) داریم:

$$\nabla_\theta J(\theta) = E_{\tau\sim p(\tau,\theta)}\left[\left(\sum_t^{T-۱}\nabla_\theta \log \pi(a_t|s_t;\theta)\right)R(\tau)\right]$$

---

[۱] Trajectory

[۲] Gradient Ascent

که این امیدریاضی را به کمک نمونه‌گیری[۱] مسیرهای حرکت می‌توان تخمین زد و در این صورت خواهیم داشت:

$$\nabla_\theta J(\theta) \approx \frac{۱}{N} \sum_{n=۱}^{N} R(\tau^{(n)}) \sum_{t \in \tau^{(n)}} \nabla_\theta \log \pi_\theta(a_t^{(n)} | s_t^{(n)})$$

برای راحت‌تر شدن فرآیند یادگیری، بهتر است پاداش هر مسیر حرکت را به کل حرکات یک مسیر ندهیم زیرا ممکن است حرکت بدی نیز در آن مسیر داشته باشیم و $r(s_{t-۱}, a_{t-۱})$ ربطی به s_t ندارد بنابراین بهتر است پاداش به ازای حرکات بعدی را برای هر (s, a) منظور کنیم زیرا حرکت فعلی تنها در حرکات بعدی تاثیر می‌گذارد و به این ترتیب خواهیم داشت:

$$\nabla_\theta J(\theta) \approx \frac{۱}{N} \sum_{n=۱}^{N} \sum_{t \in \tau^{(n)}} \sum_{t' \geq t} \gamma^{t'-t} r(s_{t'}^{\tau^{(n)}}, a_{t'}^{\tau^{(n)}}) \nabla_\theta \log \pi_\theta(a_t^{(n)} | s_t^{(n)})$$

در نهایت اگر بخواهیم به صورت شهودی رابطه‌ای که برای به روز کردن θ بدست آوردیم را بررسی کنیم می‌توانیم بگوییم هرچه پاداش مسیری بیشتر باشد احتمال a_tهای آن مسیر بیشتر می‌شود و هرچه پاداش مسیری کمتر باشد انگار که می‌خواهیم احتمال a_tهای آن را کاهش دهیم. در ادامه با روش‌هایی برای کاهش واریانس گرادیان آشنا می‌شویم.

REINFORCE ۱.۱.۲

در بخش قبل سعی کردیم تا جایی که می‌شود واریانس گرادیان‌هایی که یکی از عوامل آن واریانس زیاد در پاداش است را کم کنیم اما هنوز هم واریانس گرادیان‌هایی که از مسیرهای مختلف بدست می‌آید زیاد است و مثلا در مواردی که پاداش کلا مثبت باشد هم این روش جوابگو نیست. پس همانطور که به نظر می‌آید باید پاداشی که در هر وضعیت می‌گیریم را با مقدار متوسط پاداشی برای هر وضعیت[۲] مقایسه کنیم یا به عبارت دیگر مقدار$A(s, a) = R(s, a) - V(s)$ را در نظر بگیریم که به آن تابع سود گفته می‌شود.
روند یادگیری در روش REINFORCE را با شبه‌کد زیر بررسی می‌کنیم:

۱. یک سیاست با وزن تصادفی θ در نظر می‌گیریم

[۱] Sampling
[۲] State

۲. برای هر مسیر e

(آ) یک وضعیت تصادفی در نظر می‌گیریم

(ب) از $t = ۱$ تا زمانی که مسیر تمام شود

i. یک حرکت a_t با توجه به روش $epsilon - greedy$ انتخاب می‌کنیم.

ii. حرکت a_t را انجام می‌دهیم و عکس‌العمل محیط که شامل پاداش r_t و وضعیت جدید $s_{t+۱}$ است را مشاهده می‌کنیم.

iii. وزن شبکه‌ی سیاست را با فرمول زیر به روز می‌کنیم

$$\theta \leftarrow \theta + \eta \frac{۱}{T} \sum_{t \in \tau} A_t \log(\pi_\theta(a_t|s_t)$$

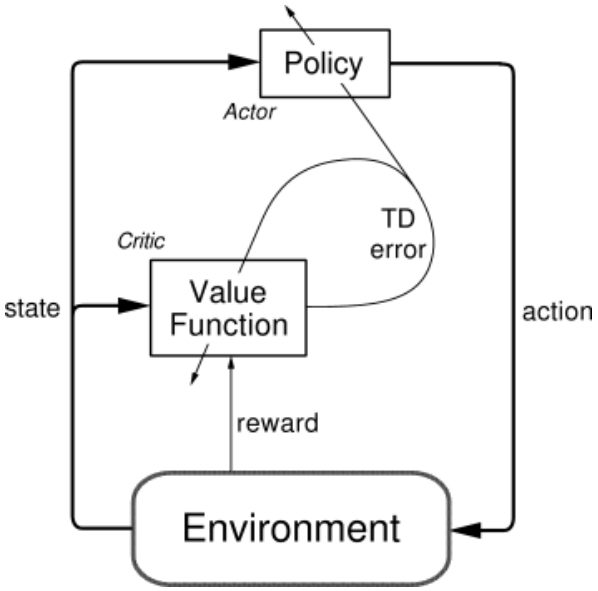
۲.۲ Actor Critic

در این روش همچنان به دنبال کاهش واریانس و معنادارتر کردن تخمین مقدار پاداش هستیم. همچنین مشکل دیگری که روش‌هایی قبلی داشتند این بود که برای بدست آوردن R_t نیاز داشتیم که تمام مسیر را طی کنیم اما در این روش با استفاده از ایده‌ی Temporal Diffrence می‌خواهیم یک شبکه برای $V(s)$ هم داشته باشیم که بتوانیم $R(s_t) = r_t + \gamma V(s_{t+۱})$ قرار دهیم تا هم نویز در پاداش کمتر شود و هم به تخمین بهتری از پاداش برسیم. به این روش که یک شبکه برای سیاست به نام actor و یک شبکه برای تشخیص میزان ارزش $(Value)$ هر وضعیت داشته باشیم به نام critic روش actor-critic می‌گویند.

همچنین یکی دیگر از مزایای داشتن شبکه‌ای برای تخمین $V(s_t)$ این است که می‌توانیم مقدار $Q(s, a)$ در حالت غیرمعین[۱] را براساس رابطه‌ی زیر تخمین بزنیم:

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+۱} \sim p(s_{t+۱}|s_t, a_t)}[V^\pi(s_{t+۱}]$$

[۱] Non determinestic



شکل ۱.۲: نحوه‌ی ارتباط بین بخش‌های actor-critic

۳.۲ Soft Actor-Critic

مقاله‌ی Soft Actor-Critic [۱]در سال ۲۰۱۸ معرفی شد و بهترین نتیجه را روی روش‌های actor-critic داشته است. در این مقاله علاوه بر بیشینه کردن پاداش بلند مدت به هدف مهمی که تا آن زمان خیلی مورد توجه نبوده پرداخته است. در این مقاله یک هدف نسبتا جدید به نام بیشینه آنتروپی تعریف شده که آنتروپی را روی حرکت‌های یک وضعیت تعریف کرده و هدف پیشنهادی خود را به شکل رابطه‌ی زیر معرفی می‌کند:

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho^\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(.|s_t)]$$

اضافه شدن آنتروپی به هدف اصلی یادگیری کمکی مزایای زیادی دارد:

- ترغیب شدن به کاوش[۱] بیشتر به دلیل اینکه می‌خواهد همه‌ی حرکات خوب را پیدا کند و اگر تنها یک حرکت خوب پیدا کند آنتروپی آن کم شده و هدف مارا ارضا نمی‌کند و این در حالی است که برای مسیرهای بد اینگونه نیست چون پاداش کمی دارند.
- سیاست می‌تواند در چند حالت خوب نزدیک بهینه داشته باشد و در حالاتی که چند حرکت بهینه برای یک وضعیت داشته باشیم به ما کمک می‌کند که برای همه‌ی آنها احتمال یکسان در نظر بگیریم.

[۱] Explore

ساختار مدل آنها به این شکل است که از سه شبکه برای تخمین $Q(s,a)$ و $V(s)$ و $\pi(a|s)$ استفاده کرده‌اند که به اختصار هدف هر یک از شبکه‌ها را بررسی می‌کنیم:

شبکه‌ی Q(s،a)

هدف شبکه‌ی Q از رابطه‌ی زیر پیروی می‌کند:

$$J_Q(\theta) = E_{(s_t,a_t)\sim\mathcal{D}}\Big[\frac{۱}{۲}\Big(Q_\theta(s_t,a_t) - r(s_t,a_t) + \gamma E_{s_{t+۱}\sim p}[V_{\bar{\psi}}(s_{t+۱})]\Big)^۲\Big]$$

شبکه‌ی V(s)

هدف شبکه‌ی V از رابطه‌ی زیر پیروی می‌کند:

$$J_V(\bar{\psi}) = E_{s_t\sim\mathcal{D}}[\frac{۱}{۲}(V_{\bar{\psi}} - E_{a_t\sim\pi_\phi}[Q_\theta(a_t|s_t) - \log\pi_\phi(a_t|s_t)])^۲]$$

شبکه‌ی $\pi(a|s)$

هدف شبکه‌ی π از رابطه‌ی زیر پیروی می‌کند:

$$J_\pi(\phi) = E_{s_t\sim\mathcal{D}}\Big[D_{KL}\Big(\pi_\phi(.|s_t)\|\frac{e^{Q_\theta(s_t,.)}}{Z_\theta(s_t)}\Big)\Big]$$

حال روش کلی Soft Actor-Criticرا به صورت شبه‌کد زیر می‌توان خلاصه کرد:

۱. ابتدا برای شبکه وزن تصادفی شروع می‌کنیم

۲. برای هر پیمایش

(آ) برای هر گام در محیط

i. $a_t \sim \pi_\phi(a_t, s_t)$

ii. $s_{t+۱} \sim p(s_{t+۱}|s_t, a_t)$

iii. $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t,a_t), s_{t+۱}\}$

(ب) برای هر قدم گرادیان

i. $\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$

ii. $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta i} J_Q(\theta_i) for i \in \{۱, ۲\}$

iii. $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$

iv. $\bar{\psi} \leftarrow \tau\psi + (۱ - \tau)\bar{\psi}$

فصل ۳

تغییر توزیع آماری

در بخش قبل روش های مختلف یادگیری کمکی را با فرض یکسان بودن توزیع آماری داده های محیط یادگیری و محیط آزمایش بررسی کردیم اما در عمل به دلیل بنیان بحرانی و گرانی سیستم‌های خودران، آموزش آن‌ها در محیط یادگیری هزینه‌ی گزافی دارد و برای حل این مشکل روش جایگزین می‌تواند استفاده از روش‌های پردازش تصویر برای افزایش شباهت توزیع آماری در زمان یادگیری و آزمایش باشد. به این ترتیب ما می‌توانیم با هزینه‌ی کمتر و سرعت بیشتر در محیط‌های شبیه‌ساز مدل خود را آموزش دهیم. در ادامه به تعریف این مشکل و بررسی روش‌های تعمیم یادگیری در توزیع‌های مختلف می‌پردازیم.

تغییر توزیع آماری داده یکی از مسایل مهم، متداول و ساده در علم داده است که همواره مسئله‌ای بدیهی به نظر آمده است ولی در این مقاله ما انواع علت‌ها و نمودهای شهودی تغییر توزیع در داده ها و مشکلاتی که بوجود می‌آورند را بررسی می‌کنیم.

۱.۳ تعریف

مسئله ی تغییر توزیع آماری زمانی رخ می‌دهد که توزیع توام داده‌ی یادگیری و آزمایش متفاوت باشند یا به عبارت دیگر$P_{train}(x, y) \neq P_{test}(x, y)$ برقرار باشد.

۲.۳ انواع تغییر در توزیع داده

تغییرات در توزیع‌های آماری انواع زیادی دارند و در این مقاله ما تغییراتی را بررسی می‌کنیم که در سیستم‌های خودران رایج هستند، این تغییرات عبارت اند از:

۱. تغییر توزیع ویژگی‌ها

۲. تغییر توزیع برچسب‌ها

۳. داده‌ی نامتعادل

۱.۲.۳ تغییر توزیع ویژگی‌ها

این تغییر در مسایل $X \to Y$ در حالتی که شرط زیر برقرار باشد رخ می‌دهد.

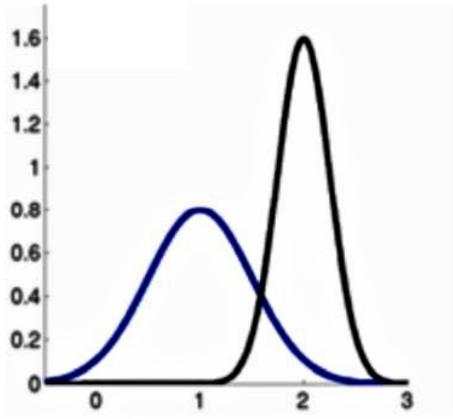
$$(P_{train}(x|y) = P_{test}(x|y), P_{train}(x) \neq P_{test}(x))$$

به عبارت دیگر زمانی که شکل داده های ورودی عوض شود مثلا از تصاویر انیمیشن به تصاویر واقعی، می‌گویند تغییر توزیع ویژگی ها رخ داده است. برای مثال می‌توانید نمونه‌هایی از تغییر ویژگی ها در تصاویر زیر مشاهده کنید.

شکل ۱.۳: عکس راست در داده‌های train و عکس سمت چپ در داده‌های test

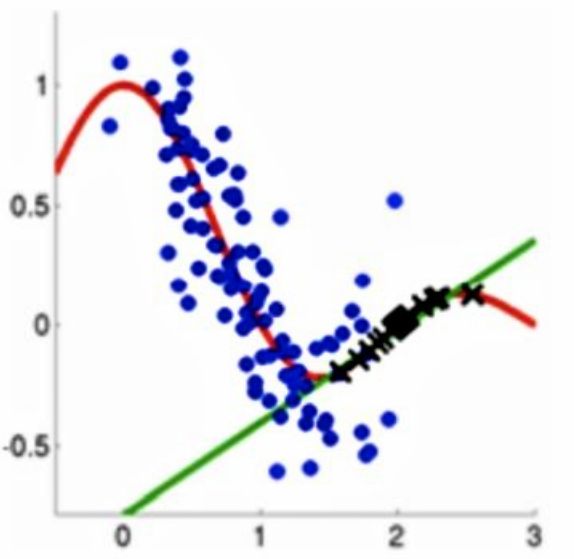
در این تغییر رایج بین محیط یادگیری و آزمایش، توزیع داده‌ها عوض شده ولی دسته‌بندی تصاویر عوض نشده و این تغییر باعث سخت شدن یادگیری مدل می‌شود.

برای نمونه تصور کنید توزیع داده‌ها به شکل زیر باشد :



شکل ۲.۳: چگالی داده‌ها

حال اگر با داده های سیاه یک مدل ساده‌ی رگرشن را آموزش دهیم به خط سبز می‌رسیم که MSE خوبی روی داده‌های سیاه دارد درحالی که روی داده‌های آزمایش که آبی هستند دقت بسیار پایینی دارد و یکی از علتهای این خطا تغییر توزیع دادهها است که در ادامه روش‌های حل این مشکل و نحوه‌ی رسیدن به تابع قرمز گفته می‌شود. از نمونه‌های شهودی دیگر تغییر توزیع داده‌های آموزش و آزمایش می‌توان به :



شکل ۳.۳: نتیجه‌ی رگرشن

- آموزش در محیط انیمیشنی شبیه‌ساز و آزمایش در محیط واقعی
- آموزش تخمین عمق در خارج و داخل خانه فقط با داده‌هایی از داخل خانه

اشاره کرد.

در نهایت مشاهده کردیم که در صورت تفاوت توزیع داده ها در زمان آموزش نسبت به زمان آزمایش مشکلات زیادی برای یادگیری شبکه به وجود می‌آید برای مثال درالگوریتم cross-validation اگر این مشکل را داشته باشیم بعضی از دسته ها ممکن است کاملا بایاس شوند و خطای ارزیابی را دچار نوسان زیادی بکنند.

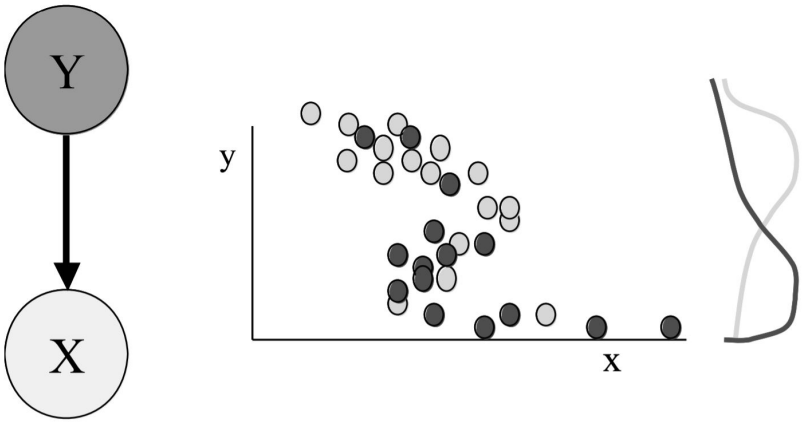
۲.۲.۳ تغییر توزیع برچسب‌ها

درحالی که تغییر وِیژگی ها در در ورودی مدل تاثیر می‌گذاشت، این نوع تغییر در توزیع داده‌ها دقیقا برعکس عمل می‌کند و در تاثیر برچسب می‌گذارد و به صورت آماری می‌توان نوشت در مسایل $Y \rightarrow X$ هنگامی که شرط زیر برقرار باشد با مشکل تغییر توزیع برچسب روبرو هستیم:

$$P_{train}(x|y) = P_{test}(x|y), P_{train}(y) \neq P_{test}(y)$$

نام دیگر این مشکل داده‌های نامتوازن است به این ترتیب که از هر برچسب تعداد متفاوت و با واریانس بالا داشته باشیم. البته دقت داشته باشید که ممکن است این مسئله ظاهرا نشان دهنده‌ی یک مشکل نباشد و در ذات مسئله چنین توزیعی بین داده ها و برچسب‌ها وجود داشته باشد (برای مثال تعداد حرکت رو به جلو در ربات ها که بیشتر از حرکت چرخش باعث برخورد می‌شود) اما منظور از این مشکل زمانی است که توزیع برچسب ها در زمان آزمایش و آزمون متفاوت باشد.

برای مثال می‌توان به مدل Bayes اشاره کرد به این صورت که با فرض دانستن $P(Y)$ می‌توان $P(Y|X)$ را به صورت علّی از $P(X|Y)P(Y)$ بدست آورد. اما مشکل زمانی پیش می‌آید که $P_{train}(Y)$ و $P_{test}(Y)$ متفاوت باشند که در این صورت دیگر نمی‌توان از $P_{train}(Y)$ برای آموزش استفاده کرد حتی در این مورد هم اگر $P_{test}(Y)$ را بدانیم می‌توانیم از این توزیع استفاده کنیم ولی در عمل این چنین نیست و ما $P_{test}(Y)$ را هم نمی‌دانیم.



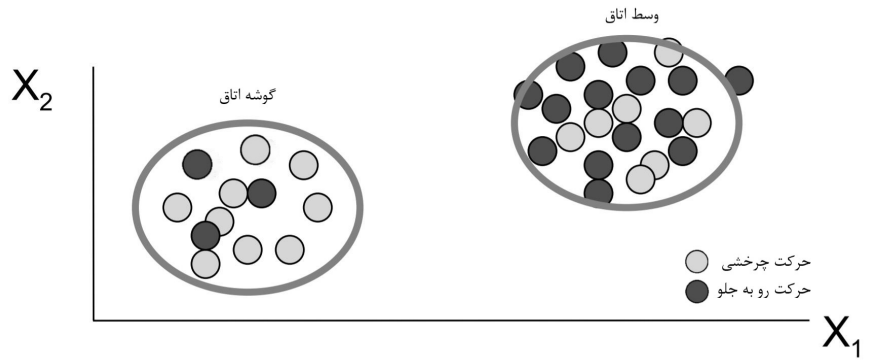
شکل ۴.۳: در این نمودار[۳] همانطور که مشاهده می‌کنید با افزایش Y، X کاهش می‌یابد پس متغیر X وابستگی قابل توجه‌ای به متغیر Y دارد و اگر توزیع Y تغییر کند مدل تغییر زیادی خواهد داشت به طوری که ممکن است به کلی خروجی را بی‌معنی کند.

۳.۲.۳ داده‌ی نامتعادل

در روش‌های مثل epsilon-greedy به خصوص مواردی که کاوش[۱] کمی داریم بسیار محتمل است که توزیع داده برای هر دسته متفاوت باشد و برای حرکاتی که نادر هستند اطلاعات کافی‌ای از توزیع برای تصمیم‌گیری و آموزش مدل ها نداشته باشیم و از طرف دیگر برای بعضی دسته‌ها حرکات بسیار زیادی جمع آوری شود که به راحتی بتوانیم آن‌ها را تعمیم دهیم. به این دسته از داده، داده‌ی نامتعادل می‌گویند.

علت عمومی پیدایش این مشکل، روش نمونه برداری در زمان جمع آوری داده است و در صورتی رخ می‌دهد که نمونه‌ها به خوبی معرف توزیع در زمان آزمون نباشند که این امر نیز ناشی از وابسته بودن نمونه‌ها به دسته‌ی آن‌ها است. این نوع عدم تعادل بین داده‌های هر دسته می‌تواند در زمان آزمون دچار تغییر نوع اول یعنی تغییر توزیع داده‌ها در زمان یادگیری و آزمون شود با این تفاوت که این تغییر ناشی از تغییر **طراحی‌شده** در زمان جمع‌آوری داده است.

برای مثال، اگر ما همیشه از گوشه‌ی اتاق برای حرکت ربات با روش epsilon-greedy نمونه برداری کنیم طبیعی است که حرکت‌هایی که منجر به حرکت رو به جلو می‌شوند در داده‌های جمع آوری شده کمتر از حرکات چرخش باشند و این نوعی نمونه برداری متعصبانه است که منجر می‌شود زمانی که در وسط اتاق هم باشیم حرکت رو به جلو نداشته باشیم.



شکل ۵.۳: اگر فقط از دسته‌ی وسط اتاق که حرکت رو به جلو باعث برخورد نمی‌شوند نمونه برداری کنیم باعث می‌شود زمانی که گوشه‌ی اتاق هستیم هم حرکت‌های رو به جلوی اشتباهی داشته باشیم.

به طور خلاصه **نمونه برداری متعصبانه**[۲] باعث می‌شود داده‌ی جمع‌آوری شده در زمان آزمون از توزیع $P(sel = ۱|x, y)$ پیروی کند و و داده‌ی تست از توزیع $P(x, y)$ پیروی کند و این اتفاق اکثرا باعث کاهش کارایی شود. برای کاهش تاثیراحتمال انتخاب شدن (x, y) می‌توان پس از جمع‌آوری داده، با استفاده از روش‌های افزودن داده با توجه به داده‌های پیشین، داده‌ی جدید تولید کرد و تعداد داده‌ها را متعادل کرد.

---

[۱] $Exploration$

[۲] Sample Selection Bias

۳.۳ تشخیص تغییر توزیع داده

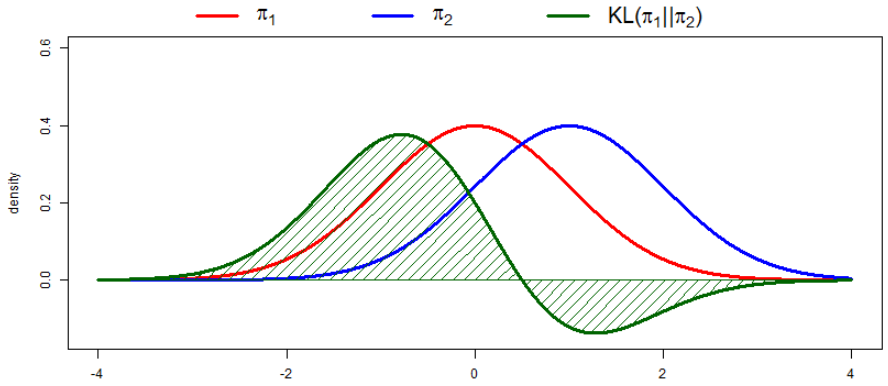
یکی از مسائل مهم پس از شناخت انواع تغییر در توزیع داده این است که در هر مجموعه داده در ابتدا بتوانیم این تغییرات و شدت آن‌ها را تشخیص دهیم و سپس آن را برطرف کنیم.
تشخیص این تغییرات به این دلیل مهم است که ممکن است در صورت استفاده از روش‌های خنثی کردن اثر تغییرات به مدلی ضعیف‌تر از مدل ساده و بدون در نظر گرفتن تغییر توزیع برسیم. این اتفاق به دلایل زیر محتمل است:

- در نظر گرفتن احتمال تغییر در توزیع داده مثل یک نویز روی داده‌ی واقعی عمل می‌کند و ممکن است باعث شود مدل به توزیع مختلفی همگرا شود و به ویژگی‌های کم اهمیتی توجه کند.
- نقص روش‌های موجود برای کاهش تاثیر تغییر توزیع داده ممکن است مدل را در یک مینیمم محلی متوقف کند.

اکنون به تشریح بعضی از راه‌های کاربردی تشخیص تغییر توزیع داده می‌پردازیم:

۱.۳.۳ فاصله‌ی آماری

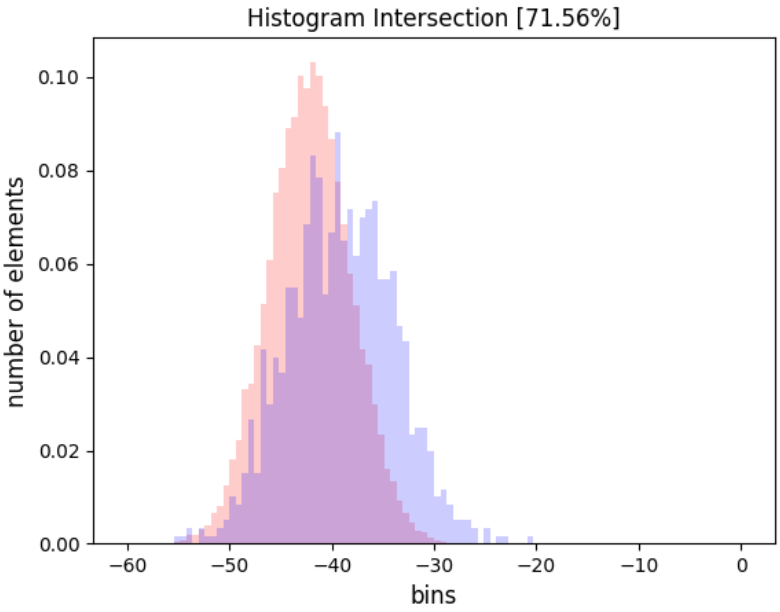
این روش بر توزیع خروجی مدل در طول زمان تکیه می‌کند، به این صورت که خروجی‌های شبکه را در زمان یادگیری و در زمان آزمون ثبت می‌کند و از طریق معیارهایی مثل اشتراک هیستوگرام، واگرایی KL[۱] و آزمون کولموگروف ـ اسمینورف و... به بررسی توزیع خروجی مدل در زمان یادگیری و آزمون می‌پردازد.
برای مثال اگر بخواهیم دو توزیع π_1 و π_2 را مقایسه کنیم :



شکل ۶.۳: هرچه دو توزیع از هم فاصله بگیرند KL بیشتر می‌شود

[۱] KL- Divergence

یا به همان شکل برای اشتراک هیستوگرام:



شکل ۷.۳: از آنجا که اشتراک دو هیستوگرام زیاد نیست، می‌توان گفت تغییر توزیع محسوس است.

همچنین می‌توان این مطالعه را روی توزیع هرلایه‌ی دیگری از مدل از جمله لایه‌های میانی یا ورودی و خروجی شبکه انجام داد و این روش به خوبی نیاز ما را برای تشخیص تغییر توزیع داده برآورده می‌کند اما عیب بزرگ این روش، مناسب نبودن برای مسایل پردازش تصویر است که دارای ویژگی‌هایی با ابعاد بالا است و همین دلیل باعث شده از روش‌های جایگزین بعدی استفاده شود.

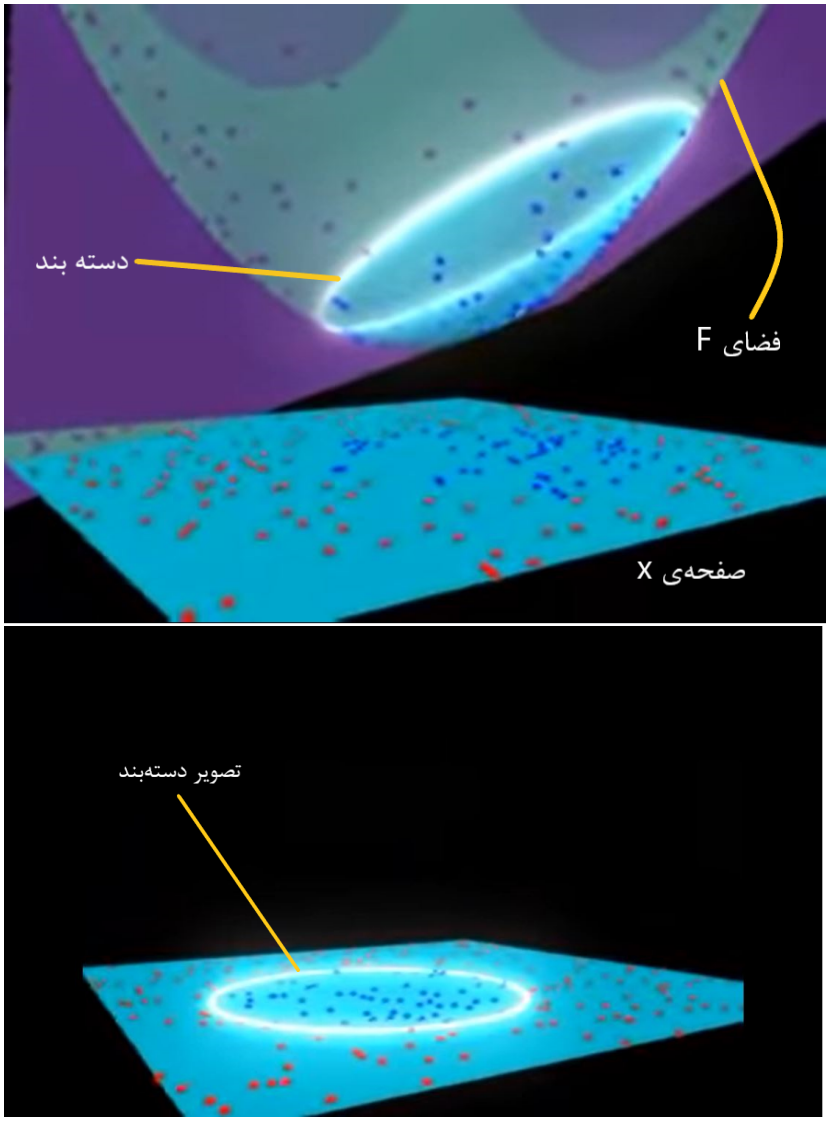
۲.۳.۳ تشخیص تازگی

یکی از راه‌های تشخیص تغییر توزیع داده در محیط‌های پیچیده روش تشخیص تازگی است. ایده اصلی این روش ساختن مدلی برای تخمین احتمال تعلق یک داده جدید به یک توزیع است.

یکی از روش‌های تشخیص داده‌های تازه یا پرت، ماشین بردار پشتیبان یک دسته‌ای [۱] است که یک روش یادگیری بدون هدایت[۲] است و فقط بر روی داده‌های نرمال که از توزیع اصلی آمده اند آموزش داده شده و محدوده‌ی داده‌های توزیع مورد نظر را یاد می‌گیرد و می‌تواند داده‌ها را با توجه به توزیع داده شده دسته بندی کند. همانطور که گفته شد اگر بخواهیم داده‌های پیچیده و با ابعاد بالا را بررسی کنیم می‌توان از ماشین بردار پشتیبان استفاده کرد . برای نمونه یک ماشین بردار دو دسته‌ای را بررسی می‌کنیم.

[۱] One-class Support Vector Machine
[۲] Unsupervised

ابتدا یک مجموعه‌ی داده به صورت $\Omega = \{(x_۱, y_۱), (x_۲, y_۲), ..., (x_n, y_n)\}$ در نظر بگیرید که $x_i \in \mathbb{R}^d$ و $y_i \in \{-۱, ۱\}$ که y_i بیانگر دسته‌ی مقصد است. روش ماشین بردار پشتیبان به این صورت عمل می‌کند که داده‌ی x_i را به یک فضای با ابعاد بالاتر F تصویر می‌کند که در آن فضا مرزبندی بین x_i ها راحت تر است و در آن فضا با یک ابر صفحه آن هارا جدا می‌کند و تصویر آن ابرصفحه در فضای اصلی x_i ها را به عنوان جدا کننده و دسته بند[۱] در نظر می‌گیرد. این صفحه به فرم $w^T + b = ۰$ است که $w \in F, b \in \mathbb{R}$ است. برای مثال یک نمونه‌ی دو بعدی ماشین بردار پشتیبان در شکل زیر آمده است:



شکل ۸.۳: ماشین بردار پشتیبان دو بعدی

پس در فضاهای با ابعاد بالا برای تشخیص تغییر توزیع داده، استفاده از روش ماشین بردار پشتبیان به جای روش‌های آماری پیشنهاد می‌شود اگرچه این روش هم اطلاع دقیقی از اینکه چه چیزی تغییر داده شده نمی‌دهد و

[۱] $classifier$

تنها وجود یا عدم وجود تغییر را تشخیص می‌دهد.

۴.۳ جمع‌بندی

به دلیل بحرانی بودن اکثر سیستم‌های خودران، در موارد بسیاری محیط آموزش و آزمون متفاوت هستند و در نتیجه با تغییر توزیع داده روبرو هستیم. در بین انواع تغییر توزیع داده‌های مذکور، تغییر در برچسب و ویژگی را در زمان آزمون می‌توان حل کرد ولی مشکل داده‌ی نامتعادل در زمان آزمون را نمی‌توان حل کرد.
به همین دلیل تغییر توزیع بین زمان یادگیری و آزمون موضوع به بسیار مهمی در یادگیری عمیق[۱] و یادگیری کمکی[۲] تبدیل شده است به نحوی که در کنفرانس‌های مطرح سال‌های اخیر مقاله‌های بسیاری درباره مسایل انتقال یادگیری[۳] شامل تطبیق دامنه[۴] و $Adverserial Robustness$ منتشر شده است که سعی در پیشبینی ورودی در زمان آزمون دارند و به نحوی می‌خواهند از داده‌های یادگیری به تعمیم پذیری[۵] برای داده‌های آزمون برسند. در فصل بعد به مسایل انتقال یادگیری می‌پردازیم.

[۱] Deep Learning
[۲] Reinforcement Learning
[۳] Transfer Learning
[۴] Domain Adaptation
[۵] Generalization

فصل ۴

انتقال یادگیری

انتقال یادگیری[۲] به دنبال فهمیدن چگونگی یادگیری از سناریوهای مربوط مختلف است به نحوی که به پیشبینی‌ای بهتر از یادگیری روی یک سناریو خاص برای همان سناریو برسیم برای مثال فرض کنید در حالی که دوچرخه سواری و موتور سواری آموخته‌ایم، می‌خواهیم رانندگی ماشین را با استفاده از انتقال یادگیری بیاموزیم در این صورت انتقال یادگیری به شما کمک می‌کند از دانسته‌های مرتبط پیشین خود در رانندگی برای یادگیری رانندگی استفاده کنید. بنابراین تغییر توزیع و انتقال یادگیری به دلیل شباهت توزیع‌هایی که در زمان یادگیری و آزمون داریم، دو موضوع بسیار نزدیک به یکدیگر هستند. در این بخش مقاله‌ی بررسی روش‌های انتقال یادگیری [۲] کمک شایانی به این تحقیق کرده است.

۱.۴ تعریف

انتقال یادگیری یک روش یادگیری ماشین است که از مدلی که یک مهارت را یاد گرفته، به عنوان نقطه شروع برای یادگرفتن مهارت مرتبط دیگری استفاده می‌کند. ایده‌ی این موضوع از روش یادگیری انسان‌ها سرچشمه می‌گیرد. انسان‌ها همواره از مهارت‌های قبلی خود در یادگیری مهارت‌های جدید استفاده می‌کنند و اکثرا هیچ مهارتی را از اول یاد نمی‌گیرند.

در واقع می‌توان یک مسئله‌ی انتقال یادگیری را به این صورت تعریف کرد که D_s دامنه‌ی مبدا و T_s مهارت هدف یادگیری است و از طرف دیگر D_t دامنهی مقصد با مهارت تعریف شده‌ی T_t است. هدف انتقال یادگیری، کمک به یادگیری تابع پیشبینی مقصد یا $f_t(.)$ در D_t با استفاده از دانش D_s و T_s است به شرطی که $D_s \neq D_t$ یا $T_s \neq T_t$ باشد.

---

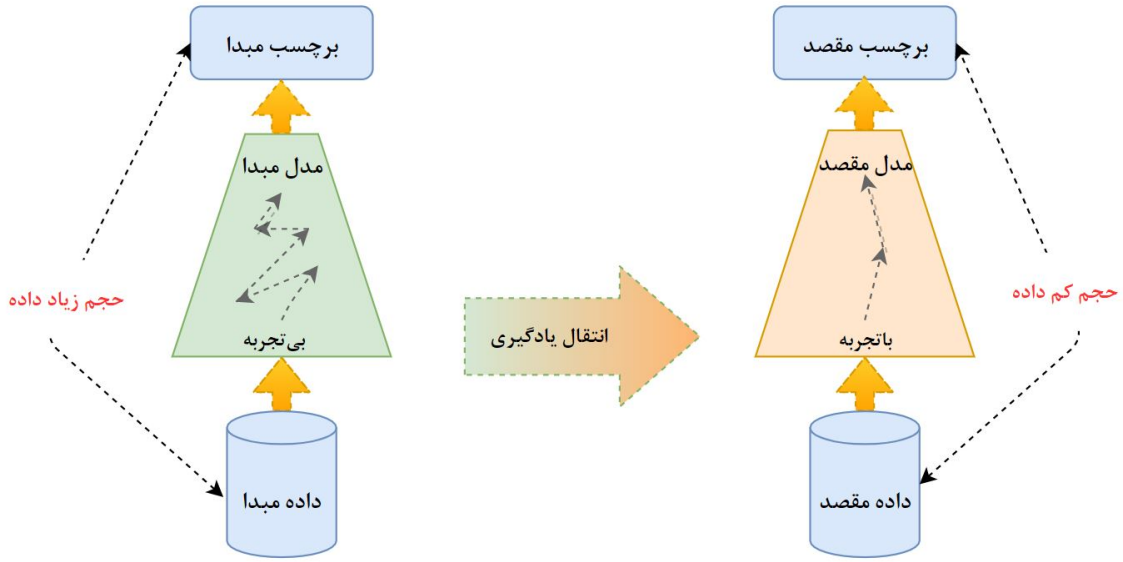
[۲] $Transfer Learning$

۲.۴ مقایسه‌ی یادگیری ماشین سنتی و انتقال یادگیری

اولین نکته‌ای که باید در ذهن داشته باشیم این است که انتقال یادگیری .یک مفهموم جدا از یادگیری ماشین نیست. یادگیری ماشین سنتی سعی دارد با استفاده از یک مجموعه داده‌ی به خصوص یک مهارت خاص را بدون هیچ دانسته‌ی پیشینی فراگیرد در حالی که انتقال یادگیری سعی می‌کند از تجربه‌ی گذشته‌اش استفاده کند و دانش پیشین مرتبط خود را برای یادگیری مهارت جدید انتقال دهد و با داده‌ی کمتری از روش سنتی مهارت جدید را یاد بگیرد.

۳.۴ کاربرد انتقال یادگیری در یادگیری عمیق

هدف یادگیری عمیق پی‌بردن به نگاشتی از ورودی به یک فضای نهان و سپس پیشبینی خروجی مطلوب از آن فضای نهان با داشتن حجم زیادی از داده است. از آنجا که یادگیری عمیق به داده‌ی زیادی نیاز دارد و لزوما برای هر مهارت جمع آوری داده‌های زیاد ممکن نیست، انتقال یادگیری این مکان را برای ما فراهم می‌کند که با داده‌ی کم از مهارت مقصد یک مهارت جدید مرتبط با مهارت اول را یاد بگیریم و از طرف دیگر به دانش قبلی قابلیت تعمیم‌پذیری اضافه کنیم.



شکل ۱.۴: یادگیری عمیق مهارت جدید با حجم کم داده مقصد

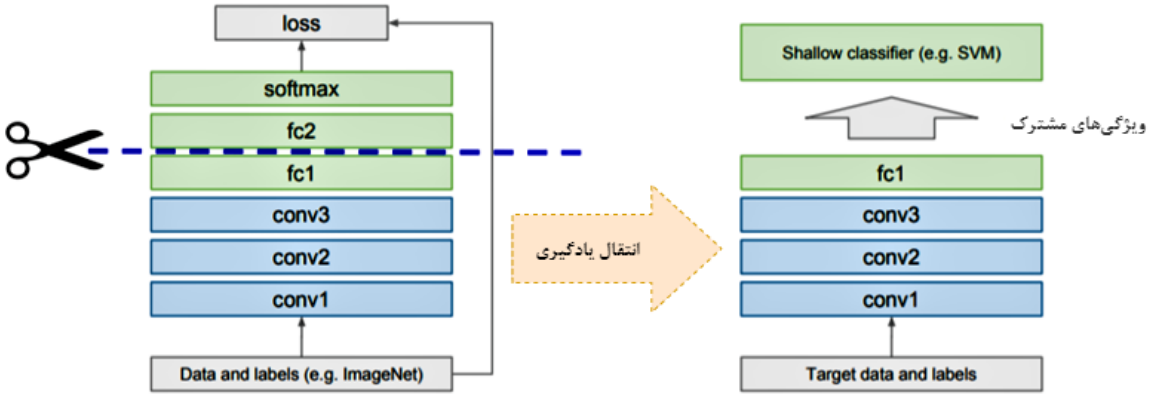
همانطور که می‌دانید مدل‌های یادگیری عمیق زمان و داده‌ی بیشتری نسبت به مدل های سنتی یادگیری ماشین برای آموزش نیاز دارند و از طرف دیگر رشد چشمگیری نسبت به مدل‌های پیشین داشته اند و در بعضی موارد از دقت انسان نیز پیشی گرفته اند. مدل‌های یادگیری عمیق از لایه‌های بسیاری تشکیل شده اند و هر لایه وظیفه‌ی یادگیری یک ویژگی خاص در تصویر (مثلا لبه، گردی و ...) را دارد. امروزه در دنیای پردازش تصویر مدل‌های

بسیاری برای مهارت‌های مختلف ساخته شده اند که کارایی بی‌نظیری دارند، ما از این مدل‌های با تجربه به عنوان پایه‌ای برای انتقال یادگیری استفاده می‌کنیم و با استفاده از عمیق تر کردن آن مدل‌ها به دنبال یادگیری مهارت‌های جدید هستیم به همین دلیل از این روش با نام انتقال عمیق نیز می‌توان یاد کرد. در ادامه به چند روش انتقال عمیق می‌پردازیم.

۱.۳.۴ استفاده از مدل‌های آماده به عنوان استخراج‌کننده‌ی ویژگی

همانطور که گفته شد مدل‌های یادگیری عمیق از لایه‌های زیادی تشکیل شده اند که این لایه‌ها، ویژگی‌های معنایی از تصویر استخراج می‌کنند و در انتها خروجی عمیق‌ترین لایه به یک یا چند لایه‌ی تصمیم گیرنده‌ی نهایی(کاملا متصل[۱]) که به اندازه‌ی بعد ویژگی‌های عمیق ترین لایه ورودی می‌گیرد و به اندازه‌ی بعد برچسب‌ها خروجی می‌دهد وصل می‌شود. پس می‌توان دریافت هرچه عمیق‌تر شویم ویژگی‌ها به مهارتی که می‌خواهیم آموزش دهیم بیشتر ارتباط پیدا می‌کنند و در لایه‌های کم عمق تر ویژگی‌های عمومی‌تری درحال استخراج هستند در نتیجه برای انتقال یادگیری، لایه های کم‌عمق‌تر برای مهارت‌های مرتبط می‌توانند مفید باشند زیرا لایه‌های عمیق بیشتر تمرکز بر مهارت اصلی دارند.

پس در این روش از انتقال یادگیری ابتدا از لایه‌های کم‌عمق‌ترِ یک شبکه‌ی آموزش داده شده‌ای که نتایج خوبی دارند(مثل $Inception$ یا $ResNet$) استفاده می‌کنند تا دانش پیشین را منتقل کنند ضمن اینکه وزن‌های آن ها را دیگر بهروز نمی‌کنند و سپس چند لایه‌ی جدید در ادامه‌ی آن‌ها میگذارند تا مهارت جدید را با استفاده از این لایه‌های جدید یاد بگیرند .

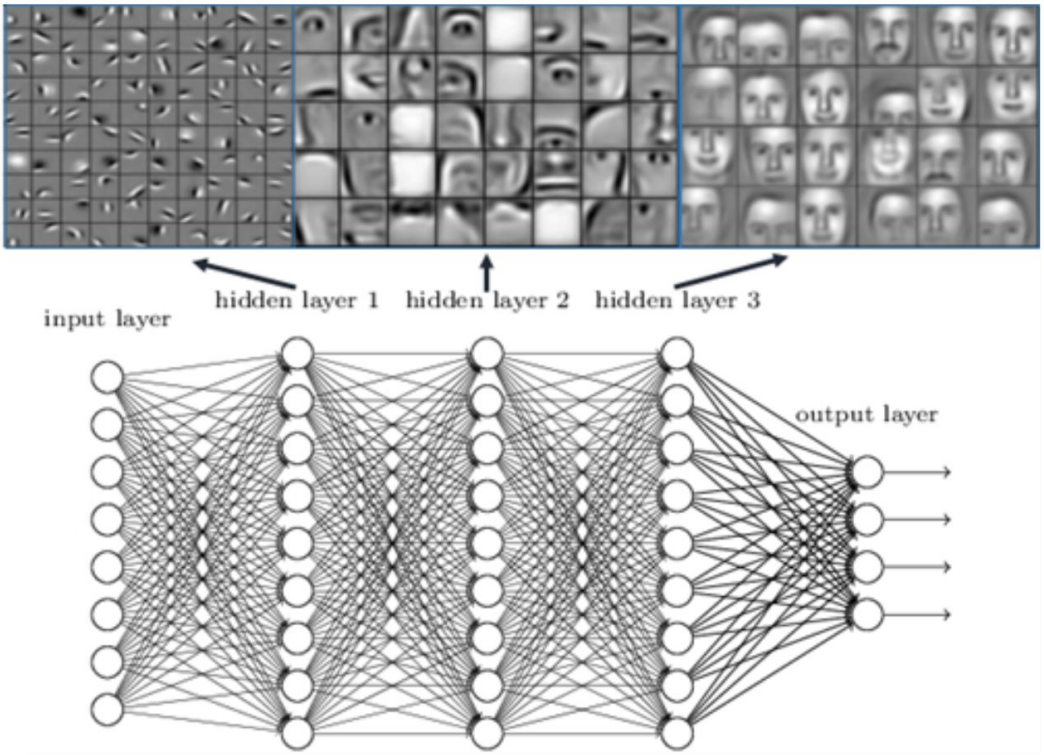


شکل ۲.۴: استفاده از مدل آماده به عنوان استخراج کننده‌ی ویژگی

[۱] $FullyConnected$

۲.۳.۴ استفاده از مدل‌های آماده به عنوان وزن اولیه مدل

این روش نیز شباهت بسیاری به روش قبل دارد تنها با این تفاوت که وزن‌های مدل آماده را هم به‌روز می‌کنیم. این روش در زمان‌هایی می‌تواند مفید باشد که ویژگی‌های مشترک مقصد تفاوت اندکی با ویژگی‌های مبدا داشته باشند برای مثال چرخش تصویر در مهارت مقصد تاثیر داشته باشد و در مهارت مبداء تاثیری نداشته و حتی از افزودگی چرخش هم در زمان آموزش مهارت اول استفاده شده باشد. در این صورت این روش را به عنوان انتقال داده استفاده می‌کنیم و با اجازه دادن به به‌روز شدن وزن‌های اولیه، با یک تغییر کوچک می‌توانیم به هدف‌مان (مهارت مقصد) برسیم.



شکل ۳.۴: شهود ویژگی‌ها نسبت به عمیق شدن لایه‌ها

۴.۴ انواع انتقال عمیق

مسایل یادگیری به کلی شامل حل یک مهارت جدید با دانستن یک مهارت از یک دامنه است. اشتراک این یادگیری جدید و تجربه‌ی پیشین مدل می‌تواند در دو حالت مختلف باشد:

- دامنه‌ی یکسان و مهارت‌های مختلف
- دامنه‌ی مختلف و مهارت‌های یکسان

در ادامه قصد داریم کلیت روش‌های مرسوم که از این دو حالت منشعب شده اند را بررسی کنیم:

۱.۴.۴ تطبیق دامنه

این روش زمانی استفاده می‌شود که $P(X_s) \neq P(X_t)$ باشد. در واقع این روش سعی در حل مشکل تغییر توزیع دامنه که در فصل قبل بررسی شد دارد و امروزه یکی از پر کاربرد ترین روش‌های انتقال عمیق است. در بخش های بعد به این موضوع بیشتر می پردازیم.

۲.۴.۴ آشفته‌کردن دامنه

این روش سعی در یاد گرفتن ویژگی‌هایی مستقل از دامنه دارد و از این طریق می‌تواند بین دامنه‌ها قابلیت انتقال داشته باشد

۳.۴.۴ یادگیری چندگانه

این روش سعی در یادگیری همزمان چند مهارت با دامنه‌ی یکسان دارد. معمولا آموزش مدل این روش به دلیل متفاوت بودن برچسب‌ها یا پاداش ها سخت‌تر از حالت عادی است.

۴.۴.۴ یادگیری با داده‌ی کم یا بدون داده

شاید بزرگترین مشکل یادگیری عمیق، نیاز به داده‌ی بسیار زیاد باشد ولی موضوعی که امروزه به کمک پیشرفت Meta Learning راحت تر شده موضوع یادگیری با داده‌ی بسیار کم[۱] یا بدون داده است. در این روش جالب هدف یاد گرفتنِ چگونه یاد گرفتن است.

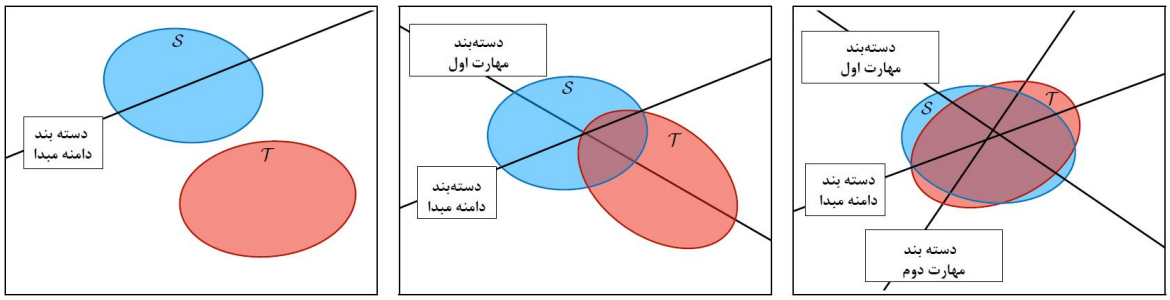
در ادامه چند مقاله‌ی مرتبط با کاربرد انتقال یادگیری برای غلبه بر تغییر توزیع داده در زمان یادگیری و آزمون را بررسی می‌کنیم.

[۱] few-shot learning

۵.۴ کارهای مرتبط

۱.۵.۴ تطبیق دامنه‌ی بدون هدایت از طریق نظارت خودآگاه

هدف این مقاله[۵] حل تغییر توزیع تصویری است و تنظیمات به این شکل است که داده‌ی برچسب‌دار از دامنه‌ی مبدا موجود است ولی داده‌ی مقصد بدون برچسب هستند و می‌خواهیم تنها با یادگیری روی دامنه‌ی مبدا به کارایی خوبی روی دامنه‌ی مقصد برسیم. در ضمن مهارت هدف آموزش هم بین هر دو محیط یکسان است.
ایده‌ی آن‌ها، یادگیری راهی برای تبدیل دامنه‌ی مبدا به مقصد به وسیله‌ی آموزش یک مهارت یکسان بر روی هر دو دامنه است. **برای همین منظور از مهارت‌هایی با برچسب‌های خودآگاه استفاده می‌کنند که هدف این روش ساختن داده‌های جدید برای آموزش مهارت‌های مرتبط و کمکی است که به کمک این مهارت‌ها می‌توانند مدلی با قابلیت تعمیم پذیری روی هم دو دامنه بدست آورند و وابستگی کمتری به دامنه داشته باشند.** برای مثال به شکل زیر دقت کنید:
در ادامه k مهارت کمکی خودآگاه را انتخاب می‌کنند و برای هر یک نیز تابع خسارت[۱] تعریف می‌کنند که خسارت



الف) دسته‌بند مهارت دامنه مبدا ب) اضافه کردن یک مهارت خودآگاه پ) اضافه کردن مهارت بیشتر

شکل ۴.۴: در شکل "الف" دسته‌بند دامنه مبدا برای دامنه مقصد کارایی ندارد و پس از اضافه کردن مهارت‌های کمکی در شکل "ب" و "پ" می‌بینیم که وزن لایه‌ها به گونه‌ای تغییر می‌کنند که دو دامنه را به فضای نزدیک‌تری تصویر می‌کنند.

مهارت iم را با L_i نشان می‌دهند و ضمنا L، نشان دهنده‌ی خسارت مهارت مشترک اصلی است. بنابراین با یک یادگیری چندگانه روبرو هستیم که $k+1$ خسارت دارد. ساختار شبکه نیز به این شکل است که یک قسمت از شبکه با وزن های مشترک بین همه‌ی شاخه‌ها مشترک است و بعد از آن شبکه تقسیم به $k+1$ شاخه‌ی خروجی به عنوان پیشبینی کننده‌ی مختص به هر مهارت می‌شود که این شاخه‌ها ساختار کوچکی دارند و با h_k نشان داده می‌شوند .

---
[۱] $loss$

فرآیند یادگیری

فرض کنید $S = \{(x_i, y_i), i = ۱...m\}$ شامل داده‌های برچسب‌دار موجود باشد و $T = \{(x_i), i = ۱...n\}$ شامل داده‌ی بدون‌برچسب از دامنه‌ی مقصد باشد. ضمنا از آنجا که فرض کردیم مهارت‌های کمکی به صورت خودآگاه هستند پس برای هرکدام یک تابع F_k داریم که به ازای هر x_i یک زوج $(f_k(x_i), \tilde{y}_i)$ تولید می‌کند.
بنابراین توابع خسارت به صورت زیر تعریف می‌شوند:
تابع خطای مهارت اصلی روی داده‌ی برچسب دار مبدا:

$$\mathcal{L}_۰(S; \phi, h_۰) = \sum_{(x,y)\in S} L_۰(h_۰(\phi(x)), y).$$

تابع خطای مهارت‌های کمکی روی همه‌ی داده‌های دامنه‌ی مبدا و مقصد:

$$\mathcal{L}_k(S, T; \phi, h_k) = \sum_{(f_k(x),\tilde{y})\in F(S)} L_k(h_k(\phi(f_k(x))), \tilde{y}) + \sum_{(f_k(x),\tilde{y})\in F(T)} L_k(h_k(\phi(f_k(x))), \tilde{y}).$$

دقت کنید که در خسارت $\mathcal{L}_k$ از هر دو داده‌ی مبدا و مقصد استفاده می‌شود ولی در $\mathcal{L}_۰$ تنها از داده‌ی برچسب‌دار S می‌توان استفاده کرد و در نهایت هدف بهینه‌سازی[۱] به فرم زیر نوشته می‌شود:

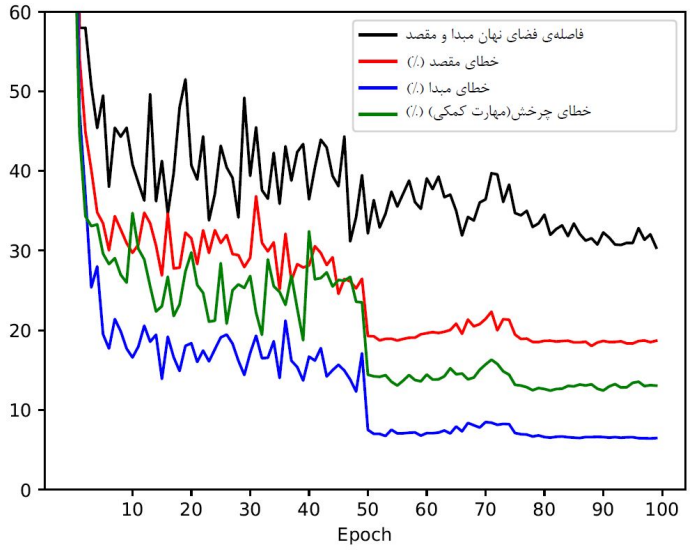
$$\min_{\phi, h_k, k=۱...K} \mathcal{L}_۰(S; \phi, h_۰) + \sum_{k=۱}^{K} \mathcal{L}_k(S, T; \phi, h_k).$$

که به معنی بدست آوردن وزن‌های مشترک و وزن‌های شاخه‌هایی است که هر دو خسارت را همزمان کمینه کنند. ضمنا پس از انجام بهینه‌سازی دیگر با شاخه‌های کمکی کاری ندارند و فقط $h_۰$ را در زمان آزمون استفاده می‌کنند، که البته در مقاله‌ی **یادگیری در زمان آزمون**[۲] اینکار را بیشتر بررسی می‌کنیم.
ایده‌ی کاربردی دیگری که در گذشته هم استفاده شده بوده را برای تنظیم ابرپارامترها[۳] و توقف زودهنگام[۴] استفاده کرده اند به این ترتیب که از خروجی شبکه‌ی مشترک برای نزدیک کردن دامنه‌ی S و T به صورت زیر استفاده می‌کنند:

$$D(S', T'; \phi) = \left\| \frac{۱}{m} \sum_{x\in S'} \phi(x) - \frac{۱}{n} \sum_{x\in T'} \phi(x) \right\|_۲$$

[۱] Optimization
[۲] Test-Time Training
[۳] Hyperparameter Tuning
[۴] Early Stopping

سپس کمینه‌ی ایپاک را از ترکیب خطی داده‌های اعتبارسنجی برچسب‌دار با $D(S', T'; \phi)$ بدست می‌آورند (خط سیاه در نمودار ۵.۴) و برای توقف‌زودهنگام استفاده می‌کند.



شکل ۵.۴: نمودار خطا حین یادگیری روی مجموعه‌داده‌ی $CIFAR-۱۰ \rightarrow STL-۱۰$

۲.۵.۴ یادگیری در زمان آزمون

این مقاله[۴] از نظر استفاده از مهارت کمکی برای نزدیک کردن همزمان تصویر دامنه‌ی مبدا و مقصد استفاده می‌کند. اما تفاوت و علت برتری این روش نسبت به روش قبل، بهره بردن از آموزش با توزیع داده‌ی آزمون است. مزیت بزرگ دیگری که این روش را عملی‌تر نیز کرده، نیاز نداشتن به داشتن اطلاعاتی از داده‌ی مقصد است درحالی‌که روش قبل به داده‌های مقصد نیز در زمان یادگیری دسترسی داشت. از طرف دیگر این ویژگی باعث میشود در زمان آزمون نه فقط برای یک دامنه‌ی مقصد بلکه برای هر دامنه‌ی مشابه دیگری قابلیت تعمیم‌پذیری داشته باشیم و عملا مدلی که آموزش دیده می‌تواند به حالت‌های مختلفی پیشبینی کند.

در واقع خطای آزمون را از $E_Q[l(x,y);\theta]$ به $E_Q[l(x,y);\theta(x)]$ تبدیل می‌کند که این به معنی آن است که دیگر به دنبال حدس زدن تغییر توزیع در زمان یادگیری نیستیم. همچنین این ایده مانع از کاهش کارایی ناشی از اضافه کردن قابلیت تعمیم‌پذیری به مدل می‌شود.

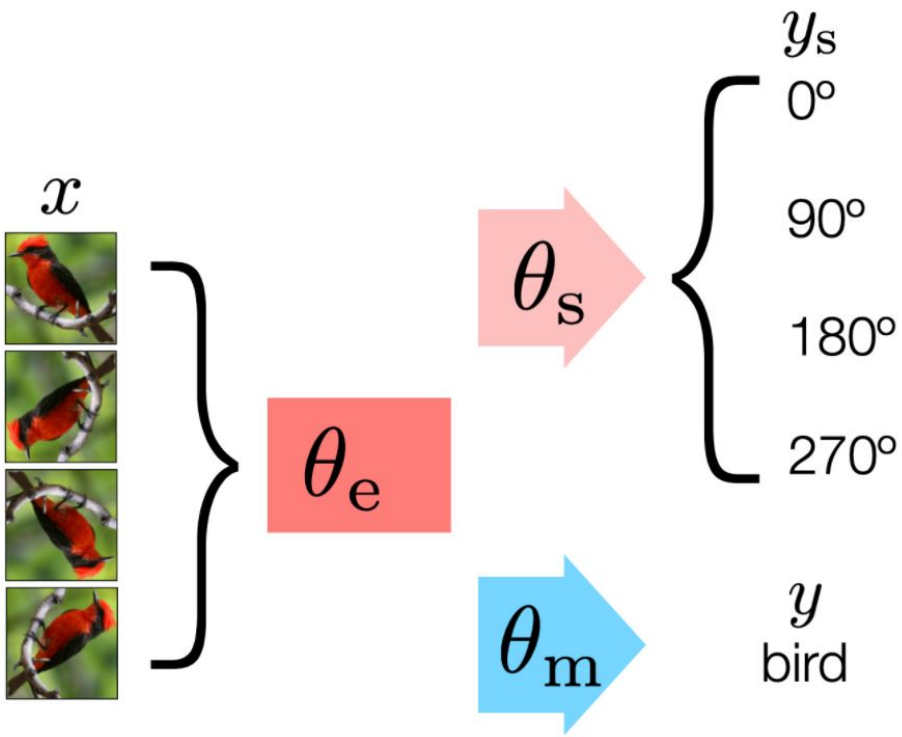
فرآیند یادگیری

ساختار شبکه‌ی این روش به این شکل است که ابتدا یک شبکه‌ی عصبی K ـ لایه دارد که وزن هر لایه را با θ_k نشان می‌دهند و سپس به یک شاخه برای مهارت اصلی با وزن‌های θ_m و یک شاخه‌ی دیگر برای مهارت کمکی

خودآگاه با وزن‌های θ_s تقسیم می‌شود. اگر فرض کنیم داده‌هایمان به شکل (x_i, y_i) از توزیع P هستند، هدف کلی بهینه‌سازی کمینه کردن تابع زیر است:

$$\min_{\theta_e,\theta_m,\theta_s} \frac{1}{n}\sum_{i=1}^{n} l_m(x_i, y_i; \theta_m, \theta_e) + l_s(x_i; \theta_s, \theta_e)$$

و برای مقداردهی اولیه‌ی شبکه‌ی مشترک از مهارت خودآگاه که آسان‌تر از مهارت اصلی است کمک می‌گیریم و سعی می‌کنیم کمینه‌ی θ_e را با استفاده از $l_s(x; \theta_s, \theta_e)$ بدست آوریم که فرض کنید مقدار بهینه‌ی θ_e^* بدست آمد، حالا برای مقداردهی اولیه‌ی θ_e از θ_e^* استفاده می‌کنیم. این روش برای مقداردهی اولیه منجر به سرعت بیشتر و نقطه شروع بهتر در زمان یادگیری می‌شود.



شکل ۶.۴: ساختار شبکه در حالتی که مهارت کمکی، مهارت تشخیص چرخش تصویر است.

بعد از مرحله‌ی آموزش در زمان آزمون، داده‌ها به دو حالت می‌توانند به شبکه داده شوند:

۱. داده‌ها به صورت پشت سر هم از توزیع Q یا از توزیع‌های نزدیک به هم بیایند که در این صورت داده‌ی x_t از داده‌ی x_{t-1} و توزیع داده‌های قبلی که از توزیع مشابهی می‌آیند بهره می‌برد. در این حالت θ_e در

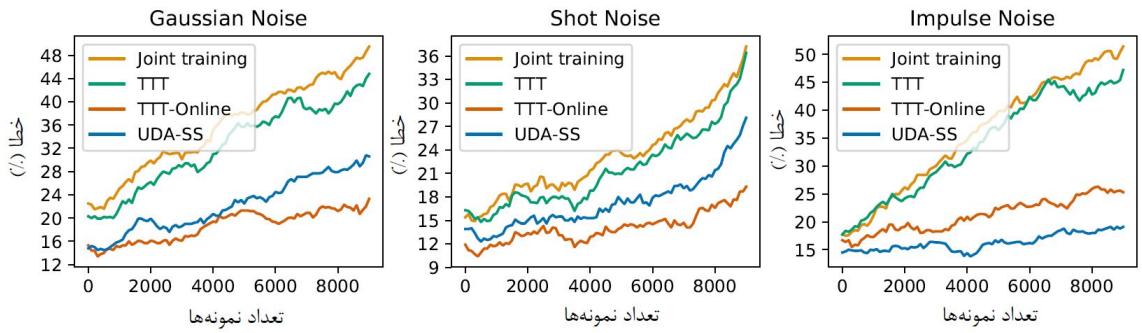
زمان آزمون هم همواره در حال بهینه شدن است. به همین علت نام این روش را روش آنلاین می‌گذاریم.

۲. داده‌ها بدون هیچ پیشفرضی روی توزیع آنها بیایند که در این صورت از توزیع داده‌های قبل نمی‌توان استفاده کرد و θ_e را نمی‌توانیم بهینه کنیم. به همین علت نام این روش را روش استاندارد می‌گذاریم.

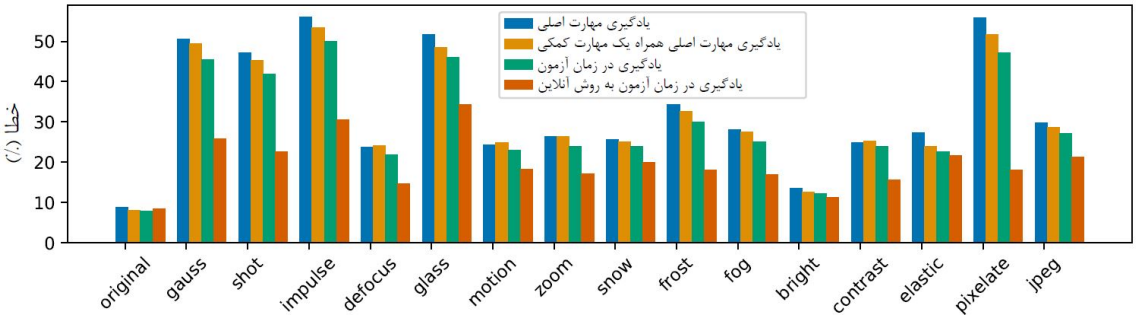
البته برای یک مقایسه‌ی عادلانه باید زمان خروجی را هم در نظر گرفت که به نظر می‌آید بخاطر یادگیری‌ای که در زمان آزمون قرار است انجام شود، روش یادگیری در زمان آزمون کند تر باشد.

نتایج

در نمودارهای زیر نتیجه‌ی مقایسه‌ی این روش با روش ۱.۵.۴ که هر هر دو برای مهارت کمکی چرخش تصویر آموزش دیده اند را در مقابل تغییر توزیع دامنه می‌بینیم:



شکل ۷.۴: مقایسه‌ی خطای آزمون روش‌های نامبرده در مقابل ۳ مرحله شدتِ تغییر توزیع ورودی و تغییر مرحله‌ای توزیغ روی مجموعه‌داده $CIFAR-۱۰$ که نویز آن‌ها به صورت افزایش انحراف معیار با افزایش تعداد نمونه‌ها ساخته شده اند. همانطور که مشاهده می‌کنید روش یادگیری در زمان آزمون آنلاین در اکثر مواقع کمترین خطا را دارد.



شکل ۸.۴: مقایسه‌ی خطای آزمون روش‌های نامبرده در مقابل ۵ مرحله شدتِ تغییر توزیع ورودی روی مجموعه‌داده‌ی $CIFAR-۱۰$. همانطور که مشاهده می‌کنید روش یادگیری در زمان آزمون آنلاین در مقابل همه‌ی تغییرات کمترین خطا را دارد.

واژه‌نامه

آزمون .Test
آموزش .Train
ابرصفحه .Hyperplane
احتمال .Probability
ارزیابی .Validation
افزایش گرادیان Gradient Ascent.
افزودن داده Data Augmentation.
امیدریاضی .Expectation
انتقال یادگیری Transfer Learning.
بحرانی .Critical
برخورد .Collision
برچسب .Label
بیشینه .Maximum
تخمین .Estimate
تشخیص تازگی Novelty Detection.
تصادفی .Random
تطبیق دامنه Domain Adaptation.
تعمیم‌پذیری .Generalization
توزیع آماری .Distribution
خسارت .Loss
خطا .Error
خودران .Autonomous
دامنه .Domain
دسته .Class
دسته‌بند .Classifier

سیاست بهینه .Policy. Optimal
شبکه عصبی Neural Network.
شبیه‌ساز .Simulator
علّی .Causal
غیرمعین Non Deterministic.

ماشین بردار پشتیبان Support Vector Machine.
متعادل .Balanced
مجموعه‌داده .Dateset
محیط .Environment
مسیر حرکت .Trajectory
مسیریابی .Navigation
نامتعادل .Unbalanced
نمونه‌برداری .Sampling
وضعیت .State
ویژگی .Feature
پاداش .Reward
پردازش .Process

کارایی .Performance
کاوش .Exploration
کمینه .Minimum
گرادیان سیاست Policy Gradients.

یادگیری عمیق Deep Learning.
یادگیری کمکی Reinforcement Learning.

مراجع

[۱] Tuomas Haarnoja et al. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. .۲۰۱۸ arXiv: 1801.01290 [cs.LG].

[۲] Sinno Jialin Pan et al. “A survey on transfer learning”. In: IEEE Transactions on Knowledge and Data Engineering .(۲۰۱۰)

[۳] Joaquin Quionero-Candela et al. Dataset Shift in Machine Learning. The MIT Press، .۲۰۰۹ ISBN: .۰۲۶۲۱۷۰۰۵۱

[۴] Yu Sun et al. “Test-Time Training with Self-Supervision for Generalization under Distribution Shifts”. In: ICML. .۲۰۲۰

[۵] Yu Sun et al. Unsupervised Domain Adaptation through Self-Supervision. .۲۰۱۹ arXiv: 1909.11825 [cs.LG].

دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی
هوش مصنوعی

عنوان

بررسی کاربردهای انتقال یادگیری در یادگیری کمکی

نگارش
امیرحسین شهیدزاده

استاد راهنما
دکتر حمیدرضا ربیعی

تابستان ۱۳۹۹