بسمه تعالی

گزارش پنجم

استاد راهنما: دکتر قدیری

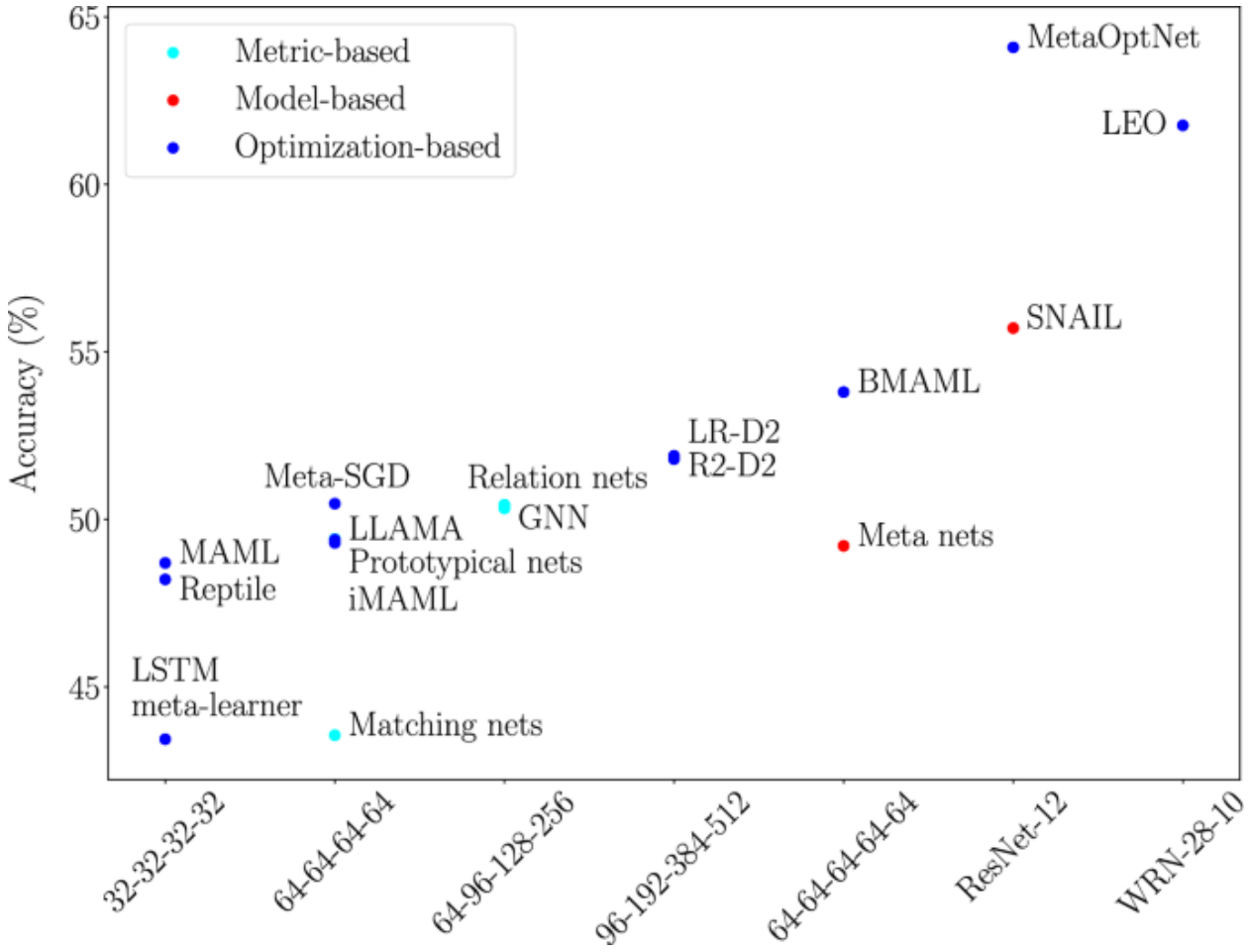
دانشجو: پردیس مرادبیکی

تابستان ۱۴۰۲

به طور کلی روش‌های مبتنی بر meta-learning را به سه دسته تقسیم می‌کنیم:

۱. Metric-based
۲. Model-based
۳. Optimization-based

براساس امتیاز دقت روش‌ها را به صورت زیر دسته بندی می‌کنیم:



روش ها را در زیر شرح می‌دهیم:

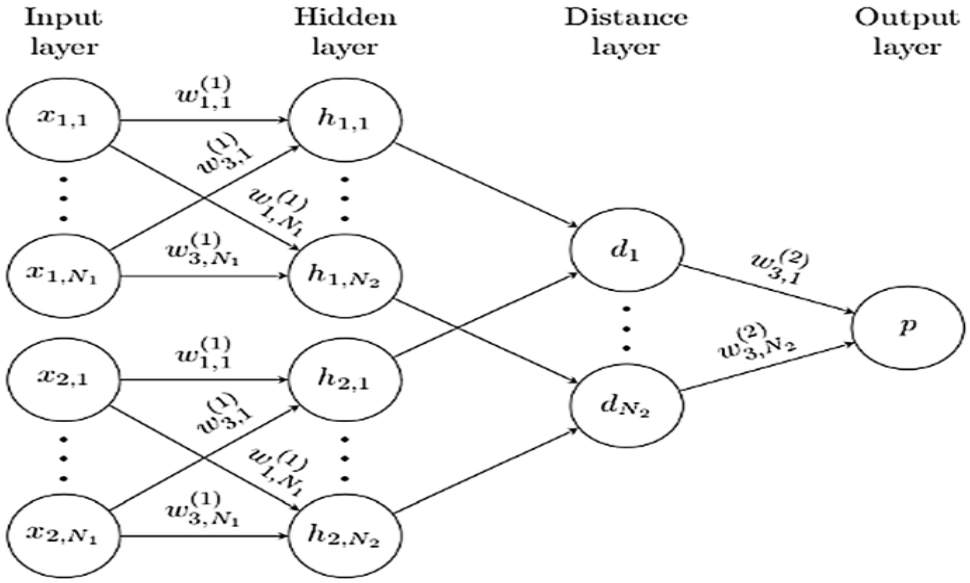
Metric-based meta-learning

این روش‌ها فضای ویژگی training را یاد می‌گیرند و برای پیش‌بینی، شباهت را حساب می‌کنند.

روش‌های مبتنی بر متریک را در زیر به صورت خلاصه شرح می‌دهیم:

۱. Siamese Neural Network: پیش‌بینی کلاس با مقایسه support set و query انجام می‌شود.

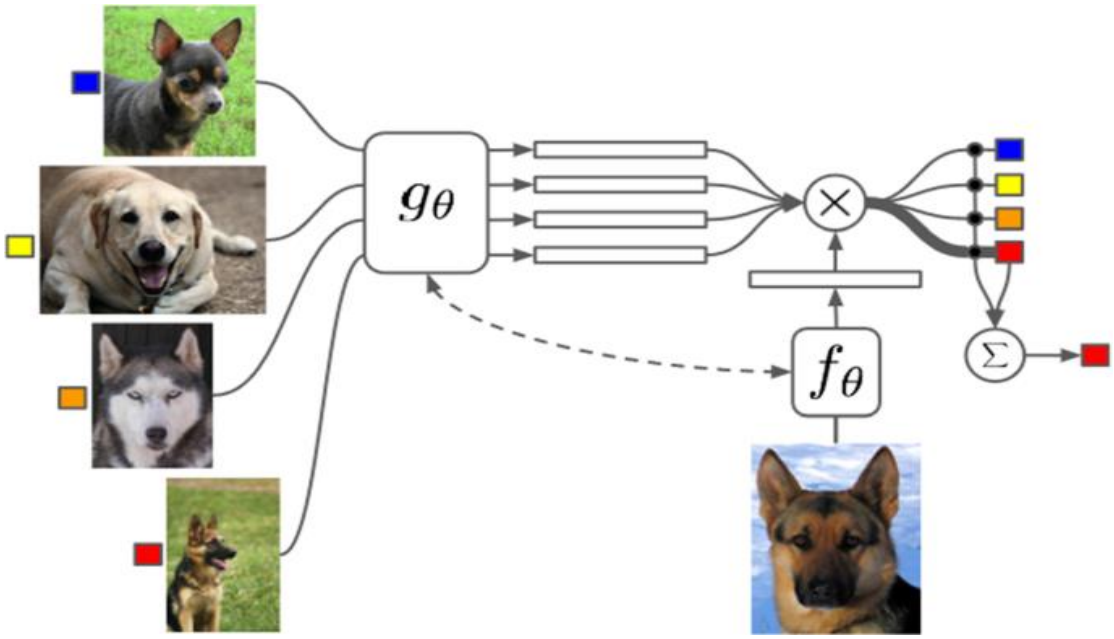
از دو شبکه عصبی با وزن یکسان تشکیل شده است. دو حالت پنهان را محاسبه می‌کنند و این حالت‌های پنهان به یک لایه distance وارد می‌شوند که بردار فاصله را حساب می‌کنند. از این بردار فاصله شباهت ورودی‌ها را حساب می‌کنند.



۲. Matching networks: مستقیم few-shot را آموزش می‌دهد و از شباهت کسینوس به عنوان تابع شباهت استفاده می‌کنند. در این روش از دو تابع embedding استفاده می‌کنیم و f و g را محاسبه می‌کنیم سپس از تابع

$$a(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{e^{c(f_{\boldsymbol{\phi}}(\boldsymbol{x}), g_{\boldsymbol{\varphi}}(\boldsymbol{x}_i))}}{\sum_{j=1}^{m} e^{c(f_{\boldsymbol{\phi}}(\boldsymbol{x}), g_{\boldsymbol{\varphi}}(\boldsymbol{x}_j))}}$$

استفاده می‌کند. هر چه شباهت بین کسینوس بین embeddingها بیشتر باشد a بزرگتر می‌شود.

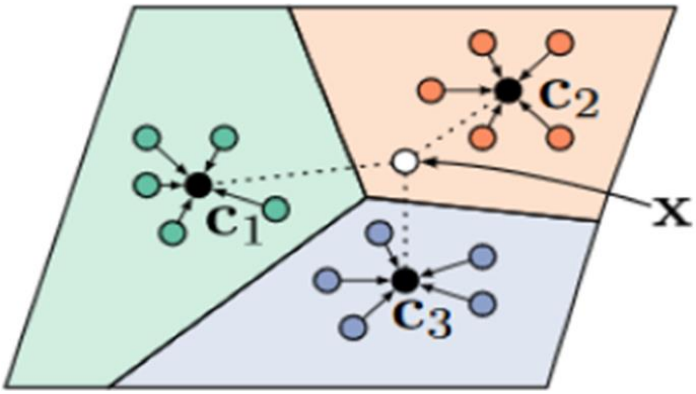


۳. Prototypical networks: تعداد مقایسه ها را کم می‌کند به جای همه نمونه ها در هر دسته با یک نمونه در ان دسته مقایسه انجام می‌شود. و به جای K*N مقایسه N مقایسه انجام می‌شود.

در این روش‌ها یک نقطه میانگین در نمونه های هر کلاس محاسبه می‌شود و فاصله هر کدام کمتر بود برای ان دسته است.

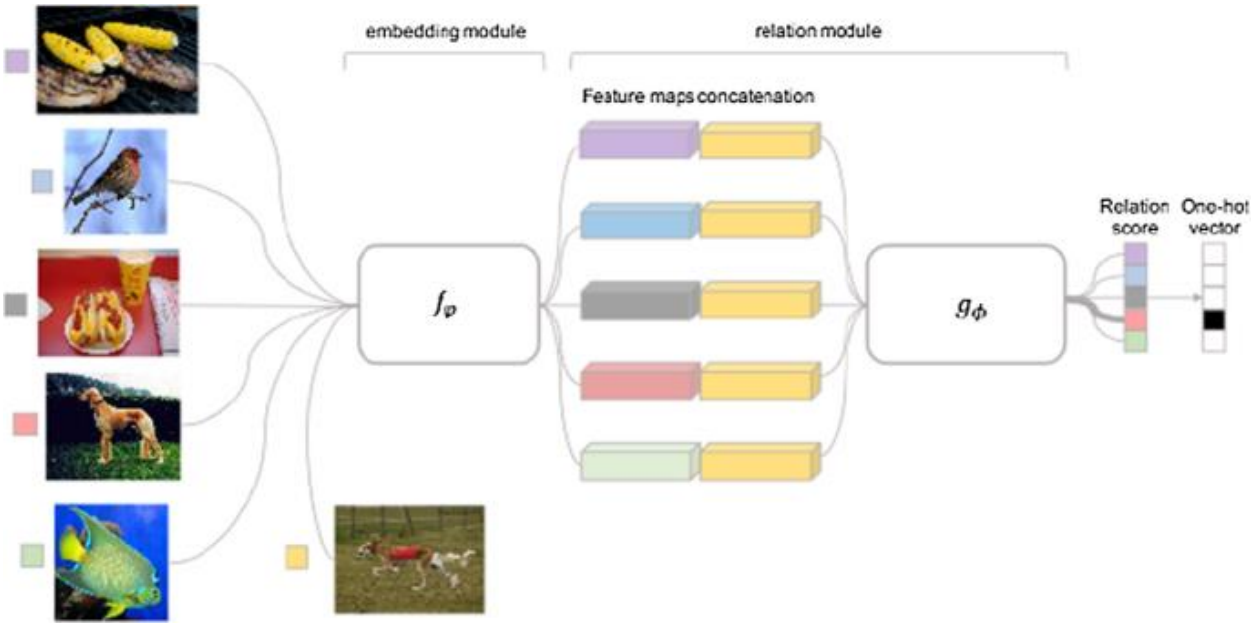
$$p_{\boldsymbol{\theta}}(y=k|\boldsymbol{x}, D^{tr}_{\mathcal{T}_j}) = \frac{exp[-d(f_\theta(\boldsymbol{x}), \boldsymbol{c}_k)]}{\sum_{y_i} exp[-d(f_\theta(\boldsymbol{x}), \boldsymbol{c}_{y_i})]}$$

هر چه این مقدار بیشتر شود به ان کلاس متعلق است.



۴. Relation networks: معیارهای شباهت ثابت و از پیش‌تعریف شده را با یک شبکه عصبی جایگزین می‌کند که امکان یادگیری تابع شباهت خاص در ان دامنه را فراهم می‌کند. به جای یک معیار از پیش

تعریف‌شده (مانند شباهت کسینوس) از یک متریک شباهت قابل آموزش استفاده می‌کنند. شبکه/ماژول تعبیه شده f_ϕ که مسئول جاسازی ورودی‌ها است، و شبکه رابطه g_ϕ که امتیاز شباهت بین ورودی‌های جدید را محاسبه می‌کند. سپس با انتخاب کلاس مناسب که بیشترین امتیاز را دارد طبقه‌بندی می‌شود.

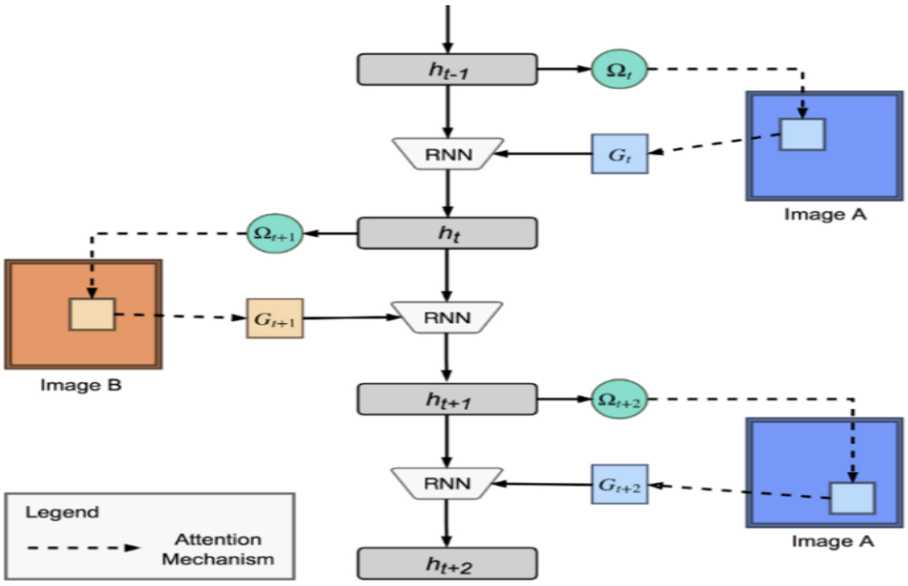


۵. Graph neural networks: جریان اطلاعات بین ورودی‌های support set و query را پارامتری می‌کند.

این روش‌ها نسبت به رویکردهای قبلی انعطاف پذیرتر است. این شبکه‌ها شامل شبکه های Siamese و prototypical می‌شوند. هر task را به صورت یک گراف کاملا متصل می‌بینند. شبکه های عصبی گراف در few-shot settings عملکرد خوبی دارند و همچنین در تنظیمات یادگیری semi-supervised و active learning settings هستند.

۶. Attentive recurrent comparators : ورودی‌ها را به‌عنوان یک کل مقایسه نمی‌کنند، بلکه بر اساس بخش‌ها مقایسه می‌کنند.

در این روش‌ها کل ورودی را مقایسه نمی‌کند بلکه با نگاه کردن به بخش‌های مختلف ورودی رویکرد قابل قبول‌تری را اتخاذ می‌کند.



مزایای کلی این روش‌ها:

۱. مفاهیم ساده است.
۲. Test-time آن سریع است.

معایب کلی این روش‌ها:

۱. وقتی یک task خارج از meta-train time وارد سیستم شود (در حالت meta-test time). قادر به جذب اطلاعات taskهای جدید در وزن‌های شبکه نیستند. در نتیجه عملکرد ممکن است آسیب ببینند.
۲. هنگامی که taskها بزرگتر می‌شوند مقایسه به صورت جفت، هزینه محاسباتی گرانی دارد.
۳. بیشتر تکنیک‌های مبتنی بر متریک به وجود نمونه‌های label گذاری شده تکیه می‌کند.

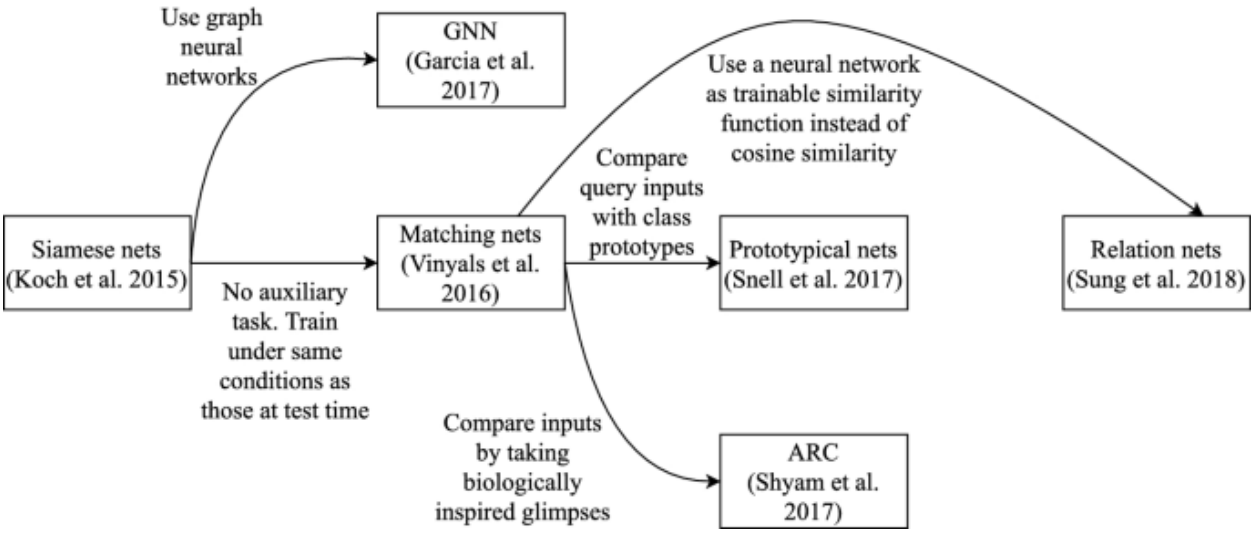
نکته:

۱. MN انعطاف پذیری بیشتری دارد
۲. PN مقایسه کمتری دارد
۳. RN از تابع شباهت قابل آموزش استفاده می‌کند و قدرتمندتر و عملکرد بهتری دارد.

۴. GNN در few-shot، semi-supervised و active-learning عملکرد خوبی دارد.

۵. ARC هزینه محاسباتی بالاتری دارد ولی از همه روش‌های متریک عملکرد بالاتری دارد.

شکل زیر خلاصه‌ای از روش‌های مبتنی بر metric است:

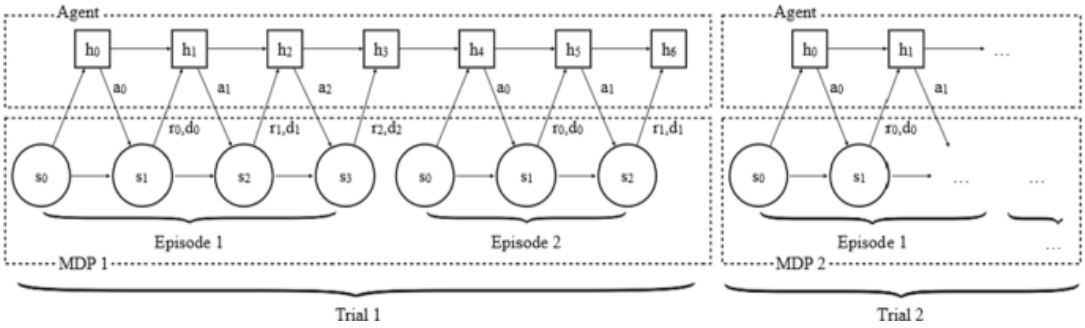


Model-based meta-learning:

در این روش‌ها برخلاف تکنیک‌های مبتنی بر متریک، که معمولاً از یک شبکه عصبی ثابت در زمان آزمایش استفاده می‌کنند، بر یک حالت adaptive internal متکی هستند. تکنیک‌های مبتنی بر مدل می‌توانند پویایی‌های داخلی را نیز یاد بگیرند، که برای پردازش و پیش‌بینی داده‌های ورودی وظایف استفاده می‌شود. تکنیک مبتنی بر مدل کل کار را در یک ماژول حافظه ذخیره می کند.

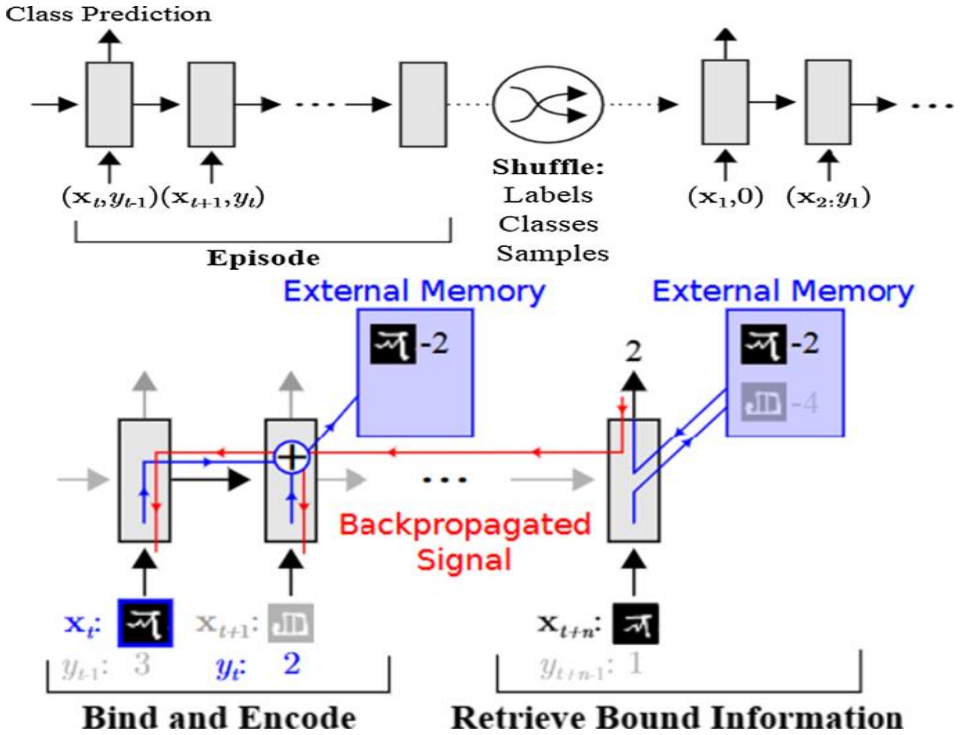
روش‌های مبتنی بر مدل را در زیر به صورت خلاصه شرح می‌دهیم:

۱. Recurrent meta-learners: این روش در محیط تقویتی اتخاذ شده است و به طور خاص برای مشکلات یادگیری تقویتی پیشنهاد شده اند. به تدریج دانش را در مورد ساختار task جمع آوری می کند، جایی که هر task به عنوان یک قسمت (یا مجموعه ای از قسمت ها) مدل می شود.
این روش‌ها اطلاعات را از قبل دخیل می‌کند. و به صورت مکرر از آن استفاده می‌کند.



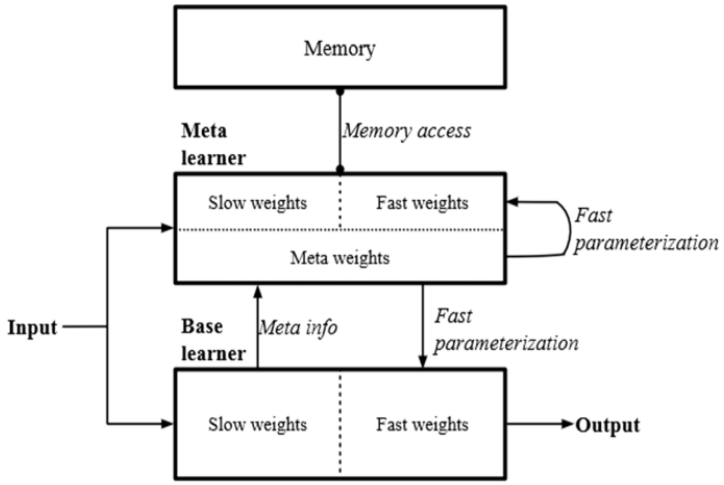
۲. Memory-augmented neural networks (MANNs) : از تغذیه کل support set به صورت متوالی در مدل استفاده می‌کند سپس با استفاده از وضعیت داخلی مدل، برای ورودی‌های query پیش‌بینی می‌کند.

این روش‌ها شبکه‌های عصبی را قادر می‌سازد تا با کمک یک حافظه خارجی سریع یاد بگیرند. سپس کنترل کننده اصلی به تدریج دانش را در بین وظایف جمع می کند، در حالی که حافظه خارجی امکان تطبیق سریع کار خاص را فراهم می کند. برای این کار از ماشین های تورینگ عصبی استفاده کردند. داده‌های یک کار به صورت دنباله‌ای پردازش می‌شوند، یعنی داده‌ها یک به یک به شبکه وارد می‌شوند. Support set ابتدا به شبکه عصبی تقویت شده با حافظه تغذیه می‌شود. پس از آن، مجموعه query پردازش می شود.



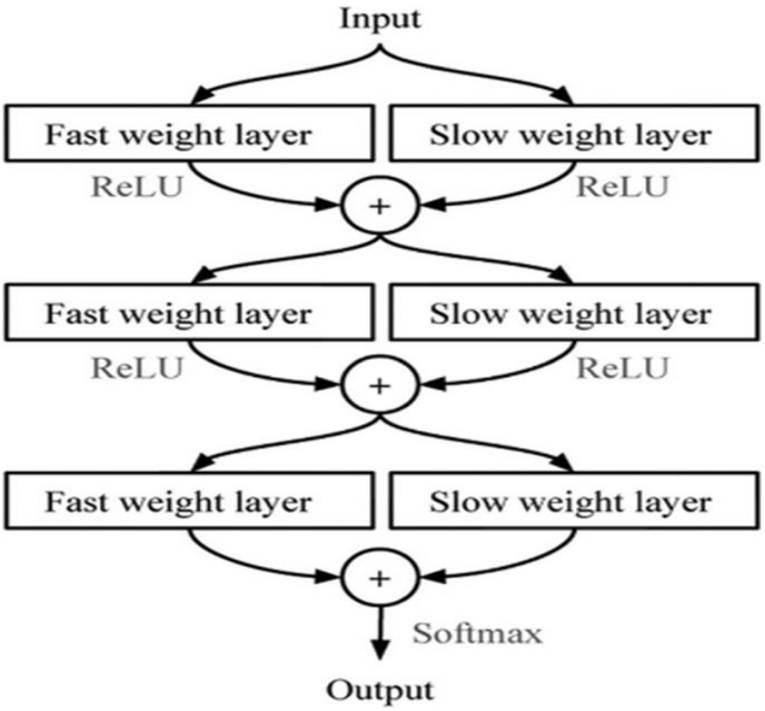
۳. Meta networks: از مسئله جعبه سیاه استفاده می‌کند. اما وزن‌های مخصوص task را برای هر کاری که با آن مواجه می‌شود را تولید می‌کند.

به دو زیر سیستم مجزا به نام‌های base-learner و meta-learner تقسیم می‌شوند. base-learner مسئول انجام task است و ارائه meta-information به meta-learner مانند loss gradients است. سپس meta-learner می‌تواند وزن‌های سریع مختص task را برای خود و base-learner محاسبه کند، به طوری که بتواند بهتر عمل کند. همانند MANN، شبکه‌های متا نیز از ایده یک ماژول حافظه خارجی بهره می‌برند. با این حال، شبکه های متا از حافظه برای اهداف دیگری استفاده می کنند. حافظه برای هر representation در پشتیبان دو جزء ذخیره می کند، یعنی representation r_i و fast weights θ^*_i. سپس از اینها برای محاسبه attention-based representation و fast weights برای ورودی‌های جدید استفاده می‌شود.



Algorithm 1 Meta networks, by Munkhdalai and Yu (2017)

Sample $S = \{(\boldsymbol{x}_i, y_i) \backsim D^{tr}_{\mathcal{T}_j}\}_{i=1}^{T}$ from the support set
for $(\boldsymbol{x}_i, y_i) \in S$ **do**
  $\mathcal{L}_i = \text{error}(u_{\boldsymbol{\phi}}(\boldsymbol{x}_i), y_i)$
end for
$\boldsymbol{\phi}^* = d_{\boldsymbol{\psi}}(\{\nabla_{\boldsymbol{\phi}}\mathcal{L}_i\}_{i=1}^{T})$
for $(\boldsymbol{x}_i, y_i) \in D^{tr}_{\mathcal{T}_j}$ **do**
  $\mathcal{L}_i = \text{error}(b_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i)$
  $\boldsymbol{\theta}^*_i = m_{\boldsymbol{\varphi}}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i)$
  Store $\boldsymbol{\theta}^*_i$ in i-th position of example-level weight memory M
  $\boldsymbol{r}_i = u_{\boldsymbol{\phi},\boldsymbol{\phi}^*}(\boldsymbol{x}_i)$
  Store $\boldsymbol{r}_i$ in i-th position of representation memory R
end for
$\mathcal{L}_{task} = 0$
for $(\boldsymbol{x}, y) \in D^{test}_{\mathcal{T}_j}$ **do**
  $\boldsymbol{r} = u_{\boldsymbol{\phi},\boldsymbol{\phi}^*}(\boldsymbol{x})$
  $\boldsymbol{a} = \text{attention}(R, \boldsymbol{r})$ ▷ a_k is the cosine similarity between $\boldsymbol{r}$ and $R(k)$
  $\boldsymbol{\theta}^* = \text{softmax}(\boldsymbol{a})^T M$
  $\mathcal{L}_{task} = \mathcal{L}_{task} + \text{error}(b_{\boldsymbol{\theta},\boldsymbol{\theta}^*}(\boldsymbol{x}), y)$
end for
Update $\Theta = \{\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}\}$ using $\nabla_{\Theta}\mathcal{L}_{task}$



۴. د

۵. ن

۶. ل

۷. ذ

۸. ذ

امکان تطبیق سریع با وظایف جدید را فراهم می کند.