**راهنمای جامع و تخصصی درک RNN، LSTM و ترنسفورمرها در یادگیری ماشین**

**مقدمه**

شبکه‌های عصبی بازگشتی (RNN)، حافظه طولانی-کوتاه‌مدت (LSTM) و ترنسفورمرها از اجزای کلیدی یادگیری ماشین و هوش مصنوعی در زمینه پردازش دنباله‌ای داده‌ها (مانند متن، صدا و سری‌های زمانی) هستند. در این مقاله، با نگاهی تخصصی و کاربردی به معرفی، مقایسه، و بررسی عمیق این سه مدل می‌پردازیم تا هم برای متخصصان علم داده و هم علاقه‌مندان کاربردی در بازار کار مفید باشد.

**۱. تعریف و دلیل پیدایش**

**RNN چیست؟**

RNN (Recurrent Neural Network) نوعی شبکه عصبی است که برای مدل‌سازی داده‌های ترتیبی (سری‌ زمانی، متن، صوت) طراحی شده است. تفاوت اصلی آن با شبکه‌های عصبی کلاسیک، حلقه بازخورد داخلی آن است که اجازه می‌دهد اطلاعات از گام‌های قبلی به گام‌های بعدی منتقل شوند.

**مشکل اصلی RNN:**

* **نابودی گرادیان (Vanishing Gradient)** هنگام یادگیری دنباله‌های بلند.

**چرا LSTM به‌وجود آمد؟**

LSTM برای غلبه بر مشکل نابودی گرادیان در RNN توسعه یافت. این مدل با استفاده از ساختار حافظه‌ای ویژه به نام "سلول" و دروازه‌هایی مانند دروازه‌ی ورودی، خروجی و فراموشی، قادر است وابستگی‌های بلندمدت را به خوبی یاد بگیرد.

**ترنسفورمرها: نسل سوم مدل‌ها**

ترنسفورمرها برای اولین بار در مقاله "Attention is All You Need" (Vaswani et al., 2017) معرفی شدند. آن‌ها بدون استفاده از بازگشت زمانی و فقط با مکانیزم توجه (Attention) کار می‌کنند، که به آن‌ها اجازه می‌دهد موازی‌سازی بالاتری در آموزش و کارایی بهتر در دنباله‌های بسیار طولانی داشته باشند.

**۲. ساختار و عملکرد**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| مدل | ساختار اصلی | انتقال اطلاعات | یادگیری وابستگی طولانی | موازی‌سازی |
| RNN | لایه با حلقه بازخورد | زمان به زمان | ضعیف | ضعیف |
| LSTM | سلول حافظه + دروازه‌ها | زمان به زمان | خوب | ضعیف |
| Transformer | Attention + Embedding | کل دنباله همزمان | بسیار عالی | عالی |

**۳. کاربردها در صنعت**

**RNN و LSTM:**

* ترجمه ماشینی اولیه (Google Translate اولیه)
* پیش‌بینی سری زمانی مالی
* تشخیص گفتار و صدا

**ترنسفورمر:**

* GPT و BERT و ChatGPT
* تولید خودکار متن، خلاصه‌سازی، QA
* کاربرد در بینایی ماشین (Vision Transformer)

**۴. پیاده‌سازی ساده در پایتون (TensorFlow / PyTorch)**

**مثال ساده LSTM در Keras:**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

model = Sequential()

model.add(LSTM(50, input\_shape=(10, 1)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

**ترنسفورمر در PyTorch:**

import torch.nn as nn

model = nn.Transformer(d\_model=512, nhead=8, num\_encoder\_layers=6)

**۵. مقایسه تخصصی**

|  |  |  |  |
| --- | --- | --- | --- |
| معیار مقایسه | RNN | LSTM | Transformer |
| سادگی پیاده‌سازی | بالا | متوسط | پیچیده |
| زمان آموزش | کند | کندتر | سریع‌تر |
| ظرفیت درک متون طولانی | پایین | خوب | عالی |
| منابع محاسباتی | کم | متوسط | زیاد |

**۶. نکات تکمیلی و ترکیب روش‌ها**

* **BiLSTM**: LSTM دوطرفه برای بهره‌گیری از اطلاعات قبلی و بعدی در متن.
* **Seq2Seq + Attention**: ترکیب RNN و attention برای ترجمه و خلاصه‌سازی.
* **Transformer + CNN**: ترکیب برای مدل‌سازی چندبعدی مانند ویدیو.

**۷. توصیه برای ورود به بازار کار**

* برای شروع: با LSTM در پروژه‌های واقعی (مانند پیش‌بینی قیمت، تحلیل احساسات) تمرین کنید.
* یادگیری ترنسفورمرها را با HuggingFace و PyTorch دنبال کنید.
* تسلط به کتابخانه‌هایی مثل transformers, torchtext, keras ضروری است.

**منابع معتبر:**

1. Vaswani et al., 2017: "Attention is All You Need"
2. DeepLearning.ai Courses by Andrew Ng
3. HuggingFace Documentation: <https://huggingface.co/docs>
4. TensorFlow and PyTorch official docs
5. ویدیوهای YouTube: Sentdex, StatQuest, Aladdin Persson

**جمع‌بندی**

مدل‌های RNN، LSTM و ترنسفورمر هرکدام در پاسخ به نیازهایی در پردازش زبان طبیعی و داده‌های ترتیبی شکل گرفته‌اند. درک دقیق ساختار، عملکرد، مزایا و محدودیت‌های هر کدام، کلید انتخاب درست آن‌ها در پروژه‌های صنعتی و پژوهشی است. برای متخصص شدن در این زمینه، علاوه‌ بر یادگیری مفهومی، تمرین عملی با داده‌های واقعی الزامی است.

**🔵 ورودی‌ها:**

* Xₜ: ورودی در زمان t (مثلاً قیمت امروز، یا کلمه فعلی)
* hₜ₋₁: خروجی زمان قبل (حافظه کوتاه‌مدت)
* Cₜ₋₁: حافظه سلول از مرحله قبلی (حافظه بلندمدت)

**🔁 ساختار کلی سلول LSTM**

در هر زمان t، سلول LSTM این کارها رو می‌کنه:

**🟡 1. دروازه فراموشی (Forget Gate)**

📍 وظیفه: **چه چیزهایی از حافظه قبلی (Cₜ₋₁) رو دور بریزیم؟**

* ورودی: Xₜ و hₜ₋₁
* تابع فعال‌سازی: سیگموید (σ) → خروجی بین 0 و 1
* خروجی این دروازه ضرب میشه در حافظه قبلی (Cₜ₋₁)

🧠 یعنی: اگر خروجی 1 باشه → نگه می‌داریم  
اگر 0 باشه → کامل فراموش می‌کنیم

**🟡 2. دروازه ورودی (Input Gate)**

📍 وظیفه: **چه اطلاعات جدیدی به حافظه اضافه کنیم؟**

شامل دو بخش:

1. سیگموید (σ): مشخص می‌کنه چه اطلاعاتی باید وارد بشن
2. tanh: یک مقدار پیشنهادی جدید برای اضافه شدن می‌سازه

→ این دو تا با هم ضرب می‌شن → مشخص می‌کنن چه چیزی وارد حافظه بشه

**🟩 3. به‌روزرسانی حافظه (Cell State)**

* حافظه جدید Cₜ با این فرمول ساخته می‌شه:

mathematica

CopyEdit

Cₜ = forget\_gate \* Cₜ₋₁ + input\_gate \* new\_candidate

یعنی:

* بخشی از حافظه قبلی نگه داشته می‌شه
* بخشی از اطلاعات جدید وارد می‌شن

**🟣 4. دروازه خروجی (Output Gate)**

📍 وظیفه: **چه چیزی به عنوان خروجی (hₜ) تولید کنیم؟**

* باز هم Xₜ و hₜ₋₁ می‌رن توی یک سیگموید
* حافظه جدید Cₜ از یک tanh عبور می‌کنه
* این دو با هم ضرب می‌شن → خروجی نهایی hₜ

**✅ خلاصه تصویری ساده:**

|  |  |  |
| --- | --- | --- |
| مرحله | کارکرد | تابع |
| دروازه فراموشی | چه چیزی رو فراموش کنیم؟ | σ |
| دروازه ورودی | چه چیزی رو وارد حافظه کنیم؟ | σ + tanh |
| حافظه سلول | ترکیب حافظه قدیم + جدید | \* و + |
| دروازه خروجی | چه چیزی رو خروجی بدیم؟ | σ + tanh |

**📦 چرا LSTM بهتر از SimpleRNN هست؟**

چون می‌تونه اطلاعات طولانی‌مدت رو **به خاطر بسپاره** (مثلاً یک کلمه کلیدی در ابتدای جمله که آخر جمله معنی می‌ده)، بر خلاف SimpleRNN که حافظه‌ی کوتاه‌مدت داره و زود فراموش می‌کنه.