مزایا و معایب درخت‌های تصمیم و شبکه‌های عصبی

درخت‌های تصمیم:

مزایا:

1. **سادگی و تفسیرپذیری:** ساختار درخت‌های تصمیم ساده و قابل فهم است، بنابراین تفسیر نتایج آن‌ها آسان است.
2. **نیاز به پیش‌پردازش کم:** نیازی به نرمال‌سازی داده‌ها یا مدیریت مقادیر گم‌شده ندارند.
3. **کارایی با داده‌های طبقه‌بندی و رگرسیون:** برای هر دو نوع داده‌های طبقه‌بندی و رگرسیون مناسب هستند.
4. **مقاومت به داده‌های پرت:** نسبت به داده‌های پرت مقاوم‌تر هستند.

معایب:

1. **مستعد بیش‌برازش:** به ویژه در داده‌های با ابعاد بالا یا پیچیده، ممکن است بیش‌برازش رخ دهد.
2. **حساسیت به تغییرات کوچک:** تغییرات کوچک در داده‌ها می‌تواند ساختار درخت را به شدت تغییر دهد.
3. **کارایی پایین در داده‌های پیچیده:** برای داده‌های با روابط غیرخطی پیچیده ممکن است عملکرد ضعیفی داشته باشند.

مناسب برای: داده‌های ساختاریافته، داده‌های با ابعاد کم تا متوسط، و مسائلی که نیاز به تفسیرپذیری دارند.

شبکه‌های عصبی:

مزایا:

1. **قدرت مدل‌سازی بالا:** توانایی یادگیری روابط غیرخطی و پیچیده در داده‌ها را دارند.
2. **انعطاف‌پذیری:** برای انواع داده‌ها (تصویر، متن، صوت) و مسائل (طبقه‌بندی، رگرسیون، خوشه‌بندی) مناسب هستند.
3. **مقیاس‌پذیری:** با افزایش حجم داده‌ها و منابع محاسباتی، عملکرد بهتری ارائه می‌دهند.

معایب:

1. **نیاز به داده‌های زیاد:** برای عملکرد خوب نیاز به حجم زیادی از داده‌ها دارند.
2. **پیچیدگی و تفسیرناپذیری:** مدل‌های پیچیده‌ای هستند و تفسیر نتایج آن‌ها دشوار است.
3. **نیاز به پیش‌پردازش:** معمولاً نیاز به نرمال‌سازی داده‌ها و مدیریت مقادیر گم‌شده دارند.
4. **زمان و منابع محاسباتی:** آموزش شبکه‌های عصبی زمان‌بر و نیازمند منابع محاسباتی قوی است.

مناسب برای: داده‌های با ابعاد بالا، داده‌های غیرساختاریافته (مانند تصاویر، متن، صوت)، و مسائل پیچیده با روابط غیرخطی.

جمع‌بندی:

- **درخت‌های تصمیم:** برای داده‌های ساختاریافته و مسائل ساده‌تر که نیاز به تفسیرپذیری دارند مناسب‌تر هستند.
- **شبکه‌های عصبی:** برای داده‌های پیچیده و غیرساختاریافته که نیاز به مدل‌سازی روابط غیرخطی دارند، مناسب‌تر هستند.

استفاده از تابع فعال‌سازی ***f(x)=x* (تابع خطی)** در لایه‌های مخفی یک **شبکه پرسپترون چندلایه (MLP)** مناسب نیست، زیرا:

1. **عدم ایجاد غیرخطی‌بودن**: اگر تمام لایه‌های مخفی از یک تابع خطی استفاده کنند، ترکیب چندین لایه در نهایت معادل یک تبدیل خطی ساده خواهد بود. یعنی شبکه هرچقدر هم عمیق باشد، نمی‌تواند روابط غیرخطی بین داده‌ها را یاد بگیرد.
2. **عدم افزایش توان مدل**: اضافه کردن لایه‌های بیشتر با تابع خطی هیچ مزیت خاصی ندارد، زیرا ترکیب چندین تبدیل خطی، همچنان یک تبدیل خطی باقی می‌ماند. در نتیجه، شبکه عملکردی مشابه یک **مدل رگرسیون خطی** خواهد داشت.
3. **ناتوانی در حل مسائل پیچیده**: بسیاری از مسائل دنیای واقعی دارای روابط غیرخطی پیچیده هستند. برای حل این مسائل، شبکه باید از توابع فعال‌سازی **غیرخطی** مانند **ReLU، سیگموید، یا تانژانت هیپربولیک** استفاده کند تا بتواند الگوهای پیچیده را بیاموزد.

در نتیجه، برای یادگیری مؤثر و افزایش قدرت تفکیک‌پذیری شبکه، باید از **توابع فعال‌سازی غیرخطی** در لایه‌های مخفی استفاده شود.

تابع **ReLU** به صورت $f(x)=\max(0,x)$ تعریف می‌شود. این تابع در تمام نقاط به جز ***x=0*** دارای مشتق است.

مشکل اصلی این است که **در نقطه *x=0*** مشتق تعریف دقیقی ندارد. با این حال، در عمل این مسئله تأثیر چندانی بر آموزش شبکه ندارد، زیرا مقدار دقیق مشتق در یک نقطه منفرد (یعنی $x=0$) در یک فضای پارامتری پیوسته تأثیر کمی دارد.

در پیاده‌سازی‌های عددی، معمولاً مقدار مشتق در $x=0$ را به صورت قراردادی ***0* یا *1*** در نظر می‌گیرند یا مقدار گرادیان را مستقیماً از مقدار قبلی به‌روزرسانی می‌کنند. به همین دلیل، **ReLU همچنان در روش‌های مبتنی بر گرادیان مانند SGD به خوبی کار می‌کند** و به دلیل سادگی محاسبات و کاهش اثر گرادیان ناپدیدشونده، در شبکه‌های عصبی عمیق پرکاربرد است.

نقاطی که می‌توانند باعث توقف یا کاهش سرعت یادگیری شوند:

1. **نقاط زین اسبی (Saddle Points):**
   a. نقاطی که گرادیان در آن‌ها صفر است، اما برخلاف مینیمم‌های محلی، برخی جهت‌ها دارای شیب مثبت و برخی شیب منفی هستند.

b. در فضاهای با ابعاد بالا، نقاط زین اسبی بسیار رایج‌تر از مینیمم‌های محلی هستند و می‌توانند باعث کاهش شدید سرعت یادگیری شوند، زیرا گرادیان در اطراف آن‌ها کوچک است.

2. **مینیمم‌های محلی بد:**

a. اگر تابع Loss دارای چندین مینیمم محلی باشد، ممکن است شبکه در یک مینیمم نامطلوب گیر بیفتد.
b. در شبکه‌های عصبی عمیق، بیشتر مینیمم‌های محلی قابل قبول هستند، اما برخی از آن‌ها ممکن است منجر به تعمیم ضعیف شوند.

چالش بزرگ‌تر:

نقاط زین اسبی چالش بزرگ‌تری هستند زیرا در فضاهای با ابعاد بالا، احتمال گیر افتادن در آن‌ها بیشتر از مینیمم‌های محلی بد است. در این نقاط، گرادیان بسیار کوچک می‌شود و یادگیری بسیار کند می‌شود، بدون اینکه شبکه به یک نقطه بهینه مناسب برسد.

راهکارها برای جلوگیری از گیر افتادن:

1. **استفاده از مقداردهی اولیه مناسب (Good Initialization):**

a. مقداردهی اولیه مناسب وزن‌ها (مثلاً **He Initialization** یا **Xavier Initialization**) می‌تواند از گیر افتادن در نقاط زین اسبی یا مینیمم‌های نامطلوب جلوگیری کند.

2. **استفاده از بهینه‌سازهای پیشرفته (Advanced Optimizers):**

a. **بهینه‌سازهایی مانند Adam یا RMSprop** با تغییر مقیاس گرادیان در جهات مختلف، می‌توانند از گیر افتادن در نقاط زین اسبی جلوگیری کنند و یادگیری را تسریع بخشند.