۱) الف) درخت تصمیم Pro interpretable (فهم لایه های درخت و تفسیر آن آسان است).

Pro اگر نیاز به preprocessing دارد. سرعت بالا و computation کم.

Cons اگر به درستی hyperparameter ها راست نکنیم overfit بسیار محتمل است.

ترانایی تک های داده های آن پیچیده باشد را ندارد. (برای کارهای ساده مناسب است)

شبکه های عصبی Pro طبق universal classifier می تقل گفت هر رابطه پیچیده و حتی دنیای پیچیده مانند صدا و عکس را نیز می تواند پردازش کند. (حتی غیر خطی)

Cons interpretable نیست. (قابلیت فهم و تفسیر متن لایه ها را ندارد.)

Cons نیاز به computation بالا و داده train زیاد دارد.

برای داده های ساده مانند عددی از decision tree و برای داده های پیچیده مانند متن و عکس از neural network استفاده می کنیم.

ب) اگر از $g(x)=x$ برای activation function استفاده کنیم در واقع کمکی نمی کند ما می خواهیم non-linearity به مدل اضافه کنیم و با این تابع فقط می تواند الگوهای خطی را تشخیص دهد. برای مثال استفاده از ReLu یا GeLu اینکار را برای ما انجام می دهد.

ج) این تابع sub-differentiable است. یعنی فقط در $x=0$ مشتق پذیر نیست که می توان به صورت قراردادی مقداری را به مشتق این نقطه نسبت داد. همچنین چون بقیه نقاط مشتق پذیر اند و به طور معمول ReLu خیلی خوب converge می کند پس یک activation func مناسب است.

د) در مسیر یادگیری ممکن است به بعضی نقاط برسیم که کار ما را سخت کنند مانند local minima ها و saddle ها.

در مینیمم محلی ما یک دره می بینیم اما به minimum global نرسیده ایم و باید به گونه ای عمل کنیم تا گیر نکنیم. یا در saddle point ها با برجستگی مواجه ایم که این نقاط از یک طرف گرادیان مثبت و از طرف دیگر گرادیان منفی دارند. یا در نقاطی که plateau می نامند یعنی حدود اگرادیان صفر و تابع مرتب نمی کند. این نقاط می توانند را به مشکل بیندازند. برای جلوگیری

وگیر نیفتادن در این نقاط روش های optimize متعددی مانند Adam, AdaGrad و momentum وجود دارد.

۱) روش adam :

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla_\theta L(\theta) \quad , m_0 = \beta$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla_\theta L^2(\theta)$$

$$\theta_j = \theta_j - \frac{\varepsilon}{\sqrt{v_{t+1}+\varepsilon'}} m_{t+1}$$

هربار با استفاده از دو متغیر m و v که انگار سرعت را شبیه سازی کرده ایم دو واقع طول step را تغییر می دهیم و آپدیت می کنیم.
این الگوریتم ترکیبی از momentum و RMSProp است.

۲) روش AdaGrad :

$$g_{t+1} = g_t + \nabla_\theta L(\theta)^2$$

$$\theta_{t+1} = \theta_t - \varepsilon \frac{\nabla_\theta L(\theta)}{\sqrt{g_t+\varepsilon'}}$$

از ل که مانند شتاب است استفاده می شود تا یک step منطقی تولید کند.