



تمرین اول

نام درس: یادگیری عمیق

استاد درس: دکتر محمدرضا محمدی

نام: محمد حقیقت

شماره دانشجویی: 403722042

گرایش: هوش مصنوعی

دانشکده: مهندسی کامپیووتر

نیم سال دوم 1403-1404

سؤال اول

Subject: Deep Learning Year: _____ Month: _____ Date: _____

$f(x) = \exp(-\|Ax - b\|^2) + \sin\left(\sum_{i=1}^n c_i x_i\right)$ دالة
 $F(x)$
 نظریه: $\|f(x) - F(x)\|_{L_2} = \left(\int_K |f(x) - F(x)|^2 dx \right)^{1/2} < \varepsilon$

$K \subset \mathbb{R}^n$, $f_m = f(x) + f_{CN}$

$f_r(x) = \exp(-\|Ax - b\|^2)$ $f_s(x) = \sin\left(\sum_{i=1}^n c_i x_i\right)$
 از این که $f_m = f_r + f_s$ و ممکن است f_s بزرگ باشد، میتوان این را برای f_s در نظر گرفت

$F = f_r + f_s$ برای نمونه f_r , f_s چیزی باشد که f_m , f_{CN} , f_{rCN}

مatrix H را در نظر گیریم که $H(y) = \frac{1}{1+y^2}$ است

$g(y) = \exp(-\|y\|^2)$ تحصیل بخش اول (۱۵۰)
 $f_m(y) = g(Ax - b)$ از دلایل مذکور, $\|y\|^2 = y_1^2 + y_2^2 + \dots + y_n^2$

$\sup_{|y_i|} |y_i - H_i(y_i)| < \frac{\varepsilon'}{n}$ $H(y) = H_1(y_1) + H_2(y_2) + \dots + H_n(y_n)$

$\sup_{|y_i|} \|y_i - H_i(y_i)\| \leq \sum_{i=1}^n |y_i - H_i(y_i)| < \varepsilon'$
yes

$f \rightarrow \exp(-t)$ دالة
 $t \in [0, R + \varepsilon']$ $\sup_{t \in [0, R + \varepsilon']} |\exp(-t) - E(t)| \leq \delta$ برای $E(t)$

SABA

Subject :

Year :

Month :

Date :

$$\text{لما يتحقق ذلك فالآن نحسب } E(H(y)) = G(y) \text{ على شكل}$$

$$|f_n(x) - G(Ax-b)| = |\exp(-\|Ax-b\|^2) - E(H(Ax-b))|$$

إذا $\|y\|^2 - H(y) \leq \epsilon'$ و $t = \|y\|^2$, $t' = H(y)$ ثم $|t - t'| \leq \epsilon'$

$$S(x) = \sum_{i=1}^n c_i x_i' \quad \text{ويمكن تدوين } f_n(x) \cdot \sin\left(\sum_{i=1}^n c_i x_i'\right) \quad \text{لديه نفس المبرهن}$$

$$T = \max_{k \in K} \left\{ c_i x_i' < 0, \quad S(x) \in [0, T] \right\} \quad \text{لديه نفس المبرهن}$$

لديه نفس المبرهن

$$\sup_{x_i \in [-m, m]} |x_i' - Q_i(x_i)| < \frac{\epsilon''}{n \max c_i}$$

$$\text{حيث } M = \max_{k \in K} |x_k|. \text{ ثم } Q(x) = \sum_{i=1}^n c_i Q_i(x_i);$$

$$|S(x) - Q(x)| < \epsilon''$$

لديه نفس المبرهن

$$\sup_{u \in [0, T \epsilon'']} |\sin(u) - S(u)| < \delta'$$

$$F_x(x) = S(Q(x))$$

$$F(x) = F_1(x) + F_2(x) = G(Ax-b) + S(Q(x))$$

$$|f(x) - F(x)| \leq |f_1(x) - F_1(x)| + |f_2(x) - F_2(x)|.$$

$$\epsilon < \|f - F\|_{L_1} \quad \text{لما يتحقق ذلك كنتم وتبين ذلك}$$

SABA

أي $\sup_{u \in [0, T \epsilon'']} |\sin(u) - S(u)| < \delta'$

سوال دوم

$$R_\lambda(F) = R(F) + \lambda \|W\|_2^2$$

$R(F)$: میانگین خطای داده های آموزشی که نشان می دهد مدل F چقدر با داده های آموزشی سازگار است.

$\|W\|_2^2$: ترم منظم سازی L_2 , که در آن $\|W\|_2^2$ نرم دوم اقلیدسی وزن ها و $0 \leq \lambda$ پارامتر منظم سازی است که شدت این جریمه را کنترل می کند.

بدون منظم سازی ($\lambda=0$) مدل فقط $(F)R$ را کمینه می کند، که ممکن است به اورفیتینگ منجر شود یعنی مدل بیش از حد به داده های آموزشی از جمله نویز آن وابسته شود و واریانس بالایی داشته باشد و تعمیم پذیری ضعیفی به داده های جدید نشان دهد. ترم L_2 وزن های بزرگ را جریمه می کند و مدل های ساده تر را تشویق می کند. هدف ما اثبات این است که این کار واریانس را کاهش می دهد و تعمیم پذیری را بهبود می بخشد سپس تاثیر λ را بررسی می کنیم.

واریانس در اینجا به حساسیت پیش بینی های مدل به تغییرات داده های آموزشی اشاره دارد. مدل با واریانس بالا بیش برآش می کند و نویز را نیز یاد می گیرد. بیایید این را شهودی و رسمی بررسی کنیم.

استدلال شهودی:

مدل خطی $x=F(x)=W^T x$ را در نظر بگیرید. کمینه کردن $(F)R$ به تنها بی ممکن است به وزن های بزرگ W منجر شود تا نویز آموزشی را کاملا بپوشاند. اگر داده های آموزشی کمی تغییر کنند W به شدت تغییر می کند و پیش بینی ها نوسان زیادی خواهند داشت (واریانس بالا).

ترم L_2 با جریمه کردن وزن های بزرگ W را به سمت صفر می کشاند. وزن های کوچک تر یعنی مدل به تغییرات کوچک در ورودی یا داده های آموزشی کمتر حساس است که واریانس را کاهش می دهد.

استدلال رسمی: تجزیه بایاس-واریانس

برای مسئله رگرسیون با خطای مربعی، خطای مورد انتظار به صورت زیر تجزیه می شود:

$$E[(y-F(x))^2] = \text{نویز} + \text{واریانس} + \text{بایاس}^2$$

بایاس: خطای دلیل سادگی بیش از حد مدل (کم برآش)

واریانس: خطای دلیل حساسیت به داده های آموزشی.

نویز: خطای غیرقابل کاهش از تصادفی بودن داده ها.

فرض کنید $F(x) = W^T x$ و تابع منظم شده را برای n نمونه کمینه می کنیم:

$$R_\lambda(W) = \frac{1}{n} \sum_{i=1}^n (y_i - W^T x_i)^2 + \lambda \|W\|_2^2$$

گرادیان نسبت به W را محاسبه و صفر می کنیم :

$$\nabla_W R_\lambda(W) = -\frac{2}{n} \sum_{i=1}^n (y_i - W^T x_i) x_i + 2\lambda W = 0$$

حل معادله :

$$W = (X^T X + n\lambda I)^{-1} X^T y$$

واریانس تخمین \hat{W} برابر است با

$$\text{Var}(W) = \sigma^2 (X^T X + n\lambda I)^{-1} X^T X (X^T X + n\lambda I)^{-1}$$

برای $\lambda = 0$ این به $(X^T X)^{-1}$ کاهش می یابد که اگر $X^T X$ مقادیر ویژه کوچک داشته باشد (بدترین حالت)، بزرگ می شود. با افزایش λ ماتریس $(X^T X + n\lambda I)$ بهتر شرطی می شود (مقادیر ویژه به اندازه $n\lambda$ افزایش می یابند)، که معکوس را کوچک تر کرده و واریانس $\hat{W} = W^T x = F(x)$ را کاهش می دهد. اما این کار بایاس را افزایش می دهد که در تعیین پذیری بررسی می کنیم.

بهبود تعیین پذیری

تعیین پذیری یعنی مدل چقدر روی داده های دیده نشده خوب عمل می کند، که با خطای مورد انتظار $E[L(y, F(x))]$ سنجیده می شود. کمینه سازی empirical risk به تنهایی می تواند به بیش برآش منجر شود. منظم سازی L_2 با کنترل پیچیدگی مدل کمک می کند.

پیچیدگی Generalization Bound و Rademacher (کران تعیین)

کران تعیین از نظریه یادگیری آماری چنین است: با احتمال $1-\delta$ برای نمونه ای با اندازه n

$$R_{\text{true}}(F) \leq R(F) + 2R_n(F) + \sqrt{\frac{\log(1/\delta)}{2n}}$$

$R_{\text{true}}(F)$: True expected risk.

$R(F)$: Empirical risk.

$2R_n^*(F)$: Empirical Rademacher complexity, measuring the capacity of F to fit random noise.

برای مدل های خطی $x = W^T x$ با $B \leq \|W\|^2$ پیچیدگی ردمابر چنین کران بندی می شود:

$$R_n^*(F) \leq \frac{B\|X\|_F}{\sqrt{n}}$$

که $\|X\|_F$ نرم فروبنیوس ماتریس داده هاست.

کمینه کردن

$$R_\lambda(F) = R(F) + \lambda \|W\|_2^2$$

به طور ضمنی

$$\|W\|_2^2 \leq \frac{R(F)}{\lambda}$$

را نتیجه می دهد. پس:

$$\|W\|_2 \leq \sqrt{\frac{R(F)}{\lambda}}$$

با افزایش λ کاهش می یابد، کران تنگ تر می شود و شکاف بین $R(F)$ و $R_{true}(F)$ کم می شود که منجر به تعمیم پذیری بهتر می شود

تأثیر λ بر کران تعمیم

$$R_{true}(F) \leq R(F) + 2 \frac{\|W\|_2\|X\|_F}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

جایگزینی

$$\|W\|_2 \leq \sqrt{\frac{R(F)}{\lambda}}$$

می رسیم به

$$R_{true}(F) \leq R(F) + 2 \frac{R(F)\|X\|_F}{\sqrt{\lambda n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

با افزایش λ ترم $\frac{R(F)\|X\|_F}{\sqrt{\lambda n}}$ کاهش می یابد (چون λ در مخرج است) و کران تنگ تر می شود. اما $R(F)$ ممکن است به دلیل بایاس افزایش یابد.

وقتی λ به سمت صفر می کند₂ $\parallel W \parallel$ بزرگ تر شده واریانس و ریسک بیش برآش بالا می رود.

λ بهینه بایاس و واریانس را متعادل می کند تا $R_{true}(F)$ کمینه شود.

سوال سوم

الف

در این مسئله که مربوط به بهینه سازی تبلیغات با استفاده از داده های گذشته است، باید داده ها را به دقت به بخش های مختلف تقسیم کنیم. این کار کمک می کند مدل بهتر آموزش ببیند و عملکرد واقعی تری داشته باشد.

♦ مجموعه آموزش (Train)

هدف: یادگیری مدل از الگوهای و روندهای تبلیغات.

چی مهمه؟ این مجموعه باید شامل انواع مختلفی از تبلیغات، کمپین ها و بازه های زمانی باشد.

پیشنهاد: بخش بزرگی از داده های قدیمی تر را انتخاب کنید تا مدل بتواند رفتار کلی بازار را یاد بگیرد.

♦ مجموعه ارزیابی (Dev)

هدف : تنظیم پارامترهای مدل و بررسی عملکرد در حین آموزش.

چی مهمه؟ داده های این مجموعه نباید توانی آموزش استفاده شده باشند.

پیشنهاد : از داده های جدیدتر نسبت به Train استفاده کنید تا مدل روی داده های ناآشنا هم تست بشه.

♦ مجموعه ارزیابی داخلی (Dev-Train)

هدف: کمک به انتخاب بهترین مدل در طول آموزش.

چی مهمه؟ این مجموعه شبیه به داده های واقعی (Test) باشد ولی کوچکتر.

پیشنهاد: بخشی از داده های Train را جدا کنید و برای ارزیابی داخلی استفاده کنید.

♦ **مجموعه آزمایش نهایی (Test)**

هدف: بررسی نهایی عملکرد مدل روی داده هایی که هیچ وقت ندیده.

چی مهمه؟ شامل تبلیغات، کمپین ها و شرایطی باشد که ممکنه در آینده اتفاق بیفته.

پیشنهاد: از جدیدترین بازه های زمانی استفاده کنید.

نکات مهم:

تقسیم بندی زمانی: داده های قدیمی تر برای Train و داده های جدیدتر برای Dev و Test مناسب ترند تا از نشت داده جلوگیری بشه.

تنوع تبلیغات: داده ها باید شامل انواع مختلف تبلیغات و نرخ های کلیک متفاوت باشند.

مدیریت موارد جدید: برای تبلیغات یا کمپین های جدید که در Train نیستند، می تونید از میانگین وزنی داده های مشابه استفاده کنید.

♦ **پیشنهاد تقسیم بندی:**

• 70٪ داده ها برای Train

• 15٪ داده ها برای Dev

• 5٪ داده ها برای Dev-Train

• 10٪ داده ها برای Test

ب ۱

آ) از قطعیت داده های ورودی اطمینان داریم ولی خطای آموزش مدل (Training Error) بالا است.

مشکل:

این وضعیت معمولاً به معنای Underfitting است. مدل شما به اندازه کافی پیچیده نیست که الگوهای موجود در داده های آموزشی را یاد بگیرد.

راه حل ها:

استفاده از مدل های پیچیده تر مانند درخت های تصمیم عمیق تر، شبکه های عصبی با لایه های بیشتر یا الگوریتم های پیچیده تر.

افزایش تعداد ویژگی ها (Feature Engineering) و افزودن ویژگی های معنادار برای غنی تر کردن ورودی ها.

افزایش تعداد دوره های آموزش (Epochs) برای دادن فرصت بیشتر به مدل برای یادگیری. کاهش میزان منظم سازی (Regularization) که ممکن است مدل را بیش از حد محدود کرده باشد.

ب) خطای آموزش مدل پایین است ولی خطای آن روی مجموعه Dev-Train همچنان بالا است.

مشکل:

این وضعیت نشان دهنده ای بیش برازش (Overfitting) روی داده های آموزشی است.

مدل روی داده های Train بسیار خوب عمل می کند اما قادر به تعمیم روی داده های جدید (Dev-Train) نیست.

راه حل ها:

افزایش میزان منظم سازی (Regularization) برای جلوگیری از یادگیری بیش از حد جزئیات در داده های Train

کاهش پیچیدگی مدل (مانند کاهش تعداد لایه های شبکه عصبی یا محدود کردن عمق درخت های تصمیم)

افزایش حجم داده های آموزشی در صورت امکان استفاده از تکنیک های داده افزایی (Data Augmentation) برای افزایش تنوع در داده های آموزشی.

ج) خطای مدل در مجموعه های Dev-Train و Train پایین است ولی روی مجموعه Dev خطأ زیاد است.

مشکل:

این وضعیت معمولاً به معنای شکاف در توزیع داده ها (Data Distribution Shift) است. داده های Dev با داده های Train و Dev-Train تفاوت قابل توجهی دارند.

راه حل ها:

بررسی تفاوت های آماری بین داده های Train و Dev مانند تفاوت در توزیع ویژگی ها جمع آوری داده های بیشتر که مشابه داده های Dev باشد.

در نظر گرفتن روش هایی مانند Domain Adaptation که مدل را قادر می سازد با تغییرات در توزیع داده ها سازگار شود.

بررسی وجود نشت داده (Data Leakage) که ممکن است باعث عملکرد غیرواقعی مدل روی داده های Train و Dev-Train شده باشد.

د) خطای Dev پایین است ولی روی مجموعه Test خطأ همچنان زیاد است.

مشکل:

این وضعیت نشان می دهد که مدل شما روی داده های Dev بیش از حد تنظیم شده است و احتمالاً Dev نماینده خوبی برای توزیع داده های آینده (Test) نیست. Overfitting

راه حل ها:

افزایش حجم داده های Dev برای افزایش نمایانگری این مجموعه.

استفاده از روش های تنظیم بیشتر روی داده های Dev-Train به جای تمرکز صرف روی Dev استفاده از تکنیک های اعتبارسنجی متقطع (Cross Validation) برای ارزیابی دقیق تر عملکرد مدل روی داده های جدید.

بررسی تفاوت های آماری میان مجموعه های Dev و Test برای درک بهتر دلایل تفاوت عملکرد.

ب 2

در سناریوی اول، یعنی "از قطعیت داده های ورودی اطمینان داریم ولی خطای آموزش مدل (Training Error) بالا است" مشکل اصلی کم برازش (Underfitting) است. در این حالت، مدل نمی تواند به خوبی الگوهای داده های آموزشی را یاد بگیرد.

آیا افزایش سایز داده های آموزش راه حل خوبی است؟

نه لزوماً. افزایش حجم داده های آموزشی به تنها یکی کمکی به کاهش خطای Train نمی کند، زیرا مشکل اصلی پیچیدگی ناکافی مدل است. اگر مدل توانایی یادگیری الگوهای موجود را نداشته باشد، حتی با داده های بیشتر نیز عملکرد آن بهبود پیدا نمی کند.

چه زمانی افزایش داده های آموزش مفید است؟

اگر داده های موجود نویز زیادی داشته باشند و داده های جدید بتوانند میانگین بهتری از توزیع واقعی ایجاد کنند.

اگر مدل ظرفیت کافی برای یادگیری را داشته باشد ولی داده های فعلی کافی نباشند.

چه راه حل هایی بهتر از افزایش داده ها هستند؟

- استفاده از مدل های پیچیده تر (مثلاً از رگرسیون خطی به شبکه عصبی برویم).
- افزودن ویژگی های معنادار (Feature Engineering) برای بهبود ورودی ها.
- کاهش منظم سازی (Regularization) که ممکن است مدل را محدود کرده باشد.
- افزایش تعداد دوره های آموزش (Epochs) در یادگیری عمیق.

ب 3

اختلاف بین پیش بینی های مدل CTR و CVR و درآمد واقعی می تواند دلایل مختلفی داشته باشد. اگر مدل های ما روی داده های آموزشی عملکرد خوبی دارند اما وقتی روی داده های Dev تست می شوند، پیش بینی هایشان با درآمد واقعی فاصله زیادی دارد، باید بررسی کنیم که چه چیزی باعث این مشکل شده است.

چرا این اختلاف به وجود می آید؟

داده های آموزشی با داده های Dev فرق دارند

گاهی اوقات داده هایی که مدل روی آن ها آموزش دیده، با داده هایی که برای ارزیابی مدل استفاده می کنیم (Dev)، تفاوت دارند. مثلاً شاید کاربران یا تبلیغات در داده های آموزشی از نظر رفتار و الگوها با کاربران واقعی متفاوت باشند. در این صورت، مدل ما روی داده های جدید عملکرد خوبی نخواهد داشت.

راه حل: بررسی کنیم که توزیع ویژگی های کلیدی مثل نرخ کلیک(CTR)، نرخ تبدیل(CVR) ، و پیشنهاد قیمت(bid) در داده های Train و Dev یکسان است یا نه. اگر تفاوت زیادی وجود دارد، شاید لازم باشد داده های آموزش را تغییر دهیم یا مدل را به روز کنیم

رفتار کاربران تغییر کرده است

ممکن است کاربران در طول زمان رفتارشان را تغییر داده باشند. برای مثال، اگر تبلیغات ما برای یک فصل خاص مناسب بوده ولی حالا در فصل جدید کاربران به آن ها علاقه ای ندارند، نرخ کلیک و نرخ تبدیل کاهش پیدا می کند.

راه حل: مدل را به طور مداوم با داده های جدید آپدیت کنیم و در آموزش مدل، به داده های جدید وزن بیشتری بدهیم.

داده های آموزش دارای نویز یا اطلاعات غلط هستند

اگر داده های آموزشی ما شامل کلیک های جعلی (مثلاً کلیک هایی که توسط بات ها انجام شده) یا اطلاعات نادرست باشند، مدل روی داده های اشتباہ آموزش می بیند و پیش بینی هایش روی داده های Dev نادرست می شود.

راه حل: داده های آموزشی را بررسی کنیم و موارد غیرعادی را حذف کنیم. همچنین، الگوریتم هایی برای شناسایی و حذف داده های جعلی پیاده سازی کنیم.

تبلیغات روی هم تأثیر می گذارند

گاهی اوقات تبلیغاتی که در کنار هم نمایش داده می شوند، روی یکدیگر تأثیر می گذارند. برای مثال، اگر یک تبلیغ خیلی جذاب در بالای صفحه نمایش داده شود، ممکن است تبلیغات پایین تر کمتر کلیک بگیرند، اما مدل این موضوع را در نظر نگرفته باشد.

راه حل : در مدل های پیش بینی CTR و CVR ، موقعیت تبلیغ و تعامل آن با تبلیغات دیگر را هم در نظر بگیریم.

مقدار bid در مدل و سیستم واقعی فرق دارد

مدل ما درآمد را بر اساس فرمول های $bid \times CTR \times CVR$ و $bid \times CTR$ پیش بینی می کند. اگر مقدار واقعی bid که در سیستم تبلیغاتی استفاده می شود، با مقدار bid ی که در مدل استفاده شده فرق داشته باشد، درآمد واقعی و پیش بینی شده متفاوت خواهد بود.

راه حل : اطمینان حاصل کنیم که داده های bid که در مدل استفاده شده، با مقدار واقعی یکسان باشد.

کیفیت تبلیغات در نظر گرفته نشده

ممکن است برخی تبلیغات از نظر بصری و محتوایی ضعیف باشند، اما مدل این فاکتورها را نادیده گرفته باشد. مدل فقط بر اساس داده های عددی مثل CTR و bid تصمیم می گیرد، اما کیفیت خود تبلیغ (مثلًا متن، تصویر یا ویدیو) روی نرخ کلیک و نرخ تبدیل تأثیر دارد.

راه حل : ویژگی هایی مثل نوع تبلیغ، طراحی بصری و کیفیت محتوای آن را هم به مدل اضافه کنیم تا پیش بینی دقیق تری داشته باشیم.

چطور این مشکل را حل کنیم؟

بررسی کنیم که آیا داده های آموزشی و Dev از نظر آماری شبیه هم هستند یا نه.

مدل را به روز کنیم تا تغییرات رفتار کاربران را یاد بگیرد.

داده های نویزی یا جعلی را حذف کنیم.

تأثیر موقعیت تبلیغ و تعامل تبلیغات با یکدیگر را در نظر بگیریم.

مطمئن شویم که bid مورد استفاده در مدل، همان bid واقعی است.

ویژگی های کیفی تبلیغ را در پیش بینی نرخ کلیک و نرخ تبدیل لحاظ کنیم.

ج

مدیریت Drift Concept تغییر ناگهانی در رفتار کاربران

برای مقابله با تغییرات ناگهانی در رفتار کاربران، چندین راهکار مؤثر وجود دارد:

(الف) به روزرسانی مداوم مدل (Incremental Learning)

به جای بازآموزی کامل مدل در بازه های زمانی طولانی، می توان از روش های یادگیری تدریجی (Incremental Learning) استفاده کرد که در آن مدل به صورت پیوسته و با داده های جدید به روز می شود.

مزیت: مدل می تواند سریع تر با تغییرات ناگهانی (مانند فصل ها، تخفیف های ویژه، یا رویدادهای خاص) سازگار شود.

(ب) پنجره های زمانی متحرک (Sliding Window)

به جای استفاده از کل داده های قدیمی، فقط از داده های اخیر (مثلاً داده های دو هفته اخیر) برای آموزش یا تنظیم مدل استفاده کنید. این روش باعث می شود مدل بیشتر روی روندهای جدید مرکز باشد.

مزیت: مدل سریع تر تغییرات ناگهانی را شناسایی و تطبیق می دهد.

(ج) الگوریتم های تشخیص Drift (Concept Drift Detection)

استفاده از الگوریتم های تشخیص تغییر، مانند :

DDM (Drift Detection Method)

ADWIN (Adaptive Windowing)

Page-Hinkley Test

این الگوریتم ها در صورت تشخیص تغییر ناگهانی در توزیع داده ها، به طور خودکار مدل را مجددآموزش می دهند.

مزیت: مدل به طور هوشمند با تغییرات ناگهانی تنظیم می شود.

د) مدل های ترکیبی(Ensemble Models)

استفاده از چندین مدل به صورت همزمان که هرکدام برای بازه های زمانی مختلف آموزش دیده اند در این روش، مدل های مختلف بسته به شرایط و نوع تغییرات فعال یا غیرفعال می شوند.

مزیت: انعطاف پذیری بالا در شرایط ناپایدار.

مدیریت داده های نامتوازن(Imbalanced Data)

در مسئله تبلیغات، داده های نامتوازن مثلًا CTR پایین برای برخی تبلیغات بسیار رایج است. برای مقابله با این مشکل:

الف) نمونه برداری مجدد(Resampling)

Over-sampling: افزایش تعداد نمونه های کلاس های کم تعداد مانند CTR بالا

Under-sampling: کاهش تعداد نمونه های کلاس های پر تعداد مانند CTR پایین

مزیت: توزیع داده ها متوازن تر شده و مدل در پیش بینی موارد نادر دقیق تر عمل می کند.

ب) الگوریتم های ویژه برای داده های نامتوازن

استفاده از الگوریتم هایی که برای مدیریت داده های نامتوازن طراحی شده اند، مثل :

SMOTE (Synthetic Minority Over-sampling Technique)

Balanced Random Forest

Cost-Sensitive Learning: افزایش جریمه برای پیش بینی های اشتباه روی کلاس های کم تعداد

مزیت: بهبود عملکرد مدل در شرایطی که داده های CTR یا CVR نامتوازن است.

ج) تنظیم وزن نمونه ها(Class Weighting)

در بسیاری از الگوریتم های یادگیری ماشین مانند Logistic Regression ، LightGBM و XGBoost می توان وزن کلاس ها را تنظیم کرد تا مدل به کلاس های کم تعداد توجه بیشتری داشته باشد.
مزیت : بدون تغییر در داده ها، تعادل در پیش بینی مدل ایجاد می شود.

بهبود عملکرد سیستم با الگوریتم های یادگیری آنلاین(Online Learning)

یادگیری آنلاین برای سیستم های تبلیغاتی بسیار مفید است، زیرا:
مدل به صورت هم زمان با ورود داده های جدید آپدیت می شود.
امکان شناسایی تغییرات ناگهانی در لحظه فراهم می شود.
منابع محاسباتی کمتر مصرف می شود زیرا نیاز به بازآموزی کامل مدل نیست.

روش های پیشنهادی برای یادگیری آنلاین:

Stochastic Gradient Descent (SGD): یادگیری تدریجی با بروزرسانی پارامترها در هر مرحله.
Bayesian Online Learning: مدل با اضافه شدن داده های جدید به طور مداوم پارامترهای خود را اصلاح می کند.
Online Random Forest: نسخه ای از Random Forest که به صورت مرحله ای به روزرسانی می شود.

محمد حقیقت - 403722042

برای رفع برخی ایرادات و ابهامات از Chatgpt استفاده شده است.