پوریا یزدانی **۴۰۰۲۴۳۰۸۲**

تمرین دوم - بخش تئوری

سوال اول)

دو روش اصلی برای این کار وجود دارد که شامل یک در مقابل همه (OvA) و یک در مقابل یک (OvO) است.

در روش OvA با فرض داشتن N کلاس مختلف، N مدل binary classification روی داده فیت می‌شودند که وظیفه هر کدام این است که پیشبینی کنند هر داده آیا متعلق به کلاس N_1 هستند یا خیر. در واقع در این روش برای هر مدل دودویی از تمام مجموعه‌داده استفاده می‌کنیم و برچسب‌ها را صرفا دودویی می‌کنیم. بدیهی است که در اینجا با مشکل imbalanced dataset مواجه خواهیم بود و نیازمند تکنینک‌های نمونه‌برداری خواهیم بود. پس از آموزش این N مدل در مرحله inference خروجی هر مدل یک عدد احتمالی خواهد بود که با در نظر گرفتن آن مدلی که بیشترین عدد را خروجی داده بود آن کلاس را برچسب گذاری می‌کنیم.

در روش OvO با فرض داشتن N کلاس برای هر جفت کلاس ممکن باید مدل آموزش دهیم یعنی $N(N-1)/2$ مدل. به عنوان مثال اگر ۳ کلاس A, B و C را داشته باشیم باید ۶ مدل آموزش دهیم که هر کدام به ترتیب مسئول تشخیص موارد پایین خواهند بود.

- A vs B
- A vs C
- B vs C

برای آموزش این مدل‌ها ما باید مجوعه‌داده اصلی برای آن مدل هرس کنیم. یعنی برای مدل اول فقط داده‌هایی را در نظر میگیریم که یا از کلاس A هستند یا B. بنابراین برای هر مدل نسبت به حالت قبل بار محاسباتی کمتری خواهیم داشت زیرا هنگام آموزش مجموعه‌داده کوچک‌تر شده است. پس از آموزش مدل‌ها و در مرحله inference یک سیستم رای‌گیری به مانند مدل‌های ensemble به اجرا گذاشته می‌شود و کلاسی که بیشترین رای را داشت به عنوان برچسب داده ورودی در نظر گرفته می‌شود.

شایان ذکر است که روش اول سبکتر است زیرا مدل‌های کمتری فیت می‌شوند اما روش دوم می‌تواند نتایج بهتری ارائه کند زیرا مدل‌های بیشتر استفاده می‌شوند و سیستم رای‌گیری نیز می‌تواند بسیاری از مشکلات مانند نویز در prediction را رفع کند.
در مواردی که مرز مشخصی بین کلاس‌ها نمی‌توان در نظر گرفت و نواحی‌ای در فضای ویژگی‌ها وجود دارد که متعلق به بیش از یک کلاس است هر کدام از دو روش با چالش‌های مواجه می‌شوند که آن‌ها را با هم بررسی می‌کینم.
همانطور که می‌دانیم در روش OvA ما کل داده برای هر مدل در نظر می‌گیریم. در نتیجه اگر مشکل بالا را داشته باشیم. مدل‌های ما ممکن است بسیار حساس شوند و احتمال overfit بالا می‌رود همینطور مشکل دیگری که می‌تواند رخ دهد این است که اگر یک threshold برای انتخاب کلاس بر اساس خروجی‌های مدل‌ها انتخاب شود، طبق خروجی‌های مدل‌ها یک نمونه می‌تواند متعلق به چند کلاس معرفی شود که البته این مشکل با در نظر گرفتن max خروجی‌ها قابل حل است اما نمی‌تواند روش قابل اطمینانی باشد. از روش‌های حل این مشکل می‌توان به اعمال مهندسی ویژگی در جهت جدا کردن بهتر کلاس‌ها از هم اشاره کرد.
در صورت همپوشانی در روش OvO اوضاع کمی بهتر می‌شود زیرا ما برای هر مدل مجموعه داده را هرس می‌کنیم و قاعدتا در پیش‌بینی کلاس‌هایی که با هم هم‌پوشانی‌ای ندارند مشکلی نخواهیم داشت ولی همچنان مدل‌هایی که کلاس‌هایشان با هم هم‌پوشانی دارند دچار حساسیت بالا می‌شوند. البته شایان ذکر است که مدل‌های این استراتژی از برخی از پیچیدگی‌ها مانند هندل کردن بیش از دو کلاسی که با هم هم‌پوشانی دارند جلوگیری می‌کند. زیرا هر مدل صرفا بین دو کلاس پیش‌بینی می‌کند. از چالش‌هایی که در اینجا ممکن است رخ دهد نزدیک بودن رای‌ها به یکدیگر است که می‌تواند اطمینان مدل را پایین آورد. این مشکل می‌تواند با رای دادن به برخی مدل‌ها بر اساس درصد خطای آن‌ها حل شود.

سوال دوم)

با توجه به اینکه فضای حالتمان دو بعدی است و دیتاست را توانسته‌ایم به طور کامل رسم کنیم با کمی دقت می‌بینیم که با دقت تقریبا بسیار بالای داده‌ها توسط یک دایره با مرکز مبدا مختصات از یکدیگر جدا می‌شوند. از طرفی مدل‌های خطی مانند logistic regression یا اگر به صورت دقیق‌تر بخواهیم بگوییم log-linear (چون دارای اکتیویشن غیر خطی است) ترکیب

خطی‌ای از ورودی را برای انجام پیش‌بینی استفاده می‌کنند در نتیجه در فضای دو بعدی تنها یک **خط** را می‌توانند به عنوان decision boundary یاد بگیرند یا در فضایی با ابعاد بالاتر یک ابر صفحه را. از آنجایی که در اینجا کلاس‌ها با یک دایره از هم جدا می‌شوند به صورت ذاتی این decision boundary غیر خطی است بنابراین از یک مدل خطی به **صورت خام نمی‌توان** برای طبقه‌بندی این کلاس‌ها استفاده کرد.

اما با تکنیک‌های مهندسی ویژگی که وجود دارد و در اسلاید هم مطرح شده است(تابع XOR)، با استفاده از یک feature mapping ساده می‌توان با همان مدل‌های خطی مسئله را حل کرد.

برای یافتن این mapping معادله دایره در صفحه را بررسی می‌کنیم.

$$(x_1 - a)^2 + (x_2 - b)^2 = r$$

می‌دانیم a و b مختصات مرکز دایره است.

در دیتاست داده شده کلاس آبی در بیرون یک دایره به مرکز مبدا مختصات و کلاس قرمز در درون آن قرار گرفته است. بدیهی است که ما با ترکیب ورودی‌های x_1 و x_2 به تنهایی نمی‌توانیم این دایره را یاد بگیریم. در نتیجه با در نظر گرفتن

$$(x_1)^2 + (x_2)^2 = x_3$$

به عنوان ویژگی سوم اقدام به نگاشت داده از فضای دوبعدی به فضای سه‌بعدی می‌کنیم. در این فضای ۳ بعدی جدید این decision boundary دایره شکل تبدیل به یک صفحه شده که به راحتی توسط مدل‌های خطی قابل یادگیری است.

در نتیجه با اعمال ماتریس feature mapping پایین روی داده‌ها مسئله را با مدل خطی حل کنیم.

$$\phi(x) = \left[x_2 x_2 (x_1)^2 + (x_2)^2\right]^T$$

سوال سوم)

انتخاب تابع خسارت در عملکرد مدل و به ویژه همگرایی مدل و نرخ آن در آموزش بسیار تاثیر می‌گذارد و دارای تاثیر مستقیم در مرحله inference مدل و مواجه شدن با نمونه‌های جدید می‌باشد. در ادامه چند نمونه پرکاربرد از توابع خسارت پرکاربرد در مدل‌های طبقه‌بندی خطی را با هم و تفاوت‌هایشان را بررسی می‌کنیم.

Hinge loss

این تابع بیشتر در SVMها استفاده می‌شود و به صورت زیر تعریف می‌شود.(با فرض y که می‌تواند مقادیر ۱ و -۱ را اخذ کند.)

$$L_{hinge} = max(0, 1 - y_i (w^T x_i + b))$$

این تابع خسارت تشویق به دسته‌بندی صحیح، همراه با ایجاد حاشیه‌ای حداقل به اندازه ۱ می‌کند در واقع زمانی که حاشیه کمتر از ۱ باشد خطی است و در غیر این صورت برابر با صفر است. در نتیجه نه تنها خطاهای طبقه‌بندی را جریمه می‌کند، بلکه طبقه‌بندی‌های صحیحی که در محدوده حاشیه هستند را نیز جریمه می‌کند.

از آنجا که خسارت برای نقاط اشتباه طبقه‌بندی‌شده به صورت خطی افزایش می‌یابد، داده‌های پرت می‌توانند با ایجاد زیان‌های بزرگ تاثیر قابل توجهی بر مدل داشته باشند. این ویژگی مدل را در برابر تغییرات کوچک در داده مقاوم می‌کند و تعمیم‌پذیری آن به داده‌های دیده‌نشده را بهبود می‌بخشد. از آنجایی که این تابع در نقطه ۱ مشتق‌پذیر نیست. بهینه‌سازهایی مانند SGD این مشکل را با استفاده از زیرگرادیان‌ها حل می‌کنند.

سرعت همگرایی ممکن است در مقایسه با توابع زیان smoothتری مانند Cross-Entropy کندتر باشد، به‌ویژه اگر بهینه‌ساز در محاسبه‌ی زیرگرادیان‌ها دچار مشکل شود.

Logistic loss

این تابع خسارت که در logistic regression استفاده می‌شود، در واقع احتمال اینکه y=1 باشد را مدل می‌کند و به صورت پایین تعریف می‌شود.

$$L_{logistic}(y_i, f(x_i)) = log(1 + e^{-y_i f(x_i)})$$

این تابع زیان به طور یکنواخت خطاهای طبقه‌بندی را جریمه می‌کند، به طوری که با افزایش اشتباه در پیش‌بینی‌ها، میزان جریمه نیز افزایش می‌یابد. در همه نقاط قابل مشتق‌گیری است که این ویژگی برای روش‌های بهینه‌سازی مبتنی بر گرادیان یک مزیت محسوب می‌شود. همچنین محدب است و تضمین می‌کند که یک global minimum یکتا وجود دارد.

ماهیت نمایی این خسارت برای نقاط اشتباه طبقه‌بندی‌شده به این معناست که نقاطی که به شدت اشتباه طبقه‌بندی شده‌اند به اندازه زیان Hinge تأثیر نامتناسبی بر مدل ندارند و به طور کلی در مجموعه داده‌های دارای نویز، همگرایی پایداری را حاصل می‌کند.

از آنجایی که این تابع همواره مشتق‌پذیر است نسبت به دیگر تابع خسارت‌ها دارای نرخ همگرایی سریع‌تری می‌باشد زیرا می‌توانیم از الگوریتم‌های بهینه‌سازی بهتری استفاده کنیم.

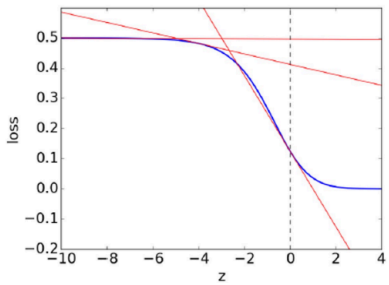
در کل در حضور داده‌های پرت، خسارت لجستیک به طور کلی نسبت به زیان Hinge مقاوم‌تر است. در حالی که هر دو تابع زیان نقاط اشتباه طبقه‌بندی‌شده را جریمه می‌کنند، نحوه جریمه آن‌ها به شکلی است که حساسیت آن‌ها به داده‌های پرت متفاوت است و برای لجیستیک مناسب تر است چون دارای ذات نمایی است. البته به طور کلی داده‌های پرت فرایند نرج همگرایی در کاهش می‌دهد و چالش ایجاد می‌کنند. البته با regularization این مشکلات قابل حل است.

Mean Square error

این تابع که در مسئله رگراسیون خطی استفاده می‌شود به صورت پایین تعریف می‌شود.

$$L_{mse} = (\widehat{y} - y_i)^2$$

با استفاده از این تابع خسارت در logsitic regression مشکلی که پیش می‌آید این است که با بزرگ‌شدن خطا مشتق این تابع رو به ۰ میل می‌کند در نتیجه باعث می‌شود که همگرایی نداشته باشیم. (با فرض استفاده از تابع اکتیویشن) همچنین نسبت به داده‌های پرت نیز بسیار حساس است.



سوال چهارم)

Subject: Date: / /

ابتدا به تعریف نگاشت‌های ویژگی چندجمله‌ای می‌پردازیم

در حالتی که تک ورودی داریم

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \end{bmatrix} \leftarrow \text{bias}$$

با در نظر گرفتن 1 در $\phi(x)$ با اضافه کردن بایاس از ابتدا در ماتریس ورودی بایاس را نیز در نظر گرفته‌ایم (در هنگام آموزش)

$$y = w^T \phi(x)$$

در حالتی که چند ورودی داشته باشیم یعنی

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

این نگاشت علاوه بر توان‌های x_i ترکیب‌های آن‌ها را نیز می‌تواند شامل شود، مثلاً

برای $n=2$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Longrightarrow \phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ (x_1)^2 \\ (x_2)^2 \\ (x_1 x_2) \end{bmatrix}$$

در این حالت داریم:

$$y^{(i)} = w^T \phi(x^{(i)}) = w_0 + \sum_{j=1}^{n} w_j x_j^{(i)} + \sum_{k,l} w_{kl} x_k^{(i)} x_l^{(i)}$$

سطر iام ماتریس داده

$$\Phi = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & (x_1)^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_1^{(n)} & x_2^{(n)} & (x_1)^2 & \end{bmatrix}$$

$$y = \Phi w$$

* در حالتی که از جواب مستقیم برای حل استفاده کنیم. (حتی الامکان)

$$w^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

Φ همان X پس از نگاشت است

بدون نگاشت $w^* = (X^T X)^{-1} X^T y$

در نتیجه می‌بینیم که ساختار بنیادین راه حل تغییر نمی‌کند.

* در حالت پیچیده‌تر با استفاده از روش‌های بر پایه گرادیان در هر مرحله داریم:

$$w_j \leftarrow w_j - \frac{\partial c}{\partial w_j} \rightsquigarrow \text{cost (average over loss)}$$

$$= w_j - \frac{\alpha}{N}\sum_{i}^{N} (\hat{y}^{(i)} - y^{(i)})\, x_j^{(i)}$$

$$\hat{y}^{(i)} = \phi(x^{(i)})\, w$$

در واقع داریم $\nabla_w J(w) = \frac{1}{m}\phi(X)^T(\phi(X)w - y)$ گرادیان

در هر حالتی که feature mapping نداشتیم صرفاً مستقیم با X کار می‌کردیم.
در نتیجه ساختار الگوریتم ثابت می‌ماند.

* مزیت اصلی این نگاشت‌ها هنگام رویارویی با مجموعه داده‌های غیرخطی است یعنی زمانی که الگوی مجموعه داده‌ای مورد نظر یک خط، صفحه یا ابرصفحه نیست.
ما با این نگاشت‌ها فضای حالت ورودی را تبدیل می‌کنیم به فضایی که در آن می‌توان روی داده خط، صفحه یا ابرصفحه فیت کرد.
* در نتیجه ما می‌توانیم بدون تغییر الگوریتم به polynomial regression دست بیابیم و توانایی تعمیم‌پذیری مدل را به شدت افزایش دهیم. *

یعنی علاوه بر داده‌های خطی، داده‌های غیرخطی را نیز یاد بگیریم.

البته باید در نظر داشته باشیم که feature mapping می‌تواند منجر به feature explosion شود که هم بار محاسباتی را بالا ببرد و هم احتمال overfit شدن را. در نتیجه عمومی‌سازی مدل پایین می‌آید. اگر کم استفاده کنیم underfit رخ می‌دهد یعنی مدل حتی

Subject: Date:

هرچقدر داده آموزش را بیشتر یاد بگیرد بازهم عمومی سازی آن مطلوب نیست. باید به دنبال حالت میانی این دو بود تا به نقطه optimal رسید