دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پروژه پایانی درس یادگیری ماشین

نگارش
امین عبدی‌پوراصل
۴۰۱۱۳۳۰۱۱

استاد درس
دکتر قنبری

بهمن ۱۴۰۲

فهرست مطالب

صفحه

فهرست اشکال

صفحه

شما همچنین می‌توانید از طریق لینک گیت‌هاب به کدها به صورت کامل نیز دسترسی داشته باشید.

۱- مقدمه: هدف مقاله

چکیده - در حوزه الگوریتم‌های یادگیری ماشین، اهمیت تابع ضرر، به‌ویژه در وظایف یادگیری تحت نظارت، اهمیت زیادی دارد. این به عنوان یک ستون اساسی عمل می‌کند که عمیقاً بر رفتار و کارایی الگوریتم‌های یادگیری نظارت شده تأثیر می‌گذارد. توابع از دست دادن سنتی، در حالی که به طور گسترده مورد استفاده قرار می‌گیرند، اغلب برای رسیدگی به داده‌های پر سر و صدا و با ابعاد بالا، مانع از تفسیرپذیری مدل می‌شوند و منجر به همگرایی آهسته در طول آموزش می‌شوند. در این مقاله، ما محدودیت‌های فوق‌الذکر را با پیشنهاد یک تابع از دست دادن قوی، محدود، پراکنده و صاف (RoBoSS) برای یادگیری نظارت شده بررسی می‌کنیم. علاوه بر این، ما تابع ضرر RoBoSS را در چارچوب ماشین بردار پشتیبان (SVM) ترکیب می‌کنیم و یک الگوریتم قوی جدید به نام Lrbss-SVM را معرفی می‌کنیم. برای تجزیه و تحلیل نظری، ویژگی طبقه بندی کالیبره شده و توانایی تعمیم نیز ارائه شده است. این تحقیقات برای به دست آوردن بینش عمیق تر در مورد عملکرد عملکرد از دست دادن RoBoSS در وظایف طبقه بندی و پتانسیل آن برای تعمیم به داده‌های نادیده بسیار مهم است. برای نشان دادن تجربی اثربخشی Lrbss-SVM پیشنهادی، آن را بر روی ۸۸ مجموعه داده‌های UCI و KEEL دنیای واقعی از حوزه‌های مختلف ارزیابی می‌کنیم. علاوه بر این، برای نشان دادن اثربخشی Lrbss-SVM پیشنهادی در حوزه زیست پزشکی، آن را بر روی دو مجموعه داده پزشکی ارزیابی کردیم: مجموعه داده سیگنال الکتروانسفالوگرام (EEG) و مجموعه داده سرطان پستان (BreaKHis). نتایج عددی برتری مدل پیشنهادی Lrbss-SVM را هم از نظر عملکرد تعمیم قابل‌توجه و هم کارایی آن در زمان آموزش اثبات می‌کند. کد مدل پیشنهادی به صورت عمومی‌در https://github.com/mtanveer1/RoBoSS در دسترس است.

در این مقاله، ما یک بررسی عمیق از رابطه متقابل بین توابع خطا و الگوریتم یادگیری نظارت شده، با استفاده از چارچوب ماشین بردار پشتیبان انجام می‌دهیم. این مطالعه صرفاً بر روی کار طبقه بندی باینری متمرکز شده است.

۲- ماشین بردار پیشتیبان[1] (SVM)

یادگیری ماشینی نظارت شده[2] (SML) یک پارادایم قدرتمند در یادگیری ماشینی است که در آن یک مدل از داده‌های برچسب‌گذاری شده یاد می‌گیرد تا روی نمونه‌های دیده نشده پیش‌بینی کند. کلید این فرآیند مفهوم توابع خطا[3] است که اختلاف بین خروجی‌های پیش بینی شده و واقعی را کمیت می‌کند. SVM نشان دهنده یک الگوریتم SML کارآمد است. این روش بر اساس مفهوم کمینه سازی ریسک ساختاری[4] (SRM) است و ریشه در تئوری یادگیری آماری[5] (SLT) دارد و پایه نظری قوی و توانایی تعمیم خوبی برای آن فراهم می‌کند.

مجموعه آموزشی با زوج مرتب‌های نمونه و برچسب $\{x_k, y_k\}_{k=1}^{n}$تعریف می‌شود، که در آن $x_k \in \mathbb{R}^m$بردار نمونه را نشان می‌دهد و $y_k \in \{۱, -۱\}$نشان دهنده برچسب مربوط به کلاس است. هدف SVM ساخت یک ابر صفحه تصمیم $w^\top x + b = 0$ با بایاس $b \in \mathbb{R}$ و وزن بردار $w \in \mathbb{R}^m$ است که با داده‌های آموزشی تخمین زده می‌شود. برای یک نقطه داده آزمایشی $\tilde{x}$، برچسب کلاس مربوطه $\tilde{y}$به صورت ۱ پیش‌بینی می‌شود اگر $w^\top \tilde{x} + b \geq 0$و در غیر این صورت -۱ پیش بینی می‌شود.

ماشین بردار پشتیبان یک الگوریتم یادگیری ماشینی نظارت شده است که برای طبقه‌بندی و رگرسیون استفاده می‌شود. هدف اصلی الگوریتم SVM یافتن ابر صفحه بهینه در فضای N بعدی (فضای ویژگی) است که بتواند نقاط داده را در طبقه‌های مختلف در فضای ویژگی جدا کند.هایپرپلن سعی میکند که حاشیه بین نزدیکترین نقاط طبقات مختلف تا حد امکان حداکثر باشد. بعدهایپرپلن به تعداد ویژگی‌ها بستگی دارد. اگر تعداد ویژگی‌های ورودی دو باشد، آنگاه‌هایپرپلن فقط یک خط است (شکل۱). اگر تعداد ویژگی‌های ورودی سه باشد، آنگاه‌هایپرپلن به یک صفحه دو بعدی تبدیل میشود.

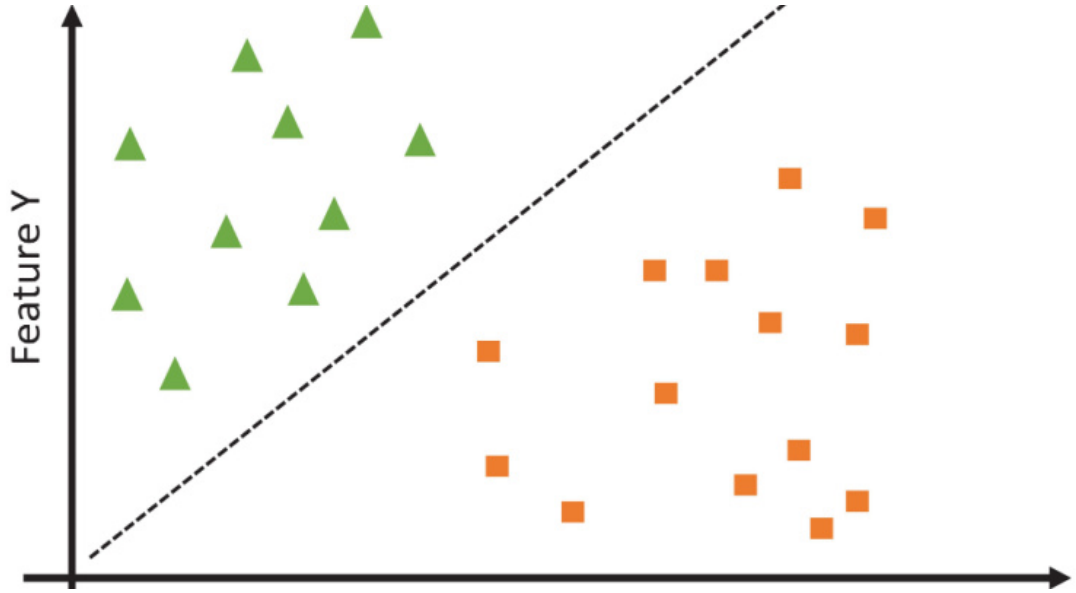
[1] Support Vector Machine

[2] Supervised Machine Learning

[3] Loss Functions

[4] Structural Risk Minimization

[5] Statistical Learning Theory



شکل ۱: یک نمونه طبقه‌بندی با استفاده از SVM برای فضای ویژگی ۲ بعدی

علاوه بر خط رسم شده، خطوط بسیار زیادی وجود دارند که می‌توانند طبقه‌بندی را انجام دهند. برای پیدا کردن بهترین خط (یا صفحه و ابرصفحه در ابعاد بالاتر) با استفاده از یک مساله بهینه‌سازی که می‌آید فاصله حاشیه خط از دو طرف تا هر دو کلاس دیتا را ماکزیمم می‌کند. به دلیل صحبت به تفصیل الگوریتم آن و تابع بهینه‌سازی این الگوریتم در کلاس، از شرح نحوه رسیدن به مساله اصلی و همچنین دوگان مساله اصلی اجتناب می‌کنیم و مستقیما تابع هدف بهینه سازی دوگان را در اینجا می‌آوریم. این مساله می‌خواهد فاصله حاشیه خط طبقه‌بندی کننده تا دادگان دو دسته را بیشینه کند.

$$\underset{\alpha}{\text{maximize}} : \frac{1}{2} \sum_{i \to m} \sum_{j \to m} \alpha_i \alpha_j t_i t_j K(x_i, x_j) - \sum_{i \to m} \alpha_i$$

کرنل SVM تابعی است که فضای ورودی را میگیرد و آن را به فضایی با ابعاد بالاتر تبدیل میکند، یعنی مسائل غیرقابل تفکیک را به مسائل قابل تفکیک تبدیل میکند که بیشتر در مسائل جداسازی غیر خطی مفید است. به زبان ساده، کرنل، تبدیل‌های داده بسیار پیچیده را انجام می‌دهد و سپس فرآیند جداسازی داده‌ها را بر اساس برچسب‌ها یا خروجی‌های تعریف‌شده پیدا می‌کند. این کرنل همان تابع K در رابطه بالا می‌باشد. در زیر معروف‌ترین کرنل‌های SVM را مشاهده می‌نمایید.

$$\text{Linear}: \ K(w,b) = w^T x + b$$
$$\text{Polynomial}: \ K(w,x) = (\gamma w^T x + b)^N$$
$$\text{Gaussian RBF}: K(w,x) = \exp(-\gamma ||x_i - x_j||^n$$
$$\text{Sigmoid}: K(x_i, x_j) = \tanh(\alpha x_i^T x_j + b)$$

برای به دست آوردن ابر صفحه بهینه، دو حالت را می‌توان در فضای ورودی در نظر گرفت: مجموعه داده‌های آموزشی قابل جداسازی خطی و مجموعه داده‌های آموزشی غیرقابل تفکیک خطی. برای حالت جداسازی خطی مدل زیر را خواهیم داشت:

$$\min_{w,b} \frac{1}{2} \| w \|^2$$
$$\text{subject to } y_k(w^\top x_k + b) \geq 1, \forall k = 1,2, \dots, n.$$

که باید پارامترهای بهینه w و b را محاسبه کنیم. این مدل به عنوان SVM حاشیه سخت[1] نامیده می‌شود زیرا لازم است هر نمونه آموزشی به درستی طبقه بندی شود. برای وضعیت غیرقابل تفکیک خطی، رویکرد پرکاربرد اجازه طبقه‌بندی اشتباه را می‌دهد و این اشتباهات را با گنجاندن تابع خطا در تابع هدف پنالتی می‌کند، که منجر به مشکل بهینه‌سازی نامحدود زیر می‌شود:

$$\min_{w,b} \frac{1}{2} \| w \|^2 + \frac{\gamma}{n} \sum_{k=1}^{n} \mathcal{L}\left(1 - y_k(w^\top x_k + b)\right), \gamma > 0$$

در این رابطه $\gamma > 0$ پارامتر trade-off است و $\mathcal{L}(u)$ که $u := ۱ - y_k(w^\top x_k + b)$ نشان دهنده تابع خطا است. از آنجایی که این مدل امکان طبقه‌بندی نادرست نمونه‌ها را می‌دهد، از آن به عنوان یک مدل SVM حاشیه نرم[2] نام برده می‌شود.

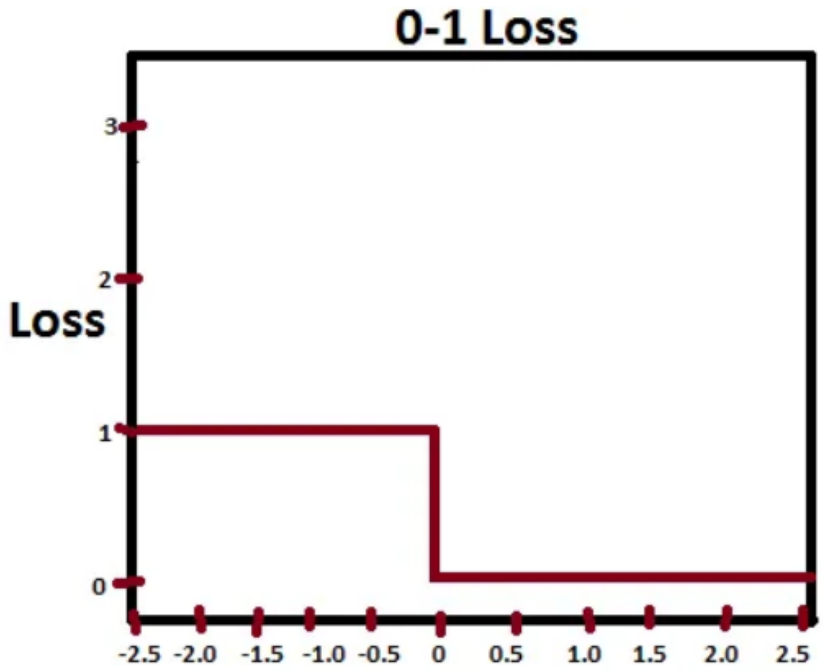
[1] Hard-Margin SVM

[2] Soft-Margin SVM

۳- توابع خطا

تابع خطای L(u) جز ضروری ماشین بردار پشتیبان است کـه اسـتحکام و پراکنـدگی SVM را کنتـرل می‌کند. تابع خطای ۰-۱ به عنوان یک تابع خطای ایده آل تعریف می‌شود که خطای ثابت ۱ را به همـه نمونه‌های طبقه بندی اشتباه و خطای ۰ را به نمونه‌های طبقه بندی شده به درستی اختصاص می‌دهد.

$$\mathcal{L}_{0-1}(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases}$$



شکل ۲: تابع خطای ۰-۱

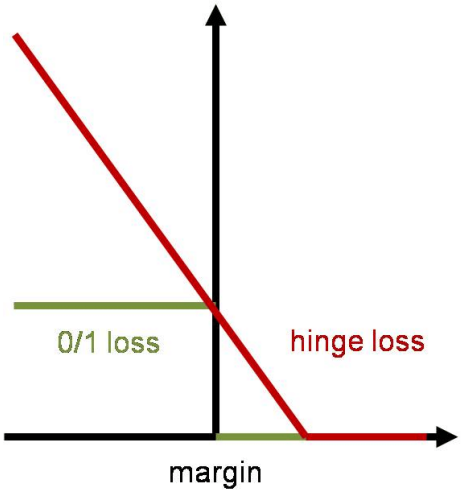
حل SVM با تابع خطای ۰-۱ NP-hard است، زیرا ناپیوسته و غیر محدب است. بـرای توسـعه SVM، کارهای زیادی برای ساخت توابع ضرر جدید انجام شده است تا مـدل‌های SVM بـا حاشـیه نـرم مـوثر جدید به دست آید. در این بخش، ما به طور مختصر چند توابع خطای معروف را بررسی خواهیم کرد.

۳-۱- تابع خطای hinge

اولین مدل SVM با حاشیه نرم Lhinge-SVM SVM است که از تـابع خطـای $\mathcal{L}_{\text{hinge}}(u)$ اسـتفاده می‌کند این خطا، خطای ناشی از پیش بینی کلاس اشتباه را بـرای یـک نمونـه انـدازه گیـری می‌کنـد.

پیش‌بینی‌هایی که درست هستند اما مطمئن نیستند را جریمه می‌کند و طبقه‌بندی‌های اشتباه را به شدت جریمه می‌کند. این تابع محدب، غیر صاف و نامحدود است.

$$\mathcal{L}_{\text{hinge}}(u) = \begin{cases} u, & u > 0, \\ 0, & u \leq 0. \end{cases}$$

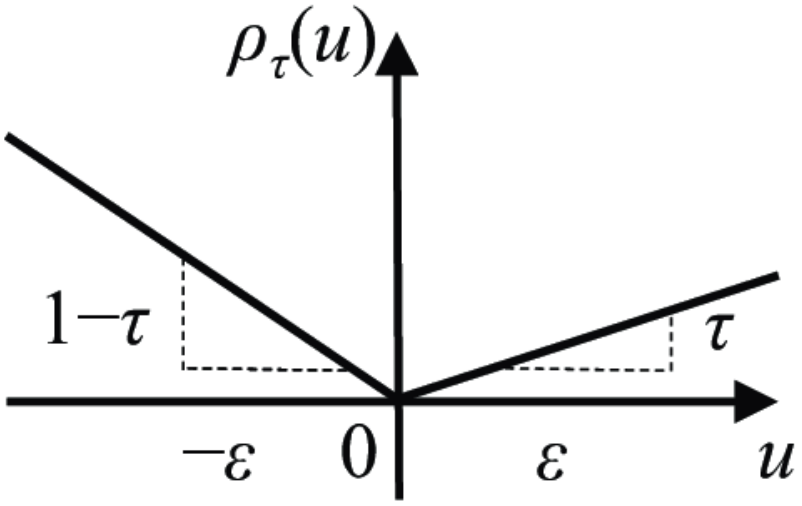


شکل ۳: مقایسه تابع خطای hinge و تابع خطای ۰-۱

۳-۲- تابع خطای pinball

خطای پین بال که همچنین به عنوان خطای چندک یا کاهش رگرسیون چندک نیز شناخته می‌شود، یک نوع تابع خطا است که در رگرسیون چندک استفاده می‌شود. این خطا انحراف بین چندک‌های پیش بینی شده و چندک‌های هدف واقعی را اندازه گیری می‌کند. این تابع به عنوان تفاوت مطلق بین مقدار هدف واقعی و مقدار پیش‌بینی‌شده تعریف می‌شود که با پارامتری به نام tau (τ) وزن می‌شود که سطح چندک را نشان می‌دهد.

$$\mathcal{L}_{\text{pin}}(u) = \begin{cases} u, & u > 0, \\ -\tau u, & u \leq 0, \end{cases} , \tau \in [۰,۱]$$



شکل ۴: تابع خطای pinball

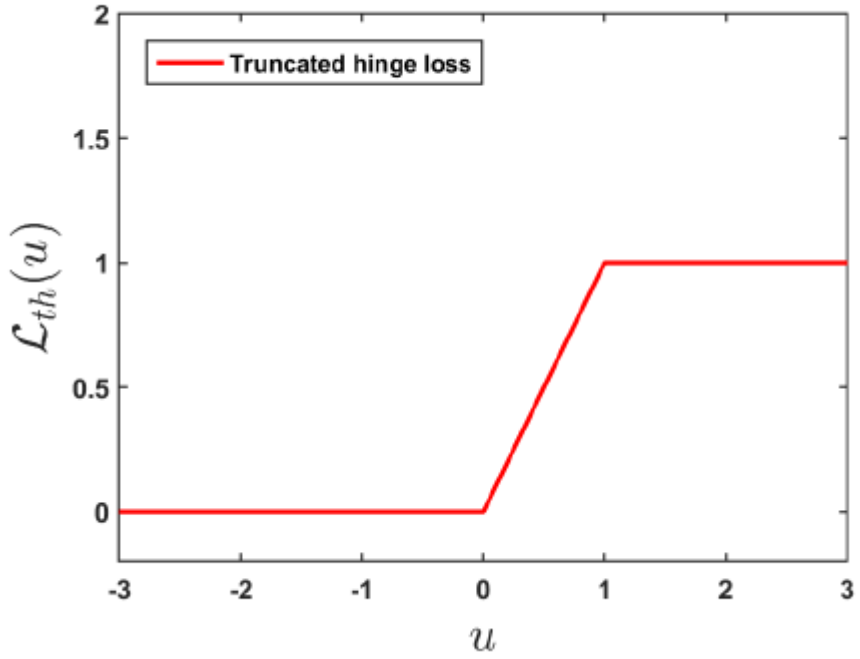
برای $\tau = 0$، تابع خطای پین بال به تابع hinge تبدیل می‌شود.تابع خطـای پین‌بـال نیـز محـدب، غیـر صاف و نامحدود است.

۳-۳- تابع خطای Truncated hinge

توابع خطای محدب، دارای بهینه منحصر به فرد هستند، استفاده از آنها آسان است و می‌توان آنهـا را بـا استفاده از ابزارهای بهینه سازی محدب به طور موثر بهینه کرد، به طوری که طبقه بندی کننده مربوطه را مستعد تحت تأثیر قرار گرفتن یا تسلط بـیش از حـد عوامـل پـرت و نـامربوط می‌کنـد. بـرای بهبـود استحکام، توابع مختلف خطای محدود در ادبیات پیشنهاد شده است. به منظور افزایش استحکام hinge-SVM، تابع خطای hinge کوتاه[1] $\mathcal{L}_{th}(u)$ را توسعه دادند. این تابع غیر محـدب، غیـر صـاف و محـدود است.

$$\mathcal{L}_{th}(u) = \begin{cases} \delta, & u \geq \delta, \\ u, & u \in (0, \delta), \\ 0, & u \leq 0, \end{cases} \qquad \delta \geq 1$$

[1] Truncated hinge loss function

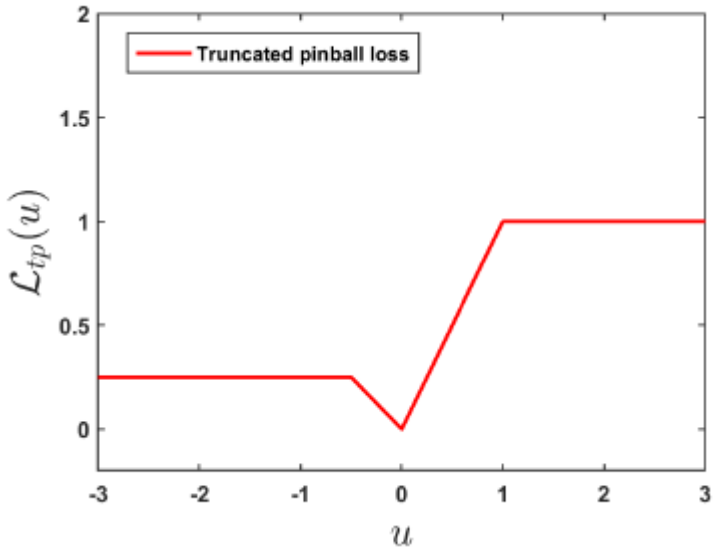


شکل ۵: تابع خطای Truncated hinge

۳-٤- تابع خطای Truncated Pinball

تابع خطای پین بال کوتاه، عناصر تابع خطای پین بال را با برش ترکیب می‌کند، که تاثیر نقاط پرت شدید را در محاسبه خطا محدود می‌کند. این کار باعث استحکام تابع خطای پین بال می‌گردد و پراکندگی را به آن اضافه می‌کند. همچنین غیر محدب، غیرصاف و محدود است.

$$\mathcal{L}_{tp}(u) = \begin{cases} \delta_1, & u \geq \delta_1, \\ u, & u \in [0, \delta_1), \\ -\tau u, & u \in \left(-\frac{\delta_2}{\tau}, 0\right), \\ \delta_2, & u \leq -\delta_2/\tau, \end{cases} \qquad \tau \in [0,1], \delta_1, \delta_2 > 0$$



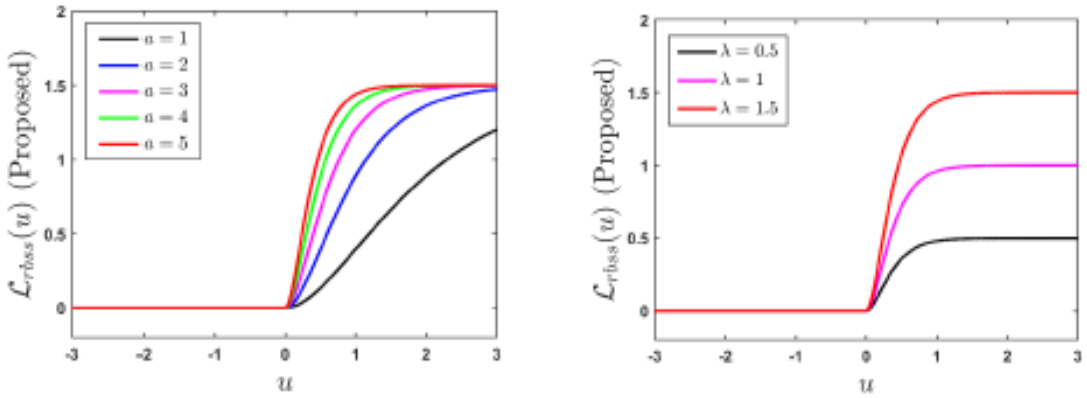
شکل ۶: تابع خطای Truncated Pinball

۴- تابع خطای معرفی‌شده مقاله

ماهیت غیر محدب و غیر هموار توابع خطای معرفی شده، چالش‌های مهمی‌را از نظر بهینه سازی محاسباتی برای حل مدل‌های SVM مربوطه ایجاد می‌کند. تمرکز اصلی این مقاله ساخت یک تابع جدید قوی، محدود، پراکنده و صاف برای یادگیری تحت نظارت است. برای بهبود استحکام، پراکندگی و صافی تلفات فوق، ما یک تابع خطای جدید به نام تابع خطای RoBoSS طراحی شده است، که به صورت زیر تعریف می‌شود:

$$\mathcal{L}_{\text{rbss}}(u) = \begin{cases} \lambda(1-(au+1)\exp(-au)), & u > 0, \\ 0, & u \le 0, \end{cases} \qquad a > 0, \lambda > 0$$

$a > 0$، $\lambda > 0$ به ترتیب پارامترهای شکل و مرز هستند. تاثیر این دو پارامتر را در شکل ۷ مشاهده می‌نمایید. توجه نمایید که از آوردن روابط ریاضی و روند رسیدن به این تابع در مقاله صرف نظر شده است اما توضیحات تابع به طور کامل آورده می‌شوند.



شکل ۷: تابع خطای RoBoSS

جدول زیرویژگی‌های مختلف توابع از دست دادن پیشرفته را با خطـای پیشـنهادی RoBoSS مقایسـه می‌کند و نشان می‌دهد که این خطای پیشنهادی همه ویژگی‌های مطلوب را دارد.

Loss function ↓\ Characteristic →	Robust	Sparse	Bounded	Convex	Smooth
Hinge loss	✗	✓	✗	✓	✗
Pinball loss	✗	✗	✗	✓	✗
Truncated hinge loss	✓	✓	✓	✗	✗
Truncated pinball loss	✓	✗	✓	✗	✗
Linex loss	✗	✗	✗	✓	✓
RoBoSS loss (Proposed)	✓	✓	✓	✗	✓

در ادامه تابع خطای RoBoSS پیشنهادی را در SVM ترکیب می‌کنیم مـدل SVM جدیـدی تعریـف می‌شود:

$$\min_{w,b} \frac{1}{2} \| w \|^2 + \frac{\gamma}{n} \sum_{k=1}^{n} \mathcal{L}_{\text{rbss}} \left(1 - y_k (w^\top x_k + b)\right).$$

۵- پیاده‌سازی مقاله

۵-۱- الگوریتم پیاده‌سازی

الگوریتم پیاده‌سازی را در زیر مشاهده می‌نمایید. در این مقاله، از چارچوب مبتنی بر گرادیان شتاب‌دار نسترو‌ف[1] (NAG) برای بهینه‌سازی Lrbss-SVM استفاده می‌کنیم. NAG به دلیل پیچیدگی محاسباتی کم و کارایی آن در رسیدگی به مشکلات در مقیاس بزرگ شناخته شده است. این رویکرد چندین مزیت از جمله کاهش نیازهای محاسباتی و بهبود سرعت را ارائه می‌کند، به‌ویژه هنگام برخورد با مشکلات در مقیاس بزرگ.

Algorithm 1 NAG-based algorithm to solve $\mathcal{L}_{rbss}$-SVM

Input:

The dataset: $\{x_k, y_k\}_{k=1}^{n}$, $y_k \in \{-1, 1\}$;

The parameters: Regularization parameter C, RoBoSS loss parameters λ and a, mini-batch size s, learning rate decay factor η, momentum parameter r, maximum iteration number N;

Initialize: model parameter β_0, velocity v_0, learning rate α;

Output:

The classifiers parameters: β;

1: Select s samples $\{x_k, y_k\}_{k=1}^{s}$ uniformly at random.

2: Computing ξ_k :

$$\xi_k = 1 - y_k \left(\sum_{j=1}^{s} \beta_j \mathcal{K}(x_k, x_j) \right), \quad k = 1, \cdots, s; \tag{20}$$

3: Temporary update: $\widetilde{\beta}_t = \beta_t + r v_t$;

4: Compute gradient:

$$\nabla f(\beta_t) = \mathcal{K}\beta - \frac{\gamma}{s}\lambda \sum_{j=1}^{s} a^2 \xi_j \exp(-a\xi_j) y_j \mathcal{K}_j, \tag{21}$$

where $\mathcal{K}$ is the Gaussian kernel matrix.

5: Update velocity: $v_t = r v_{t-1} - \alpha_{t-1} \nabla f(\beta_t)$;

6: Update model parameter: $\beta_{t+1} = \beta_t + v_t$;

7: Update learning rate: $\alpha_{t+1} = \alpha_t \exp(-\eta t)$;

8: Update current iteration number: $t = t + 1$.

Until:

$t = N$.

Return: β_t.

[1] Nestrov accelerated gradient (NAG)

برای پیاده‌سازی الگوریتم با استفاده از تابع RoBoSS_NAG_function (لینک) تابع خطای RoBoSS را با استفاده از الگوریتم NAG برای بهینه سازی پیاده سازی می‌کنیم. این تابع پارامترهای ورودی شامل داده‌های آموزشی و آزمایشی، پارامترهای خطای RoBoSS (a و b)، پارامتر trade-off (C)، پارامتر کرنل (mew)، بچ سایز (m) و حداکثر تعداد تکرار (max_iter) یادگیری را می‌گیرد و پارامتر بهینه (gamma_opt)، صحت طبقه‌بندی (Accuracy) و زمان آموزش (زمان) را برمی‌گرداند. Kو r نیز پارامترهای کم کردن میزان نرخ اصلاح و مومنتوم هستند.

```
function [gamma_opt,Accuracy,time] =
RoBoSS_NAG_function(alltrain,test,a,b,C,k,r,max_iter,t
,m,mew)
```

ابتدا از همه نمونه‌های آموزشی m تا که سایز بچ است را جدا می‌کنیم و ویژگی‌ها و برچسب‌ها را از نمونه‌های انتخاب شده استخراج می‌کنیم.

```
for i=1:m
    rand_data(i,:)=alltrain(rand_num(i),:);
end
%% xrand and yrand are the feature matrix and labels
of m randomly selected training samples.
xrand=rand_data(:,1:end-1); yrand=rand_data(:,end);

%% Split the feature and label of the Test set
Xtest=test(:,1:end-1);
Ytest=test(:,end);
```

پس از آن با استفاده از نمونه‌های انتخاب شده، ماتریس کرنل (omega) را برای داده‌های آموزشی محاسبه می‌کنیم که از هسته گاوسی با پارامتر هسته مشخص شده (`mew`) استفاده می‌کنیم.

```
%%%generating the kernel matrix for the training data
using m randomly selected training samples.
XX=sum(xrand.^2,2)*ones(1,m);

omega=XX+XX'-2*(xrand*xrand');    %%%omega is the
kernel matrix for data X.

omega=exp(-omega./(2*mew^2));
```

پس از تعیین نرخ اولیه پارامترها، تـابع خطـای RoBoss را محاسـبه و سـپس گرادیـان آن را محاسـبه می‌کنیم.

```
% initialize the parameters
n1=size(xrand,2);          % feature in dataset
eta0=0.01;                 % learning rate of NAG
algorithm
gamma=0.01*ones(m,1);      % initialize model parameter
v=0.01*ones(m,1);          % initialize velocity for NAG
algorithm



%%% finding \xi_k

q=zeros(m,1);  % This is summation term in xi_k (See
RoBoSS paper)
for i=1:m
    q(i)=sum(gamma.*omega(:,i));
end


u=zeros(m,1);    % This is xi_i
for i=1:m
    u(i)=1-(yrand(i)*q(i));
end


% derivate of loss
E=zeros(m,m);
for i=1:m
    if u(i)>0
        E(i,:)= -b*a^2*u(i)*exp(-
a*u(i))*yrand(i)*omega(i,:);
    elseif u(i) >= 0
        E(i,:)= zeros(1,m);
    end
end
```

حلقه بهینه‌سازی زیر، الگوریتم NAG را برای تکرارهای «max_iter» پیاده‌سازی می‌کند و پارامتر مدل گاما را با استفاده از الگوریتم NAG به روز می‌کند.

```
for i = 1:max_iter
    t = t + 1;

    gamma=gamma+r*v;
    grad= (gamma/l)+ (C/m)*sum(E,1)';
    v=r*v-eta0*grad;
    gamma=gamma+v;
    eta0=eta0*exp(-k*t);
end
```

پس از آن برای دادگان تست تعیین شده، ماتریس کرنل محاسبه می‌شود و برچسب‌ها را برای داده‌های آزمون بر اساس مقادیر تابع فرضیه پیش بینی می‌کند.

```
% Return optimal solution and function value
gamma_opt = gamma;

XK=xrand; %storing X in another matrix so that all the
upgradation while calculating kernel will be done in
new matrix.

p=size(Xtest,1);
%HT=zeros(m,n);
omega1=-2*XK*Xtest';
XK=sum(XK.^2,2)*ones(1,p);
Xtest=sum(Xtest.^2,2)*ones(1,m);
omega1=omega1+XK+Xtest';
omega1=exp(-omega1./2*mew^2); %%omega1 is the kernel
matrix corresponding to test data projected on
training data(including univwersum)

HT=omega1.*yrand;


f=sign(HT'*gamma_opt);
```

در نهایت صحت را با استفاده از برچسب‌های پیش‌بینی‌شده و برچسب‌های حقیقت زمینی از داده‌های آزمون محاسبه می‌کنیم.

```
%% Finding Accuracy using true positive(tp), true
negative(tn), false positive(fp) and false
negative(fn).
tp=0;tn=0;fp=0;fn=0;
for j=1:length(Ytest)
    if Ytest(j)>0
        if Ytest(j)==f(j)
            tp=tp+1;
        else
            fn=fn+1;
        end
    end
    if Ytest(j)<0
        if Ytest(j)==f(j)
            tn=tn+1;
        else
            fp=fp+1;
        end
    end
end
Accuracy=((tp+tn)/(tp+fn+fp+tn))*100;
```

۲-۵- دیتاست مورد استفاده

در این پروژه از دیتاست پایگاه داده دیابت Pima Indians Diabetes Database استفاده شده است. این پایگاه داده، دیتاست معروفی است که در یادگیری ماشینی و آمار برای مدل‌سازی و تجزیه و تحلیل پیش‌بینی‌کننده استفاده می‌شود. این دیتاست از مطالعه‌ای که توسط موسسه ملی دیابت و بیماری‌های گوارشی و کلیوی (NIDDK) انجام شد، نشات گرفت و شامل داده‌های جمع‌آوری‌شده از زنان هندی پیما در نزدیکی فینیکس، آریزونا است. مجموعه داده حاوی ویژگی‌های مختلف مرتبط با سلامتی مانند غلظت گلوکز، فشار خون، شاخص توده بدنی (BMI)، سن، و وضعیت دیابت است (اینکه آیا فرد در طی پنج سال از جمع آوری داده‌ها به دیابت مبتلا شده است).

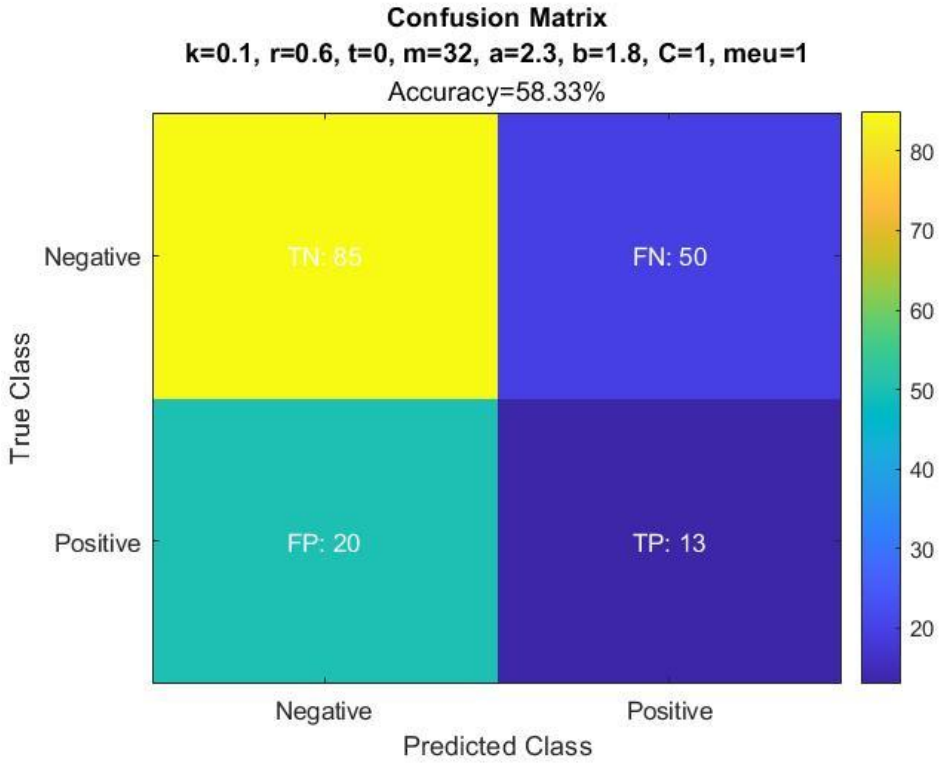
در این پروژه از این مجموعه داده برای طبقه‌بندی باینری با هدف پیش‌بینی شروع دیابت بر اساس ویژگی‌های داده شده استفاده کرده ایم. این مجموعه داده شامل ۷۶۸ نمونه با ۸ ویژگی است. از این تعداد، ۶۰۰ نمونه را به عنوان داده آموزش و مابقی را به عنوان دادگان تست در نظر گرفتیم.

۵-۳- نتایج پیاده‌سازی

در این بخش به ازای پارامترهای ورودی متفاوت نتایج را بدست می‌آوریم و تغییرات پارامترها را بررسی خواهیم کرد.

ابتدا در کد، یک بخش اضافه می‌کنیم تا در نتایج ماتریس به هم ریختگی نیز ترسیم شود. ابتدا باید برچسب‌های داده را تبدیل به -۱ و ۱ کنیم و آن سپس به صورت تصادفی از آن دادگان آموزش و تست را جدا کنیم. در ابتدای امر نتایج پیاده سازی را برای این مقادیر مشاهده می‌نمایید:

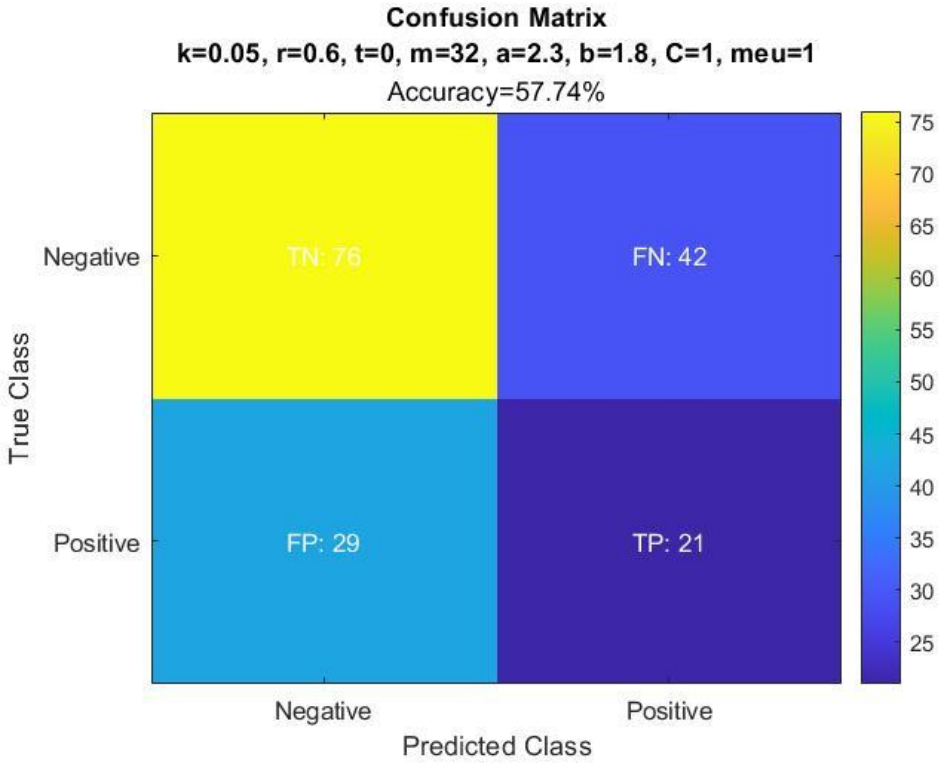
```
k = 0.1; % learning rate decay factor
r=0.6; % momentum parameter
max_iter = 1000; % maximum iteration number
t=0;
m=2^5; % mini batch size
a=2.3;  %  a and b are loss parameter
b=1.8;
C=1;  % tradeoff parameter
mew=1; % kernel parameter
```



شکل ۸: ماتریس به هم ریختگی و مقدار صحت برای پارامترهای پیش‌فرض

۵-۳-۱- تغییر k=0.05

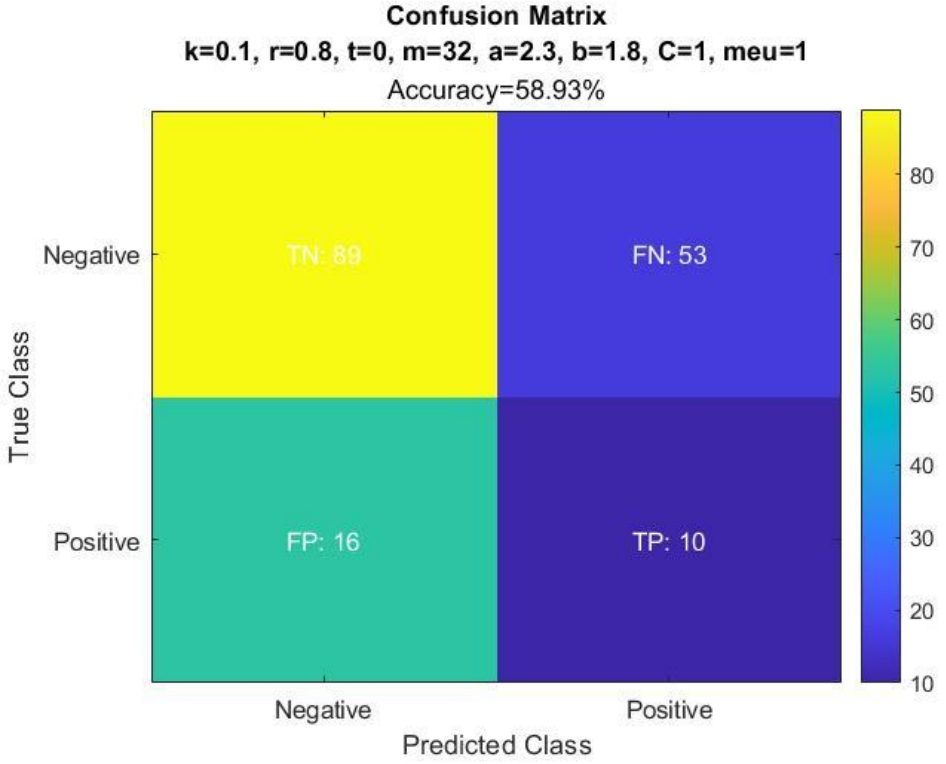
با تغییر این مقدار، در اپوک‌های نهایی مقدار نرخ یادگیری کمتری خواهیم داشت و در نتیجه انتظار داریم کمی‌بیشتر آموزش صورت بگیرد. اما این آموزش بیشتر به معنای بهتر شدن صحت طبقه‌بندی برای داده تست لزوما نخواهد بود، چون حتی ممکن است دچار overfit کردن کدل ما نیز بشود. شکل ۹ نتایج این تغییر را نشان می‌دهد.



شکل ۹: ماتریس به هم ریختگی و مقدار صحت با تغییر k=0.05

۵-۳-۲- تغییر r=0.8

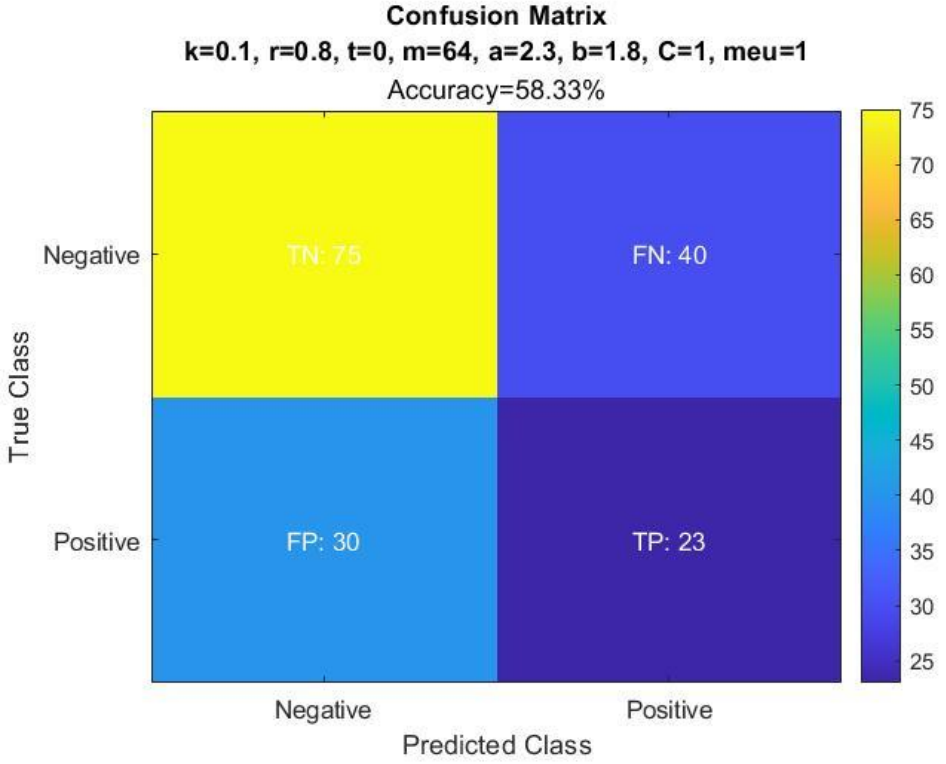
در این بخش با همان مقدار k=0.1 پیش خواهیم رفت. R مقدار ضریب مومنتوم استو هر چه بالاتر باشد، تاثیر دادگان جدید کمتر می‌شود و حتی تاثیر دادگان پرت آموزشی کمتر وارد مدل می‌شود. شکل ۱۰ نتایج این بخش را نشان می‌دهد. این کار توانسته کمی‌صحت را بالاتر ببرد.



شکل ۱۰: ماتریس به هم ریختگی و مقدار صحت با تغییر r=0.8

۵-۳-۳-تغییر سایز بچ به ۶۴

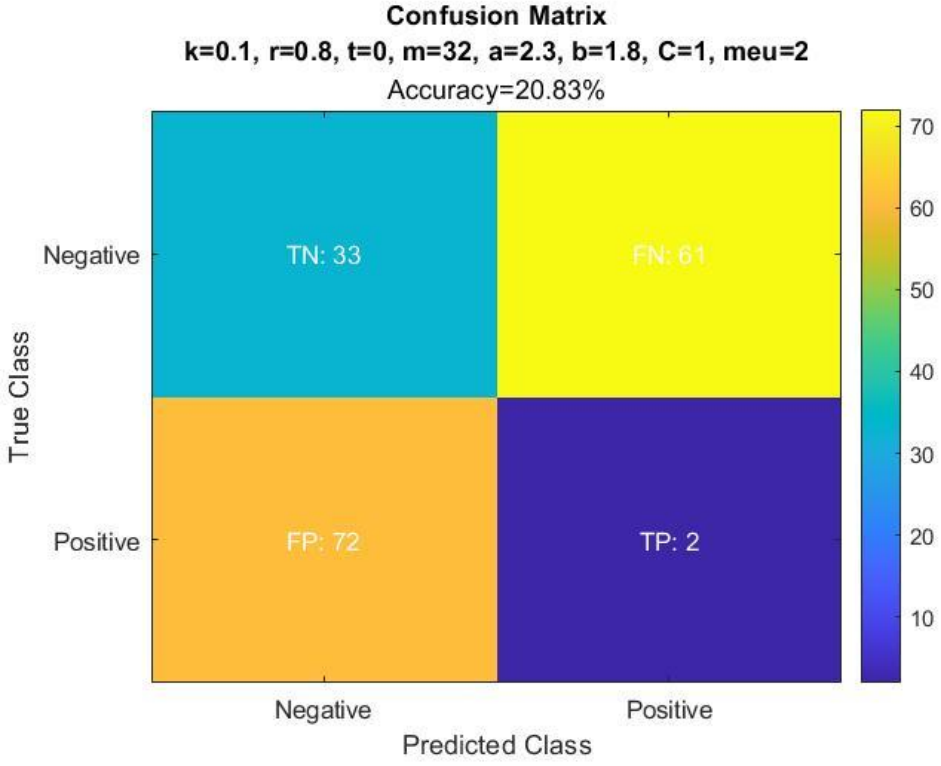
با توجه اطلاعاتی که از قبل داریم، می‌دانیم که بچ سایز در تعداد اپوک آموزشی ۱۰۰۰ نمی‌تواند تاثیر زیادی بر صحت طبقه‌بندی بگذارد و تاثیر خود را بر سرعت بیشتر آموزش با بالاتر رفتن مقدار سایز نشان می‌دهد. با این حال نتایج را برای سایز بچ ۶۴ در شکل ۱۱ مشاهده می‌نمایید. در این بخش و ادامه با مقدار r=0.8 پیش می‌رویم.



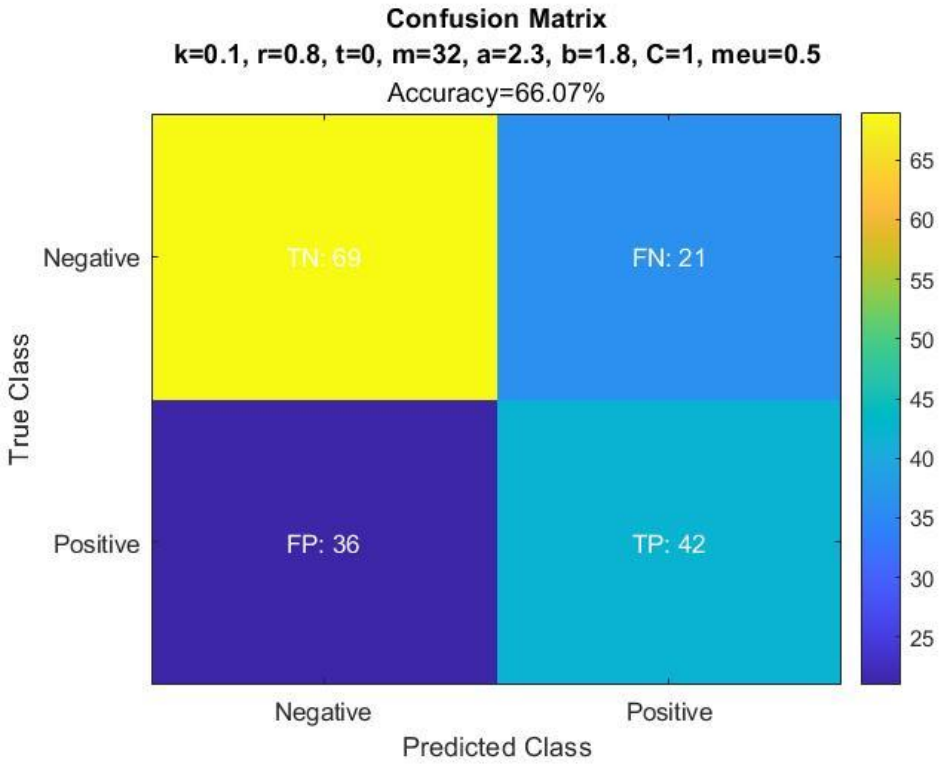
شکل ۱۰: ماتریس به هم ریختگی و مقدار صحت با تغییر بچ سایز به ۶۴

۵-۳-۴- تغییر meu

دو شکل زیر نتیجه تغییر این پارامتر از ۱ به ۲ و ۰.۵ هستند. همانطور که مشاهده می‌شود به کوچکتر کردن این پارامتر می‌توان صحت را بالاتر برد. این پارامتر بسته به دیتاست مورد استفاده می‌تواند مقدار بهینه متفاوتی داشته باشد و یک ایده شخصی برای این بخش می‌تواند پیاده سازی یک شبکه عصبی برای مشخص کردن مقدار بهینه این پارامتر باشد.



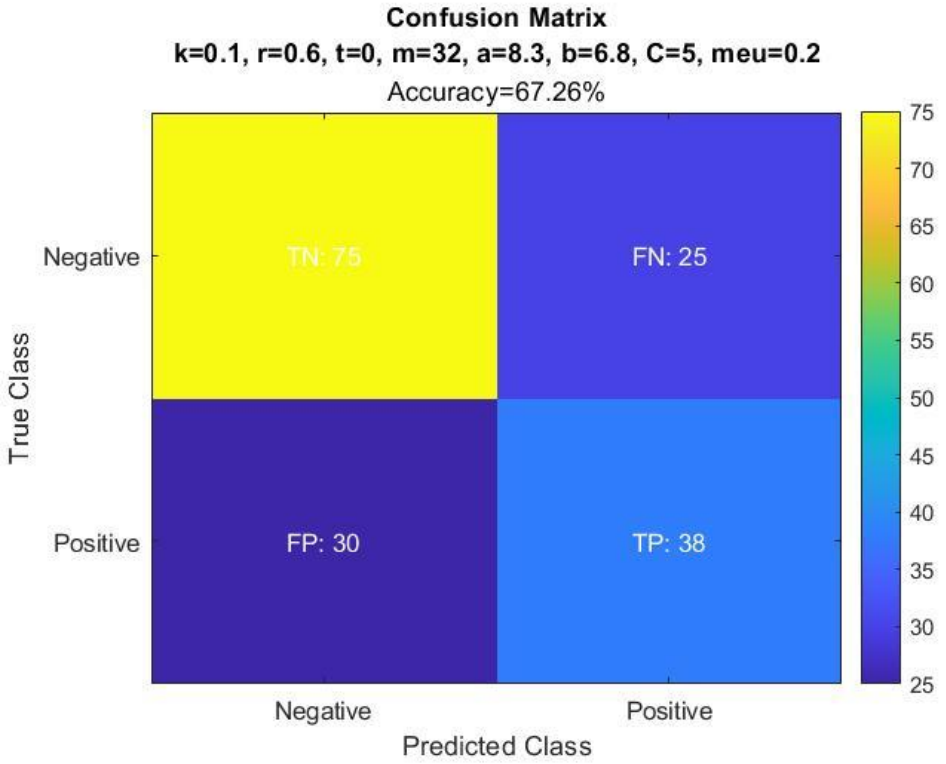
شکل ۱۱: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 2



شکل ۱۲: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 0.5

۵-۳-۵- بالابردن پارامترهای تابع خطا

با تغییر a و b با صحیح و خطا مانند حالت قبل باید پی برد که چه تاثیری روی این دیتاست خواهند گذاشت و مانند حالت قبل تاثیرها برای هر دیتاست متفاوت خواهد بود. شکل ۱۳ نمایشی از نتیجه به ازای بالابردن مقدار این پارامترها خواهد بود.



شکل ۱۲: ماتریس به هم ریختگی و مقدار صحت با پارامترهای خطا

باید توجه داشت که با هر بار ران کردن کد پیوست‌شده، به دلیل انتخاب تصادفی دادگان، میزان صحت متفاوت خواهد بود. در این بخش برای مقایسه پارامترها، با ران اول، این تصادفی انتخاب کردن برای آموزش و تست خاموش می‌شود تا بتوان با داده‌های یکسان مقایسه بین پارامترهای ورودی را نشان داد.