

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

پروژه پایانی درس یادگیری ماشین

نگارش

امین عبدی­پوراصل

401133011

استاد درس

دکتر قنبری

بهمن 1402

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

[1- مقدمه: هدف مقاله 4](#_Toc159296485)

[2- ماشین بردار پیشتیبان (SVM) 5](#_Toc159296486)

[3- توابع خطا 8](#_Toc159296487)

[3-1- تابع خطای hinge 8](#_Toc159296488)

[2-3- تابع خطای pinball 9](#_Toc159296489)

[3-3- تابع خطای Truncated hinge 10](#_Toc159296490)

[3-4- تابع خطای Truncated Pinball 11](#_Toc159296491)

[4- تابع خطای معرفی‌شده مقاله 12](#_Toc159296492)

[5- پیاده‌سازی مقاله 14](#_Toc159296493)

[1-5- الگوریتم پیاده‌سازی 14](#_Toc159296494)

[2-5- دیتاست مورد استفاده 18](#_Toc159296495)

[3-5- نتایج پیاده‌سازی 19](#_Toc159296496)

[1-3-5- تغییر k=0.05 20](#_Toc159296497)

[2-3-5- تغییر r=0.8 20](#_Toc159296498)

[3-3-5-تغییر سایز بچ به 64 21](#_Toc159296499)

[4-3-5- تغییر meu 22](#_Toc159296500)

[5-3-5- بالابردن پارامترهای تابع خطا 24](#_Toc159296501)

|  |  |
| --- | --- |
| فهرست اشكال | صفحه |

[شکل 1: یک نمونه طبقه‌بندی با استفاده از SVM برای فضای ویژگی 2 بعدی 6](#_Toc159296674)

[شکل 2: تابع خطای 0-1 8](#_Toc159296675)

[شکل 3: مقایسه تابع خطای hinge و تابع خطای 0-1 9](#_Toc159296676)

[شکل 4: تابع خطای pinball 10](#_Toc159296677)

[شکل 5: تابع خطای Truncated hinge 11](#_Toc159296678)

[شکل 6: تابع خطای Truncated Pinball 12](#_Toc159296679)

[شکل 8: ماتریس به هم ریختگی و مقدار صحت برای پارامترهای پیش‌فرض 19](#_Toc159296680)

[شکل 9: ماتریس به هم ریختگی و مقدار صحت با تغییر k=0.05 20](#_Toc159296681)

[شکل 10: ماتریس به هم ریختگی و مقدار صحت با تغییر r=0.8 21](#_Toc159296682)

[شکل 10: ماتریس به هم ریختگی و مقدار صحت با تغییر بچ سایز به 64 22](#_Toc159296683)

[شکل 11: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 2 23](#_Toc159296684)

[شکل 12: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 0.5 23](#_Toc159296685)

[شکل 12: ماتریس به هم ریختگی و مقدار صحت با پارامترهای خطا 24](#_Toc159296686)

**شما همچنین می‌توانید از طریق** [**لینک گیت‌هاب**](https://github.com/abdipourasl/Machine-Learning-1402) **به کدها به صورت کامل نیز دسترسی داشته باشید.**

# 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 88 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.

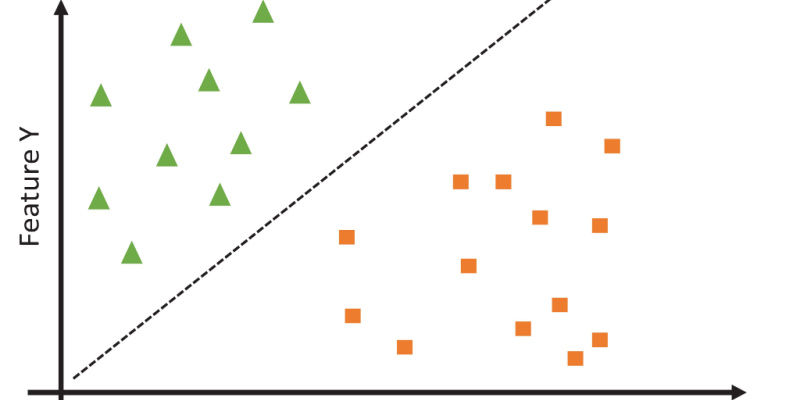
در این مقاله، ما یک بررسی عمیق از رابطه متقابل بین توابع خطا و الگوریتم یادگیری نظارت شده، با استفاده از چارچوب ماشین بردار پشتیبان انجام می‌دهیم. این مطالعه صرفاً بر روی کار طبقه بندی باینری متمرکز شده است.

# ماشین بردار پیشتیبان[[1]](#footnote-1) (SVM)

یادگیری ماشینی نظارت شده[[2]](#footnote-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]](#footnote-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]](#footnote-4) (SRM) است و ریشه در تئوری یادگیری آماری[[5]](#footnote-5) (SLT) دارد و پایه نظری قوی و توانایی تعمیم خوبی برای آن فراهم می‌کند.

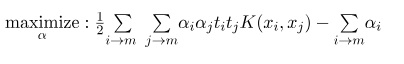
مجموعه آموزشی با زوج مرتب‌های نمونه و برچسب تعریف می‌شود، که در آن بردار نمونه را نشان می‌دهد و نشان دهنده برچسب مربوط به کلاس است. هدف SVM ساخت یک ابر صفحه تصمیم با بایاس و وزن بردار است که با داده‌های آموزشی تخمین زده می‌شود. برای یک نقطه داده آزمایشی ، برچسب کلاس مربوطه به صورت 1 پیش‌بینی می‌شود اگر و در غیر این صورت -1 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 (شکل1). اگر تعداد ویژگی‌های ورودی سه باشد، آنگاه‌هایپرپلن به یک صفحه دو بعدی تبدیل میشود.

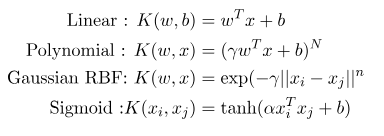


شکل 1: یک نمونه طبقه‌بندی با استفاده از SVM برای فضای ویژگی 2 بعدی

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.



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.



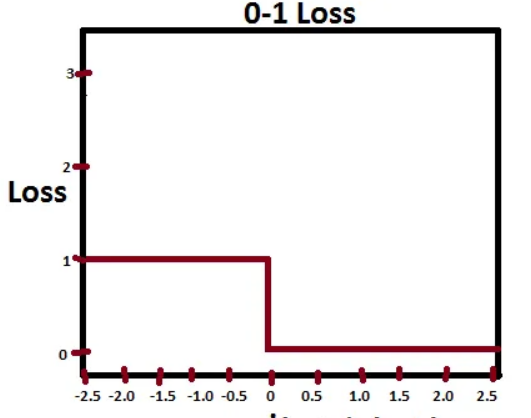
برای به دست آوردن ابر صفحه بهینه، دو حالت را می‌توان در فضای ورودی در نظر گرفت: مجموعه داده‌های آموزشی قابل جداسازی خطی و مجموعه داده‌های آموزشی غیرقابل تفکیک خطی. برای حالت جداسازی خطی مدل زیر را خواهیم داشت:

که باید پارامترهای بهینه w و b را محاسبه کنیم. این مدل به عنوان SVM حاشیه سخت[[6]](#footnote-6) 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:

در این رابطه پارامتر trade-off است و که 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[[7]](#footnote-7) نام برده می‌شود.

# توابع خطا

تابع خطای L(u) جز ضروری ماشین بردار پشتیبان است که استحکام و پراکندگی SVM را کنترل می‌کند. تابع خطای 0-1 به عنوان یک تابع خطای ایده آل تعریف می‌شود که خطای ثابت 1 را به همه نمونه‌های طبقه بندی اشتباه و خطای 0 را به نمونه‌های طبقه بندی شده به درستی اختصاص می‌دهد.

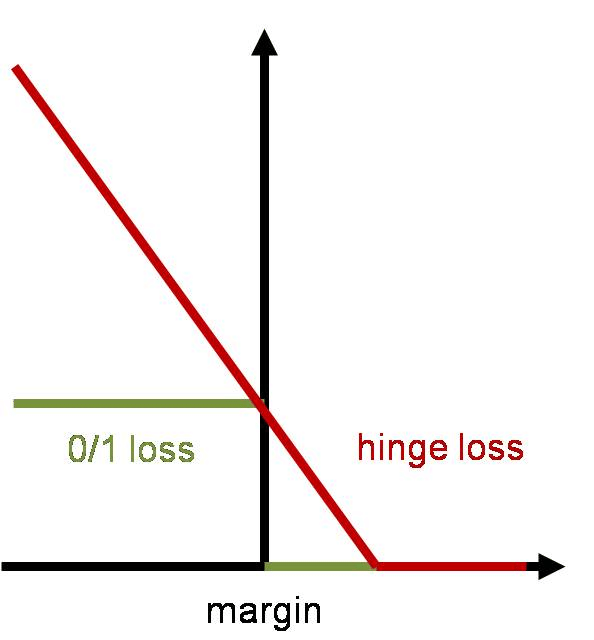


شکل 2: تابع خطای 0-1

حل SVM با تابع خطای 0-1 NP-hard است، زیرا ناپیوسته و غیر محدب است. برای توسعه SVM، کارهای زیادی برای ساخت توابع ضرر جدید انجام شده است تا مدل‌های SVM با حاشیه نرم موثر جدید به دست آید. در این بخش، ما به طور مختصر چند توابع خطای معروف را بررسی خواهیم کرد.

## تابع خطای hinge

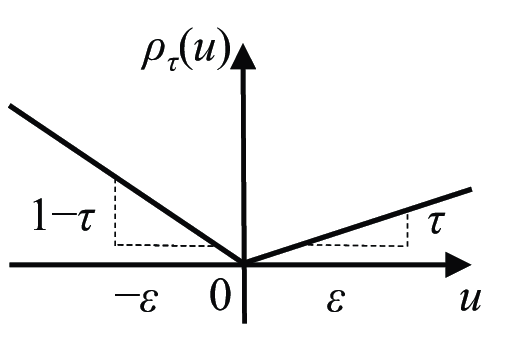
اولین مدل SVM با حاشیه نرم SVM Lhinge-SVM است که از تابع خطای استفاده می‌کند این خطا، خطای ناشی از پیش بینی کلاس اشتباه را برای یک نمونه اندازه گیری می‌کند. پیش‌بینی‌هایی که درست هستند اما مطمئن نیستند را جریمه می‌کند و طبقه‌بندی‌های اشتباه را به شدت جریمه می‌کند. این تابع محدب، غیر صاف و نامحدود است.



شکل 3: مقایسه تابع خطای hinge و تابع خطای 0-1

## 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.

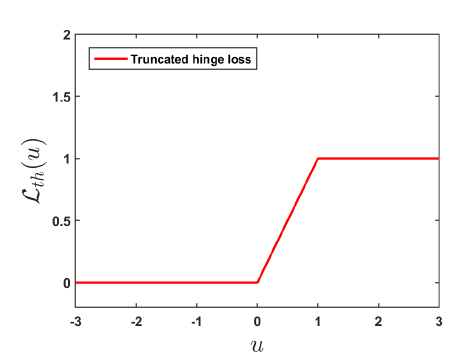


شکل 4: تابع خطای pinball

برای τ = 0، تابع خطای پین بال به تابع hinge تبدیل می‌شود.تابع خطای پین‌بال نیز محدب، غیر صاف و نامحدود است.

## تابع خطای Truncated hinge

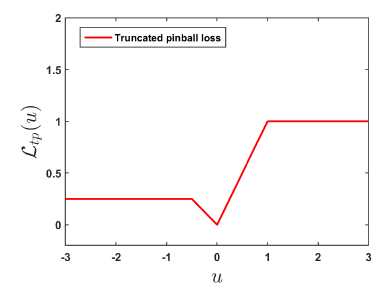
توابع خطای محدب، دارای بهینه منحصر به فرد هستند، استفاده از آنها آسان است و می‌توان آنها را با استفاده از ابزارهای بهینه سازی محدب به طور موثر بهینه کرد، به طوری که طبقه بندی کننده مربوطه را مستعد تحت تأثیر قرار گرفتن یا تسلط بیش از حد عوامل پرت و نامربوط می‌کند. برای بهبود استحکام، توابع مختلف خطای محدود در ادبیات پیشنهاد شده است. به منظور افزایش استحکام hinge-SVM، تابع خطای hinge کوتاه[[8]](#footnote-8) را توسعه دادند. این تابع غیر محدب، غیر صاف و محدود است.



شکل 5: تابع خطای Truncated hinge

## 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.

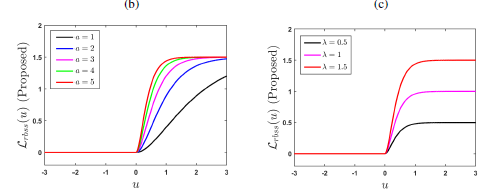


شکل 6: تابع خطای Truncated Pinball

# تابع خطای معرفی‌شده مقاله

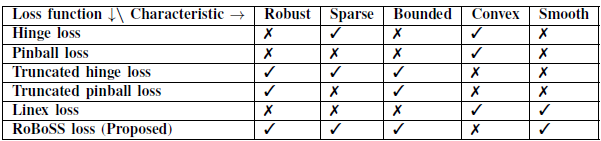
ماهیت غیر محدب و غیر هموار توابع خطای معرفی شده، چالش‌های مهمی‌را از نظر بهینه سازی محاسباتی برای حل مدل‌های SVM مربوطه ایجاد می‌کند. تمرکز اصلی این مقاله ساخت یک تابع جدید قوی، محدود، پراکنده و صاف برای یادگیری تحت نظارت است. برای بهبود استحکام، پراکندگی و صافی تلفات فوق، ما یک تابع خطای جدید به نام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

a > 0، λ > 0 به ترتیب پارامترهای شکل و مرز هستند. تاثیر این دو پارامتر را در شکل‌ 7 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 7: تابع خطای RoBoSS

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

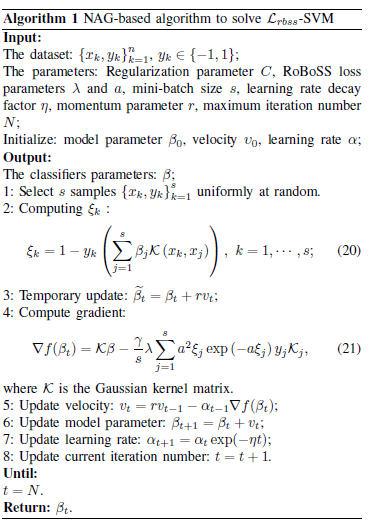


در ادامه تابع خطای RoBoSS پیشنهادی را در SVM ترکیب می‌کنیم مدل SVM جدیدی تعریف می‌شود:

# پیاده‌سازی مقاله

## 1-5- الگوریتم پیاده‌سازی

الگوریتم پیاده‌سازی را در زیر مشاهده می‌نمایید. در این مقاله، از چارچوب مبتنی بر گرادیان شتاب‌دار نستروف[[9]](#footnote-9) (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. ین رویکرد چندین مزیت از جمله کاهش نیازهای محاسباتی و بهبود سرعت را ارائه می‌کند، به‌ویژه هنگام برخورد با مشکلات در مقیاس بزرگ.



برای پیاده‌سازی الگوریتم با استفاده از از تابع RoBoSS\_NAG\_function ([لینک](https://github.com/mtanveer1/RoBoSS)) 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، تابع خطای RoBoss را محاسبه و سپس گرادیان آن را محاسبه 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;

## 2-5- 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 768 نمونه با 8 ویژگی است. از این تعداد، 600 نمونه را به عنوان داده آموزش و مابقی را به عنوان دادگان تست در نظر گرفتیم.

## 3-5- 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 -1 و 1 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;

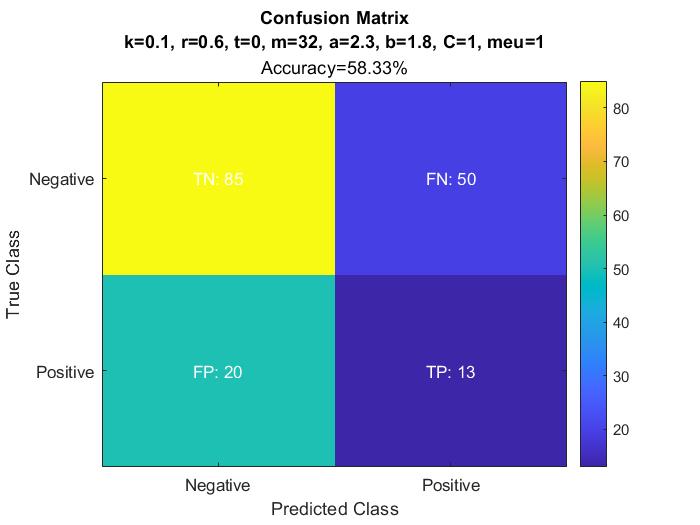
m=2^5; % mini batch size

a=2.3; % a and b are loss parameter

b=1.8;

C=1; % tradeoff parameter

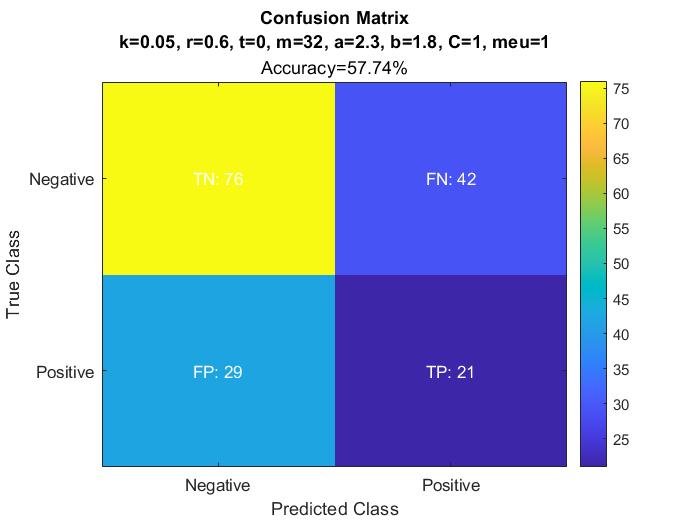
mew=1; % kernel parameter



شکل 8: ماتریس به هم ریختگی و مقدار صحت برای پارامترهای پیش‌فرض

### 5-3-1- تغییر k=0.05

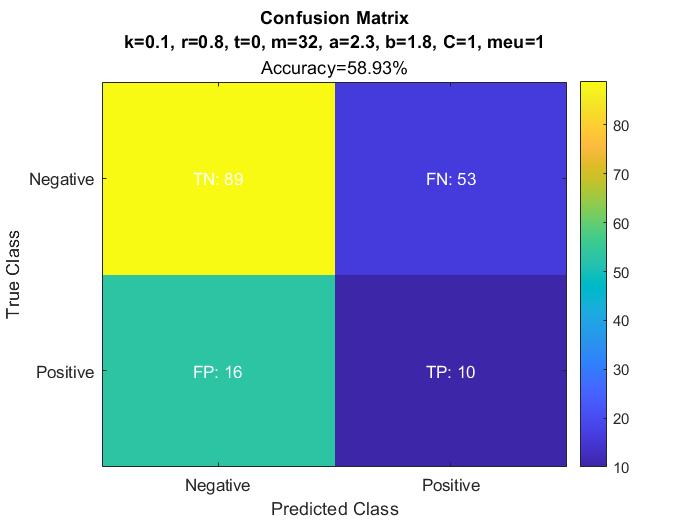
با تغییر این مقدار، در اپوک‌های نهایی مقدار نرخ یادگیری کمتری خواهیم داشت و در نتیجه انتظار داریم کمی‌بیشتر آموزش صورت بگیرد. اما این آموزش بیشتر به معنای بهتر شدن صحت طبقه‌بندی برای داده تست لزوما نخواهد بود، چون حتی ممکن است دچار overfit کردن کدل ما نیز بشود. شکل 9 نتایج این تغییر را نشان می‌دهد.



شکل 9: ماتریس به هم ریختگی و مقدار صحت با تغییر k=0.05

### 2-3-5- تغییر r=0.8

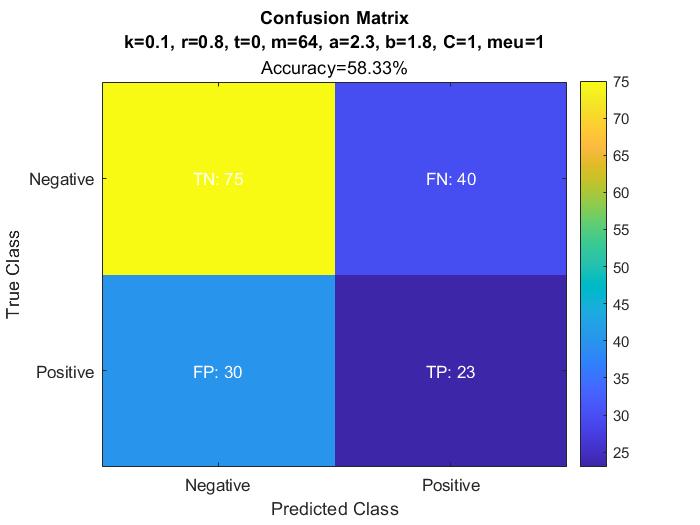
در این بخش با همان مقدار k=0.1 پیش خواهیم رفت. R مقدار ضریب مومنتوم استو هر چه بالاتر باشد، تاثیر دادگان جدید کمتر می‌شود و حتی تاثیر دادگان پرت آموزشی کمتر وارد مدل می‌شود. شکل 10 نتایج این بخش را نشان می‌دهد. این کار توانسته کمی‌صحت را بالاتر ببرد.



شکل 10: ماتریس به هم ریختگی و مقدار صحت با تغییر r=0.8

### 3-3-5-تغییر سایز بچ به 64

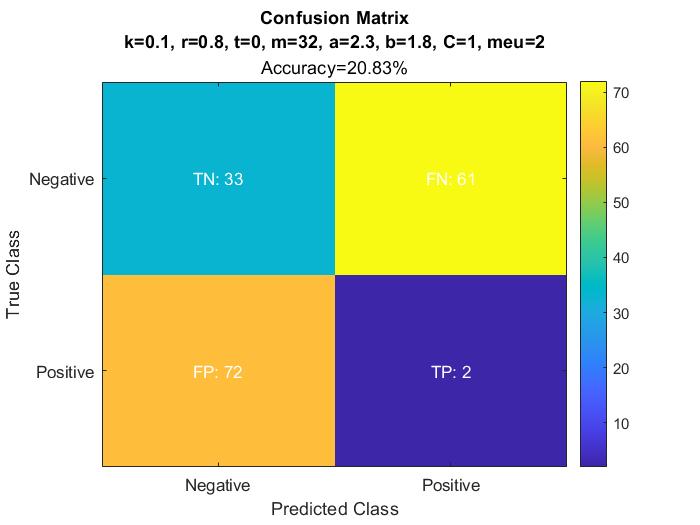
با توجه اطلاعاتی که از قبل داریم، می‌دانیم که بچ سایز در تعداد اپوک آموزشی 1000 نمی‌تواند تاثیر زیادی بر صحت طبقه‌بندی بگذارد و تاثیر خود را بر سرعت بیشتر آموزش با بالاتر رفتن مقدار سایز نشان می‌دهد. با این حال نتایج را برای سایز بچ 64 در شکل 11 مشاهده می‌نمایید. در این بخش و ادامه با مقدار r=0.8 پیش می‌رویم.



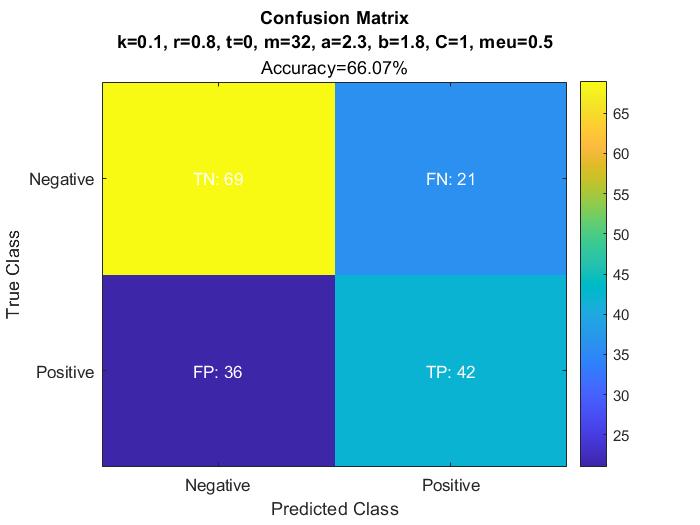
شکل 10: ماتریس به هم ریختگی و مقدار صحت با تغییر بچ سایز به 64

### 4-3-5- تغییر meu

دو شکل زیر نتیجه تغییر این پارامتر از 1 به 2 و 0.5 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.



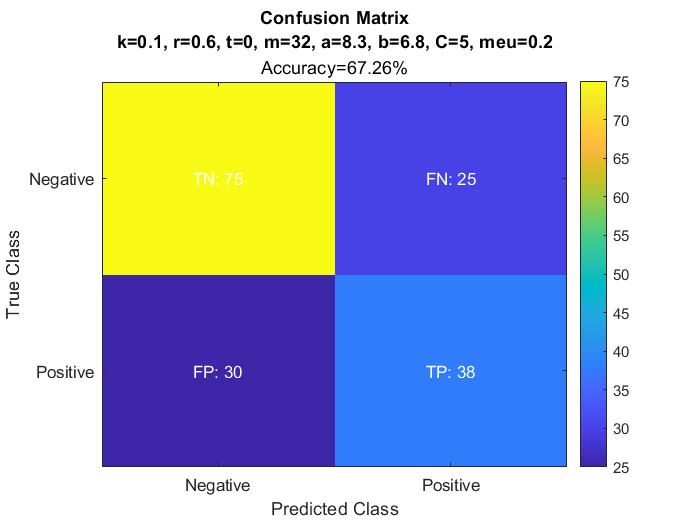
شکل 11: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 2



شکل 12: ماتریس به هم ریختگی و مقدار صحت با تغییر meu = 0.5

### 5-3-5- 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 13 نمایشی از نتیجه به ازای بالابردن مقدار این پارامترها خواهد بود.



شکل 12: 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.

1. Support Vector Machine [↑](#footnote-ref-1)
2. Supervised Machine Learning [↑](#footnote-ref-2)
3. Loss Functions [↑](#footnote-ref-3)
4. Structural Risk Minimization [↑](#footnote-ref-4)
5. Statistical Learning Theory [↑](#footnote-ref-5)
6. Hard-Margin SVM [↑](#footnote-ref-6)
7. Soft-Margin SVM [↑](#footnote-ref-7)
8. Truncated hinge loss function [↑](#footnote-ref-8)
9. Nestrov accelerated gradient (NAG) [↑](#footnote-ref-9)