eman ta zabal zazu

UPV EHU

ZIENTZIA
ETA TEKNOLOGIA
FAKULTATEA

FACULTAD
DE CIENCIA
Y TECNOLOGÍA

50 URTE AÑOS
1968 - 2018

Biba Zientzia!
Ciencia Viva

Ikasketa automatikoaren oinarriak eta algoritmoak: 'Support Vector Machine' eta 'Kernel Methods'.

Gradu Amaierako Lana
Matematikako Gradua

Julen Ercibengoa Calvo

Jon Asier Barcena Petisco
Irakasleak zuzendutako lana

Leioa, 2024ko ekainaren 18a

Aurkibidea

Sarrera

Ikasketa automatikoa bizidunen ikasketa prozesua simulatzen saiatzen den matematikako eta konputazioko arloa da. Horren inguruko lehenengo artikuluak 40ko hamarkadan agertu ziren eta hardware konputazionalean egon diren aurrera pausuekin, gaur egun ordenagailu bat duen edonork ikasketa automatikoko teknika konplexuak erraz aplika ditzake.

Ikasketa automatikoaren atzean dagoen ideia ikusteko, azter dezagun ume baten ikasketa prozesua. Ume batentzat kontzeptu gehienak berriak dira, eta horiek zer diren jakiteko, eskuak erabiltzen ditu. Adibidez, kafe katilu baten testura nola sentitzen den jakiteko, umeak eskuekin katilua hartu nahiko du. Hori egiten duen egunetako batean, baliteke katilua kafe beroaz beteta egotea, eta umeak katilua ukitzen duenean mina sentitzea. Orduan, umeak pentsa lezake "zergatik oraingoan mina hartu dut, katilu bera ukitu dudan beste egun guztietan mina hartu ez badut?". Umeak katiluan arreta jarriko du eta ohartuko da katilutik kea irteten den bakoitzean katilua beroa dagoela. Hau da, umeak gauza berri bat ikasiko du: "katilutik kea irteten bada, katilua ukitzerakoan mina hartuko dut".

Ondorio hori lortzea erraza izan daiteke, azken finean, begiekin ikusten dena kopiatzea besterik ez da: katiluan kea baldin badago umeak mina hartuko du, bestela ez. Aldiz, umeak ez du hori bakarrik ikasten. Umeak lortu berri duen informazioa beste kasu batzuetara estrapolatuko du: lapikotik kea ateratzen denean ere badaki seguruenik lapikoa beroa dagoela eta ez duela ukitu behar. Era berean, platerean duen janariak kea baldin badu, seguruenik janaria beroa dagoela pentsatuko du.

Hau da, umeak egin duena ez da soilik memorizazioa izan, informazioa beste egoera orokorragoetara estrapolatzea lortu du, nahiz eta bere hasierako informazioak lapikoekin eta janariarekin zerikusirik ez izan, kafe katiluekin baizik.

Hori da, besteak beste, ikasketa automatikoaren helburua: kontzeptu baten inguruko informazioa izanik, hau da, lagin bat (datu multzo bat) izanik, laginetik kanpo dauden elementuak ahalik eta hoberen aurresango dituen

funtzio bat lortzea.

Ikasketa automatikoaren bitartez problema oso konplexuak ebatz daitezke, gainera milaka aplikazio desberdin ditu: mugikorren aurpegi-hautematea, spama detektatzea, testuak gaiaren arabera sailkatzea, akzioen prezioaren aurresatea, kotxe automatikoak...

Gu sailkapen automatikoko algoritmoetan zentratuko gara, hau da, lagineko kanpoko elementuak hainbat multzoetan sailkatzea izango dugu helburu. Zehazki, informazioa bi multzo desberdinetan sailkatzen duen algoritmo bat aztertuko dugu sakonki: Euskarri Bektoredun Makina (ingelesez, 'Support Vector Machine' edo SVM). Esan bezala, algoritmo honek klasifikazio dikotomikorako balio du, hala nola, kafe-katilu bat beroa dagoen edo hotza dagoen. Beste adibide zehatzago bat emanez, tomografia baten emaitzak ikusiz gaixoak minbizia ote duen detektatzeko balio duen funtzio bat sor dezake SVM algoritmoak.

Lan honen helburua ikasketa automatikoaren oinarri teorikoak aztertzea eta horiek SVM algoritmoa azaltzeko erabiltzea da. Algoritmoa sakonki ikusi ondoren, algoritmo desberdinen zehaztasuna konparatzeko erabiltzen den datu-base batean aplikatuko dugu: MNIST datu-basean (ikusi [2] artikulua hainbat algoritmo desberdinek datu-base horretan lortutako zehaztasunak ikusteko). Datu-base horrek eskuz idatzitako digituak gordetzen ditu, orduan, datu-base horretan entrenamenduak egin ostean, eskuz idatzitako digituak automatikoki sailkatzea lortuko dugu. Sailkapen horiek nola egiten diren argi ikusteko, Python lengoaia erabiliz programa interaktibo bat idatzi dut eta horren erabilpena ondorengo bideoan ikus daiteke: `https://youtu.be/EEV36u69Ga4`.

Lanaren egitura

Lanak bost kapitulu ditu: 'Ikasketa automatikoaren oinarri teorikoak', 'Galera Minimizazio Erregulatua eta aplikazio teknikak', 'Support Vector Machine', 'Kernel Metodoak eta klasifikazio anizkoitza' eta 'Algoritmoen inplementazioa'.

Lehenengo kapituluan ikasketa automatikoaren oinarriak azalduko ditugu. Lehenik eta behin oinarrizko definizioak emango ditugu ondoren definizio horiek erabiliz 'ikastea' kontzeptua modelizatzeko. Azkenik, 'ikasketa' kontzeptuaren karakterizazio oso erabilgarri bat emango dugu: VC-dimentsioa.

Bigarren kapituluan oinarri teorikoekin jarraituko dugu, izan ere, beste ikasketa metodo bat ikusiko dugu, eta baita ordura arte ikusitako kontzeptuak

aplikatzeko teknika batzuk ere. Lehenik eta behin, 'Galera Minimizazio Erregulatu' ikasketa metodoa aztertuko dugu. Ondoren, 'Gradientearen Beherapena' zer den ikusiko dugu horren hainbat bertsio desberdin emanez, hala nola, 'Gradienteraen Beherapen Estokastikoa'. Azkenik, 'Gradientearen Beherapen Estokastikoa' 'Galera Minimizazio Erregulaturako' nola aplika daitekeen aztertuko dugu.

Hirugarren kapituluan lanaren oinarrizko algoritmoa azalduko dugu: 'Support Vector Machine' (SVM). Gainera, lehenengo kapituluko oinarri teorikoetan lortutako emaitzekin erlazionatuko dugu algoritmoa eta bere efikazia teoria horrekin justifikatuko dugu. Bestalde, bigarren kapituluko 'Galera Minimizazio Erregulatuarekin' lotuko dugu eta horko emaitzak erabiliz algoritmoa aplikatzeko pseudokode bat ikusiko dugu.

Laugarren kapituluari dagokionez, lehenik eta behin algoritmoak duen muga nagusi bat ikusiko dugu. Ondoren, muga hori nola gainditu aztertuko dugu 'kernel metodoak' ikusiz. Hori egiterakoan, SVM algoritmoan 'kernel metodoak' aplikatzen dituen pseudokode bat ikusiko dugu. Azkenik, lehen esan bezala, SVM algoritmoa klasifikazio dikotomikorako prestatua dagoenez, muga hori ere gainditu egingo dugu algoritmoa klasifikazio anizkoitzerako prestatuz.

Bosgarren kapituluan azken lau kapitulu hauetan ikusitakoa praktikan jarriko dugu: algoritmoa digituen sailkapenerako entrenatuko dugu MNIST datu-basea erabiliz. Algoritmoa erabiliz digituen sailkapenerako balio duten hainbat eredu lortuko ditugu eta eredu horiek Pythonen bitartez eginiko aplikazio interaktibo batean sartuko ditugu. Horrela, era bisual batean ikus ahal izango dugu ereduek digituak nola sailkatzen dituzten.

Lanak lau apendize ditu. Lehen apendizean Konbexutasunaren eta Lipschitztasunaren inguruko oinarrizkoak diren hainbat emaitza agertzen dira. Bigarren apendizeari dagokionez, bertan ebatzitako hainbat ariketa daude. Hirugarren apendizean, SVM algoritmoa MNIST datu-basean entrenatzerakoan lortutako hainbat emaitza gehigarri agertzen dira. Azkenik, laugarren apendizean lana egiteko beharrezkoa izan den Pythoneko kodea agertzen da: hala nola SVM algoritmoaren eskuz idatzitako inplementazioa Pythonen edo lanean zehar erabilitako grafikoen kodea. Bestalde, kode guztia nire GitHub orrialdean aurki daiteke: `https://github.com/JulenErcibengoa`.

1. kapitulua

Ikasketa automatikoaren oinarri teorikoak

Sarreran aipatu dugun moduan, ikasketa automatikoak lagin bat erabiliz funtzioak entrenatzen ditu laginean ez dauden datuak ongi aurresateko. Ikus dezagun nola modeliza daitekeen kontzeptu hori. Horretarako, lehenik eta behin oinarrizko definizioak emango ditugu 1.1 atalean. Ondoren, ikastea zer den definituko dugu PAC-ikasgarritasun agnostikoa kontzeptua emanik 1.2 atalean. Hori egin ostean ikasketa kontzeptua orokortuko dugu eta VC-dimentsioa zer den ikusi 1.3 eta 1.4 ataletan, izan ere, VC-dimentsioaren bitartez, ikasketaren karakterizazio garrantzitsu bat emango dugu.

1.1 Hasierako definizioak

1.1.1. definizioa. (Domeinu-multzoa edo domeinua) Izan bedi $\mathcal{X}$ multzo bat edozein non bere elementuak izendatu nahi ditugun, hau da, multzo horren elementu bakoitzari izen bat jarri nahi diogu elementuaren egituraren arabera. Orduan $\mathcal{X}$ multzoari *domeinua* deritzogu.

1.1.2. definizioa. (Izen-multzoa) Aurreko definizioko egoeran, izan bedi $\mathcal{Y}$ multzoa, $\mathcal{X}$ multzoko elementuak izendatzeko erabiliko ditugun izen posibleen multzoa. Orduan, $\mathcal{Y}$ multzoari *izen-multzoa* deritzogu.

1.1.1. oharra. Klasifikazio problemen kasuan oso ohikoa da $\mathcal{Y} = \{-1, 1\}$ hartzea. Adibidez, tomografia batean birikako minbizia detektatzeko probleman, -1 minbizia ez izatea izan liteke eta 1 minbizia izatea.

1.1.3. definizioa. (Lagina) Izan bedi $\mathcal{X} \times \mathcal{Y}$ multzoaren azpimultzo finitua orokorrean S letraz adieraziko duguna, hau da, $S = \{(x_i, y_i)\}_{i=1}^{m}$. Orduan, S multzoa *lagina* dela esango dugu.

1.1.2. oharra. Lagina berez segida bat da, baina guk aztertuko dugun algoritmorako ordenak axola ez duenez, multzo moduan kontsideratuko dugu.

1.1.4. definizioa. (Hipotesia) Izan bitez $\mathcal{X}$ domeinua eta $\mathcal{Y}$ izen-multzoa. Orduan, $h : \mathcal{X} \to \mathcal{Y}$ motako funtzioa *hipotesia* dela esango dugu.

Ohartu hipotesiek domeinuko elementuak izendatu egiten dituztela. Gure algoritmoak irteera moduan itzultzea nahiko genukeen funtzioa, ahalik eta hipotesi 'hoberena' izango da. Orain, 'hoberena' kontzeptua definituko dugu matematikoki. Horretarako, lehenik eta behin hipotesien errorea zer den ikusiko dugu.

1.1.3. oharra. Hemendik aurrera, aurkakoa ez esatekotan, $\mathcal{X}$ multzoa domeinu bat izango da, $\mathcal{Y}$ izen-multzo bat eta $S \subset \mathcal{X} \times \mathcal{Y}$ lagin bat izango da. Bestalde, m letra S laginaren tamaina adierazteko erabiliko dugu.

1.1.4. oharra. Hemendik aurrera, $\mathcal{D}$ $\mathcal{X} \times \mathcal{Y}$ multzoaren gaineko probabilitate banaketa bat izango da. $\mathcal{D}$ banaketatik lortutako elementuak (x, y) motakoak izango dira eta suposatuko dugu y izendapena x elementuaren izendapen egokia dela. Bestalde, $(x, y) \sim \mathcal{D}$ jarriko dugu (x, y) elementua $\mathcal{D}$ banaketatik lortu dugula adierazteko.

1.1.5. definizioa. (Errore erreala) Izan bedi h hipotesia, bere *errore erreala* $\mathcal{D}$ banaketak sortutako zorizko bikote bat gaizki aurresateko probabilitatea da, ondorengo moduan adieraziko duguna:

$$L_{\mathcal{D}}(h) \coloneqq \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[h(x) \neq y].$$

1.1.6. definizioa. (Entrenamendu errorea edo errore enpirikoa) Izan bedi h hipotesia. Horren S lagineko *errore enpirikoa* lagineko gaizki izendatutako elementuen proportzioa da eta $L_S(h)$ letraz adieraziko dugu. Hau da:

$$L_S(h) \coloneqq \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m} \text{ non } [m] = \{1, 2, \ldots, m\}.$$

Errore errealak hipotesien egokitasuna aztertzeko balio dute: geroz eta errore erreal txikiagoa izan, orduan eta hipotesi 'hobea' dugu. Errore erreala $\mathcal{D}$ banaketaren menpe dagoenez, eta banaketa hori ezezaguna denez, praktikan ezin da kalkulatu, ondorioz, errore enpirikoa erabiliko dugu, izan ere, laginaren menpe dago soilik.

1.1.7. definizioa. (Errore Enpirikoaren Minimizazioa, ERM[1]) *ERM metodoa* ikasketa metodo bat da, hau da, h hipotesi bat aukeratzeko modu

[1] Ingelesez, "Empirical Risk Minimization".

bat da, $\mathrm{L}_S(h)$ minimizatzea duena helburu. Ikasketa metodo hori aplikatzen duten algoritmoek itzulitako hipotesiek errore enpiriko minimoa dute.

1.1.5. oharra. 2. kapituluan Gradientearen Beherapena ikusiko dugu: $L_S(h)$ minimizatzeko erabil daitekeen den metodo bat.

1.1.6. oharra. ('Overfitting' fenomenoa) Errore erreala minimizatu ordez errore enpirikoa minimizatzeak arazo bat dakar: gerta daiteke errore erreala altua izatea entrenamendu errorea baxua izanik, hau da, hipotesiak lagina ondoegi aurresaten du eta laginetik kanpoko elementuak ez ditu ongi aurresaten (ikusi [1] 35. orrialdean). Egoera horri 'overfitting' deritzaio eta hori konpontzeko ohiko soluzio bat hipotesi posible kopurua murriztea da. Horretarako, edozein hipotesi hartu beharrean, multzo zehatz bateko hipotesiak hartuko ditugu.

1.1.8. definizioa. (Hipotesi klasea) Izan bedi hipotesiz osaturiko multzo bat, orokorrean $\mathcal{H}$ letraz izendatuko duguna. Orduan $\mathcal{H}$ multzoari *hipotesi klase* deituko diogu.

1.1.9. definizioa. Izan bitez $f : X \to Y$ aplikazio bat eta $S \subset X$ multzo bat. Orduan, f funtzioaren *argumentu minimoa* ondorengoa da:

$$\operatorname*{argmin}_{x \in S} f(x) := \{x \in S : \forall y \in S, f(x) \leq f(y)\}.$$

1.1.7. oharra. Antzeko moduan definitzen da *argumentu maximoa*.

1.1.10. definizioa. (ERM subjektibitate induzituarekin) Izan bitez $\mathcal{H}$ hipotesi klase bat eta S lagina. Orduan, *ERM subjektibitate induzituarekin* ikasketa metodoa erabiltzen duten algoritmoek $\mathcal{H}$ klaseko hipotesi bat aukeratuko dute, h_S, non $h_S \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$. Metodoa $\mathrm{ERM}_{\mathcal{H}}(S)$ moduan izendatzen da.

1.1.8. oharra. Orokorrean h_S letra erabiliko dugu $\mathcal{H}$ hipotesi klase batetik S lagina erabiliz algoritmo batek itzultzen duen hipotesia adierazteko.

1.2 PAC-Ikasgarritasun agnostikoa

Behin oinarrizko definizioak izanik, orain gure helburua $\mathrm{ERM}_{\mathcal{H}}$ metodoak zein hipotesi klaseetarako ez duen 'overfitting' egingo jakitea da. Hori aztertzeko ondorengo definizioa behar dugu:

1.2.1. definizioa. (banaketa suposizioa) S laginak *banaketa suposizioa* beteko du baldin eta S lagineko ale guztiak $\mathcal{D}$ banaketatik modu askean

lortu badira. Hori gertatzen bada honela adieraziko dugu: $S \sim \mathcal{D}^m$ non $m = |S|$.

Behin definizio hau emanda, ikastea zer den modelizatuko dugu. Intuitiboki ikastea ondorengo moduan definituko dugu: lagin tamaina nahiko handi baterako, algoritmo batek hipotesi klase batetik itzultzen duen h_S hipotesiaren errore erreala, hipotesi klase horretan lor daitekeen errore erreal txikiena edo soilik zertxobait handiagoa baldin bada. Hori formalki azaltzen badugu ondorengo definizioa gelditzen zaigu:

1.2.2. definizioa. (PAC-Ikasgarritasun Agnostikoa) Hipotesi klase bat, $\mathcal{H}$, *agnostikoki PAC-ikasgarria* dela diogu ondorengoa betetzen bada: $\exists\ m_{\mathcal{H}} : (0,1)^2 \longrightarrow \mathbb{N}$ eta ikasketa algoritmo bat non $\forall \epsilon, \delta \in (0,1)$, edozein $\mathcal{X} \times \mathcal{Y}$ gaineko $\mathcal{D}$ banaketarako, eta algoritmoa $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ tamainako $S \sim \mathcal{D}^m$ lagin batekin entrenatzen badugu, algoritmoak h_S hipotesi bat itzultzen badu non S laginaren aukeraketaren gainean gutxienez $(1 - \delta)$ probabilitatearekin ondorengoa gertatzen den:

$$\mathrm{L}_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

1.2.1. oharra. Orokorrean, ϵ gure hipotesiaren egokitasuna neurtzen duen parametroa izango da, bestalde δ letraz adieraziko dugu lagin ezegoki bat lortzeko probabilitatea. Orduan, $(1 - \delta)$ gure hipotesiak ϵ zehaztasuna lortzeko konfiantza parametroa izango da.

1.2.2. oharra. PAC-ikasgarritasuna defini daiteke ere, baina gu kasu agnostikoan zentratuko gara lanean zehar.

$m_{\mathcal{H}}$ funtzioak $\mathcal{H}$ klasea ikasteko behar dugun ale kopuru minimoa neurtzen du, kontzeptu horri *lagin konplexutasuna* esango diogu. Ohartu gerta litekeela $m_{\mathcal{H}}$ funtzio desberdin asko izatea PAC-ikasgarritasun agnostikoaren baldintzak betetzen dituztenak, beraz, kontzeptu hau definitzerakoan arazo hori kontuan izan behar dugu.

1.2.3. definizioa. (Lagin konplexutasuna) Aurreko egoeran, $\mathcal{H}$ klasearen *lagin konplexutasuna* $m_{\mathcal{H}}$ funtzio guztien artean lortzen den balio txikiena izango da, hau da, edozein $\epsilon, \delta, m_{\mathcal{H}}$ izanik, PAC-ikasgarritasun agnostikoa bermatzen duen zenbaki oso txikiena izango da lagin konplexutasuna.

1.3 Ikasketa metodo orokorragoak

Orain, aurreko ataletan ikusi ditugun ikasketa metodoak orokortu egingo ditugu, errore ezberdinak definitzeko kontzeptu bat aztertuz.

1.3.1. definizioa. (Galera funtzioa) Izan bitez $\mathcal{H}$ multzoa (hipotesi-klase rola duena, baina edozein multzo izan daiteke) eta Z multzoa. Orduan, edozein funtzio $l : \mathcal{H} \times Z \longrightarrow \mathbb{R}_+$ *galera funtzioa* izango da.

1.3.1. adibidea. Lanean zehar aztertuko ditugun kasuetarako, $Z = \mathcal{X} \times \mathcal{Y}$ izango da. Adibidez, $Z = \mathbb{R}^d \times \{\pm 1\}$ izango da 3, 4 eta 5. kapituluetan.

1.3.1. oharra. Askotan Z multzoari domeinu deituko diogu, nahiz eta multzoko elementuak izendatzea ez izan helburua, Z multzoaren azpimultzo bat izendatzea baizik.

Galera funtzioak $h \in \mathcal{H}$ hipotesi batek $z \in Z$ domeinuko elementu baten gainean egin duen errorea neurtzen du. Galera funtzioak itzulitako balioa 0 izateak hipotesiak domeinuko elementua modu egokian izendatu duela esan nahi du.

1.3.2. definizioa. (Errore erreal orokorra) $h \in \mathcal{H}$ hipotesi baten *errore erreal orokorra* ondorengoa da: l galera funtzio baterako itxarotako balioa Z domeinu gaineko $\mathcal{D}$ probabilitate banaketarekiko. Hau da,

$$L_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[l(h, z)].$$

1.3.3. definizioa. (Entrenamendu errore orokorrak) Izan bitez $S = (z_1, z_2, \ldots, z_m) \in Z^m$ lagina eta l galera funtzioa. $h \in \mathcal{H}$ hipotesiaren *entrenamendu errore orokorra* $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, z_i)$ da.

1.3.4. definizioa. $0-1$ galera funtzioa klasifikazio dikotomikorako erabiltzen da. Kasu honetan $Z = \mathcal{X} \times \{\pm 1\}$ dugu eta honela definitzen da:

$$l^{0-1}(h, (x, y)) = \begin{cases} 0 & , h(x) = y, \\ 1 & , h(x) \neq y. \end{cases} \tag{1.1}$$

1.3.2. oharra. l^{0-1} galera funtzioak sortzen duen errore erreal orokorrak ondorengoa betetzen duela froga daiteke (ikusi [1] 48. orrialdean):

$$L_{\mathcal{D}}^{0-1}(h) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[l^{0-1}(h, z)] = \mathop{\mathbb{P}}_{(x,y) \sim \mathcal{D}}[h(x) \neq y].$$

Hau da, galera funtzioen bitartez 1.1 ataleko definizioak orokortzen dira.

1.3.3. oharra. Batzuetan $(\mathcal{H}, Z, l)$ hirukoteari *ikasketa problema* deritzogu. Hau da, egoera horretan, $\mathcal{H}$ hipotesi-klasea izango da, Z domeinua eta l galera funtzioa.

1.3.5. definizioa. (PAC-Ikasgarritasun agnostikoa galera funtzio orokorrentzako) Izan bedi $(\mathcal{H}, Z, l)$ ikasketa problema. Orduan, $\mathcal{H}$ hipotesi

klasea *agnostikoki PAC-ikasgarria* da Z multzoa eta l galera funtzioarekiko baldin eta $\exists m_{\mathcal{H}} : (0,1)^2 \longrightarrow \mathbb{N}$ eta ikasketa algoritmo bat non $\forall \epsilon, \delta \in (0,1)$ eta edozein Z multzoaren gaineko $\mathcal{D}$ banaketarako, algoritmoa $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ tamainako $S \sim \mathcal{D}^m$ laginean exekutatuz, $h_S \in \mathcal{H}$ hipotesia itzultzen badu non, S laginaren gaineko aukeraketan gutxienez $(1-\delta)$ probabilitatearekin, $L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$ beteko den $L_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[l(h,z)]$ izanik.

1.3.4. oharra. Hemendik aurrera erabiliko dugun ikasketa teoria, galera funtzio orokorren ikasketa teoria izango da.

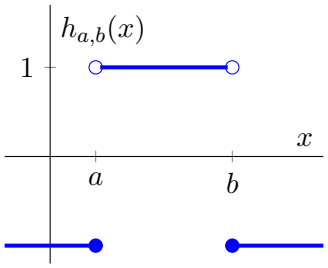
1.4 VC-Dimentsioa

Atal honetan PAC-ikasgarritasun agnostikoaren karakterizazio bat emango dugu. Horretarako, hipotesi klase baten VC-dimentsioa kontzeptua sartuko dugu.

1.4.1. definizioa. Izan bitez $\mathcal{H}$ hipotesi klase bat $\mathcal{X}$ domeinutik $\{-1,1\}$ izen-multzora doazen funtzioez osatua eta $C = \{c_1, c_2, \ldots, c_m\} \subset \mathcal{X}$. Orduan, $\mathcal{H}$ *klasearen murrizketa* C *multzoan*, $\mathcal{H}$ bitartez lor daitezkeen C multzotik $\{-1,1\}$ multzora doazen funtzio multzoa da, $\mathcal{H}_C$ izendaturik. Hau da, $\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m)) : h \in \mathcal{H}\}$ non C multzotik $\{-1,1\}$ multzora doan funtzio bakoitza $\{-1,1\}^{|C|}$ multzoko bektore moduan adierazten dugun.

1.4.2. definizioa. $\mathcal{H}$ hipotesi klase batek $C \subset \mathcal{X}$ multzo finitu bat *partitzen* du baldin eta $\mathcal{H}$ klasearen C gaineko murrizketa C multzotik $\{-1,1\}$ multzora doazen funtzio guztien multzoa bada. Hau da, $|\mathcal{H}_C| = 2^{|C|}$ bada.

1.4.1. adibidea. Izan bitez $\mathcal{X} = \mathbb{R}$ domeinua eta $\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}$ non $h_{a,b} : \mathbb{R} \to \{-1,1\}$ eta $h_{a,b}(x) = (-1)^{\mathbb{1}_{[x \notin (a,b)]}}$ (ikusi 1.1. irudia).



1.1. irudia. $h_{a,b}$ funtzioaren adibide orokor bat.

Har dezagun $C = \{1,2\}$. Ikus dezagun $\mathcal{H}$ hipotesi-klaseak C multzoa partitzen duela. Horretarako, ikusi behar dugu $|\mathcal{H}_C| = 2^{|C|}$ dela, kasu honetan,

$|C| = 2$ denez, $\mathcal{H}_C = \{(-1,-1),(-1,1),(1,-1),(1,1)\}$ dela ikusi behar dugu. Hori erraz lortzen da (a,b) bikoteak era egokian hartzen badira:

(i) $1 < 2 < a < b \Longrightarrow (h_{a,b}(1), h_{a,b}(2)) = (-1,-1)$.

(ii) $1 < a < 2 < b \Longrightarrow (h_{a,b}(1), h_{a,b}(2)) = (-1,1)$.

(iii) $a < 1 < 2 < b \Longrightarrow (h_{a,b}(1), h_{a,b}(2)) = (1,1)$.

(iv) $a < 1 < b < 2 \Longrightarrow (h_{a,b}(1), h_{a,b}(2)) = (1,-1)$.

1.4.3. definizioa. (VC-Dimentsioa) $\mathcal{H}$ hipotesi klase baten *VC-dimentsioa* , VCdim($\mathcal{H}$) moduan izendatua, $\mathcal{H}$ klaseak partitu dezakeen $C \subset \mathcal{X}$ multzo handienaren tamaina da. $\mathcal{H}$ klaseak nahi bezain tamaina handiko multzoak partitu baditzake, $\mathcal{H}$ klasearen VC-dimentsioa infinitua dela diogu.

1.4.2. adibidea. Aurreko adibidearekin jarraituz, jada frogatu dugu VCdim($\mathcal{H}$) ≥ 2 dela. Har dezagun $C = \{c_1, c_2, c_3\}$ multzo bat edozein non $|C| = 3$ eta demagun, inongo orokorpenik galdu gabe, $c_1 < c_2 < c_3$ dela. Orduan, erraz froga daiteke $(1,-1,1)$ ezin dela lortu $\mathcal{H}$ klaseko hipotesien bidez. Ondorioz, VCdim($\mathcal{H}$) < 3 da eta frogatu dugu VCdim($\mathcal{H}$) $= 2$ dela.

PAC-ikasgarritasunaren oinarrizko teorema

Behin VC dimentsioa azaldu dugula, ikasgarritasunaren karakterizazio oso erabilgarri bat emango dugu. Horrek lagin konplexutasuna bornatzeko balioko digu, hau da, problema baten aurrean beharko dugun laginaren tamaina teorikoa jakitea ahalbidetuko digu.

1.4.1. teorema. Izan bitez $\mathcal{H}$ hipotesi klase bat $\mathcal{X}$ domeinutik $\{-1,1\}$ multzora doazen funtzioak dituena eta l^{0-1} galera funtzioa. Orduan, ondorengoak baliokideak dira:

(i) $\mathcal{H}$ klasearen VC dimentsioa finitua da.

(ii) Edozein ERM algoritmo $\mathcal{H}$ klasearentzako egokia izango da PAC-ikasgarritasun agnostikoa bermatzeko.

1.4.2. teorema. (teoremaren bertsio kuantitatiboa) Izan bitez $\mathcal{H}$ hipotesi klase bat $\mathcal{X}$ domeinutik $\{-1,1\}$ multzora doazen funtzioak dituena eta l^{0-1} galera funtzioa. Suposa dezagun VCdim($\mathcal{H}$) $= d < \infty$. Orduan $\mathcal{H}$ agnostikoki PAC-ikasgarria da eta existitzen dira C_1, C_2 konstanteak non lagin konplexutasunak ondorengoa beteko duen:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}.$$

1.4.1. oharra. Aurreko bi teoremen frogak [1] liburuan aurki daitezke. Bestalde, lanean zehar frogatzen ez diren emaitzak, besterik ez zehaztekotan, liburu horretan aurki daitezke.

2. kapitulua

Galera Minimizazio Erregulatua eta aplikazio teknikak

Aurrerago ikusiko dugun moduan, SVM algoritmoaren atzean dagoen teoria Galera Minimizazio Erregulatua da. Ondorioz, algoritmoarekin hasi baino lehen teoria hori azalduko dugu 2.1 atalean. Bestalde, ikasketa automatikoan oso erabilia den minimizazio teknika bat ikusiko dugu 2.2 atalean: Gradientearen Beherapen Estokastikoa. Azkenik, Galera Minimizazio Erregulatua Gradientearen Beherapen Estokastikoaren bitartez aplikatuko dugu 2.3 atalean.

2.1 Galera Minimizazio Erregulatua

Gogora dezagun orain arte ikusi ditugun ikasketa definizioetan $\mathcal{H}$ hipotesi klase bat, Z domeinu bat eta $l : \mathcal{H} \times Z \to \mathbb{R}_+$ galera funtzio bat izan ditugula. Orokorrean, $Z = \mathcal{X} \times \mathcal{Y}$ moduan hartu dugu eta $\mathcal{H}$ klaseko elementuak $\mathcal{X}$ multzotik $\mathcal{Y}$ multzora doazen funtzioak izan dira.

2.1.1. oharra. Hemendik aurrera, kapitulu honetan zehar, $\mathcal{H}$ multzoko funtzioak $\mathbb{R}^d$ multzoko elementuekin identifikatuko ditugu. Orduan, $h \in \mathcal{H}$ hipotesi bakoitza w bektore batekin identifikatuko dugu $w \in \mathbb{R}^d$ izanik eta, notazio abusua eginez, $w \in \mathcal{H} \subset \mathbb{R}^d$ idatziko dugu. Bestalde, lan osoan zehar norma euklidearra eta horren biderkadura eskalarra erabiliko ditugu.

2.1.2. oharra. Identifikazio horrekin jarraituz, ohartu galera funtzioaren kasuan, $w \in \mathbb{R}^d$, $z \in Z$ eta $l : \mathcal{H} \times Z \to \mathbb{R}_+$ galera funtzioa izanik, $l(w, z)$ idaztea ez dela anbiguoa, izan ere $w \in \mathbb{R}^d$ bektorea $h \in \mathcal{H}$ aplikazioarekin identifikatzen dugu. Orduan, askotan $l : \mathbb{R}^d \times Z \to \mathbb{R}_+$ moduan kontsideratuko dugu zuzenean.

2.1.1. definizioa. Izan bitez $(\mathcal{H}, Z, l)$ ikasketa problema eta $R : \mathbb{R}^d \to \mathbb{R}$ funtzioa (funtzio erregulatzaile deituko duguna). Orduan, *Galera Minimizazio Erregulatua* ikasketa metodo bat da $L_S(w) + R(w)$ minimizatzea duena helburu. Hau da, Galera Minimizazio Erregulatu ikasketa metodoaren bitartez lortutako hipotesiak $\underset{w\in\mathcal{H}}{\operatorname{argmin}}(L_S(w) + R(w))$ multzoan egongo dira.

Ikasketa metodo honen atzean dagoen ideia da $R(w)$ funtzioak w hipotesiaren 'konplexutasuna' neurtzen duela.

2.1.3. oharra. Guk ondorengo funtzio erregulatzailea aztertuko dugu: $R(w) = \lambda\|w\|^2$ non $\lambda > 0$. Funtzio erregulatzaile horri *Tikhonoven* erregulatzailea deritzaio.

2.1.1 'Overfitting' fenomenoa Galera Minimizazio Erregulatuarentzako

1. kapituluan aipatu dugun moduan, *overfitting* fenomenoa gertatzen da algoritmo baten bitartez lortzen dugun hipotesiaren errore errealaren eta entrenamendu errorearen arteko desberdintasuna oso handia denean.

2.1.4. oharra. Hemendik aurrera A ikasketa algoritmo bat dugula esaten dugunean, $A : Z^m \to \mathcal{H}$ motako aplikazio bat izango da. Ohartu Z^m multzoko elementuak laginak direla.

Orain Galera Minimizazio Erregulatua ikasketa metodoaren *overfitting* fenomenoa bornatuko dugu, hau da, $A(S) = \underset{w\in\mathcal{H}}{\operatorname{argmin}}(L_S(w) + \lambda\|w\|^2)$ algoritmoaren errore errealaren eta errore enpirikoaren arteko diferentzia bornatuko dugu.

2.1.5. oharra. Konbexutasunaren eta Lipschitztasunaren inguruko beharrezko emaitzak eta definizioak A eranskinean daude. Bestalde, galera funtzioa konbexua edo lipschitziarra dela esaten dugunean, $l(\cdot, z)$ funtzioa konbexua edo lipschitziarra dela diogu z lagineko ale finko bat izanik.

2.1.1. teorema. Izan bedi $(\mathcal{H}, Z, l)$ ikasketa problema eta demagun galera funtzioa konbexua eta ρ-Lipschitziarra dela. Orduan, Galera Minimizazio Erregulatua $\lambda\|w\|^2$ funtzio erregulatzailearekin inplementatutako edozein A algoritmok ondorengoa beteko du:

$$\underset{S\sim\mathcal{D}^m}{\mathbb{E}}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] \leq \frac{2\rho^2}{\lambda m}.$$

2.1.6. oharra. (Ordezko galera funtzioak) Kasu askotan galera funtzioak ez dira konbexuak izango. Hori konpontzeko, konbexua ez den galera funtzioa

konbexua den beste ordezko galera funtzio batekin goitik bornatzen da. Azken ordezko funtzio hori erabiltzen da galera funtzio moduan, jatorrizkoa erabili beharrean.

2.1.2. teorema. Izan bedi $(\mathcal{H}, Z, l)$ ikasketa problema eta demagun galera funtzioa konbexua eta ρ-Lipschitziarra dela. Orduan, Galera Minimizazio Erregulatu metodoak $R(w) = \lambda\|w\|^2$ funtzio erregulatzailearekin ondorengoa betetzen du:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] \leq L_{\mathcal{D}}(w) + \lambda\|w\|^2 + \frac{2\rho^2}{\lambda m}, \forall w \in \mathcal{H}.$$

2.1.7. oharra. Hau da, geroz eta λ handiagoa hartu, orduan eta *overfitting* gutxiago dugu 2.1.1. teoremaren arabera. Baina era berean, geroz eta λ handiagoa hartu, orduan eta errore erreal handiagoa izan dezakegu 2.1.2. teoremaren arabera. Normalean λ egokia aukeratzeko, 5. kapituluan ikusiko dugun moduan, hainbatetan entrenatzen da algoritmoa λ balio desbetarako eta errore enpiriko txikiena lortzen duen balioa hartzen da.

2.1.8. oharra. Ohartu 1. kapituluko definizioetan errore erreala bornatzen genuela zuzenean eta orain errore errealaren itxarotako balioa bornatzen dugula. B.3 ariketan ikusiko dugun moduan, mota horretako borneak PAC ikasgarritasun agnostikoa bermatzeko balio dute.

2.2 Gradientearen Beherapen Estokastikoa (SGD)

Behin SVM algoritmoak behar duen teoria ikusita, teoria hori praktikan nola aplikatu ikusiko dugu. Horretarako Gradientearen Beherapenaren metodoa eta horren hainbat bertsio aztertuko ditugu.

2.2.1 Gradientearen Beherapena

Izan bedi $f : \mathbb{R}^d \to \mathbb{R}$ funtzio konbexu diferentziagarria. Orduan *Gradientearen Beherapen algoritmoak* f funtzioaren minimoa bilatzen du iteratiboki. $w^{(1)}$ hasierako balio bat eta $\eta > 0$ koefizientea (*urratsa* deiturikoa) hartzen ditu datu moduan eta iterazio bakoitzean ondorengo araua aplikatzen du:

$$w^{(t+1)} = w^{(t)} - \eta\nabla f(w^{(t)}). \tag{2.1}$$

Orduan, $T > 0$ iterazio ondoren algoritmoak $\hat{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$ bektorea itzultzen du.

2.2.1. oharra. Azken bektorea itzultzea ohikoa da ere (ikusi [13] liburua), baina guk garatuko dugun teorian bektoreen batez-bestekoa itzuliko du algoritmoak.

Algoritmoaren atzean dagoen ideia da gradienteak malda handieneko noranzkoa azaltzen duela, orduan, gradientearen aurkako noranzkoan urratsak emanez minimora iristea espero da. Ikus dezagun algoritmoaren konbergentzia justifikatzen duen teorema bat.

2.2.1. teorema. Izan bitez f funtzio konbexu ρ-Lipschitziarra eta diferentziagarria, eta $w^* \in \underset{\{w:\|w\|\leq B\}}{\operatorname{argmin}} f(w)$. Gradientearen Beherapenaren algoritmoa f funtzioan T iterazioz exekutatzen badugu $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$ urratsarekin, orduan algoritmoak itzultzen duen $\hat{w}$ bektoreak ondorengoa beteko du:

$$f(\hat{w}) - f(w^*) \leq \frac{B\rho}{\sqrt{T}}.$$

2.2.2 Subgradienteak

Ohartu Gradientearen Beherapena aplikatu ahal izateko funtzioa diferentziagarria izan behar dela gradientea kalkulatu ahal izateko. Orain, arazo hori gainditu egingo dugu. Horretarako, gradientea erabili beharrean, ondoren definituko ditugun subgradienteak erabiliko ditugu iterazio bakoitzean. Gradienteen inguruko A.1.1. proposizioa orokortuta ondorengo lema dugu.

2.2.2. oharra. Ondorengo lau lemen frogak [4] eta [6] liburuetan aurki daitezke.

2.2.2. lema. Izan bedi C multzo ireki eta konbexua. Orduan, $f : C \to \mathbb{R}$ funtzio konbexua da baldin eta soilik baldin edozein $w \in C$, existitzen da $v \in \mathbb{R}^d$ non

$$f(u) \geq f(w) + \langle u - w, v \rangle, \forall u \in C. \tag{2.2}$$

2.2.1. definizioa. Izan bitez $f : C \to \mathbb{R}$ funtzio konbexua eta $w \in C$ bektorearentzako (2.2) ekuazioa betetzen duen $v \in \mathbb{R}^d$ bektorea. Orduan, v bektoreari f funtzioaren w bektoreko *subgradientea* deritzaio. f funtzioaren w bektoreko subgradienteen multzoari *multzo diferentziala* deritzogu eta $\partial f(w)$ izendatuko dugu.

Ondorengo emaitzek subgradienteak kalkulatzeko balioko digute.

2.2.3. lema. Aurreko egoeran egonik, f funtzioa diferentziagarria bada w bektorean, orduan $\partial f(w) = \{\nabla f(w)\}$.

2.2.4. lema. Izan bedi $g(w) = \max_{i\in[r]} g_i(w)$ non $g_i : C \to \mathbb{R}$ funtzio konbexu diferentziagarria den $i = 1, \ldots, r$ guztietarako. Izan bedi $w \in C$ eta demagun $j \in \underset{i\in[r]}{\operatorname{argmax}} g_i(w)$. Orduan $\nabla g_j(w) \in \partial g(w)$.

2.2.5. lema. Izan bedi $f : C \to \mathbb{R}$ funtzio konbexua. Orduan, f funtzioa ρ-Lipschitziarra da C multzoarekiko baldin eta soilik baldin edozein $w \in C$ eta $v \in \partial f(w)$ bektorerako, $\|v\| \leq \rho$. badugu.

Orduan, funtzio konbexu ez diferentziagarrietarako gradientearen beherapen algoritmoa aplikatzeko egin behar den bakarra f funtzioaren $w^{(t)}$ bektoreko subgradientea erabiltzea da, $\nabla f(w^{(t)})$ gradientearen ordez.

2.2.3. oharra. Subgradienteak erabiltzerakoan ere 2.2.1. teorema betetzen da.

2.2.3 Gradientearen Beherapen Estokastikoa (SGD)

Gradientearen Beherapen Estokastikoan (SGD)[1] iterazio bakoitzean zehazki gradientearen (edo subgradientearen) aurkako noranzkoan urratsak eman beharrean (gogoratu (2.1) formula), ausazko bektore bat hartzen da noranzko moduan eta ausazko bektore horren itxarotako balioa gradientea (edo subgradientea) izatea eskatzen da.

Hau da, iterazio bakoitzean $v_t \in \mathbb{R}^d$ bektore bat hartzen da ausaz non $\mathbb{E}[v_t|w^{(t)}] \in \partial f(w^{(t)})$ den. Orduan, hurrengo iterazioa

$$w^{(t+1)} = w^{(t)} - \eta v_t$$

bektorea da.

2.2.6. teorema. Izan bitez $B, \rho > 0$, f funtzio konbexua eta $w^* \in \operatorname{argmin}_{\{w:\|w\|\leq B\}} f(w)$. Demagun Gradientearen Beherapen Estokastikoaren algoritmoa f funtzioan T iterazioz exekutatzen dugula $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$ urratsarekin. Demagun gainera $\|v_t\| \leq \rho$ gertatzen dela $\forall t = 1, \ldots, T$. Orduan algoritmoak itzultzen duen $\hat{w}$ bektoreak ondorengoa beteko du:

$$\mathbb{E}[f(\hat{w})] - f(w^*) \leq \frac{B\rho}{\sqrt{T}}.$$

2.2.4. oharra. SGD algoritmoaren aldaera bat urratsa t aldagaiaren menpe dagoen funtzio bat izatea da. Orduan, η erabili beharrean, η_t erabiltzen da. Adibidez, ondorengo urratsa izan dezakegu: $\eta_t = 1/(\lambda t)$ non $\lambda > 0$.

2.2.4 SGD funtzio konbexu indartsuetarako

Atal honetan funtzio berezi batzuentzako aztertuko dugu SGD algoritmoa: funtzio konbexu λ-indartsuentzako. Ondorengo atalean ikusiko dugun mo-

[1] 'Stochastic Gradient Descent' ingelesez

duan, Galera Minimizazio Erregulatua funtzio konbexu λ-indartsua da, ondorioz, atal hau erabiliko dugu problema hori ebazteko.

2.2.2. definizioa. Izan bedi $f : \mathbb{R}^d \to \mathbb{R}$. Orduan esango dugu f *funtzio konbexu λ-indartsua* dela baldin eta edozein $w, u \in \mathbb{R}^d$ bektorerako eta edozein $\alpha \in (0,1)$ baliorako ondorengoa betetzen bada:

$$f(\alpha w + (1-\alpha)u) \leq \alpha f(w) + (1-\alpha)f(u) - \frac{\lambda}{2}\alpha(1-\alpha)\|w-u\|^2. \quad (2.3)$$

2.2.7. lema.

(i) $f(w) = \lambda\|w\|^2$ funtzio konbexu 2λ-indartsua da.

(ii) f funtzio konbexu λ-indartsua bada eta g konbexua bada, orduan $f+g$ funtzio konbexu λ-indartsua da.

Froga. B.1 ariketan aztertuko dugu. □

Behin definizioa emanik, Gradientearen Beherapen Estokastikoaren algoritmoan sartuko gara. Horretarako, gogoratu urratsari buruzko 2.2.4. oharraz.

1. Algoritmoa: SGD λ-indartsuak diren f funtzioetarako

Datuak: f funtzio λ-indartsua eta T iterazio kopurua.
Emaitza: $\operatorname{argmin}_{w\in\mathbb{R}^d} f(w)$ bektorearen hurbilpen bat.
$w^{(1)} = \vec{0}$
Egin $t = 1, \ldots, T$ **guztietarako**
  Aukeratu v_t ausaz non $\mathbb{E}[v_t|w^{(t)}] \in \partial f(w^{(t)})$ betetzen den.
  $w^{(t+1)} = w^{(t)} - \frac{1}{\lambda t} v_t$
Itzuli: $\hat{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

2.2.8. teorema. Izan bitez $B, \rho > 0$, f funtzio konbexu λ-indartsua eta $w^* \in \operatorname{argmin}_{w\in\mathbb{R}^d} f(w)$. Demagun goiko algoritmoa exekutatzen dugula T iteraziorekin f funtzioan eta demagun $\mathbb{E}[\|v_t\|^2] \leq \rho^2$ dela. Orduan:

$$\mathbb{E}[f(\hat{w})] - f(w^*) \leq \frac{\rho^2}{2\lambda T}(1 + \log(T)).$$

2.3 SGD Galera Minimizazio Erregulatuarentzako

SVM algoritmoaren atzean dagoen teoria Galera Minimizazio Erregulatua denez, ikus dezagun nola ebatzi problema hori SGD algoritmoa erabilita. Hau da, ondorengo problema ebatzi nahi dugu:

$$\min_{w\in\mathbb{R}^d}\left(\frac{\lambda}{2}\|w\|^2 + L_S(w)\right). \quad (2.4)$$

2.3.1. oharra. Tikhonoven Erregulatzailea erabiltzen ari gara, baina λ erabili beharrean $\lambda/2$ erabiliko dugu erosotasunagatik.

2.3.2. oharra. Galera Minimizazio Erregulatuarentzako orain arte ikusi ditugun emaitzetan, galera funtzioa konbexua izan da. Kasu honetan ere horrela mantenduko dugu.

Izan bedi $f(w) = \frac{\lambda}{2}\|w\|^2 + L_S(w)$. Ohartu f funtzio konbexu λ-indartsua dela (ikusi B.2 ariketa), ondorioz, 2.2.4 atalean ikusi dugun algoritmoa aplika dezakegu hemen. Hori egiteko v_t aukeratzeko modu bat lortu behar dugu horien itxarotako balioa f funtzioaren $w^{(t)}$ puntuko subgradientea dela bermatzen duena.

2.3.1. lema. Izan bedi $z \in S$ lagineko ausazko ale bat eta har dezagun $q_t \in \partial l(w^{(t)}, z)$ galera funtzioaren subgradientea $w^{(t)}$ bektorean. Orduan, $\mathbb{E}[\lambda w^{(t)} + q_t]$ balioa f funtzioaren $w^{(t)}$ bektoreko subgradientea da.

Froga. Izan bedi $u \in \mathbb{R}^d$ edozein. Ondorengoa betetzen dela ikusi behar dugu:

$$\frac{\lambda}{2}\|u\|^2 + L_S(u) - \frac{\lambda}{2}\|w^{(t)}\|^2 - L_S(w^{(t)}) \geq \left\langle u - w^{(t)}, \mathbb{E}[\lambda w^{(t)} + q_t | w^{(t)}] \right\rangle.$$

Desberdintzaren eskuineko aldea garatzen badugu:

$$\left\langle u - w^{(t)}, \mathbb{E}[\lambda w^{(t)} + q_t | w^{(t)}] \right\rangle = \langle u - w^{(t)}, \lambda w^{(t)} \rangle + \langle u - w^{(t)}, \mathbb{E}[q_t | w^{(t)}] \rangle.$$

Alde batetik, ohartu $\nabla(\frac{\lambda}{2}\|w^{(t)}\|^2) = \lambda w^{(t)}$ dela. Orduan, 2.2.3. lema eta subgradientearen definizioa erabiliz ondorengo desberdintza dugu:

$$\frac{\lambda}{2}\|u\|^2 - \frac{\lambda}{2}\|w^{(t)}\|^2 \geq \langle u - w^{(t)}, \lambda w^{(t)} \rangle.$$

Ondorioz, ondorengoa frogatzen badugu amaitu egingo genuke:

$$L_S(u) - L_S(w^{(t)}) \geq \langle u - w^{(t)}, \mathbb{E}[q_t | w^{(t)}] \rangle. \tag{2.5}$$

Hori frogatzeko, ohartu $\mathbb{E}[q_t | w^{(t)}] = \frac{1}{m}\sum_{i=1}^{m} q_t^{(i)}$ dela non $q_t^{(i)} \in \partial l(w^{(t)}, z_i)$ den $i = 1, \ldots, m$ guztietarako. Ondorioz:

$$l(u, z_i) - l(w^{(t)}, z_i) \geq \langle u - w^{(t)}, q_t^{(i)} \rangle \tag{2.6}$$

dugu $i = 1, \ldots, m$ guztietarako. (2.6) ekuazioa batzen badugu $i = 1, \ldots, m$ balioetarako, eta desberdintza horri m zatitzen badiogu (kontuan izanik $m > 0$ dela), orduan (2.5) lortzen dugu. □

Aurreko lema aplikatuz, $q_t \in \partial l(w^{(t)}, z)$ harturik, algoritmoak t. iterazioan lortzen duen $t+1$. bektorea honela geldituko litzateke:

$$\begin{aligned}
w^{(t+1)} &= w^{(t)} - \frac{1}{\lambda t}\big(\lambda w^{(t)} + q_t\big) \\
&= \frac{t-1}{t} w^{(t)} - \frac{1}{\lambda t} q_t \\
&= \frac{t-1}{t}\left(\frac{t-2}{t-1} w^{(t-1)} - \frac{1}{\lambda(t-1)} q_{t-1}\right) - \frac{1}{\lambda t} q_t \\
&= \frac{t-2}{t} w^{(t-1)} - \frac{1}{\lambda t}\big(q_{t-1} + q_t\big) \\
&= \frac{t-2}{t}\left(\frac{t-3}{t-2} w^{(t-2)} - \frac{1}{\lambda(t-2)} q_{t-2}\right) - \frac{1}{\lambda t}\big(q_{t-1} + q_t\big) \\
&= \frac{t-3}{t} w^{(t-2)} - \frac{1}{\lambda t}\big(q_{t-2} + q_{t-1} + q_t\big) \\
&= \dots \\
&= -\frac{1}{\lambda t} \sum_{i=1}^{t} q_i.
\end{aligned}$$

Ondorioz, aurreko guztia algoritmo moduan laburtzen badugu:

2. Algoritmoa: SGD Galera Minimizazio Erregulatuarentzako

Datuak: S lagina eta T iterazio kopurua.
Emaitza: 2.4 problema ebaztea galera funtzioa konbexua izanik.
$w^{(1)} = \vec{0}$
Egin $t = 1, \dots, T$ **guztietarako**
  Aukeratu $z \in S$ ausaz era uniformean eta lortu $q_t \in \partial l(w^{(t)}, z)$
  $w^{(t+1)} = -\frac{1}{\lambda t} \sum_{i=1}^{t} q_i$
Itzuli: $\hat{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

Bestalde, demagun galera funtzioa ρ-Lipschitziarra dela, orduan, 2.2.5. lema erabilita, edozein $t = 1, \dots, T$ baliorentzako, $\|q_t\| \leq \rho$ izango da, ondorioz $\|\lambda w^{(t)}\| \leq \rho$ eta beraz, desberdintza triangeluarra erabiliz ondorengoa dugu: $\mathbb{E}[\|\lambda w^{(t)} + q_t\|^2] \leq 4\rho^2$. Orduan, 2.2.8. teorema aplikatuz, T iterazioz algoritmoa exekutatuz ondorengoa lortzen dugu:

$$\mathbb{E}[f(\hat{w})] - f(w^*) \leq \frac{2\rho^2}{\lambda T}(1 + \log(T)).$$

3. kapitulua

Support Vector Machine

Atal honetan Euskarri Bektoredun Makina[1] (SVM) algoritmoa azalduko dugu. Hori jorratu baino lehen, algoritmoaren oinarri izango diren hainbat definizio ikusi behar ditugu, hala nola, espazio erdibikariak 3.1 atalean. Ondoren, algoritmoaren bertsio sinpleena (eta limitatuena) ikusiko dugu: SVM-sendoa 3.2.1 atalean. Aurrerago, limitazio horiek gainditukо ditugu SVM-leuna azalduz 3.2.2 atalean. Azkenik, azken bertsio horren inplementazio bat ikusiko dugu aurreko ataleko Gradiantearen Beherapen Estokastikoa (SGD) eta Galera Minimizazio Erregulatua erabiliz 3.3 atalean.

3.1 Espazio erdibikariak

Has gaitezen SVM algoritmoak erabiltzen duen hipotesi klasea definitzen. Algoritmo honen helburua klasifikazio dikotomikoko problemak ebaztea da, hori egiteko algoritmoak hiperplanoak erabiltzen ditu.

3.1.1. oharra. Hemendik aurrera klasifikazio problemak bakarrik kontsideratuko ditugu. Zehazki, $\mathcal{X} = \mathbb{R}^d$ domeinua eta $\mathcal{Y} = \{\pm 1\}$ izen-multzoa hartuko ditugu.

3.1.1. definizioa. *Funtzio afinen multzoa* (edo *hiperplanoen multzoa*) ondorengo multzoa da:

$$L_d = \{h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

non $h_{w,b}(x) = \langle w, x\rangle + b = \left(\sum_{i=1}^{d} w_i x_i\right) + b$. Era berean, beste notazio batekin: $L_d = \{x \mapsto \langle w, x\rangle + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$.

Behin hiperplanoak definituta, horiekin hipotesi klase bat osatuko dugu. Horretarako, ohartu edozein $w \in \mathbb{R}^d$ eta edozein $b \in \mathbb{R}$ elementuetarako,

[1] 'Support Vector Machine' ingelesez.

$h_{w,b} : \mathbb{R}^d \to \mathbb{R}$ motakoa dela. Hau da, $\mathcal{Y} = \{\pm 1\}$ kasuan gaudenez, guretzako ez dira hipotesi egokiak. Orduan, hiperplanoen bidezko hipotesi klasea $\phi : \mathbb{R} \to \mathcal{Y}$ zeinu funtzioa L_d multzoko elementuekin konposatuz lortuko dugu.

3.1.2. definizioa. (Espazio erdibikarien klasea) *Espazio erdibikariak* klasifikazio dikotomikorako erabiltzen diren hipotesiak dira, $\mathcal{X} = \mathbb{R}^d$ eta $\mathcal{Y} = \{\pm 1\}$ harturik. Orduan, *espazio erdibikarien klasea* ondorengo multzoa da:

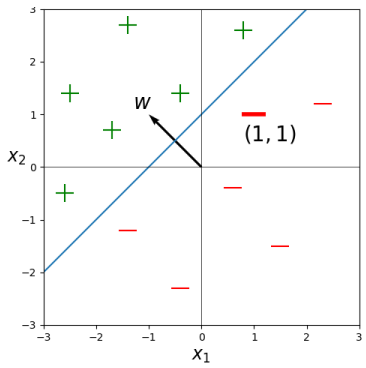
$$HS_d = \text{sign} \circ L_d = \{x \mapsto \text{sign}(h_{w,b}(x)) : h_{w,b}(x) \in L_d\}$$

non

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0, \\ -1, & x < 0. \end{cases}$$

HS_d multzoko elementuei *espazio erdibikari* deituko diegu eta (w, b) moduan izendatuko ditugu, $b = 0$ kasuan *espazio erdibikari homogeneo* deituko diegu. Laburbilduz, HS_d multzoko edozein elementuk (w eta b-ren menpe dagoena), x bektorea ondorengoan transformatzen du: $\text{sign}(\langle w, x \rangle + b)$.

3.1.2. oharra. Askotan *espazio erdibikari* hitza L_d multzoko elementuen kernelek definitzen duten hiperplanoaz hitz egiteko erabiliko dugu, hau da, $\langle w, x \rangle + b = 0$ hiperplanoaz. Orduan, ohartu espazio erdibikari bakoitzak $\mathbb{R}^d$ bi zatitan banatzen duela: alde positiboa eta alde negatiboa.



3.1. irudia. Adibide bat $\mathcal{X} = \mathbb{R}^2$ kasurako. $w = (-1, 1)$ dugu eta $b = -1$, orduan $(1, 1)$ puntuaren kasuan: $\text{sign}(h_{w,b}((1, 1))) = \text{sign}(-1 \cdot 1 + 1 \cdot 1 - 1) = \text{sign}(-1) = -1$.

3.1.3. oharra. (Hiperplanoen homogeneizazioa) Kasu batzuetan komenigarria izan daiteke b terminoa 'kentzea' eta hiperplano homogeneoekin lan egitea. Horretarako, $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$ badugu,

$$w' := (b, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$$

definituko dugu eta edozein $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ bektore

$$x' := (1, x_1, x_2, \ldots, x_d) \in \mathbb{R}^{d+1}$$

bektorean transformatuko dugu. Hori eginez gero,

$$h_{w,b}(x) = \langle w, x \rangle + b = \langle w', x' \rangle = h_{w',0}(x')$$

izango da. Hau da, edozein $\mathbb{R}^d$ espazioko funtzio afin $\mathbb{R}^{d+1}$ espazioko hiperplano homogeneo bezala adieraz daiteke klasifikazio problemak ebazterako orduan. Ondorioz, hemendik aurrera askotan espazio erdibikari homogeneoekin egingo dugu lan zuzenan.

Espazio erdibikarien VC-dimentsioa

Orain espazio erdibikariak agnostikoki PAC-ikasgarriak ote diren jakitea gustatuko litzaiguke. Horretarako, 1.4 atalean aztertutako karakterizazioa erabiliko dugu.

3.1.1. teorema. Espazio erdibikari homogeneoen klasearen VC-dimentsioa $\mathbb{R}^d$ multzoan d da.

Froga. Lehenik eta behin, izan bedi $\{e_i\}_{i=1}^d$ bektore kanonikoen multzoa. Multzo hori espazio erdibikari homogeneoen klaseak partitzen du: hartu $\{y_1, \ldots, y_d\} \subset \{\pm 1\}$ izendapenen multzoa eta definitu $w = (y_1, \ldots, y_d)$. Orduan, ohartu $\text{sign}(\langle w, e_i \rangle) = y_i$ dela edozein $i \in [d]$ baliorako.

Ondoren, izan bitez $x_1, \ldots, x_{d+1} \in \mathbb{R}^d$ bektoreak. Orduan, $\mathbb{R}^d$ espazio bektorialaren dimentsioa $\mathbb{R}$ gorputzaren gainean d denez, existitzen dira $a_1, \ldots, a_{d+1}$ zenbaki errealak, denak aldi berean zero ez izanik, non $\sum_{i=1}^{d+1} a_i x_i = 0$. Izan bitez $I = \{i \in [d+1] : a_i > 0\}$ eta $J = \{j \in [d+1] : a_j < 0\}$ multzoak eta ohartu bi multzo horietako bat ez-hutsa dela. Suposa dezagun bi multzoak ez-hutsak direla, orduan:

$$\sum_{i \in I} a_i x_i = \sum_{j \in J} |a_j| x_j.$$

Demagun orain $\{x_i\}_{i=1}^{d+1}$ multzoa espazio erdibikari homogeneoek partitzen dutela. Orduan existitzen da $w \in \mathbb{R}^d$ non $\langle w, x_i \rangle > 0$ edozein $i \in I$ baliorako eta $\langle w, x_j \rangle < 0$ edozein $j \in J$. Hortik ondorengoa ondorioztatzen dugu:

$$0 < \sum_{i \in I} a_i \langle x_i, w \rangle = \left\langle \sum_{i \in I} a_i x_i, w \right\rangle = \left\langle \sum_{j \in J} |a_j| x_j, w \right\rangle = \sum_{j \in J} |a_j| \langle x_j, w \rangle < 0.$$

Azkenik, demagun J multzo hutsa dela, orduan nahikoa da eskuineko azken desberdintza berdintza batengatik ordezkatzea. Gauza bera gertatzen da I multzo hutsa bada, baina ezkerreko desberdintza aldatu beharko litzateke. □

3.1.2. teorema. Espazio erdibikari ez-homogeneoen (orokorren) klasearen VC-dimentsioa $\mathbb{R}^d$ multzoan $d+1$ da.

Froga. Izan bitez $\vec{0}, e_1, \ldots, e_d \in \mathbb{R}^d$ bektoreak eta $\{y_0, y_1, \ldots, y_d\} \subset \{\pm 1\}$ izendapenen multzo bat. Defini dezagun $b = y_0$ eta $w = (y_1, y_2, \ldots, y_d) - (y_0, \ldots, y_0)$. Orduan, ohartu $\langle w, \vec{0}\rangle + b = y_0$ dela eta edozein $i \in [d]$ baliorako,

$$\langle w, e_i\rangle + b = y_i - y_0 + y_0 = y_i.$$

Bestalde, demagun $x_1, \ldots, x_{d+2} \in \mathbb{R}^d$ bektoreak espazio erdibikari orokorren klaseak partitzen dituela. Orduan, 3.1.3. oharra erabiliz, horrek esan nahi du $\mathbb{R}^{d+1}$ espazioko $d+2$ bektore ditugula espazio erdibikari homogeneo batek partitzen dituena, 3.1.1. teorema erabiliz absurdoa dena. □

3.1.4. oharra. Aurreko bi teoremen ondorioz eta 1.4 ataleko azken teorema erabilita, espazio erdibikariak agnostikoki PAC-ikasgarriak dira edozein ERM algoritmo erabilita laginaren tamaina $\Omega(\frac{d+\log(1/\delta)}{\epsilon^2})$ ordenekoa baldin bada.

3.2 SVM algoritmoa

Jada nahikoa teoria dugu Euskarri Bektoredun Makinak (SVM) azaltzeko. Lehenik, SVM-sendoa azalduko dugu, datuak itxura zehatz bat dutenerako balio duena. Ondoren, baldintzak pixka bat lausotuz, SVM-leuna azalduko dugu, problema orokorragoak ebazteko balioko diguna.

3.2.1. definizioa. (Linealki banangarritasuna) Izan bedi $S = \{(x_i, y_i)\}_{i=1}^m$ lagina non $x_i \in \mathbb{R}^d$ eta $y_i \in \{\pm 1\}$. Orduan S *linealki banangarria* dela diogu espazio erdibikari bat existitzen bada, (w, b), non $y_i = \text{sign}(\langle w, x_i\rangle + b), \forall i \in [m] = 1, 2, \ldots, m$. Baliokideki, $y_i(\langle w, x_i\rangle + b) > 0, \forall i \in [m]$ gertatzen bada.

Lagin baterako baldintza hori betetzen duen edozein espazio erdibikari ERM hipotesi bat izango da, izan ere, horren l^{0-1} errorea 0 izango da. Adibidez, 3.1. irudiko lagina lagin banangarria da eta irudiko espazio erdibikariaren $0-1$ errorea 0 da. Espazio erdibikari guztietatik, zein hartu beharko genuke? Horretarako margina kontzeptua sartuko dugu.

3.2.1. lema. x bektorearen eta (w, b) espazio erdibikariak induzitzen duen hiperplanoarekiko arteko distantzia, $\|w\| = 1$ izanik, $|\langle w, x\rangle + b|$ da.

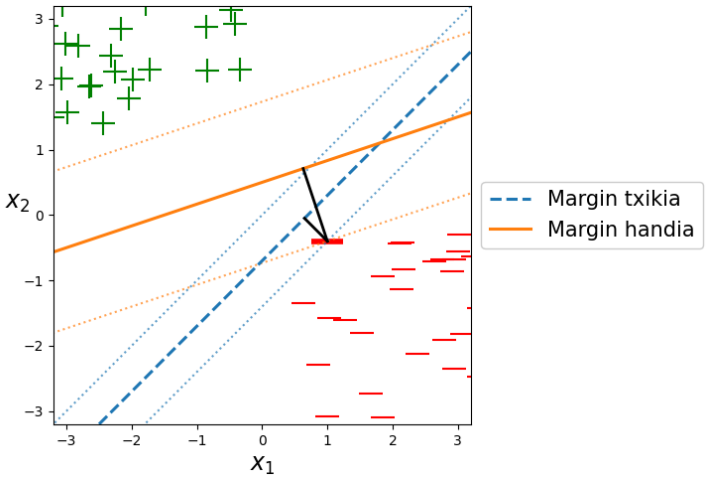
3.2.2. definizioa. (margina) Espazio erdibikari baten lagin batekiko *margina*, lagineko puntuen eta espazio erdibikariaren arteko distantzia minimoa da. Hau da, (w, b) espazio erdibikaria izanik non $\|w\| = 1$ den eta S lagina izanik, margina $\min\{|\langle w, x\rangle + b|, (x, y) \in S\}$ balioa da.

3.2.1. oharra. Erraz froga daiteke (ikusi B.4 ariketa) (w, b) espazio erdibikariak S lagin osoa ongi sailkatzen duenean, margina ondorengoaren baliokidea dela: $\min_{i\in[m]} y_i(\langle w, x_i\rangle + b)$.

3.2.1 SVM-sendoa

Aurreko kontzeptuak definituta, SVM-sendo algoritmoa azalduko dugu. Algoritmo horrek linealki banangarria den lagin bat hartzen du eta margin handieneko espazio erdibikari bat itzultzen du.

Intuitiboki, margin handia duten espazio erdibikariak margin txikia dutenak baino hobeak izango dira, izan ere, margin handia izateak esan nahi du lagineko elementuak apur bat perturbatzen baditugu, hala ere elementu horiek ongi klasifikatuko direla (ikusi 3.2. irudia). Aurrerago ikusiko dugun moduan (3.2.3. teoreman), hau ez da soilik intuizioa izango, marginak errorea goitik bornatuko duelako.



3.2. irudia. Margin desberdineko bi espazio erdibikari eta haien margina irudikatuta, $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^2 \times \{\pm 1\}$ kasurako.

Ondorioz, $\|w\| = 1$ duen espazio erdibikari baten margina $\min_{i\in[m]} |\langle w, x_i\rangle + b|$ denez, SVM-sendo algoritmoak ondorengoa lortzea du helburu:

$$\operatorname*{argmax}_{(w,b):\|w\|=1} \left[\min_{i\in[m]} |\langle w, x_i\rangle + b| \text{ non } y_i(\langle w, x_i\rangle + b) > 0, \forall i \in [m] \right]. \tag{3.1}$$

Bestalde, lagina linealki banangarria den kasuetarako, B.4 ariketan ikusiko dugun moduan, (3.1) ebaztea edo ondorengoa ebaztea baliokidea da:

$$\operatorname*{argmax}_{(w,b):\|w\|=1} \left[\min_{i\in[m]} y_i(\langle w, x_i\rangle + b) \right]. \tag{3.2}$$

Hau azalduta, jada SVM-sendoaren pseudokodea ikus dezakegu.

3. Pseudokodea: SVM-sendoa

Datuak: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ linealki banangarria
Emaitza: (3.2) problemaren emaitza

$$(w_0, b_0) = \underset{(w,b)}{\operatorname{argmin}} \|w\|^2 \text{ non } y_i(\langle w, x_i\rangle + b) \geq 1, \forall i \in [m] \tag{3.3}$$

Itzuli: $\hat{w} = \frac{w_0}{\|w_0\|}, \hat{b} = \frac{b_0}{\|w_0\|}$.

Ohartu pseudokodean minimizatu behar den funtzioa, hau da, (3.3) ekuazioa, eta (3.2) problema desberdinak direla. Hala ere, ondorengo lemaren ondorioz, pseudokodeak (3.2) problema ebazten du:

3.2.2. lema. SVM-sendo algoritmoko irteera (3.2) problemako soluzioa da.

Froga. Izan bedi (w, b) (3.2) problemaren soluzioa. Defini dezagun horren margina ondorengo moduan: $\gamma = \min_{i\in[m]} y_i(\langle w, x_i\rangle + b)$. Orduan, minimoa dela erabiliz, edozein $i \in [m]$ baliorako ondorengoa dugu:

$$y_i\left(\left\langle \frac{w}{\gamma}, x_i \right\rangle + \frac{b}{\gamma}\right) \geq 1.$$

Ondorioz, $(\frac{w}{\gamma}, \frac{b}{\gamma})$ bikoteak (3.3) problemako baldintzak betetzen ditu. Bestalde, izan bitez (3.3) probleman definituriko w_0, $\hat{w}$ eta $\hat{b}$. Orduan, $\|w_0\| \leq \|\frac{w}{\gamma}\| = \frac{1}{\gamma}$ dugu eta ondorioz, edozein $i \in [m]$ baliorako,

$$y_i(\langle \hat{w}, x_i\rangle + \hat{b}) = \frac{1}{\|w_0\|} y_i(\langle w_0, x_i\rangle + b_0) \geq \frac{1}{\|w_0\|} \geq \gamma$$

izango da. $\|\hat{w}\| = 1$ denez, orduan $(\hat{w}, \hat{b})$ bikotea (3.2) problemaren soluzio optimoa izango da. □

SVM-sendoaren lagin konplexutasuna

Lehen azaldu dugun moduan, $\mathbb{R}^d$ domeinuko espazio erdibikarien VC-dimentsioa $d + 1$ da. PAC-ikasketaren oinarrizko teoremaren arabera (1.4.2. teorema), lagin tamaina d/ϵ^2 baino askoz txikiagoa bada, orduan ez da egongo algoritmorik ϵ-zehaztasunarekin espazio erdibikariak ikasteko balioko duena[2]. Hau arazo larria izan daiteke d oso handia bada (adibidez, 5. kapituluan $d = 784$ da eta $\epsilon = 0.05$ harturik, $d/\epsilon^2 = 313600$ izango litzateke). Arazoa konpontzeko baldintza berri bat sartuko dugu: datuak γ

[2]Hau da, ez da egongo algoritmorik ϵ edo txikiagoko errorea duen espazio erdibikaririk itzuliko duena.

marginarekin banangarriak izatea.

Kontzeptu hau sartzerakoan beste arazo bat dugu: demagun $S = \{(x_i, y_i)\}_{i=1}^m$ lagina γ marginarekin banangarria dela, hau da, $\min_{i\in[m]} |\langle w, x_i\rangle + b| \geq \gamma$. Orduan, lagina edozein $\alpha > 0$ zenbakirekin biderkatzerakoan, $S' = \{(\alpha x_i, y_i)\}_{i=1}^m$ lagin berria $\alpha\gamma$ marginarekin banangarria izango da. Ondorioz, marginaren kontzeptua definitzerakoan datuen eskala kontuan hartu behar da.

3.2.3. definizioa. ((γ, ρ) marginarekin banangarria) Izan bedi $\mathbb{R}^d \times \{\pm 1\}$ gaineko $\mathcal{D}$ probabilitate banaketa. Esango dugu $\mathcal{D}$ banaketa (γ, ρ) marginarekin banangarria dela existitzen bada (w, b) espazio erdibikari bat $\|w\| = 1$ izanik non edozein $(x, y) \sim \mathcal{D}$ harturik, $y(\langle w, x\rangle + b) \geq \gamma$ eta $\|x\| \leq \rho$ gertatuko diren.

3.2.3. teorema. Izan bedi $\mathbb{R}^d \times \{\pm 1\}$ gaineko $\mathcal{D}$ probabilitate banaketa (γ, ρ) marginarekin banangarria dena espazio erdibikari homogeneo batekin. Orduan, m lagin tamainaren aukeraketan gutxienez $1 - \delta$ probabilitatearekin, SVM-sendoa algoritmoak itzultzen duen irteeraren l^{0-1} errorea gehienez ondorengoa da:

$$\sqrt{\frac{4(\gamma/\rho)^2}{m}} + \sqrt{\frac{2\log(2\delta)}{m}}.$$

3.2.2 SVM-leuna

Behin SVM-sendoa azalduta, algoritmoa orokortu egingo dugu SVM-leuna aztertuz. Datuak linealki banangarriak izatea baldintza oso sendoa denez, baldintza horiek leundu egingo ditugu: gogoratu SVM-sendoak ondorengo baldintza inposatzen duela: $y_i(\langle w, x_i\rangle + b) \geq 1, \forall i \in [m]$.

Intuitiboki, logikoa da baldintza leuntzeko modua izatea lagineko hainbat elementurentzako baldintza hori apurtzen uztea. Hori modelizatzeko $\xi_1, \ldots, \xi_m$ aldagai ez-negatiboak erabiliko ditugu ondorengo baldintza berria lortuz: $y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \forall i \in [m]$. Orduan, SVM-leun algoritmoak ξ_i balioen batezbestekoa eta w bektorearen norma minimizatzen ditu.

Hori guztia kontuan izanik, ondorengo algoritmoa lortzen dugu:

4. Pseudokodea: SVM-leuna

Datuak: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ lagina eta $\lambda > 0$ parametroa.

$$\min_{w,b,\xi} \left(\lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right) \text{ non } y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in [m] \tag{3.4}$$

Itzuli: w, b.

Ohartu geroz eta λ handiagoa izan orduan eta klasifikazio oker gehiago uzten direla lagineko elementuetan, eta alderantziz.

SVM-leuna eta Galera Minimizazio Erregulatua

Orain SVM-leunaren errorea bornatuko dugu. Horretarako 2.1 atalean ikusi dugunarekin erlaziontuko dugu SVM-leuna. Atal horretan galera funtzio Konbexu eta Lipschitziarretarako garatu dugu Galera Minimizazio Erregulatuko teoria, baina l^{0-1} galera funtzioa ez da konbexua (ikusi B.5 ariketa). Ondorioz, 2.1.6. oharrean esan dugun moduan, $0-1$ galera funtzioa erabili beharrean ondorengo galera funtzioa erabiliko dugu:

3.2.4. definizioa. (banda-galera) Izan bitez $Z = \mathbb{R}^d \times \{\pm 1\}$ eta $\mathcal{H} \subset \mathbb{R}^d \times \mathbb{R}$. Orduan, *banda-galera* funtzioa ondorengo $l^{\text{hinge}} : \mathcal{H} \times Z \to \mathbb{R}$ funtzioa da:

$$l^{\text{hinge}}((w,b),(x,y)) = \max\{0, 1 - y(\langle w, x \rangle + b)\}. \tag{3.5}$$

3.2.2. oharra. Kontuan izan 2. kapituluan azaldu dugun moduan, kasu honetan $\mathcal{H}$ hipotesi-klaseko elementuak $\mathbb{R}^d \times \mathbb{R}$ multzoko elementuekin identifikatzen ditugula, espazio erdibikariak adieraziz.

3.2.3. oharra. Banda-galera funtzioa ordezko galera funtzio egokia dela B.5 ariketan aztertuko dugu.

Orduan, izan bedi ondorengo Galera Minimizazio Erregulatu problema:

$$\min_{w,b} \left[\lambda \|w\|^2 + L_S^{\text{hinge}}((w,b)) \right] \tag{3.6}$$

non $L_S^{\text{hinge}}((w,b)) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}$ den.

3.2.4. lema. (3.4) eta (3.6) minimizazio problemak baliokideak dira.

Froga. Finka dezagun (w,b) bikotea eta kontsidera dezagun (3.6) problemako ξ bektorearen minimizazioa. Finka dezagun $i \in [m]$ bat, orduan ohartu ξ_i balioarentzako esleipen hoberena ondorengoa dela:

$$\xi_i = \begin{cases} 0, & y_i(\langle w, x_i \rangle + b) \geq 1, \\ 1 - y_i(\langle w, x_i \rangle + b), & \text{bestela.} \end{cases}$$

Hau da, $\xi_i = l^{\text{hinge}}((w,b),(x_i,y_i))$ dugu definizioz. □

3.1.3. oharrean azaldu moduan, espazio erdibikari homogeneoak kontuan hartuko ditugu hemendik aurrera. Orduan, ondorengo optimizazio problema dugu:

$$\min_{w\in\mathbb{R}^d}\left(\lambda\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max\{0, 1 - y_i\langle w, x_i\rangle\}\right). \tag{3.7}$$

Orduan, SVM-leunaren errorea bornatzeko 2.1.2. teorema aplikatuko dugu.

3.2.5. korolarioa. Izan bedi $\mathcal{X}\times\{0,1\}$ gaineko $\mathcal{D}$ banaketa, non $\mathcal{X} = \{x\in\mathbb{R}^d : \|x\|\leq\rho\}$. Exekutatu SVM-leuna algoritmoa, (3.7) ekuazioa, $S\sim\mathcal{D}^m$ lagin batean eta izan bedi $A(S)$ algoritmoaren itzulera. Orduan, edozein $w\in\mathbb{R}^d$,

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{0-1}(A(S))] \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{\text{hinge}}(A(S))] \leq L_{\mathcal{D}}^{\text{hinge}}(w) + \lambda\|w\|^2 + \frac{2\rho^2}{\lambda m}.$$

Bestalde, edozein $B>0$, $\lambda = \sqrt{\frac{2\rho^2}{B^2 m}}$ moduan hartzen badugu, orduan:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{0-1}(A(S))] \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{\text{hinge}}(A(S))] \leq \min_{w:\|w\|\leq B} L_{\mathcal{D}}^{\text{hinge}}(w) + \sqrt{\frac{8\rho^2B^2}{m}}.$$

Borneen konparaketa

Ohartu eman berri ditugun SVM-sendo eta SVM-leun algoritmoen borneak dimentsioekiko askeak direla (hau garrantzitsua izango da 4. kapitulurako). Aldiz, espazio erdibikarien VC-dimentsioa d denez $\mathbb{R}^d$ multzoan, ERM bidezko hipotesiaren errorea $\sqrt{d/m}$ bornearekin txikitzen da (gogoratu 1.4.2. teorema). Ikus dezagun adibide bat SVM bornea askoz hobea dena.

3.2.1. adibidea. SVM-leun algoritmoa eta horren bornea kontsideratuko dugu. Ikusiko dugu $\rho^2B^2 \ll d$, beraz, SVM bornea hobea izango da. Demagun testu dokumentu bat kirolen inguruan den ala ez klasifikatu nahi dugula. Dokumentuak bektore moduan adierazteko, demagun d hitz kopuruko hiztegi bat dugula, orduan, dokumentu bat $x\in\{0,1\}^d$ elementu bat izango da non $x_i = 1$ den hiztegiko i-garren hitza dokumentuan baldin badago, bestela $x_i = 0$ izango da. Orduan, edozein dokumentutan dauden hitz desberdin kopurua $\rho^2 = \|x\|^2 = x_1^2 + x_2^2 + \cdots + x_d^2$ non $x_i\in\{0,1\}$ izango da. Azkenik, espazio erdibikari batek 'pisuak' esleitzen dizkie hitzei: logikoa da pentsatzea, problema honentzako, 100 hitzei pisu positibo eta negatiboak esleituz nahikoa izango dela dokumentu bat ongi klasifikatzeko. Beraz, $B^2\leq 100$ jar dezakegu. Orokorrean, $\rho^2B^2\leq 10.000$ izatea ez da

astakeria bat eta bestalde, hiztegiek gutxi-gora-behera 100.000 hitz inguru dituzte. Hau da, $\rho^2 B^2 \ll d$.

3.3 SVM-leunaren inplementazioa SGD algoritmoa erabiliz

Ikusi dugunez, SVM-leuna algoritmoa Galera Minimizazio Erregulatu problema moduan ikus daiteke, ondorioz, 2.3 atalean azaldu dugun teoria erabiliko dugu problema hau inplementatzeko.

Lehenik eta behin, gogora dezagun t iterazioan $t+1$. bektorea lortzeko araua ondorengoa dela: $w^{(t+1)} = -\frac{1}{\lambda t}\sum_{i=1}^{t} q_i$ non q_i galera funtzioaren $w^{(i)}$ bektorean subgradientea den, i. iterazioan ausaz lortutako adibidearen menpe.

Gure galera funtzioa banda-galera denez, horren subgradienteak kalkulatuko ditugu B.5 ariketan. Ondorioz, i. iterazioan $(x, y) \in S$ ausaz uniformeki aukeratuko dugu eta, $q_i = \vec{0}$ izango da $y\langle w^{(i)}, x\rangle \geq 1$ baldin bada, bestela $q_i = -yx$.

Bestalde, $\theta^{(t)} = -\sum_{i=1}^{t} q_i$ moduan definitzen badugu, 2.3 atalean azaldu dugun **2. Algoritmoa** honela gelditzen da (3.7) problema ebazteko:

5. Algoritmoa: SGD SVM-leuna ebazteko

Datuak: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ lagina, $\lambda > 0$ parametroa eta T iterazio kopurua.
Emaitza: (3.7) minimizazio problema ebatzi.
$\theta^{(1)} = \vec{0}$
Egin $t = 1, 2, \ldots, T$ **guztietarako**
  $w^{(t)} = \frac{1}{\lambda t}\theta^{(t)}$
  $i \leftarrow [m]$ multzoko uniformeki ausaz hartutako balio bat
  Baldin $y_i\langle w^{(t)}, x_i\rangle < 1$ **orduan egin**
    $\theta^{(t+1)} \leftarrow \theta^{(t)} + y_i x_i$
  Hori ez betetzekotan, egin
    $\theta^{(t+1)} \leftarrow \theta^{(t)}$

Itzuli: $\hat{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$.

4. kapitulua

Kernel Metodoak eta klasifikazio anizkoitza

Kapitulu honetan SVM algoritmoak dituen muga batzuei aurre egingo diegu. SVM-leunean datuen nolabaiteko linealtasuna behar da ongi funtziona dezan. Hori konpontzeko, 'Kernel trikimailua' azalduko dugu 4.1 eta 4.2 ataletan. Metodo horrek SVM algoritmoak datu ez linealekin lan egitera ahalbidetuko digu eta ondorioz, 4.3 atalean, algoritmoaren bertsio orokorra emango dugu kernelak erabilita.

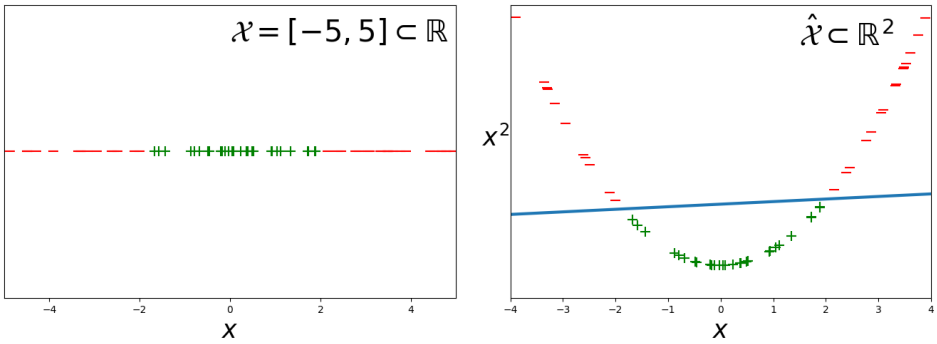
Bestalde, orain arte ikusi dugun SVM algoritmoak ebatz ditzakeen problemak klasifikazio dikotomikoko problemak dira, hau da, bi klase sailkatzeko zereginak. Praktikan, bi klase baino gehiagoko problemak izatea ohikoa denez (adibidez 5. kapituluko probleman), klase anizkoitzeko klasifikazioetara orokortuko dugu SVM algoritmoa 4.4 atalean.

4.1 Txertaketak espazio handiagoetara

Algoritmoak ongi funtziona dezan datuen linealtasuna behar dela ikusteko, demagun ondorengo datu multzoa dugula: $\mathcal{X} = [-5, 5] \subset \mathbb{R}$ ondorengo izendatzeko araua izanik: $y_i = 1, |x| \in [-2, 2]$ bada, bestela $y_i = -1$. Argi dago 4.1. irudiko $S \subset \mathcal{X} \times \mathcal{Y}$ ezkerreko lagina ez dela linealki banangarria, ezin baitugu aurkitu hiperplano bat (puntu bat) datuak ongi bananduko dituena.

Orduan, transforma dezagun gure jatorrizko domeinua dimentsio handiagoko domienu batera, $\hat{\mathcal{X}}$, eta ebatz dezagun problema domeinu berri horretan. Horretarako, defini dezagun $\psi : \mathbb{R} \to \mathbb{R}^2$ honela: $\psi(x) = (x, x^2)$.

4.1. irudiaren ezker aldean agertzen den laginari ψ transformazioa aplikatzen badiogu, eskubiko lagin berria gelditzen zaigu. Oraingoan, lagin hori linealki



4.1. irudia. Lagin ez-lineal baten linealizazioa dimentsio handiagoko domeinu baterako transformazio baten bitartez.

banangarria da: aurki dezakegu hiperplano bat datuak ongi klasifikatuko dituena (adibidez eskubiko irudiko zuzena).

Orduan, $x \in \mathcal{X}$ datu berri bat klasifikatu nahi dugunean, zuzenean klasifikatu beharrean, $\psi(x)$ klasifikatuko dugu $\hat{\mathcal{X}}$ domeinu berrian eta x bektoreari esleituko diogu $\psi(x)$ elementuaren izendapena.

4.1.1. definizioa. Izan bedi $\mathbb{H}$ espazio bektoriala. Orduan, esango dugu $\mathbb{H}$ *Hilberten espazioa* dela biderketa eskalar bat definitua badu eta biderketa horrek induzitzen duen normak $\mathbb{H}$ espazio metriko oso bilakatzen badu.

4.1.2. definizioa. Izan bitez $\mathcal{X}$ domeinua eta $\psi : \mathcal{X} \longrightarrow \mathbb{H}$ aplikazioa non $\mathbb{H}$ Hilberten espazioa den. Orduan $\mathbb{H}$ espazioari *espazio karakteristiko* deritzaio. Bestalde, domeinuko elementuak ψ bitartez $\mathbb{H}$ espaziora eramatea, *txertatzea* dela esango dugu eta ψ *txertaketa funtzioa* dela esango dugu.

Orokorrean ondorengo prozedura egingo dugu:

(i) Aurreko egoeran egonik (definiziokoan) eta $\mathcal{Y}$ izen-multzoa izanik, $S = \{(x_i, y_i)\}_{i=1}^m$ lagina $S' = \{(\psi(x_i), y_i)\}_{i=1}^m$ laginean transformatuko dugu.

(ii) h espazio erdibikaria espazio karakteristikoan entrenatuko dugu S' erabiliz.

(iii) $x \in \mathcal{X}$ domeinuko elementu berri bat izendatu nahi dugunean, $h(\psi(x))$ moduan izendatuko dugu.

Argi dago ikasketaren arrakasta zuzenki lotua dagoela ψ egoki bat aukeratzean. Zorionez, hainbat ψ funtzio desberdin existitzen dira.

Teknika honek bi arazo nagusi ditu: $\mathbb{H}$ espazioaren dimentsioa oso altua bada[1], alde batetik, konputazionalki baliteke konplexuegia izatea $\mathbb{H}$ espazio karakteristikoan lan egitea, bestetik, baliteke lagin handiegi bat behar izatea ikasketa prozesua egiteko. Adibidez, $\mathbb{H} = \mathbb{R}^n$ baldin bada $n \in \mathbb{N}$ oso altua izanik, horren VC-dimentsioa $n+1$ denez, baliteke lagin oso handia behar izatea ikasketa bermatzeko. SVM algoritmoaren kasuan azken arazo hori aurreko atalean konpondu dugu, SVM-ren borneak erabiliz, baina oraindik $\mathbb{H}$ espazioan kalkuluak egitearen arazoa konpondu behar dugu. Hori konpontzeko 'kernel trikimailua' azalduko dugu (ingelesez 'kernel tricks').

4.2 Kernelen trikimailua

Kernelak zer diren definitu aurretik, horien atzean dagoen motibazioa azalduko dugu. Lehenik eta behin, ohartu 3. ataleko SVM minimizazio problema desberdinek ondorengo itxura dutela:

$$\min_w \left[f\Big(\langle w, \psi(x_1)\rangle, \dots, \langle w, \psi(x_m)\rangle \Big) + R\big(\|w\|\big) \right] \tag{4.1}$$

non $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ edozein funtzio den eta $R : \mathbb{R}_+ \to \mathbb{R}$ funtzio monotono ez-beherakorra den, ψ txertaketa funtzio bat izanik.

4.2.1. adibidea.

(i) $\min_w \Big(\frac{1}{m} \sum_{i=1}^m \max\{0, 1 - y_i \langle w, x_i\rangle\} + \lambda \|w\|^2 \Big)$ da SVM-leunaren problema. Orduan, nahikoa da

$$R(a) = \lambda a^2; f(a_1, \dots, a_m) = \frac{1}{m} \sum_{i=1}^m \max\{0, 1 - y_i a_i\}$$

moduan hartzea, $\psi(x_i) = x_i$ izanik kasu honetan.

(ii) $\min_{(w,b)} \Big(\|w\|^2 \Big)$ non $\forall i \in [m], y_i(\langle w, x_i\rangle + b) \geq 1$ da SVM-sendoaren problema. Orduan nahikoa da

$$R(a) = a^2; f(x) = \begin{cases} 0, & \exists b, y_i(a_i + b) \geq 1, \forall i \in [m], \\ \infty, & \text{bestela}, \end{cases}$$

moduan hartzea, $\psi(x_i) = x_i$ izanik.

4.2.1. teorema. Izan bedi ψ $\mathcal{X}$ domeinutik Hilberten espazio batera doan txertaketa funtzio bat. Orduan, existitzen da $\alpha \in \mathbb{R}^m$ non $w = \sum_{i=1}^m \alpha_i \psi(x_i)$ izango den (4.1) problemaren soluzio optimoa.

[1] Dimentsio infinituko espazioetan ere lan egingo baitugu.

Froga. Izan bedi (4.1) problemaren $\hat{w}$ soluzio bat. Orduan, $\hat{w}$ Hilberten espazio bateko elementu bat denez, ondorengo moduan berridatz dezakegu:

$$\hat{w} = \sum_{i=1}^{m} \alpha_i \psi(x_i) + u$$

non $\langle u, \psi(x_i)\rangle = 0$ edozein $i = 1, \dots, m$ baliorako. Definitu $w := \hat{w} - u$ moduan. Orduan, ortogonaltasunarengatik, $\|\hat{w}\|^2 = \|w\|^2 + \|u\|^2$ dugu, ondorioz, $\|w\| \leq \|\hat{w}\|$ dugu. R funtzio ez-beherakorra denez, $R(\|w\|) \leq R(\|\hat{w}\|)$ dugu. Horrez gain, $i = 1, \dots, m$ guztietarako:

$$\langle w, \psi(x_i)\rangle = \langle \hat{w} - u, \psi(x_i)\rangle = \langle \hat{w}, \psi(x_i)\rangle.$$

Orduan:

$$f\Big(\langle w, \psi(x_1)\rangle, \dots, \langle w, \psi(x_m)\rangle\Big) = f\Big(\langle \hat{w}, \psi(x_1)\rangle, \dots, \langle \hat{w}, \psi(x_m)\rangle\Big).$$

Hau da, w ere (4.1) problemaren soluzioa izango da eta $w = \sum_{i=1}^{m} \alpha_i \psi(x_i)$ denez egina dago. □

Aurreko teorema erabilita, (4.1) ekuazioa w bektorearekiko minimizatu beharrean α bektorearekiko minimiza dezakegu. $w = \sum_{i=1}^{m} \alpha_i \psi(x_i)$ baldin badugu, ondorengoak idatz ditzakegu:

(i) $\langle w, \psi(x)\rangle = \Big\langle \sum_{i=1}^{m} \alpha_i \psi(x_i), \psi(x) \Big\rangle = \sum_{i=1}^{m} \alpha_i \Big\langle \psi(x_i), \psi(x) \Big\rangle.$

(ii) $\|w\|^2 = \Big\langle \sum_{i=1}^{m} \alpha_i \psi(x_i), \sum_{j=1}^{m} \alpha_j \psi(x_j) \Big\rangle = \sum_{i,j=1}^{m} \alpha_i \alpha_j \Big\langle \psi(x_i), \psi(x_j) \Big\rangle.$

Ohartu $\Big\langle \psi(x_i), \psi(x_j) \Big\rangle$ terminoa errepikatu egiten dela bi kasuetarako. Definituz $K(x, x') = \Big\langle \psi(x), \psi(x') \Big\rangle$ moduan, (4.1) problema ebatzi beharrean ondorengo problema ebatz dezakegu:

$$\min_{\alpha \in \mathbb{R}^m} \left[f\Big(\sum_{i=1}^{m} \alpha_i K(x_i, x_1), \dots, \sum_{i=1}^{m} \alpha_i K(x_i, x_m) \Big) + R\Big(\sqrt{\sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i, x_j)} \Big) \right]. \tag{4.2}$$

4.2.1. definizioa. (Kernel funtzioak) Izan bedi ψ txertaketa funtzio bat $\mathcal{X}$ domeinutik Hilberten espazio batera. Orduan K *Kernel funtzioa* ondorengo moduan definitzen da: $K(x, x') = \langle \psi(x), \psi(x')\rangle$.

Ohartu problema berri hau ebazteko ez dugula zuzenean espazio karakteristikoetan lan egin behar, jakin behar dugun gauza bakarra espazio horretan biderketa eskalarrak egitea da, hau da, Kernel funtzioa kalkulatzen jakitea.

Bestalde, espazio erdibikari homogeneoen kasuan egonik, behin α minimoa lortu dugula $x \in \mathcal{X}$ elementu berri bat klasifikatzeko $\text{sign}(\langle w, \psi(x)\rangle)$ kalkulatu behar dugu. Baina hori kalkulatzeko oraingoan ere ez dugu w behar:

$$\langle w, \psi(x)\rangle = \sum_{i=1}^{m} \alpha_i \Big\langle \psi(x_i), \psi(x) \Big\rangle = \sum_{i=1}^{m} \alpha_i K(x_i, x). \tag{4.3}$$

Berriro ere, behar dugun bakarra Kernel funtzioa kalkulatzea da. Ikus ditzagun hainbat Kernel funtzio desberdin.

4.2.2. adibidea. (Kernel lineala) *Kernel lineala* honela definitzen da: $K(x, x') = \langle x, x'\rangle$.

4.2.3. adibidea. (Kernel polinomiala) $k \in \mathbb{N}$ mailako *kernel polinomiala* honela definitzen da: $K(x, x') = (1 + \langle x, x'\rangle)^k$.

4.2.4. adibidea. (Kernel gaussiarra) Izan bitez $\mathcal{X} = \mathbb{R}$ domeinua eta ψ txertaketa funtzioa ondorengo moduan definitua: edozein $n \geq 0$ zenbakirako $\psi_n(x) = \frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n$ definiturik, $\psi(x) = (\psi_n(x))_{n=0}^{\infty} \in \ell^2$ izango da. Orduan, txertaketa funtzio horrek definitzen duen kernel funtzioa ondorengoa da:

$$\begin{aligned}\langle \psi(x), \psi(x')\rangle &= \sum_{n=0}^{\infty} \Big(\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n \Big) \Big(\frac{1}{\sqrt{n!}} e^{-\frac{(x')^2}{2}} (x')^n \Big) = e^{-\frac{x^2+(x')^2}{2}} \sum_{n=0}^{\infty} \frac{(xx')^n}{n!} \\ &= e^{-\frac{x^2+(x')^2}{2}} e^{xx'} = e^{-\frac{x^2-2xx'+(x')^2}{2}} = e^{-\frac{|x-x'|^2}{2}}.\end{aligned}$$

Orokorrean, $\mathcal{X} = \mathbb{R}^d$ kasurako, izan bedi $\sigma > 0$. Orduan, σ parametrodun *kernel gaussiarra* honela definitzen da: $K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma}}$.

4.2.1. oharra. Froga daiteke aurreko adibideak benetan kernel funtzioak direla, hau da, existitzen direla ψ txertaketa funtzioak non $K(x, x') = \langle \psi(x), \psi(x')\rangle$ diren (frogak [3] eta [14] liburuetan aurki daitezke).

4.3 SVM-leuna kernelak erabiliz

Aurreko guztia laburbilduz, ondorengo optimizazio problema ebatzi behar da:

$$\min_{w} \left(\frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y\langle w, \psi(x_i)\rangle\} \right). \tag{4.4}$$

Garatu dugun teoriaren zentzua jarraituz, problema hori kernel aplikazioa soilik erabiliz ebatzi nahi dugu, ψ zehazki erabili gabe. Ohartu 3.3 ataleko algoritmoa aplikatzen badugu (4.4) probleman, prozeduratik $w^{(t)}$ bektorea $\{\psi(x_1), \dots, \psi(x_m)\}$ bektore multzoak sortzen duen espazio bektorialean egongo dela. Hau da:

$$w^{(t)} = \sum_{j=1}^{m} \alpha_j^{(t)} \psi(x_j). \tag{4.5}$$

Orduan, $w^{(t)}$ bektoreak gorde beharrean, $\alpha^{(t)}$ bektoreak gordeko ditugu eta horiekin lan egingo dugu. Bestalde, ohartu berriro ere 3.3 ataleko algoritmoko $\theta^{(t)}$ bektoreekin ere gauza bera gertatzen dela, ondorioz:

$$\theta^{(t)} = \sum_{j=1}^{m} \beta_j^{(t)} \psi(x_j). \tag{4.6}$$

Orduan, lehen bezala, $\theta^{(t)}$ bektoreak gorde beharrean, $\beta^{(t)}$ bektoreak gordeko ditugu. Azkenik, izan bedi K kernel funtzio bat. Orduan, α eta β bektoreak ondorengo moduan eguneratuko ditugu:

6. Algoritmoa: SGD kernel bidezko SVM-leuna ebazteko

Datuak: $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ lagina, $\lambda > 0$ parametroa, T iterazio kopurua eta K kernel funtzioa.
Emaitza: (4.4) problema ebatzi.
$\beta^{(1)} \leftarrow 0$
Egin $t = 1, 2, \dots, T$ **guztietarako**
  $\alpha^{(t)} = \frac{1}{\lambda t}\beta^{(t)}$
  $i \leftarrow [m]$ multzoko uniformeki ausaz hartutako balio bat
  Edozein $j \neq i$, definitu $\beta_j^{(t+1)} \leftarrow \beta_j^{(t)}$
  Baldin $y_i \sum_{j=1}^{m} \alpha_j^{(t)} K(x_j, x_i) < 1$ **orduan egin**
    $\beta_i^{(t+1)} \leftarrow \beta_i^{(t)} + y_i$
  Hori ez betetzekotan, egin
    $\beta_i^{(t+1)} \leftarrow \beta_i^{(t)}$
Itzuli: $w = \sum_{j=1}^{m} \hat{\alpha}_j \psi(x_j)$ non $\hat{\alpha} = \frac{1}{T}\sum_{t=1}^{T} \alpha^{(t)}$ den.

4.3.1. lema. Izan bitez $\mathcal{X} = \mathbb{R}^d$ domeinua, $\mathcal{Y} = \{\pm 1\}$ izen-multzoa eta horietatik lortutako $S = \{(x_i, y_i)\}_{i=1}^m$ lagina. Bestalde, izan bitez K kernel funtzioa, ψ izanik horren txertaketa funtzioa eta $\hat{S} = \{(\psi(x_i), y_i)\}_{i=1}^m$ S laginaren txertaketa espazio karakteristikoan. Orduan, 3.3 ataleko algoritmoa $\hat{S}$ laginean exekutatuz itzultzen duen bektorea $\hat{w}$ bada, eta goiko algoritmoa S laginean aplikatzen badugu horrek itzultzen duen bektorea w izanik, $\hat{w} = w$ izango da.

Froga. Ikusiko dugu t iterazio bakoitzerako, (4.6) eta (4.5) ekuazioak betetzen direla non $\theta^{(t)}$ eta $w^{(t)}$ 3.3 ataleko algoritmoan erabiltzen diren bektoreak diren ($\hat{S}$ laginean exekutatuz).

Eraikitzeko moduarengatik, $\alpha^{(t)} = \frac{1}{\lambda t}\beta^{(t)}$ eta $w^{(t)} = \frac{1}{\lambda t}\theta^{(t)}$ dira. Orduan, (4.6) baldintza frogatzen badugu, (4.5) baldintza frogatua gelditzuko da, izan ere,

$$w^{(t)} = \frac{1}{\lambda t}\theta^{(t)} = \frac{1}{\lambda t}\sum_{j=1}^{m}\beta_j^{(t)}\psi(x_j) = \sum_{j=1}^{m}\frac{1}{\lambda t}\beta_j^{(t)}\psi(x_j) = \sum_{j=1}^{m}\alpha_j^{(t)}\psi(x_j)$$

gelditzuko litzateke. Orduan, froga dezagun (4.6) baldintza indukzioz. $t = 1$ denean, berdintza tribiala da. Demagun baldintza $t \geq 1$ balioren baterako betetzen dela, orduan:

$$y_i\langle w^{(t)}, \psi(x_i)\rangle = y_i\left\langle \sum_{j=1}^{m}\alpha_j^{(t)}\psi(x_j), \psi(x_i)\right\rangle = y_i\sum_{j=1}^{m}\alpha_j^{(t)}K(x_j, x_i).$$

Hau da, baldintza baliokidea dute bi algoritmoek eta beraz, θ eta β bektoreen algoritmo bakoitzeko eguneraketa baldintzak kontuan izanik:

$$\theta^{(t+1)} = \theta^{(t)} + y_i\psi(x_i) = \sum_{j=1}^{m}\beta_j^{(t)}\psi(x_j) + y_i\psi(x_i) = \sum_{j=1}^{m}\beta_j^{(t+1)}\psi(x_j).$$

□

4.3.1. oharra. Gogoratu guri $\hat{\alpha}$ bektorea interesatzen zaigula (4.3) formularen bitartez klasifikazio berriak egiteko. Ondorioz, praktikan ψ ezagutzen ez dugunez, w gorde beharrean $\hat{\alpha}$ gordeko dugu.

4.4 SVM klasifikazio anizkoitzerako

Orain arte izen multzoa $\mathcal{Y} = \{\pm 1\}$ izan da, hau da, klasifikazio dikotomikoko problemak aztertu ditugu SVM algoritmoa erabilita. Orain gure helburua $\mathcal{Y} = \{k_1, k_2, \ldots k_r\}$ r kategoria dituen klasifikazio problemak ebaztea da.

4.4.1 Bat-besteen-aurka

Inongo orokorpenik galdu gabe, izan bedi $\mathcal{Y} = \{1, 2, 3, \ldots, r\}$ izenen multzoa. Klasifikazio anizkoitzeko problema 'bat-besteen-aurka' ikuspuntutik ebazteko, r hipotesi desberdin sortuko ditugu.

Izan bedi $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ lagina. Orduan, r lagin berri desberdin sortuko ditugu: $S_1, \ldots, S_r$ non $i = 1, 2, \ldots, r$ guztietarako:

$$S_i = \left\{ \left(x_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}\right), \ldots, \left(x_m, (-1)^{\mathbb{1}_{[y_m \neq i]}}\right) \right\}, (-1)^{\mathbb{1}_{[y_j \neq i]}} = \begin{cases} -1 & , y_j \neq i, \\ 1 & , y_j = i. \end{cases}$$

Hau da, $i = 1, \ldots, r$ klase bakoitzerako S_i lagin berri bat sortuko dugu ondorengo arauarekin: lagineko izenak 1 izango dira domeinuko elementuak i. klasean badaude, beste kasu guztietan -1.

Behin lagin berriak sorturik, $i = 1, \ldots, r$ klase guztietarako SVM bitartez $h_i : \mathcal{X} \to \{\pm 1\}$ hipotesiak sortuko ditugu, orduan, $x \in \mathcal{X}$ elementu berri bat lortzen dugunean, horren izena, $h(x)$, ondorengo moduan lortuko dugu:

$$h(x) \in \underset{i \in [r]}{\operatorname{argmax}}\, h_i(x).$$

Metodo honek arazo bat du: baliteke $x \in \mathcal{X}$ bektore batentzako eta $h_i \neq h_j$ bi hipotesi desberdinentzako, $h_i(x) = h_j(x) = 1$ izatea. Arazo hori konpontzeko, gogora dezagun 3.2.1. lemak dioen moduan, w hiperplano baten eta x bektore baten arteko distantzia $|\langle w, x \rangle|$ dela kasu homogeneoan (gogoratu 3.1.3. oharra), orduan $\langle w_i, x \rangle$ maximoa duen hipotesia hartuko dugu.

4.4.1. oharra. Ez dugu hartuko $|\langle w_i, x \rangle|$ balioen maximoa duen hipotesia, izan ere, $\langle w_i, x \rangle$ negatiboa denean ez zaigu interesatzen, sailkapen ezegokia adierazten duelako.

Orduan, ondorengo moduan ebatziko dugu klase anizkoitzeko problema kasu homogeneoan:

7. Algoritmoa: SVM klasifikazio anizkoitzerako

Datuak: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ lagina.
Egin $i \in \mathcal{Y}$ **guztietarako**
  $S_i \leftarrow \{(x_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}), \ldots, (x_m, (-1)^{\mathbb{1}_{[y_m \neq i]}})\}$
  $w_i \leftarrow \mathrm{SVM}(S_i)$
Itzuli: $h(x) \in \underset{i \in \mathcal{Y}}{\operatorname{argmax}} \langle w_i, x \rangle$.

5. kapitulua

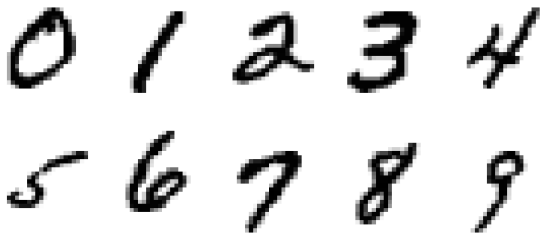
Algoritmoen inplementazioa

Kapitulu honetan aurreko ataletan ikasitakoa praktikan jarriko dugu MNIST datu-basearekin. Lehenik eta behin, datu-basea aztertuko dugu 5.1 atalean. Ondoren, ikasketa automatikoko eredu-aukeraketa teknika bat ikusiko dugu 5.2 atalean. Hori egiterakoan ereduak entrenatuko ditugu: lehenik eskuz programatu dudan algoritmoa 5.4 atalean (kodea D.1 eranskinean dago), eta ondoren Python-ek jada inplementatua duen "SciKit-Learn" eredua [8] 5.5 ataletan. Azkenik, bi eredu horiek konparatuko ditugu 5.6 atalean eta entrenatu ditugun ereduak erabiliz egin dudan aplikazio interaktiboa ikusi 5.7 atalean.

5.1 MNIST datu-basea

MNIST datu-basea eskuz idatzitako digituen bilduma bat da 1994ean sortua [9]. Ikasketa Automatikoko munduan asko erabiltzen da algoritmoen zehaztasuna aztertzeko, izan ere, oso datu-base garbia da: jada balio galduak eta balio arraroak tratatuak daude.

MNIST-en kasuan digitu bakoitza adierazteko $\mathbb{R}^{28\times 28}$ multzoko matrizea



5.1. irudia. MNIST datu-baseko hainbat adibide.

erabiltzen da, matrizeko elementuak pixel bakoitzaren gris-eskala adieraziz. Hau da, pixel bakoitzak 0 eta 255 tarteko balioak gordetzen ditu, 0 zuria izanik eta 255 beltza (ikusi [9] web-orriko azalpenak). Ondoren, errenkadak bata bestearen ondoan jarriz, matrizea bektore moduan adierazten da eta hori da entrenatzeko ematen zaiona algoritmoari. Hau da, $\mathcal{X} = \mathbb{R}^{784}$ da domeinua. Bestalde,

$$\mathcal{Y} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

da izen-multzoa, hau da, 10 klase desberdin ditugu.

MNIST datu-baseak bi multzotan banatutako 70.000 digitu biltzen ditu. Multzo batek, 60.000 digitu dituenak, eredua entrenatzeko balio du, eta beste 10.000 digituek eredu aukeraketa egiteko balio dute. Bi multzo horietan, klase bakoitzak gutxienez %9-ko errepresentazioa du, oso ona dena kontuan izanik 10 klase desberdin ditugula.

5.2 Ereduen aukeraketa eta test-multzoa

Orain arte ikusi ditugun algoritmo gehienetan (SVM-sendoan izan ezik), beti izan dugu guk zehaztu behar dugun parametroren bat. Adibidez, Kernelen bitartezko SVM-leunean, bi parametro ditugu, alde batetik λ parametroa eta bestetik Kernel funtzioak berak duen parametroa (kernel gaussiarrean σ eta kernel polinomialean maila).

Eredu aukeraketa teknika errazena lagin gehigarri bat lortzea da, test-multzo deituko dioguna, eta lagin hori ereduaren errorea estimatzeko erabiltzea. Hau da, gure eredua, h, $\mathcal{D}$ banaketatik lortutako S laginarekin entrenatu badugu, V multzoa banaketa berdinetik lortuko dugu, baina S multzoarekiko askea izango da.

Orduan, eredu baten egokitasuna test-multzo horretan duen errorearen baitan egongo da. Hau da, test-multzoan errorea handia baldin badu, eredua txarra izango dela ondorioztatuko dugu.

5.2.1. oharra. Test-multzoa lortzeko lagin berri bat har dezakegu, baina baliokidea da (eta askotan aukera hobea) jada dugun laginetik zati bat ausaz hartzea test-multzo moduan, eta soberakoa lagin moduan erabiltzea.

5.2.1 Test-multzo bidezko eredu aukeraketa

Demagun r eredu (hipotesi) desberdin ditugula S laginarekin entrenatuta. Horiek SVM aplikatuz lor ditzakegu bakoitzari parametro desberdinak ezarriz. Izenda dezagun $\mathcal{H}$ eredu horiek dituen multzoari: $\mathcal{H} = \{h_1, h_2, \ldots, h_r\}$.

Orduan, hortik eredu hoberena lortzeko, V test-multzo bat hartuko dugu eta $\mathcal{H}$ multzoko hipotesien artean test multzoan errore enpiriko txikiena duena hartuko dugu. Hau da, $\mathrm{ERM}_{\mathcal{H}}(V)$ aplikatuko dugu.

5.3 Datuen tratamendua

Lehenago aipatu dugu MNIST datu-basea jada tratatua dagoen datu-base bat dela. Hala ere, SVM algoritmoa datuen eskalarekiko oso sentibera da [8], izan ere, aurreko kapituluetan ikusi dugun moduan algoritmoaren oinarrian biderkadura eskalarrak daude: $\langle w, \psi(x)\rangle$. Bereziki, 3.1.3. oharrean aipatu dugun moduan, lagina 'homogeneizatu' egin behar dugunez, lehenengo koordenatuaren balioak 1 izango dira beti, aldiz MNIST-ek baditu koordenatu asko 255 balioa dutenak (digituen gorputzak).

Arazo hori konpontzeko teknika asko daude, besteak beste datuen normalizazioa eta estandarizazioa. Guk MNIST-en datu-baseko balioak 0 eta 255 tartean daudela dakigunez, datu guztien balioak 255-ekin zatituko ditugu, datu guztiak 0 eta 1 tartean egoteko.

5.4 Eskuz idatzitako algoritmoa

Atal honetan lan osoan zehar aztertu dugun SVM algoritmoa inplementatuko dugu. Horretarako, SVM-leuna algoritmoa inplementatuko dugu kernelen trikimailua Gradientearen Beherapen Estokastikoaren bitartez aplikatuz (6 eta 7. algoritmoen konbinazioa aplikatuko dugu).

5.4.1. oharra. 4. kapituluko kernel bidezko SVM-leuna algoritmoaren SGD bitarteko eskuz idatzi dudan Pythoneko inplementazioa D.1 eranskinean ikus daiteke. Bestalde, orain ikusiko ditugun emaitzak D.2.1 eta D.2.2 eranskinetan ikus daitezke. Kode osoa nire GitHub orrialdean ikus daiteke ere: `https://github.com/JulenErcibengoa`.

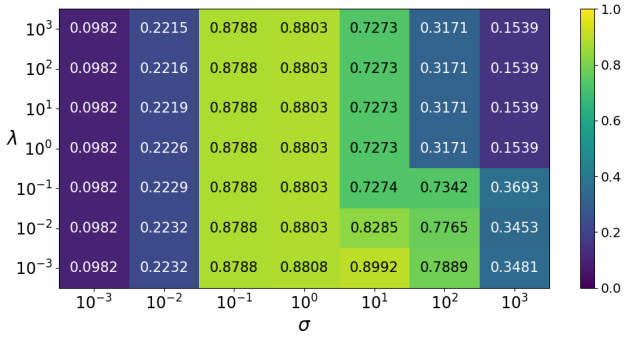
Gogoratu 6. algoritmoak hainbat parametro jasotzen dituela. Alde batetik, λ parametroa jasotzen du eta bestetik, kernelak duen parametroa jasotzen du. Orduan, inplementazio hori erabilita parametro egokienak aurkitzeko hainbat eredu entrenatuko ditugu parametro desberdinak erabiliz. Guk ereduak Kernel polinomialarekin eta gaussiarrarekin entrenatuko ditugu.

Orokorrean, parametro egokienak bilatzeko eskala logaritmiko batetik lortutako balioak hartzea ohikoa da [12] lanean egiten den bezala, besteak beste. Kernel gaussiarrarekin hasiz, ondorengo parametroak erabiliko ditugu: $(\lambda, \sigma) = \{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}^2$, hau da, 49 eredu. Bestalde, eredu

bakoitza entrenatzeko erabiliko dugun iterazio kopurua T = 1000 izango da.

Hau da, D.1 eranskineko kodean ikus daitekeen bezala, digitu bakoitzerako 1000 iterazio erabiliko ditugu, ondorioz, eredu bakoitzak 10000 iterazio erabiliko ditu guztira. Horrek esan nahi du digitu bakoitza entrenatzeko lagineko 1000 ale erabiltzen direla.

5.4.2. oharra. Ereduen errore enpirikoa kalkulatu beharrean, asmatze-proportzioa kalkulatu dugu. Hau da, test-multzotik ongi klasifikatzen dituen aleen proportzioa kalkulatuko dugu. Ohartu bi kontzeptuak baliokideak direla, bata minimizatu nahi dugula eta bestea maximizatu, hurrenez hurren. Aldaketa hau sinbolikoa besterik ez da.



5.2. irudia. Eskuz inplementatutako ereduen asmatze-proportzioa Kernel gaussiarrarekin eta 1000 iterazioko exekuzioekin.

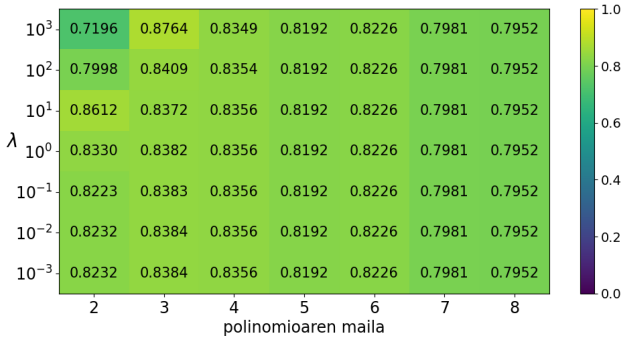
Entrenamendu horiek egiterakoan, 5.2. irudian ikus daitekeen moduan, badirudi geroz eta λ txikiagoa izan, orduan eta eredu hobea dugula. Ondorioz, lambda balio txikiagoetarako entrenatuko dugu algoritmoa, zehazki $\lambda \in \{10^{-10}, 10^{-9}, \ldots, 10^{-4}\}$ balioak aztertuko ditugu, σ lehengoko balio berdinetarako aztertuz.

Entrenamendu hauek amaitzerakoan, 5.2. eta C.1. irudietan ikus daitekeen moduan, ondorioztatzen dugu kernel gaussiarrerako eskuz idatzitako inplementazioan eredu hoberena $(\lambda, \sigma) = (10^{-3}, 10)$ parametroak dituena dela, 0.8992 asmatze proportzioarekin.

5.4.3. oharra. Eredu hauek lortzeko zoria erabiltzen denez, esperimentu hauek errepikagarriak egiteko kodean ausazko hazi bat erabili dut.

Egin berri dugun prozesua errepikatuko dugu kernel polinomialarekin ondorengo parametroekin: $(\lambda, k) = \{10^{-3}, \ldots, 10^{3}\} \times \{2, 3, 4, \ldots, 8\}$ non k polinomioaren maila den. Polinomioaren maila horiek aukeratu ditugu, izan ere, [2] lanean 4. mailako kernel polinomialarekin %1.1 errorea lortzen da,

hau da, 0.989 asmatze proportzioa, ondorioz, asmatze proportzio hori bilatu nahian inguruko mailak aztertu ditugu.



5.3. irudia. Eskuz inplementatutako ereduen asmatze-proportzioa Kernel polinomialarekin eta 1000 iterazioko exekuzioekin.

Entrenamendu hauek amaitu ondoren, kernel polinomiala erabiliz lortu dugun eredu hoberena $(\lambda, k) = (10^3, 3)$ da, 0.8764 asmatze proportzioarekin.

5.5 Pythoneko pakete bateko algoritmoa

Python erabiliz, "Scikit-learn" paketea erabiliko dugu. Paketearen dokumentazioan jartzen duen moduan [8], azaldu dugun teoriarekiko hainbat desberdintasun ditu. Paketeak ebazten duen problema ondorengoa da:

$$\min_w \Big(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\} \Big). \tag{5.1}$$

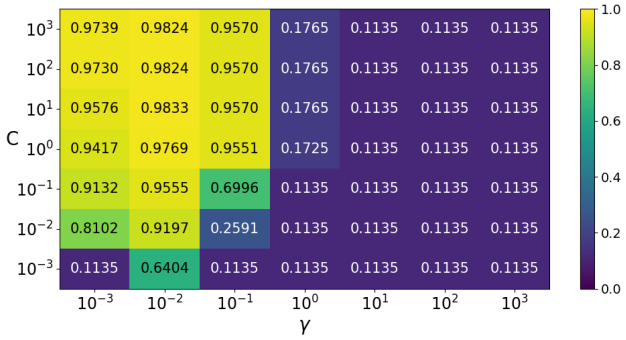
Hau da, gure λ parametroa eta haien C parametroa alderantziz proportzionalak dira, gainera laginaren tamainaren menpe dago proportzio hori. Bestalde, nahiz eta problema baliokidea ebazten duen, modu desberdinean ebazten da. Guk Gradiante Estokastikoaren teknika erabiltzen dugu, baina paketean SVM-en problema duala erabiliz ebazten dute[1] [10] [11].

Bestalde, aipatu beharra dago haien kernel gaussiarra desberdin definitua dagoela: $e^{-\gamma\|x-x'\|^2}$, gurea ondorengo moduan egonik: $e^{\frac{-\|x-x'\|^2}{2\sigma}}$. Hau da, gure σ eta haien γ parametroen arteko erlazioa ondorengoa da: $1/(2\sigma) = \gamma$.

[1] Paketeak badu inplementatua SVM SGD teknika erabilita [7], baina zoritxarrez klase anizkoitzetarako ez dago prestatua.

5.5.1 Inplementazioa

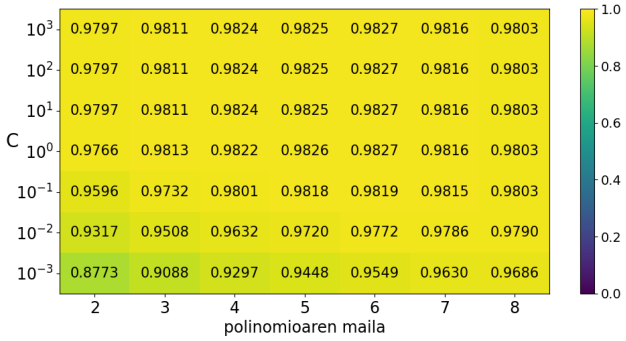
Aipatu bezala, parametroak zehazteko hainbat eredu entrenatuko ditugu eta horietatik hoberena test-multzoarekin aukeratuko dugu. Lehenik eta behin, Kernel gaussiarra erabiliko dugu eta lehen egin dugun moduan ondorengo parametroak erabiliko ditugu: $(C, \gamma) = \{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}^2$.



5.4. irudia. Scikit-Learn paketearen inplementazioko ereduen asmatze-proportzioa Kernel gaussiarrarekin.

Eredu bakoitza entrenatu ondoren 5.4. irudiko emaitzak lortu ditugu. Hau da, Scikit-Learn paketea erabiliz, lortu dugun eredu hoberena $(C, \gamma) = (10, 10^{-2})$ da, 0.9833 asmatze-proportzioarekin.

Era berean, kernel polinomialarekin egin dugu azterketa berdina. Kasu honetan ondorengo parametroak erabili ditugu: $(C, k) = \{10^{-3}, \ldots, 10^{3}\} \times \{2, 3, 4, \ldots, 8\}$ non k polinomioaren maila den.



5.5. irudia. Scikit-Learn paketearen inplementazioko ereduen asmatze-proportzioa Kernel polinomialarekin.

Hau egiterakoan, kernel polinomiala duen eredu hoberena, 5.5. irudian ikus

daitekeen moduan, $(C, k) = (10, 6)$ parametroak dituena da 0.9827 asmatze proportzioarekin.

5.6 Konparaketa eta ondorioak

Lehenik eta behin, nire inplementazioko ereduen eta Scikit-Learn paketeko ereduen arteko desberdintasun nagusia asmatze-proportzioen arteko diferentzian dago. Ohartu nire ereduarekin digitu bakoitza entrenatzeko lagineko 1000 ale erabili ditugula. Aldiz, Scikit-Learn ereduak digitu bakoitza entrenatzeko lagin osoa (60000 ale) erabili du (beste modu batera ebazten duelako problema). Ondorioz, nire inplementazioko eredu hobeak lortzeko, aurreko ataletan lortu ditugun eredu hoberenak 10000 iteraziorekin entrenatuko ditugu. Hori egiterakoan, ondorengo emaitzak lortu ditugu:

(i) Kernel gaussiarrarekin, $(\lambda, \sigma) = (10^{-3}, 10)$ izanik, 10000 iteraziorekin digitu bakoitzerako lortu dugun asmatze-proportzioa 0.9332 da.

(ii) Kernel polinomialarekin, $(\lambda, k) = (10^3, 3)$ izanik, 10000 iteraziorekin digitu bakoitzerako lortu dugun asmatze-proportzioa 0.9232 da.

Nahiz eta 10 aldiz ale gehiago erabili, eredu hauek Scikit-learn paketeko ereduen mailara ez dira iristen:

(i) Kernel gaussiarrarekin, $(C, \gamma) = (10, 10^{-2})$ izanik, Scikit-learn paketearen bitartez lortu dugun asmatze-proportzioa 0.9833 da.

(ii) Kernel polinomialarekin, $(C, k) = (10, 6)$ izanik, Scikit-learn paketearen bitartez lortu dugun asmatze-proportzioa 0.9827 da.

Hori garbi ikus daiteke C.2 eranskinean, izan ere, bertan 4 eredu desberdin hauen konfusio-matrizeak ikus daitezke, gaizki klasifikatutako hainbat digiturekin batera, eta argi ikusten da Scikit-learn paketeko ereduek gaizki klasifikatutako digituak klasifikatzen zailagoak direla (ez direlako garbi marraztutako digituak).

Bestalde, lehen aipatu dugun λ eta C parametroen arteko alderantzizko erlazioa enpirikoki ikusi dugu. Ohartu Kernel gaussiarraren kasuan, λ txikitzen dugun heinean nire ereduen asmatze-proportzioa handitzen dela, aldiz C txikitzen dugun heinean Scikit-learn ereduen asmatze-proportzioa txikitzen dela.

Horrez gain, kernel gaussiarraren kasuan, gogora dezagun σ eta γ parametroen arteko erlazioa ondorengoa dela: $1/(2\sigma) = \gamma$. Hau da, alderantziz proportzionalak dira. Berriro ere, erlazio hau enpirikoki ikusi dugu: σ parametroaren balio hoberena 10 da γ parametroaren balio hoberena 10^{-2} izanik.

Azkenik, ohartu kernel polinomialaren kasuan, orokorrean asmatze-proportzio nahiko onak lortzen direla edozein parametro erabiliz bai nire inplementazioan eta baita Scikit-learn paketearen inplementazioan ere.

5.7 Aplikazio Interaktiboa

Lana borobiltzeko, SVM algoritmoak duen indarra intuitiboki ikusteko Pythonen aplikazio bat sortu dut. Besteak beste, aplikazioak digituak marrazten uzten du eta, kapitulu honetan zehar entrenatu ditugun eredu hoberenak erabiliz, marraztutako digituak aurresaten ditu.

Aplikazioaren bitartez, bisualki ereduen zehaztasuna ikus daiteke, izan ere, aplikazioak nire inplementazioko eredu baten edo Scikit-learn paketeko eredu baten artean aukeratzen uzten du. Orduan, baliteke digitu bat Scikit-learn paketeko eredu batek ongi klasifikatzea baina nire inplementazioko eredu batek gaizki klasifikatzea (asmatze-proportzio okerragoa dutelako).

Bestalde, aplikazioak $\mathcal{X} = \mathbb{R}^2$ eta $\mathcal{Y} = \{\pm 1\}$ kasurako lagin sortzaile bat du. Horren bitartez kernel desberdinak eta parametro desberdinak konpara daitezke modu erraz batean.

Aplikazioaren funtzionamendua azaltzen duen bideoa ondorengo linkean ikus daiteke `https://youtu.be/EEV36u69Ga4`.

A. eranskina

Konbexutasuna eta Lipschitztasuna

Eranskin honetan Konbexutasunari eta Lipschitztasunari buruzko beharrezkoak ditugun hainbat kontzeptu ikusiko ditugu. Eranskin honetako emaitzak analisi konbexuko liburuetan lor daitezke, hala nola, [4] edo [6] liburuetan.

A.1 Konbexutasuna

A.1.1. definizioa. (multzo konbexua) Izan bedi $\mathbb{R}^d$ espazio bektorialaren C azpimultzo bat. Orduan C *multzo konbexua* dela diogu baldin eta edozein $u, v \in C$ eta $\alpha \in [0,1]$ hartuta, $\alpha u + (1-\alpha)v \in C$ gertatzen bada.

A.1.2. definizioa. (funtzio konbexua) Izan bedi C multzo konbexua. Orduan, $f : C \to \mathbb{R}$ *funtzio konbexua* dela diogu baldin eta edozein $u, v \in C$ eta $\alpha \in [0,1]$ harturik ondorengoa betetzen bada:

$$f(\alpha u + (1-\alpha)v) \leq \alpha f(u) + (1-\alpha) f(v).$$

A.1.1. proposizioa. Izan bedi $f : C \to \mathbb{R}$ funtzio konbexua eta deribagarria eta izan bedi $w \in C$. Orduan,

$$f(u) \geq f(w) + \langle u - w, \nabla f(w) \rangle, \forall u \in C \tag{A.1}$$

gertatzen da non $\nabla f(w) = \left(\frac{\partial f(w)}{\partial w_1}, \ldots, \frac{\partial f(w)}{\partial w_d}\right)$ den.

A.1.2. lema. Izan bedi $f : \mathbb{R} \to \mathbb{R}$ bi aldiz deribagarria den funtzioa. Orduan, ondorengoak baliokideak dira:

(i) f konbexua da.

(ii) f funtzioaren deribatua monotonoki ez-beherakorra da.

(iii) f funtzioaren bigarren deribatua ez-negatiboa da.

A.1.3. lema. Izan bitez $g : \mathbb{R} \to \mathbb{R}$, $y \in \mathbb{R}$ eta $x \in \mathbb{R}^d$. Bestalde, izan bedi $f : \mathbb{R}^d \to \mathbb{R}$ non $f(w) = g(\langle w, x\rangle + y)$. Orduan, g konbexua bada f konbexua izango da.

A.1.4. lema. Izan bitez $f_i : \mathbb{R}^d \to \mathbb{R}$ funtzio konbexuak $i = 1, \ldots, r$ balioetarako. Orduan, ondorengo funtzioak ere konbexuak dira:

(i) $g(x) = \max_{i \in [r]} f_i(x)$.

(ii) $\sum_{i=1}^{r} w_i f_i(x)$ non $w_i \geq 0, \forall i \in [r]$.

A.2 Lipschitztasuna

A.2.1. definizioa. Izan bedi $C \subset \mathbb{R}^d$. Orduan $f : \mathbb{R}^d \to \mathbb{R}^k$ funtzioa *ρ-Lipschitziarra* dela diogu C multzoarekiko edozein $w_1, w_2 \in C$ harturik,

$$\|f(w_1) - f(w_2)\| \leq \rho \|w_1 - w_2\|$$

baldin bada.

A.2.1. adibidea. Izan bitez $v \in \mathbb{R}^d$ eta $b \in \mathbb{R}$. Orduan $f : \mathbb{R}^d \to \mathbb{R}$ non $f(w) = \langle v, w\rangle + b$ funtzioa $\|v\|$-Lipschitziarra da.

B. eranskina

Ariketak

B.1 1. Ariketa

Frogatu 2.2.7. lema. Hau da:

(i) $f(w) = \lambda\|w\|^2$ **funtzio konbexu** 2λ**-indartsua da.**

(ii) Izan bitez f **funtzio konbexu** λ**-indartsua eta** g **funtzio konbexua, orduan** $f + g$ **funtzio konbexu** λ**-indartsua da.**

(i) *Froga.* Lehenik eta behin, ohartu ondorengo bi berdintzak ditugula, norma euklidearrarekin lan egiten ari garelako:

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2\langle x, y\rangle, \quad \forall x, y \in \mathbb{R}^d, \tag{B.1}$$
$$\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2\langle x, y\rangle, \quad \forall x, y \in \mathbb{R}^d. \tag{B.2}$$

Ondoren, ohartu ondorengo desberdintza frogatu behar dugula:

$$f(\alpha w + (1-\alpha)u) \leq \alpha f(w) + (1-\alpha)f(u) - \frac{2\lambda}{2}\alpha(1-\alpha)\|w-u\|^2.$$

Edo, baliokideki:

$$f(\alpha w + (1-\alpha)u) - \alpha f(w) - (1-\alpha)f(u) \leq -\lambda\alpha(1-\alpha)\|w-u\|^2.$$

Azter dezagun orduan desberdintzaren ezkerreko aldea. $f = \lambda\|w\|^2$ denez ondorengoa dugu:

$$\lambda\|\alpha w + (1-\alpha)u\|^2 - \alpha\lambda\|w\|^2 - (1-\alpha)\lambda\|u\|^2.$$

Orduan, (B.1) formula erabiliz honela gelditzen da:

$$\lambda\Big(\|\alpha w\|^2 + \|(1-\alpha)u\|^2 + 2\langle \alpha w, (1-\alpha)u\rangle - \alpha\|w\|^2 - (1-\alpha)\|u\|^2\Big).$$

Hori sinplifikatuz:

$$\lambda\bigg(\alpha^2\|w\|^2+(1-\alpha)^2\|u\|^2+2\alpha(1-\alpha)\langle w,u\rangle-\alpha\|w\|^2-(1-\alpha)\|u\|^2\bigg).$$

Ondorioz:

$$\lambda\bigg(\alpha(\alpha-1)\|w\|^2+(1-\alpha)(1-\alpha-1)\|u\|^2+2\alpha(1-\alpha)\langle w,u\rangle\bigg).$$

Aldagai komunak ateraz:

$$\lambda\alpha(1-\alpha)\bigg(-\|w\|^2-\|u\|^2+2\langle w,u\rangle\bigg).$$

Azkenik, B.2 formula erabilita ondorengoa gelditzen zaigu:

$$-\lambda\alpha(1-\alpha)\|w-u\|^2.$$

Hau da:

$$f(\alpha w+(1-\alpha)u)-\alpha f(w)-(1-\alpha)f(u)=-\lambda\alpha(1-\alpha)\|w-u\|^2,$$

nahi bezala. □

(ii) *Froga.* Defini dezagun $h=f+g$ moduan eta izan bitez $w,u\in\mathbb{R}^d$ eta $\alpha\in(0,1)$. Orduan:

$$\begin{aligned}
&h(\alpha w+(1-\alpha)u)\\
&=f(\alpha w+(1-\alpha)u)+g(\alpha w+(1-\alpha)u)\\
&\leq \alpha f(w)+(1-\alpha)f(u)-\frac{\lambda}{2}\alpha(1-\alpha)\|w-u\|^2+\alpha g(w)+(1-\alpha)g(u)\\
&=\alpha h(w)+(1-\alpha)h(u)-\frac{\lambda}{2}\alpha(1-\alpha)\|w-u\|^2,
\end{aligned}$$

nahi bezala. □

B.2 2. Ariketa

Izan bedi $f : \mathbb{R}^d \to \mathbb{R}$ funtzioa non $f(w) = \frac{\lambda}{2}\|w\|^2 + L_S(w)$ den $\lambda > 0$ izanik eta

$$L_S(w) = \frac{1}{m}\sum_{i=1}^{m} l(w, z_i)$$

den, $S = \{z_i\}_{i=1}^m$ lagina izanik. Bestalde, izan bedi l galera funtzioa (gogoratu 1.3.3. definizioa). Demagun l galera-funtzio konbexua dela w aldagaiarekiko. Orduan, frogatu f funtzio konbexu λ-indartsua dela.

Froga. Lehenik eta behin, Galera Minimizazio Erregulatuan ikusi dugun bezala, ohartu hipotesi multzoa $\mathcal{H} = \mathbb{R}^d$ da. Galera funtzioa konbexua denez, A.1.4. lemako bigarren atala erabiliz eta $\frac{1}{m} \geq 0$ dela kontuan izanik, $L_S(w)$ funtzioa konbexua dela ondorioztatzen dugu.

Bestalde, B.1 ariketako lehenengo atalaren arabera, $\frac{\lambda}{2}\|w\|^2$ funtzioa λ-indartsua da. Orduan, ariketa berdineko bigarren atala erabiliz, $L_S(w)$ konbexua denez eta $\frac{\lambda}{2}\|w\|^2$ λ-indartsua, orduan f funtzioa λ-indartsua dela ondorioztatzen dugu. □

B.3 3. Ariketa

Izan bedi A algoritmoa ondorengoa bermatzen duena: $m \geq m_{\mathcal{H}}(\epsilon)$ lagin tamaina baldin badugu, orduan edozein $\mathcal{D}$ banaketarako:

$$\underset{S\sim\mathcal{D}^m}{\mathbb{E}}[L_{\mathcal{D}}(A(S))] \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

Orduan, frogatu edozein $\delta \in (0,1)$ baliorako, $m \geq m_{\mathcal{H}}(\epsilon\delta)$ badugu, orduan gutxienez $1-\delta$ probabilitatearekin ondorengoa betetzen da:

$$L_{\mathcal{D}}(A(S)) < \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

(i) *Froga.* Lehenik eta behin, defini dezagun ondorengo zorizko aldagaia:

$$Z = L_{\mathcal{D}}(A(S)) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h).$$

A algoritmoak $\mathcal{H}$ klaseko hipotesiak itzultzen dituenez, argi dago Z zorizko aldagaia ez-negatiboa izango dela. Orduan, Markov-en desberdintza erabiliz

$$\mathbb{P}[Z \geq \epsilon] \leq \frac{\mathbb{E}[Z]}{\epsilon}$$

izango dugu. Hau da,

$$\mathbb{P}[Z \geq \epsilon] \leq \frac{\mathbb{E}[Z]}{\epsilon} = \frac{\underset{S\sim\mathcal{D}^m}{\mathbb{E}}[L_{\mathcal{D}}(A(S)) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)]}{\epsilon}.$$

Orduan, $\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$ laginarekiko askea denez, konstante moduan hartuz eta $m \geq m_{\mathcal{H}}(\epsilon\delta)$ dela kontuan izanik ondorengoa dugu:

$$\mathbb{P}[Z \geq \epsilon] \leq \frac{\underset{S\sim\mathcal{D}^m}{\mathbb{E}}[L_{\mathcal{D}}(A(S))] - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)}{\epsilon} \leq \frac{\epsilon\delta}{\epsilon} = \delta$$

Beraz, gutxienez $1-\delta$ probabilitatearekin, $Z < \epsilon$ izango da, hau da:

$$L_{\mathcal{D}}(A(S)) < \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

Ondorioz, PAC-ikasgarritasun agnostikoa bermatu dugu. □

B.4 4. Ariketa

Izan bedi $S \subset \mathbb{R}^d \times \{\pm 1\}$ linealki banangarria den m tamainako lagina. Orduan, frogatu ondorengo bi problemak baliokideak direla:

(i) 3.1 problema, hau da:

$$\underset{(w,b):\|w\|=1}{\operatorname{argmax}} \left[\min_{i\in[m]} |\langle w, x_i\rangle + b| \text{ **non** } y_i(\langle w, x_i\rangle + b) > 0, \forall i \in [m] \right] \quad \text{(B.3)}$$

(ii) 3.2 problema, hau da:

$$\underset{(w,b):\|w\|=1}{\operatorname{argmax}} \left[\min_{i\in[m]} y_i(\langle w, x_i\rangle + b) \right] \quad \text{(B.4)}$$

Froga. Lehenik eta behin, defini dezagun ondorengo multzoa:

$$\mathcal{G} := \{(w,b) \in \mathbb{R}^d \times \mathbb{R}, y_i(\langle w, x_i\rangle + b) > 0, \forall i \in [m]\}.$$

Hau da, $\mathcal{G}$ multzoa S lagineko elementu guztiak ongi sailkatzen dituzten espazio erdibikarien multzoa da.

Lehenik eta behin, ohartu (B.3) problemako soluzioak $\mathcal{G}$ multzoan daudela definizioz. Ikus dezagun orain (B.4) problemako soluzioak ere $\mathcal{G}$ multzoan egongo direla.

Horretarako, izan bedi (w,b) (B.4) problemako soluzioa eta demagun, absurdura eramanez, existitzen dela $j \in [m]$ balio bat non $y_j(\langle w, x_j\rangle + b) \leq 0$. Baliteke j hartzeko hainbat aukera izatea (hainbat balio negatibo baldin badaude), beraz, gehiago zehaztuz, har dezagun j non

$$y_j(\langle w, x_j\rangle + b) = \min_{i\in[m]} y_i(\langle w, x_i\rangle + b) \leq 0$$

betetzen den.

S lagina linealki banangarria denez, existitzen da $(\hat{w}, \hat{b})$ espazio erdibikari bat non $i \in [m]$ guztietarako

$$y_i(\langle \hat{w}, x_i\rangle + \hat{b}) > 0$$

den. Orduan, bi emaitzak elkartuz ondorengoa dugu:

$$\min_{i\in[m]} y_i(\langle w, x_i\rangle + b) \leq 0 < \min_{i\in[m]} y_i(\langle \hat{w}, x_i\rangle + \hat{b}).$$

Baina hori absurdoa da (w,b) (B.4) problemako soluzioa delako. Ondorioz, (w,b) (B.4) problemako soluzioa baldin bada, $(w,b) \in \mathcal{G}$ izango da.

Behin hori frogatu dugula, ikus dezagun edozein $(w, b) \in \mathcal{G}$ baldin badugu, ondorengo berdintza dugula:

$$\min_{i\in[m]} y_i(\langle w, x_i\rangle + b) = \min_{i\in[m]} |\langle w, x_i\rangle + b|. \tag{B.5}$$

Ohartu berdintza hori frogatzerakoan amaitu dugula, izan ere, bi ekuazioen soluzioak $\mathcal{G}$ multzoan egongo dira eta ondorioz, (B.5) berdintza erabiliz baliokidetasuna lortzen da.

(B.5) berdintza frogatzeko nahikoa da definizioak erabiltzea. Alde batetik, ohartu $i \in [m]$ guztietarako

$$|\langle w, x_i\rangle + b| = \begin{cases} \langle w, x_i\rangle + b, & \langle w, x_i\rangle + b \geq 0, \\ -(\langle w, x_i\rangle + b), & \langle w, x_i\rangle + b < 0, \end{cases}$$

dela. Bestetik, (w, b) espazio erdibikariak S lagineko ale guztiak ongi sailkatzen dituenez eta $\mathcal{Y} = \{\pm 1\}$ denez, $i \in [m]$ guztietarako

$$y_i = \begin{cases} 1, & \langle w, x_i\rangle + b \geq 0, \\ -1, & \langle w, x_i\rangle + b < 0, \end{cases}$$

izango da. Orduan,

$$y_i(\langle w, x_i\rangle + b) = \begin{cases} 1 \cdot (\langle w, x_i\rangle + b), & \langle w, x_i\rangle + b \geq 0, \\ -1 \cdot (\langle w, x_i\rangle + b), & \langle w, x_i\rangle + b < 0. \end{cases}$$

Hau da, $i \in [m]$ guztietarako,

$$y_i(\langle w, x_i\rangle + b) = |\langle w, x_i\rangle + b|$$

betetzen da nahi bezala. □

B.5 5. Ariketa

Izan bedi 3.2.4. definizioko l^{hinge} banda-galera funtzioa. Hau da, izan bitez $\mathcal{X} = \mathbb{R}^d$ domeinua, $\mathcal{Y} = \{\pm 1\}$ izen-multzoa, $Z = \mathcal{X} \times \mathcal{Y}$ multzoa eta $\mathcal{H} \subset \mathbb{R}^d \times \mathbb{R}$ hipotesi-klasea espazio erdibikariak adierazten dituena. Orduan, $l^{\text{hinge}} : \mathcal{H} \times Z \to \mathbb{R}$ dugu non

$$l^{\text{hinge}}((w,b),(x,y)) = \max\{0, 1 - y(\langle w, x\rangle + b)\}$$

Ariketa honen helburua banda-galeraren l^{0-1} galera-funtzioaren ordezko-funtzio izateko egokitasuna aztertzea da. Frogatu ondorengoak:

(i) Frogatu l^{0-1} funtzioa ez dela konbexua.

(ii) Frogatu banda-galera funtzioak $0-1$ galera funtzioa goitik bornatzen duela, hau da, frogatu edozein $(x,y) \in \mathcal{X} \times \mathcal{Y}$ alerako eta edozein (w,b) espazio erdibikarirako,

$$l^{0-1}((w,b),(x,y)) \leq l^{\text{hinge}}((w,b),(x,y))$$

dela non l^{0-1} 1.3.4. definizioko galera funtzioa den.

(iii) Frogatu banda-galera funtzioa konbexua dela.

(iv) Hemendik aurrera kontsidera dezagun kasu homogeneoa, hau da,

$$l^{\text{hinge}}(w,(x,y)) = \max\{0, 1 - y\langle w, x\rangle\}$$

Zein da banda-galeraren subgradientea w bektorean?

(v) Frogatu l^{hinge} funtzioa $\|x\|$-Lipschitz dela (kasu homogeneorako).

(i) *Froga.* Notazioa sinplifikatzeko, ez-konbexutasuna (w,b) aldagaiaren menpe aztertu behar dugunez lagineko elementu finko bat izanik, finka dezagun (x,y) lagineko elementua eta idatz dezagun $0-1$ galera funtzioa ondorengo moduan:

$$l^{0-1}((w,b),(x,y)) = l^{0-1}((w,b)).$$

Orduan, $\alpha \in (0,1)$ balioa eta $(w_1,b_1),(w_2,b_2)$ bikote egokiak aurkitu behar ditugu lagineko elementu zehatz baterako ondorengo berdintza betetzen ez dutenak:

$$l^{0-1}(\alpha(w_1,b_1)+(1-\alpha)(w_2,b_2)) \leq \alpha \cdot l^{0-1}((w_1,b_1))+(1-\alpha)\cdot l^{0-1}((w_2,b_2))$$

Har ditzagun ondorengo balioak $\mathcal{X} = \mathbb{R}^2$ kasurako:

$$w_1 = (1,1), b_1 = -1,$$
$$w_2 = (1,1), b_2 = -3.$$

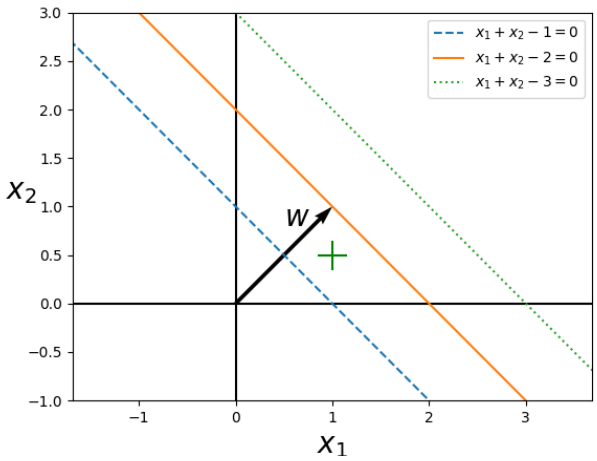
Bestalde, har dezagun $\alpha = 1/2$. Orduan, bi hipotesien konbinazioa ondorengoa izango da:

$$\begin{aligned}\alpha(w_1,b_1) + (1-\alpha)(w_2,b_2) &= (\alpha w_1, \alpha b_1) + ((1-\alpha)w_2, (1-\alpha)b_2) \\ &= (\alpha w_1 + (1-\alpha)w_2, \alpha b_1 + (1-\alpha)b_2) \\ &= ((\alpha + 1 - \alpha)(1,1), -\alpha + (1-\alpha)\cdot(-3)) \\ &= ((1,1), -2).\end{aligned}$$

Ohartu hiru espazio erdibikariek sortzen dituzten hiperplanoak ondorengoak direla:

$$w_1 = (1,1), b_1 = -1 \Longrightarrow x_1 + x_2 - 1 = 0,$$
$$w_2 = (1,1), b_2 = -3 \Longrightarrow x_1 + x_2 - 3 = 0,$$
$$\hat{w} := (1,1), \hat{b} := -2 \Longrightarrow x_1 + x_2 - 2 = 0,$$

non (x_1, x_2) $\mathbb{R}^2$ espazioko ardatzak diren. Horiek irudikatzen baldin baditugu B.1. irudia lortzen dugu.



B.1. irudia. Hiru hiperplanoen irudikapena.

Orduan, ohartu lagineko elementuren bat $x_1 + x_2 - 1 = 0$ hiperplanoaren eta $x_1 + x_2 - 2 = 0$ hiperplanoaren artean hartzen badugu

bere izendapena 1 izanik, $x_1+x_2-1=0$ hiperplanoak soilik sailkatuko duela ongi.

Adibidez, har dezagun $(x,y)=((1,1/2),1)$ elementua eta azter ditzagun hiru hipotesi horien sailkapenak elementu horrentzako:

$$\begin{aligned}(w_1,b_1) &\Longrightarrow \operatorname{sign}(1+1/2-1)=\operatorname{sign}(1/2)=1,\\ (w_2,b_2) &\Longrightarrow \operatorname{sign}(1+1/2-3)=\operatorname{sign}(-3/2)=-1,\\ (\hat{w},\hat{b}) &\Longrightarrow \operatorname{sign}(1+1/2-2)=\operatorname{sign}(-1/2)=-1.\end{aligned}$$

Orduan, hartu dugun lagineko elementuaren izendapena 1 denez, ohartu hiru hipotesietatik elementua ongi sailkatuko duen bakarra (w_1,b_1) izango dela. Ondorioz, $(x,y)=((1,1/2),1)$ elementuarentzako:

$$\begin{aligned}l^{0-1}(w_1,b_1)&=0,\\ l^{0-1}(w_2,b_2)&=1,\\ l^{0-1}(\hat{w},\hat{b})&=1.\end{aligned}$$

Beraz, kasu honetarako desberdintza ondorengoa denez ($\alpha=1/2$ hartu dugulako):

$$l^{0-1}(\hat{w},\hat{b})\leq 1/2\cdot l^{0-1}(w_1,b_1)+1/2\cdot l^{0-1}(w_2,b_2)$$

Orduan ondorengo kontraesanera iristen gara:

$$1\leq 1/2\cdot 0+1/2\cdot 1\Longrightarrow 1\leq 1/2$$

Eta ondorioz, $0-1$ galera funtzioa ez da konbexua.

□

(ii) *Froga.* Lehenik eta behin, 1.3.4. definizioa erabiliz:

$$l^{0-1}((w,b),(x,y))=\begin{cases}0, & y(\langle w,x\rangle+b)\geq 0\\ 1, & y(\langle w,x\rangle+b)<0\end{cases}$$

dugu. Ohartu zeinu funtzioa aplikatu beharrean baliokidea den formulazioa erabili dugula. Orduan, bi kasu desberdinduko ditugu.

Lehenik eta behin, demagun $y(\langle w,x\rangle+b)<0$ dugula. Hau da, $l^{0-1}((w,b),(x,y))=1$ izango da eta bestalde, desberdintza zertxobait manipulatuz, $1<1-y(\langle w,x\rangle+b)$ izango dugu. Hau da,

$$1-y(\langle w,x\rangle+b)>0$$

izango denez,

$$l^{\text{hinge}}((w,b),(x,y)) = 1 - y(\langle w,x\rangle + b)$$

izango dugu. Ondorioz, aurreko guztia elkartuz:

$$l^{0-1}((w,b),(x,y)) = 1 < 1 - y(\langle w,x\rangle + b) = l^{\text{hinge}}((w,b),(x,y))$$

Bestalde, demagun orain $y(\langle w,x\rangle + b) \geq 0$ dugula. Orduan, kasu honetan $l^{0-1}((w,b),(x,y)) = 0$ izango da. Orduan,

$$0 \leq \max\{0, 1 - y(\langle w,x\rangle + b)\}$$

denez,

$$l^{0-1}((w,b),(x,y)) = 0 \leq \max\{0, 1-y(\langle w,x\rangle+b)\} = l^{\text{hinge}}((w,b),(x,y))$$

izango da nahi bezala. □

(iii) *Froga.* Funtzioaren konbexutasuna hipotesiaren aldagaiarekiko aztertu behar dugunez, finka dezagun (x,y) lagineko elementua. Orduan, notazioa errazteko $l^{\text{hinge}}((w,b),(x,y)) = l^{\text{hinge}}((w,b))$ idatziko dugu.

Banda-galera funtzioa konbexua dela frogatzeko, lehenik eta behin A.1.4. lemaren lehenengo atala erabiliko dugu. Orduan, 0 funtzio konbexua denez, ikusi behar dugun bakarra da ea

$$f((w,b)) := 1 - y(\langle w,x\rangle + b)$$

funtzioa konbexua den.

Hori aztertzeko A.1.3. lema erabiliko dugu. Lema erabili ahal izateko, defini dezagun $\hat{x} = y(x_1,\ldots,x_d,1) \in \mathbb{R}^{d+1}$ eta ohartu

$$f((w,b)) = f(w_1,\ldots,w_d,b) = 1 - \langle (w_1,\ldots,w_d,b),\hat{x}\rangle.$$

Lemaren notazioa erabiltzen badugu, $g : \mathbb{R} \to \mathbb{R}$ ondorengo moduan definituko genuke:

$$g(x) = 1 - x.$$

Azkenik, A.1.2. lemarengatik g funtzioa konbexua denez, f funtzioa konbexua izango da eta ondorioz l^{hinge} ere. □

(iv) Banda-galera funtzioaren subgradienteak lortzeko 2.2.4. lema erabiliko dugu. Lehenik eta behin, subgradientea kalkulatzeko w aldagaiaren menpe egin behar dugunez, finka dezagun (x,y) lagineko elementua eta idatz dezagun $l^{\text{hinge}}(w,(x,y)) = l^{\text{hinge}}(w)$ moduan. Bestalde, notazioa

argiago izan dadin eta 2.2.4. lemaren notazioa jarraitzeko, ondorengo bi funtzioak definituko ditugu:

$$\begin{aligned} g_1(w) &= 0, \\ g_2(w) &= 1 - y\langle w, x\rangle. \end{aligned}$$

Hau da, $l^{\text{hinge}}(w) = \max\{g_1(w), g_2(w)\}$.

Orduan, bi kasu bananduko ditugu. Alde batetik, demagun $g_1(w) < g_2(w)$ dela. Orduan, lemaren arabera $\nabla g_2(w) \in \partial l^{\text{hinge}}(w)$ izango da. Bestalde, erraz kalkulatzen da

$$\begin{aligned} \nabla g_2(w) &= \nabla(1 - y(w_1x_1 + \cdots + w_dx_d)) \\ &= -yx \end{aligned}$$

dela.

Bestetik, demagun orain $g_1(w) \geq g_2(w)$ kasuan gaudela. Orduan, lemaren arabera $\nabla g_1(w) \in \partial l^{\text{hinge}}(w)$ izango da eta ohartu $\nabla g_1(w) = \nabla 0 = \vec{0}$ dela.

Laburbilduz, l^{hinge} funtzioaren w bektoreko subgradientea ondorengo v_w bektorea da:

$$v_w = \begin{cases} \vec{0}, & 1 - y\langle w, x\rangle \leq 0, \\ -yx, & 1 - y\langle w, x\rangle > 0. \end{cases}$$

Gainera, $g_1(w)$ eta $g_2(w)$ funtzioak konbexuak eta diferentziagarriak direnez, [5] apunteetako 7. orrialdeko teorian oinarrituz, l^{hinge} funtzioaren subgradiente guztiak v_w moduan definitzen dira.

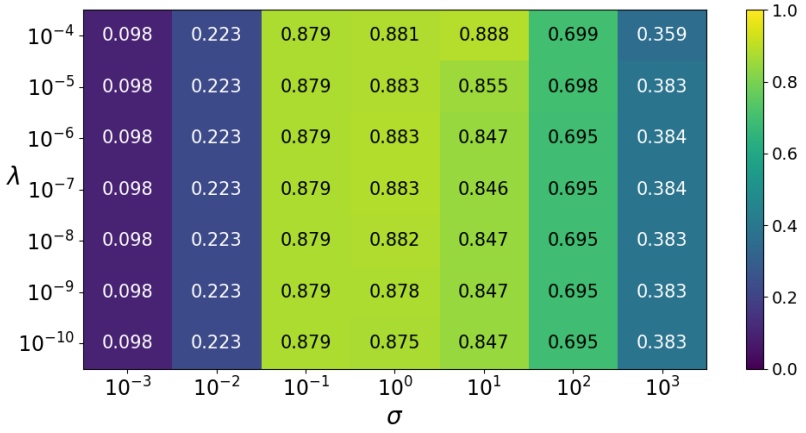
(v) *Froga.* Aurreko atala erabiliz, galera-funtzioaren subgradienteek αx forma dute $|\alpha| \leq 1$ izanik. Ondorioz, 2.2.5. lema aplikatzen baldin badugu, l^{hinge} funtzioa $\|x\|$-Lipschitz da. □

C. eranskina

5. kapituluko hainbat emaitza gehigarri

Eranskin honetan 5. kapituluko hainbat azterketa gehigarriren emaitzak adieraziko ditugu. Emaitza horiek forma bisualean emateko grafiko asko behar direnez, eta atal honetan ikusiko ditugun emaitzak lanerako esanguratsuak izan ez direnez, eranskin moduan jarri ditut.

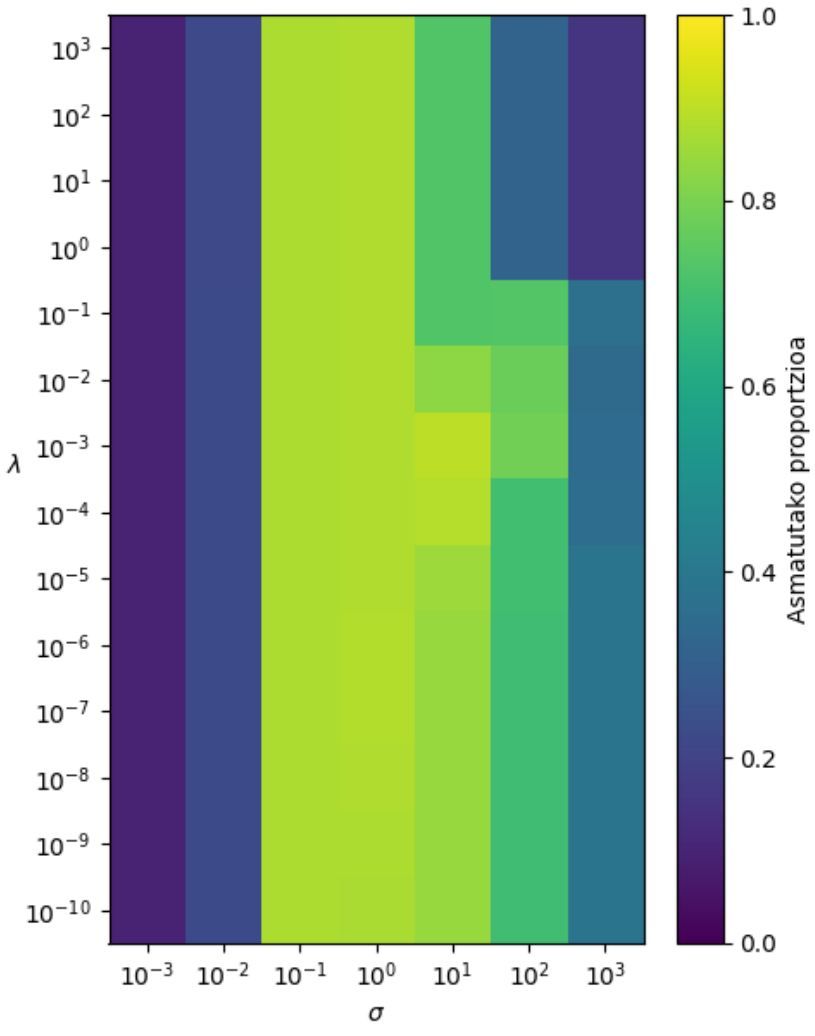
C.1 Kernel gaussiarraren bitarteko entrenamendu gehiago



C.1. irudia. Nire inplementazioko ereduen asmatze-proportzioa Kernel gaussiarrarekin eta 1000 iterazioko exekuzioekin: bigarren azterketa.

5. kapituluan zehar, kernel gaussiarreko nire inplementazioko ereduetan, bazidurien geroz eta λ txikiagoa hartu, orduan eta eredu hobea genuela.

Ondorioz, λ balio txikiagoetarako beste hainbat entrenamendu egin ditugu, baina ez dugu ezer esanguratsua lortu, C.1. irudian ikus daitekeen moduan. Bestalde, C.2. irudian bi entrenamenduak elkartuak ikus daitezke, asmatze-proportzioa parametroen arabera nola aldatzen den hobeto ikusteko (kolorearen arabera).



C.2. irudia. Nire inplementazioko ereduen asmatze-proportzioa Kernel gaussiarrarekin eta 1000 iterazioko exekuzioekin: bi azterketak elkartuak.

C.2 Eredu hoberenen konfusio-matrizeak eta gaizki klasifikatu diren hainbat digitu

Lanean aipatu dugun moduan, ereduen asmatze-proportzioa ondorengoa izan da:

(i) Nire eredua, kernel gaussiarrarekin: 0.9332,

(ii) Nire eredua, kernel polinomialarekin: 0.9232,

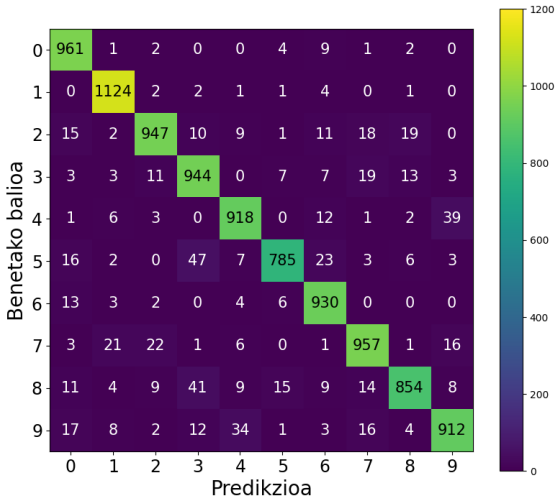
(iii) Scikit paketeko eredua, kernel gaussiarrarekin: 0.9833,

(iv) Scikit paketeko eredua, kernel polinomialarekin: 0.9827.

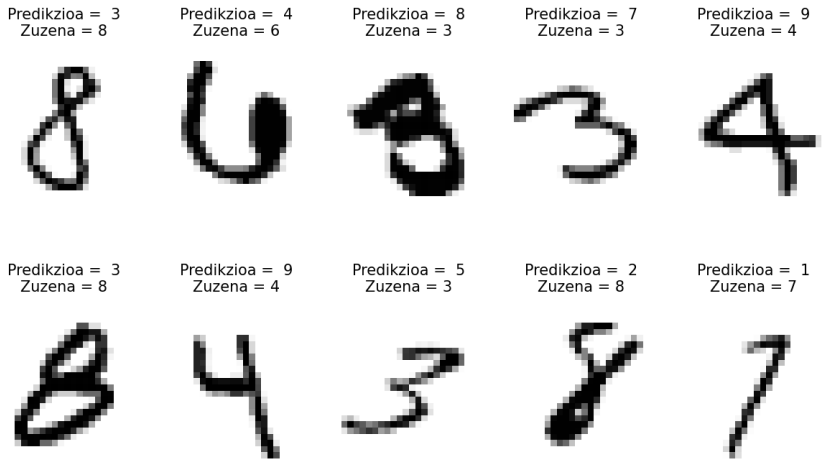
Lau eredu horien konfusio-matrizeak ikusiko ditugu ondoren. Hau da, test-multzoko elementu guztien artean (10.000 elementu guztira), zeintzuk ongi klasifikatu diren eta zeintzuk gaizki.

Bestalde, gaizki klasifikatutako elementuen artean, ausaz hainbat digitu ikusiko ditugu haien klasifikazioarekin eta benetako izenarekin batera.

Nire inplementazioko ereduak: kernel gaussiarra

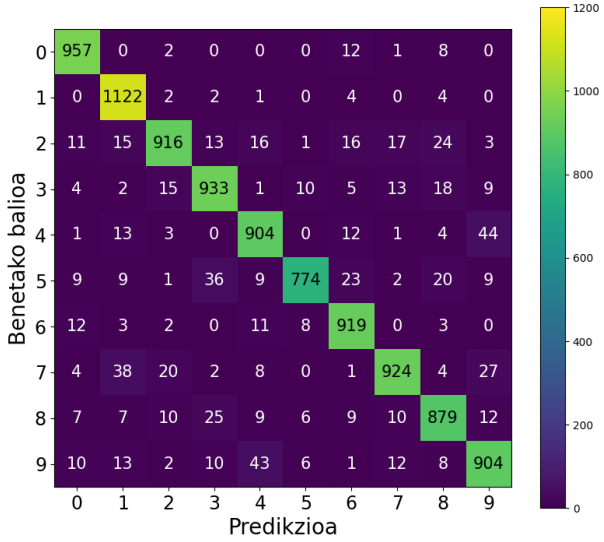


C.3. irudia. Nire inplementazioko Kernel gaussiarreko eredu hoberenak egin dituen klasifikazioen konfusio-matrizea.

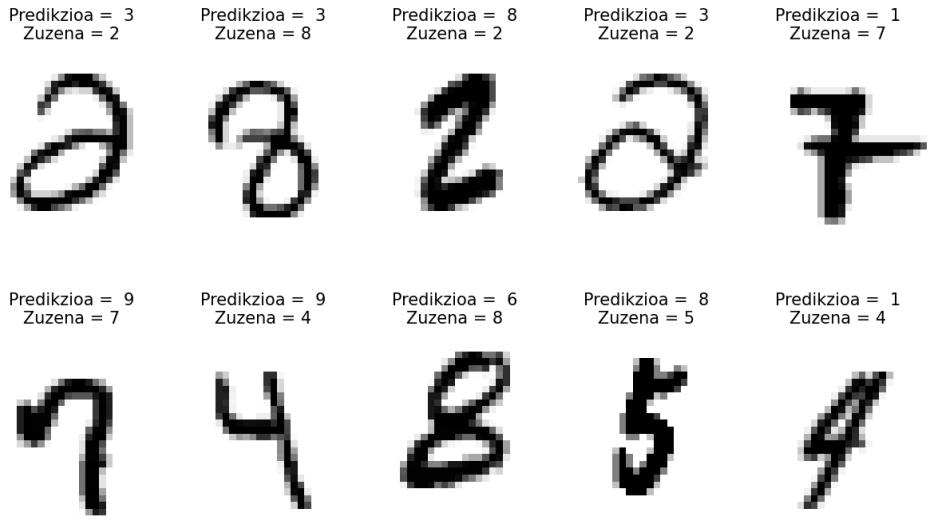


C.4. irudia. Nire inplementazioko Kernel gaussiarreko eredu hoberenak gaizki klasifikatutako hainbat digitu.

Nire inplementazioko ereduak: kernel polinomiala

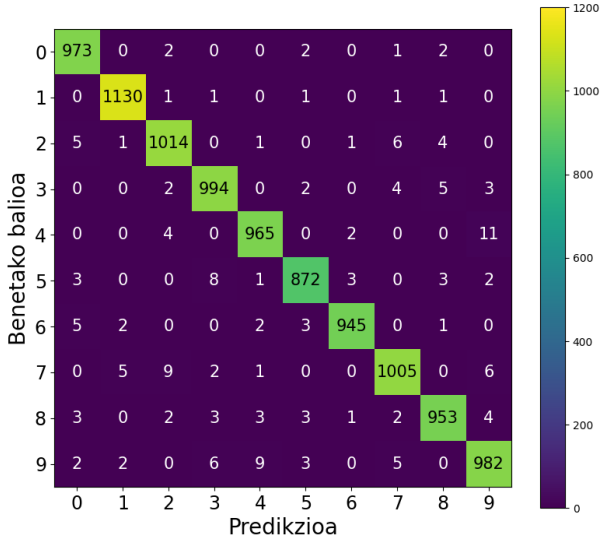


C.5. irudia. Nire inplementazioko Kernel polinomialeko eredu hoberenak egin dituen klasifikazioen konfusio-matrizea.

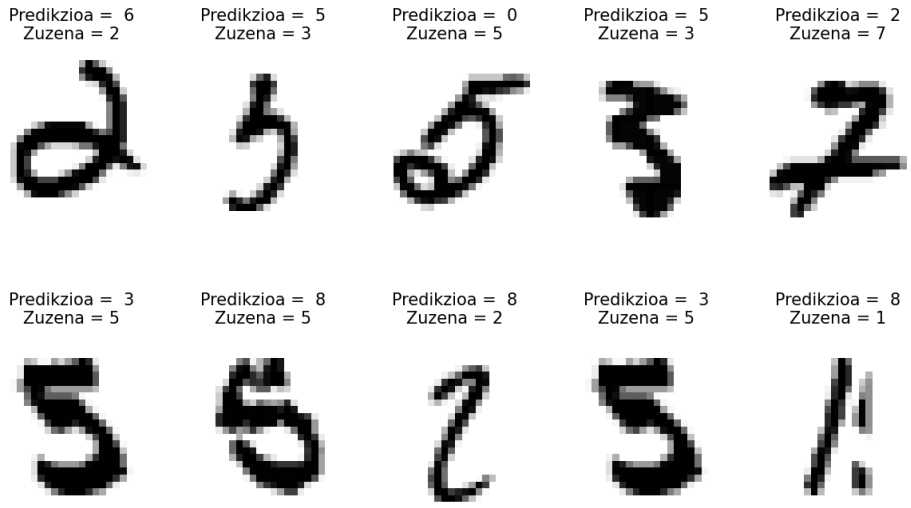


C.6. irudia. Nire inplementazioko Kernel polinomialeko eredu hoberenak gaizki klasifikatutako hainbat digitu.

Scikit-learn paketeko ereduak: kernel gaussiarra

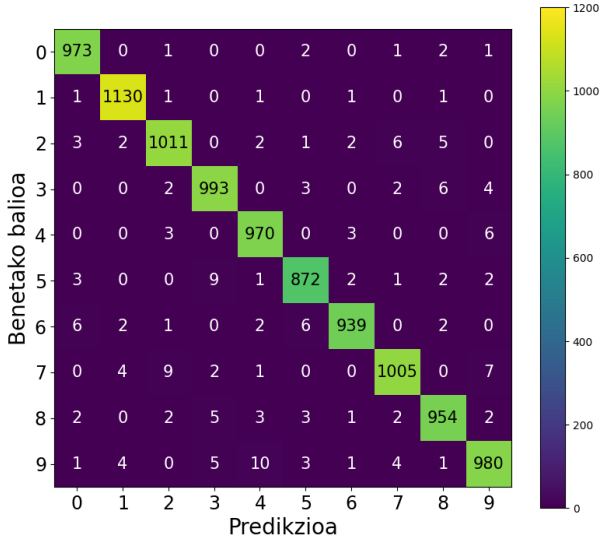


C.7. irudia. Scikit-learn paketeko Kernel gaussiarreko eredu hoberenak egin dituen klasifikazioen konfusio-matrizea.

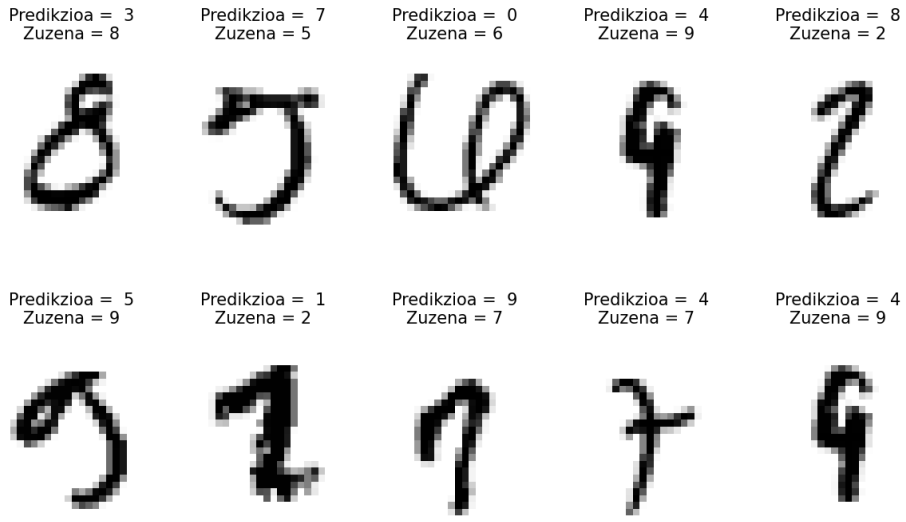


C.8. irudia. Scikit-learn paketeko Kernel gaussiarreko eredu hoberenak gaizki klasifikatutako hainbat digitu.

Scikit-learn paketeko ereduak: kernel polinomiala



C.9. irudia. Scikit-learn paketeko Kernel polinomialeko eredu hoberenak egin dituen klasifikazioen konfusio-matrizea.



C.10. irudia. Scikit-learn paketeko Kernel polinomialeko eredu hoberenak gaizki klasifikatutako hainbat digitu.

Ondorio moduan, nahiz eta konfusio-matrizeak antzekoak izan, garbi ikus daiteke Scikit paketeko ereduek zehaztasun hobea dutela klasifikatzerako orduan. Hori garbi ikusten da gaizki klasifikatutako digituak aztertzen baditugu, izan ere, nabari da Scikit paketeko ereduek gaizki klasifikatutako digituak klasifikatzen zailagoak direla (kaligrafia txarreko digituak direlako, hala nola).

D. eranskina

5. kapituluko kodea

Eranskin honetan algoritmoaren inplementazioa Pythonen dago, egin ditudan analisien eta grafiko guztien kodearekin batera. Aplikazio interaktiboaren kodeari dagokionez, hori nire GitHub orrialdean egongo da (`https://github.com/JulenErcibengoa`), izan ere, aplikazioaren kodeak ez du inongo interes zientifikorik eta luzeegia da hemen jartzeko.

D.1 Algoritmoaren inplementazioa

Hemen kernel bidezko SVM-leunaren inplementazioa Pythonen agertzen da. Hau da, 6 eta 7. algoritmoen eta 4.3 formularen Python lengoaiean idatzitako inplementazio bat da. Kodeari dagokionez, 6. algoritmoa 41-93 lerroetan dago. Ohartu lehenik eta behin 3.1.3. oharra jarraituz, lagina homogeneizatu egiten dela kodeko 46. lerroan. Ondoren, kodea bi zatitan banantzen da: alde batetik kasu dikotomikoa eta bestetik klase anizkoitzerako kasua. Kasu dikotomikorako zuzenean aplikatzen da 6. algoritmoa eta klase anizkoitzerako kasuan 4 kapituluan agertzen den 7. algoritmoa aplikatzen da.

Algoritmoa exekutatzean ohartu α bektoreak gordetzen direla, kernel funtzioaren bitartez kalkuluak egiteko eta txertaketa funtzioarekin kalkulurik egin behar ez izateko.

Azkenik, ohartu predikzioak egiteko 4.3 formula erabiltzen dela 95 eta 111 lerroen artean. Era berean, kasu dikotomikoa eta klase anizkoitzeko kasuak desberdintzen dira eta azken honetan, 4. kapituluko azken atalean azaldutako formula erabiltzen da.

```
1 import random
2 import numpy as np
3 import math
4
5 def polynomial_kernel(xi,xj,deg = 2,sigma = 0.1):
6     return (np.dot(xi,xj) + 1)**deg
```

```
7
8  def gaussian_kernel(xi,xj,sigma = 0.1,deg = 2):
9      return np.exp( (-1) * (np.dot(xi-xj,xi-xj))/ (2*sigma) )
10
11 def linear_kernel(xi,xj,sigma = 0.1,deg = 2):
12     return np.dot(xi,xj)
13
14 class Nire_SGD_kernelekin:
15     def __init__(self,koeficient,kernel,sigma = 0.1,deg = 2):
16         self.entrenatuta = False
17         self.klaseanitz = False
18         self.alpha = None # Modeloaren entrenatu ondorengo
       aldagaiak hemen egongo dira
19         self.X = None
20         self.Y_bakarrak = None
21         self.m = None
22         self.koeficient = koeficient
23         self.sigma = sigma
24         self.deg = deg
25
26
27         # Kernel aukeraketa:
28         if kernel == "kernel gaussiarra":
29             self.kernel = gaussian_kernel
30         elif kernel == "kernel polinomiala":
31             self.kernel = polynomial_kernel
32         elif kernel == "kernel lineala":
33             self.kernel = linear_kernel
34         else:
35             print("Sartu duzun kernela ez da zuzena!")
36             self.kernel = gaussian_kernel
37
38     def kernelak_kalkulatu(self,x):
39         return [self.kernel(xi,x,deg = self.deg, sigma = self.
       sigma) for xi in self.X]
40
41     def fit(self,X,Y,iter = 10000):
42         """
43         Algoritmoa entrenatzen du X bektore eta Y izeneko lagin
        batekin.
44         """
45         self.m = len(X)
46         self.X =  np.concatenate( (np.ones((self.m,1)),X) ,
       axis = 1) # Ondoren predikzioak egiteko gorde egingo dugu
       klasearen barruan
47         if len(np.unique(Y)) == 2: # Klasifikazio dikotomikoa
48             T = iter
49             alpha = np.zeros([T,self.m])
50             beta = np.zeros([T+1,self.m]) # T+1 jartzen dugu
       azken iterazioan arazorik ez egoteko, hala ere ez dugu
       iterazio horko informazioa erabiliko
51             # Algoritmoa:
52             for t in range(T):
53                 # 1. Pausua:
```

```
54                alpha[t,:] = 1 / (self.koeficient * (t+1)) *
    beta[t,:] # t+1 egiten dugu Pythonen indizeak 0-n hasten
    direlako
55                # 2. Pausua:
56                i = random.randint(1,self.m) - 1 # -1 egiten
    dugu indizeen arazoa konpontzeko
57                # 3. Pausua:
58                beta[t+1,:] = beta[t,:] # Oraingoz beta_i^(t+1)
     = beta_i^(t) da, gero aldatuko dugu
59                # 4. Pausua, kernelak kalkulatu:
60                kernels = self.kernelak_kalkulatu(self.X[i])
61                if (Y[i]*np.dot(alpha[t,:],kernels) < 1):
62                    beta[t+1,i] = beta[t,i] + Y[i]
63                else:
64                    beta[t+1,i] = beta[t,i]
65            self.alpha =  (1/T) * np.sum(alpha,0) # Predikzioak
     egiteko definitu
66
67        else: # Klasifikazio anitzkoitza
68            self.klaseanitz = True
69            self.alpha = [] # k klaserako, k hipotesi
    desberdinen 'outputak'
70            self.Y_bakarrak = np.unique(Y)
71            Y_desberdinak = []
72            for izena in self.Y_bakarrak:
73                Y_desberdinak.append([1 if elementua == izena
    else -1 for elementua in Y])
74            r = 0
75            for Y_desb in Y_desberdinak:
76                # Entrenatu (goiko kode berdina)
77                T = iter
78                alpha = np.zeros([T,self.m])
79                beta = np.zeros([T+1,self.m])
80                for t in range(T):
81                    alpha[t,:] = 1 / (self.koeficient * (t+1))
    * beta[t,:]
82                    i = random.randint(1,self.m) - 1
83                    beta[t+1,:] = beta[t,:]
84                    kernels = self.kernelak_kalkulatu(self.X[i
    ])
85                    if (Y_desb[i]*np.dot(alpha[t,:],kernels) <
    1):
86                        beta[t+1,i] = beta[t,i] + Y_desb[i]
87                    else:
88                        beta[t+1,i] = beta[t,i]
89                self.alpha.append((1/T) * np.sum(alpha,0))
90                r+=1
91                print(f"{r}/{len(self.Y_bakarrak)} klase
    entrenatuta")
92        self.entrenatuta = True
93        print("Entrenamendua amaituta!")
94
95    def predict(self,x):
96        """
```

```
97          x domeinuko elementu berri batentzako, jada
    entrenatutako modeloak ematen duen x-ren izena itzuliko du.
98          """
99          x_predict = np.concatenate(([1],x))
100         if self.entrenatuta and not self.klaseanitz: #
    Klasifikazio normala (Y = {+1, -1})
101             kernelak = self.kernelak_kalkulatu(x_predict)
102             balioa = np.dot(self.alpha,kernelak)
103             return 1 if balioa > 0 else -1
104         elif self.entrenatuta and self.klaseanitz: #
    Klasifikazio anitzkoitza
105             kernelak = self.kernelak_kalkulatu(x_predict)
106             balioak = []
107             for alpha in self.alpha:
108                 balioak.append(np.dot(alpha,kernelak))
109             return self.Y_bakarrak[np.argmax(balioak)]
110         else:
111             print("Modeloa oraindik ez da entrenatu")
112
113     def predict_anitzkoitza(self,x_multzoa):
114         labels = np.zeros(len(x_multzoa))
115         for i,x in enumerate(x_multzoa):
116             labels[i] = self.predict(x)
117         return labels
118
119     def score(self,X,Y):
120         """
121         X bektoreko eta Y izeneko datu-multzo bat emanik,
    modeloak dauden datu guztietatik ongi klasifikatzen dituen
    proportzioa itzuliko du
122         """
123         if self.entrenatuta:
124             m = len(X)
125             klasifikazio_zuzen_kopurua = 0
126             for i,bektore in enumerate(X):
127                 if self.predict(bektore) == Y[i]:
128                     klasifikazio_zuzen_kopurua += 1
129             return klasifikazio_zuzen_kopurua / m
130         print("Modeloa oraindik ez da entrenatu")
```

D.2 Eredu hoberena hautatzeko kodea

Ondorengo kodean 5.4 ataleko eta 5.5 ataleko emaitzak lortzeko Pythoneko scriptak daude hurrenez hurren. Hau da, horietan hainbat parametro ezberdineko eredu entrenatzen dira eta ondoren eredu horien asmatze-proportzioak matrize moduan adierazten dira.

Kodean zehar atal asko komentario moduan jarriak daude, izan ere, scriptak informazioa gordetzeko sistema bat du. Hau da, lehenengo exekuzioan hainbat aldagai sortu eta gordetzen dira ordenagailuan, hurrengo exekuzioetan aldiz, berriro sortu beharrean, jada ordenagailuan gordeta daudenez inportatu egiten dira. Horrela, 49 eredu desberdin jarraian entrenatu beharrean, gutxika-gutxika entrena daitezke. Hau guztiz beharrezkoa zen, izan ere, jarraian entrenatu behar izango balira, ordenagailua egun gehiegi egongo litzateke exekutatzen gelditu gabe eta, entrenamenduan erroreren bat egongo balitz (argia joaten bada adibidez), lortutako informazio guztia galduko litzateke.

D.2.1 Nire inplementazioko ereduak

```
import sys
import os

oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
bide_orokorra = os.path.dirname(oraingo_bidea)
sys.path.insert(0,os.path.join(bide_orokorra, "Algoritmoak"))

from SGD_soft_SVM_Kernels import Nire_SGD_kernelekin
import numpy as np
import pandas as pd
import pickle
import matplotlib.pyplot as plt
import random



# # -------------------------------
# # ------DATU BASEA INPORTATU-----
# # -------------------------------
mnist_test_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_test.csv")
mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_train.csv")

entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
testeatzeko_datuak = pd.read_csv(mnist_test_bidea)
X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
Y_entrenamendu = entrenamendu_datuak["label"]
X_test = testeatzeko_datuak.iloc[:,1:]
```

```
28 Y_test = testeatzeko_datuak["label"]
29 X_entrenamendu = X_entrenamendu / 255
30 X_test = X_test / 255
31 print("Entrenamenduko klaseen agertze proportzioa:")
32 print(Y_entrenamendu.value_counts()/len(Y_entrenamendu))
33 print()
34 print("Testeko klaseen agertze proportzioa:")
35 print(Y_test.value_counts()/len(Y_test))
36 print()
37 # # --------------------------------
38 # # --------------------------------
39 # # --------------------------------
40
41
42
43 # # --------------------------------
44 # # INFORMAZIOA GORDETZEKO LEKUAK
45 # # SORTU (LEHENENGO ALDIZ EJEKUTATZEAN)
46 # # --------------------------------
47 # # Hau lehenengo aldiz exekutatzean bakarrik exekutatu,
     bestela informazioa galdu egingo da
48
49 # Notak_matrizea_Nire_rbf = np.zeros((7,7))
50 # Notak_matrizea_Nire_rbf_handitua = np.zeros((14,7))
51 # Notak_matrizea_Nire_poly = np.zeros((7,7))
52
53 # pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
     oraingo_bidea,"Modeloen_informazioa","
     Notak_matrizea_Nire_rbf.pkl"),"wb"))
54 # pickle.dump(Notak_matrizea_Nire_rbf_handitua,open(os.path.
     join(oraingo_bidea,"Modeloen_informazioa","
     Notak_matrizea_Nire_rbf_handitua.pkl"),"wb"))
55 # pickle.dump(Notak_matrizea_Nire_poly,open(os.path.join(
     oraingo_bidea, "Modeloen_informazioa","
     Notak_matrizea_Nire_poly.pkl"),"wb"))
56
57 # with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
     ",'Nire_modeloen_notak.txt'), 'w') as informazioa:
58 #     informazioa.write("Nire modelo desberdinak, kernel
     gaussiarra\n\n")
59
60 # # ------------------------------
61 # # ------------------------------
62 # # ------------------------------
63
64
65 # # --------------------------------
66 # # INFORMAZIOA GORDETZEKO LEKUAK
67 # # IREKI (LEHENENGO ITERAZIOAN EZ)
68 # # --------------------------------
69
70 Notak_matrizea_Nire_rbf = pickle.load(open(os.path.join(
     oraingo_bidea,"Modeloen_informazioa","
     Notak_matrizea_Nire_rbf.pkl"),"rb"))
```

```
Notak_matrizea_Nire_rbf_handitua = pickle.load(open(os.path.join(oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_Nire_rbf_handitua.pkl"),"rb"))
# Notak_matrizea_Nire_rbf_handitua[7:,:] = Notak_matrizea_Nire_rbf
# pickle.dump(Notak_matrizea_Nire_rbf_handitua,open(os.path.join(oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_Nire_rbf_handitua.pkl"),"wb"))
Notak_matrizea_Nire_poly = pickle.load(open(os.path.join(oraingo_bidea, "Modeloen_informazioa","Notak_matrizea_Nire_poly.pkl"),"rb"))
print(f"INFORMAZIOA KARGATUTA: \n\n Noten matrizea kernel gaussiarra = \n{Notak_matrizea_Nire_rbf}\n\n Noten matrizea kernel gaussiarra handitua = \n{Notak_matrizea_Nire_rbf_handitua}\n\n Noten matrizea kernel polinomiala = \n{Notak_matrizea_Nire_poly}\n\n")


# # Grafikoa RBF
matrizea = np.flip(Notak_matrizea_Nire_rbf,0)
plt.figure(figsize=(10, 5))
plt.imshow(matrizea, aspect="auto")
plt.xticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j in [-3,-2,-1,0,1,2,3]], fontsize = 17)
plt.xlabel(r"$\sigma$", fontsize = 20)
plt.yticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j in [3,2,1,0,-1,-2,-3]], fontsize = 17)
plt.ylabel(r"$\lambda$", rotation = 0, fontsize = 20)
# plt.title("Nire modelo desberdinen asmatze proportzioa baliozta-multzoan:\nKernel gaussiarra 1000 iteraziorekin klase bakoitzerako")
cbar = plt.colorbar()
cbar.ax.tick_params(labelsize=14)
plt.clim(0,1)
plt.tight_layout(pad = 0.2)
for i in range(matrizea.shape[0]):
    for j in range(matrizea.shape[1]):
        plt.text(j, i, '{:.4f}'.format(matrizea[i, j]), ha='center', va='center', color='white' if matrizea[i, j] < 0.5 else "black", fontsize = 15)
plt.show()


# # Grafikoa RBF handitua
matrizea = np.flip(Notak_matrizea_Nire_rbf_handitua,0)
plt.figure(figsize=(5, 7))
plt.imshow(matrizea, aspect="auto")
plt.xticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j in [-3,-2,-1,0,1,2,3]], fontsize = 14)
plt.xlabel(r"$\sigma$", fontsize = 15)
plt.yticks(np.arange(0, 14, 1), [r'$10^{{{}}}$'.format(j) for j in [-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3][::-1]], fontsize = 14)
plt.ylabel(r"$\lambda$", rotation = 0, fontsize = 15)
```

```
105 # plt.title("Nire modelo desberdinen asmatze proportzioa
        baliozta-multzoan:\nKernel gaussiarra 1000 iteraziorekin
        klase bakoitzerako, bertsio handitua")
106 cbar = plt.colorbar()
107 cbar.ax.tick_params(labelsize=14)
108 plt.clim(0,1)
109 # plt.tight_layout(pad = 0.2)
110 # for i in range(matrizea.shape[0]):
111 #     for j in range(matrizea.shape[1]):
112 #         plt.text(j, i, '{:.2f}'.format(matrizea[i, j]), ha='
        center', va='center', color='white' if matrizea[i, j] < 0.5
         else "black")
113 plt.show()
114
115
116 # # Grafikoa RBF handitua (goiko zatia bakarrik)
117 matrizea = np.flip(Notak_matrizea_Nire_rbf_handitua[0:7,:],0)
118 plt.figure(figsize=(10, 5))
119 plt.imshow(matrizea,aspect="auto")
120 plt.xticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
        in [-3,-2,-1,0,1,2,3]], fontsize = 17)
121 plt.xlabel(r"$\sigma$", fontsize = 20)
122 plt.yticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
        in [-10,-9,-8,-7,-6,-5,-4][::-1]], fontsize = 17)
123 plt.ylabel(r"$\lambda$", rotation = 0, fontsize = 20)
124 # plt.title("Nire modelo desberdinen asmatze proportzioa
        baliozta-multzoan:\nKernel gaussiarra 1000 iteraziorekin
        klase bakoitzerako, bigarren zatia")
125 cbar = plt.colorbar()
126 cbar.ax.tick_params(labelsize=14)
127 plt.clim(0,1)
128 plt.tight_layout(pad = 0.2)
129 for i in range(matrizea.shape[0]):
130     for j in range(matrizea.shape[1]):
131         plt.text(j, i, '{:.3f}'.format(matrizea[i, j]), ha='
        center', va='center', color='white' if matrizea[i, j] < 0.5
         else "black", fontsize = 15)
132 plt.show()
133
134
135 # # Grafikoa kernel polinomiala
136 matrizea = np.flip(Notak_matrizea_Nire_poly,0)
137 plt.figure(figsize=(10, 5))
138 plt.imshow(matrizea, aspect= "auto")
139 plt.xticks(np.arange(0, 7, 1), [2,3,4,5,6,7,8], fontsize = 17)
140 plt.xlabel("polinomioaren maila", fontsize = 17)
141 plt.yticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
        in [-3,-2,-1,0,1,2,3][::-1]], fontsize = 17)
142 plt.ylabel(r"$\lambda$", rotation = 0, fontsize = 20)
143 # plt.title("Nire modelo desberdinen asmatze proportzioa
        baliozta-multzoan:\nKernel polinomiala 1000 iteraziorekin
        klase bakoitzerako")
144 cbar = plt.colorbar()
145 cbar.ax.tick_params(labelsize=14)
```

```
146 plt.clim(0,1)
147 plt.tight_layout(pad = 0.2)
148 for i in range(matrizea.shape[0]):
149     for j in range(matrizea.shape[1]):
150         plt.text(j, i, '{:.4f}'.format(matrizea[i, j]), ha='
    center', va='center', color='white' if matrizea[i, j] < 0.5
     else "black", fontsize = 15)
151 plt.show()
152
153
154 # # ----------------------------
155 # # ----------------------------
156 # # ----------------------------
157
158
159
160 # # Entrenamendu asko egiteko aldi berean (joblib):
161 def entrenatu (C,param,kernela,i,j,Nota_matrizea):
162     if kernela == "rbf":
163         if Nota_matrizea[i,j] == 0:
164             print(f"Hasi da ({i},{j}) posizioa, C = {C}, sigma
    = {param}")
165             modeloa = Nire_SGD_kernelekin(koeficient=C,kernel="
    kernel gaussiarra",sigma=param)
166             random.seed(123)
167             modeloa.fit(X_entrenamendu.values,Y_entrenamendu.
    values,iter=1000)
168             nota = modeloa.score(X_test.values,Y_test.values)
169             Nota_matrizea[i][j] = nota
170             with open(os.path.join(oraingo_bidea, "
    Modeloen_informazioa",'Nire_modeloen_notak.txt'), 'a') as
    informazioa:
171                 informazioa.write(f"(i,j) = ({i},{j}), C = {C},
     sigma = {param} --> Nota = {Nota_matrizea[i,j]}\n")
172
173             print(f"Eginda ({i},{j}) posizioaren entrenamendua,
     horrela gelditu da noten matrizea:")
174             print(Nota_matrizea)
175         else:
176             print(f"Modeloa ({i},{j}) posizioan jada entrenatua
     izan da:\nC = {C}, sigma = {param}, nota = {Nota_matrizea[
    i,j]}")
177
178     elif kernela == "poly":
179         if Nota_matrizea[i,j] == 0:
180             print(f"Hasi da ({i},{j}) posizioa, C = {C}, maila
    = {param}")
181
182             modeloa = Nire_SGD_kernelekin(koeficient=C,kernel="
    kernel polinomiala",deg=param)
183             random.seed(123)
184             modeloa.fit(X_entrenamendu.values,Y_entrenamendu.
    values,iter = 1000)
185             nota = modeloa.score(X_test.values,Y_test.values)
```

```
186             Nota_matrizea[i][j] = nota
187
188             with open(os.path.join(oraingo_bidea, "
    Modeloen_informazioa",'Nire_modeloen_notak.txt'), 'a') as
    informazioa:
189                 informazioa.write(f"(i,j) = ({i},{j}), C = {C},
     maila = {param} --> Nota = {Nota_matrizea[i,j]}\n")
190             print(f"Eginda ({i},{j}) posizioaren entrenamendua,
     horrela gelditu da noten matrizea:")
191             print(Nota_matrizea)
192         else:
193             print(f"Modeloa ({i},{j}) posizioan jada entrenatua
     izan da:\nC = {C}, maila = {param}, nota = {Nota_matrizea[
    i,j]}")
194
195
196
197
198 # # ----------------------------
199 # # -----KERNEL GAUSSIARRA------
200 # # ----------------------------
201
202 # # Parametroak:
203 C_parametroak = np.logspace(-3, 3, 7)
204 gamma_parametroak = np.logspace(-3, 3, 7)
205
206 # # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
207 # # bestela informazioa galdu egingo da)
208 for i,C in enumerate(C_parametroak):
209     for j,gamma in enumerate(gamma_parametroak):
210         Notak_matrizea_Nire_rbf = pickle.load(open(os.path.join
    (oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_Nire_rbf.pkl"),"rb"))
211         entrenatu(C,gamma,"rbf",i,j,Notak_matrizea_Nire_rbf)
212         pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_Nire_rbf.pkl"),"wb"))
213
214
215
216 # # -------------------------
217 # #    KERNEL GAUSSIARRA
218 # #    MATRIZE HANDITUA
219 # # -------------------------
220
221 # with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
    ",'Nire_modeloen_notak.txt'), 'a') as informazioa:
222 #     informazioa.write("\n\nNire modelo desberdinak, kernel
    gaussiarra, bertsio handitua\n\n")
223
224 # # Parametroak:
225 C_parametroak = np.logspace(-10, 3, 14)
226 gamma_parametroak = np.logspace(-3, 3, 7)
```

```
227
228 # # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
229 # # bestela informazioa galdu egingo da)
230 for i in  range(len(C_parametroak)-1,-1,-1):
231     C = C_parametroak[i]
232     for j,gamma in enumerate(gamma_parametroak):
233         Notak_matrizea_Nire_rbf_handitua = pickle.load(open(os.
    path.join(oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_Nire_rbf_handitua.pkl"),"rb"))
234         entrenatu(C,gamma,"rbf",i,j,
    Notak_matrizea_Nire_rbf_handitua)
235         pickle.dump(Notak_matrizea_Nire_rbf_handitua,open(os.
    path.join(oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_Nire_rbf_handitua.pkl"),"wb"))
236
237
238
239 # # ----------------------
240 # # --KERNEL POLINOMIALA--
241 # # ----------------------
242
243 # with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
    ",'Nire_modeloen_notak.txt'), 'a') as informazioa:
244 #     informazioa.write("\n\nNire modelo desberdinak, kernel
    polinomiala\n\n")
245
246 # # Parametroak:
247 C_parametroak = np.logspace(-3, 3, 7)
248 maila_desberdinak = [2,3,4,5,6,7,8]
249
250 # # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
251 # # bestela informazioa galdu egingo da)
252 for i,C in enumerate(C_parametroak):
253     for j,d in enumerate(maila_desberdinak):
254         Notak_matrizea_Nire_poly = pickle.load(open(os.path.
    join(oraingo_bidea, "Modeloen_informazioa","
    Notak_matrizea_Nire_poly.pkl"),"rb"))
255         entrenatu(C,d,"poly",i,j,Notak_matrizea_Nire_poly)
256         pickle.dump(Notak_matrizea_Nire_poly,open(os.path.join(
    oraingo_bidea, "Modeloen_informazioa","
    Notak_matrizea_Nire_poly.pkl"),"wb"))
```

D.2.2 Scikit paketeko ereduak

```
1 import sys
2 import os
3
4 oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
5 bide_orokorra = os.path.dirname(oraingo_bidea)
6
7 from sklearn.svm import SVC
8 import numpy as np
```

```
9  import pandas as pd
10 import time
11 import pickle
12 import matplotlib.pyplot as plt
13
14
15
16
17
18 # # ------------------------
19 # # --DATU BASEA INPORTATU--
20 # # ------------------------
21 mnist_test_bidea = os.path.join(bide_orokorra, "Datu_basea\
       mnist_test.csv")
22 mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
       mnist_train.csv")
23
24 entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
25 testeatzeko_datuak = pd.read_csv(mnist_test_bidea)
26 X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
27 Y_entrenamendu = entrenamendu_datuak["label"]
28 X_test = testeatzeko_datuak.iloc[:,1:]
29 Y_test = testeatzeko_datuak["label"]
30 X_entrenamendu = X_entrenamendu / 255
31 X_test = X_test / 255
32 print("Entrenamenduko klaseen agertze proportzioa:")
33 print(Y_entrenamendu.value_counts()/len(Y_entrenamendu))
34 print()
35 print("Testeko klaseen agertze proportzioa:")
36 print(Y_test.value_counts()/len(Y_test))
37 print()
38 # # ------------------------
39 # # ------------------------
40 # # ------------------------
41
42
43
44 # # ------------------------
45 # # -----INFORMAZIOA GORDETZEKO
46 # #  LEKUAK SORTU (LEHENENGO ALDIZ
47 # # EJEKUTATZEAN)-----
48 # # ------------------------
49 # # Hau lehenengo aldiz exekutatzean bakarrik exekutatu,
       bestela informazioa galdu egingo da
50
51 # Notak_matrizea_rbf = np.zeros((7,7))
52 # Notak_matrizea_poly = np.zeros((7,7))
53 # Notak_matrizea_poly_handia = np.zeros((21,19))
54
55 # pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
       oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_rbf.
       pkl"),"wb"))
56 # pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
       oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_poly.
```

```
    pkl"),"wb"))
# pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"wb"))

# with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
    ",'Modeloen_notak.txt'), 'w') as informazioa:
#     informazioa.write("Scikit-Learn modelo desberdinak,
    kernel gaussiarra\n\n")

# # --------------------
# # --------------------
# # --------------------



# # --------------------
# # -------INFORMAZIOA
# # GORDETZEKO LEKUAK IREKI
# (LEHENENGO ITERAZIOAN EZ)
# # --------------------

Notak_matrizea_rbf = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_rbf.
    pkl"),"rb"))
Notak_matrizea_poly = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_poly.
    pkl"),"rb"))
Notak_matrizea_poly_handia = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"rb"))
# Notak_matrizea_poly_handia[7:14,0:7] = Notak_matrizea_poly
# pickle.dump(Notak_matrizea_Nire_rbf,open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"wb"))
print(f"INFORMAZIOA KARGATUTA: \n\n Noten matrizea kernel
    gaussiarra = \n{Notak_matrizea_rbf}\n\n Noten matrizea
    kernel polinomiala = \n{Notak_matrizea_poly}\n\nNoten
    matrizea kernel polinomiala handia = \n{
    Notak_matrizea_poly_handia}\n\n")



# # Grafikoa RBF:
matrizea = np.flip(Notak_matrizea_rbf,0)
plt.figure(figsize=(10, 5))
plt.imshow(matrizea, aspect= "auto")
plt.xticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
    in [-3,-2,-1,0,1,2,3]], fontsize = 17)
plt.xlabel(r"$\gamma$", fontsize = 20)
plt.yticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
    in [-3,-2,-1,0,1,2,3][::-1]], fontsize = 17)
plt.ylabel("C", rotation = 0, fontsize = 20)
# plt.title("Modelo desberdinen asmatze proportzioa baliozta-
```

```
    multzoan:\nKernel gaussiarra")
cbar = plt.colorbar()
cbar.ax.tick_params(labelsize=14)
plt.clim(0,1)
plt.tight_layout(pad = 0.2)
for i in range(matrizea.shape[0]):
    for j in range(matrizea.shape[1]):
        plt.text(j, i, '{:.4f}'.format(matrizea[i, j]), ha='
    center', va='center', color='white' if matrizea[i, j] < 0.5
     else "black", fontsize = 15)
plt.show()


# # Grafikoa poly
matrizea = np.flip(Notak_matrizea_poly,0)
plt.figure(figsize = (10,5))
plt.imshow(matrizea, aspect= "auto")
plt.xticks(np.arange(0, 7, 1), [2,3,4,5,6,7,8], fontsize = 17)
plt.xlabel("polinomioaren maila", fontsize = 17)
plt.yticks(np.arange(0, 7, 1), [r'$10^{{{}}}$'.format(j) for j
    in [-3,-2,-1,0,1,2,3][::-1]], fontsize = 17)
plt.ylabel("C", rotation = 0, fontsize = 20)
# plt.title("Modelo desberdinen asmatze proportzioa baliozta-
    multzoan:\nKernel polinomiala")
cbar = plt.colorbar()
cbar.ax.tick_params(labelsize=14)
plt.clim(0,1)
plt.tight_layout(pad = 0.2)
for i in range(matrizea.shape[0]):
    for j in range(matrizea.shape[1]):
        plt.text(j, i, '{:.4f}'.format(matrizea[i, j]), ha='
    center', va='center', color='white' if matrizea[i, j] < 0.5
     else "black", fontsize = 15)
plt.show()


# # Grafikoa poly handitua
matrizea = np.flip(Notak_matrizea_poly_handia,0)
plt.figure(figsize=(10, 5))
plt.imshow(matrizea, aspect = "auto")
plt.xticks(np.arange(0, 19, 1), [i for i in range(2,21)])
plt.xlabel("polinomioaren maila")
plt.yticks(np.arange(0, 21, 1), [r'$10^{{{}}}$'.format(j) for j
     in range(10,-11,-1)])
plt.ylabel("C", rotation = 0)
plt.title("Modelo desberdinen asmatze proportzioa baliozta-
    multzoan:\nKernel polinomiala: bertsio handia")
plt.colorbar(label='Asmatutako proportzioa')
plt.clim(0,1)
plt.tight_layout(pad = 0.2)
# for i in range(Notak_matrizea_poly_handia.shape[0]):
#     for j in range(Notak_matrizea_poly_handia.shape[1]):
#         plt.text(j, i, '{:.1f}'.format(
    Notak_matrizea_poly_handia[i, j]), ha='center', va='center
```

```
    ', color='white' if Notak_matrizea_poly_handia[i, j] < 0.5
    else "black")
136 plt.show()
137
138 # # ------------------------
139 # # ------------------------
140 # # ------------------------
141
142
143
144 # # Entrenamendu asko egiteko aldi berean (joblib):
145 def entrenatu (C,param,kernela,i,j,Nota_matrizea):
146     if kernela == "rbf":
147         if Nota_matrizea[i,j] == 0:
148             print(f"Hasi da ({i},{j}) posizioa, C = {C}, gamma
    = {param}")
149
150             modeloa = SVC(C=C,kernel="rbf",gamma=param)
151             modeloa.fit(X_entrenamendu,Y_entrenamendu)
152             nota = modeloa.score(X_test,Y_test)
153             Nota_matrizea[i][j] = nota
154             with open(os.path.join(oraingo_bidea, "
    Modeloen_informazioa",'Modeloen_notak.txt'), 'a') as
    informazioa:
155                 informazioa.write(f"(i,j) = ({i},{j}), C = {C},
     gamma = {param} --> Nota = {Nota_matrizea[i,j]}\n")
156
157             print(f"Eginda ({i},{j}) posizioaren entrenamendua,
     horrela gelditu da noten matrizea:")
158             print(Nota_matrizea)
159         else:
160             print(f"Modeloa ({i},{j}) posizioan jada entrenatua
     izan da:\nC = {C}, gamma = {param}, nota = {Nota_matrizea[
    i,j]}")
161
162     elif kernela == "poly":
163         if Nota_matrizea[i,j] == 0:
164             print(f"Hasi da ({i},{j}) posizioa, C = {C}, maila
    = {param}")
165
166             modeloa = SVC(C=C,kernel="poly",degree=param,coef0
    =1)
167             modeloa.fit(X_entrenamendu,Y_entrenamendu)
168             nota = modeloa.score(X_test,Y_test)
169             Nota_matrizea[i][j] = nota
170
171             with open(os.path.join(oraingo_bidea, "
    Modeloen_informazioa",'Modeloen_notak.txt'), 'a') as
    informazioa:
172                 informazioa.write(f"(i,j) = ({i},{j}), C = {C},
     maila = {param} --> Nota = {Nota_matrizea[i,j]}\n")
173             print(f"Eginda ({i},{j}) posizioaren entrenamendua,
     horrela gelditu da noten matrizea:")
174             print(Nota_matrizea)
```

```
175         else:
176             print(f"Modeloa ({i},{j}) posizioan jada entrenatua
     izan da:\nC = {C}, maila = {param}, nota = {Nota_matrizea[
    i,j]}")
177
178
179
180 # # ----------------------
181 # # ---KERNEL GAUSSIARRA--
182 # # ----------------------
183
184 # # Parametroak:
185 C_parametroak = np.logspace(-3, 3, 7)
186 gamma_parametroak = np.logspace(-3, 3, 7)
187
188 # # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
189 # # bestela informazioa galdu egingo da)
190 for i,C in enumerate(C_parametroak):
191     for j,gamma in enumerate(gamma_parametroak):
192         Notak_matrizea_rbf = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_rbf.
    pkl"),"rb"))
193         entrenatu(C,gamma,"rbf",i,j,Notak_matrizea_rbf)
194         pickle.dump(Notak_matrizea_rbf,open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_rbf.
    pkl"),"wb"))
195
196
197
198
199 # # ----------------------
200 # # --KERNEL POLINOMIALA-
201 # # ----------------------
202
203 # with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
    ",'Modeloen_notak.txt'), 'a') as informazioa:
204 #     informazioa.write("\n\nScikit-Learn modelo desberdinak,
    kernel polinomiala\n\n")
205
206 # # Parametroak:
207 C_parametroak = np.logspace(-3, 3, 7)
208 maila_desberdinak = [2,3,4,5,6,7,8]
209
210 # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
211 # bestela informazioa galdu egingo da)
212 for i,C in enumerate(C_parametroak):
213     for j,d in enumerate(maila_desberdinak):
214         Notak_matrizea_poly = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_poly.
    pkl"),"rb"))
215         entrenatu(C,d,"poly",i,j,Notak_matrizea_poly)
216         pickle.dump(Notak_matrizea_poly,open(os.path.join(
```

```
    oraingo_bidea,"Modeloen_informazioa","Notak_matrizea_poly.
    pkl"),"wb"))
217
218
219
220
221 # # -----------------
222 # # KERNEL POLINOMIAL HANDIA-
223 # # -----------------
224
225 # with open(os.path.join(oraingo_bidea, "Modeloen_informazioa
    ",'Modeloen_notak.txt'), 'a') as informazioa:
226 #     informazioa.write("\n\nScikit-Learn modelo desberdinak,
    kernel polinomiala: bertsio handitua\n\n")
227
228 # # Parametroak:
229 C_parametroak = np.logspace(-10, 10, 21)
230 maila_desberdinak = [i for i in range(2,21)]
231
232 # # Entrenatu (entrenamendua geldi daiteke, baina goiko kodea
    komentatu egin behar da berriro hasterakoan entrenatzen,
233 # # bestela informazioa galdu egingo da)
234 for i,C in enumerate(C_parametroak):
235     for j,d in enumerate(maila_desberdinak):
236         Notak_matrizea_poly_handia = pickle.load(open(os.path.
    join(oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"rb"))
237         entrenatu(C,d,"poly",i,j,Notak_matrizea_poly_handia)
238         pickle.dump(Notak_matrizea_poly_handia,open(os.path.
    join(oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"wb"))
239
240
241 # # -------------------------------GRAFIKOA EGIN
    ---------------------------------
242 Notak_matrizea_poly_handia = pickle.load(open(os.path.join(
    oraingo_bidea,"Modeloen_informazioa","
    Notak_matrizea_poly_handia.pkl"),"rb"))
243 plt.imshow(Notak_matrizea_poly_handia)
244 plt.xticks(np.arange(0, 19, 1), [i for i in range(2,21)])
245 plt.xlabel("polinomioaren maila")
246 plt.yticks(np.arange(0, 21, 1), [r'$10^{{{}}}$'.format(j) for j
     in range(-10,11)])
247 plt.ylabel("C", rotation = 0)
248 plt.title("Modelo desberdinen asmatze proportzioa baliozta-
    multzoan:\nKernel polinomiala: bertsio handia")
249 plt.colorbar(label='Asmatutako proportzioa')
250 plt.clim(0,1)
251 plt.tight_layout(pad = 0.2)
252 # for i in range(Notak_matrizea_poly_handia.shape[0]):
253 #     for j in range(Notak_matrizea_poly_handia.shape[1]):
254 #         plt.text(j, i, '{:.1f}'.format(
    Notak_matrizea_poly_handia[i, j]), ha='center', va='center
    ', color='white' if Notak_matrizea_poly_handia[i, j] < 0.5
```

```
    else "black")
plt.show()


# with open('Modeloen_notak.txt', 'r') as informazioa:
#     info = informazioa.read()
#     print(info)
```

D.3 Eredu hoberenak entrenatzen

Atal honetan, aurreko atalean lortutako emaitzekin, eredu hoberenak entrenatu eta gordetzen dira. Hau da, parametro egokiko ereduak entrenatzen dira eta ordenagailuan gordetzen dira ondoren erabili ahal izateko.

Lanean zehar aipatutako 4 ereduak entrenatzen dira kode honetan, eta haien asmatze-proportzioa kalkulatzen da.

D.3.1 Nire inplementazioko ereduak

```
import sys
import os

oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
bide_orokorra = os.path.dirname(oraingo_bidea)
sys.path.insert(0,os.path.join(bide_orokorra, "Algoritmoak"))


import pandas as pd
import numpy as np
from SGD_soft_SVM_Kernels import Nire_SGD_kernelekin
import time
import pickle
import random

# # --------------------------------
# # ------DATU BASEA INPORTATU-----
# # --------------------------------

mnist_test_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_test.csv")
mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_train.csv")

entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
testeatzeko_datuak = pd.read_csv(mnist_test_bidea)
X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
Y_entrenamendu = entrenamendu_datuak["label"]
X_test = testeatzeko_datuak.iloc[:,1:]
Y_test = testeatzeko_datuak["label"]

# Transformazioa:
X_entrenamendu = X_entrenamendu / 255
X_test = X_test / 255

seed_number = 123


# # -------------------
# # MODELO HOBERENA:
# # KERNEL GAUSSIARRA
```

```
40 # # -------------------
41
42 # # Modeloa sortu:
43 modelo_gauss = Nire_SGD_kernelekin(koeficient=10**(-3), kernel
       = "kernel gaussiarra", sigma = 10)
44 t0 = time.time()
45 random.seed(seed_number)
46 modelo_gauss.fit(X_entrenamendu.values, Y_entrenamendu.values,
       iter = 10000)
47 t1 = time.time()
48 nota = modelo_gauss.score(X_test.values, Y_test.values)
49
50 # # Informazioa gorde:
51 modelo_gauss_info = []
52 modelo_gauss_info.append(t1-t0)
53 modelo_gauss_info.append(nota)
54
55 # # Informazioa inprimatu
56 print(f"Kernel gaussiarreko modeloaren nota: {nota}")
57 print(f"Kernel gaussiarreko modeloaren entrenamendu denbora: {
       modelo_gauss_info[0]}")
58
59 # # Modeloa gorde:
60 pickle.dump(modelo_gauss, open(os.path.join(bide_orokorra, "
       Entrenatutako_modeloak","Nire_modelo_hoberena_gauss.pkl"),
       "wb"))
61 pickle.dump(modelo_gauss_info, open(os.path.join(bide_orokorra,
        "Entrenatutako_modeloak","Nire_modelo_hoberena_gauss_info.
       pkl"), "wb"))
62
63
64
65
66
67
68 # # -------------------
69 # # MODELO HOBERENA:
70 # # KERNEL POLINOMIALA
71 # # -------------------
72
73 # # Modeloa sortu:
74 modelo_poly = Nire_SGD_kernelekin(koeficient=10**(3), kernel =
       "kernel polinomiala", deg = 3)
75 t0 = time.time()
76 random.seed(seed_number)
77 modelo_poly.fit(X_entrenamendu.values, Y_entrenamendu.values,
       iter = 10000)
78 t1 = time.time()
79 nota = modelo_poly.score(X_test.values, Y_test.values)
80
81 # # Informazioa gorde:
82 modelo_poly_info = []
83 modelo_poly_info.append(t1-t0)
84 modelo_poly_info.append(nota)
```

```
85
86 # # Informazioa inprimatu
87 print(f"Kernel polinomialeko modeloaren nota: {nota}")
88 print(f"Kernel polinomialeko modeloaren entrenamendu denbora: {
       modelo_poly_info[0]}")
89
90 # # Modeloa gorde:
91 pickle.dump(modelo_poly, open(os.path.join(bide_orokorra, "
       Entrenatutako_modeloak","Nire_modelo_hoberena_poly.pkl"), "
       wb"))
92 pickle.dump(modelo_poly_info, open(os.path.join(bide_orokorra,
       "Entrenatutako_modeloak","Nire_modelo_hoberena_poly_info.
       pkl"), "wb"))
```

D.3.2 Scikit paketeko ereduak

```
1  import sys
2  import os
3
4  oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
5  bide_orokorra = os.path.dirname(oraingo_bidea)
6
7  from sklearn.svm import SVC
8  import pandas as pd
9  import numpy as np
10 import time
11 import pickle
12 import random
13
14 # # --------------------------------
15 # # ------DATU BASEA INPORTATU-----
16 # # --------------------------------
17 mnist_test_bidea = os.path.join(bide_orokorra, "Datu_basea\
       mnist_test.csv")
18 mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
       mnist_train.csv")
19
20 entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
21 testeatzeko_datuak = pd.read_csv(mnist_test_bidea)
22 X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
23 Y_entrenamendu = entrenamendu_datuak["label"]
24 X_test = testeatzeko_datuak.iloc[:,1:]
25 Y_test = testeatzeko_datuak["label"]
26
27 # Transformazioa:
28 X_entrenamendu = X_entrenamendu / 255
29 X_test = X_test / 255
30
31
32
33 # # -------------------
34 # # MODELO HOBERENA:
35 # # KERNEL GAUSSIARRA
36 # # -------------------
```

```
37
38 # # Modeloa sortu:
39 modelo_gauss = SVC(C = 10, kernel = "rbf", gamma = 10**(-2))
40 t0 = time.time()
41 modelo_gauss.fit(X_entrenamendu.values, Y_entrenamendu.values)
42 t1 = time.time()
43 nota = modelo_gauss.score(X_test.values, Y_test.values)
44
45 # # Informazioa gorde:
46 modelo_gauss_info = []
47 modelo_gauss_info.append(t1-t0)
48 modelo_gauss_info.append(nota)
49
50 # # Informazioa inprimatu
51 print(f"Kernel gaussiarreko modeloaren nota: {nota}")
52 print(f"Kernel gaussiarreko modeloaren entrenamendu denbora: {
       modelo_gauss_info[0]}")
53
54 # # Modeloa gorde:
55 pickle.dump(modelo_gauss, open(os.path.join(bide_orokorra, "
       Entrenatutako_modeloak","Scikit_modelo_hoberena_gauss.pkl")
       , "wb"))
56 pickle.dump(modelo_gauss_info, open(os.path.join(bide_orokorra,
        "Entrenatutako_modeloak","
       Scikit_modelo_hoberena_gauss_info.pkl"), "wb"))
57
58
59
60 # # --------------------
61 # # MODELO HOBERENA:
62 # # KERNEL POLINOMIALA
63 # # --------------------
64
65 # # Modeloa sortu:
66 modelo_poly = SVC(C=10, kernel = "poly", degree = 6, coef0 =1)
67 t0 = time.time()
68 modelo_poly.fit(X_entrenamendu.values, Y_entrenamendu.values)
69 t1 = time.time()
70 nota = modelo_poly.score(X_test.values, Y_test.values)
71
72 # # Informazioa gorde:
73 modelo_poly_info = []
74 modelo_poly_info.append(t1-t0)
75 modelo_poly_info.append(nota)
76
77 # # Informazioa inprimatu
78 print(f"Kernel polinomialeko modeloaren nota: {nota}")
79 print(f"Kernel polinomialeko modeloaren entrenamendu denbora: {
       modelo_poly_info[0]}")
80
81 # # Modeloa gorde:
82 pickle.dump(modelo_poly, open(os.path.join(bide_orokorra, "
       Entrenatutako_modeloak","Scikit_modelo_hoberena_poly.pkl"),
        "wb"))
```

```
pickle.dump(modelo_poly_info, open(os.path.join(bide_orokorra,
    "Entrenatutako_modeloak","Scikit_modelo_hoberena_poly_info.
    pkl"), "wb"))
```

D.3.3 Eredu hoberenen konfusio-matrizea eta gaizki sailkatutako digituak adierazten

Atal honetako kodeak aurreko kodearen bitartez lortutako 4 ereduen konfusio-matrizeak lortzeko eta gaizki klasifikatutako hainbat digitu ikusteko balio du. Hau da, C.2 ataleko grafikoak lortzeko balio du ondorengo kodeak.

```
import sys
import os
import matplotlib.pyplot as plt

oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
bide_orokorra = os.path.dirname(oraingo_bidea)
sys.path.insert(0,os.path.join(bide_orokorra, "Algoritmoak"))


import pandas as pd
import numpy as np
from SGD_soft_SVM_Kernels import Nire_SGD_kernelekin
import time
import pickle
import random
from sklearn.metrics import confusion_matrix

# # --------------------------------
# # ------DATU BASEA INPORTATU-----
# # --------------------------------

mnist_test_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_test.csv")
mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_train.csv")

entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
testeatzeko_datuak = pd.read_csv(mnist_test_bidea)
X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
Y_entrenamendu = entrenamendu_datuak["label"]
X_test = testeatzeko_datuak.iloc[:,1:]
Y_test = testeatzeko_datuak["label"]

# Transformazioa:
X_entrenamendu = X_entrenamendu / 255
X_test = X_test / 255




# # --------------------
# # NIRE MODELOA KERNEL
# #    GAUSSIARRA
```

```
# # --------------------

# # Modeloa inportatu
Nire_gauss = pickle.load(open(os.path.join(bide_orokorra,"Entrenatutako_modeloak","Nire_modelo_hoberena_gauss.pkl"),"rb"))
Nire_gauss_info = pickle.load(open(os.path.join(bide_orokorra,"Entrenatutako_modeloak","Nire_modelo_hoberena_gauss_info.pkl"),"rb"))

# # Informazioa erakutsi:
print("Nire modeloa, kernel gausiarra:")
print(f"Behar izan duen denbora: {Nire_gauss_info[0]}")
print(f"Lortu duen asmatze-proportzioa: {Nire_gauss_info[1]}")
print()

# # Conffusion Matrix egin:
# predikzioak_Nire_gauss = Nire_gauss.predict_anitzkoitza(X_test.values)
# conf_matrix_Nire_gauss = confusion_matrix(Y_test.values, predikzioak_Nire_gauss, labels= np.unique(Y_test.values))
# pickle.dump(conf_matrix_Nire_gauss, open(os.path.join(oraingo_bidea, "Conf_matrix_gordeta", "conf_matrix_Nire_gauss"),"wb"))
conf_matrix_Nire_gauss = pickle.load(open(os.path.join(oraingo_bidea, "Conf_matrix_gordeta", "conf_matrix_Nire_gauss"),"rb"))

matrizea = conf_matrix_Nire_gauss
plt.figure(figsize=(8, 7))
plt.imshow(matrizea, aspect= "equal")
plt.xticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in range(10)], fontsize = 17)
plt.xlabel("Predikzioa", fontsize = 20)
plt.yticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in range(10)], fontsize = 17)
plt.ylabel("Benetako balioa", fontsize = 20)
# plt.title("Modelo desberdinen asmatze proportzioa baliozta-multzoan:\nKernel gaussiarra")
cbar = plt.colorbar()
cbar.ax.tick_params(labelsize=10)
plt.clim(0,1200)
plt.tight_layout(pad = 0.2)
for i in range(matrizea.shape[0]):
    for j in range(matrizea.shape[1]):
        plt.text(j, i, '{:.0f}'.format(matrizea[i, j]), ha='center', va='center', color='white' if matrizea[i, j] < 500 else "black", fontsize = 15)
plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "Nire_gauss_conf_matrix.png"))



# # Gaizki egindako predikzioak bisualizatu:
```

```
random.seed(123)
fig, axs = plt.subplots(2,5, figsize = (12,8))
k = random.randint(0, len(Y_test)-1)
for i in range(2):
    for j in range(5):
        predikzioa =  Nire_gauss.predict(X_test.iloc[k,:])
        zuzena = Y_test.iloc[k]
        while predikzioa == zuzena:
            k = random.randint(0, len(Y_test)-1)
            predikzioa =  Nire_gauss.predict(X_test.iloc[k,:])
            zuzena = Y_test.iloc[k]
        adib = np.array(X_test.iloc[k, :])
        adib = np.reshape(adib, (28,28))
        axs[i,j].imshow(adib, cmap = "gray_r")
        axs[i,j].axis("off")
        axs[i,j].text(0.5,1.2,f"Predikzioa =  {str(predikzioa)}\nZuzena = {str(zuzena)}", fontsize = 15,
    horizontalalignment = "center", verticalalignment = "top",
    transform = axs[i,j].transAxes)
        k += 1

plt.tight_layout(pad = 0)
# plt.show()
plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "Nire_gauss_gaizki_klasifikatuak.png"))



# # --------------------
# # NIRE MODELOA KERNEL
# #    POLINOMIALA
# # --------------------

# # Modeloa inportatu
Nire_poly = pickle.load(open(os.path.join(bide_orokorra,"Entrenatutako_modeloak","Nire_modelo_hoberena_poly.pkl"),"rb"))
Nire_poly_info = pickle.load(open(os.path.join(bide_orokorra,"Entrenatutako_modeloak","Nire_modelo_hoberena_poly_info.pkl"),"rb"))

# # Informazioa erakutsi:

print("Nire modeloa, kernel polinomiala:")
print(f"Behar izan duen denbora: {Nire_poly_info[0]}")
print(f"Lortu duen asmatze-proportzioa: {Nire_poly_info[1]}")
print()


# # Conffusion Matrix egin:
# predikzioak_Nire_poly = Nire_poly.predict_anitzkoitza(X_test.values)
```

```
124 # conf_matrix_Nire_poly = confusion_matrix(Y_test.values,
        predikzioak_Nire_poly, labels= np.unique(Y_test.values))
125 # pickle.dump(conf_matrix_Nire_poly, open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Nire_poly"),"wb"))
126 conf_matrix_Nire_poly = pickle.load(open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Nire_poly"),"rb"))
127
128 matrizea = conf_matrix_Nire_poly
129 plt.figure(figsize=(8, 7))
130 plt.imshow(matrizea, aspect= "equal")
131 plt.xticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
132 plt.xlabel("Predikzioa", fontsize = 20)
133 plt.yticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
134 plt.ylabel("Benetako balioa", fontsize = 20)
135 # plt.title("Modelo desberdinen asmatze proportzioa baliozta-
        multzoan:\nKernel gaussiarra")
136 cbar = plt.colorbar()
137 cbar.ax.tick_params(labelsize=10)
138 plt.clim(0,1200)
139 plt.tight_layout(pad = 0.2)
140 for i in range(matrizea.shape[0]):
141     for j in range(matrizea.shape[1]):
142         plt.text(j, i, '{:.0f}'.format(matrizea[i, j]), ha='
        center', va='center', color='white' if matrizea[i, j] < 500
         else "black", fontsize = 15)
143 plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "
        Nire_poly_conf_matrix.png"))
144
145
146
147 # # Gaizki egindako predikzioak bisualizatu:
148 fig, axs = plt.subplots(2,5, figsize = (12,8))
149 k = random.randint(0, len(Y_test)-1)
150 for i in range(2):
151     for j in range(5):
152         predikzioa =  Nire_poly.predict(X_test.iloc[k,:])
153         zuzena = Y_test.iloc[k]
154         while predikzioa == zuzena:
155             k = random.randint(0, len(Y_test)-1)
156             predikzioa =  Nire_poly.predict(X_test.iloc[k,:])
157             zuzena = Y_test.iloc[k]
158         adib = np.array(X_test.iloc[k, :])
159         adib = np.reshape(adib, (28,28))
160         axs[i,j].imshow(adib, cmap = "gray_r")
161         axs[i,j].axis("off")
162         axs[i,j].text(0.5,1.2,f"Predikzioa =  {str(predikzioa)
        }\nZuzena = {str(zuzena)}", fontsize = 15,
        horizontalalignment = "center", verticalalignment = "top",
        transform = axs[i,j].transAxes)
163         k += 1
```

```
164
165 plt.tight_layout(pad = 0)
166 # plt.show()
167 plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "
        Nire_poly_gaizki_klasifikatuak.png"))
168
169
170
171
172 # # --------------------
173 # # SCIKIT MODELOA KERNEL
174 # #    GAUSSIARRA
175 # # --------------------
176
177 # # Modeloa inportatu
178 Scikit_gauss = pickle.load(open(os.path.join(bide_orokorra,"
        Entrenatutako_modeloak","Scikit_modelo_hoberena_gauss.pkl")
        ,"rb"))
179 Scikit_gauss_info = pickle.load(open(os.path.join(bide_orokorra
        ,"Entrenatutako_modeloak","
        Scikit_modelo_hoberena_gauss_info.pkl"),"rb"))
180
181 # # Informazioa erakutsi:
182 print("Scikit-learn modeloa, kernel gausiarra:")
183 print(f"Behar izan duen denbora: {Scikit_gauss_info[0]}")
184 print(f"Lortu duen asmatze-proportzioa: {Scikit_gauss_info[1]}"
        )
185 print()
186
187 # # Conffusion Matrix egin:
188
189 # predikzioak_Scikit_gauss = Scikit_gauss.predict(X_test.values
        )
190 # conf_matrix_Scikit_gauss = confusion_matrix(Y_test.values,
        predikzioak_Scikit_gauss, labels= np.unique(Y_test.values))
191 # pickle.dump(conf_matrix_Scikit_gauss, open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Scikit_gauss"),"wb"))
192 conf_matrix_Scikit_gauss = pickle.load(open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Scikit_gauss"),"rb"))
193
194 matrizea = conf_matrix_Scikit_gauss
195 plt.figure(figsize=(8, 7))
196 plt.imshow(matrizea, aspect= "equal")
197 plt.xticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
198 plt.xlabel("Predikzioa", fontsize = 20)
199 plt.yticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
200 plt.ylabel("Benetako balioa", fontsize = 20)
201 # plt.title("Modelo desberdinen asmatze proportzioa baliozta-
        multzoan:\nKernel gaussiarra")
202 cbar = plt.colorbar()
```

```
cbar.ax.tick_params(labelsize=10)
plt.clim(0,1200)
plt.tight_layout(pad = 0.2)
for i in range(matrizea.shape[0]):
    for j in range(matrizea.shape[1]):
        plt.text(j, i, '{:.0f}'.format(matrizea[i, j]), ha='center', va='center', color='white' if matrizea[i, j] < 500 else "black", fontsize = 15)
plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "Scikit_gauss_conf_matrix.png"))



# # Gaizki egindako predikzioak bisualizatu:
fig, axs = plt.subplots(2,5, figsize = (12,8))
k = random.randint(0, len(Y_test)-1)
for i in range(2):
    for j in range(5):
        predikzioa =  Scikit_gauss.predict(np.array(X_test.iloc[k,:].tolist()).reshape(1,-1))[0]
        zuzena = Y_test.iloc[k]
        while predikzioa == zuzena:
            k = random.randint(0, len(Y_test)-1)
            predikzioa =  Scikit_gauss.predict(np.array(X_test.iloc[k,:].tolist()).reshape(1,-1))[0]
            zuzena = Y_test.iloc[k]
        adib = np.array(X_test.iloc[k, :])
        adib = np.reshape(adib, (28,28))
        axs[i,j].imshow(adib, cmap = "gray_r")
        axs[i,j].axis("off")
        axs[i,j].text(0.5,1.2,f"Predikzioa =  {str(predikzioa)}\nZuzena = {str(zuzena)}", fontsize = 15, horizontalalignment = "center", verticalalignment = "top", transform = axs[i,j].transAxes)
        k += 1

plt.tight_layout(pad = 0)
# plt.show()
plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "Scikit_gauss_gaizki_klasifikatuak.png"))



# # --------------------
# # SCIKI MODELOA KERNEL
# #    POLINOMIALA
# # --------------------

# # Modeloa inportatu
Scikit_poly = pickle.load(open(os.path.join(bide_orokorra,"Entrenatutako_modeloak","Scikit_modelo_hoberena_poly.pkl"), "rb"))
```

```
245 Scikit_poly_info = pickle.load(open(os.path.join(bide_orokorra,
        "Entrenatutako_modeloak","Scikit_modelo_hoberena_poly_info.
        pkl"),"rb"))
246
247 # # Informazioa erakutsi:
248
249 print("Scikit-learn modeloa, kernel polinomiala:")
250 print(f"Behar izan duen denbora: {Scikit_poly_info[0]}")
251 print(f"Lortu duen asmatze-proportzioa: {Scikit_poly_info[1]}")
252 print()
253
254
255 # # Conffusion Matrix egin:
256
257 # predikzioak_Scikit_poly = Scikit_poly.predict(X_test.values)
258 # conf_matrix_Scikit_poly = confusion_matrix(Y_test.values,
        predikzioak_Scikit_poly, labels= np.unique(Y_test.values))
259 # pickle.dump(conf_matrix_Scikit_poly, open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Scikit_poly"),"wb"))
260 conf_matrix_Scikit_poly = pickle.load(open(os.path.join(
        oraingo_bidea, "Conf_matrix_gordeta", "
        conf_matrix_Scikit_poly"),"rb"))
261
262 matrizea = conf_matrix_Scikit_poly
263 plt.figure(figsize=(8, 7))
264 plt.imshow(matrizea, aspect= "equal")
265 plt.xticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
266 plt.xlabel("Predikzioa", fontsize = 20)
267 plt.yticks(np.arange(0, 10, 1), [r'${{{}}}$'.format(j) for j in
         range(10)], fontsize = 17)
268 plt.ylabel("Benetako balioa", fontsize = 20)
269 # plt.title("Modelo desberdinen asmatze proportzioa baliozta-
        multzoan:\nKernel gaussiarra")
270 cbar = plt.colorbar()
271 cbar.ax.tick_params(labelsize=10)
272 plt.clim(0,1200)
273 plt.tight_layout(pad = 0.2)
274 for i in range(matrizea.shape[0]):
275     for j in range(matrizea.shape[1]):
276         plt.text(j, i, '{:.0f}'.format(matrizea[i, j]), ha='
        center', va='center', color='white' if matrizea[i, j] < 500
         else "black", fontsize = 15)
277 plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "
        Scikit_poly_conf_matrix.png"))
278
279
280
281 # # Gaizki egindako predikzioak bisualizatu:
282 fig, axs = plt.subplots(2,5, figsize = (12,8))
283 k = random.randint(0, len(Y_test)-1)
284 for i in range(2):
285     for j in range(5):
```

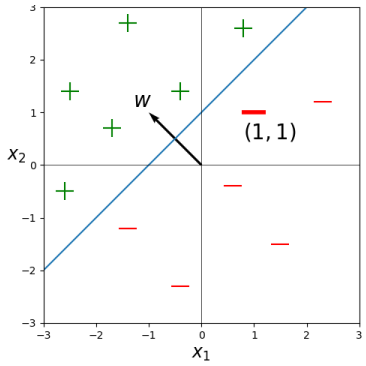
```
286         predikzioa =  Scikit_poly.predict(np.array(X_test.
    values[k]).reshape(1,-1))[0]
287         zuzena = Y_test.iloc[k]
288         while predikzioa == zuzena:
289             k = random.randint(0, len(Y_test)-1)
290             predikzioa =  Scikit_poly.predict(np.array(X_test.
    values[k]).reshape(1,-1))[0]
291             zuzena = Y_test.iloc[k]
292         adib = np.array(X_test.iloc[k, :])
293         adib = np.reshape(adib, (28,28))
294         axs[i,j].imshow(adib, cmap = "gray_r")
295         axs[i,j].axis("off")
296         axs[i,j].text(0.5,1.2,f"Predikzioa =  {str(predikzioa)
    }\nZuzena = {str(zuzena)}", fontsize = 15,
    horizontalalignment = "center", verticalalignment = "top",
    transform = axs[i,j].transAxes)
297         k += 1
298
299 plt.tight_layout(pad = 0)
300 # plt.show()
301 plt.savefig(os.path.join(oraingo_bidea, "Irudiak", "
    Scikit_poly_gaizki_klasifikatuak.png"))
```

D.4 Grafikoen kodea

Azken atal honetan, lan osoan zehar agertzen diren irudiak sortzeko behar den kodea dago.

3.1. irudia

Irudia ondorengoa da:



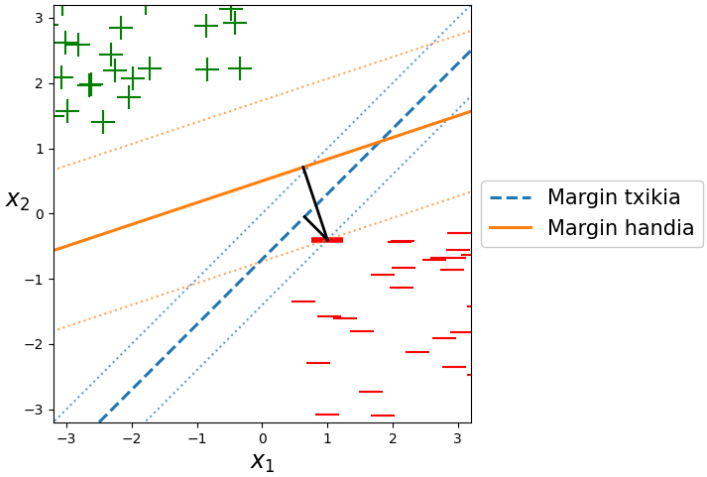
D.1. irudia. Adibide bat $\mathcal{X} = \mathbb{R}^2$ kasurako. $w = (-1, 1)$ dugu eta $b = -1$, orduan $(1, 1)$ puntuaren kasuan: $\text{sign}(h_{w,b}((1, 1))) = \text{sign}(-1 \cdot 1 + 1 \cdot 1 - 1) = \text{sign}(-1) = -1$.

```
import matplotlib.pyplot as plt
import numpy as np

m = 1
b = -1
# Lerroa marrazteko puntuak:
x = np.linspace(-10, 10, 100)
y = m * x - b

# Grafikoa sortu:
plt.plot(x, y, label=r'$-x + y - 1 = 0$')
# plt.title('Espazio zatitzailea '+r'$\vec{w} = (-1,1)$' + '
    eta ' + r'$b = -1$',fontsize = 15)
plt.xlabel(r'$x_1$', fontsize = 17)
plt.ylabel(r'$x_2$',rotation = 0, fontsize = 17)
plt.tight_layout(pad = 0.2)

# plt.grid(True)
plt.xlim([-3,3])
plt.ylim([-3,3])
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.gca().set_aspect('equal', adjustable='box')

# Puntuak gehitu
```

```
plt.scatter([2.3, -0.4, -1.4,0.6,1.5], [1.2,
    -2.3,-1.2,-0.4,-1.5], color='red', marker='_', s = 300)
plt.scatter([-1.7, -0.4, 0.8,-2.5,-2.6, -1.4], [0.7, 1.4,
    2.6,1.4,-0.5, 2.7], color='green', marker='+', s = 300)
plt.scatter([1],[1], color='red', marker='_', s = 500,
    linewidth = 4)
plt.text(0.8,0.5, r'$(1,1)$', color='black', fontsize=20)

# Bektorea gehitu
q = plt.quiver(0,0,-1,1,angles = 'xy',scale_units = 'xy', scale
     = 1, color = 'black')
plt.text(-1.3,1.1, r'$w$', color='black', fontsize=20)

# l =plt.legend(fontsize = 15, loc = 'upper left')
# l.get_frame().set_alpha(1)
plt.show()
```

3.2. irudia

Irudia ondorengoa da:



D.2. irudia. Margin desberdineko bi espazio erdibikari eta haien margina irudikatuta, $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^2 \times \{\pm 1\}$ kasurako.

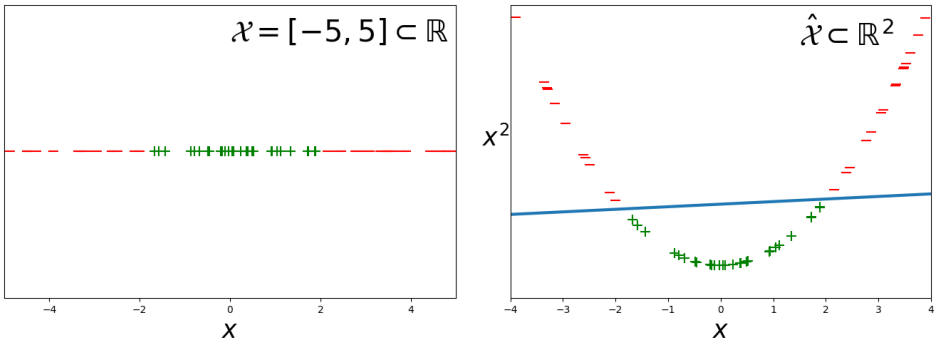
```
import matplotlib.pyplot as plt
import numpy as np
import random

def sortu_puntuak (n_puntu , x_min, x_max, y_min, y_max):
    kx = []
    ky = []
    while len(kx) != n_puntu:
        x = random.uniform(x_min, x_max)
        y = random.uniform(y_min, y_max)
```

```
11          kx.append(x)
12          ky.append(y)
13      return kx, ky
14
15  # Lerroa marrazteko puntuak:
16  x = np.linspace(-10, 10, 100)
17  y1 = x - 0.7
18  y2 = x/3 + 0.5
19
20  y3 = -0.4 + x - 1
21  y4 = x
22
23  y5 = -0.4 + 1/3*(x-1)
24  y6 = 1.81999 + 1/3*(x-0.26)
25  # Grafikoa sortu:
26
27
28  plt.plot(x, y3, label = 'Margin txikia', linestyle = "dotted",
        alpha = 0.7, c = "tab:blue")
29  plt.plot(x, y4, label = 'Margin txikia', linestyle = "dotted",
        alpha = 0.7, c = "tab:blue")
30  plt.plot(x, y5, label = 'Margin txikia', linestyle = "dotted",
        alpha = 0.7, c = "tab:orange")
31  plt.plot(x, y6, label = 'Margin txikia', linestyle = "dotted",
        alpha = 0.7, c = "tab:orange")
32
33
34
35  hiper_1, = plt.plot(x, y1, label='Margin txikia', linewidth =
        2.2, linestyle = "--")
36  hiper_2, = plt.plot(x, y2, label = 'Margin handia', linewidth =
         2.2)
37  plt.xlabel(r'$x_1$', fontsize = 17)
38  plt.ylabel(r'$x_2$',rotation = 0, fontsize = 17)
39  plt.tight_layout(pad = 0.2)
40
41  # plt.grid(True)
42  plt.xlim([-3.2,3.2])
43  plt.ylim([-3.2,3.2])
44  # plt.axhline(0, color='black',linewidth=0.5)
45  # plt.axvline(0, color='black',linewidth=0.5)
46  plt.gca().set_aspect('equal', adjustable='box')
47
48  # Puntuak gehitu
49  random.seed(121)
50
51  n = 15
52  # Negatiboak
53  kx_minus, ky_minus = sortu_puntuak(n+5, 0.6, 3.5, -0.6, -3.5)
54  plt.scatter(kx_minus, ky_minus, color='red', marker='_', s =
        300)
55  kx_minus, ky_minus = sortu_puntuak(n-n//2, 1.8, 3.5, -0.3, -1)
56  plt.scatter(kx_minus, ky_minus, color='red', marker='_', s =
        300)
```

```
57
58 n = 20
59 kx_plus, ky_plus = sortu_puntuak(n, -1.4, -3.5, 1.4, 3.5)
60 plt.scatter(kx_plus, ky_plus, color='green', marker='+', s =
       300)
61 kx_minus, ky_minus = sortu_puntuak(n-n//3 * 2, -0.3, -1,2.2,
       3.5)
62 plt.scatter(kx_minus, ky_minus, color='green', marker='+', s =
       300)
63
64
65 # Distantzien adierazpenak:
66 plt.plot([1, 13/20], [-0.4, -0.05], color='black',linewidth =
       2)
67 plt.plot([1, 63/100], [-0.4, 71/100], color='black',linewidth =
        2)
68 plt.scatter([1],[-0.4], color='red', marker='_', s = 500,
       linewidth = 4)
69
70 l = plt.legend(handles=[hiper_1, hiper_2], fontsize=15, loc='
       center left', bbox_to_anchor=(1, 0.5))
71 l.get_frame().set_alpha(1)
72
73 plt.show()
```

4.1. irudia

Irudiak ondorengoak dira:



D.3. irudia. Lagin ez-lineal baten linealizazioa dimentsio handiagoko domeinu baterako transformazio baten bitartez.

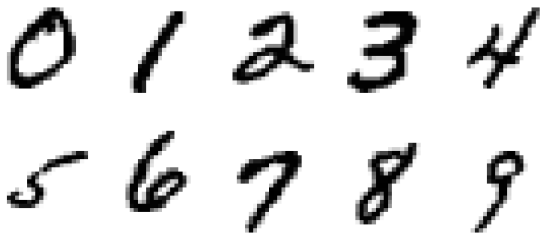
```
1 import random
2 import math
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 random.seed(122)
7
```

```
n1 = 30 # r = 2 zirkunferentzia barruko puntu kopurua
n2 = 40 # r = 2 zirkunferentzia kanpoko puntu kopurua
s = 100 # puntuen tamaina

x1 = []
x2 = []
erro_2 = math.sqrt(2)
# Zirkunferentzia barruko puntuak
while len(x1) < n1:
    x = random.uniform(-2,2)
    if abs(x) <= 2:
        x1.append(x)

# Zirkunferentzia kanpokoak
while len(x2) < n2:
    x = random.uniform(-5,5)
    if abs(x) > 2:
        x2.append(x)
plt.scatter(x1,np.zeros((1,len(x1))), s = s, color = "green",
    marker= "+",label = "")
plt.scatter(x2,np.zeros((1,len(x2))),s = s, color = "red",
    marker= "_",label = "")
plt.xlim([-5,5])
plt.ylim([-.2 ,.2])
# plt.grid()
# plt.gca().set_aspect('equal', adjustable='box')
plt.gca().set_yticklabels([])
plt.tick_params(axis='y', which='both', left=False, right=False
    )
plt.xlabel(r'$x$', fontsize = 25)
# l = plt.legend(["+","-"])
# l.get_frame().set_alpha(1)
plt.tight_layout(pad = 0.2)
plt.text(0, .15, r"$\mathcal{X} = [-5,5] \subset \BbbR$",
    fontsize=30, color='black')
plt.show()




# 2d grafikoa
x1 = np.array(x1)
y1 = x1**2

x2 = np.array(x2)
y2 = x2**2

plt.scatter(x1,y1, s = s, color = "green", marker= "+")
plt.scatter(x2,y2,s = s, color = "red", marker= "_")
# plt.grid()
# plt.gca().set_aspect('equal', adjustable='box')
plt.gca().set_yticklabels([])
```

```
plt.tick_params(axis='y', which='both', left=False, right=False
    )
plt.xlabel(r'$x$', fontsize = 25)
plt.ylabel(r'$x^2$', rotation = 0, fontsize = 25, labelpad= 8)
# l = plt.legend(["+","-"])
# l.get_frame().set_alpha(1)


m = 27 / 130 - 0.05
b = 12227 / 3250
# Lerroa marrazteko puntuak:
x = np.linspace(-5, 5, 100)
y = m * x + b
plt.plot(x,y,linewidth = 3)
# l = plt.legend(["+","-"], loc = "upper center")
# l.get_frame().set_alpha(1)
plt.xlim([-4,4])
plt.ylim([-2,16])
plt.text(1.5, 13.5, r"$\hat{\mathcal{X}} \subset \BbbR^2$",
    fontsize=30, color='black')
plt.tight_layout(pad = 0.2)
plt.show()
```

5.1. irudia

Irudia ondorengoa da:



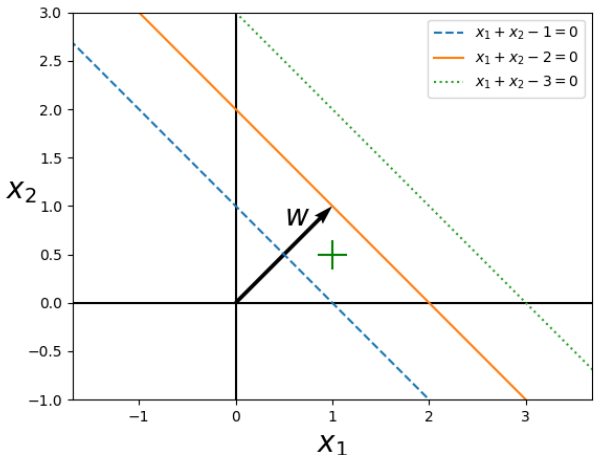
D.4. irudia. MNIST datu-baseko hainbat adibide.

```
import sys
import os

oraingo_bidea = os.path.dirname(os.path.realpath(__file__))
bide_orokorra = os.path.dirname(oraingo_bidea)

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

mnist_train_bidea = os.path.join(bide_orokorra, "Datu_basea\
    mnist_train.csv")
```

```
entrenamendu_datuak = pd.read_csv(mnist_train_bidea)
X_entrenamendu = entrenamendu_datuak.iloc[:,1:]
Y_entrenamendu = entrenamendu_datuak["label"]

fig, axs = plt.subplots(2, 5, figsize=(12, 6))
k = 0
digitua = 0
for i in range(2):
    for j in range(5):
        while Y_entrenamendu[k] != digitua:
            k+=1
        adib = np.array(X_entrenamendu.iloc[k, :])
        adib = np.reshape(adib, (28, 28))
        axs[i, j].imshow(adib, cmap="gray_r")
        axs[i, j].axis('off')
        digitua += 1

plt.tight_layout(pad = 0.2)
plt.show()
```

B.1. irudia

Irudia ondorengoa da:



D.5. irudia. Hiru hiperplanoen irudikapena.

```
import numpy as np
import matplotlib.pyplot as plt


x = np.linspace(-6, 6, 100)

y = 2 - x
```

```
8
9  y1 = 1 - x
10
11 y2 = 3 - x
12 plt.axis('equal')
13
14
15 plt.vlines(x = 0, ymin = -2, ymax = 4, colors = "black")
16 plt.hlines(y = 0, xmin = -2, xmax = 4, colors = "black")
17 # plt.grid()
18
19 plt.plot(x,y1, linestyle = "--", label =  r"$x_1 + x_2 - 1 = 0$
       ")
20
21 plt.plot(x,y,label = r"$x_1 + x_2 - 2 = 0$")
22
23 plt.plot(x,y2, linestyle = "dotted", label =  r"$x_1 + x_2 - 3
       = 0$")
24
25 plt.xlim([-1,3])
26 plt.ylim([-1,3])
27
28 plt.scatter([1],[0.5], c="green", s = 400, marker= "+")
29 plt.quiver(0,0,1,1, angles='xy', scale_units='xy', scale=1)
30
31 plt.xlabel(r"$x_1$",fontsize = 20)
32
33 plt.ylabel(r"$x_2$",fontsize = 20, rotation = 0)
34 plt.text(.5,.8, r"$w$",fontsize = 20)
35
36
37
38
39 plt.legend()
40 plt.show()
```

Bibliografia

[1] Shai Ben-David and Shai Shalev-Shwartz, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014, https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/

[2] Yoshua Bengio, Leon Bottou, Patrick Haffner and Yann Lecun, Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, Volume: 86, Issue: 11, 1998, https://ieeexplore.ieee.org/document/726791

[3] Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006, https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

[4] Jonathan Borwein and Adrian S. Lewis, Convex Analysis and Nonlinear Optimization, Canadian Mathematical Society, 2006, https://carmamaths.org/resources/jon/Preprints/Books/CaNo2/cano2f.pdf

[5] S. Boyd, J. Duchi, M. Pilanci and L. Vandenberghe, Subgradients, Notes for EE364b, Standford University, April 13, 2022, https://stanford.edu/class/ee364b/lectures/subgradients_notes.pdf

[6] Stephen Boyd and Lieven Vandenberghe, Convex optimization, Cambridge University Press, 2004, https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

[7] Matthieu Brucher and David Cournapeau, Scikit-learn Support Vector Machines documentation with Stochastic Gradient Descent, 2007, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDOneClassSVM.html

[8] Matthieu Brucher and David Cournapeau, Scikit-learn Support Vector Machines documentation web-page, 2007, https://scikit-learn.org/stable/modules/svm.html

[9] Christopher Burges, Corinna Cortes, Yann LeCun, The MNIST database of handwritten digits, 1994, http://yann.lecun.com/exdb/mnist/

[10] Chinh-Chung Chang and Chih-Jen Lin, LIBSVM: A Library for Support Vector Machines, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2001, https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

[11] Frederick S. Hillier, Gerald J. Lieberman, Introducción a la investigación de operadores, McGraw Hill, Novena edición, 2010, https://dudasytareas.wordpress.com/wp-content/uploads/2017/05/hillier_lieberman.pdf

[12] Chih-Wei Hsu and Chih-Jen Lin, A Comparison of Methods for Multi-class Support Vector Machines, Department of Computer Science and Information Engineering, National Taiwan University, page 18, 2002, https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf

[13] Eugenio Mijangos, Zenbakizko metodoak MATLAB erabiliz, Euskal Herriko Unibertsitatearen Argitalpen Zerbitzua, 2. edizioa, 2021, https://web-argitalpena.adm.ehu.es/listaproductos.asp?IdProducts=UCPDF215457

[14] Bernhard Schölkopf and Alexander J. Smola, Learning with Kernels, The MIT Press, 2002, https://mitpress.mit.edu/9780262536578/learning-with-kernels/

Indizea