Examen APC

Bloc 1: Introducció

1- Defineix Aprenentatge Computacional i dona exemples d'aplicacions

L'Aprenentatge Computacional és la ciència que fa que els computadors aprenguin i actuïn com ho fan els humans. A més, milloren el seu aprenentatge amb el temps i a mida que van fent proves. La forma de aprendre d'aquests computadors és exposar-los a dades i informació en forma d'observacions i de interacció amb el mon real. Dintre de l'aprenentatge computacional hi ha 4 grans tipus:

- **Aprenentatge Supervisat:** En aquesta classe, els data scientists apliquen els algorismes amb les dades etiquetades i definint les variables les quals es vol que l'algorisme accedeixi per fer les correlacions. L'input i l'output de l'algorisme són especificats.
  Exemples: Classificadors i models regressors.
- **Aprenentatge No Supervisat:** Aquest tipus d'aprenentatge computacional implica algorismes que s'entrenen amb dades sense etiquetar. L'algorisme busca entre les dades alguna connexió útil. Les dades amb les que l'algorisme s'entrena i les prediccions o recomanacions que allibera son predeterminades.
  Exemples: Detecció d'anomalies, Reducció de la dimensionalitat
- **Aprenentatge Semi-Supervisat:** Aquest tipus d'aprenentatge involucra un mix dels dos tipus previs. El data scientist pot ficar la majoria de dades d'entrenament etiquetades però el model és totalment lliure d'explorar les dades pel seu compte i desenvolupar el seu propi enteniment sobre el data set.
  Exemples: Traducció computacional, Etiquetar dades.
- **Aprenentatge Reforçat:** És típicament utilitzat per ensenyar a una maquina com completar un procés amb molts nivells pels quals hi ha unes regles ben definides. Els data scientists programen un algorisme per a completar la tasca i donar senyals positives i negatives sobre com completar la tasca. En gran mesura però, l'algorisme decideix pel seu compte quins nivells va escollint.
  Exemples: Robòtica, Gameplay en videojocs.

Per tal d'escollir el millor model d'aprenentatge computacional, hi ha un procés per facilitar-nos el treball:

1- Ajuntar el problema amb les dades d'entrada que s'haurien de considerar per la solució. En aquest procés es necessita un gran entendriment del problema.
2- Recollir dades, transformar-les i etiquetar-les si es necessari.
3- Escollir quin algorisme s'ha d'utilitzar i testejar-lo per veure com rendeix
4- Canviar petits detalls i anar comprovant l'output per tal d'aconseguir un nivell acceptable d'exactitud.

2- Descriu els diferents tipus d'aprenentatge computacional

Dintre de l'aprenentatge computacional hi ha 4 grans tipus:

- **Aprenentatge Supervisat:** En aquesta clase, els data scientists apliquen els algorismes amb les dades etiquetades i definint les variables les quals es vol que l'algorisme accedeixi per fer les correlacions. L'input i l'output de l'algorisme són especificats. Aquest algorisme va canviant com opera amb les dades i quins pesos atribueix a les dades fins que fa prediccions acurades al donar-li dades noves.

Exemples: Classificadors i models regressors.

- **Aprenentatge No Supervisat:** Aquest tipus d'aprenentatge computacional implica algorismes que s'entrenen amb dades sense etiquetar. L'algorisme busca entre les dades alguna connexió útil. Les dades amb les que l'algorisme s'entrena i les prediccions o recomanacions que allibera son predeterminades.

Exemples: Detecció d'anomalies, Reducció de la dimensionalitat

- **Aprenentatge Semi-Supervisat:** Aquest tipus d'aprenentatge involucra un mix dels dos tipus previs. El data scientist pot ficar la majoria de dades d'entrenament etiquetades però el model és totalment lliure d'explorar les dades pel seu compte i desenvolupar el seu propi enteniment sobre el data set.

Exemples: Traducció computacional, Etiquetar dades.

- **Aprenentatge Reforçat:** És típicament utilitzat per ensenyar a una maquina com completar un procés amb molts nivells pels quals hi ha unes regles ben definides. Els data scientists programen un algorisme per a completar la tasca i donar senyals positives i negatives sobre com completar la tasca. En gran mesura però, l'algorisme decideix pel seu compte quins nivells va escollint.

Exemples: Robòtica, Gameplay en videojocs.

3- Defineix i diferencia entre Classificació i Regressió

- **Classificació:** S'encarrega de predir una etiqueta discreta Y. Classificació es el problema d'assignar noves observacions a la classe que s'hi assembli més, basada en el model de classificació construït a partir del model amb les dades d'entrenament.
  L'exactitud de les classificacions dependrà de l'efectivitat de l'algorisme que s'escull, de com s'aplica i de quant valuoses son les dades que s'han utilitzat per entrenar el model. Dintre dels classificadors existeixen 2 tipus. El primer seria el **regressor logístic.** Aquest model té com a sortida la probabilitat d'una variable categòrica objectiu d'una classe especifica. Normalment és de 0 o 1 (binaria), però es pot utilitzar per classificar entre moltes categories.
  Després vindrien els **Vectors de Suport (SVM).** Soluciona el mateix problema que la regressió logística (classificació amb dos classes), però l'algorisme es presenta en forma geomètrica. Els SVM es basen en definir línies geomètriques que defineixen la frontera entre una classe o una altra. Una mida que es fa servir en els SVM és el marge, que és la distancia entre la línia i els punts més pròxims de les dos classes. Els SVM s'encarreguen de maximitzar aquest marge.
- **Regressió:** S'encarrega de predir una variable objectiu continua Y. Permet estimar un valor basat en una dada d'entrada X. En aquest model la variable objectiu és la variable desconeguda que es vol predir, sent continua (no hi ha forats al valor on Y es pot trobar). Variables discretes serien variables que nomes poden prendre valors exactes. La data d'entrada X inclou tota la informació rellevant sobre el data set que es pot utilitzar per predir l'objectiu. Aquests atributs s'anomenen característiques (features), que poden ser numèrics o categòrics. A la teva regressió voldràs les màximes observacions d'entrenament com sigui possible dels teus features per així que el teu model estigui ben entrenat per saber la relació entre X i Y. Les dades es separen en train i test, on el train té etiquetes per saber com entrenar-se el model. El test no té aquestes etiquetes, ja que serveixen per predir la variable Y i comparar-lo amb el resultat real d'aquest valor Y. Aquest pas és important ja que el nostre model ha de saber desenvolupar-se en tots els terrenys, sobretot en aquells en els que encara no ha estat. Dintre de la regressió es poden trobar 3 fases: La **regressió linial** s'encarrega de dibuixar la línia predictora on depenent de X estarà la nostra variable de sortida Y. Despres vindria el **Descens de Gradient**, que s'encarrega de buscar la pèrdua mínima en la nostra funció del model fent iteracions, obtenint cada vegada una millor aproximació al resultat. L'**Overfitting** és el resultat de fer massa aproximació al problema conegut, sense pensar en una possible situació nova o imprevista. El nostre model ha entrenat tant unes dades en concret que la línia de regressió que no es podrà adaptar a nous paràmetres. L'**Underfitting** és tot el contrari, hem entrenat massa poc el nostre model i no tindrà uns valors que s'adeqüin als resultats reals.
- **Diferències:**
  - Classificació = Predir una classe discreta / Regressió = Predir una quantitat continua
  - Un classificador pot predir un valor continu però es un forma de probabilitat per l'etiqueta de classe / Un regressor pot predir un valor discret però en forma d'un enter (quantitat)
  - Les prediccions dels classificadors es poden evaluar utilitzant l'exactitud, mentre que els regressors no / Els regressors es poden evaluar amb el RootMSE mentre que els classificadors no

4- Defineix i diferencia entre L'IA simbòlica i la subsimbòlica

Una IA és simbòlica quan es desenvolupa una intel·ligència que es basa en regles i coneixement interpretades. Aquesta interpretació és òbviament humana, i tots ho podem entendre. Aquestes IAs simbòliques tenen un grau d'enteniment molt gran, ja que som nosaltres mateixos qui ho dissenyem i qui ho interpretem, per tant la maquina esta treballant amb dades "humanes". Aquest simbolisme es pot expressar en les pròpies etiquetes que es donen a les variables, a la representació gràfica que es dona a un resultat o a un conjunt d'entrada de dades o a una interfície gràfica amb la que ens ajudem per controlar el programa.

En canvi, una IA subsimbòlica no manipulen representacions simbòliques per tal de trobar una solució als problemes, sinó que fan càlculs segons principis que la IA mateixa ha pogut demostrar per tal de arribar a una solució. Els conceptes estan representats com a un set de nombres, vectors, matrius o tensors. S'adrecen a un problema d'una forma local, tenint en compte un concepte únic i particular del problema a solucionar.

Punts a tenir en compte a l'hora de comparar una IA simbòlica d'una subsimbolica:

- Una IA simbòlica implica que tots els seus nivells estan basats en representacions llegibles simbòliques humanes on s'utilitza la lògica i la cerca per solucionar el problema.
- Un punt fort d'una IA simbòlica es que el procés d'enteniment pot ser fàcilment entenedor. En una representació simbòlica es pot explicar fàcilment com s'ha arribat a una certa conclusió i quines passes raonables s'han pres.
- Un punt molt fluix d'una representació subsimbòlica és que és molt difícil trobar una lògica o una causa de perquè s'ha arribat a un resultat en concret. És un tema delicat ja que redueix de gran forma el seu manteniment o la seva modificació.
- Un punt molt feble d'una representació simbòlica és que al procés d'aprenentatge les regles i el coneixement ha de ser codificat pel programador, cosa que pot ser un problema.
- Alguns sistemes subsimbòlics no poden prendre decisions d'alt risc.
- Les representacions simbòliques han estat limitades al mon acadèmic i als laboratoris universitaris en recerques petites amb els gegants industrials.

Bloc 2.1: Regressió de Dades Numèriques

1- Defineix el Descens del Gradient en un problema de Regressió Lineal

El descens del gradient és l'algorisme més utilitzat en aprenentatge computacional. Dins del descens de gradient s'inclou una funció de cost que es la que volem minimitzar. Aquesta funció de cost s'utilitza per monitoritzar els errors en les prediccions del nostre model.

Aquest error es sol quantificar en MSE (Mean Squared Error)

Dintre del Descens del Gradient tenim una variable anomenada Learning Rate. Aquesta variable ens ajuda a no sobrepassar-nos o no anar massa curts. Si aquest valor és massa petit, el nostre algorisme trigarà molt en trobar el resultat, mentre que si el valor és massa gran pot ser que passem per sobre del mínim que estem buscant i no es tingui en compte.

2- Explica l'efecte del coeficient d'aprenentatge en regressió

El Learning Rate és un parametre dintre del descens de gradient que ens controla quant estem ajustant els pesos. Contra més baix sigui el valor, l'algorisme recorrerà molt més lent la nostra recta. Primerament es pot pensar que millor agafar un valor molt petit, però la realitat és que el nostre algorisme pot no acabar mai si optem per aquest camí.

A més a més, el Learning Rate afecta a com de ràpid el nostre model convergirà amb el mínim que s'està buscant, o dit d'una altra manera, la millor precisió.

Pot arribar un moment que quan mantenim el nostre Learning Rate en un valor fixe no millorem la pèrdua. En aquest moment, és bona opció canviar el nostre Learning Rate perquè així no ens quedem estancats en el mateix punt sempre.

3- Descriu en què consisteix la regressió multivariada/polinomial i en què es diferencia de la regressió lineal univariada

Aquest tipus de Regressió s'utilitza quan s'estudia la possible relació entre varies variables independents i una altra dependent.

La Regressió Lineal Múltiple ha de tenir certes condicions:

- Variable dependent (resultat) ha de ser escalar (numèrica) o ordinal de més de 5 categories que tinguin un cert ordre intern
- Variables independents (explicacions) han de ser 1-0 o bé tenir més de 5 categories
- Les variables independents no poden estar altament correlacionades entre elles.
- Les relacions entre variables independents i variable dependent han de ser lineals

La regressió múltiple s'utilitza en la identificació de variables explicatives, descartant aquelles variables que no tenen influencia en la resposta, tenint uns valors més nets i útils. Una altra aplicació pot ser la detecció d'interaccions entre variables independents que afecten la dependent. Per últim, també poden servir per identificar les variables confuses, ja que són molt difícils de trobar però afecten molt al problema, i si es poden identificar d'una manera poc costosa ajudarà molt a trobar una solució optima del problema

En la regressió lineal univariada, la relació entre la variable independent i la dependent és lineal, ja que si es canvia 1 punt de la variable independent la mitjana de la depenent canviarà en un valor específic. En la regressió multivariada tenim més d'1 variable a tenir en compte de cara a obtenir el resultat, així que no hi ha un valor fixe a l'hora de modificar en una unitat una de les variables independents. Aixi doncs, al ser una corba, el canvi del nostre resultat final no depèn de quant canvies alguna de les variables independents, sinó d'on la canvies dintre de l'espai de cerca (NO és un valor constant).

Bloc 2.2: Regularitzador i Recomanadors

1- Defineix què és l'overfitting i l'underfitting en regressió.

Dintre del model predictor tenim 2 variables importants que ens ajuden a que el nostre model sigui el més correcte possible. Aquests són el Bias i la Variància.

Contra més complex és el nostre model, el nostre bias va baixant però al variància va pujant. Contra més paràmetres són afegits al model, la complexitat puja i la variància es converteix en la preocupació més gran mentre que la bias baixa sensiblement.

La variància defineix la desviació de les prediccions. Si el nostre resultat predictor canvia molt entre canvis de train set

s, la nostra variància puja. És molt mala senyal ja que volem que la predicció sigui el més universal possible, donant-nos el màxim % d'encert sigui quin sigui el nostre ambient.

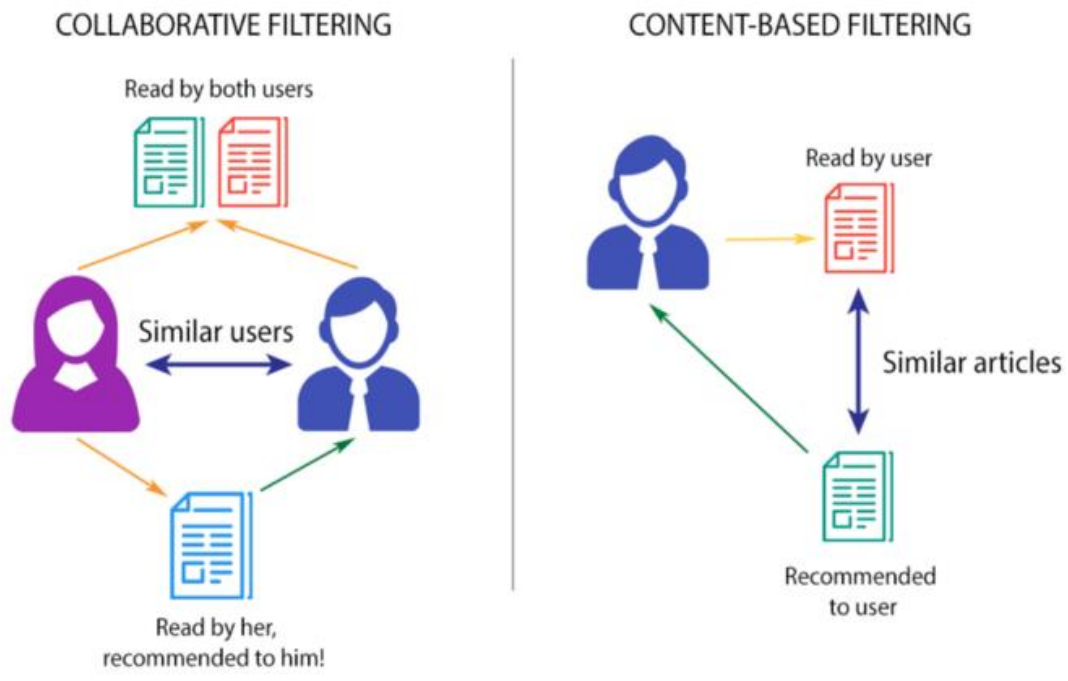
El bias ens mostra quant a prop es troba el model predictor del nostre trainset despres de predir el valor. Com més gran és el nostre bias més ràpid i millor entendrà però menys flexible serà. Un bias molt alt perd totalment la capacitat de predicció.

- **Overfitting:** És l'efecte d'entrenar el nostre model amb massa precisió sobre el trainset. Aquest model sempre tindrà mals resultats al test perquè s'ha sobreexposat al nostre train i ha perdut tota la capacitat de predicció. Alta Variancia, com que el nostre model és massa complex, la variància està en nivells molt alts i el bias en un molt baix
- **Underfitting:** El nostre predictor no aprèn res ja que s'ha entrenat molt poc. S'hauria de canviar el pes del trainset o canviar els valors a aprendre. El nostre model tindrà un resultat molt pobre, ja que no serà capaç de predir amb precisió el resultat. Això comporta que el bias és molt alt, s'hauria de baixar aquesta similitud entre el trainset i el nostre resultat després de predir.

2- Descriu i diferencia entre content-based i collaborative recommenders

- **Content-based:** Aquest metode està basat en una descripció de l'item i un perfil de preferències de l'usuari. Aquets mètodes són idonis per situacions on es coneix la dada d'un item (nom, localització, descripció) però no se'n sap res de l'usuari. Els Content-based tracten les recomanacions com una classificació especifica per cada usuari i aprèn un perfil segons agrada o no agrada basat en les propietats del producte. En aquest sistema les keywords s'utilitzen per descriure els items i el perfil es construeix a partir dels tipus d'item que agraden a l'usuari. L'algorisme compara els items que li van agradar en el passat o que esta explorant actualment.
- **Collaborative:** Aquest mètode construeix prediccions automàtiques depenent dels interessos d'un usuari amb l'afegit de les preferències que es recullen d'altres usuaris. Hi ha 3 tipus de filtres colaboratius:
    - **User-User:** El recomanador més utilitzat. Si a un usuari que és molt similar a tu li agrada un item, aquest item també t'agradarà a tu. Aquest algorisme és eficient quan el nombre de usuaris és menor al nombre d'items.
    - **Item-Item:** Aquest recomanador et junta items similars als que t'han agradat prèviament. Aquest algorisme és eficient quan el nombre d'items és superior al dels usuaris.
    - **User-Item:** Combina els dos anteriors per crear un nou recomanador. L'objectiu és crear relacions poc costoses per tots els items i tots els usuaris. Aquest algorisme és eficient quan hi ha un nombre gran d'usuaris que han votat un nou item abans que aquest pugui ser recomanat.

La gran diferencia entre els dos tipus de recomanadors és que mentre al collaborative el que es té en compte és que un usuari sigui similar a tu (tingui els mateixos gustos que tu) per a recomanar items nous (items que haurà agradat a l'usuari similar), al content-based es fixa en el contingut que tu consumeixes, ja que et recomana contingut que és similar al que ja has indicat que t'agrada o consumit.



3- Descriu com un content-based recommender s'implementa com un regressor

Primerament, cal buscar un o varis descriptors que ens ajudin a identificar el tipus dels items que tenim per tal de classificar-los i recomanar-los. Alguns d'aquests descriptors poden ser el genere de les pel·lícules, l'autor d'un llibre, l'utilitat o la marca d'un objecte, etc.

La implementació d'aquest algorisme comença amb la features matrix. Aquesta matriu conté els descriptors de cada item. Per tal de que el nostre model aprengui sobre el data set, l'entrenem per tal d'obtenir el nostre pes que té cada descriptor. Per això, entrenem el nostre conjunt de dades train (X) per tal que ens doni el resultat esperat (Y), i amb aquest resultat ens calculem el nostre pes (W) i els nostres ratings predits ($\hat{Y}$). La formula bàsica és:

$\hat{Y} = X * W$

El procés exacte per entrenar el nostre model serà:

1- Creem una matriu W random de mida u*n
2- Afegim un bias a W (w0) i l'igualem a 1. Ara la nostra matriu W té la mida u*(n+1)
3- Afegim un intercept item a X (x0) i l'igualem a 1. Ara la primera columna d'X estarà plena de 1's i serà de dimensió m*(n+1)
4- La funció de predicció per aquest problema és: $\hat{Y} = hw(x) = w0x0 + w1x1 + \ldots wnxn = W^T X$
5- Ara creem una funció de cost per la regressió basada en Least Squared Errors per calcular el cost del nostre model
6- Creem una funció per calcular el gradient de la funció de cost
7- Apliquem el descens del gradient per identificar els pesos (W) fins que deixin de canviar
8- Els pesos finals s'utilitzaran per predir els nous ratings basats en la formula $\hat{Y} = X * W$

Bloc 3.1: Regressió Logística i Multi-Classe

1- Descriu en què consisteix la regressió logística

La regressió logística utilitza una equació com a representació, com la regressió lineal. Aquesta funció logística s'anomena també funció de sigmoidea.

Els inputs son combinats linealment amb pesos o valors de coeficient per tal de predir el output. La principal diferencia entre el regressor lineal és que al logístic el valor que es vol predir de l'output és un valor binary en comptes d'un valor numèric.

On Y és l'output predit, b0 és el bias o termini interceptor i b1 és el coeficient per un unic valor d'input (X). Cada columna en el teu input data té associat un coeficient b que ha de ser apres del teu training data. La representació del model que es vol desar en memòria són els coeficients de la equació (b)

Els coeficients (b) de la nostra regressió logística s'han d'estimar a partir del nostre training data utilitzant el "Maximum-likelihood". Els millors coeficients resultaran en un model que faran prediccions d'un valor molt proper al 1 per la classe predeterminada i un valor molt alt al 0 en l'altra classe. La intuïció d'aquest algorisme s'encarrega de trobar els valors dels coeficients que minimitzen l'error de les probabilitats predites pel model d'aquest data. Si el resultat predit és < 0,5 la classe serà la predeterminada i si és >= 0,5 la classe serà l'altra.

2- Defineix el Descens del Gradient en un problema de regressió logística

El Descens de Gradient en un problema de regressió logística s'utilitza per trobar el cost mínim optimitzant la funció l'algorisme de likelihood. Obtenim el cost de la funció logística i es minimitza amb el descens de gradient per tal d'aprendre el model de regressió logística.

A l'hora de trobar els pesos del model, també podem utilitzar el descens del gradient fent pasos en la direcció del gradient per tal de maximitzar el likelihood. A més, s'inclou un coeficient (learning rate) per l'actualització dels pesos.

El resum del descens del gradient és que es vol trobar el punt d'error més baix cambiant els pesos dels coeficients. Es troba el punt mínim recorrent la gràfica en punts aleatoris en sentit contrari i descendent, fins que es troba una part plana a la gràfica, o sigui, que no canvien els pesos sense importar si se'n va cap endavant o cap enrere.

3- Defineix i diferencia entre Bagging i Boosting en el context d'un regressor multi-classe

Ambdos són tècniques que combinen múltiples models entrenats per separat pel mateix problema que es combinen per obtenir millors resultats.

- **Bagging:** Fan la fase d'aprenentatge paral·lelament als altres models sense haver-hi contacte entre ells per després al final combinar-se aplicant un procés que els hi fa la mitja. Es creen models d'aprenentatge que es centrin en subconjunts diferents del mateix conjunt, de tal manera que tinguin resultats similars, però no iguals. D'aquesta manera el model resultant final serà una mitjana dels models independents que resultarà en un model molt més robust.
  Una de les principals avantatges d'aquest procés és que és fàcilment paral·lelitzable
- **Boosting:** Fan la fase d'aprenentatge seqüencialment amb alt índex d'adaptació respecte l'anterior model tractat. El resultat final és l'ultim model creat, ja que té les millores aplicades de tots els models anteriors. Aquest mètode es centra en millorar els errors o les parts més difícils de ponderar del model actual per tenir-ho en compte en els següents. Aquest procés és seqüencial, per tant no es pot paral·lelitzar.

Una de les diferencies més grans és que el bagging s'encarrega de tenir un model amb menys variància que els seus components mentre que el boosting s'encarrega de produir models forts amb el mínim index de bias que els seus components.

Bloc 3.2: Classificacio i SVMs

1- Defineix la Màquina de Vectors de Suport (SVM) i descriu les seves propietats

L'SVM és un dels millors classificadors gràcies a que té moltes possibilitats i es considerat un dels millors paradigmes en l'aprenentatge computacional.

Les SVM es recolzen en els Maximal Margin Classifier. Es tracta de trobar la frontera on es diferencia entre una classe i una altra. La forma més fàcil és trobant la recta dins del pla que delimita aquesta diferenciació, sent la millor recta aquella que maximitza el marge entre els dos punts més propers a la recta de les dos classes.

En alguns casos aquesta recta és impossible de trobar, bé perquè no es necessita una recta, sinó un cercle o una corba, o perquè les classes es distribueixen en un espai de 3 dimensions i el que es necessita és un pla que en delimiti 2 dimensions.

- Com hem parlat abans, una de les seves propietats és que fan servir el Maximal Margin Classifier, que a la vegada minimitzen els errors de classificació i maximitzar el marge respecte la recta
- El Kernel és un altre aspecte important en el nostre SVM, es sol escollir un kernel Gaussià amb un únic paràmetre.
- Els paràmetres del kernel juguen un paper molt important. En el cas del Gaussià s'aplica el paràmetre gamma.
- El paràmetre C del soft margin

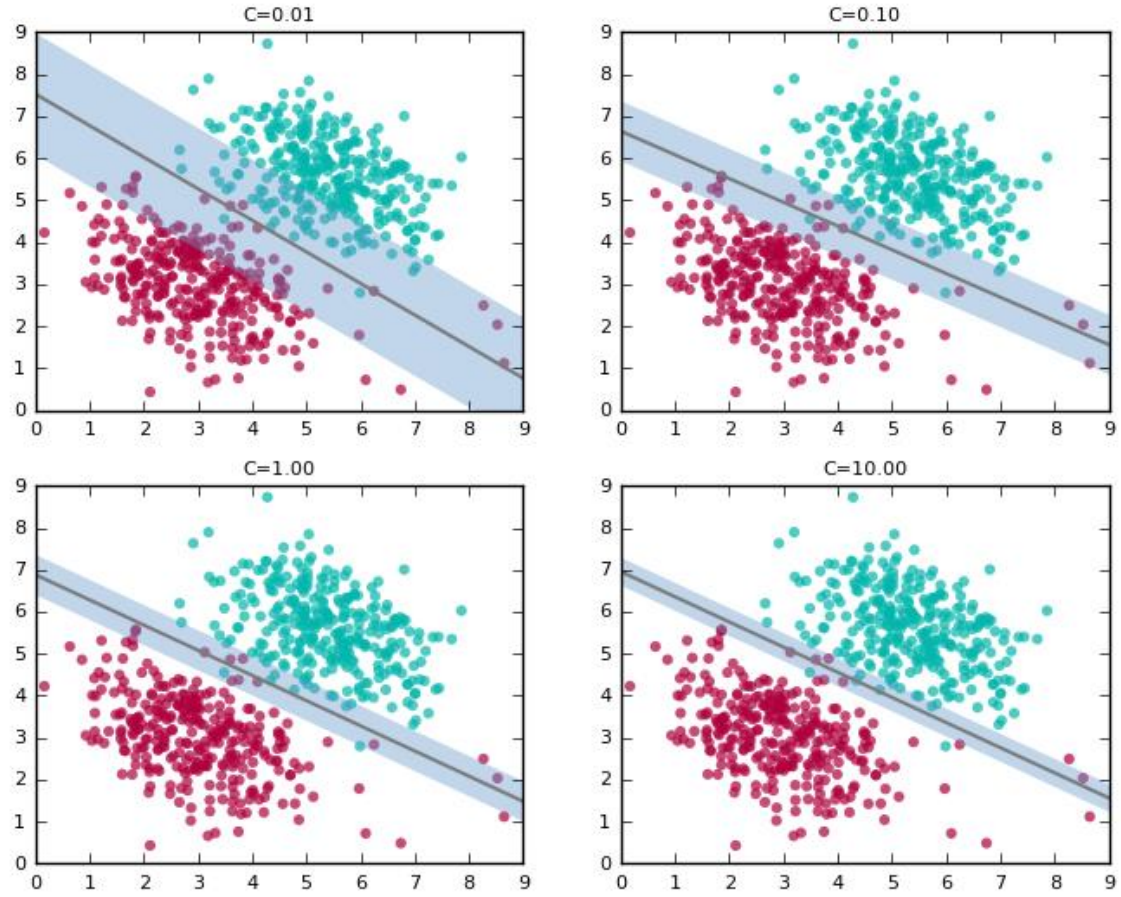
2- Descriu com les variables slack s'utilitzen en les SVM per permetre errors de classificació (soft-margin SVM)

Per tal de "relaxar" les dades que no poden ser classificades (separades) correctament amb un SVM s'introdueix el terme d'slack. Les variables slack mesuren la distancia errònia (diferent classe) dels errors respecte la recta.

Quan hi ha una correcta classificació i el punt estudiat no viola el marge, l'slack serà 0. Quan hi ha una correcta classificació però el punt viola el marge, l'slack estarà entre 0 i 1. Aixo pot passar quan el punt està situat dintre del marge però també dintre del hiperplà. Per ultim, tenim que el punt està situat en el costat incorrecte del marge, llavors l'slack serà més gran que 1.

Aquesta variable slack està controlada per una variable C en l'algorisme del SVM. Controla la compensació entre els errors d'entrenament i els marges rígids, creant un soft-margin que a la vegada que permet errors dintre el model, també els penalitza. Contra més gran ajustem aquest valor de C menys errors permetrem dintre del nostre model.

Contra més valor de C menys soft-margin tenim i s'intenta acomodar aquestes dades a la classe esperada però intenta forçar massa el margin. En canvi, si el posem a un valor molt baix al ser més passiu es registra un margin molt millor però es perd molta precisió en el training data si ho comparem amb un valor alt de C.



3- Descriu l'ús, avantatges i inconvenients dels kernels en les SVM

Normalment quan es vol aplicar algun SVM, a l'hora de delimitar els espais entre les classes s'utilitzen línies rectes, un pla recte o un hiperplà N-dimensional. Malauradament aquests casos són molt idíl·lics i no solen passar en gran proporció. Es poden tractar problemes de SVM amb més de dos variables predictores, corbes no lineals, casos on les dades no poden estar completament separats o que vulguem codificar un classificador de més de dos categories.

La utilització de Kernels ens ajuda a fer representacions en un espai de característiques de major dimensió que augmenta la capacitat computacional. Un exemple de Kernels poden ser el Perceptron, la Gaussiana o la Sigmoid.

Les avantatges és que són fàcils d'implementar per tal de arribar a una solució que no es podria arribar d'una manera normal. També ens assegurem que ens retorna una optimització útil.

El problema més gran en l'ús dels Kernels és que al fer créixer l'espai fas créixer el numero de possibles solucions, pel que el model pot ser molt ambigu. Un altre problema pot ser la generació d'overffiting, o sigui, poca generalització, crear un model massa condicionat pel training data.