IAA - PRAC

Daniel Roca Forcada

Taula de continguts

Exercici 1: reducció de la dimensionalitat

Apartat A

Genereu un conjunt de dades sintètiques amb N=1000 observacions i 4 variables. La primera variable (x1) ha de seguir una distribució normal univariada amb mitjana 0.5 i desviació típica 0.2, és a dir x1~N(0.5, 0.1). La segona variables serà x2~N(4.5, 0.6). La tercera variable serà una combinació lineal de la primera x3 = 3*x1 + 1. La quarta, una combinació no lineal de les dues primeres x4 = 2*x12 + x2. Finalment, construïu una matriu de dades A de tamany Nx4 amb les variables x1, x2, x3, x4.

En el fitxer ***exercici1.py*** tenim implementada la generació de les variables, tant les independents com les combinacions lineals. Aquest fitxer fa servir els mètodes implementats en l'arxiu ***dataProcessing.py i variablesCreator.py***, per a la creació i normalització de variables.

Apartat B

Per cada combinació de variables, representeu les dades gràficament com a núvols de punts bidimensionals (x1 vs x2, etc.). Podeu fer servir la funció scatter de la llibreria matplotlib: http://matplotlib.org/api/pyplot_api.html?highlight=scatter#matplotlib.pyplot.scatter. Comenteu breument les gràfiques obtingudes i la forma del núvol de punts tenint en compte les propietats de les dades.

La representació de les combinacions de les variables queda tal com mostren les següents figures:

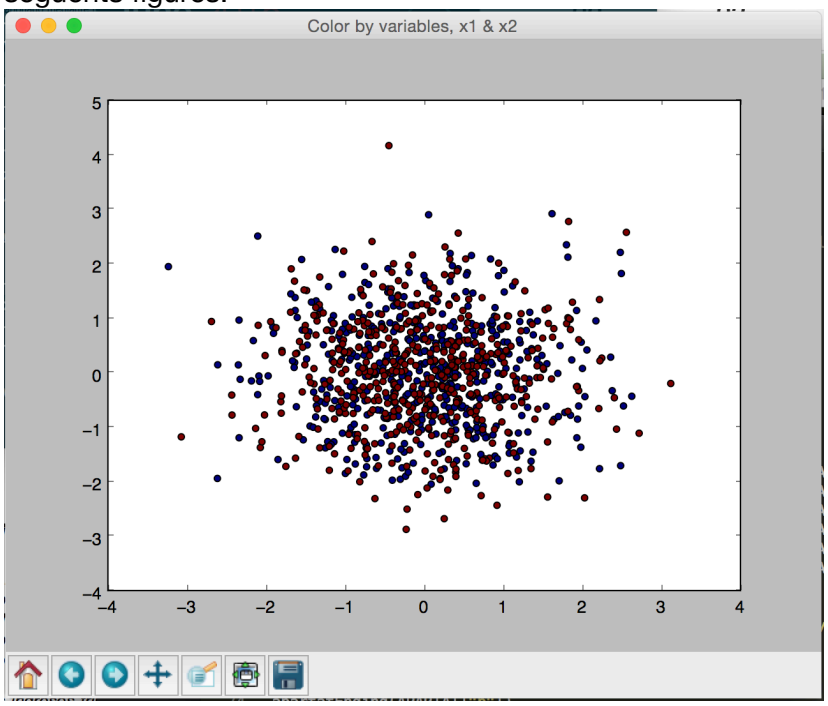

Figura 1. Variables x1 & x2

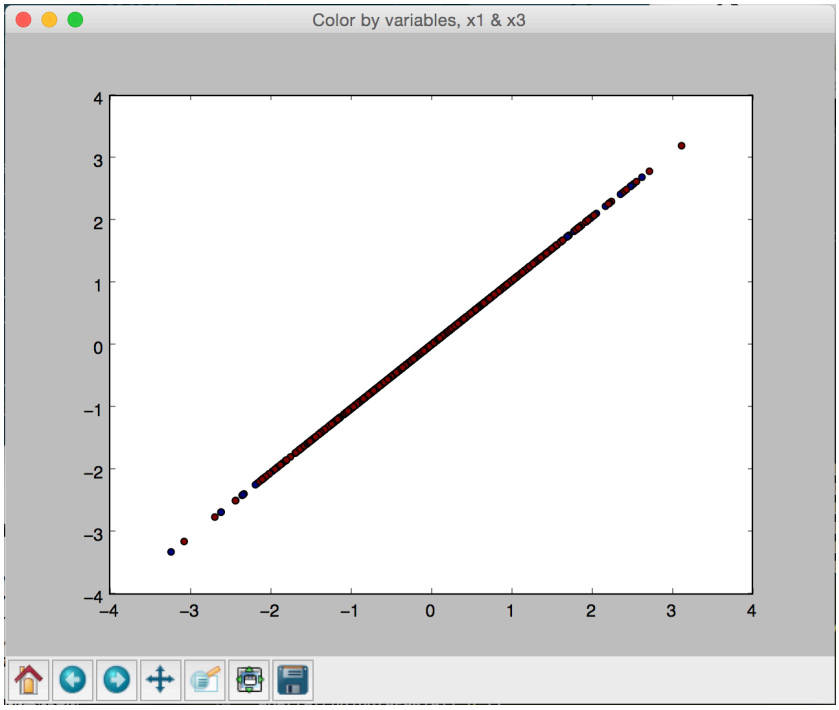

Figura 2. Variables x1 & x3

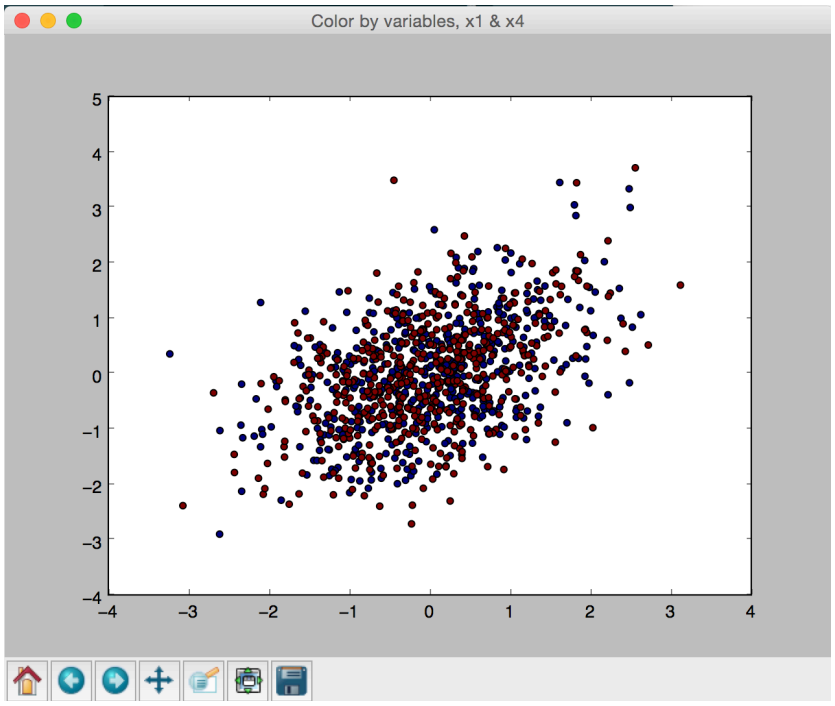

Figura 3. Variables x1 & x4

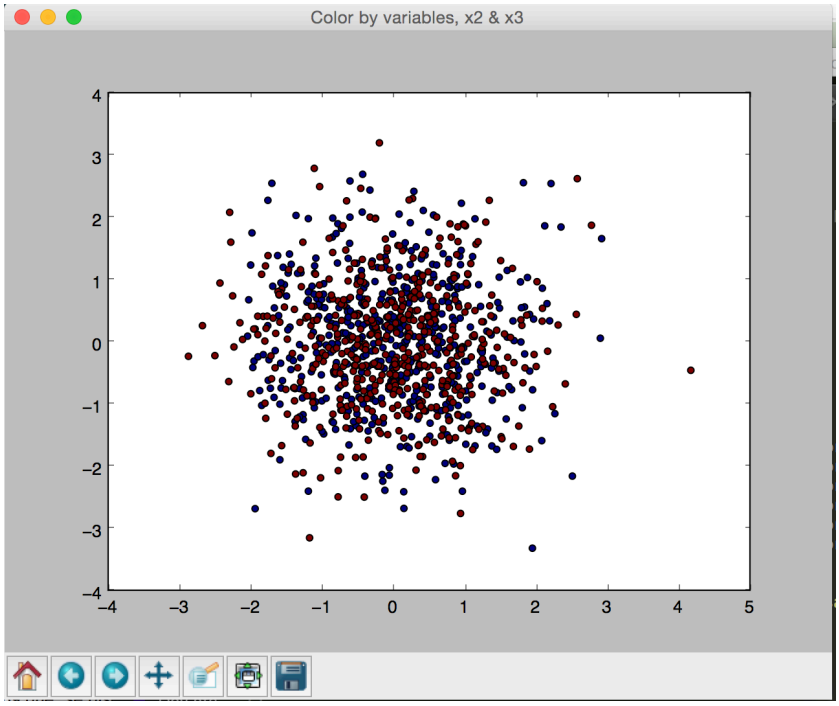

Figura 4. Variables x2 & x3

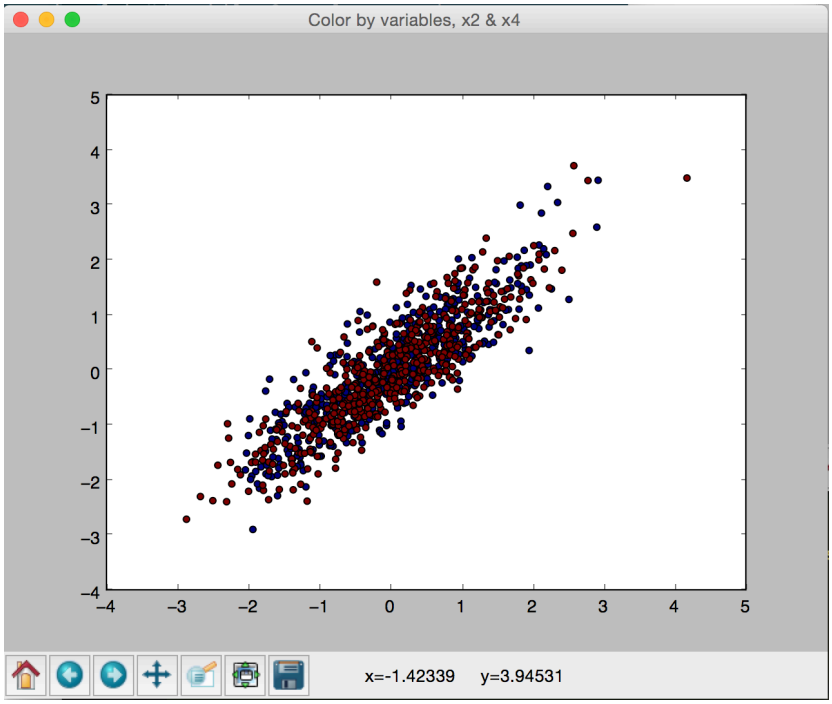

Figura 5. Variables x2 & x4

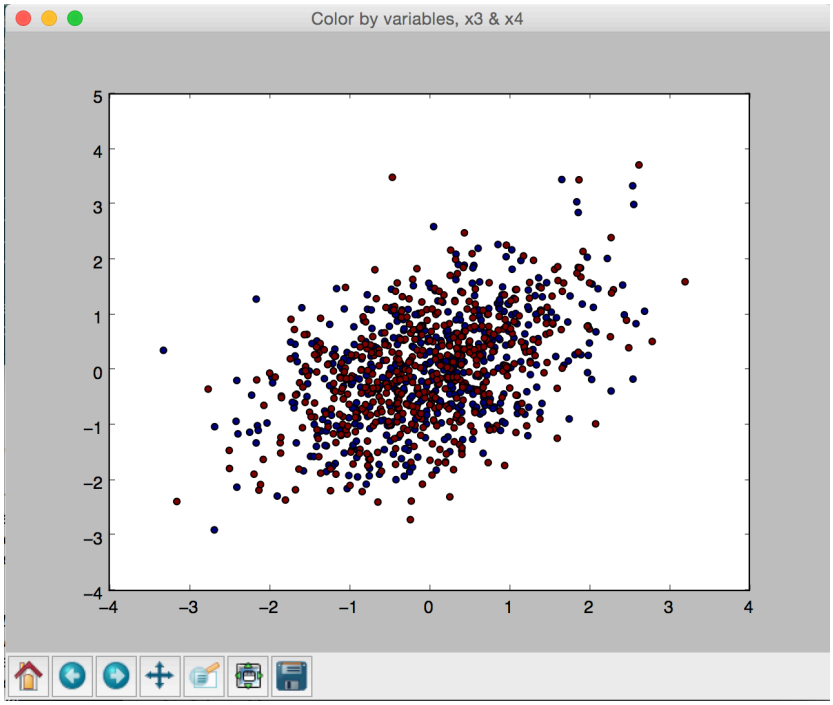

Figura 6. Variables x3 & x4

Com podem apreciar, els gràfics resultants de comparar variables que són una relació lineal amb la variable origen prenen una forma de recta diagonal de 45º. Això es veu clarament en la figura 5 i en la figura 3.
Quan comparem variables que no tenen relació lineal entre elles veiem que els núvols de punts són molt més centrats i no hi ha cap relació aparent entre elles.

Tots els càlculs els tenim implementats en l'arxiu ***exercici1.py*** i fan servir mètodes de l'arxiu ***graphics.py*** per a simplificar la representació.

Apartat C

Apliqueu PCA a les dades anteriors i comenteu els resultats obtinguts tenint en compte les propietats de les dades. La funció PCA de scikit-learn està descrita a http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html . Quantes variables descorrelacionades hi han? Quants components principals són necessaris per explicar un 95% de la variabilitat de les dades?

Al aplicar PCA i calcular els components principals, veiem ràpidament que necessitem un mínim de dos components per a poder explicar un 95% de la variabilitat. Sincerament, en aquest cas concret no hauria fet falta fer l'estudi aplicant PCA i calculant, ja que veiem en seguida que al només tenir dues variables independents (descorrelacionades) només necessitarem 2 components per a explicar el 99-100% de la variabilitat: de les quatre variables

amb les que treballem només n'hi ha dues d'independents, ja que les altres dues són relacions lineals de una o més variables.

Concretament, els resultats obtinguts amb l'execució de l'script són:
`[(0, 0.0), (1, 0.60962139915180813), (2, 0.9998500673950288), (3, 1.0), (4, 1.0)]`
Essent la representació de quin percentatge de variabilitat assolim amb un nombre determinat de components. (primer nombre = nombre de components, segon nombre = percentatge de variabilitat explicada)

Apartat D

Compareu els resultats obtinguts als apartats anteriors amb els que s'obtindrien en cas que les 4 variables estiguessin distribuïdes amb distribucions independents x~N(1, 0.1). Justifiqueu raonadament els resultats obtinguts.

Tornem a realitzar totes les operacions dels apartats anteriors, menys la representació de les gràfiques, i veiem que aquest cop necessitem 4 components per tal d'aconseguir un 95% de la variabilitat. A diferència de l'apartat anterior, en aquest cas tenim 4 variables independents, de manera que com que no hi ha relació entre elles les necessitem a totes per a explicar el 95% de la variabilitat.

Concretament, els resultats obtinguts amb l'execució de l'script són:
`[(0, 0.0), (1, 0.27424689553619747), (2, 0.52346685926596015), (3, 0.7706089070809734), (4, 1.0)]`
Essent la representació de quin percentatge de variabilitat assolim amb un nombre determinat de components. (primer nombre = nombre de components, segon nombre = percentatge de variabilitat explicada)

Exercici 2

Apartat A

Genereu un conjunt de dades bidimensional *A1* amb N=2000 observacions amb una distribució normal amb mitjana [5, -4] i matriu de covariança [2, -1; -1, 2], i un altre conjunt *A2* amb mitjana [1, -3] i matriu de covariança [1, 1.5; 1.5, 3]. Feu servir la següent funció de la llibreria *numpy*: http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.multivariate_normal.html

Generem els conjunts de dades demanats mitjançant la implementació del fitxer ***exercici2.py*** fent servir també l'arxiu ***variablesCreator.py***.

Apartat B

Representeu les dades de tots dos conjunts en forma de núvol de punts en un únic gràfic. Feu servir símbols de color diferent per representar les dades de cada conjunt.

Representem els conjunts de dades mitjançant matplotlib i obtenim els següents gràfics:

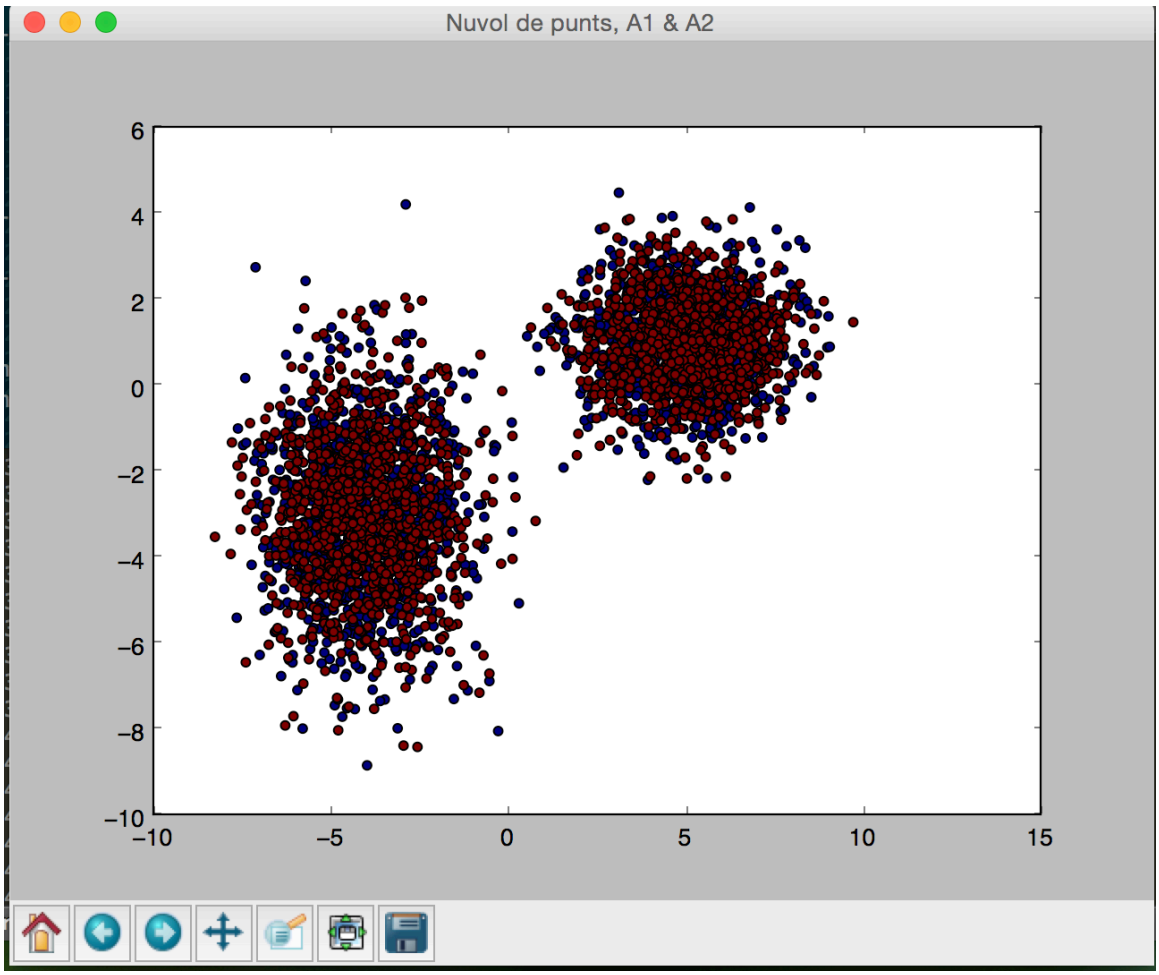

Figura 7. Variables A1 & A2

La distribució dels punts segons els colors no m'acaba de convèncer, tot i que es poden veure dos grans conjunts molt diferenciats, crec que la implementació dels colors diferents per a cada conjunt no ha acabat de ser satisfactòria del tot. A ***exercici2.py*** i ***graphics.py*** trobem tots els mètodes necessaris per a realitzar aquest tasca.

Apartat C

Construïu un conjunt de dades d'entrenament (*training*) amb les 1000 primeres observacions de cada conjunt de dades i un conjunt de prova

(*test*) amb les altres 1000. Feu servir les següents eines de classificació disponibles a les llibreries scikit-learn per dissenyar un sistema de classificació automàtica dels conjunts A1 i A2 a partir de les dades de test. Naïve Bayes: Funció GaussianNB del paquet sklearn.naive_bayes. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html En cada cas, determineu el percentatge d'encerts i errors de classificació sobre el conjunt de validació.

Per a aconseguir aquesta implementació de manera ràpida i fàcil, s'ha decidit fer servir Pandas (http://pandas.pydata.org) i la estructura de dades DataFrame. També ho hauríem pogut fer directament amb arrays o amb numpyarrays però d'aquesta manera sortim una mica del que hem estat fent servir durant tota la pràctica i provem noves eines.

El que fem és afegir un nou valor a les estructures A1 i A2, 1 i 2, respectivament, que actuen com a "classe" dels valors de cadascuna de les estructures, de manera que després podrem entrenar el nostre sistema per a que les pugui classificar. Entrenem el sistema i un cop entrenat, fem un test amb les dades de test (la resta de dades).
Els resultats obtinguts no són massa satisfactoris, pel meu parer, ja que només obtenim una classificació de un 33%.

La implementació la trobem en l'arxiu ***exercici2.py.***

Apartat D

Repetiu l'anàlisi anterior quan la mitjana del primer conjunt de dades *A1* és [2, -4] en comptes de [5, -4]. Comenteu raonadament els resultats obtinguts i justifiqueu les diferències observades tenint en compte les característiques de les dades sintètiques utilitzades.

En aquest apartat fem el mateix que en l'apartat anterior, només amb l'afegit que canviem la generació del conjunt de dades A1. Els resultats obtinguts són exactament els mateixos que en l'apartat anterior, cosa que em fa dubtar de la implementació, doncs un cop hem canviat la mitjana del conjunt A1 s'hauria de veure una diferència en els resultats.

Tenim la implementació en l'arxiu ***exercici2.py.***