Paula Castillo, 184646
Helena Cots, 183951

PRÀCTICA 3: "Classification Methods"

Com en les pràctiques anteriors, el nostre objectiu és aconseguir classificar unes imatges de cares segons les emocions que aquestes estiguin reproduint. Això s'aconsegueix mitjançant una base de dades amb imatges ja classificades, algoritmes de classificació de dades i, en aquesta pràctica, algoritmes de reducció de dimensionalitat de les dades.

Els mètodes de reducció de dimensionalitat que utilitzem són PCA i LDA. A més, utilitzarem tres mètodes de classificació diferents: *Mahalanobis distance*, *Super Vector Machine (SVM)* and *kernel SVM* (els més utilitzats són *gaussian* i *polynomial*).

Per als mètodes de reducció PCA i LDA utilitzem les mateixes línies de codi que vam utilitzar en la funció de *testTemplate* en la pràctica anterior. Amb LDA hem de tenir en compte que abans d'aplicar la reducció fem primer una reducció de les dades amb PCA a 500 dimensions per a que el codi ens funcioni amb agilitat. Reduïm les dades d'entrenament i les de test. Tot i així, quan fem el mètode de classificació per la distància de Mahalanobis el que hem de fer abans de calcular la distància és, tant si hem decidit com no fer la reducció, aplicar una reducció de les dades a 13 dimensions per a que ens pugui fer la comparació entre les dues classes.

Pel que fa als mètodes de classificació, hem implementat el mètode de SVM per a diverses classes. Per a dur-ho a terme, hem fet servir la funció de Matlab *fitcecoc(trainSamples,labelsTrain)* que serveix per a l'aprententatge en multiclasses combinant models SVM binaris. Després de classificar les dades, prediem a quina emoció corresponen les noves imatges d'entrada. Al utilitzar aquest mètode classificador, és molt recomanable fer una prèvia reducció de les dades per tal d'evitar que el programa trigui molt a compilar. El mateix passa quan implementem el mètode kernelSVM. En aquest cas, utilitzem la funció de *templateSVM()* de Matlab que ens retorna una plantilla d'aprenentatge SVM adequada per a l'entrenement de models multiclasse. Posteriorment, tornem a fer servir la funció *fitcecoc(trainSamples,labelsTrain,'Learners',t),* amb els paràmetres *Learners,* que és el que indica que és una classificació binària, i *t* que és la plantilla. Per últim prediem les emocions.

Per a la distància de Mahalanobis, hem utilitzat la funció ja implementada de Matlab *mahal(testSamples,trainEmotion)*. El que fa aquesta funció és calcular-nos la distància dos vectors, o en aquest cas dues matrius a partir de diferents classes. Per exemple, la distància de Mahalanobis es defineix com a una mesura de 'diferències' entre dos vectors x i y amb la mateixa distribució amb matriu de covariança S:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}.$$

Per a poder veure la comparació entre les precisions obtingudes combinant els diferents mètodes creem una taul. Hem decidit que, a l'hora d'escollir les dimensions a reduir amb els algoritmes per mostrar les precisions, n'escollirem 200 (menys quan calculem la distància de Mahalanobis que en necessitarem només 13). D'aquesta forma, no perdem tana informació.

reduce classify	Raw data	PCA (reducing to N dimensions)	LDA (reducing to N dimensions)
Mahalanobis	No podem fer-ho.	0.6774	0.5061
SVM	0.6606	0.7844	0.7003
Kernel SVM (gaussian kernel)	0.7691	0.7676	0.5031
Kernel SVM (polynomial kernel)	0.8211	0.8456	0.5489

Com veiem, obtenim unes millors precisions de la classificació de les dades que en la primera pràctica. Veiem que amb el mètode de Super Vector Machine s'assoleix una precisió del 70% tant si ho hem amb PCA com en LDA. Tot i així, amb 200 dimensions dóna millors resultats la reducció amb PCA per a tots els mètodes classificadors. En aquest cas, classificant amb la distància de Mahalanobis no proveeix gaires bons resultats, per tant, descartaríem fer-la servir amb aquesta base de dades.

Per últim, amb kernel SVM obtenim millors precisions amb el kernel polinomial que amb el gaussià, tot i ser els dos força bons. Amb el *polynomial* tenim la millor *accuracy* aconseguida, combinant aquest mètode de classificació amb una reducció de les dades a 200 dimensions amb LDA resulta amb una precisió del 84.5 %