Index

Objectius

L'objectiu de la pràctica és implementar, validar i compara dos sistemes de classificació automàtica d'imatges de cares utilitzant la base de dades LFW. Una part de la pràctica es dedicarà a l'extracció de característiques rellevants de les imatges mitjançant estratègies diferents i una altra a validar diferents algoritmes de classificació supervisada per classificar les dades- L'avaluació de la pràctica no només valorarà la implementació realitzada sinó els comentaris sobre el funcionament de cada sistema i els motius que expliquen les discrepàncies en el comportament de cada estratègia.

Descripció de la pràctica

En primer lloc cal familiaritzar-se amb l'exemple d'aplicació de sistema de reconeixement de cares 'eigenfaces' descrit a http://scikitlearn.org/0.15/auto_examples/applications/face_recognition.html
S'hauran de tenir en compte les següents consideracions generals en tots els exercicis plantejats en la pràctica:

- Només es consideraran aquells personatges dels que hi hagi un nombre d'observacions (cares) superior a 200 imatges (paràmetres 'min_faces_per_person' de la funció 'fetch_lfw_people').
- Les imatges s'utilitzen sense cap factor de redimensionat (paràmetre resize = 1 en de la funció 'fetch_lfw_people'). Si no s'indica el contrari, les imatges es carregaran en escala de frisos (color = False, per defecte) i en la seva versió funneled (funneled = True, correcció o alineació prèvia de les imatges).

Una vegada que hagueu pogut executar i interpretar correctament els resultats de l'exemple passarem a realitzar els següents exercicis:

Exercici 1: eigenfaces + SVM (3 punts)

a) Executeu el codi de l'exemple eigenfaces + SVM retenint 10, 100 i 400 components principals en la descomposició PCA. En cada cas, utilitzeu la funció 'cross_val_score' per fer una validació creuada amb K = 10 iteracions. Compari el valor mitjà de l'exactitud (accuracy) i comenteu raonadament els resultats obtinguts
http://scikitlearn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Per fer la validació creuada he utilitzat la següent comanda de cross_val_score.

```
print("Cross_Val_Score for ",n_components, " components.",
                    cross_val_score(clf,X_train_pca,y_train, cv=10).mean())
```

Cross_Val_Score for 10 components. 0.805110837438
Cross_Val_Score for 100 components. 0.937315270936
Cross_Val_Score for 400 components. 0.840086206897

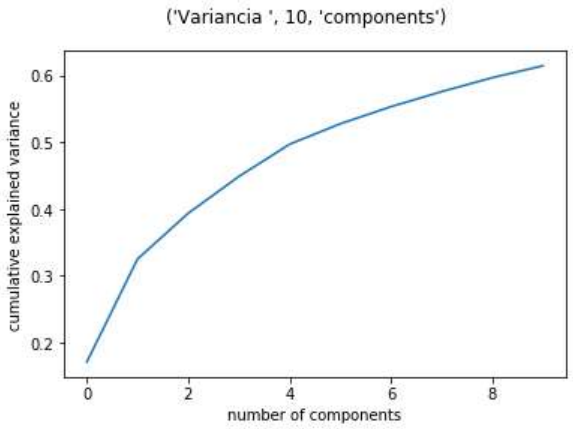
La reducció de dimensionalitat amb PCA ajuda a eliminar aspectes tot els components que no representin informació rellevant com soroll i exemples atípics i així obtenir els components més representatius. Però al fer la validació creuada segons el número de components principals es veu que arriba un punt que per més components que afegim les prediccions no es millora el resultat sinó que empitjora, i això en part pot ser degut a entrenar amb un conjunt d'entrenament que no representi la generalització del model que estem definint.

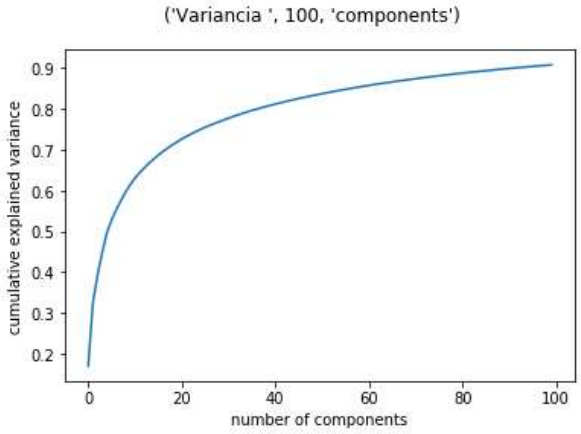
b) Representeu gràficament la variància acumulada explicada per cada un dels components principals utilitzant l'indicador 'explained_variances_ratio_'. Indiqueu de forma aproximada quants components PCA cal retenir per a explicar un 95% de la variabilitat de les dades.

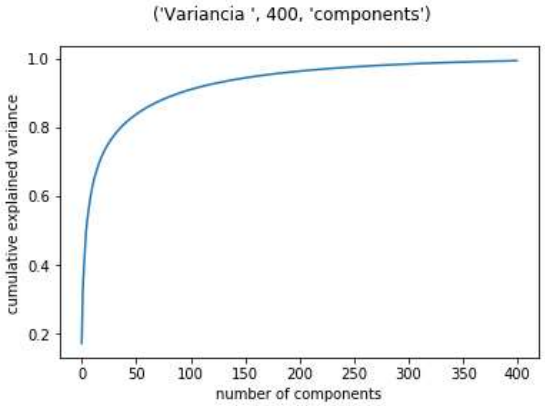
Per calcular quants components representen el 95% de la variabilitat he utilitzat el següent codi que mostra l'acumulació de la variabilitat segons els components ordenats per els que més aporten. Per tant al calcular amb diferents números de components arribaré a un punt que el resultat en ocasions sigui superior a 0.95 que representa el 95% i per tant tindrem el número de components que representa el 95% de la variabilitat de dades que en aquest cas ha sigut de 166 o 167 segons.

```
print("Resultat 1b:")
print(np.cumsum(pca.explained_variance_ratio_))
```

Adjunto gràfiques de la variància acumulada per a 10, 100 i 400 components. Amb aproximadament 166 o 167 components obtenim resultats del 95% de variància acumulada.







c) Obtingui una matriu de confusió en el cas d'utilitzar 400 components PCA i descrigui de forma raonada els encerts i errors del procediment de reconeixement de cares.

Al utilitzar 400 components i aplicar PCA

d) obtenim la següent taula de confusió.

		Classe real	
		positiva	negativa
predicció	positiva	50 (tp)	9 (fp)
	negativa	10 (fn)	123 (tn)

Tant el positius vertaders (tp) com els negatius vertaders (tn) son prediccions correctes, mentre que els falsos positius (fp) i fals negatiu (fn) son classificacions errònies.

Exercici 2: classificació amb CNN (3 punts)

En aquest exercici partim de l'exemple de classificació de l'apartat 6.4 dels material de l'assignatura 'Classificació d'imatges amb xarxes neuronals convolucionals (CNN)'. Com sabeu, aquest exemple utilitza la xarxa neuronal convolucional senzilla tant per extreure característiques de les imatges (primeres 4 capes) com per implementar una xarxa neuronal que classifiqui les cares (últimes dues capes). En aquest exemple la tècnica s'aplicava a la base de dades MNIST (Codi mnistcnn.py, adjunt). Ara es demana el següent:

a) Adapteu el codi de l'exemple per a aplicar-lo a la base de dades LFW amb aquells personatges amb més de 200 cares. Comenteu els canvis necessaris tant el pre-processament de dades com en la definició de la xarxa neuronal convolucional.

He afegit els imports següents per tal descarregar les dades i crear les dades en entrenament i test.

```
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
```

He afegit el següent codi per llegir dades:

```
# Download the data, if not already on disk and load it as numpy arrays
lfw_people = fetch_lfw_people(min_faces_per_person=200, resize=1, color = False, funneled=True)

# introspect the images arrays to find the shapes (for plotting)
n_samples, h, w = lfw_people.images.shape

# for machine learning we use the 2 data directly (as relative pixel
# positions info is ignored by this model)
X = lfw_people.data
n_features = X.shape[1]

# the label to predict is the id of the person
y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]

print("Total dataset size:")
print("n_samples: %d" % n_samples)
print("n_features: %d" % n_features)
```

```
print("n_classes: %d" % n_classes)
```

Ocultat la càrrega de dades anterior i sustituir amb la càrrega de les imatges:

```
#(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

He modificat els paràmetres de shape amb les mides de les fotos, valors h i w. He hagut de fer el canvi a valors reals apart perquè si ho feia com l'exemple mostrava error.

```
#Dimensionar les imatges en vectors
#X_train = X_train.reshape(X_train.shape[0],1,28,28).astype('float32')
#X_test = X_test.reshape(X_test.shape[0],1,28,28).astype('float32')
X_train = X_train.reshape(X_train.shape[0],1,125,94)
X_test = X_test.reshape(X_test.shape[0],1,125,94)
X_train  = X_train.astype('float32')
X_test   = X_test.astype('float32')
```

He modificat a la primera capa (convolucional) els paràmetres input_shape amb les mides de les fotos.

```
#Primera capa ( convolucional ) :
#model.add(Conv2D(32,(5, 5), activation='relu', input_shape=(1, 28, 28)))
model.add(Conv2D(32,(5, 5), activation='relu', input_shape=(1, 125, 94)))
```

b) Comenteu els resultats obtinguts analitzant en detall si el classificador millora en cada iteració (epoch) de la fase d'entrenament. Consulteu també la predicció que realitza el model de les dades de test per valorar de forma crítica els resultats que proporciona aquest sistema de reconeixement de cares. Compari els resultats amb els obtinguts en l'exercici 1. Quins aspectes del problema permeten explicar els resultats? Quines accions serien necessàries per millorar-los?

El classificador manté l'exactitud de 62,50% durant totes les iteracions i ni millora amb els paràmetres utilitzats per entrenar i el conjunt. Es pot modificar el batch_size però ens pot portar a error si l'ultim conjunt sigui massa petit i no representatiu.

Per tal de millorar al crear el conjunt de entrenament i test, utilitzem el paràmetre 'random_state' es pot millorar. Per exemple amb random_state=42 millora fins al 69,27%. També seria recomanable utilitzar un classificador amb el conjunt de dades resultant de la CNN.

He obtingut el següent resultat:

```
Total dataset size:
n_samples: 766
n_features: 11750
n_classes: 2
h: 125
w: 94
Train on 574 samples, validate on 192 samples
Epoch 1/10
15s - loss: 3.3295 - acc: 0.6411 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 2/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 3/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
```

```
Epoch 4/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 5/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 6/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 7/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 8/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 9/10
14s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Epoch 10/10
15s - loss: 4.6052 - acc: 0.7143 - val_loss: 6.0443 - val_acc: 0.6250
Exactitud del modelo: 62.50%
```

Exercici 3: deepfaces + SVM (4 punts)

En aquest exercici es planteja el disseny un sistema de reconeixement de cares en el qual la xarxa neuronal convolucional s'utilitzi com extractor de característiques i tot seguit s'apliqui un classificador SVM com en el primer exercici. Això ens permetrà realitzar una comparació quantitativa entre les característiques basades en PCA o basades en CNN per aquest problema en particular.

a) A partir del model CNN entrenat en l'exercici anterior, defineixi un nou model utilitzant la funció 'Model' de les llibreries Keras per construir una xarxa neuronal convolucional d'extracció d'atributs en què la sortida de la mateixa sigui la sortida de la capa completament connectada https://keras.io/models/model/#model-class-api

Per a això haurà de nomenar la capa en qüestió

```
model.add(Dense(128, activation='relu',name='f'))
```

i tot seguit especificar que la sortida de la nova xarxa sigui aquesta capa

```
model1 = Model(inputs=model.input,outputs=model.get_layer('f').output)
```

Perquè la capa sigui completament connectada, a la capa 6 utilitzo la següent línia, on es posa nom al layer de sortida.

model.add(Dense(128, activation='relu',name='f'))'.

Després de cridar al model creem la sortida de la nova xarxa amb:

model1 = Model(inputs=model.input,outputs=model.get_layer('f').output)

Aquest model ja el podem utilitzar per obtenir el nou conjunt de dades on aplicarem el classificar.

X_train_cnn = model1.predict(X_train)

b) Utilitzeu els atributs extrets per la xarxa convolucional per entrenar una SMV equivalent a la que es va utilitzar en l'apartat a) del primer exercici.

A partir del conjunt extret de la xarxa convolucional entrenem una SMV com a l'exercici 1.

```
# Aplicar els classificadores sobre el nou conjunt X-train_cnn
print("Fitting the classifier to the training set")
param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
              'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }
clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
clf.fit(X_train_cnn,numpy.argmax(y_train,axis=1))
```

c) Feu servir la mateixa mesura de validació que en l'exercici 1 per a comparar els resultats obtinguts amb aquest enfocament 'deepfaces + SVM' amb els de l'exercici 1 'eigenfaces + SVM'. Raoneu la resposta tenint en compte les dades disponibles tant d'entrenament com de validació per a cada personatge a reconèixer.

La mesura amb cross_val_score end dona els següents resultats:

```
print("Cross_Val_Score:",cross_val_score(clf,X_train_cnn,numpy.argmax(y_train,axis=1), cv=10).mean())
```

Apartat1: Cross_Val_Score for 100 components. 0.944305807623
Apartat3: Cross_Val_Score: 0.878201970443

Amb CNN i SVM s'obté un valor inferior a PCA i SVM, això vol dir que amb la configuració actual de CNN, PCA dona millors resultats.

d) Comenteu raonadament els motius pels quals creu que s'expliquen les discrepàncies en reconeixement de cares entre les tres tècniques plantejades en la pràctica (exercicis 1,2 i 3). Proposeu estratègies que permetin millorar el rendiment de cada sistema.

Per als diferents exercicis s'ha obtingut un valor diferent de l'exactitud del model.

Exercici 1. PCA amb SVM: amb 100 components obtenim valors del 93-95%
Exercici 2. Classificació CNN: obtenim el millor de 62,50%.
Exercici 3. CNN amb SVM: obtenim un Cross_Val_Score de 0.874591651543.

PCA al reduir els atributs que millor representa el conjunt de dades, ajuda a obtenir nou conjunt un millor resultat.

Proposa les següents estratègies de millora:

- Per a PCA + SVM, si utilitzem el número de components, podem millorar si modifiquem el número de components.
- La classificació CNN al crear el conjunt de entrenament i test, utilitzant el paràmetre 'random_state' es pot millorar. Per exemple amb random_state=42 millora fins al 69,27%. També es pot utilitzar la estratègia 'Data augmentation' tant per evitar el sobre-entrenament com per augmentar les dades d'entrenament amb petites variacions que es poden donar en un escenari real com per exemple rotar imatges diversos graus o ampliar les imatges.
- Per a CNN + SVM, a part de la estratègia anterior de 'Data augmentation', també podem ampliar els paràmetres com el kernel o els paràmetres emmagatzemats a param_grid per tal de provar més valors per C, gamma o altres paràmetres.