Ciencia de Datos

Práctico N°5
GMM (EM), LDA y PCA

Problema 1: scikit-learn se implementa el modelo de mezcla de gausianas, usando el algoritmo expectation-maximization (EM), en `GaussianMixture`.

a) Revisar en la documentación los pros y contras de `GaussianMixture`.

b) Informarse sobre el origen del dataset Iris flower. Para invocarlo en scikit-learn usar:

```
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.DESCR)
```

Separar un conjunto de `train`, reservando un 25 % para test. Notar que usando la clase (`y`) como valor del parámetro `stratify` en la función `train_test_split()` se separa los datos de forma estratificada manteniendo el balance entre las clases.

c) Implementar el código GMM covariances provisto por scikit-learn para visualizar en 2D los clusters derivados de la aplicación de GMM sobre el dataset de Iris. Utilizar los conjuntos de `train` y `test` del ítem anterior, no los construídos en el tutorial.

Prestar atención a las opciones del parámetro `covariance`:

- `full: cada componente tiene su propia matriz general de covarianza.`
- `tied: todos los componentes tienen la misma matriz de covarianza general.`
- `diag: cada componente tiene su propia matriz de covarianza diagonal.`
- `spherical: cada componente tiene su propia varianza singular.`

d) El tutorial presenta una proyección 2D de las distribuciuones gaussianas 4D, usando las columnas `[0,1]` del dataset, que se corresponden con los atribiutos `sepal length` y `sepal width`. Modificar el código para mostrar la proyección sobre el plano definido por los atributos `petal length` y `petal width`.

Notar que este es un ejercicio de *aprendizaje no-supervisado*, el conjunto de test rotulado con la clase sólo se utiliza para visualizar la estructura de los clústers comparativamente con las clases.

Problema 2: Estudiar las implementaciones de Linear Discriminant Analysis (LDA) y Principal Component Analysis (PCA) provistas por scikit-learn.

a) Implementar ambos análisis sobre el `iris` data set.

b) Calcular la fracción de varianza explicada por las primeras componentes de cada método. Tener en cuenta que en LDA: *n_components no puede ser mayor que min(n_features, n_classes-1)*

c) Discutir las siguientes afirmaciones:

- *PCA identifica la combinación de atributos (componentes principales o direcciones en el espacio de características) que explican la mayor variación en los datos.*
- *LDA intenta identificar atributos que expliquen la mayor variabilidad entre clases.*
- *LDA, a diferencia de PCA, es un método supervisado, que utiliza las etiquetas de clases conocidas.*

Problema 3: Estudiar el dataset sobre cáncer de mama provisto por scikit-learn:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
print(cancer.DESCR)
```

a) Identificar los nombres y números de clases y atributos. ¿Cuántos ejemplos tiene el dataset?
b) ¿Por qué no puede implementarse LDA sobre este dataset?
c) Calcular la fracción de varianza explicada por las primeras 10 componentes de PCA. En base a lo calculado, establecer un criterio de corte para seleccion de atributos.
d) Graficar los datos proyectados sobre el plano definido por las dos primeras componentes. ¿Son suficientes estas dos componentes para separar las clases?
e) Estandarizar los datos usando la función StandardScaler().
Observación: *Estandarizar los datos es un requisito común para muchos estimadores de machine learning; pueden comportarse mal si las características individuales no se asemejan a una distribución normal estandar (es decir, Gaussian con media 0 y varianza unitaria).*
f) Entrenar el modelo de Naïve Bayes reservando un 25 % de ejemplos para test, usando el dataset reescalado. Reportar `accuracy` y las métricas provistas por la función classification_report.
g) Reentrenar el modelo de Naïve Bayes usando la proyección sobre el subespacio definido por las primeras tres componentes PCA. Reportar las métricas del ítem anterior, comparar y expresar una conclusión.

Problema 4: Estudiar el dataset `Labeled Faces in the Wild` provisto por scikit-learn:

```
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)
```

a) Determinar el número de `features`, `samples` y `classes`. ¿Qué son los atributos?
b) Implementar PCA sobre este dataset reteniendo 150 componentes y para acelerar el algoritmo usar la opción `svd_solver='randomized'`.
c) Graficar las primeras componentes (eigenfaces).
d) Graficar la fracción de varianza acumulada en función del número de componentes y escoger el número de componentes que explica aproximadamente el 80 % de la varianza acumulada.
e) Reconstruir las caras con las 150 componentes de PCA y graficar en dos filas las primeras 10 caras originales y reconstruidas para comparar las imágenes.
f) Reservar el 25 % del dataset para testing y clasificar las caras usando Naïve Bayes. Previo al entrenamiento, estandarizar las imágenes restando la media y dividiendo por el desvío estándar. Reportar accuracy y analizar la matriz de confusión. Identificar las confusiones.
g) Con el número de componentes elegido en el ítem **(d)**, para explicar el 80 % de la varianza acumulada, repetir el análisis de los items **(e)** y **(f)**.

Técnicas no paramétricas

Problema 5: Siendo $p(x) \sim U(0, a)$ la distribución uniforme sobre $[0, a]$ y $\phi(x) = \exp(x)$ con $x > 0$, el kernel exponencial,
a) mostrar que la esperanza del estimador basado en ventana de Parzen exponencial, de arista h_n,

resulta

$$E\left[\hat{p}_n(x)\right] = \begin{cases} 0 & x < 0\,, \\ \frac{1}{a}\left(1 - e^{-x/h_n}\right) & 0 \le x \le a\,, \\ \frac{1}{a}\left(e^{a/h_n} - 1\right) e^{-x/h_n} & a \le x\,. \end{cases}$$

b) Graficar esta curva con $a = 1$, y usando los valores $h_n = 1, 1/4$ y $1/16$.
c) ¿Cuán pequeño tiene que ser h_n para obtener menos del 1 % de desvio sobre el 99 % del rango $0 < x < a$?
d) Calcular h_n para la condición anterior si a=1 y graficar $\overline{p}_n(x)$ en el rango $0 \le x \le 0{,}05$.

Problema 6: Estudiar la *regla del vecino más cercano.*
Denotando con $P_n(e)$ la probabilidad de error para la regla del vecino más cercano con muestras de tamaño n y

$$P = \lim_{n\to\infty} p_n(e)\,,$$

probar que se cumplen las desigualdades

$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right),$$

donde P^* es el error de Bayes y c el número de clases.

Problema 7: Estudiar la implementación de $k-$nearest neighbors provista por scikit-learn.
a) Indagar sobre las métricas que pueden utilizarse para el cómputo de distancias.
b) Aplicar $k-$nearest neighbors para clasificar el iris dataset. Analizar las fronteras de decisión en 2D resultantes con $k = 3, 5, 7$.
c) Comparar con el resultado de la clasificación con naïve Bayes. Discutir las matrices de confusión resultantes.

Problema 8: Utilice alguna librería con implementación de Kernel Density Estimation (KDE). Pueden ayudarse con los recursos de la clase IX.
a) Usar este clasificador con Kernel gaussiano con la hand-written digits database, usando el ancho de banda default. Comparar con el valor estimado usando `GridSearchCV`. ¿Qué exactitud (accuracy) se obtiene usando el bandwidth estimado y cuál es la exactitud usando el bandwidth default?
b) Encontrar el ancho de banda óptimo usando `GridSearchCV` para clasificadores con kernels `exponential` y `epanechnikov` para la database digits. Comparar el valor de accuracy obtenido con el bandwidth default.

Regresión Lineal y Logística

Problema 9: Implementar `Perceptron` para clasificar el Breast cancer Wisconsin dataset provisto en `scikit-learn`.
a) Evaluar el modelo imprimiendo un `classification_report` y la matriz de confusión.
b) Comparar con los resultados obtenidos con el modelo Naïve Bayes.

Problema 10: La regresión como clasificador.

Considerar un problema con dos clases: `(0,1)` no correlacionadas. Los datos de la primer clase provienen de una distribución gaussiana con $\mu = 0{,}5$ y $\sigma = 0{,}5$, mientras que los de la segunda clase de una distribución gaussiana con $\mu = 2{,}5$ y $\sigma = 0{,}5$.

a) Generar 50 datos sintéticos para cada clase y graficarlos usando la clase como ordenada.

Ayuda: Para los siguientes dos ítems, estudiar la entrada de `scikit-learn` sobre la función logística.

b) Ajustar los datos con una recta utilizando el modelo `LinearRegression` de `scikit-learn`. Tener en cuenta que los modelos requieren los datos como vectores columnas. Para trasponer un arreglo puede usarse: `x = X.reshape((-1, 1))`, donde `X` son los todos los datos concatenados. Superponer la recta de ajuste en el gráfico con los datos:

```
linear = LinearRegression().fit(x, y)
y_lin = linear.coef_* X + linear.intercept_
```

c) Ajustar los datos con la función logística utilizando el modelo `LogisticRegression` de `scikit-learn`. Superponer la función ajustada en el gráfico anterior:

```
X_test = np.linspace(0, 3, 300)
y_log = expit(X_test * logistic.coef_+ logistic.intercept_).ravel()
```

donde la función `expit` es la función logística o sigmoide, definida por $\text{expit}(x) = 1/(1+\exp(-x))$, y la provee `scipy`: `from scipy.special import expit`.

d) Discutir cómo usar las regresiones para clasificar los datos. ¿Cómo puede asignarse probabilidad a cada clase en la clasificación usando las regresiones?

e) Reconstruir el gráfico anterior usando ahora los mismos datos generados para la clase 0, mientras que los de la segunda clase sintetizarlos usando $\mu = 1{,}5, \sigma = 0{,}5$.

f) Observar cómo se modificaron las regresiones. Expresar una conclusión.

Problema 11: Implementar LogisticRegressionCV para clasificar el Breast cancer dataset.

a) ¿Por qué motivo este modelo utiliza cross validation? Informarse sobre el concepto de regularización para prevenir *overfitting*.

Interpretar los siguientes valores de los parámetros: `cv=5, penalty='l2', solver='liblinear', tol=1e-6, max_iter=int(1e6)`.

b) Evaluar el modelo imprimiendo un `classification_report` y la matriz de confusión. Comparar con los resultados de los modelos implementados anteriormente.

FaMAF 2025