TALLER II
PCA+LDA
Diagnóstico Bayesiano de Diabetes

Brian Keith
briankeithn@gmail.com

Exequiel Fuentes
efulet@gmail.com

21 de julio de 2014

1. Introducción

El objetivo del taller es construir un clasificador que permita diagnosticar la aparición de diabetes dentro de cinco años dados los ocho primeros atributos. El clasificador debe ser implementado utilizando PCA, LDA y un clasificador Gaussiano basado en Naïve Bayes. Se pretende mostrar el funcionamiento de *Principal Component Analysis* (PCA) y *Linear Discriminant Analysis* (LDA) utilizando un clasificador de Naïve Bayes Gaussiano sobre el conjunto de datos destinado al diagnóstico de diabetes. Se usarán las implementaciones de la librería *scikit-learn* para realizar las pruebas y además se desarrollará una versión personalizada del clasificador bayesiano en Python.
Los métodos presentados aquí son técnicas lineales utilizadas ampliamente en el área de aprendizaje automático. PCA es una técnica de reducción de dimensionalidad que tiene por objetivo resumir los datos en sus dimensiones más importantes de forma que se retenga la mayor cantidad de información. LDA es una técnica que entrega una proyección de los datos sobre un subespacio vectorial, permite determinar la clase a la que pertenece un ejemplo. Naïve Bayes es una técnica probabilística que se utiliza para clasificar ejemplos bajo la suposición de independencia condicional.
En particular, se presenta un clasificador que utiliza PCA para reducir la dimensionalidad de los datos, luego aplica LDA para determinar el subespacio discriminante y finalmente se usa Naïve Bayes asumiendo que los datos siguen una distribución normal para clasificar los ejemplos.
Se mostrará gráficamente el efecto de la cantidad de dimensiones utilizadas por PCA y se encontrará la dimensión óptima para los datos utilizados, además se presenta detalladamente la implementación del clasificador bayesiano. El clasificador implementado entrega exactamente los mismos resultados que la implementación de la librería *scikit-learn*.

1.1. Contexto del problema

El término diabetes comprende un grupo heterogéneo de enfermedades que sólo tiene en común la excreción de volúmenes excesivos de orina acompañada de una sed incontrolada. Generalmente se refiere a su subtipo más común, *la diabetes mellitus*, un grupo de trastornos metabólicos caracterizados por la presencia persistente de concentraciones elevadas de glucosa en el plasma sanguíneo.
La diabetes representa un grave problema de salud y constituye una de las causas de mortalidad más comunes en el mundo. El padecimiento de esta enfermedad aumenta el riesgo de desarrollar diversas complicaciones como problemas renales, ceguera, daño a los nervios y enfermedades cardíacas, las cuales afectan la calidad de vida del paciente y en el peor de los casos, conducen a la muerte.
Para facilitar la gestión, el seguimiento y el diagnóstico temprano de pacientes, se utilizan una serie de técnicas; una de ellas son los modelos matemáticos de predicción.
Se propone construir un modelo matemático con la capacidad de predecir si un paciente tiene diabetes o no. Se cuenta con un conjunto de datos que será utilizado para la construcción y validación del modelo matemático.

1.2. Datos utilizados

El conjunto de datos utilizado corresponde a una muestra de 768 pacientes. Se impusieron varias restricciones en estas muestras obtenidas desde una base de datos más grande. Todos los pacientes en esta muestra son mujeres de al menos 21 años de herencia indígena Pima. La tabla resume la información contenida en los datos.

#	**Atributo**	**Descripción**
1	Pregnant	Cantidad de veces que la mujer ha estado embarazada.
2	Plasma Glucose	Concentración de glucosa en la sangre dos horas después de la prueba de tolerancia a la glucosa (mm Hg).
3	Diastolic BP	Presión sanguínea diastólica.
4	Triceps SFT	Espesor de la piel del tríceps (mm).
5	Serum-Insulin	Cantidad de insulina en dos horas (muU/ml).
6	BMI	Índice de masa corporal (Kg / mm)
7	DPF	Antecedentes Familiares.
8	Age	Edad del paciente (año).
9	Class	Aparición de diabetes dentro de cinco años.

Cuadro 1: Detalle de los atributos expuestos en los datos.

Se muestra la distribución de cada clase dentro de la muestra, la clase 0 corresponde a los individuos sanos (sin diabetes) y la clae 1 corresponde a los individuos que presentaron diabetes dentro de 5 años de los examenes realizados.

Clase	**Cantidad de Instancias**
0 (Negativo)	500
1 (Positivo)	268

Cuadro 2: Distriución por Clase

Estos datos serán separados en una proporción del 70 % para entrenamiento y de 30 % para validación.

2. Fundamentos Teóricos

2.1. Principal Component Analysis

En esta sección se explican los fundamentos teóricos de PCA, no se ha implementado manualmente este método, en cambio solamente se ha usado la implementación de la librería *scikit-learn*.
El objetivo es reducir la dimensionalidad a costa de una pequeña pérdida de información. En principio esto se realiza a través de una rotación, es decir, un cambio de coordenadas. Esta rotación se hace de tal manera que los nuevos ejes se alinean con las direcciones que presenta la mayor variabilidad.
Esto se logra mediante la obtención de los autovalores y autovectores de la matriz de covarianzas de los datos. Es posible eliminar las dimensiones con menor variabilidad sin pérdida grande de información.
El algoritmo para aplicar PCA a una matriz X de dimensiones $N \times d$ en una matriz $N \times m$ es el siguiente:

a. Centralizar los datos (substraer la media).

b. Calcular la matriz de covarianzas $d \times d$ de los datos, mediante la siguiente fórmula:

$$\Sigma = \frac{1}{N-1} \cdot X^T X$$

c. Calcular los autovalores de la matriz de covarianzas.

d. Calcular los autovectores asociados a cada autovalor.

e. Seleccionar los m autovectores asociados a los m autovalores de mayor tamaño para formar la base del nuevo subespacio.

f. Proyectar los datos sobre el nuevo subespacio, esta proyección corresponde a los datos con las dimensiones reducidas.

2.2. Linear Discriminant Analysis

En esta sección se explican los fundamentos teóricos de LDA, no se ha implementado manualmente este método, en cambio solamente se ha usado la implementación de la librería *scikit-learn*.
LDA es un método que encuentra la combinación lineal de variables que mejor separa los datos en dos o más clases. Para el caso expuesto aquí se tienen dos clases (diabético o sano). Dadas c clases LDA encontrará un subespacio de dimensión no mayor que $c-1$ donde proyectar los datos, de tal forma que se maximice la razón entre la variabilidad entre las clases sobre la variabilidad dentro de las clases, intuitivamente esto permite que las clases sean más fáciles de distinguir.
La base del nuevo subespacio se obtiene maximizando la siguiente función:

$$J(w) = \frac{w_i^T S_B w_i}{w_i^T S_W w_i}$$

Donde S_B y S_W son las matrices de dispersión entre clases y dentro de cada clase respectivamente. La solución de esta ecuación viene dada por la ecuación generalizada de autovectores:

$$S_B W = S_W W D$$

Donde S es la matriz diagonal de autovalores y W es la matriz donde cada columna es un vector de la nueva base.
Una vez encontrada esta base es posible realizar la proyección de los datos y obtener el valor discriminante, una vez hecho esto se requiere tener una función discriminante, en el caso implementado aquí se ha utilizado como base el modelo de Naïve Bayes explicado en la siguiente sección.

2.3. Naïve Bayes

2.3.1. Naïve Bayes General

En esta sección se explica el modelo básico de Naïve Bayes para clasificar dado un input discreto, luego, en base a lo expuesto se extenderá el método para input continuo. La lógica subyacente al método de Naïve Bayes (NB) es una de las propiedades del álgebra de probabilidades, el llamado teorema de Bayes, que expresa lo siguiente. Dados dos eventos A y B se tiene que

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

Esta ecuación se deduce de la ley de multiplicación, en base a este teorema se construye el método de NB como se explicará en los siguientes párrafos. El modelo utilizado por el clasificador corresponde a la siguiente probabilidad condicional

$$P(C|A_1, A_2, ..., A_n) \tag{2}$$

Donde C es la variable de clase dependiente y los A_i son los diferentes atributos predictores. Usando el teorema de Bayes se puede obtener que

$$P(C|A_1, A_2, ..., A_n) = \frac{P(C) \cdot P(A_1, A_2, ..., A_n|C)}{P(A_1, A_2, ..., A_n)} \tag{3}$$

El objetivo del clasificador es determinar, del conjunto de posibles valores que puede tomar C, cuál es el más probable. Para simplificar los cálculos se debe notar que el denominador permanece constante independiente del valor de C. Por lo tanto, basta encontrar el valor de C que maximiza la siguiente probabilidad:

$$P(C) \cdot P(A_1, A_2, ..., A_n|C) \tag{4}$$

Mediante el uso de la definición de probabilidad condicional y la suposición de independencia entre los atributos predictores se llega a la siguiente formulación equivalente:

$$P(C) \cdot P(A_1|C) \cdot P(A_2|C) \cdot ... \cdot P(A_n|C) \tag{5}$$

Encontrar la clase C que maximiza esta expresión corresponde a maximizar la expresión de la ecuación (3) y por lo tanto es equivalente a encontrar la clase más probable dada la información que se tiene. La función clasificadora de Naïve Bayes se puede entonces expresar de la siguiente forma:

$$classify(a_1, ..., a_n) = argmax_c \left[P(C = c) \cdot \prod_{i=1}^{n} P(A_i = a_i|C = c)) \right] \tag{6}$$

Los valores de cada probabilidad condicional $P(A_i = a_i|C = c)$ corresponden a estimaciones basadas en los datos empíricos sobre los que se esté trabajando.

2.3.2. Naïve Bayes Gaussiano

Utilizando los fundamentos explicados en la sección anterior se puede construir un clasificador que utilice como base datos que siguen una distribución Gaussiana. Las fórmulas expuestas en esta sección se utilizan para la construcción del clasificador bayesiano solicitado.
En base a la última ecuación de la sección anterior se pueden reemplazar los valores discretos de probabilidad por una función de densidad de probabilidad, obteniéndose el siguiente clasificador:

$$classify(a_1, ..., a_n) = argmax_c \left[P(C = c) \cdot \prod_{i=1}^{n} f(A_i = a_i|C = c) \right] \tag{7}$$

Donde f es una función de densidad de probabilidad condicionada para el valor de la clase dado. Si se toma f como la distribución de probabilidad normal, se tiene que:

$$f(x|c) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu_c)^2/2\sigma_c^2} \tag{8}$$

Donde σ_c es la desviación estándar y mu_c es la media para la clase c. Trabajar directamente con esta definición es complejo, es por ello que es más apropiado maximizar el logaritmo de la función dada que utilizar la fórmula directamente. El clasificador queda definido entonces de la siguiente forma:

$$classify(a_1, ..., a_n) = argmax_c \left[\ln P(C = c) \cdot \prod_{i=1}^{n} f(A_i = a_i | C = c) \right] \tag{9}$$

Esto se puede separar como sigue:

$$classify(a_1, ..., a_n) = argmax_c \left[\ln P(C = c) + \sum_{i=1}^{n} \ln f(A_i = a_i | C = c) \right] \tag{10}$$

Para el caso en que existe un único input predictor ($n = 1$), se tendrá la siguiente fórmula simplificada:

$$classify(a) = argmax_c \left[\ln P(C = c) + \ln f(A = a | C = c) \right] \tag{11}$$

Reemplazando la fórmula de la distribución normal se obtiene:

$$classify(a) = argmax_c \left[\ln P(C = c) - 0{,}5 \cdot \ln\left(2\pi\sigma_c^2\right) - \frac{(x - \mu_c)^2}{2\sigma_c^2} \right] \tag{12}$$

Notar que se puede separar el logaritmo de 2 y al sumarse como constante podría despreciarse de la fórmula al maximizar, el clasififcador final sería:

$$classify(a) = argmax_c \left[\ln P(C = c) - 0{,}5 \cdot \ln\left(\pi\sigma_c^2\right) - 0{,}5 \cdot \frac{(x - \mu_c)^2}{\sigma_c^2} \right] \tag{13}$$

Básicamente al tener dos clases el clasificador se reduce a comparar cuál de las dos entrega la mayor probabilidad logarítmica, para ello es necesario estimar los parámetros μ_c y σ_c^2, para ello se utilizan las siguientes fórmulas estadísticas.
La media se estima mediante la siguiente fórmula para cada clase:

$$\mu_c = \frac{1}{N_c} \cdot \sum_{c_i = c} x_i \tag{14}$$

Donde N_c es la cantidad de muestras que pertenecen a la clase c. Esta fórmula indica que se debe obtener la media aritmética considerando todos los elementos cuya clase es c.
La varianza se estima mediante la siguiente fórmula para cada clase:

$$\sigma_c^2 = \frac{1}{N_c - 1} \cdot \sum_{c_i = c} (x_i - \mu_c)^2 \tag{15}$$

Esta fórmula indica que se debe obtener la varianza muestral considerando todos los elementos cuya clase es c. Notar que se divide por $N - 1$ y no N para que el estimador sea insesgado.
Finalmente, es necesario estimar las probabilidades de cada clase $P(C = c)$ esto se realiza mediante la siguiente fórmula:

$$P(C = c) = \frac{N_c}{N} \tag{16}$$

Donde N correspondería al total de datos entre todas las clases.

3. Implementación

En esta sección se muestran las implementaciones en Python de todos los elementos que fueron agregados sobre el código inicial que se entregó en clases. No se muestran las refactorizaciones realizadas para mejorar el orden del código inicial, traspasándolo desde un código estructurado a una versión orientada a objetos.

3.1. Dimensión Óptima

```
def find_optimal_dimension(x_train, x_test, y_train, y_test, dimensions):
    r = 0  # Classification rate
    k = 1  # Number of components
    for n_components in xrange(1, dimensions + 1):
        # Entrenar PCA+LDA con la cantidad de componentes dada.
        lda_train, lda_test = pca_lda(x_train, x_test, y_train, n_components)
        # Clasificar Bayes
        gnb = GaussianNB()
        gnb.fit(lda_train, y_train)
        r_i = gnb.score(lda_test, y_test)
        if r_i > r:
            r = r_i
            k = n_components
    return k
```

Este método obtiene la dimensión óptima para realizar PCA , para ello se prueba con la cantidad de componentes desde 1 hasta el máximo de dimensiones que poseen los datos (que viene dado por el parámetro de dimensiones). Se realiza PCA+LDA y luego se aplica el clasificador bayesiano de *scikit-learn*. El algoritmo busca el clasificador que obtenga la mayor precisión de clasificación.

3.2. Gráfico de Probabilidad Condicional

```
    def conditional_probability(self, x, y):
        """Este metodo construye el grafico de las funciones de probabilidad usando Bayes.
        """
        # --- Probabilidades
        pylab.figure()

        #Clases...
        pylab.scatter(x, y[:,0], color='blue', label='$\mathcal{P}(D^-|LDA)$')
        pylab.scatter(x, y[:,1], color='red', label='$\mathcal{P}(D^+|LDA)$')

        #Etiquetas...
        pylab.xlabel('$LDA$')
        pylab.ylabel('$P(DIABETES|LDA)$')
        pylab.legend()
```

Este método grafica la probabilidad de tener o no diabetes (es decir, la probabilidad de pertenecer a una cierta clase) dado el valor de la proyección LDA. Utiliza el método *scatter* de la librería *pylab* para graficar los probabilidades de cada clase.

3.3. Naïve Bayes

3.3.1. Entrenamiento

```
def fit(self, training_set, training_set_classes):
    # Se separan los elementos positivos de los negativos.
    lda_data_positive = training_set[training_set_classes == 1]
    lda_data_negative = training_set[training_set_classes == 0]

    # Se estiman las medias.
    self._mu_positive = np.mean(lda_data_positive)
    self._mu_negative = np.mean(lda_data_negative)

    # Se estiman las varianzas...
    self._var_positive = np.var(lda_data_positive)
    self._var_negative = np.var(lda_data_negative)

    # Se estima la probabilidad a priori (p_negative se obtendria con el complemento)
    self._p_positive = float(len(lda_data_positive)) / len(training_set)
    self._p_negative = 1 - self._p_positive
```

Este método se encarga de ajustar los parámetros requeridos por el clasificador, primero se separan los datos entre las clases, luego una vez hecho esto se estiman las medias y varianzas de cada clase. Finalmente se estiman las probabilidades de cada clase, todo esto se almacenado en los atributos de la instancia del clasificador.

3.3.2. Predicción

```
def predict(self, testing_set):
    # Se inicializan las variables requeridas por el clasificador.
    n = len(testing_set)

    # Se calcula la log-probabilidad de la clase positiva.
    log_p_positive = np.log(self._p_positive)
    pdf_positive = - 0.5 * np.sum(np.log(np.pi * self._var_positive))
    pdf_positive -= 0.5 * np.sum(((testing_set - self._mu_positive) ** 2) /
                                 self._var_positive, 1)
    positive_discriminant = log_p_positive + pdf_positive

    # Se calcula la la log-probabilidad de la clase negativa.
    log_p_negative = np.log(self._p_negative)
    pdf_negative = - 0.5 * np.sum(np.log(np.pi * self._var_negative))
    pdf_negative -= 0.5 * np.sum(((testing_set - self._mu_negative) ** 2) /
                                 self._var_negative, 1)
    negative_discriminant = log_p_negative + pdf_negative

    # Se retorna el conjunto de predicciones para cada caso de prueba.
    return [int(i) for i in positive_discriminant > negative_discriminant]
```

Este método se encarga de determinar las clases a las que pertenece un conjunto de datos de prueba. Para ello se calcula la probabilidad logarítmica de que cada ejemplo pertenezca a la clase positiva y negativa. Luego se obtiene un vector que contiene el valor 1 en las posiciones donde la probabilidad logarítmica de la clase positiva es mayor que la probabilidad logarítmica de la clase negativa, y un 0 de otro modo. Este vector es el mismo que entrega la implementación de *scikit-learn* del clasificador. Finalmente, se debe mencionar que se utiliza la probabilidad logarítmica pues es matemáticamente más simple de manejar y que las probabilidades vienen dadas por el uso de la distribución normal como es de esperarse de un clasificador Gaussiano.

3.3.3. Evaluación

```
def score(self, testing_set, testing_set_classes):
    # Se calcula la precision.
    testing_pred = self.predict(testing_set)
    mislabeled_points = (testing_pred != testing_set_classes).sum()
    score = 1 - float(mislabeled_points) / len(testing_set)

    #Se retorna el valor calculado.
    return score
```

Este método se encarga de determinar la precisión del clasificador entrenado, para ello obtiene el vector de predicciones y lo compara con el vector real de clases. La precisión viene dada por la cantidad de aciertos sobre la cantidad total de datos probados, o alternativamente se puede calcular como el complemento de la razón de fallos, que corresponde a la forma en que se ha implementado.

4. Resultados Experimentales

4.1. Resumen General

A continuación se muestran los resultados del clasificador bayesiano para el diagnóstico de diabetes. Notar que tanto la librería como la implementación propia corresponden en todos los casos, por lo que solo se muestran los resultados obtenidos con la librería.

Dimensión	LDA	Probabilidad
1	-0.13	0.48
2	-0.29	0.48
3	-0.29	0.49
4	-0.31	0.49
5	-0.31	0.49
6	-0.30	0.49
7	-0.30	0.49
8	-0.30	0.49

Cuadro 3: Puntos de intersección de las probabilidades condicionales para cada dimensión.

Notar que siempre se intersectan alrededor de 0.5, en cambio notar que varía el valor de LDA en el eje X, lo cual implica que la frontera de clasificación va variando hasta converger al valor de -0.30.

Dimensión	Puntos Mal Clasificados	Precisión Clasificador
1	75	0.6753
2	62	0.7316
3	62	0.7316
4	64	0.7229
5	59	0.7446
6	54	0.7662
7	51	0.7792
8	52	0.7750

Cuadro 4: Rendimiento del clasificador según dimensiones de PCA.

El desempeño óptimo ocurre cuando se utilizan 7 dimensiones para PCA, esto implica una reducción de dimensionalidad de 1 respecto a las 8 dimensiones originales. Notar que el mayor cambio de rendimiento se produce al agregar la segunda dimensión en PCA.

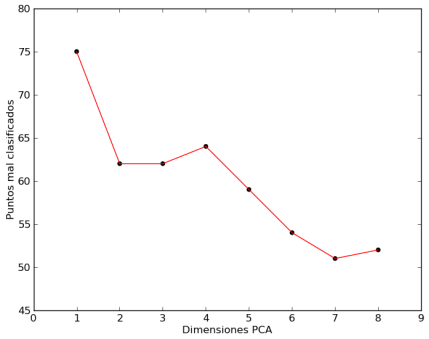

Figura 1: Puntos mal clasificados vs. dimensión PCA.

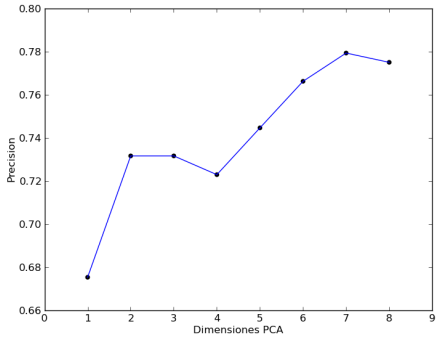

Figura 2: Precisión vs. dimensión PCA.

Lógicamente el gráfico de la precisión es un reflejo de los puntos mal clasificados. Notar que el óptimo ocurre justamente cuando la dimensión de PCA es 7, solo una dimensión menos que el espacio original, esto implica que para este problema en específico PCA no aporta mucho al clasificador, notar que la cantidad de puntos mal clasificados solo se reduce en 1. Esto en general no se cumple, ya que en muchos casos el uso de PCA permite mejorar el rendimiento del clasificador al eliminar ruido de los datos y reducir la *maldición de la dimensión*.
Se presenta en las siguientes páginas los resultados obtenidos para diferentes cantidades de componentes utilizadas en PCA. Se detallan los resultados más importantes, los gráficos sobrantes se han anexado en el Apéndice A.

4.2. Resultados Detallados

4.2.1. PCA+LDA 1-D

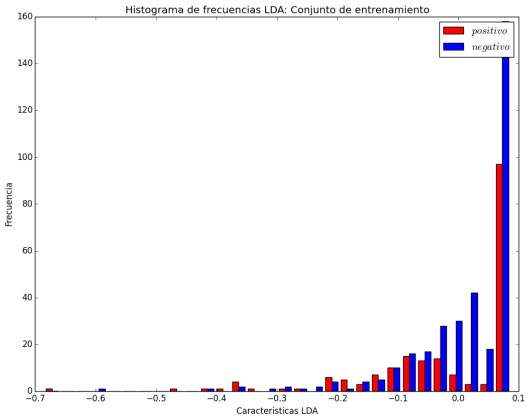

Figura 3: Histograma para 1-D

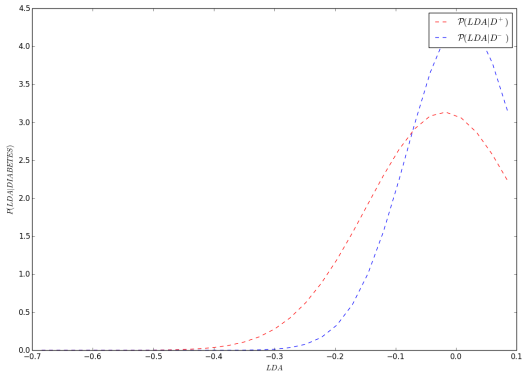

Figura 4: Función de densidad de probabilidad ajustada para 1-D

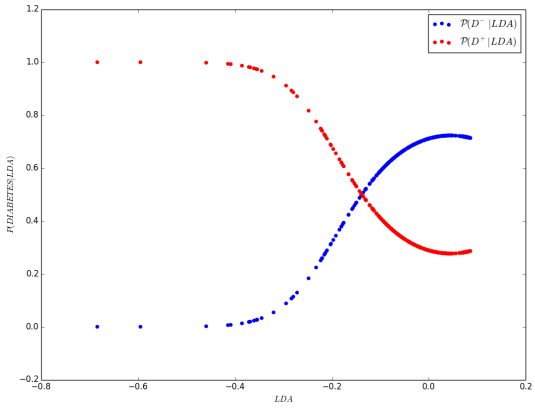

Figura 5: Probabilidad condicional de clase para 1-D

Al utilizar una única dimensión para PCA se puede observar una clara pérdida de la información, esto hace que la proyección obtenida por LDA no permita clasificar de manera óptima los ejemplos de prueba.
Se puede observar que las distribuciones se encuentran totalmente superpuestas y es prácticamente imposible distinguir entre datos positivos (diagnosticados con diabetes) y datos negativos (personas sanas).
La posición actual del punto de intersección para este caso se encuentra con un claro sesgo a la derecha, se verá que al aumenta la dimensión este punto empieza a moverse a lo largo del eje de las abscisas hacia la izquierda, de la misma forma el centro de la distribución del valor de LDA para casos positivos se desplazará en esa dirección.
La precisión para este caso es de 67.53 %, muy detrás de todos los otros casos que superan el 70 %. Se debe destacar que a pesar del evidente problema con la superposición de igual logra clasificar más de dos tercios de los ejemplos de prueba correctamente.

4.2.2. PCA+LDA 2-D

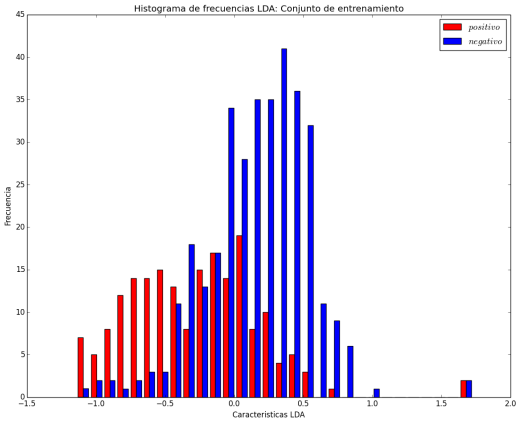

Figura 6: Histograma para 2-D

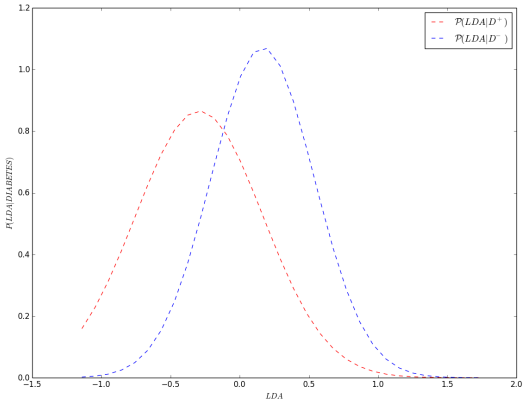

Figura 7: Función de densidad de probabilidad ajustada para 2-D

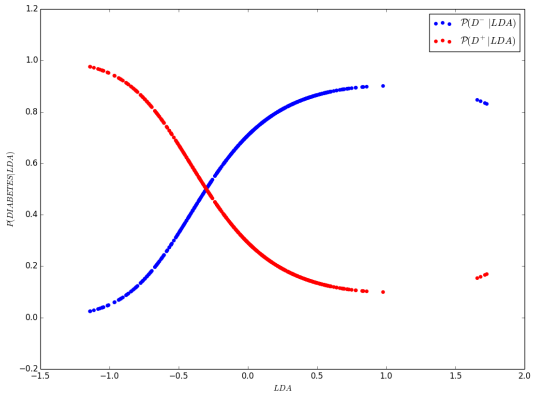

Figura 8: Probabilidad condicional de clase para 2-D

Al utilizar dos dimensiones para PCA se observa una relativa mejora respecto al caso unidimensional. La pérdida de información ya no es tan alta, aunque aún está lejos de ser óptima.
Se puede observar que las distribuciones se encuentran menos superpuestas que en el caso anterior (tanto en el histograma como en el gráfico de funciones de densidad de probabilidad), esta vez es un tanto más probable distinguir entre las dos clases.
La posición actual del punto de intersección se ha desplazado hacia la izquierda, desde -0.13 que para el caso anterior hasta -0.29, a partir desde este momento el punto de intersección variará mucho, a pesar que la precisión si variará.
La precisión para este caso es de 73.16 %, si bien no es tan óptimo como otros casos ya empieza a ser competitivo con los clasificadores que utilizaron PCA con otras dimensiones. Se debe notar que agregar esta dimensión produce el aumento más grande sobre la precisión y después se puede observar que cada dimensión aporta menos. Esto se debe a que las dimensiones son agregadas en orden descendente respecto a la variabilidad que contienen.

4.2.3. PCA+LDA 7-D

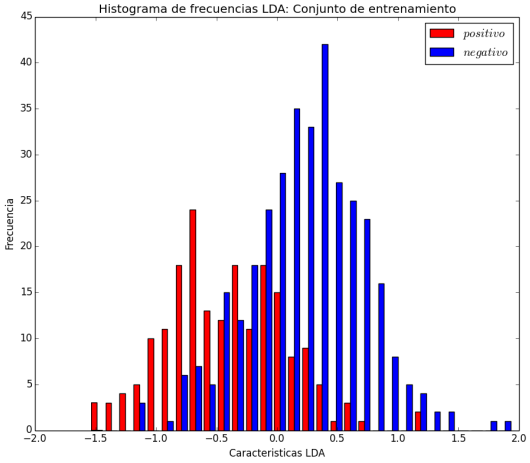

Figura 9: Histograma para 7-D

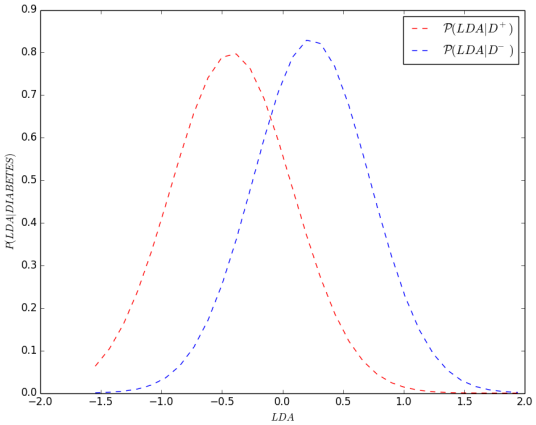

Figura 10: Función de densidad de probabilidad ajustada para 7-D

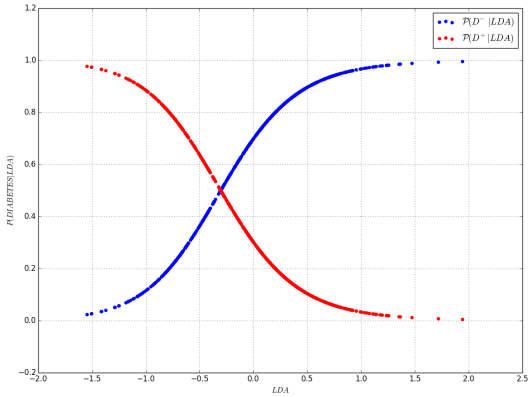

Figura 11: Probabilidad condicional de clase para 7-D

Al utilizar siete dimensiones para PCA se observa que se obtiene la mejor clasificación con respecto a todas las otras dimensiones probadas (ver Anexo A, donde se encuentran las imágenes de los gráficos de las otras dimensiones probadas). La dimensión óptima se calculó utilizando un algoritmo que busca el clasificador que obtenga la mayor precisión de clasificación, de acuerdo a lo mostrado en la sección de implementación.
Se puede observar que las distribuciones se encuentran menos superpuestas con respecto a las otras dimensiones probadas. Para este problema en particular la ganancia obtenida con PCA+LDA no fue demasiada, la dimensionalidad se redujo en uno respecto al valor original, y se logró clasificar un ejemplo más de manera correcta.

La posición actual del punto de intersección se encuentra en LDA igual a -0.30, para este valor no se puede decir si el paciente padece diabetes o no. Nótese que los valores LDA se pueden utilizar para crear un criterio discriminante, entonces se pueda dividir el gráfico que se muestra en la Figura 11 en los siguientes intervalos:

- Si LDA se encuentra entre $(-\infty; -0{,}4)$ existe una mayor probabilidad que el paciente padezca diabetes.
- Si LDA se encuentra cercano al punto de intersección, es decir el intervalo $(-0{,}4; -0{,}2)$, no se puede determinar con certeza si el paciente padece diabetes o no.
- Si LDA se encuentra entre $(-0{,}2; \infty)$ existe una menor probabilidad que el paciente padezca diabetes o por otro lado que no padezca diabetes.

Se ha definido que la región de incertidumbre se encontrará a 0.1 desde el punto de intersección en ambas direcciones. El valor 0.1 ha sido seleccionado observando las curvas empíricas de probabilidad condicional (notar que se ha agregado una grilla en este gráfico para facilitar la lectura) y no se debe tomar como un valor exacto, aunque provee una estimación educada para la región de incertidumbre.
La precisión de clasificación sobre los datos de prueba si un paciente padece diabetes o no es de un 77.92 %. Se determinó que este es el valor óptimo de rendimiento para el clasificador mediante este enfoque, cabe mencionar que utilizando otro enfoque más complejo se podría obtener una precisión más alta. Aunque en comparación a los clasificadores encontrados en la bibliografía para estos datos en particular la precisión alcanzada es similar.

5. Conclusiones

Se ha completado satisfactoriamente el objetivo del taller II que correspondía a construir un diagnosticador de diabetes utilizando PCA, LDA y un clasificador Gaussiano basado en Naïve Bayes. Este trabajo ha contribuido en mejorar el entendimiento de las técnicas de clasificación aprendidas en clases mediante su uso e implementación. Además, ha permitido profundizar el conocimiento adquirido en cátedra, ya que el desarrollo del taller ha requerido de una extensa recopilación bibliográfica del tema además de estudiar las posibles implementaciones del clasificador bayesiano.
Se ha mostrado además la clara diferencia que existe en la precisión entregada por el clasificador dependiendo de la cantidad de componentes que hayan sido usados con PCA, se ha comprobado empíricamente que la mejor precisión se obtiene con siete dimensiones. Además se ha visto que la mayor mejora de la precisión del método se produce en el salto desde una dimensión a dos dimensiones para PCA. Cabe destacar que después de agregar la segunda dimensión el rendimiento del clasificador crece más lentamente, esto debido a que cada dimensión contiene cada vez menor variabilidad y por lo tanto no aporta tanta información como las primeras dos. Es decir el crecimiento de precisión respecto a la cantidad de dimensiones sigue la ley de rendimientos decrecientes. Después del óptimo la calidad del clasificador empezó a disminuir a medida que se aumentaba la cantidad de dimensiones en PCA.
Se debe mencionar que si para PCA se utiliza una única dimensión el rendimiento del algoritmo es bajo respecto a los demás casos, esto debido a que si se hace una reducción solo sobre la primera componente principal se pierde mucha información contenida en otras dimensiones de los datos del problema.
Los datos utilizados para el desarrollo de este taller presentaron la particularidad que PCA no redujo mucho la cantidad de dimensiones, esto no ocurre en el caso general, donde el uso de PCA permite reducir en gran medida la dimensión de los datos al eliminar ruido e información redundante. En este caso la dimensión solo se redujo desde ocho a siete, un cambio no muy significativo y que solo aportó en una mejora de un ejemplo más clasificado correctamente. En otros casos el aporte de PCA puede ser significativo y permitir grandes mejoras del rendimiento, pues permite reducir la *maldición de la dimensión* (como se pudo ver en el último control de lectura).
Una propuesta interesante sería probar la implementación realizada con datos de otro problema de mayores dimensiones en función de observar si PCA produce un cambio significativo. También sería interesante generalizar el código implementado a problemas con más de dos clases, ya que la implementación actual del clasificador bayesiano propio solo admite problemas de clasificación binaria, a diferencia de la implementación de la librería *scikit-learn* que admite problemas de clasificación n-arios.
Sería interesante poder obtener cuáles son las dimensiones seleccionadas y cuáles son las dos primeras componentes principales, esto permitiría entregar una interpretación dentro del contexto del problema. Además, sería posible interpretar el criterio de clasificación en función de las variables definidas, sin embargo, las librerías de *scikit-learn* no entregan una manera simple de obtener esta información, por lo tanto sería necesario diseñar e implementar los algoritmos PCA y LDA sólo para obtener estos datos, pero se ha considerado que esto está fuera del alcance del taller.
El uso de la librería *scikit-learn* en el desarrollo del taller facilitó bastante la implementación, ya que esta contenía todos los métodos requeridos ya implementados, lo que permitió simplemente realizar los llamados para aplicarlos. Lamentablemente la librería no se pudo hacer funcionar en Windows, por lo que para evitar problemas se realizó toda la implementación en un ambiente Linux.
Por último, queda por mencionar los problemas encontrados durante el desarrollo del taller, a decir, la implementación propia del clasificador bayesiano, esto debido a que la bibliografía referenciada contenía diferentes métodos para determinar la clase a la que pertenecía una instancia de los datos. Si bien todos estos métodos son equivalentes, hubo problema con la implementación de las fórmulas debido a ciertas incongruencias encontradas en las fuentes, una vez se tuvo las fórmulas correctas para el clasificador, este funcionó de manera idéntica al clasificador de la librería *scikit-learn*, confirmandose así que la implementación realizada es correcta.

Referencias

[1] S. Russell. (2010) *Artificial Intelligence: A Modern Approach*.

[2] Peña, D. (2002). *Análisis de datos multivariantes (Vol. 24)*. Madrid: McGraw-Hill.

[3] Sridhar Mahadevan. *Linear Classification Models*. Recuperado Julio 19, 2014, desde http://edlab-www.cs.umass.edu/cs589/2010-lectures/bayesian-classification.pdf

[4] *Código fuente scikit-learn*. Recuperado Julio 19, 2014, desde https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/naive_bayes.py

[5] *Linear Discriminant Analysis - Part I*. Recuperado Julio 19, 2014, desde https://onlinecourses.science.psu.edu/stat557/book/export/html/35

[6] Jia Li. *Linear Discriminant Analysis*. Recuperado Julio 19, 2014, desde http://sites.stat.psu.edu/ jiali/course/stat597e/notes2/lda.pdf

[7] Iyad Batal, *Principal Component Analysis*. http://people.cs.pitt.edu/ iyad/PCA.pdf

[8] Meng Xu, *An intuitive explanation of PCA*. http://mengnote.blogspot.com/2013/05/an-intuitive-explanation-of-pca.html

[9] Bekios-Calfa, J., Buenaposada, J. M., & Baumela, L. (2011). *Revisiting linear discriminant techniques in gender recognition*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(4), 858-864.

[10] Vincent Sigillito. (1990). *Pima Indians Diabetes Data Set*, The Johns Hopkins University. Recuperado Julio 19, 2014, desde https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[11] Raj Anand, Vishnu Pratap Singh Kirar, Kavita Burs. (2012). *Data Pre-processing and Neural Network Algorithms for Diagnosis of Type II Diabetes: A Survey* International Journal of Engineering and Advanced Technology (IJEAT).

Apéndice A Gráficos Restantes

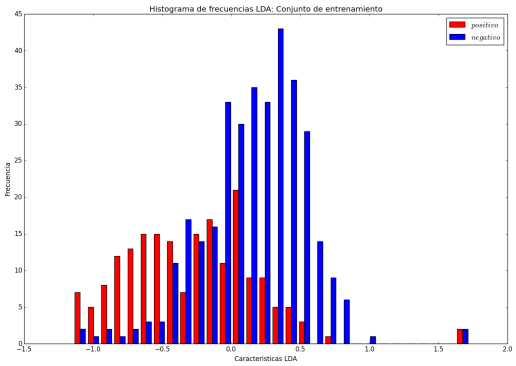

Figura A12: Histograma para 3-D

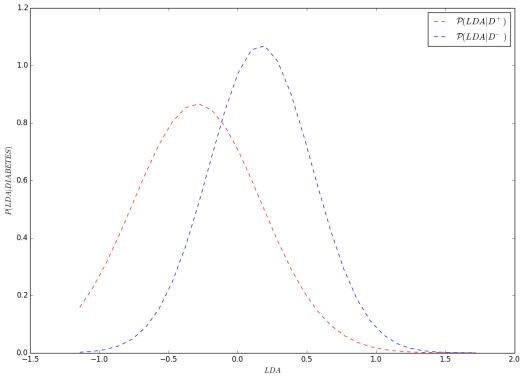

Figura A13: Función de densidad de probabilidad ajustada para 3-D

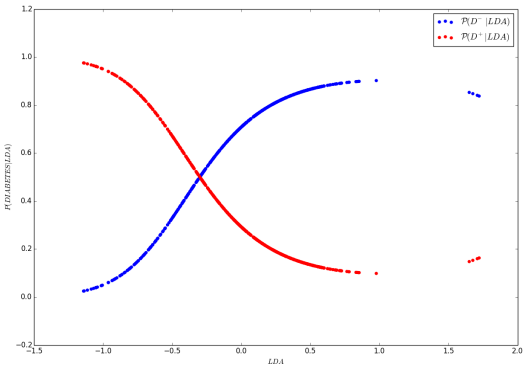

Figura A14: Probabilidad condicional de clase para 3-D

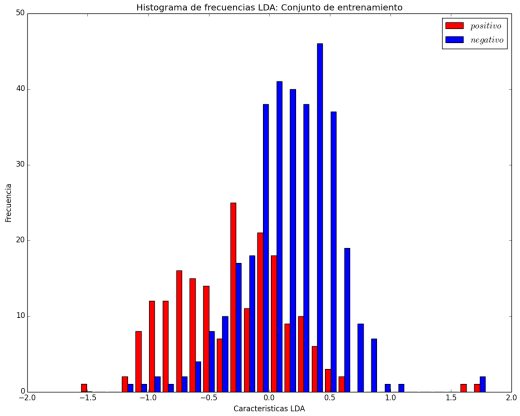

Figura A15: Histograma para 4-D

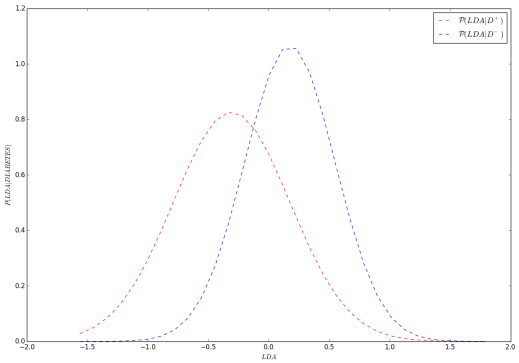

Figura A16: Función de densidad de probabilidad ajustada para 4-D

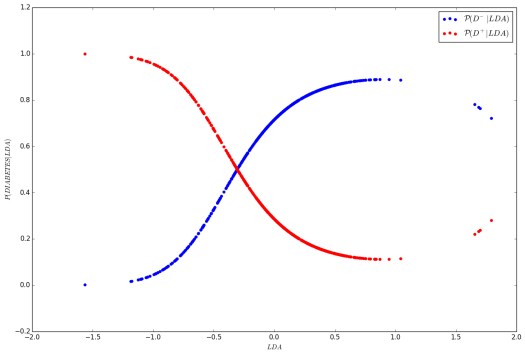

Figura A17: Probabilidad condicional de clase para 4-D

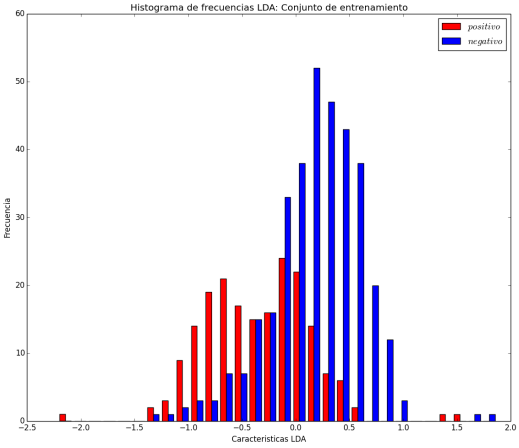

Figura A18: Histograma para 5-D

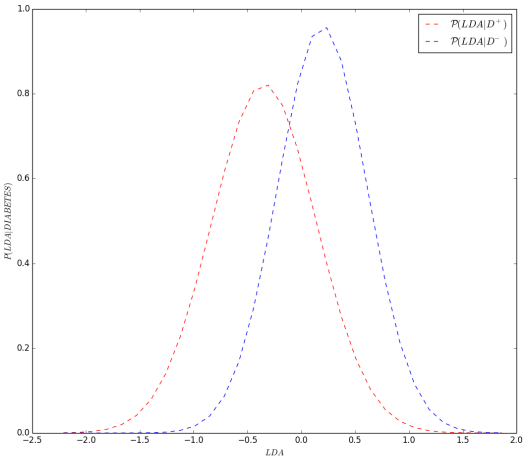

Figura A19: Función de densidad de probabilidad ajustada para 5-D

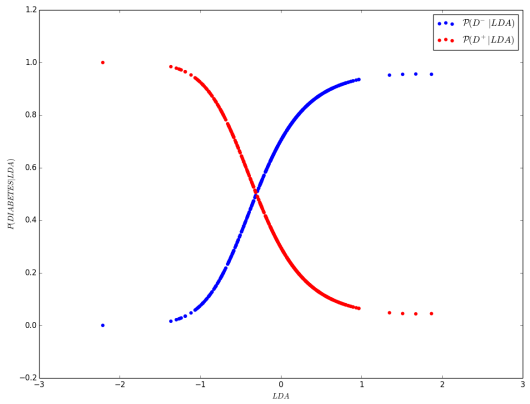

Figura A20: Probabilidad condicional de clase para 5-D

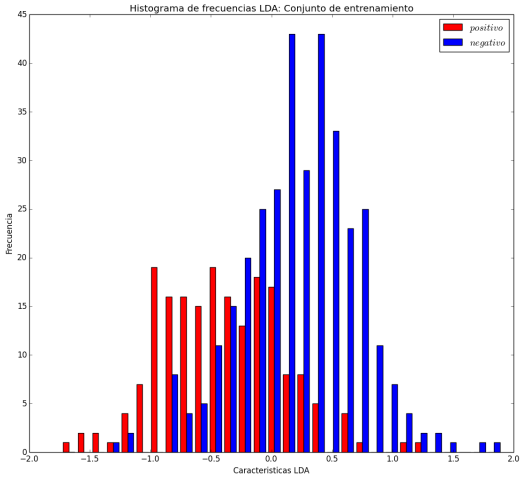

Figura A21: Histograma para 6-D

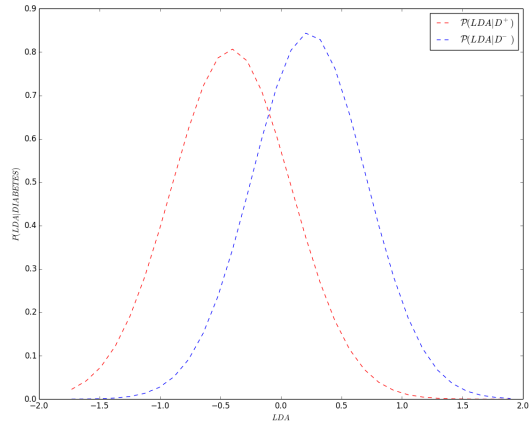

Figura A22: Función de densidad de probabilidad ajustada para 6-D

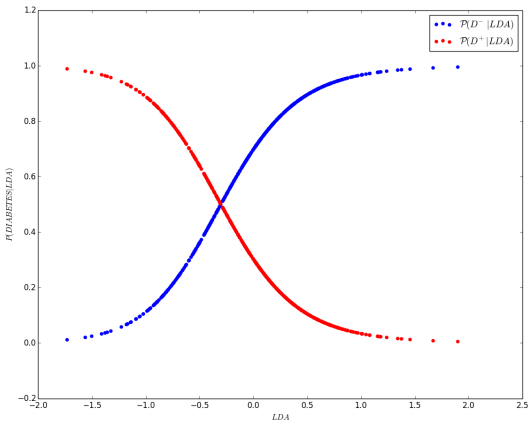

Figura A23: Probabilidad condicional de clase para 6-D

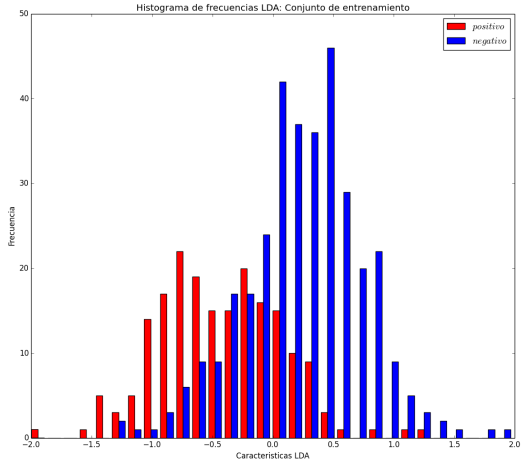

Figura A24: Histograma para 8-D

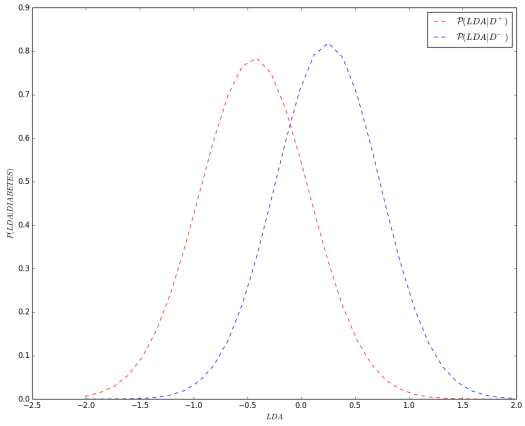

Figura A25: Función de densidad de probabilidad ajustada para 8-D

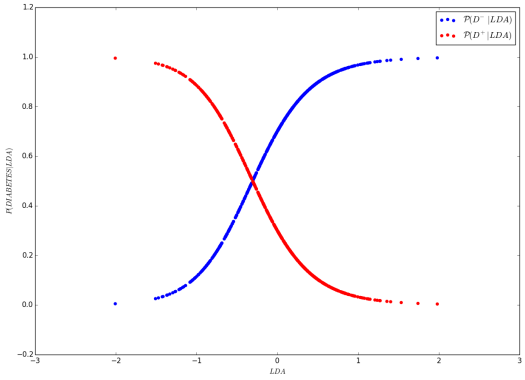

Figura A26: Probabilidad condicional de clase para 8-D