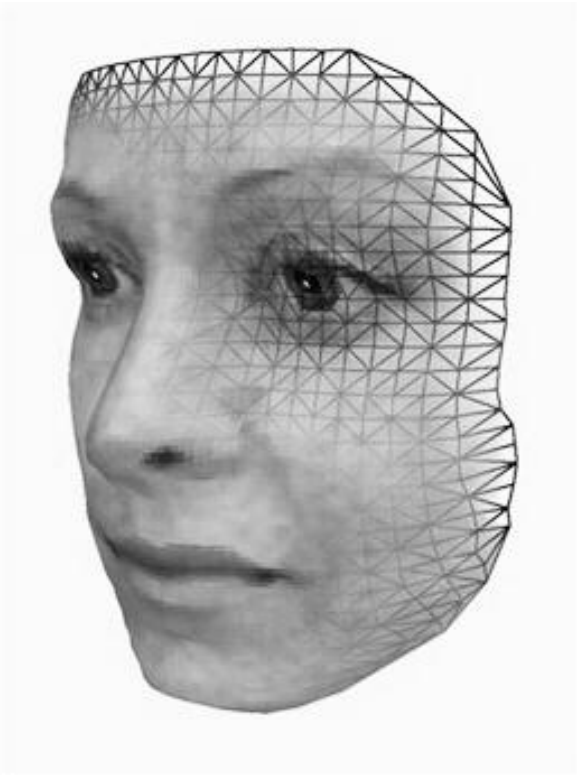
# Introducción al Reconocimiento de Patrones

**Curso 2011**

## Proyecto Final

# RECONOCIMIENTO DE CARAS Y DESEMPEÑO DE CLASIFICADORES



**Camila Riverón**

**Lucía Marroig**

# ÍNDICE

Resumen ..... 4
Reconocimiento de caras ..... 4
**Introducción** ..... 4
**Funcionamiento** ..... 5
**Enfoque del proyecto** ..... 5
**Características utilizadas** ..... 6
**Análisis del problema** ..... 7
Selección de características ..... 8
**Descripción general** ..... 8
**Enfoque específico de nuestro problema** ..... 8
**Aplicación** ..... 9
Algoritmos de Clasificación ..... 10
**K-Vecinos más cercanos** ..... 10
**Introducción** ..... 10
**Elección de k** ..... 11
**Regla de clasificación de los k-vecinos más cercanos k-NN** ..... 12
**Regla del vecino más cercano 1-NN** ..... 12
**Ventajas** ..... 12
**Desventajas** ..... 12
**K vecinos más cercanos en Weka** ..... 13
**K vecinos VS. Vecino más cercano** ..... 13
**Árboles de decisión** ..... 13
**Introducción** ..... 13
**Proceso de clasificación** ..... 14
**Impureza** ..... 14
**Algoritmo c4.5** ..... 15
**Bagging** ..... 16
**Introducción** ..... 16
**Algoritmo** ..... 16
**Ventajas** ..... 17
**Desventajas** ..... 17
**Por qué funciona Bagging?** ..... 17
Clasificación ..... 18
**Vecinos más cercanos** ..... 18
**Árboles de Decisión** ..... 18
**Bagging** ..... 19

**Resultados** ..... 21
Clasificación induciendo ruido ..... 21
**Introducción** ..... 21
**Vecinos más cercanos** ..... 22
**Árboles de Decisión** ..... 22
**Bagging** ..... 23
bonus ..... 24
**Clasificación sin selección de características** ..... 24
**Discusión** ..... 25
**Clasificación con vector de características alternativo** ..... 26
**K-vecinos** ..... 26
**Árboles de decisión** ..... 27
**Bagging** ..... 27
Conclusiones ..... 27
**Selección** ..... 27
**Clasificación** ..... 28
**Clasificación Aplicando Ruido** ..... 29
Bibliografía y Herramientas ..... 30
**Bibliografía** ..... 30
**Herramientas** ..... 30

## RESUMEN

En este documento presentamos nuestro proyecto, en el cual trabajaremos con la BioID face database para estudiar el desempeño de tres clasificadores diferentes en el reconocimiento de caras. Los clasificadores seleccionados son Vecinos más cercano, Bagging y Árboles de Decisión.

Se hará un estudio de falsos positivos/falsos negativos, porcentajes de acierto, dependencia con los parámetros de los clasificadores, etc.

Las características extraídas de las imágenes de la BioID face database se basan en la posición manual de los ojos, por ello luego estas se alteraran y se realizará un análisis del desempeño de los mismos clasificadores, cuando las características se extraen introduciendo un error y se verá cuál de ellos es más sensible a este hecho.

Se realiza un análisis del problema y posibles dificultades que pueden surgir por no poseer un conjunto de características óptimo para el problema.

## RECONOCIMIENTO DE CARAS

### INTRODUCCIÓN

El reconocimiento de caras se convirtió en los últimos años en un área de investigación que abarca varias disciplinas, como procesamiento de imágenes, reconocimiento de patrones, visión por computadora (subcampo de la inteligencia artificial) y redes neuronales.

Involucra a investigadores del área de informática como también a neurocientíficos y psicólogos. Se puede considerar dentro del campo de reconocimiento de objetos, donde la cara es un objeto tridimensional bajo variaciones de iluminación, pose, etc., que será identificada basada en su proyección en dos dimensiones.

Su principal uso actualmente es en sistemas de seguridad para el reconocimiento de empleados en empresas, etc. En estos sistemas se utiliza un lector que define las características del rostro, y cuando este solicita el acceso, se verifica comparando los datos obtenidos con la base de datos. Futuros usos podrían ser el reconocimiento de caras en una caja registradora para que la misma permanezca cerrada en el caso de no reconocer a la persona que tiene intenciones de abrirla; así como también en los cajeros automáticos como método de sustitución del actual PIN. Otro uso de similar

índole y que seguramente sea el que crezca más rápido es la autenticación a través del rostro para loguearse a redes sociales, emails y cualquier servicio online que requiera identificación, este caso se verá más adelante.

## FUNCIONAMIENTO

El proceso general del reconocimiento de caras consta de cuatro módulos principales [1]:

1. **Detección de la cara:** detecta que hay una cara en la imagen, sin identificarla. Si se trata de un video, también podemos hacer un seguimiento de la cara. Proporciona la localización y la escala a la que encontramos la cara.
2. **Alineación de la cara:** localiza las componentes de la cara y, mediante transformaciones geométricas, la normaliza respecto propiedades geométricas, como el tamaño y la pose, y fotométricas como la iluminación. Para normalizar las imágenes de caras, se pueden seguir diferentes reglas, como la distancia entre las pupilas, la posición de la nariz, o la distancia entre las comisuras de los labios.
3. **Extracción de características:** proporciona información para distinguir entre las caras de diferentes personas según variaciones geométricas o fotométricas.
4. **Reconocimiento:** el vector de características extraído se compara con los vectores extraídos de las caras de la base de datos. Si encuentra uno con un porcentaje elevado de similitud, nos devuelve la identidad de la cara; si no, nos indica que es una cara desconocida.

## ENFOQUE DEL PROYECTO

Para este proyecto trabajamos con una base de datos de caras de la página BioID (www.bioid.com). Esta página trata la autenticación de identidad en los diferentes servicios online; redes sociales por ejemplo. Primero el usuario se registra en BioID y se le toman muestras con la cámara web y micrófono, del rostro y voz respectivamente, las cuales quedan almacenadas. Luego cuando el usuario desea loguearse a un sitio que acepta BioID, se pide la autenticación a éste tomando los datos del usuario a loguearse y BioID se encarga buscando en su base de datos si el usuario es quien dice ser.

Esta página posee una base de datos, BioID face database [2] la cual es expuesta para que los investigadores puedan comparar sus algoritmos con otros. La misma trata de representar escenas de la vida cotidiana en diferentes ambientes.

La BioID face database consta de 1521 imágenes en escala de grises con una resolución de 384x286 pixel. Cada una de ellas es una vista frontal de una de las 23 personas diferentes que posee la base. Por razones de comparación las imágenes contienen la posición de los ojos tomada manualmente en archivos adjuntos a las mismas.

**Figura 1. Imágenes de la base de datos BioID face database.**

## CARACTERÍSTICAS UTILIZADAS

BioID face database tiene además de las imágenes, archivos con las características extraídas para cada una de ellas y son estos los valores que utilizaremos como vector de características. Este vector se conforma con los siguientes 20 puntos de la cara:

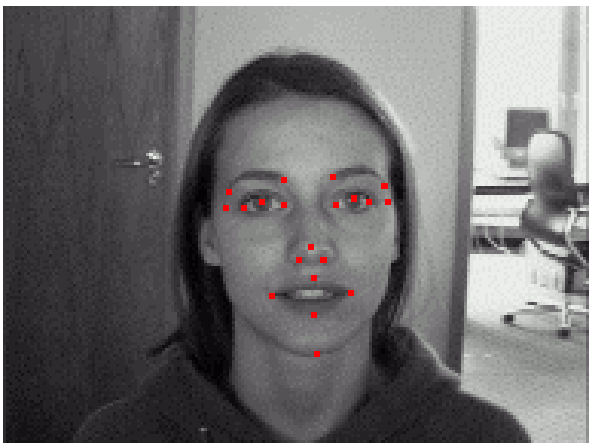


**Figura 2. Imagen de la BioID face database con los 20 puntos usados como características.**

Los 20 puntos son:

- pupila derecha
- pupila izquierda
- esquina derecha de la boca
- esquina izquierda de la boca
- extremo exterior de la ceja derecha

- extremo interior de la ceja derecha
- extremo interior de la ceja izquierda
- extremo exterior de la ceja izquierda
- sien derecha
- esquina externa del ojo derecho
- esquina interna del ojo derecho
- esquina interna del ojo izquierdo
- esquina externa del ojo izquierdo
- sien izquierda
- punta de la nariz
- narina derecha
- narina izquierda
- punto medio del borde externo superior de la boca
- punto medio del borde externo inferior de la boca
- punta de la barbilla

## ANÁLISIS DEL PROBLEMA

Las características brindadas por BioID face database están compuestas por las coordenadas de determinados puntos de la cara como ya dijimos. Como es natural de pensar esta puede no ser la mejor representación del problema: el reconocimiento de caras. Esto sucede ya que si una misma persona se toma una foto donde su cara aparece en el borde derecho de la imagen y luego se toma otra donde su cara aparece en el izquierdo, basándose en el criterio natural de la mayoría de los clasificadores (el cual es la distancia entre las características de los diferentes patrones) obtendríamos que se tratan de personas distintas cuando realmente no lo son.

Parecería más razonable encontrar ciertas características que vayan más allá de la posición espacial de la cara en la imagen y se enfoquen en distancias relativas, por ejemplo.

Nuestro proyecto se basa en las características crudas ya que decidimos enfocarnos en el tema de clasificadores y su reacción ante el ruido, dejando de lado el tema de extracción y selección de características (aunque se ve brevemente este último). Pero debido a la importancia que tendría este tema si los algoritmos fueran a utilizarse en un problema real (sistema de seguridad, etc.) decidimos hacer una prueba con características derivadas de las coordenadas la cual se presenta al final del documento.

# SELECCIÓN DE CARACTERÍSTICAS

## DESCRIPCIÓN GENERAL

La selección de características consiste en tomar un subconjunto del espacio de características que contenga aquellas más relevantes. Este proceso se realiza por tres razones fundamentales: para mejorar la clasificación, para aumentar la velocidad de procesamiento y para tener un mejor entendimiento del proceso a través del cual se genera la información.

El proceso de selección de características consiste en quitar aquellas que no contribuyen en la construcción de un buen predictor. Esto puede lograrse encontrando algún subconjunto óptimo del espacio de características o realizando un ranking de todas las características potencialmente relevantes. Por ejemplo, ante un espacio que cuenta con una gran cantidad de características se puede lograr quitar características redundantes, pero se debe tener cuidado ya que algunas de estas podrían ser relevantes.

Al proceso de selección podemos brindarle tres tipos de enfoques distintos. En primer lugar podemos utilizar el **filtrado**, consiste en seleccionar las características independientemente del clasificador utilizando alguna función de "relevancia". Por otro lado podemos utilizar el enfoque **encapsulado**, la idea es seleccionar subconjuntos de características en función de un clasificador específico. Este tiene la desventaja de ser muy costoso computacionalmente ya que se debe explorar todo el espacio de subconjuntos, además se debe proveer algún mecanismo para realizar la búsqueda por este espacio. Por último tenemos el método **intrínseco**, en este caso se realiza la selección en el proceso de aprendizaje, el cual devuelve las características seleccionadas y el clasificador entrenado.

## ENFOQUE ESPECÍFICO DE NUESTRO PROBLEMA

En el reconocimiento de caras los puntos anatómicos son los más importantes por su valor biológico y es por ellos que somos capaces de distinguir visualmente a las personas. Las caras son modeladas con un número de características biológicas limitado y es por ello que en general los algoritmos utilizan además otra información, como ser proporciones de distancias entre los distintos puntos y ángulos entre estos. [5]

Los puntos a seleccionar deben de generar un modelo simple y no demasiado sensible a pequeños cambios en los rostros. Esto nos lleva a un modelo con pocos puntos y correlacionados entre sí. La poca dimensionalidad a su vez reduce el tiempo computacional y mejora la robustez.

Para realizar la selección de características (puntos del rosto) optamos por utilizar un método de selección manual. Decidimos utilizar este método ya que conocemos bien el problema que estamos analizando y su espacio de características, y por lo tanto tenemos una visión global del mismo.

Contamos originalmente con 20 características que representan distintos puntos sobre el rostro de cada persona como fueron descriptas anteriormente. Cada característica se compone de un par de coordenadas (posición horizontal, posición vertical). Consideramos que si aplicamos un algoritmo de selección de características automático podríamos obtener resultados que separen estos pares lo cuál a priori no sería muy conveniente, ya que una única coordenada de un punto podría ser descartada quedando solamente la otra, que por sí sola carece de significado.

Hay varios aspectos a considerar a la hora de realizar una selección manual. Por un lado decidimos tomar un número fijo de características con las que queremos quedarnos, y a partir de éste evaluar el desempeño para los posibles subconjuntos que cuenten con esta cardinalidad. Esta es una opción frente a considerar todas las posibles combinaciones de subconjuntos a partir de la totalidad del espacio de características y evaluar el desempeño para cada uno de éstas. Este método es prácticamente inaplicable ya que al ser manual la selección implicaría un costo de tiempo demasiado elevado.

Por otra parte elegimos los árboles de decisión como factor de selección, es decir, evaluamos el desempeño de cada subconjunto generando árboles de decisión y observando el error de clasificación. El subconjunto elegido fue aquel que presentó una tasa de aciertos más elevada. Consideramos que los árboles de decisión representaban un buen criterio para descartar características ya que son sensibles al "ruido" y a las características irrelevantes.

## APLICACIÓN

Comenzamos fijando distintos tamaños para el conjunto de características objetivo, p=8, p=10, p=12, p=15 y p=20. Dividimos el espacio de características en muchos subconjuntos con los tamaños objetivos y generamos árboles de decisión para evaluar desempeño.

De este proceso llegamos a que el "mejor" subconjunto es de tamaño 8, y se compone de ambas pupilas, punto interno de las cejas, la punta de la nariz, ambos bordes

laterales de los labios y el centro del mentón, con aproximadamente un 67% de aciertos a la hora de la clasificación, el resto de los subconjuntos generados dieron tasas de acierto menores a la recién mencionada.

A continuación se muestra una imagen con el conjunto de características seleccionadas.

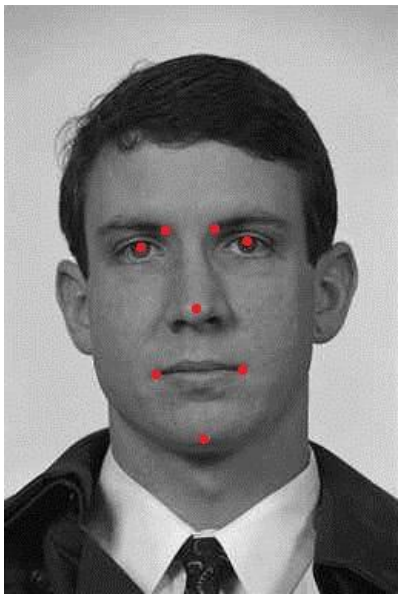


**Figura 3. Subconjunto de características con mejor desempeño.**

# ALGORITMOS DE CLASIFICACIÓN

## K-VECINOS MÁS CERCANOS

### INTRODUCCIÓN

*K-Vecinos más cercanos se trata de una técnica* ***supervisada no paramétrica****.*

- Por supervisada nos referimos a que el aprendizaje se basa en un conjunto de patrones de los cuales tenemos como información a qué clase pertenece cada uno.
- Por no paramétrica nos referimos a que no conocemos la distribución que presentan los patrones y por lo tanto estimamos el valor de la función densidad o la probabilidad a posteriori de que un patrón **x** pertenezca a la clase **w**

directamente a partir de la información proporcionada por el conjunto de patrones.

Esta técnica consta de tomar los k **patrones más cercanos** al que queremos clasificar (patrón X), delimitándose así un círculo con centro X en el caso que el espacio de características sea de dos dimensiones y una hiperesfera en el caso de dimensiones mayores.

La principal idea que se maneja con esta técnica de estimación es que el área del círculo que encierra los k patrones más cercanos será menor en regiones densamente pobladas y mayor en donde los patrones están más dispersos.

Llamamos:

- **n** al número total de patrones.
- **k** al valor elegido cómo la cantidad de vecinos más cercanos a X.
- $\mathbf{w_i}$ la **clase i**, i=1...c.
- $\mathbf{n_i}$ la cantidad total de patrones pertenecientes a la clase i.
- $\mathbf{k_i}$ es el número de patrones vecinos más cercanos pertenecientes a la clase i.
- **V(X)** volumen que delimita los k vecinos más cercanos de X.

## ELECCIÓN DE K

Tenemos que: $$\boldsymbol{p(x/wi) = \frac{ki}{niV(x)}}$$

Donde el $k_i$ depende del $n_i$, debiéndose cumplir para que el algoritmo sea consistente e insesgado*, que $k_i$ crezca a medida que lo hace $n_i$, pero que la cantidad de muestras $k_i$ no se acerque a la cantidad total de muestras, esto es:

$$\lim_{ni\to\infty} ki(ni) = \infty$$

Y

$$\lim_{ni\to\infty} \frac{ki}{ni} = 0$$

Es por ello que se toma un k de la forma $\boldsymbol{k = c\sqrt{n}}$.

* Estimador consistente: sus valores se acercan cada vez más del valor del parámetro a estimar según el tamaño de la muestra aumenta.
Estimador insesgado: el valor esperado es igual al valor que se busca.

## REGLA DE CLASIFICACIÓN DE LOS K-VECINOS MÁS CERCANOS K-NN

Se puede tomar como la probabilidad de una clase $w_i$ a priori $P(wi) = \frac{ni}{n}$ o tomar como que todas las clases son equiprobables. Se opta por la primera opción. Luego se calculan las probabilidades a posteriori directamente para quedarnos con **la clase con mayor presencia** entre los k vecinos que rodean a X, esto es:

$$p(x, wi) = p(x/wi)P(wi) = \frac{ki}{niV(x)}\frac{ni}{n} = \frac{ki}{nV(x)}$$

$$P\left(wi/x\right) = \frac{p(x,wi)}{\sum_{j=1}^{c} p(c,wc)} = \frac{ki}{k}$$

Y por lo tanto **la regla de decisión** es la siguiente

Selecciono $w_j$ si se cumple $kj = \max_{i=1..c}\{ki(x)\}$

## REGLA DEL VECINO MÁS CERCANO 1-NN

Si tomamos k=1 tenemos lo que se conoce como la regla del vecino más cercano en la cual se le asigna al patrón X la clase del patrón más cercano pudiéndose tomar distintos tipos de medidas para distancias (Manhattan, Euclídea, etc.)

## VENTAJAS

Como puede verse, estas reglas proporcionan una estimación directa de la probabilidad a posteriori de cada una de las clases y las reglas de clasificación son **sencillas y fáciles de interpretar**.

Hay algunas técnicas de **reducción de ruido** que sólo funcionan para k-NN que pueden ser eficaces para mejorar la precisión de este clasificador.

## DESVENTAJAS

Como hace todo el trabajo en tiempo de ejecución el algoritmo puede ser **muy lento** en el caso de conjuntos muy grandes.

Es muy sensible a **características redundantes** porque estas colaboran para la proximidad entre los patrones.

### K VECINOS MÁS CERCANOS EN WEKA

Los algoritmos del Vecino más cercano y K Vecinos más cercanos se encuentran en Weka en los llamados **Lazy Classifiers** [4]. Lazy Classifiers se debe a que realmente no se aprende nada de las muestras de aprendizaje, sino que todo el trabajo se hace al momento de clasificar un nuevo patrón.

Para el caso del Vecino más cercano el algoritmo se llama IB1 y para K Vecinos más cercanos IBk. Ambos algoritmos utilizan como medida de distancia por defecto la distancia Euclídea, aunque esta opción se puede modificar.

Se puede usar para el caso de IBk pesos en las medidas de distancia penalizando los patrones que se encuentran más lejos de la muestra a clasificar.

### K VECINOS VS. VECINO MÁS CERCANO

K Vecinos se diferencia del Vecino más cercano en el hecho de tratar mejor los ruidos, ya que asigna al nuevo patrón a la clase mayoritaria entre sus vecinos y no sólo a la clase del más cercano, teniendo cierta tolerancia sobre este tema.

## ÁRBOLES DE DECISIÓN

### INTRODUCCIÓN

Los **árboles de decisión** son una herramienta de clasificación sin métrica que permite analizar decisiones secuenciales. Se basan en el uso de resultados y en probabilidades asociadas, y usualmente se los representa con notación de grafos para poder visualizarlos. La idea principal es que dado un conjunto de datos etiquetados ($X_n$, $Y_n$) con $1 \leq n \leq N$, donde $X_n \in R_n$ y $Y_n \in C$, C=1… c lo que se quiere es construir un modelo de clasificación en el cual a cada nuevo dato $X_n$ podamos asociarle una clase c.

Se puede optar por la construcción de árboles de decisión binarios, es decir cada nodo tiene exactamente 2 hijos o ninguno (las hojas), o árboles m-arios donde los nodos tienen más cantidad de hijos. Los primeros tienen la ventaja de que su construcción es más fácil de automatizar utilizando algoritmos computacionales, de todas formas es posible representar cualquier tipo de árbol cómo uno binario.

Los árboles de decisión son algoritmos inestables, es decir que pequeñas modificaciones en el conjunto de entrenamiento pueden generar grandes cambios en el clasificador. Esta afirmación se puede comprobar claramente al observar los

resultados que se obtienen cuando se vuelve a ejecutar el procedimiento de partición del conjunto de datos y construcción del árbol.

Por otra parte este tipo de clasificadores son intuitivos, ya que el proceso de decisión puede ser generado a partir de decisiones simples. Además para construir árboles de decisión se pueden utilizar tanto características cualitativas como cuantitativas y tienen la ventaja de que se pueden aplicar tanto a problemas discretos como continuos.

## PROCESO DE CLASIFICACIÓN

Para cada nodo interno (o nodo categórico) del árbol se realiza una "pregunta" acerca de una o varias características, esta tiene una cantidad de respuestas acotada y que además cada una de estas nos llevará a un nodo hijo distinto. El nodo hijo puede ser otro nodo categórico o un nodo hoja. Los nodos hoja son etiquetados con una única clase pero varios nodos hoja distintos pueden contener una etiqueta de la misma clase. Cuando un patrón alcanza uno de estos nodos se le asigna la clase del mismo.

Sabiendo esto, podemos ver que la clasificación consiste entonces en recorrer el árbol desde la raíz hasta alguna hoja a la que llegaremos contestando las preguntas que surgen en cada nodo interno. Estas preguntas suelen ser fórmulas matemáticas que combinan una o más características. La razón para utilizar fórmulas es que nos permiten automatizar el proceso de construcción del árbol, así como establecer límites claros de decisión en los que no cabe lugar para opiniones subjetivas o relativas. Lo que se logra con las preguntas es dividir el conjunto de datos en varias partes, en donde cada una de ellas corresponderá a algún nodo hijo. A su vez en los nodos hijos se puede volver a dividir la información en otros conjuntos más pequeños. Y así continúa el proceso hasta obtener el árbol completo. Luego se puede optar por "podar" el árbol para contrarrestar el sobre entrenamiento que se pudo haber generado, o de lo contrario se elige una función objetivo para determinar hasta donde seguirlo construyendo de forma de no llegar a resultados tan particulares.

Si el camino desde la raíz a los nodos siempre es del mismo largo podemos afirmar que estamos hablando de un árbol balanceado. Si esto no sucede se dice que son árboles no balanceados, esto nos da la pauta de que en algunas zonas la clasificación se puede lograr con una menor cantidad de preguntas que en otras zonas donde se necesita hacer un análisis más exhaustivo para poder clasificar el patrón (en general en los bordes o límites).

## IMPUREZA

Al dividir el conjunto de datos en varias particiones, como explicamos antes, lo que queremos es lograr que los conjuntos resultantes sean lo más puros posibles. Es decir queremos lograr la mayor homogeneidad. Para lograr esto utilizamos una función de impureza $\phi(n)$. $\Phi(n)$ está definida sobre c-uplas de la forma $(d_1, d_2, \ldots, d_c)$ con $0 \leq di$ y

$\Sigma di=1$. La impureza máxima se logra cuando todas las clases están representadas en la misma proporción y la impureza es mínima ($\phi(n)=0$) cuando todos los patrones pertenecen a una única clase. Entonces la impureza para un nodo está dada por $i(N)= \phi\ (P(W_1), P(W_2),\ldots, P(W_c))$ donde $P(W_i)$ es igual al cociente de elementos de la clase i, sobre la cantidad total de elementos.

## ALGORITMO C4.5

En Weka encontramos los algoritmos de clasificación de árboles de decisión en la sección "Trees". En nuestro caso utilizaremos el algoritmo J48 que corresponde al algoritmo de árboles C4.5 visto en el curso. En este algoritmo lo que se hace para cada nodo del árbol es elegir un atributo de los datos que divida el conjunto de muestras de forma más eficiente. Es decir que particione los datos en buenos subconjuntos de una clase u otra. El criterio que utiliza es el de ganancia de información o diferencia de entropía, que se utiliza para elegir el atributo que dividirá los datos. El atributo que posea la mayor ganancia de información normalizada se elige como parámetro de decisión. El algoritmo continúa de forma recursiva subdividiendo cada conjunto generado en el paso anterior.

Para este algoritmo hay 3 pasos base para la recursión, éstos son:

- Todas las muestras en la lista pertenecen a la misma clase. Cuando esto sucede, simplemente crea un nodo de hoja para el árbol de decisión diciendo que elija esa clase.
- Ninguna de las características proporciona ninguna ganancia de información. En este caso, C4.5 crea un nodo de decisión más arriba del árbol utilizando el valor esperado de la clase.
- Instancia de la clase previamente no vista encontrada. Una vez más, C4.5 crea un nodo de decisión más arriba en el árbol con el valor esperado.

Ahora presentamos un pseudocódigo del algoritmo de recursión:

1. Se verifican los pasos base.
2. Para cada atributo **a**
    1. Encontrar la ganancia de información normalizada de la división de **a**
3. Se elige el mejor **a** como el atributo con la ganancia de información normalizada más alta
4. Se crea un *nodo* de decisión que divida según el mejor **a** encontrado
5. Se repite el proceso para los subconjuntos generados a partir de la división anterior, y se agregan estos nodos como hijos.

# BAGGING

## INTRODUCCIÓN

Bagging (**B**ootstrap **agg**regat**ing**) es un clasificador que fue propuesto por Leo Breiman en 1994. Este se construye combinando la salida de varios clasificadores con el objetivo de generar uno más potente. Se utilizan como base clasificadores buenos pero inestables, en general árboles de decisión, pero se pueden utilizar otros modelos como ser redes neuronales. Es aplicable tanto a problemas de 2 clases como a aquellos multiclase, como es nuestro caso.

La idea de Bagging es generar a partir de un conjunto de entrenamiento D de tamaño n, m nuevos conjuntos de entrenamiento $D_i$ de tamaño $n' > n$. Estos nuevos conjuntos se construyen con la generación de muestras boostrap.

Estas muestras boostrap se construyen de forma randómica a partir del conjunto de entrenamiento original. Dependiendo de la forma en que se aplique el algoritmo, se puede utilizar todo el conjunto de entrenamiento o cierta parte de él para luego agregar más datos al azar, los cuales son réplicas de los originales.

Si el modelo predice una salida numérica, el algoritmo bagging se crea promediando las salidas de los distintos clasificadores. Si el modelo es de clasificación, la salida del clasificador combinado será aquella clase que resulte ser elegida por la mayoría de los diferentes clasificadores.

## ALGORITMO

Para poder aplicar el algoritmo de Bagging primero dividimos el conjunto de datos en un conjunto de entrenamiento D y otro de test T.

Etapa de entrenamiento

Para K=1 hasta L, siendo L la cantidad de clasificadores base

- Tomamos una muestra boostrap $D_k$ a partir de D.
- Construimos un clasificador $C_k$ con el conjunto de entrenamiento generado.
- Lo agregamos al metaclasificador.

Etapa de clasificación

1. Para cada clasificador guardar la clase seleccionada utilizando ese modelo.
2. Se devuelve la clase que fue elegida por la mayoría de los clasificadores.

Como podemos observar tanto las etapas de entrenamiento como las de clasificación para los algoritmos base, se pueden ejecutar en paralelo en distintos procesadores si es necesario.

### VENTAJAS

Este tipo de clasificador permite mejorar el porcentaje de aciertos con respecto a los clasificadores originales, así como mejorar el sobreajuste que pueden producir los algoritmos que éste combina y reducir la varianza.

### DESVENTAJAS

Como contracara debemos resaltar que este clasificador no es útil para mejorar modelos lineales, y tampoco permite mejorar el desempeño de clasificadores estables como por ejemplo k vecinos más cercanos. Además si el tamaño muestral es muy grande los clasificadores devuelven resultados muy similares y por tanto en general no vale la pena aplicar bagging.

### POR QUÉ FUNCIONA BAGGING?

Si las salidas de los clasificadores base son independientes y además los clasificadores tienen la misma precisión, la elección de clase por voto mayoritario asegura mejorar la performance con respecto a uno de estos clasificadores ejecutado de forma individual. Bagging apunta a construir clasificadores independientes basados en tomar muestras boostrap como conjuntos de entrenamiento.

## CLASIFICACIÓN

Pasamos a usar nuestro conjunto seleccionado de características y empleamos los diferentes algoritmos en Weka para estudiar el desempeño de los mismos en la tarea de clasificación de las imágenes.

### VECINOS MÁS CERCANOS

Creamos una tabla con diferentes valores de K y salidas de la prueba realizada con validación cruzada de 10 iteraciones.

| K vecinos | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **1** | 80.2761 % | 0.20 | 0.011 | 0.896 |
| **10** | 71.6634 % | 0.29 | 0.016 | 0.951 |
| **20** | 65.286 % | 0.35 | 0.02 | 0.953 |
| **50** | 54.2406 % | 0.46 | 0.026 | 0.937 |

*Promedios ponderados de la propiedad para cada clase.

Vemos que el mejor resultado se da con k=1, esto sucede ya que las imágenes se toman en momentos casi consecutivos y por lo tanto las coordenadas son muy similares entre imágenes de una misma persona. Al aumentar el valor de k el algoritmo toma también otras imágenes más lejanas para la clasificación y se da un mayor nivel de error en la misma como consecuencia.

### ÁRBOLES DE DECISIÓN

Nuevamente procedemos a realizar el proceso de clasificación pero esta vez utilizando árboles de decisión en Weka. Utilizamos el algoritmo Tree J48 como ya mencionamos en la sección árboles de decisión. En este caso utilizamos distintos porcentajes para los datos de entrenamiento y el resto los utilizamos como datos de test. Los datos de entrenamiento son elegidos al azar utilizando la herramienta proporcionada por Weka "percentage Split". Los resultados que obtenemos son los siguientes.

| Porcentaje de Entrenamiento | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **25%** | 51.2708 % | 0.487 | 0.028 | 0.776 |
| **50%** | 58.6842 % | 0.413 | 0.026 | 0.814 |
| **66%** | 62.2824 % | 0.377 | 0.023 | 0.836 |
| **75%** | 61.3158 % | 0.387 | 0.021 | 0.811 |

*Promedios ponderados de la propiedad para cada clase.

Vemos que al aumentar el tamaño del conjunto de entrenamiento (hasta cierto punto) también aumentamos el desempeño del clasificador para los datos de test. Esto se da porque en general, cuanto más muestras tenemos de las cuales aprender más datos tiene nuestro clasificador al entrenarse y por lo tanto tiene una mejor base para su predicción. De todas formas debe tenerse cuidado al entrenar este tipo de clasificador ya que los árboles de decisión son algoritmos muy inestables y por lo tanto suelen sobre ajustarse demasiado a los datos de entrenamiento, generándose árboles muy particulares que no representan de forma correcta la realidad. Éste podría ser el caso al utilizar un conjunto de entrenamiento que se compone del 75% de los datos, como podemos ver en la tabla anterior los resultados en este caso no son tan buenos como los antes obtenidos.

Ahora procedemos a correr el mismo algoritmo pero esta vez utilizando todos los datos para entrenar el árbol y validación cruzada de 10 iteraciones. En este caso se entrena y se realiza el test para los mismos datos.

| Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|
| **67.1926 %** | 0.672 | 0.019 | 0.854 |

*Promedios ponderados de la propiedad para cada clase.

Vemos esta vez que se obtiene aún un mejor desempeño que en los casos anteriores. Esto se debe a que se utilizaron todos los datos como entrenamiento y luego se testea sobre estos mismos, por lo tanto aunque el árbol esté sobre ajustado a nuestros datos al probarlos sobre sí mismos obtendremos buenos resultados. Es decir que si probamos nuestro clasificador con nuevos datos seguramente obtengamos un menor desempeño.

## BAGGING

Utilizamos el algoritmo de Bagging propuesto por Weka donde determinamos el tamaño de los subconjuntos de entrenamientos como un porcentaje del tamaño del conjunto de entrenamiento original, siendo por defecto el %100 en Weka. Tomamos como clasificador base el algoritmo C4.5 y como conjunto de entrenamiento el 66% del conjunto total dejando el resto para testing. Variando el número de iteraciones obtuvimos los siguientes resultados:

| Iteraciones | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **5** | 68.2785 % | 0.317 | 0.019 | 0.926 |
| **10** | 72.9207 % | 0.271 | 0.016 | 0.952 |
| **20** | 74.2747 % | 0.257 | 0.023 | 0.963 |
| **50** | 71.7602 % | 0.282 | 0.016 | 0.964 |

*Promedios ponderados de la propiedad para cada clase.

Vemos que el algoritmo presenta mejor performance que en el caso de C4.5, corroborándose que Bagging logra una mejor performance utilizando como base clasificadores buenos pero con cierta inestabilidad.

Un punto a destacar es que al incrementar las iteraciones la performance mejora pero después de cierto punto esto ya no es así. Al tratarse la clasificación por una "mayoría de votos" y por ser los árboles de decisión tan sensibles a los outliers es probable que a la larga tener muchos "votantes" no sea la mejor opción. Hicimos un estudio más exhaustivo y observamos que acontecía en iteraciones intermedias:

| Iteraciones | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **20** | **74.2747 %** | 0.257 | 0.023 | 0.963 |
| **25** | 73.8878 % | 0.261 | 0.015 | 0.964 |
| **30** | 73.6944 % | 0.263 | 0.016 | 0.965 |
| **35** | 75.0484 % | 0.250 | 0.015 | 0.966 |
| **40** | **75.8221 %** | 0.242 | 0.014 | 0.966 |
| **45** | 75.0484 % | 0.250 | 0.015 | 0.967 |
| **50** | 71.7602 % | 0.282 | 0.016 | 0.964 |

Efectivamente el algoritmo parece oscilar en su performance después de cierto punto; el error al aumentar las iteraciones se "estanca" como podemos ver en el siguiente gráfico:

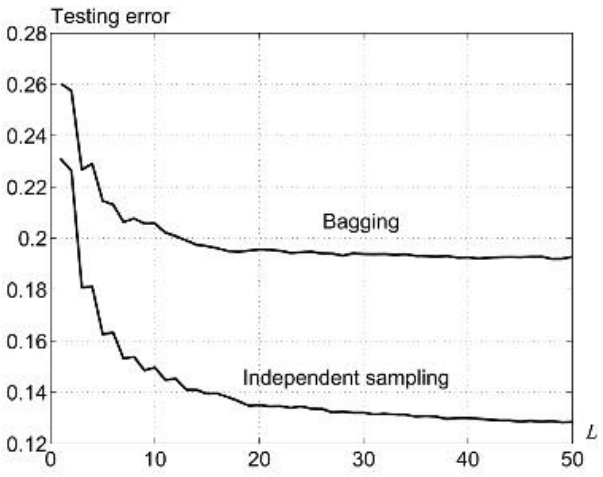



**Figura 4. Bagging – Error de testeo vs. Número de iteraciones**

## RESULTADOS

Observamos que en general se dio un muy buen desempeño a pesar del hecho mencionado antes de que las características en sí sean coordenadas de puntos en una imagen y no tendrían por qué representar efectivamente a una persona.

La explicación a este hecho como ya se mencionó en la clasificación con K Vecinos es que las fotos son tomadas casi simultáneamente, y por lo tanto cercanía en las características sí corresponde a una misma persona en la mayoría de los casos. Pero si se presentaran nuevas fotos tomadas en otro momento, seguramente no coincidan las coordenadas y no nos daría un buen desempeño.

A los efectos de nuestro objetivo, el cual es estudiar el desempeño de clasificadores y su comportamiento frente a la aplicación de ruido, no es de importancia el hecho de que al presentar nuevos datos el resultado obtenido sea positivo ya que sólo trabajamos con la base de datos brindada por BioID. Pero sí es de importancia destacar lo que podría suceder.

# CLASIFICACIÓN INDUCIENDO RUIDO

## INTRODUCCIÓN

Con el programa generado en Java que utilizamos para crear el archivo .arff de datos para Weka (a partir de los archivos adjuntos de características de la BioId face database), agregamos ruido a las muestras utilizando la clase Random ya que provee un método nextGaussian el cual devuelve un número randómico con distribución gaussiana de media 0 y varianza 1.

Decidimos colocar ruido sólo en los ojos y luego en todas las características para estudiar la reacción de los clasificadores. Luego colocamos un mayor nivel de ruido para ver cuánto se deterioraba la performance frente a este hecho.

Aplicamos los algoritmos a las muestras con ruido con los parámetros en los que mostraron mejor performance con las muestras originales.

## VECINOS MÁS CERCANOS

En el caso de K Vecinos más cercanos la mejor performance se dio para k=1, que es conocido como Vecino más cercano.

Recordamos que el valor obtenido en este caso fue:

| K vecinos | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **1** | 80.2761 % | 0.20 | 0.011 | 0.896 |

Volvemos a aplicar el algoritmo a las diferentes muestras alteradas y obtenemos el siguiente resultado:

| Ruido | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **Sólo ojos** | 80.1446 % | 0.199 | 0.011 | 0.895 |
| **Sólo ojos **** | 79.5529 % | 0.204 | 0.012 | 0.891 |
| **Todas las caract.** | 78.764 % | 0.212 | 0.012 | 0.887 |
| **Todas las caract.**** | 75.8711 % | 0.241 | 0.014 | 0.873 |
| **Nariz y Pera** | 80.2104 % | 0.198 | 0.011 | 0.896 |

**Mayor nivel de ruido.

Como prueba colocamos ruido solamente en la punta de la nariz y en la pera para ver qué tan relevantes son estas características, y como se observa en la tabla, la performance es prácticamente igual a la obtenida sin él. No es así en el caso de los ojos, que claramente son unas de las características más importantes a la hora de identificar a una persona.

Obviamente el mayor nivel de deterioro se da cuando todas las características presentan ruido, llegando a ser este de un poco más del %4.

## ÁRBOLES DE DECISIÓN

Ahora procedemos a realizar la clasificación con ruido para los árboles de decisión. Nuevamente utilizamos el caso en el que obtuvimos mejores resultados, éste consistía en utilizar validación cruzada de 10 iteraciones y todo el conjunto de datos para entrenamiento y test. El resultado de esta etapa sin aplicación de ruido fue el siguiente.

| Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|
| **67.1926 %** | 0.672 | 0.019 | 0.854 |

Ahora procedemos a probar este mismo algoritmo, en las mismas condiciones pero esta vez utilizando el conjunto de datos luego de aplicarle ruido, según la siguiente tabla.

| Ruido | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **Sólo ojos** | 64.1026 % | 0.359 | 0.020 | 0.834 |
| **Sólo ojos **** | 63.9053 % | 0.361 | 0.021 | 0.835 |
| **Todas las caract.** | 63.3794 % | 0.366 | 0.022 | 0.835 |
| **Todas las caract.**** | 60.4865 % | 0.395 | 0.023 | 0.818 |
| **Ambas cejas** | 64.2341 % | 0.358 | 0.020 | 0.844 |

**Mayor nivel de ruido.

Nuevamente comprobamos el deterioro que se produce en la performance al aplicar ruido. Esta vez el grado de deterioro es aún mayor que en el caso de k-vecinos, dado que en el peor caso, es decir cuando se aplicó ruido a todas las características, el grado de aciertos disminuyó casi %7.

Volvemos a realizar una prueba para ver la influencia de otras características distintas de los ojos. Esta vez aplicamos ruido a ambos puntos internos de las cejas, y podemos ver que estas características si bien son relevantes, no lo son tanto como los ojos, ya que la performance no se ve tan deteriorada.

## BAGGING

Por último repetiremos el análisis para el clasificador Bagging. El mejor resultado que obtuvimos a la hora de la clasificación fue para el caso de 20 iteraciones utilizando 66% de los datos para entrenamiento y como clasificadores base árboles C4.5, donde obtuvimos la siguiente performance.

| Iteraciones | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **20** | 74.2747 % | 0.257 | 0.023 | 0.963 |

Aplicando ruido al conjunto de datos para las mismas condiciones que en los casos anteriores, y utilizando también 20 iteraciones del algoritmo de bagging obtenemos los resultados que se observan a continuación.

| Ruido | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **Sólo ojos** | 72.533 % | 0.275 | 0.016 | 0.958 |
| **Sólo ojos **** | 72.149 % | 0.279 | 0.016 | 0.951 |
| **Todas las caract.** | 71.7602 % | 0.282 | 0.016 | 0.954 |
| **Todas las caract.**** | 71.1799 % | 0.288 | 0.016 | 0.956 |
| **Comisura Labios** | 72.9207 % | 0.271 | 0.015 | 0.959 |

**Mayor nivel de ruido.

Podemos ver que la performance se vuelve a degradar en comparación con la obtenida a partir de los datos originales. En este caso realizamos una prueba para ver el desempeño de aplicar ruido a los puntos que corresponden a las comisuras de los labios. Vemos que sucede algo similar a lo que sucedía con las cejas utilizando el algoritmo C4.5, las características si bien son relevantes, no lo son tanto como los ojos ya que su perturbación no afecta tanto en la performance del algoritmo.

Nuevamente el deterioro más significante lo apreciamos cuando aplicamos el mayor nivel de perturbación a todas las características, obteniendo una performance 3 puntos inferior a la original.

## BONUS

### CLASIFICACIÓN SIN SELECCIÓN DE CARACTERÍSTICAS

Estudiamos a continuación que acontece si dejamos de lado la etapa de selección de características; tomamos las 20 características propuestas por BioId face database y aplicamos los mismos algoritmos.

A priori el sentido común nos dice que la performance va a mejorar ya que se tiene más información a la que acudir para la clasificación. Pero esto puede no ser así, ya que si todas las características que obviamos no eran relevantes, entonces lo único que estaríamos induciendo es información redundante y enlenteciendo el tiempo de ejecución de los algoritmos sin lograr mejores resultados.

Pasamos a correr los algoritmos con los parámetros que presentaron mejor performance al igual que en el estudio del ruido. Y los comparamos con los valores de desempeño de los clasificadores en el caso de utilizarse selección de características los cuales eran:

| Algoritmo | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **K Vecinos** | 80.2761 % | 0.20 | 0.011 | 0.896 |
| **Árboles de decisión** | 67.1926 % | 0.672 | 0.019 | 0.854 |
| **Bagging** | 74.2747 % | 0.257 | 0.023 | 0.963 |

El desempeño obtenido para cada clasificador sin selección fue el siguiente:

| Algoritmo | Porcentaje de acierto | Falsos negativos* | Falsos positivos* | Área ROC* |
|---|---|---|---|---|
| **K Vecinos** | 80.8021 % | 0.20 | 0.010 | 0.900 |
| **Árboles de decisión** | 64.8258 % | 0.352 | 0.021 | 0.837 |
| **Bagging** | 74.4681 % | 0.255 | 0.015 | 0.965 |

## DISCUSIÓN

De estos resultados se desprenden muchos puntos a discutir y analizar.

- K Vecinos es levemente más eficaz, esto puede ser por el hecho de habernos obviado alguna característica relevante. La diferencia es muy pequeña por lo que habría que evaluar el uso que quisiera dársele al algoritmo para determinar si vale la pena aumentar la dimensionalidad del vector de características teniendo como contrapartida el hecho de lograr un algoritmo más lento.
- Árboles de decisión se deteriora notoriamente y esto se debe cómo habíamos expresado con anterioridad a que es muy susceptible a características irrelevantes. Por lo tanto la selección de características en este caso es necesaria y obligatoria ya que si no lo hacemos estamos creando un clasificador que consume más tiempo (al ser el árbol más grande) y poco eficaz.
- Bagging al igual que K Vecinos nos demuestra su robustez al no presentar mayores diferencias en performance comparado al caso con selección. El algoritmo presenta una leve mejora de performance pero que no se contrarresta al hecho de generar un clasificador más lento. Un hecho a destacar es que el número de falsos positivos disminuye notoriamente; dependiendo del uso que quisiera dársele al clasificador este puede ser un punto a favor para no realizar la selección de características, como ser el acceso de usuarios a un lugar restringido en el cuál es importante no dejar pasar a intrusos.

## CLASIFICACIÓN CON VECTOR DE CARACTERÍSTICAS ALTERNATIVO

Tal como mencionamos antes vamos a realizar otro análisis que consiste en generar un nuevo espacio de características, que intuitivamente represente mejor nuestro problema y nos permita obtener un desempeño mayor para los clasificadores utilizados.

Este nuevo conjunto consta de 3 características generadas para cada una de las imágenes de la base de datos a partir de las originales. La idea consiste en obtener características que permitan medir las posiciones relativas de las coordenadas dentro del rostro, y no posiciones de cada coordenada respecto al marco de la foto como teníamos anteriormente. Describimos cada una de ellas a continuación.

- Medida de la distancia entre ambas pupilas.
- Cociente de distancia entre ambos ojos y distancia desde la pupila derecha hasta la punta de la nariz.
- Cociente de distancia entre ambas pupilas y distancia desde la pupila izquierda hasta la punta de la nariz.

En dos de los casos elegimos realizar el cociente para cubrir casos de traslación hacia atrás y adelante en las distintas fotos. Es decir, para que las características sigan siendo muy similares en caso de que la persona aparezca muy cerca en una foto, y en la siguiente se encuentre muy alejada.

Ahora procedemos a evaluar este nuevo conjunto de datos para los 3 clasificadores analizados en nuestro proyecto.

### K-VECINOS

| K vecinos | Porcentaje de acierto |
|---|---|
| **1** | 49.1124 % |
| **10** | 54.7009 % |
| **20** | 53.1229 % |
| **50** | 47.8632 % |

### ÁRBOLES DE DECISIÓN

| Porcentaje de Entrenamiento | Porcentaje de acierto |
|---|---|
| **25%** | 44.8729 % |
| **50%** | 51.1842 % |
| **66%** | 52.6112 % |
| **75%** | 47.3684 % |

### BAGGING

| Iteraciones | Porcentaje de acierto |
|---|---|
| **5** | 52.8046 % |
| **10** | 55.5126 % |
| **20** | 54.352 % |
| **50** | 55.3191 % |

De forma contraria a lo que esperábamos, vemos que al aplicar los algoritmos al nuevo conjunto de características no mejoramos el desempeño, sino que este se ve deteriorado significativamente. Sin embargo, esto tiene sentido considerando que las imágenes que componen la base de datos cuentan con rostros de una misma persona posicionados de forma tal que las distancias tomadas varían. Por ejemplo, observamos que una de las personas aparece completamente de frente en una fotografía, y luego en otra aparece mirando un poco hacia arriba con el mentón levantado. Esto nos conduce a que la distancia de los ojos en ambas sea casi idéntica, pero la distancia de la nariz a los ojos cambie de forma importante, lo que nos lleva a cometer errores de clasificación.

## CONCLUSIONES

### SELECCIÓN

A partir del proceso de selección de características realizado pudimos ver que utilizando un conjunto más reducido que el original, podemos obtener desempeños muy buenos, que hasta podrían superar la performance que se logra al utilizar todas las características.

En problemas de esta índole existen distintos “tipos” de características las cuales podemos dividir en los siguientes grupos. Están aquellas muy relevantes, las cuales al

eliminarse degradan la performance significativamente. Luego, existen otras características que no afectan en el resultado final; es decir éstas últimas son irrelevantes a la hora de la clasificación y por tanto pueden eliminarse sin que ésto afecte en lo más mínimo. Por último podemos identificar características que aplicadas sobres ciertos clasificadores afectan negativamente la performance.

Por tanto creemos que en la mayoría de los casos es fundamental realizar el proceso de selección de características para lograr identificar aquellas realmente significativas e influyentes y almacenarlas para utilizar a la hora de la clasificación, y a su vez eliminar todas aquellas características redundantes y las que no aportan información o degradan la performance. Así obtenemos un subconjunto derivado del original sobre el cuál se aplican distintos algoritmos de clasificación. Esto nos brinda mejores tiempos de respuesta y eficiencia al tratar con un conjunto de menor dimensión.

## CLASIFICACIÓN

La clasificación realizada nos mostró cómo se comportan los distintos algoritmos de clasificación al aplicarse sobre un mismo conjunto de características. Vemos que la clasificación utilizando árboles de decisión fue la más ineficiente, donde obtuvimos porcentajes de aciertos inferiores al 70% en todos los casos, más precisamente rondaban entre el 50% y el 68%. Esto es natural considerando que la base de datos sobre la que trabajamos no es tan extensa, además de tener en cuenta que este tipo de clasificador es muy inestable y susceptible a pequeñas variaciones en el conjunto de datos.

Luego tenemos el algoritmo de k-vecinos en el cual obtuvimos porcentajes de acierto que van desde el 54% al 80%. Vemos que comparado a los árboles de decisión y bajo ciertas circunstancias logramos resultados mucho mejores. Esto concuerda con el análisis teórico realizado en el cuál vimos que k-vecinos y vecinos más cercanos son algoritmos más estables.

Por último el algoritmo de Bagging nos brindó una performance en torno del 68% y el 75%. Si bien no logra alcanzar la performance obtenida con k-vecinos vemos que supera ampliamente a la de los árboles de decisión. Esto es muy importante ya que el algoritmo de bagging se compone de éstos, lo cual confirma la mejora significativa que se logra al construir este metaclasificador.

## CLASIFICACIÓN APLICANDO RUIDO

De los resultados obtenidos en esta etapa del análisis podemos ver que los algoritmos de k-vecinos y bagging son más robustos que los árboles de decisión. Esto se debe a que al aplicar ruido la degradación de la performance de este último algoritmo fue bastante mayor que la de los otros dos, dando la pauta de que es más susceptible ante cambios en las características. Este resultado es natural sabiendo que los árboles de decisión son algoritmos muy inestables, mientras que k-vecinos es más estable.

Por otra parte si bien bagging se construye a partir de árboles, vemos que logramos mejorar la inestabilidad de estos, permitiendo construir un algoritmo mucho más estable y robusto.

Aquí también pudimos ver que dentro de las características que elegimos en la etapa de selección, los ojos resultaron ser las más relevantes, ya que al perturbarlas, la performance se degrada de forma mucho mayor que si se aplica esta misma distorsión sobre otras características.

# BIBLIOGRAFÍA Y HERRAMIENTAS

## BIBLIOGRAFÍA


- [1] http://es.wikipedia.org/wiki/Sistema_de_reconocimiento_facial
- [2]BioID Face Data Base - https://www.bioid.com/download-center/software/bioid-face-database.html
- [3] Duda, Hart, Pattern Classification (2nd Edition) Wiley
- [4] Paper k-Nearest Neighbour Classifiers - Pádraig Cunningham and Sarah Jane Delany - http://www.csi.ucd.ie/files/UCD-CSI-2007-4.pdf
- [5] Paper - How effective are landmarks and their geometry for face recognition? - J. Shi a, A. Samal a, D. Marx b. - 2005
- Notas y transparencias del curso.
- Combining Pattern Classifiers - Ludmila I. Kuncheva.
- http://www.mitecnologico.com
- http://www.dia.fi.upm.es/~concha/AA_temaB10_metaclasificadores_concha2.pdf
- http://www.maia.ub.es/~sergio/linked/julian08.pdf
- http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/tema13s.pdf


## HERRAMIENTAS

- Weka
- BioID Face Data Base