Informe Proyecto Final
Clasificación de bosques por información cartográfica

Matías Osorio Mirambell - mosorio@fing.edu.uy
Santiago Silva Dalla Rizza - smsilva@antel.com.uy
Tutores: Pablo Musé - Sergio Martínez
7 de diciembre de 2015

Problema a resolver

El problema a resolver consiste en clasificar el tipo mayoritario de árboles en bosques nativos. El estudio incluye cuatro áreas de bosques naturales situados en el norte del estado de Colorado, EEUU. Los siete tipos de árboles se observan en la Figura1. La base de datos se llama *coverttype* y se encuentra disponible en https://archive.ics.uci.edu/ml/datasets/Covertype.

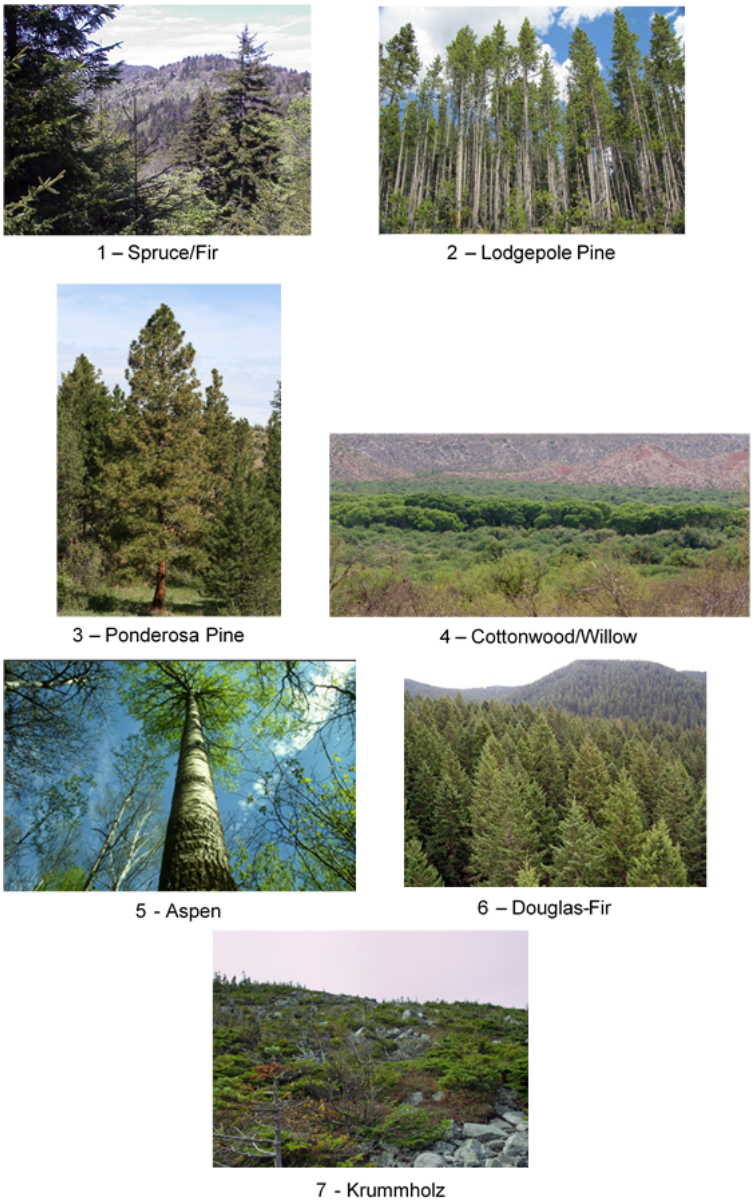

Figura 1: Tipos de bosques a ser clasificados.

Se cuenta con un conjunto de entrenamiento de 7560 observaciones conteniendo las características con las que se trabajará y el tipo de árbol. La descripción de las características se presenta en el Anexo A.

El problema requiere obtener un 82 % de desempeño, además de observar que sucede con el desempeño cuando se modifican variables y se agregan otras nuevas.

Enfoque

El enfoque utilizado fue usar Weka en primer lugar y luego migrar a MatLab. Weka permite realizar test rápidos y sin realizar programación, tanto a los características como a conjuntos de ella. Además tiene librerías bastante completas con herramientas muy buenas. Por lo tanto, el primer paso a seguir fue aprovechar dichas implementaciones para encontrar cuales serían los clasificadores con mejor desempeño para este conjunto de datos. Luego de esto, se migró a MatLab, implementando dichos clasificadores y así tener más libertad y facilidad en el editado de los datos y en las gráficas, así como también para la creación de nuevas características y substitución de otras.

Weka

Los clasificadores utilizados en Weka se presentan en la Tabla 1. Dicha tabla además es acompañada con la información del tipo de características utilizadas, dado que algunas son binarias. El tipo de características utilizadas se encuentra bajo la columna **Carac. usadas**, y señala los tests que se realizaron utilizando solo las características binarias o las características no binarias.

Clasificador	**Carac. usadas**	**Acierto (%)**
J48	B	58.3
J48	NB	72.9
J48	T	77.6
KNN (K=1)	T	78.8
KNN (K=2)	T	75.7
KNN (K=5)	T	75.9
Naive Bayes	T	66.0
SVM	T	59.5
SVM + PCA	T	74.9
Random Forest (100 trees)	T	83.7
Random Forest + PCA (100 trees)	T	80.1

Tabla 1: Resultados obtenidos con Weka bajo el conjunto de entrenamiento utilizando validación cruzada. Las características que se usaron fueron solo las binarias (B), solo las no binarias (NB) y todo el conjunto (T).

De acuerdo a los resultados obtenidos y mostrados en la Tabla 1, se observa claramente que el único clasificador que supera el mínimo requerido (82 %) es Random Forest. Se observó además (en la salida de Weka) que en la gran mayoría de los casos investigados, la matriz de confusión obtenida muestra que la clase 1 y 2 se confunden bastante, al igual que las clases 3 y 6. Esto puede observarse de alguna manera en la Figura 2. Para este caso las características 2, 3 y 4 hacen que los puntos queden mezclados. Realizando observaciones sobre todo el espacio de características, la única que tiene algún grado de discriminación de clases de manera fácil a simple vista es la característica 1.

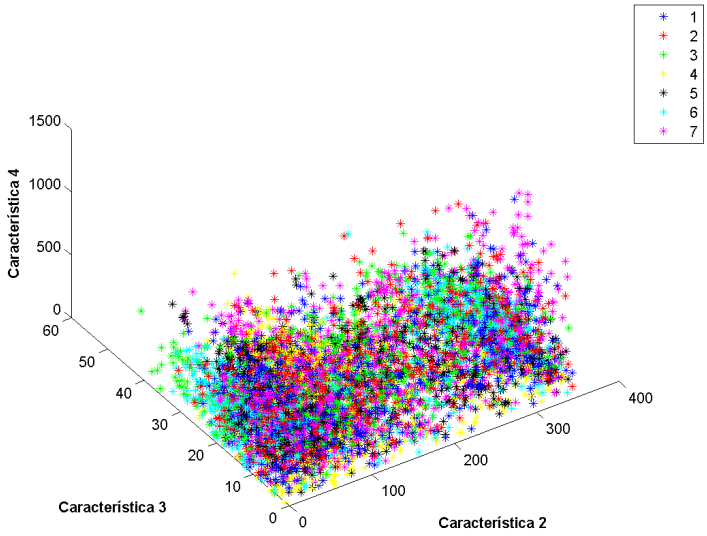

Figura 2: Observaciones en función de las características originales 2, 3 y 4.

Algoritmos a utilizar

Random Forest

Random Forest es un algoritmo de clasificación en el cual se construyen múltiples árboles de decisión en la etapa de entrenamiento, y las etiquetas de las distintas clases son designadas a la salida por medio de la moda de la clasificación o realizando un promedio de distintos árboles individuales (*bagging*)[1][2].

La idea esencial del *bagging* es promediar muchos modelos ruidosos pero aproximadamente imparciales, y por tanto reducir la variación. Los árboles son los candidatos ideales para el *bagging*, dado que ellos pueden registrar estructuras de interacción compleja en los datos, y si crecen suficientemente profundo, tienen relativamente baja parcialidad. Producto de que los árboles son notoriamente ruidosos, ellos se benefician grandemente al promediar[3]. Cada árbol es construido usando el siguiente algoritmo:

1. Sea N el número de casos de prueba, M es el número de variables en el clasificador.
2. Sea m el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; m debe ser mucho menor que M.
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir de manera aleatoria m variables en las cuales basar la decisión. Calcular la mejor partición a partir de las m variables del conjunto de entrenamiento.

Para la predicción un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción.

Cuando se utiliza Random Forest, hay que considerar que los datos pueden llegar a estar sobreajustados, caso que sucede a menudo con este tipo de algoritmo.

Principal Components Analysis (PCA)

PCA es un método de no supervisado que se utiliza básicamente para extraer características, simplificando así un conjunto de datos, y donde este nuevo conjunto de variables es una combinación lineal de las anteriores y son no correlacionadas. Geométricamente esto puede interpretarse como una rotación de ejes a un nuevo sistema de coordenadas, donde este nuevo sistema está definido por direcciones propias, maximizando en estas direcciones la varianza.

Básicamente lo que se realiza es un procedimiento en el cual, por medio de una transformación ortogonal, se convierte el espacio de observaciones, a un espacio de variables no correlacionadas denominadas componentes principales. Esta transformación se define de tal manera que la primer componente principal, sea la componente con mayor varianza; la segunda componente principal la de segunda mayor varianza y así sucesivamente.

Implementación en MatLab

La implementación de Random Forest en MatLab se dió a través de la librería nativa *TreeBagger*[4] del propio programa. Se probó además una librería llamada *randomforest-matlab*, que al momento de probarla daba prácticamente los mismos resultados que *TreeBagger*, por lo que se decidió por la implementación nativa.

Se desarrolló en MatLab un programa *main.m* el cual realiza la carga de datos a partir de los archivos provistos, realiza un pre-procesamiento de los datos (será explicado más adelante este punto), particiona en conjunto de entrenamiento y test, llama a la función *nuevasCaracteristicas.m* que define nuevas características a partir de las brindadas, y realiza el entrenamiento para los métodos implementados. A lo último se obtiene el resultado final entrenando con el conjunto correspondiente brindado y evaluando con el conjunto de test *Recpat_2015_test.csv*.

Resolución

A continuación se presentan diversos puntos que se siguieron para la resolución del problema.

Estudio de características y clasificación sin selección

En primer lugar se estudiaron y entendieron las características que se iban a utilizar. Esto es importante, sobre todo, al momento de crear nuevas y modificar las existentes ya que debe haber cierta coherencia entre la observación y la realidad.

Este problema tiene 54 características a ser utilizadas. De antemano se puede pensar que a mayor cantidad de características, mejor puede llegar a ser el desempeño del clasificador, pero en la práctica esto no sucede así. De acuerdo a lo estudiado, una buena práctica es la relación $n > 10d$, siendo n la cantidad de muestras de la clase y d la cantidad de características. En este caso se tiene un valor de $d = 54$, por lo que la cantidad de muestra de cada clase debería ser mayor a 540. Para este problema, la clase con menos instancias (clase 2) tiene 910, por lo que debería ser posible resolver el problema con la información brindada.

A su vez, se observó el balanceo entre clases en el conjunto de entrenamiento, y se concluyó que las mismas se encuentran balanceadas bastante bien, debido que las clases 1,3,4,5,6 y 7 tienen entre 1087 y 1153 muestras. La única que podría estar desbalanceada es la clase 2 con 910 muestras, correspondiendo a un desbalance entre 19 % y 27 % como límites. Se decidió entonces no utilizar ninguna técnica de balanceo porque se consideró que a pesar de este pequeño desbalance, no se afectaría en mucho el desempeño del clasificador.

Como primer paso a la resolución del problema se aplicó el algoritmo *TreeBagger* a los datos de entrenamiento definiendo un 80 % del total como el conjunto de entrenamiento y el 20 % restante como conjunto de test, tomados de manera aleatoria a través de la función *randperm* de MatLab, obteniendo como resultado un 98.7 % de aciertos para el conjunto de entrenamiento y un 83.4 % para el conjunto de test[1]. Para esta implementación de Random Forest se utilizaron 800 árboles, y el desempeño del clasificador a través de la variación de distintos parámetros se verá más adelante.

Para intentar mejorar el desempeño se realizaron normalizaciones a los datos. Estas normalizaciones las sufrieron solamente las características no binarias, dado que las binarias ya están normalizadas. Se realizaron dos tipos de normalizaciones:

1. Normalización con máximo: Se divide cada observación de la característica i, sobre el máximo de la característica i.

2. Normalización con media y varianza: Se resta cada observación de la característica i, el promedio de dicha característica, y al resultado se lo divide entre la varianza de la característica i (proceso de blanqueado).

La manera que se diseñó para encontrar que tipo de normalización es la óptima fue clasificar mediante el algoritmo de Random Forest, con un total de 200 árboles y realizando validación cruzada en 10 corridas. Para esta implementación y al normalizar con máximo, se obtuvo un porcentaje de aciertos de 98.6 % en el conjunto

[1] Los resultados obtenidos se realizaron mediante validación cruzada en 10 partes. Este procedimiento se repitió para los siguientes resultados.

de entrenamiento y un 80.5 % en el conjunto de test, mientras que cuando se implementó lo mismo pero con normalización mediante media y varianza se obtuvo un porcentaje de aciertos de 98.6 % para el conjunto de entrenamiento y 82 % para el conjunto de test. Por lo tanto, para el resto de las pruebas se consideraron los datos sin normalizar y normalización mediante media y varianza, dado que son los métodos que superan el mínimo requerido.

Un aspecto importante de las características se pudo observar a través de la importancia que Random Forest le da a las mismas al momento de clasificar. En la Figura 3 se ve que las características que tienen mayor decisión en la clasificación son las no binarias (número 1 a 10). Sin embargo al momento de tratarlas por separado, el desempeño disminuye considerablemente como se pudo observar en la Tabla 1. Es por esto que se decidió (y será explicado en detalle más adelante) construir nuevas características a partir de las existentes, siguiendo algún tipo de lógica en la construcción. Luego, se intentará reducir la dimensionalidad del nuevo set y verificar que los desempeños siguen siendo bastante similares.

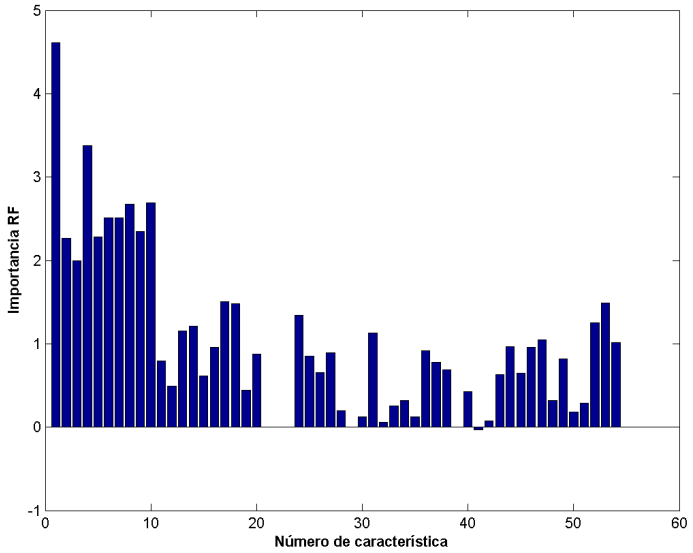

Figura 3: Importancia de las características obtenidas en el entrenamiento mediante Random Forest.

Desempeño de Random Forest

Otro aspecto que se estudió fue el desempeño de Random Forest a través de la modificación de distintos parámetros como el número de árboles a construir (*NumTrees*) y la cantidad mínima de observaciones en cada nodo del árbol, tomadas al azar (*NVarToSample*), siendo este valor por defecto la raíz cuadrada del número de características que se dan. Los datos que se utilizaron para realizar dichos test fueron los originales (sin agregado de nuevas características) y se utilizaron sin normalizar.

Variación de parámetro *NumTrees*

El parámetro *NumTrees* se varió de acuerdo a la Tabla 2, donde además se muestran los resultados obtenidos:

De acuerdo a los resultados observados en la Tabla 2 se encuentra que **el número de árboles óptimos es de 900**, viendo que los aciertos en el conjunto de entrenamiento son muy buenos (casi 100 % de aciertos), mientras que el desempeño en el conjunto de test, si bien disminuye un poco, se mantiene dentro de los mínimos requeridos.

Variación de parámetro *NVarToSample*

De la misma manera que se varió el parámetro *NumTrees*, se cambió el parámetro *NVarToSample* que controla la cantidad de muestras mínimas que tiene cada nodo del árbol. Este parámetro por defecto se encuentra en 1 y se varió desde 1 hasta 10. El estudio se realizó bajo el mismo conjunto de características que se utilizaron cuando se varió el parámetro *NumTrees*, tomándose un valor de 200 en dicho parámetro para que el costo computacional no sea excesivo. Los resultados obtenidos se muestran en la Tabla 3.

NumTrees	Acierto entrenamiento (%)	Acierto test (%)
200	98.6	82.2
300	98.7	82.9
400	98.7	81.9
500	98.7	82.2
700	98.7	82.6
900	98.9	83.0
1000	98.8	82.6

Tabla 2: Resultados obtenidos variando el parámetro *NumTrees* sin agregar características, ni normalizar, utilizando validación cruzada.

NVarToSample	Acierto entrenamiento (%)	Acierto test (%)
5	92.9	80.2
6	95.6	81.0
7	98.6	82.2
8	98.6	81.8
9	99.2	82.7
10	99.6	83.4
11	99.9	82.8
12	100	83.5
13	100	83.4
14	100	83.1

Tabla 3: Resultados obtenidos variando el parámetro *NVarToSample* sin agregar características, ni normalizar, utilizando validación cruzada.

Se puede concluir de lo anterior que los mejores parámetros para entrenar, se dan cuando se usa el algoritmo **Random Forest con 900 árboles y 12 muestras mínimas que tiene cada nodo del árbol**.

Creación de nuevas características

Para intentar aumentar el desempeño de la clasificación, se procedió a crear nuevas características a partir de las existentes. Las características creadas fueron las siguientes:

- elev_hydroVert_sum = Elevation + Vertical_Distance_To_Hydrology
- elev_hydroVert_resta_abs = |Elevation - Vertical_Distance_To_Hydrology|
- elev_hydroHori_sum = Elevation + Horizontal_Distance_To_Hydrology
- elev_hydroHori_resta_abs = |Elevation - Horizontal_Distance_To_Hydrology|
- pend_hydro = $\sqrt{\text{Horizontal\_Distance\_To\_Hydrology}^2 + \text{Vertical\_Distance\_To\_Hydrology}^2}$
- horiHydro_plus_horiFire_cuad = $\text{Horizontal\_Distance\_To\_Hydrology}^2 + \text{Horizontal\_Distance\_To\_Fire\_Points}^2$
- horiHydro_resta_horiFire_abs = |Horizontal_Distance_To_Hydrology-Horizontal_Distance_To_Fire_Points|
- horiHydro_plus_horiRoad = Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways
- horiHydro_plus_horiRoad_cuad = $\text{Horizontal\_Distance\_To\_Hydrology}^2 + \text{Horizontal\_Distance\_To\_Roadways}^2$
- horiHydro_resta_horiRoad_abs = |Horizontal_Distance_To_Hydrology-Horizontal_Distance_To_Roadways|
- horiFire_plus_horiRoad = Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways
- horiFire_plus_horiRoad_cuad = $\text{Horizontal\_Distance\_To\_Hydrology}^2 + \text{Horizontal\_Distance\_To\_Roadways}^2$

- horiFire_resta_horiRoad_abs = |Horizontal_Distance_To_Hydrology-Horizontal_Distance_To_Roadways|
- horiHydro_plus_horiFire = Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Fire_Points
- hill9_over_elev = Hillshade_9am / Elevation
- hillNoon_over_elev = Hillshade_Noon / Elevation
- hill3_over_elev = Hillshade_3pm / Elevation
- sin_aspect = $\sin$(aspect)

Estas características fueron creadas en función de que se consideraron aspectos de manera intuitiva como por ejemplo, que puede haber especies que crezcan en lugares más altos y alejados de puntos de agua. Además puede ocurrir que la actividad humana modifique el crecimiento de alguna especie, por lo que sería lógico pensar que haya especies que crezcan o se adapten mejor, cuanto más lejos de una carretera se encuentren. También la altura puede estar relacionada con el tipo de terreno y así se podría llegar a discriminar entre clases. A su vez, se escogió la característica *Elevation* como una de las principales a ser sumada o restada frente a otras, dado que el algoritmo de Random Forest la sitúa como la característica más importante antes de agregar nuevas.

Cuando estas son implementadas y se clasifica mediante el algoritmo de Random Forest, se llega a obtener un 100 % de aciertos en el conjunto de entrenamiento y un 85.2 % de aciertos en el conjunto de test. Una de las principales preocupaciones en este punto es que el método puede estar realizando un sobreajuste a los datos, explicando así el resultado que se obtuvo para el conjunto de entrenamiento.

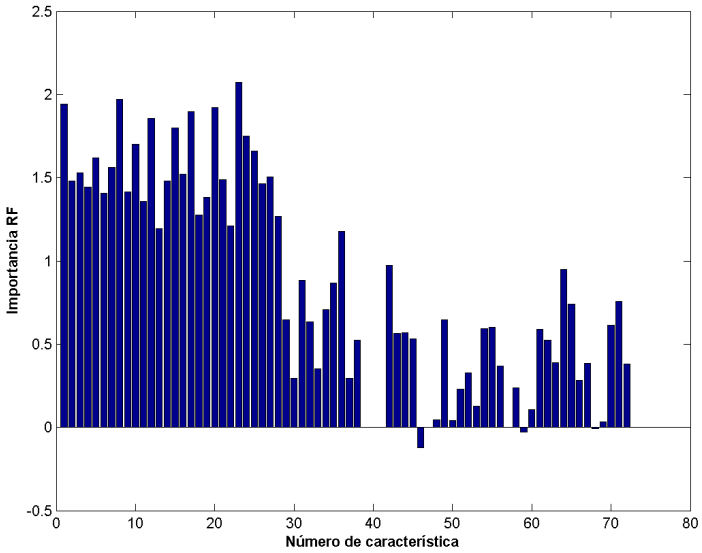

Figura 4: Importancia de las características obtenidas en el entrenamiento mediante Random Forest.

A través de la Figura 4 se puede observar que las características que más discriminan con Random Forest son las no binarias, y en gran mayoría, las que se crearon. Las 15 primeras características en importancia en este caso son: horiFire_resta_horiRoad_abs, horiHydro_resta_horiFire_abs, elev_hydroVert_resta_abs, Elevation, pend_hydro, Hillshade_Noon, horiHydro_resta_horiRoad_abs, hill3_over_elev, hill9_over_elev, Horizontal_Distance_To_Fire_Points, Horizontal_Distance_To_Hydrology, horiHydro_plus_horiFire, Hillshade_9am, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Roadways. A modo de ejemplo, en la Figuras 5 y 6, se puede observar que las tres primeras características brindan una separación más visible en los datos, haciendo que sea mejor la clasificación.

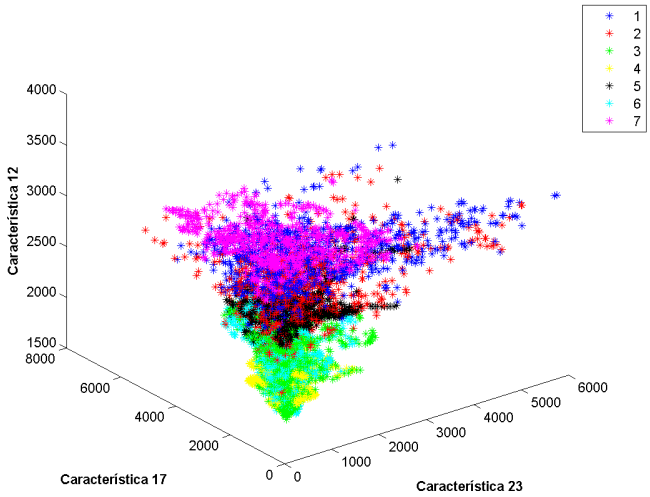

Figura 5: Conjunto de entrenamiento en función de las tres características con mayor importancia según Random Forest.

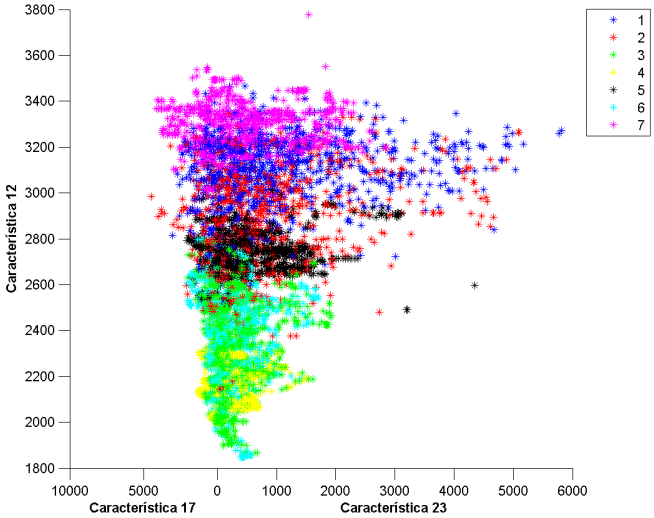

Figura 6: Conjunto de entrenamiento en función de las tres características con mayor importancia según Random Forest

PCA

Se intentó mediante PCA[2] transformar el espacio de características a utilizar, en un espacio que maximice las varianzas de cada una de ellas, tanto para cuando se agregan características, como cuando no se hace. Esto permitiría entonces escoger el mejor conjunto de características, intentando reducir el espacio lo más posible. La información para cumplir con dicho fin y establecer un mínimo de características a utilizar, se extrajo de la gráfica de varianza acumulada en función del número de características para cada caso.

En la Figura 7 se observa la varianza acumulada con PCA del set de características originales, sin normalizar. Se puede ver claramente que las primeras 7 características satisfacen el criterio de tener más de 95 % de la varianza acumulada, por lo que se pueden escoger dichas características y así reducir la dimensionalidad del problema. Sin embargo, cuando se grafican juntas las primeras tres características escogidas, se puede observar que las mismas no se separan como uno pretendería, continuando las observaciones de distintas clases bastante juntas entre sí. Esto último se puede observar en las Figuras 8 y 9.

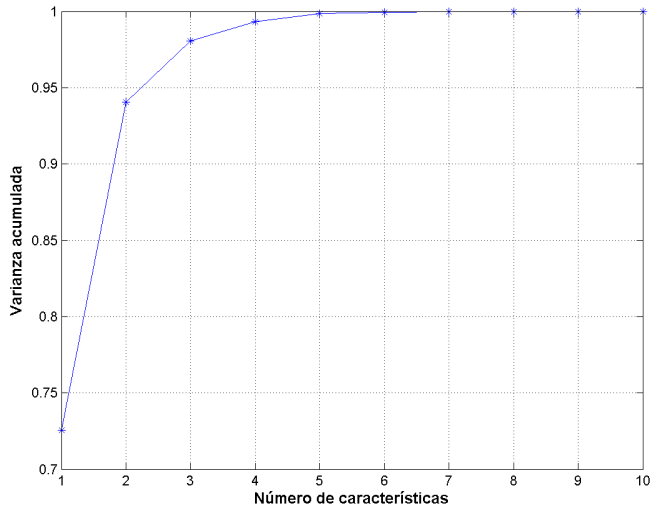

Figura 7: Varianza acumulada en PCA.

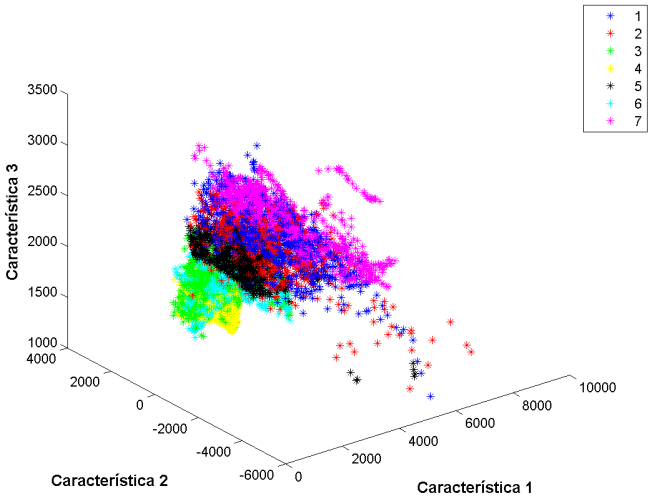

Figura 8: Primeras tres mejores características de PCA.

[2]Esta parte se implementó en Weka y MatLab, debido a que los algoritmos que se desarrollaron en MatLab tienen un costo computacional mucho mayor a los implementados en Weka. Debido a que las operaciones de PCA son simples (hallar matrices de valores propios y transformar el subespacio multiplicando por vectores), se realizaron las gráficas directamente en MatLab.

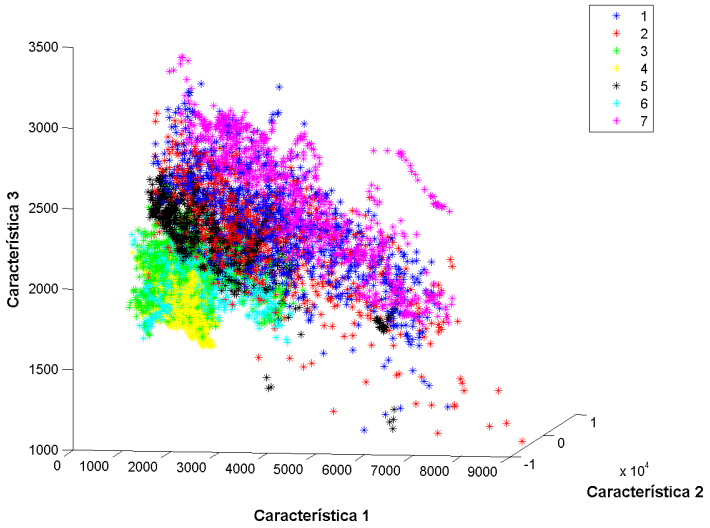

Figura 9: Primeras tres mejores características de PCA.

Implementando el clasificador Random Forest con PCA en Weka, se obtuvieron los resultados de la Tabla 4.

Características	**Normalización**	**Caract. PCA**	**Caract.@Var. Acum. = 1**	**Acierto (%)**
Originales	NO	7	7	74.9
Originales	NO	15	7	79.6
Originales	NO	25	7	80.2
Originales	SI (Media y Varianza)	10	9	78.4
Originales	SI (Media y Varianza)	20	9	79.9
Originales + Agregadas	NO	3	2	58.7
Originales + Agregadas	NO	10	2	82.6
Originales + Agregadas	NO	20	2	82.9
Originales + Agregadas	SI (Media y Varianza)	15	15	83.0
Originales + Agregadas	SI (Media y Varianza)	20	15	83.2
Originales + Agregadas	SI (Media y Varianza)	30	15	83.1

Tabla 4: Resultados obtenidos entrenando con Random Forest + PCA. La columna **Caract. PCA** representa la cantidad de características de PCA que se utilizaron. La columna **Caract.@Var. Acum. = 1** lleva la cuenta de la mínima cantidad de características para la cual la varianza acumulada vale 1.

De la Tabla 4 se puede leer que PCA no mejora sustancialmente el desempeño del clasificador, respecto a lo que se venía observando. Sin embargo, permite reducir la dimensionalidad considerablemente: de un espacio original de características originales y agregadas de 73, se pasa a un espacio de 30 características sin perder demasiado desempeño y cumpliendo con el mínimo requerido.

En este punto cabe preguntarse si las nuevas características están aportando de manera correcta al correcto desempeño del clasificador. Para ello se realizó el siguiente procedimiento de test en Weka:

1. A través del evaluador *CfsSubsetEval* se evaluaron todas las características. A través de este método es posible encontrar el subconjunto de características que presenta una mayor correlación entre las clases y que a la vez las muestras del subconjunto estén lo menos correlacionadas posibles entre sí. Para este caso las características que se seleccionaron fueron las número 1,11,12,13,14,23,25,26,31,35,40,41,43,53,61,66,68,69 y 70, bajo el criterio de que si en la validación cruzada, seis o más folders utilizan la característica se mantiene; de lo contrario se eliminará.

2. Se filtró el conjunto de entrenamiento para que solo queden dichas características a través del filtro *Remove*.

3. Se procedió a evaluar mediante Random Forest (es decir, realizar las mismas clasificaciones que al principio), pero con el parámetro de características reducido. A modo de ejemplo y para no ser tan extensos, se encontró que el desempeño utilizando el conjunto de características normalizadas con media y varianza fue de 77.2 %.

Este último desempeño, si bien no llega al mínimo que se requiere, indica que las características creadas de manera artificial (se utilizan un total de siete en un total de 19 al reducir) logran un buen desempeño y disminuye de gran manera la dimensión del problema, lo cual es un buen recurso cuando existen problemas con la dimensionalidad y la regla práctica $n > 10d$.

Desempeño de conjunto de test final

Para comprobar el desempeño del clasificador, fue proporcionado un archivo de test para el cual se probó el mejor clasificador obtenido en la etapa de entrenamiento. Este fue el Random Forest con 900 árboles y 12 observaciones mínimas por nodo. Se probó bajo dos instancias, siempre creando las nuevas variables definidas más arriba: 1) Normalizando los datos a través de media y varianza; 2) Sin normalizar los datos.

Para el caso de la normalización con media y varianza se obtiene un acierto de 83.7 %, mientras que para el caso sin normalización se obtuvo un acierto de 85.5 %, cumpliendo así los mínimos requeridos.

Un ejemplo de matriz de confusión para una de las corridas cuyo desempeño fue 85.5 % se puede observar en la Tabla 5.

1	*2*	*3*	*4*	*5*	*6*	*7*	
771	158	0	0	11	3	64	*1*
186	669	25	0	84	35	8	*2*
0	5	837	44	17	104	0	*3*
0	0	16	979	0	12	0	*4*
1	38	13	0	939	16	0	*5*
0	3	86	27	10	881	0	*6*
32	1	0	0	0	0	974	*7*

Tabla 5: Matriz de confusión obtenida de una corrida de la clasificación sin aplicar normalización.

Se puede ver que las clases problemáticas en la clasificación son las clases 1 y 2, mientras que las demás no tienen demasiada interferencia con otras.

Conclusiones

En el presente trabajo se realizó una clasificación de tipo de suelos mediante información cartográfica a partir de la base de datos *coverttype*. Dicha clasificación tenía que lograr al menos un 82 % de aciertos.

En primer lugar se estudiaron las características observando que, si bien se encuentran en un número tal que la regla de buena práctica para que no aparezca la maldición de la dimensionalidad se cumple, no se obtienen los desempeños mínimos requeridos con los clasificadores comunes, excepto solo con Random Forest. Pese a eso, los resultados obtenidos fueron bastante coherentes con la bibliografía que se encontró en la web y que trabajaban sobre esta misma base de datos. Blackard y Dean[5] obtuvieron un desempeño de aproximadamente 70 % utilizando redes neuronales en un trabajo temprano en 1999. A su vez Bagnall[6] probó diferentes aproximaciones mediante SVM, redes neuronales, K-vecinos, árboles, entre otros, obteniendo desempeños en conjuntos de test entre 67 % y 84 %, obteniendo el máximo con el algoritmo C5. Otro trabajo interesante es el de Trebar y Steele[7], donde se utlizan métodos de SVM en cascada, obteniendo desempeños por arriba de 97 %. Cabe destacar que este tipo de métodos superan ampliamente lo que se pretendía lograr en este trabajo, pero es un buen ejemplo de como proceder para tener mejores resultados.

Con respecto a las características creadas, las mismas cumplieron con la función de aportar más información al problema y discriminar mejor. Observando la Figura 6 ya se ve que las mejores tres nuevas características realizan una especie de estiramiento en dicho espacio, haciendo que las muestras puedan ser un poco mejor clasficadas. Si bien no todas las características se usan (como se vió al filtrar), una gran parte de ellas si lo hace, y se obtiene además buenas clasificaciones cuando se reduce la dimensionalidad mediante PCA utilizado con *CfsSubsetEval*. También se observó que las características binarias no aportan demasiada información para discriminar entre clase, pero cuando se las une al resto aportan entre un 1 % y 2 % de mejora en el desempeño de todos los clasificadores probados.

Un aspecto que se estudió, pero debería ser realizado nuevamente, es la obtención de los parámetros óptimos para la construcción del clasificador Random Forest. El mayor problema aquí es el costo computacional que

requiere el entrenamiento (corridas de treinta minutos aproximadamente en árboles mayores a 700 nodos). Otro problema que se encontró al momento de realizar esta búsqueda, fue que inicialmente se intentó hacerlo en Weka pero la memoria interna del programa se llenaba a partir de árboles con tamaño mayor a 400 nodos. Pese a ello, los parámetros obtenidos (*NumTrees*=900; *NVarToSample*=12, para la implementación *TreeBagger* de MatLab) llevan a buenos resultados, por lo que la búsqueda implementada se puede considerar buena.

Para este caso estudiado, el realizar PCA no asegura un aumento en el desempeño del clasificador. Esto ocurre porque las observaciones se encuentran demasiado mezcladas entre sí. De hecho en la matriz de confusión que se presentó para el mejor desempeño las clases 1 y 2 son las que mayor problemas tienen. Una forma de solucionar esto puede ser ensamblar estas dos clases en una, ejecutar la clasificación y luego encontrar una forma de clasificar rompiendo el ensamblaje inicial.

Para finalizar, se obtuvo un desempeño de 83.7 % normalizando los datos y de 85.5 % sin normalizar, cumpliendo con el objetivo pedido. Esta diferencia en el desempeño debido a la normalización puede ocurrir porque al normalizar, de alguna manera se pueden estar juntando los datos , es decir, "comprimiéndolos", haciendo que la clasificación empeore.

Como trabajo a futuro interesante sería ver como mejorar el desempeño del clasificador SVM para este caso en particular, e intentar realizar un reducción de dimensiones a través del uso de los algoritmos mencionados anteriormente, y que el desempeño supere el mínimo establecido.

Referencias

[1] Leo Breiman, *"Random Forests"*, Machine Learning 45 (1), 2001.

[2] Ho Tin Kam, *"Random Decision Forests"*, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, 1995.

[3] Wikipedia, *"Random Forests"*, Última consulta: 6/12/2015.

[4] MatLab Documentation Center, *"TreeBagger"*, Última consulta: 6/12/2015.

[5] Jock Blackard, Denis Dean, *"Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables"*, Computers and Electronics in Agriculture, 1999.

[6] A. J. Bagnall, G. C. Cawley, *"Learning Classifier Systems for Data Mining: A Comparison of XCS with Other Classifiers for the Forest Cover Data Set"*.

[7] Mira Trebar, Nigel Steele, *"Application of distributed SVM architectures in classifying forest data covert types"*, Computers and Electronics in Agriculture, 2008.

A. Características originales

En la Tabla 6 se muestran las características originales que posee la base de datos utilizada. A su vez cabe definir los siguientes atributos utilizados:

- Wilderness Areas:
    1. Rawah Wilderness Area
    2. Neota Wilderness Area
    3. Comanche Peak Wilderness Area
    4. Cache la Poudre Wilderness Area
- Soil Types: 1 to 40 : based on the USFS Ecological Landtype Units for this study area
- Forest Cover Types:
    1. Spruce/Fir
    2. Lodgepole Pine
    3. Ponderosa Pine
    4. Cottonwood/Willow
    5. Aspen
    6. Douglas-fir
    7. Krummholz

Name	Data type	Measurement	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance_To_Hydrology	quantitative	meters	Horz. Dist. to nearest surface water features
Vertical_Distance_To_Hydrology	quantitative	meters	Vert. Dist. to nearest surface water features
Horizontal_Distance_To_Roadways	quantitative	meters	Horz. Dist. to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points	quantitative	meters	Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns)	qualitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type (40 binary columns)	qualitative	0 (absence) or 1 (presence)	Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Tabla 6: Características de la base de datos *Covertype*.