UNIVERSIDAD
DE GRANADA

Facultad de Ciencias

Escuela Técnica Superior de Ingeniería
Informática y Telecomunicaciones

Doble Grado de Ingeniería
Informático y Matemáticas

TRABAJO DE FIN DE GRADO

Estudio de la información mutua y el test delta como criterios para la selección de variables.

Presentado por:

Iñaki Madinabeitia Cabrera

Curso académico 2019-2020

Estudio de la información mutua y el test delta como criterios para la selección de variables.

Iñaki Madinabeitia Cabrera

Iñaki Madinabeitia Cabrera *Estudio de la información mutua y el test delta como criterios para la selección de variables.*.

Trabajo de fin de Grado. Curso académico 2019-2020.

Responsable de tutorización

Alberto Guillén Perales
Departamento de Arquitectura y Tecnología de Computadores

Doble Grado de Ingeniería Informático y Matemáticas

Facultad de Ciencias

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Universidad de Granada

Declaración de originalidad

D./Dña. Iñaki Madinabeitia Cabrera

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2019-2020, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 5 de septiembre de 2019

Fdo: Iñaki Madinabeitia Cabrera

Índice general

Summary

It is a well-known fact that there is more data than a few years ago, and that there are many science fields focused on extracting information from the data. However, more data does not clearly mean easier information - there are strange phenomenons that occure with this overage of information. For instance, the curse of dimensionality. Curse of dimensionality are issues that happen when working with high-dimensions, that we would not understand in 3-dimensions.

In other words, preprocessing the data is key to extract as much information as we can, and feature selection is a fundamental part of it. We can select the 80% of the data that explain the model, for example, or we can go deeper. Here, we will study concepts of information theory, single entropy, entropy, and mutual information. These concepts are based on the idea of measure the importance of the information we are receiving, so concepts of probability and expectation will take a great part in this. With mutual information between two variables we try to know something of a variable only knowing the other variable. Also, we will understand why this is important as a criterion to select features.

On the other hand, Delta Test will be studied too. Concepts as k-nearest neighbor and noise estimation will appear as Delta Test is an estimador of the noise variance between the data X and the result Y. Oppositely to mutual information, the less the Delta Test is, the better. We will justify why Delta Test provides a good criterion to feature selection. We will implement algorithms based on MI and DT, and then we will propose a combination of both - the MI-DT. With the idea to create a stronger model, we will take both criterions to unify them on one, and select features based on it.

Simple examples will be tested, and a real one too, as well as big ideas to optimize implementation concepts such as entropy and mutual information for two variables, and a quick brute force idea to search for all the solutions in a $2^d - 1$ space.

We will test MI, DT and MI-DT, as well as the algorithms based on them, and compare against themselves.

Key words. *Mutual information, Delta test, feature selection, machine learning, data science.*

Introducción

Hoy en día se trabaja con cantidades de datos muy grandes, y el uso del Machine Learning sobre los datos está a la orden del día. Sin embargo, no es lo mismo aplicar un mismo tipo de regresión a unos datos con 20 variables que a unos con 5, pues además de la lentitud de la regresión sobre 20 variables que será mucho más comparado con el de 5, también aparece la maldición de la dimensionalidad: cuantas más dimensiones tengamos, más fenómenos extraños, desde el punto de vista de 3 dimensiones, aparecerá en nuestro aprendizaje.

El preprocesamiento de datos forma parte muy fundamental en Machine Learning, y una gran parte es la selección de variables. Aquí abordaremos este tema durante todo el proyecto, investigando y estudiando sobre conceptos como la entropía, la información mutua y el test Delta para poder llegar a los criterios basados en MI y DT para seleccionar características, cada uno con su extensa justificación. Además de implementar y estudiar varios algoritmos basados en MI y DT, propondremos una unión de ambos con idea de llegar a un modelo más robusto.

Parte I.

Resumen.

1. Resumen

El preprocesamiento de datos es una de las partes fundamentales a la hora de aplicar algún método de Machine Learning como podría ser la regresión lineal o la regresión logística. De hecho, la reducción de la dimensioón para evitar la maldición de la dimensionalización resulta de extrema importancia, así como para agilizar el proceso.

Se va a estudiar la Información Mutua y su criterio para seleccionar variables, se va a estudiar el Test Delta y su criterio y justificación para seleccionar variables, y se va a proponer una mezcla de ambas con la idea de alcanzar un método robusto que sepa equilibrar las variables que seleccionaría cada método y escoger el mejor.

Para ambos métodos se va a estudiar el k-ésimo vecino más cercano, el Test Delta por definición y la Información Mutua como aproximación de la entropía, concepto importante que nos hará adentrarnos en las primeras secciones en la Teoría de la Información.

Se llevarán a cabo varios algoritmos de selección de variables basados en MI y en DT, y se elaborarán ejemplos sencillos que permitirán vislumbrar mejor su comportamiento.

Palabras clave. *Información mutua, Delta Test, selección de variables, machine learning, ciencia de datos*.

Parte II.

Objetivos.

2. Objetivos

Los objetivos clave que se abordan en este proyecto son los siguientes:

- Análisis comparativo de estimadores de información mutua.
- Optimización de estimadores de información mutua.
- Idea e implementación de un cálculo óptimo del Test Delta en un espacio de $2^d - 1$ soluciones, donde d es la dimensión del problema.
- Análisis comparativo entre la información mutua y el Test Delta.
- Construcción de modelos que determinen el subconjunto de variables más adecuado para problemas de regresión.
- Propuesta de integración de la información mutua y el test Delta.
- Análisis de dicha propuesta.

Parte III.

Estado del arte.

3. Información Mutua

3.1. Resultados previos

Teorema 3.1. ***Teorema de Radon-Nikodym.*** *Sea (X, E) un espacio medible, con dos medidas σ-finitas μ y v. Si v es absolutamente continua respecto de μ, entonces existe una función medible $f : X \to [0, \infty[$ tal que para todo conjunto medible $A \subset X$:*

$$v(A) = \int_A f d\mu$$

Definición 3.1. La f definida en el Teorema 3.1 es única casi por doquier. A esta función se le denomina **la derivada de Radon-Nikodym**.

3.2. Entropía de Shannon y entropía conjunta

La idea básica de la entropía en teoría de la información es que un mensaje contiene mayor entropía cuanta más sorpresa genere su contenido. En otras palabras, si un evento es muy probable que ocurra, dicho evento tiene poca entropía pues si ocurriese no generaría ningún especial interés, ya que ocurrió lo que se esperaba.

Antes de preguntarnos cómo medir esta información sobre una variable aleatoria, vamos a preguntarnos cómo podemos medir la información que nos proporciona un suceso. En computación, resulta interesante expresar la información en unidades de bits, luego esto nos motiva a utilizar el logaritmo en base de 2. Sin embargo, ¿cuál es el parámetro en dicho logaritmo en base 2? Teniendo una probabilidad, junto con lo dicho en el anterior párrafo de que un suceso nos da más información cuanto menos probable es que ocurra (inversamente proporcional a la probabilidad del suceso), nos queda que:

Definición 3.2. Sea un suceso E y una probabilidad P (cuyo dominio contiene a E), definimos el **contenido de la información** del suceso bajo probabilidad P como: [McMo8]

$$I(E) := \log_2 \left(\frac{1}{P(E)} \right) = -\log_2 \left(P(E) \right)$$

Procedemos a establecer cómo medir la información que nos da una variable aleatoria, mediante la entropía de Shannon:

Definición 3.3. Sea una variable aleatoria X que toma valores x_i con $i = 1, 2, ..., n$, y sea P una distribución de probabilidad (cuyo dominio contiene a los valores de X), definimos la **entropía de Shannon**: [Sha8x]

$$H(X) := E[-\log_2(P(X))] = -E[\log_2(P(X))]$$

Vemos que en el caso discreto, siendo p la función masa de probabilidad de P (esto es, $p(x_i) = P[X = x_i]$), nos queda que:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2(p(x_i))$$

Habiendo definido la entropía de una variable aleatoria, podemos extender la definición a una pareja de variables aleatorias de forma sencilla: podemos tomar dicha pareja como un vector de variables aleatorias.

Definición 3.4. La **entropía conjunta** de dos variables aleatorias X e Y con una distribución conjunta P es definida como: [CT]

$$H(X, Y) := -E[\log_2(P(X, Y))]$$

Vemos que en el caso discreto, siendo p la función masa de probabilidad de P (esto es, $p(x, y) = P[X = x, Y = y]$), nos queda que:

$$H(X, Y) = -\sum_{x\in X}\sum_{y\in Y} p(x, y)\log_2(p(x, y))$$

Notación. De ahora en adelante, nos referiremos a log como $\log_2$.

3.3. Divergencia de Kullback-Leibler

En estadística, la divergencia de Kullback-Leibler entre dos distribuciones de probabilidad es una cantidad que mide cuán diferente es una distribución respecto de la otra [KL68].

Definición 3.5. Sean P y Q distribuciones de probabilidad que toman valores en el mismo espacio X, y además P es absolutamente continua respecto de Q ($\forall x \in X,\ Q(x) = 0 \implies P(x) = 0$), la **divergencia de Kullback-Leibler** es la esperanza de la diferencia logarítmica entre P y Q, donde se toma la esperanza usando las probabilidades de P. Es decir:

$$D_{KL}(P||Q) = \int_X \log\left(\frac{dP}{dQ}\right) dP$$

donde $\frac{dP}{dQ}$ es la derivada de Radon-Nikodym (Def. 3.1) de P respecto de Q.

Notación. Sea X una variable aleatoria y P una distribución de probabilidad, nos referimos con $P(x_i)$ a $P[X = x_i]$.

En el caso discreto, tenemos que la divergencia de Kullback-Leibler es:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Que equivalentemente es:

$$D_{KL}(P||Q) = - \sum_{x \in X} P(x) \log \left(\frac{Q(x)}{P(x)} \right)$$

Pues:

$$\sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) = \sum_{x \in X} P(x) \left(- \log \left(\frac{Q(x)}{P(x)} \right) \right)$$
$$= \sum_{x \in X} -P(x) \log \left(\frac{Q(x)}{P(x)} \right) = - \sum_{x \in X} P(x) \log \left(\frac{Q(x)}{P(x)} \right)$$

En el caso continuo, tenemos que:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$$

donde p y q son las funciones densidad de P y Q.

3.4. Definición y propiedades de la Información Mutua

La información mutua entre dos variables aleatorias es una medida que cuantifica la cantidad de información (en bits) que se obtiene de una variable aleatoria observando la otra.

Definición 3.6. Sean X e Y dos variables aleatorias cuya distribución conjunta es $P_{X,Y}$ y cuyas distribuciones marginales son P_X y P_Y respectivamente, se define la **información mutua** como:

$$I(X, Y) := D_{KL}(P_{X,Y}||P_X P_Y)$$

En el caso discreto, si X toma n valores e Y toma m valores, tenemos que

$$I(X, Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_i) \log \left(\frac{P(x_i, y_i)}{P(x_i)P(y_i)} \right)$$

En el caso continuo, siendo p la función de densidad de P, tenemos que

$$I(X,Y) = \int_X \int_Y p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) dxdy$$

La idea clave para entender la información mutua es la siguiente: la información mutua entre dos variables aleatorias es la intersección entre la entropía de una y la entropía de la otra. Es decir, $I(X,Y) = H(X) \cap H(Y)$. En la figura Figura 3.1 podemos visualizar esta idea.

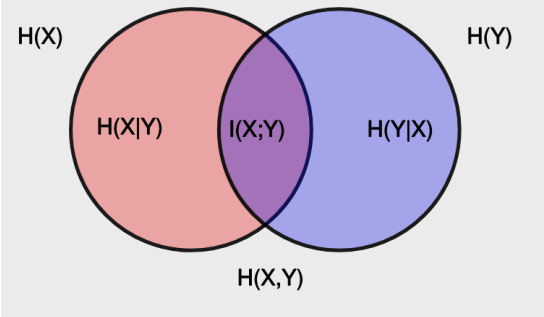

Figura 3.1.: Información mutua como intersección de $H(X) \cap H(Y)$ [Mac].

Esto también se puede ver de la forma: $I(X,Y) = H(X) + H(Y) - H(X,Y)$.

Proposición 3.1.

$$I(X,Y) = H(X) + H(Y) - H(X,Y)$$

Demostración.

$$H(X) + H(Y) - H(X,Y)$$
$$= -\sum_{x \in X} p(x) log(p(x)) - \sum_{y \in Y} p(y) log(p(y)) + \sum_{x \in X} \sum_{y \in Y} p(x,y) log(p(x,y))$$

Usando una propiedad de las probabilidades marginales, que es $p(x) = \sum_y p(x,y)$

$$= -\sum_x \sum_y p(x,y) log(p(x)) - \sum_x \sum_y p(x,y) log(p(y)) + \sum_x \sum_y p(x,y) log(p(x,y))$$

$$= \sum_x \sum_y p(x,y) \left(log(p(x,y)) - log(p(x)) - log(p(y)) \right)$$

Usando la propiedad del logaritmo de $log(a) - log(b) = log(a/b)$

$$= \sum_x \sum_y p(x,y) log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

$$= I(X,Y)$$

En definitiva, la información mutua mide el grado de dependencia estadística entre dos variables, luego es una medida que reduce la incertidumbre sobre el valor de una variable conociendo el valor de la otra variable.

4. Test Delta

Una técnica que no usa la información mutua y que al igual que la información mutua sirve para realizar una selección de variables, que veremos más adelante, es el **Test Delta**.

4.1. Test Delta

El Test Delta es una técnica para estimar la varianza del ruido, o el error cuadrático medio (MSE), que se puede lograr sin sobreajustamiento [Jon04]. Es útil para evaluar la correlación no lineal entre dos variables aleatorias, que son las parejas de entradas y salida. Es una generalización del enfoque propuesto en [Jon04], que básicamente se basa en el hecho de que la esperanza condicional se acerca la varianza del ruido cuando la distancia entre los puntos de los datos tiende a cero. Veremos esto más adelante.

El Test Delta trabaja bajo la suposición de que si dos puntos están cerca en el espacio de entrada, entonces sus correspondientes puntos de salida deben estar cerca. Si no es así, asumimos que es debido al ruido [KWW].

El único requisito es que el sistema sea suave (que la transformación de entrada a salida sea continua y que la primera derivada parcial respecto del espacio de entrada esté acotada) [KWW].

4.1.1. Estimación de la varianza del ruido

Sea el conjunto de datos de la forma:

$$\{(x_i, y_i) t.q, 1 \leq i \leq N\}$$

donde x_i la entrada como vector en $\mathbb{R}^d$ y y_i la salida como escalar en $\mathbb{R}$, asumimos con sistema suave que los valores de salida siguen la siguiente forma:

$$y_i = f(x_{i1}, ..., x_{id}) + r_i$$

donde f es una función suave (derivadas continuas en cualquier orden) y r_i es un valor aleatorio que representa el ruido [KWW]. Los términos del ruido r_i asumimos que son independientes idénticamente distribuidos con media cero [ELL$^+$].

La estimación de la varianza del ruido es el estudio que trata de dar una estimación de la varianza del ruido dado algunos datos sin considerar aspectos específicos de la función f [ELL$^+$].

4.1.2. Vecino más cercano

El estimador de la varianza del ruido considerado está basado en una aproximación mediante el vecino más cercano. El NN (*nearest neighbour*, vecino más cercano) de un punto x está definido como el *único* punto que minimiza la distancia métrica respecto a x en el espacio de entrada [ELL+].

$$NN(i) := arg\min_{j \neq i} ||x_i - x_j||^2$$

Observamos que $NN(i)$ devuelve el índice j del x_j más cercano a x_i.

En este contexto, usamos la distancia euclídea como distancia métrica.

4.1.3. Definición de Test Delta

El Test Delta se basa en realizar una estimación del ruido entre pares de puntos de entrada y salida, y aplicar el Test Delta para obtener el conjunto de variables que minimice el ruido entre los puntos.

Definición 4.1. Sea X una variable que toma N valores en $\mathbb{R}^d$ (cada elemento es un vector de dimensión d), sea Y una variable que toma N valores en $\mathbb{R}$, donde N es el número de muestras y d la dimensión de la entrada, y sea $NN(i,k)$ el índice del k-ésimo vecino más cercano a x_i; se define el **Test Delta** como: [PP94]

$$\delta_{N,k} := \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - y_{NN(i,k)} \right)^2$$

Luego para el vecino más cercano, nos quedaría ($k = 1$):

$$\delta_N := \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - y_{NN(i)} \right)^2$$

Es decir, aproximamos la varianza del ruido considerando la diferencia de las salidas asociándolo con la cercanía de los puntos en el espacio de entrada [ELL+]. Es decir:

$$Var(r) \approx \delta_N$$

4.1.4. Demostración del Test Delta como buen estimador

Acotación.

Sean X_i $(1 \leq i \leq M)$ variables aleatorias independientes idénticamente distribuidas de acuerdo a una acotada probabilidad de densidad p (por ejemplo, $p \leq ||p||_\infty$), siendo M el número de muestras. Asumimos que el rango de las variables está incluida en un conjunto compacto C contenido en $\mathbb{R}^n$. Con $NN(i,k)$ denotamos el k-ésimo vecino más cercano de X_i, y $d_{i,k}$ es $||X_i - X_{NN(i,k)}||$ con la norma euclídea. Sean $B(x_0, r) := \{x \in \mathbb{R}^n \text{ t.q. } ||x - x_0|| < r\}$ bolas de vecindario en $\mathbb{R}^n$, definimos la función:

$$\omega_{x_0}(r) = \int_{B(x_o,r)} p(x)dx$$

que corresponde a la probabilidad de que un punto de C esté contenido en la bola $B(x_0, r)$.

Proposición 4.1. *Para cada $\alpha > 0$*

$$E[\omega_{X_i}(d_{i,k})^\alpha | X_i] = \frac{\Gamma(k+\alpha)\Gamma(M)}{\Gamma(k)\Gamma(M+\alpha)}$$

donde $\Gamma(\cdot)$ es la función Gamma y α se refiere al α-ésimo momento de las distribuciones de la distancia al vecino más cercano [Eva],[KSG].

La demostración está basada en las propiedades de la función Beta establecidas en [Eva], donde también se muestra que

$$\frac{\Gamma(M)}{\Gamma(M+\alpha)} = M^{-\alpha} + O(M^{-\alpha-1})$$

denotando el término constante

$$c(k, \alpha, n) := \frac{\Gamma(k + \frac{\alpha}{n})}{\Gamma(k)}$$

y V_n como la medida de Lebesgue de la bola unidad en $\mathbb{R}^n$ (su volumen).

Con todo esto, pasamos a demostrar la proposición que necesitamos.

Proposición 4.2. *Para la constante $c(k, \alpha, n) \geq 1$,*

$$E[d_{i,k}^\alpha] \geq c(k, \alpha, n) V_n^{-\alpha/n} ||p||_\infty^{-\alpha/n} \frac{\Gamma(M)}{\Gamma(M + \alpha/n)}$$

Demostración. Una manipulación algebraica de $E[d_{i,k}^{\alpha}]$ nos lleva a

$$E[d_{i,k}^{\alpha}] \geq L(X_i)^{-\alpha/n} E[\omega_{X_i}(d_{i,k})^{\alpha/n}|X_i]$$

donde

$$L(x) = \sup_{0<r<\infty} \omega_x(r)/r^n$$

Con la proposición 1,

$$E[\omega_{X_i}(d_{i,k})^{\alpha/n}|X_i] = c(k,\alpha,n)\frac{\Gamma(M)}{\Gamma(M+\alpha/n)}$$

De $L(X_i) \leq ||p||_{\infty} V_n$ sacamos que

$$E[(d_{i,k}^{\alpha})|X_i] \geq c(k,\alpha,n) V_n^{-\alpha/n} ||p||_{\infty}^{-\alpha/n} \frac{\Gamma(M)}{\Gamma(M+\alpha/n)}$$

El hecho de que $E[d_{i,1}]$ es del orden de $M^{-1/n}$ nos muestra que cuando la dimensión del espacio de entrada crece, la distancia media al vecino más cercano tiende lentamente a cero cuando el número de muestras incrementa.

4.2. Test Delta para la selección de variables

En esta sección vamos a seguir el razonamiento de [Eir], aportando la base teórica que nos permite afirmar que el Test Delta es capaz de identificar el mejor subconjunto de variables para modelar. Como el propósito del Test Delta es trabajar con datos con ruido, es imposible formular una afirmación matemática sólida que muestre que el Test Delta siempre va a elegir las variables apropiadas. Por tanto vamos a hablar de esperanza, y mostraremos que la esperanza del Test Delta es mínima en el conjunto de las mejores variables, asumiendo primero que el número de datos M es suficientemente grande.

Vamos a suponer M variables aleatorias X_i, con $i = 1, ..., M$, que son independientes e idénticamente distribiudas de acuerdo a una probabilidad p. p es una probabilidad continua en un abierto acotado $C \subset \mathbb{R}^d$ y $p(x) > 0 \forall x \in C$.

Sea $f : C \to \mathbb{R}$ diferenciable y las variables aleatorias $Y_i = f(X_i) + r_i$, donde los r_i están independientemente distribuidos de acuerdo a una distribución de media o y varianza σ^2. Sea $I = \{1, ..., d\}$ el conjunto de variables de entrada, y $\tilde{I}$ los subconjuntos de I que corresponden a las posibles selecciones de variables. Redefinimos el Test Delta:

$$\delta : P(I) \to \mathbb{R}$$

$$\delta(\tilde{I}) := \frac{1}{2M} \sum_{i=1}^{M} \left(y_i - y_{NN_{\tilde{I}}(i)} \right)^2$$

$$NN_{\tilde{I}}(i) := arg\,\min_{j \neq i} ||x_i - x_j||_{\tilde{I}}^2$$

$$||x_i - x_j||_{\tilde{I}}^2 := \sum_{k \in \tilde{I}} \left(x_i^{(k)} - x_j^{(k)} \right)^2$$

donde $x_i^{(k)}$ simboliza la componente k-ésima de x_i.

4.2.1. Estudio lineal

Como el Test Delta está basado en el vecino más cercano, que es un fenónemo local, el método puede ser analizado localmente para dar una vista intuitiva de su comportamiento.

Asumimos que los datos $x_i \in (0,1)^d \forall i \in \{1, ..., M\}$ son i.i.d. con la distribución uniforme en el hipercubo unidad sin bordes. Sea $y_i = f(x_i) + r_i \ \forall i \in \{1, ..., M\}$, con $f(x_i) = a_0 + \sum_{k=1}^{d} a_k x_i^{(k)}$. Es decir, es una combinación lineal de x_i añadiéndole ruido.

Sea $D \in P(\{1, ..., d\})$ el conjunto de variables que realmente afectan a la salida, $D = \{kt.q.\partial_k f$ no es zero en alguna parte $\} = \{kt.q.a_k \neq 0\}$.

Proposición 4.3. $S \neq D \implies E[\delta(S)] > E[\delta(D)]$

Demostración. Página 16 [Eir]

Podemos extender el resultado anterior a funciones diferenciables con distribuciones continuas. Aplicamos el Teorema del Valor Medio para dar un punto $\tilde{x}_i$ que pertenece al segmento que une x_i y $x_{NN_S(i)}$.

4.2.2. Análisis del Test Delta

Entonces tenemos que si $\tilde{x}_i = (x_i^{(\tilde{I}_1)}, x_i^{(\tilde{I}_2)}, ...)$ es la proyección de cada punto al subespacio correspondiente a las variables seleccionadas, y $\tilde{x}_i'$ corresponde a los componentes que no están en $\tilde{I}$ y es la proyección al subespacio correspondiente a $I \setminus \tilde{I}$, definimos

$$\tilde{f}(\tilde{x}) = \int_C f(\tilde{x}, \tilde{x}') \tilde{p}(\tilde{x}') d\tilde{x}'$$

donde $\tilde{p}(\tilde{x}')$ es la marginal

$$\tilde{p}(\tilde{x}') = \int_C p(\tilde{x}, \tilde{x}') d\tilde{x}$$

Ahora podemos ver $\tilde{f}(\tilde{x}_i)$ como la mejor aproximación a y_i usando solamente las variables en $\tilde{I}$, pues

$$\tilde{f}(\tilde{x}_i) = E[Y_i | \tilde{X}_i = \tilde{x}]$$

Con todo esto dicho, pasamos a dos importantes proposiciones.

Proposición 4.4. *Si $\tilde{I}$ es de tal forma que existe un $x_0 \in C$ donde $\tilde{f}(\tilde{x}_0) \neq f(x_0)$, entonces, para un M suficientemente grande, $E[\delta(\tilde{I})] > E[\delta(I)]$*

Demostración. Página 18 [Eir].

Proposición 4.5. *Si $\tilde{I}$ y $\hat{I}$ son de tal forma que $\forall i : \tilde{f}(\tilde{x}_i) = \hat{f}(\hat{x}_i) = f(x_i)$ (ambos son suficientes para explicar f), y $\#\tilde{I} < \#\hat{I}$, entonces para un M suficientemente grande, $E[\delta(\tilde{I})] < E[\delta(\hat{I})]$.*

Demostración. Página 19 [Eir].

Con todo lo explicado, podemos pasar al teorema importante.

Teorema 4.1. *Asumiendo un número suficientemente grande de puntos, la esperanza del Test Delta alcanza el mínimo con el subconjunto más pequeño de I que pueda explicar completamente f en c.*

Demostración. Teniendo un número suficientemente grande de puntos, por la segunda proposición la selección que minimiza debe ser capaz de explicar f, y por la tercera proposición debe ser la más pequeña.

4.2.3. Observación final

En [Eir] se muestra que la varianza del Test Delta converge a o cuanto más crece M. Como la esperanza del Test Delta bajo las asunciones reflejadas alcanza el mínimo por la 'mejor' selección, esto significa que la probabilidad de que el método seleccione las variables indicadas incrementa cuanto más crece el número de muestras.

Parte IV.

Problema de la selección de variables.

5. Problema de la selección de variables

La selección de variables (*input selection*) es uno de los mayores problemas en Aprendizaje Automático, especialmente cuando el número de características es relativamente grande comparado con el número de observaciones.

Matemáticamente hablando, un conjunto finito de variables es suficiente para extraer un modelo exacto de infinitas observaciones. En la práctica, no tenemos infinitas observaciones, por tanto el tamaño necesario de variables incrementa drásticamente con el número de observaciones. [RHJL]

5.1. Maldición de la dimensionalidad

En estadística, la maldición de la dimensionalidad se refiere a varios fenómenos que ocurren al analizar y organizar datos de grandes dimensiones que no suceden en dimensiones pequeñas.

Un ejemplo es que, si quisiéramos hallar cuántos puntos son suficientes para representar un cubo unidimensional (equivalentemente, un intervalo de longitud 1) para que la distancia entre cualesquiera dos puntos cercanos sea 10^{-2}, tendríamos que resolver esta simple ecuación (siendo $x > 0$ la cantidad de puntos):

$$\frac{1}{x} = 10^{-2}$$

luego bastan 10^2 puntos, pues $1/10^2 = 10^{-2}$. Sin embargo, si nos preguntamos lo mismo pero en el caso de un hipercubo de diez dimensiones, resolveríamos la misma ecuación por cada dimensión y entonces bastarían $10^{2^{10}} = 10^{20}$ puntos.

En aprendizaje automático se da el **fenómeno de Hughes** [Hug59], que nos dice que teniendo el número de muestras de entrenamiento fijo, el poder predictivo del clasificador o regresor primero tiene una relación directamente proporcional al número de dimensiones (aumenta cuando aumenta el número de dimensiones), pero luego se vuelve indirectamente proporcional (disminuye al aumentar el número de dimensiones).

5.2. Selección de variables

Ahora, añadimos la premisa fundamental para resolver la maldición de dimensionalidad seleccionando variables:

Premisa 5.1. Los datos contienen características que son redundantes o irrelevantes. Por tanto, dichas características pueden ser eliminadas sin que esto conlleve a una gran pérdida de información.

En aprendizaje automático y en estadística, la selección de variables consiste en escoger un subconjunto de características relevantes, evitando aquellas redundantes o irrelevantes. Además de evitar la *maldición de la dimensionalidad*, también encontramos estas propiedades:

- Simplifica modelos.
- Agiliza los períodos de entrenamiento.
- Favorece la generalización y evita el sobreentrenamiento.

5.3. Ejemplo de algoritmo que usa Información Mutua: mRMR

Mínima redundancia máxima relevancia, o también conocido como **mRMR**, es un algoritmo que selecciona un subconjunto de características que mejor caracteriza las propiedades estadísticas de una variable de clasificación objetivo (target), sujeto a las restricciones de que estas características sean tan diferentes entre sí como sea posible, pero marginalmente sean lo más similar posible a la variable de clasificación objetivo. Es decir, que estén muy alejadas unas características de las otras pero que guarden una alta correlación con la variable de clasificación. Esta correlación puede verse estimada con la dependencia estadística entre las variables. La información mutua puede ser usada para cuantificar dicha dependencia, pues mRMR puede verse como una aproximación de maximizar la dependencia entre las distribuciones conjuntas de las características seleccionadas y de la variable de clasificación [ALC10].

Para características continuas, el *F-statistic* puede ser usado para calcular la correlación de la clase (relevancia) y el coeficiente de correlación de Pearson puede ser usado para calcular la correlación entre las características (redundancia). Luego, las características pueden ser seleccionadas una por una mediante una búsqueda *greedy* que optimice la función objetivo. Dos tipos de funciones objetivo comúnmente usados son **MID (criterio de Mutual Information Difference)**, que representa la diferencia de la relevancia entre las características, y **MIQ (criterio de Mutual Information Quotient)** que representa el cociente de la redundancia entre las características [RGFO 9].

5.3.1. Extensiones

Seleccionar variables puede no ser una tarea trivial: por ejemplo en el estudio sobre expresiones genéticas se tiene el carácter temporal de los datos. Sin embargo, la mayoría de los enfoques de selección de características desarrollados para datos de microarrays no pueden manejar datos multivariantes temporales sin un aplanamiento previo de datos, lo que resulta en una pérdida de información temporal. Extendiendo mRMR, se propuso un enfoque temporal de mínima redundancia máxima relevancia (TMRMR) que tuvo buenos resultados [RGFO 9].

Parte V.

Estimadores de información mutua

6. Información mutua para dos variables

Para simplificación del modelo, de ahora en adelante vamos a tomar la definición de la información mutua (y por tanto de la entropía de Shannon) tomando p como la probabilidad uniforme, donde $p(x) = P[X = x]$, $p(x, y) = P[X = x, Y = y]$. Por tanto, sea X una variable aleatoria discreta que toma n valores, la probabilidad $p(x)$ donde $x \in X$ es

$$p(x) = \frac{K(x)}{N}$$

donde $K(x)$ es la aplicación que determina cuántas veces aparece x en X. Se podría ver como la intersección entre X y una variable que toma el elemento x n veces, o se podría ver como una operación lógica 'AND' entre el vector X y otro vector con el elemento x n veces.

7. Optimizaciones y propuestas

Se ha basado y optimizado las funciones disponibles en [MM] de 'single_entropy' (entropía de una variable), 'entropy' (entropía entre dos variables) y 'mutual_information' (información mutua entre dos variables).

7.1. Entropía para una variable

Como se ha explicado con anterioridad, al trabajar con la probabilidad uniforme debemos contar cuántas veces aparece x en la variable X. La optimización en este caso viene por parte de la misma estructura de datos. Se ha transformado X en un diccionario cuyas claves son los distintos elementos que tiene X, y cuyos valores son las veces que aparece cada clave en X. Por tanto, $p(x)$ son los valores de la clave x dividido por el número de elementos de X.

7.2. Entropía para dos variables

La entropía para dos variables se puede ver de la siguiente manera, tal y como se visualiza en [MM]: $\forall x \in X, \forall y \in Y, p(x,y)$ es el número de elementos que hay en la intersección (entendiéndose como operación lógica 'AND') dividido por el número de elementos que contienen las variables X e Y.

A nivel de implementación, la traducción de lo explicado acaba en dos bucles anidados que recorren X e Y, respectivamente. La propuesta de optimización en este caso se basa en la transformación de dos bucles anidados que recorren X e Y, a dos bucles anidados que recorren X y otro que como máximo recorre las posiciones donde X es x de forma factorial:

$\forall x \in X$, obtenemos los lugares donde aparece x en X. Ahora, si llamamos P_x a la variable que nos indica dichos lugares, $\forall p \in P_x$, mientras que P_x no esté vacío, seleccionamos la y en la posición p de Y, borramos la posición p de P_x y llevamos el contador a 1, y ahora recorremos el resto de posiciones: si se repite la y de la nueva posición con la y de la posición p, aumentamos el contador en 1 y borramos la posición de P_x, y sino seguimos recorriendo. Entonces aplicamos la fórmula de la entropía donde $p(x,y)$ es el contador dividido por el número de elementos que tiene X e Y (tras esto volvemos a la parte de 'mientras que P_x no esté vacío').

En Python, el código queda de la siguiente manera:

```
for x in set(X):
    posiciones_x = where(X[:,0]==x)[0].tolist()
    while len(posiciones_x) > 0:
        y = X[:,1][posiciones_x[0]]
        cont_y = 1
        del posiciones_x[0]
        # We get the first y_i, count it (uniform probability)
        # and we remove the position from posiciones_x.
        y = X[:,1][posiciones_x[0]]
        cont_y = 1
        del posiciones_x[0]
        # We iterate in posiciones_x in a specific way: if we remove a
        # position then we don't update i, else i += 1.
        # Note that len(posiciones_x) is being updated inside the loop.
        i = 0
        while i < len(posiciones_x):
            # We find y_j.
            z = X[:,1][posiciones_x[i]]
            # If y_i == y_j, then we count it and remove.
            if y == z:
                cont_y += 1
                del posiciones_x[i]
            # Else, we iterate to the next element.
            else:
                i += 1
        # p(x,y) with uniform probability.
        pxy = cont_y / n_rows
        if pxy > 0.0:
            summation += pxy * math.log(pxy, log_base)
if summation == 0.0:
    return summation
else:
    return - summation
```

En un caso con dos variables arbitrarias, esta propuesta sería con mucha probabilidad más ineficiente que la opción dada, ya que pasamos de un algoritmo cuadrático $O(n^2)$ a uno $O(n(m!))$, con $m < n$, y generalmente $n(m!) > n^2$ (para $n = 4$ y $m = 3$ ya se da). Sin embargo, no trabajamos con dos variables arbitrarias: cuando apliquemos la entropía entre dos variables, lo haremos sobre una variable X y la variable de clasificación Y, que usualmente sus elementos diferentes son pocos comparado al abanico de posibilidades de X. Es por tanto que dejamos implementado y propuesto este algoritmo.

7.3. Información mutua entre dos variables

Esta optimización es justo la unión de la primera optimización y la propuesta en el apartado anterior, ya que están basadas en $p(x)$ y $p(x,y)$, y ambas aparecen en la definición de información mutua.

8. Información mutua en un conjunto de variables

Con la información mutua entre dos variables definida, es de gran utilidad preguntarnos cuál es la información mutua en un conjunto de variables. Es decir, sea $X = \{X_1, ..., X_N\}$, con cada X_i siendo una variable que puede tomar n valores, y sea Y la variable de clasificación (sin pérdida de generalidad podemos agrupar X e Y y llamar X_N a Y), ¿qué información mutua hay entre X e Y?

Vamos a indicar las opciones que hemos valorado a lo largo del proyecto, y la respuesta final.

8.1. La suma

La primera opción que tomamos fue la suma:

$$I(X_1, X_2, ..., X_N) := \sum_{i=1}^{N} I(X_i, X_N)$$

Sin embargo, es un mal planteamiento: por definición, ni siquiera podemos volver a escribir la información mutua en base a entropías, y por la idea que vendrá más adelante de la selección de variables. Sobre esto último, más adelante seleccionaremos las variables que del conjunto de variables dé una información mutua mayor, pues serían las variables que más nos aporta información sobre Y sin conocerla. Imaginemos ahora un caso en el que Y dependa de dos variables de X, pero X está compuesto de 5 variables. Esta mala extensión de la información mutua multivariante nos llevaría a seleccionar un subconjunto de X como las variables que más información nos dan sobre Y sin poder justificar que realmente sea así. En la práctica, hemos hecho esta misma comprobación y cuantas más variables habían, mayor era la información mutua, y seleccionaba todo el conjunto X.

8.2. La media

$$I(X_1, X_2, ..., X_N) := \frac{1}{N} \sum_{i=1}^{N} I(X_i, X_N)$$

Esta opción es digna de mención pues, aunque sea una mala extensión con mucho parecido a lo anterior, en realidad el segundo criterio de descarte es contrario. Es decir, volvemos a no poder escribir la información mutua en base a entropías, pero ahora la selección de variables cambia drásticamente: al realizar la media, ahora la variable de X que devolverá

mayor información mutua (como la que más explica la variable Y) sí tendrá relación con Y, pero el subconjunto que dará mayor información mutua siempre consistirá de una sola variable, independientemente de si está basado o no de más variables. Es decir, si por ejemplo Y está basado en las variables X_0 y X_1, y $I(X_0, Y) > I(X_1, Y)$, como Y está basado en X_0 y X_1 se tiene que $I(X_i, Y) > I(X_j, Y)$ con $i = 0, 1$ y $j = 2, .., N$; y como $I(X_0, Y) > I(X_1, Y)$, se tiene que

$$I(X_0, Y) = \frac{I(X_0, Y)}{1} = \frac{2 * I(X_0, Y)}{2} = \frac{I(X_0, Y) + I(X_0, Y)}{2} > \frac{I(X_0, Y) + I(X_1, Y)}{2}$$

Luego siempre tendríamos que la variable X_0, la variable que da mayor información mutua en solitario, sería la seleccionada.

8.3. La extensión teórica más convincente

Volvamos a ver cómo podemos escribir la información mutua en base a entropías. Como hemos demostrado en 2 dimensiones, la extensión natural a 3 dimensiones sería la siguiente:

$$I(X_1, X_2, X_3)$$

$$= H(X_1) + H(X_2) + H(X_3) - H(X_1, X_2) - H(X_2, X_3) - H(X_1, X_3) + H(X_1, X_2, X_3)$$

donde la extensión de la entropía es natural viniendo de la extensión de una variable a dos:

$$H(X_1, X_2, X_3) = -E[log(P(X_1, X_2, X_3)] = -\sum_{x \in X_1} \sum_{y \in X_2} \sum_{z \in X_3} p(x, y, z) log(p(x, y, z))$$

Es decir, que para n variables, la información mutua multivariante sería:

$$MMI(X_1, ..., X_n) = \sum_{k=1}^{n} (-1)^{k+1} \sum_{M \subseteq \{1,...,n\}, |M|=k} H(X_{M_1}, X_{M_2}, ..., X_{M_k}$$

donde el interior de la suma se toma con todos los posibles índices tomados desde 1 hasta n de cardinalidad k. Esta definición nos lleva a la definición de McGill (1954) y Fano (1961)[BGMS].

8.3.1. Problemas con la implementación

Pese a que teóricamente parece que la información mutua multivariante está hallada, nos encontramos con la dificultad de implementar la entropía multivariante. Es decir, la expresión es la siguiente:

$$H(X_1, ..., X_N) = - \sum_{(x_1,...,x_N) \in (X_1,...,X_N)} p(x_1, ..., x_N) log(p(x_1, ..., x_N))$$

Concretamente, el problema subyace en la expresión $p(x_1, ..., x_N)$. Recordando la discusión de la entropía para dos variables del capítulo anterior, lo que tendríamos que implementar es: $\forall x_1 \in X$, $\forall x_2 \in X_2$, ..., $\forall x_N \in X_N$, $p(x_1, ..., x_N)$ es el número de elementos que hay en la intersección (entendiéndose como operación lógica 'AND') dividido por el número de elementos de X_1.

Tras la tremenda dificultad de intentar encontrar una optimización para reducir tantos bucles anidados, hallamos una estimación de la información mutua multivariante, que de hecho es la más utilizada y mucho más sencilla.

8.4. Estimación a la información mutua multivariante

También llamado correlación total, multiinformación, interactión estocástica e integración, la primera extensión de la información mutua multivariante fue expresada como[BGMS]:

$$MI(X_1, ..., X_n) = \left(\sum_{i=1}^{n} H(X_i) \right) - H(X_1, ..., X_n)$$

Esta estimación de la información mutua multivariante es la implementada en [BV]. Sobre el problema de la entropía, se ha usado el estimador del k vecino más cercano, ya que la otra forma lleva a muchos cómputos y, además, en el caso de variables continuas a integrales. Por tanto, se escoge un criterio de aproximación para evaluarlos numéricamente. La base subyace en estimar la entropía con la media de las distancias al k vecino más cercano [SHL].

Parte VI.

Cálculo del Test Delta

9. K-ésimo vecino más cercano

Para el cómputo del Test Delta necesitamos hallar el k-ésimo vecino más cercano. Para ello, elaboramos la función con entrada el índice de la variable a la que se hallará dicho vecino, y el parámetro k. Hemos usado la función 'NearestNeighbors' del paquete 'sklearn.neighbors', con número de vecinos $k+1$ y con norma L_2 (norma euclídea). Lo ajustamos a nuestro conjunto de datos X, y con este objeto llamamos a la función 'kneighbors' con la variable en forma de lista (bajo 'reshape(1,-1)'). Devolvemos el k-ésimo elemento.

La razón por la que pedimos $k+1$ vecinos y devolvemos el k-ésimo elemento es porque en la definición de 'NearestNeighbors', éste no elimina el propio elemento del conjunto de búsqueda, luego siempre se da que el vecino más cercano es él mismo.

10. Test Delta

Para el Test Delta, pedimos como parámetro k. Inicializamos la suma a 0, y por cada variable (columna) del conjunto de datos, hallamos el k-ésimo vecino más cercano, y actualizamos la suma con

$$suma = suma + (y_i - y_{NN(x_i)})^2$$

donde $NN(x_i)$ es el índice del k-ésimo vecino más cercano de x_i.

Al finalizar el bucle, devolvemos la suma dividido por $2n$, donde n es el número de variables.

Parte VII.

Propuesta de combinación de Test Delta con Información Mutua.

11. Propuesta de combinación de Test Delta con Información Mutua

Sabiendo las ventajas que nos aporta la información mutua, en concreto la selección de variables y los algoritmos basados en ella para sacar las características que reflejen mínima redundancia y máxima relevancia, así como las ventajas del rápido cálculo del Test Delta como propia técnica de selección de variables, surge la siguiente idea.

Combinar la información mutua con el test Delta.

A priori, una técnica de selección de variables que escoja un subconjunto que refleje las propiedades estadísticas de la variable de clasificación, que entre ellas sean tan diferentes entre sí como sea posible pero que marginalmente sean lo más similar posible, y que además las variables seleccionadas minimizan el ruido entre la entrada y la salida, nos hace pensar que puede obtener mejores resultados que la Información Mutua y el Test Delta por separado.

Se estudia como criterio la maximización de la resta entre MI y DT, conservando así la idea de maximizar MI y minimizar DT, además de un estudio de ajuste por parámetros con un α que acompaña a MI, y un β que acompaña a DT.

En el caso de unos datos con un gran número de muestras, se deja planteado una segunda versión de MI-DT, basado en maximizar $MI + 1/(delta + \beta)$, con $\beta = 0{,}01$ por ejemplo. La idea que subyace es la de centrarnos en DT cuando éste valga poco. Pero para que esta información sea útil, es importante que el número de muestras sea grande.

Parte VIII.

Implementación.

12. Algoritmo optimizado de Fuerza Bruta

En nuestros experimentos, vamos a comparar los resultados obteniendo siempre el conjunto de variables que proporciona mayor información mutua o menor Test Delta de entre todos los posibles subconjuntos de X. También aplicaremos la misma idea para la propuesta de integración de MI con Delta, y en este apartado quedará explicado.

12.1. Conceptos

Si X tiene d variables, al querer obtener todos los posibles subconjuntos de X, cada subconjunto posible está determinado por si X_0 aparece o no aparece en el subconjunto, si X_1 aparece o no aparece... así sucesivamente hasta X_d. Es decir, por cada posible subconjunto de X, cada variable X_i tiene únicamente dos posibilidades: aparecer o no aparecer. Por tanto, hay 2^d posibles conjuntos en un principio.

Sin embargo, hay un conjunto en concreto que nunca aparecerá entre los posibles subconjuntos de X. El conjunto vacío: el conjunto en las que no aparece ninguna de las variables de X. Por tanto, ahora sí, nos quedan $2^d - 1$ posibles subconjuntos de X, luego es un problema con un espacio de solucioens de $2^d - 1$.

12.2. Implementación

Como cada variable tiene únicamente dos posibilidades, elegimos representar el problema con bits. Es decir, si $d = 7$, la información mutua que obtendremos en la posición 5 es en realidad la información mutua obtenida de 0000101; esto es, la información mutua entre la antepenúltima y la última variable de X. Por lo explicado con anterioridad, tendremos un vector de $2^d - 1$ elementos.

La idea es la siguiente: guardamos en la primera posición la información mutua de X con todas sus variables, y sea i desde 1 hasta $2^d - 1$, hallamos el número i en binario, le añadimos a la izquierda tantos 0 como hagan falta para que sea un número de $2^d - 1$ bits, y ahora por cada 1 que encontremos en el número en binario, añadimos su posición a un vector. Tras acabar este pequeño bucle, realizamos la información mutua con las variables que señala el vector de posiciones con X, y guardamos estas informaciones mutuas en otro vector.

Al terminar de recorrer los $2^d - 1$ números, hallamos la posición del máximo (en caso de la información mutua) o la del mínimo (en caso del Test Delta), transformamos dicha posición a binario y volvemos a indicar las posiciones donde se encuentran los 1 del número binario. Y se devuelven dichas posiciones, pues son las variables que han alcanzado el óptimo.

13. Algoritmos Filter

Se han realizado algoritmos Filter basados en la información mutua, en el Test Delta y en ambos unidos. La idea que subyace es la misma y quedará explicado en esta sección.

Lo explicamos desde el punto de vista del Test Delta. Para MI y MI-DT habría que cambiar el criterio de minimizar por maximizar.

1. Mientras delta siga descendiendo (el delta actual es menor que el anterior)
   - Recorremos las variables (columnas) de X
     - Borramos la columna en X
     - Aplicamos delta al subconjunto restante, y actualizamos el mínimo delta
   - Recuperamos el X, y le borramos la columna cuyo subconjunto restante dio el mínimo delta
2. Devolvemos las columnas del subconjunto no eliminado

14. mRMR MI

Misma idea que un algoritmo Filter, pero sin borrar columnas y teniendo en cuenta la redundancia:

1. Mientras MI siga ascendiendo
   - Recorremos las variables (columnas) de X
     - Hallamos la MI entre la variable y Y (relevancia)
     - Sumamos la MI entre esta variable y entre cada variable que ya haya sido seleccionada (redundancia)
     - Aplicamos máxima relevancia-mínima redundancia (relevancia - redundancia normalizada), y actualizamos el máximo
   - Eliminamos la columna que dio el máximo, y la guardamos para devolver
2. Devolvemos las columnas del subconjunto eliminado

15. Información sobre la implementación

Todo el proyecto, tanto el código como los datos usados y obtenidos, así como los resultados en forma de tablas o figuras, y esta misma memoria, se encuentran en [Cab].

Parte IX.

Experimentos.

16. Uso y visualización

Todo el proyecto, tanto el código como los datos usados y obtenidos, así como los resultados en forma de tablas y figuras, y esta misma memoria, se encuentran en [Cab].

La información mutua, el Test Delta, los distintos algoritmos para calcularlos y su unión están implementadas en la clase *Informacion* dentro del archivo *info.py*. Hay ejemplos de uso que además llevan a los ejemplos que vamos a ver a continuación, y más.

Se crea un objeto *Informacion* pasando el conjunto de datos X (y se toma Y como la última variable de X), o pasando X e Y por separado. Una vez hecho esto se pueden llamar a las funciones del objeto creado.

Las tablas con todos los ejemplos creados están en *datos*, y la forma de llegar a ellos en el archivo *script_info_examples.py*, que para entender mejor cómo se logra cada ejemplo se recomienda mirar *info_examples.py*. Las figuras están en la carpeta *plots*, y el script para llegar a ellas es *script_info_plots.py*. Además se incluye el archivo *mutual_info.py* para el cálculo de la MI y las entropías, cuyo código proviene de [BV].

17. Introducción

A partir de ahora en los ejemplos, salvo que se diga lo contrario, tienen 1000 muestras (filas).

Los ejemplos que se proveen aquí están especificados en [Cab], preparados para ser repetidos en cualquier máquina. De hecho se han sólo los tipos de ejemplos significativos de un total de 28 ejemplos, donde la mayoría son modificaciones añadiendo ruido, cambiando subconjunto de mejores variables... También se ha hecho en 2 ejemplos un estudio de los parámetros α y β, que son los que acompañan a MI y DT, respectivamente, en MI-DT.

Lo que se espera encontrar es un decrecimiento significativo del MI de X según aumenta el número de variables que no influyen en Y, de igual manera un crecimiento significativo del DT de X según aumenta el número de variables, y esperar encontrar los casos donde la Fuerza Bruta MI se mantiene en el máximo, la Fuerza Bruta de Delta Test en el mínimo, y en qué situaciones no. Y en esas situaciones, percibir el comportamiento de la mezcla de MI-DT.

En cuanto a tiempos, por complicación en cuanto al cálculo de la entropía esperamos que BF MI tarde significativamente más que BF DT, y como la mezcla BF MI-DT aplica las dos esperamos que lo que tarde se acerque a la suma de ambas.

En cuanto a porcentaje de acierto, más difícil preveemos que será acertar cuantas más variables y más complicada sea la función Y.

18. Ejemplo A: Dimensión 1

El primer ejemplo es el ejemplo más sencillo: creamos X con una sola variable, y definimos Y de la siguiente forma:

$$Y(x) = 4x^2 + 3$$

En la figura izquierda 18.1 podemos visualizar cómo efectivamente el MI de X va decayendo según se aumenta el número de variables, pues Y sólo dependía de la primera variable. También así crece el Delta Test, sin embargo no lo hace de forma monótona, lo cual para un ejemplo tan sencillo ya nos hace pensar que DT puede no significar de mucha utilidad con 1000 muestras.

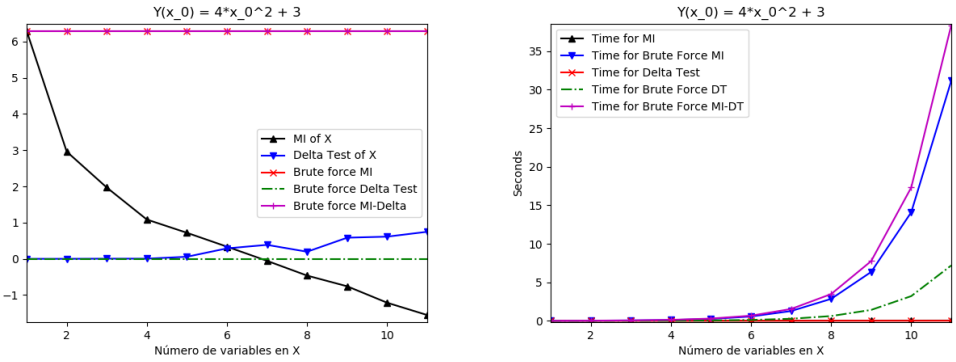

Figura 18.1.: Datos y tiempos del ejemplo A.

En cuanto a tiempos (figura derecha 18.1), efectivamente BF DT es muy rápido en comparación con BF MI. La mezcla BF MI-DT se asemeja a la tardanza de la suma de ambas.

Sobre el porcentaje de aciertos, los tres algoritmos han acertado con la variable que depende de Y con X teniendo desde 1 hasta 11 variables.

19. Ejemplo B: Dos dimensiones

X ahora tiene dos variables, y definimos Y de la forma

$$Y(x,y) = \sqrt{x^2+y^2} * (x^2+y^2)$$

En la figura izquierda 19.1 podemos apreciar el comportamiento extraño del Delta Test de X, y lo vemos claramente en la siguiente figura 19.2 cuando pasa de un 100 % de acierto a un 0 % al pasar de 7 a 8 variables, y a partir de ese momento se mantiene en 0 %.

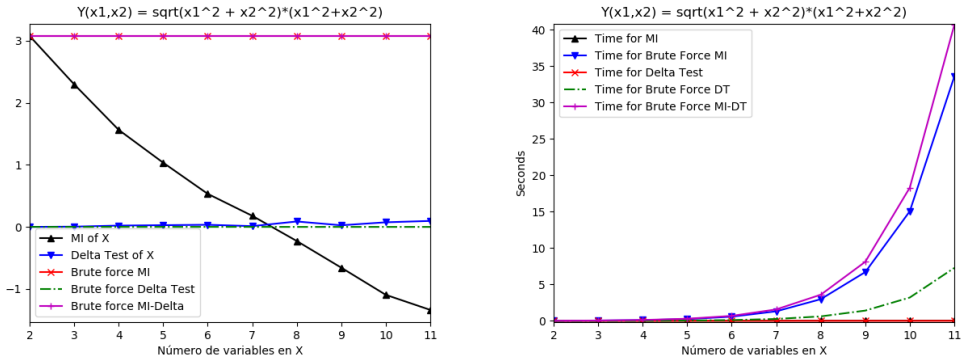

Figura 19.1.: Datos y tiempos del ejemplo B.

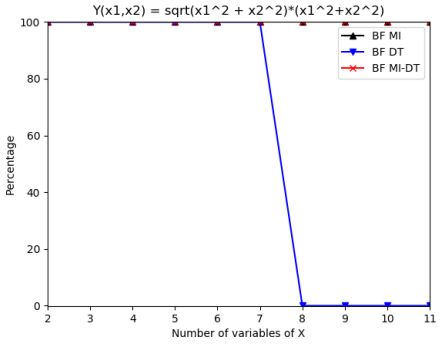

Figura 19.2.: Porcentaje de acierto de B.

Recordamos que las propiedades tan buenas que tiene el Delta Test necesita varias asunciones, este análisis sugiere reflexionar sobre la cantidad de muestras: era importante que M fuese lo suficientemente grande, y 1000 puede no serlo para 8 variables o más.

Se han realizado 4 variantes más a este ejemplo: añadiendo a las Y un ruido y el porcentaje de importancia del ruido. En concreto, dos ejemplos con un ruido que sigue distribución uniforme y porcentajes del 10 % y 50 %, y lo mismo para ruido con distribución normal de media 0.5 y desviación típica 1. No ha alterado nada los datos, de hecho las gráficas son idénticas a las expuestas (pueden verse en [Cab]).

Efectivamente, era el número de muestras. Tras visualizar estos resultados, se ha repetido el ejemplo B pero con 6 veces más muestras que antes. En la tabla 'ejemploB_6n' de [Cab] queda reflejado que el 100 % de acierto en DT se mantiene (así como el de los demás).

20. Ejemplo E: Tres dimensiones

Ahora X contiene tres variables en un principio, e Y se define como:

$$Y(x,y,z) = \sin(\sqrt{x^2+y^2})/(x^2+y^2) + 20(z-0{,}5)^2$$

una de las funciones clásicas de Cherkassky [CM].

En la figura izquierda 20.1, observamos que al pasar de 8 a 9 variables DT estima un cambio significativo, a peor, de ruido. Sin embargo, BF DT se mantiene en 0, luego según la primera proposición sobre el Test Delta, el único subconjunto que alcanza el mínimo es el subconjunto de las mejores variables.

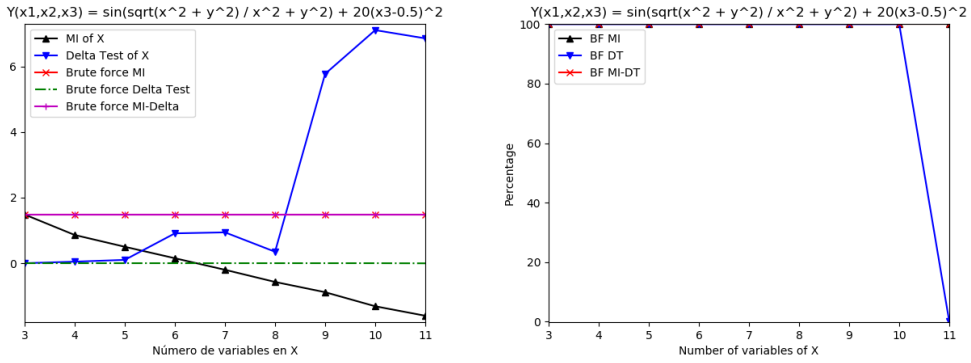

Figura 20.1.: Datos y porcentaje de acierto del ejemplo C.

Sin embargo, podemos ver en la figura derecha 20.1 que, pese a que BF DT no cambia al llegar a 11 variables, sí cambia de subconjunto y de hecho a un subconjunto que no contiene a ninguna variable de las mejores. Esto entonces contradiría la primera proposición, sin embargo nuevamente este análisis sugiere que es que no se da una de las hipótesis para que la proposición se dé: el tamaño de muestras no es lo suficientemente grande para el Test Delta.

21. Ejemplo C: Todos los algoritmos

Repetimos el mismo ejemplo C pero ahora vamos a estudiar cómo se comportan los distintos algoritmos implementados.

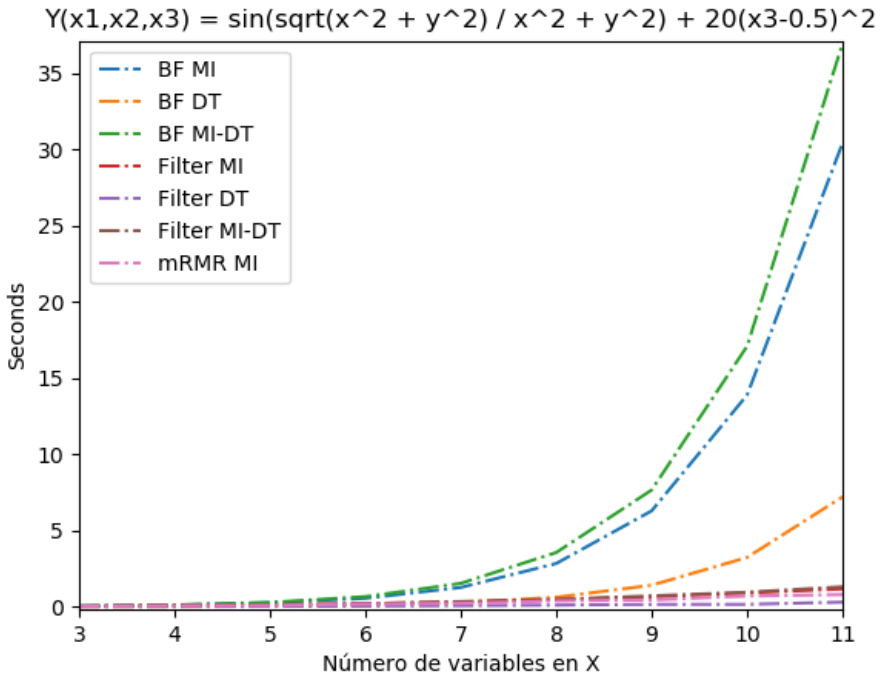

Figura 21.1.: Tiempos del ejemplo C con todos los algoritmos.

En la figura 21.1 podemos comprobar que los algoritmos de Fuerza Bruta destacan por su lentitud. Esto era de esperar, comprobar todos los posibles subconjuntos es exponencial ($O(2^n)$) en cuanto al número de variables, mientras que los Filter y mRMR son algoritmos Greedy, rápidos.

Si eliminamos los algoritmos de Fuerza Bruta nos queda la figura 21.2, donde destaca significativamente Filter DT como el más rápido, debido a la facilidad de sus cálculos comparados a los de MI y entropía, y entre los MI gana mRMR. Es curioso destacar lo parecido que tardan los Filter MI y MI-DT, esto nos da la prueba definitiva de que son los cálculos del MI lo que se lleva gran parte del tiempo de cálculo.

Vamos a la parte más importante. En la figura 21.3 podemos apreciar el porcentaje de aciertos de los distintos algoritmos. Podemos ver que los BF adquieren y mantienen el 100 %

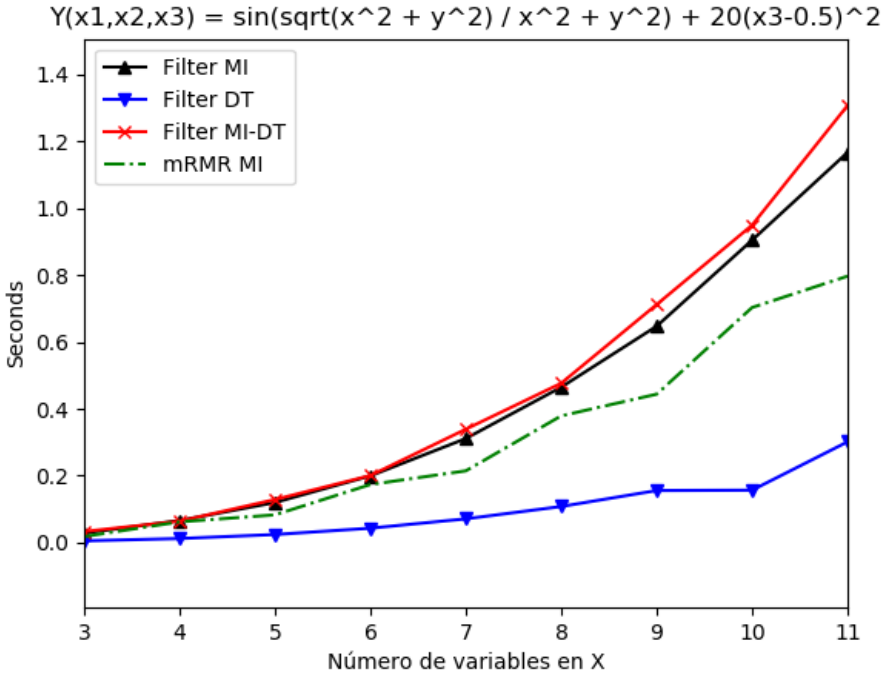

Figura 21.2.: Tiempos del ejemplo C sin contar los de Fuerza Bruta.

de acierto salvo BF DT que baja en la última variable al 0 % como habíamos estudiado antes, Filter DT también adquiere y mantiene el 100 % salvo con 7 variables que baja a casi el 70 %. Filter MI-DT se ve afectado por el DT en esas 11 variables, y lo más llamativo es mRMR MI. Cuantas más variables tiene X, más puede comparar en cuanto a relevancia y redundancia, y en 6 variables llega al 100 % y se mantiene, cuando con 3 variables se queda con menos del 40 %.

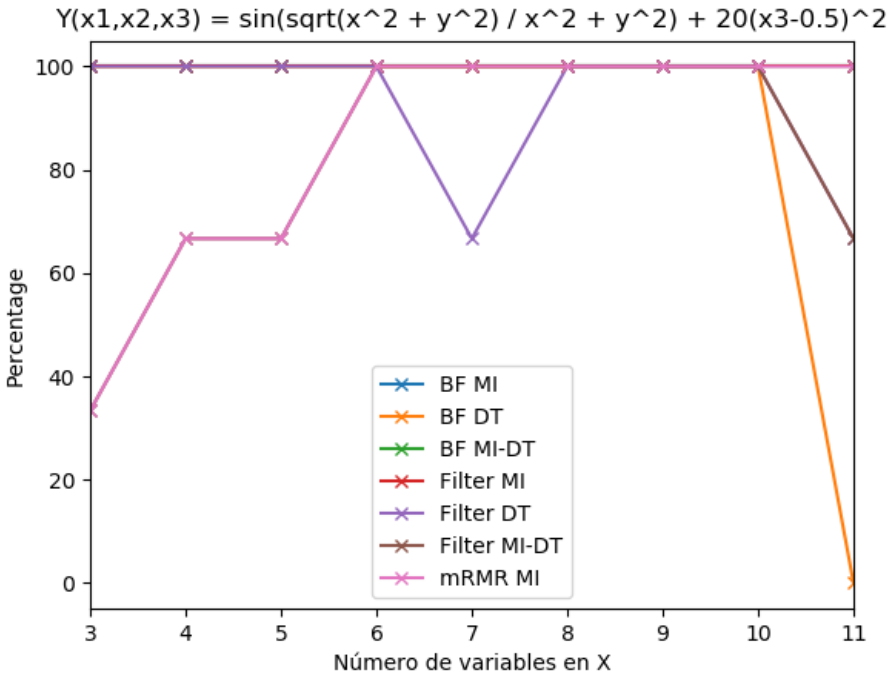

Figura 21.3.: Porcentaje de acierto del ejemplo C con todos los algoritmos.

22. Ejemplos de más dimensiones y ejemplo real

22.1. Ejemplo F: Cuatro dimensiones

Definimos Y como

$$Y(x_0, x_1, x_2, x_3) = \frac{\sin(\sqrt{x_1^2 + x_2^2}}{x_1^2 + x_2^2} + 20(x_3 - 0l5)^2 + \cos(x_4)x4^3$$

que contiene un término al cubo multiplicado por un coseno.

Además de este ejemplo se han hecho 4 variaciones añadiendo ruido con distribución uniforme y con distribución gaussiana de la misma manera. Todas ellas han dado el mismo resultado: los tres algoritmos de Fuerza Bruta seleccionan bien 3 columnas, pero les falta una cuarta columna, para X de 4 a 11 dimensiones.

Por tanto, los tres algoritmos poseen un 75 % de acierto.

22.2. Ejemplo D: Cinco dimensiones

$$Y(x_0, x_1, x_2, x_3, x_4) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0{,}5)^2 + 10x_3 + 5x_4$$

una función clásica de Cherkassky [CM].

MI obtiene un 60 % de acierto, y DT y MI-DT tan sólo un 20 % de acierto.

Para arreglar esto, en este ejemplo se ha aplicado la búsqueda de parámetros α y β sobre el conjunto $\{0{,}5, 1, 2, 3, 5, 10\}$ para ajustar la MI y el DT del MI-DT. En este caso, el parámetro α (MI) debe ser 4 veces superior a β (DT) para que MI-DT llegue al 60 %.

22.3. Ejemplo H: 6 dimensiones

$$Y(x_0, x_1, x_2, x_3, x_4, x_5) = \frac{\sin(\sqrt{x_1^2 + x_2^2}}{x_1^2 + x_2^2} + 20(x_3 - 0l5)^2 + \cos(x_4)x4^3 - (x_4 - x_5)^2$$

Se ha probado con 1000 y 6000 muestras. Aquí MI y MI-DT coinciden en tener siempre un 50 % de acierto. Sin embargo DT tiene siembre un 16 % para 1000 muestras, mantiene también un 50 % hasta que llega a 10 y 11 variables, que se reduce a 33 %.

Este ejemplo será estudiado en el futuro con mayor número de muestras, pues si se consiguiese llegar a una mejor aproximación con Delta Test, MI-DT y su consiguiente búsqueda de parámetros podría verse beneficiado.

22.4. Ejemplo G: 7 dimensiones

$$Y(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = \frac{\sin(\sqrt{x_1^2 + x_2^2})}{x_1^2 + x_2^2} + 20(x_3 - 0{,}5)^2 + \cos(x_4)x4^3 - (x_4 - x_5/x_6)^2$$

Hemos realizado dos pruebas: con 1000 y con 6000 muestras.

Con 1000 muestras, los tres obtienen aproximadamente un 28.58 % de acierto, salvo DT al llegar a 11 variables que obtiene un 42.86 %.

Con 6000 muestras, DT se mantiene en 28.58 %, así como MI-DT, sin embargo MI sube a 42.86 % para todas las variables.

Este ejemplo también será estudiado en el futuro con mayor número de muestras.

22.5. Ejemplo real

Para finalizar, vamos a un ejemplo real de aproximadamente 2500 muestras y 11 variables, donde vamos a aplicar una regresión para ver cuánto se empeora, se iguala o mejora al quedarnos con el subconjunto de mejores variables de cada método de Fuerza Bruta.

Hemos hallado los datos de vino rojo en [For], los hemos leído y les hemos restado la media y dividido por la desviación típica como preprocesamiento de datos. Hemos aplicado los algoritmos de Fuerza Bruta de MI, DT y MI-DT.

Seguidamente, hemos aplicado regresión lineal al conjunto: primero sin seleccionar variables (columnas), que podríamos referirnos como la regresión original que se haría sin reducción de dimensiones, luego con las variables de BF MI, luego BF DT y finalmente BF MI-DT. La tabla 22.1 muestra el porcentaje de aciertos de los distintos métodos. En este caso real es el algoritmo BF MI el que más error obtiene, y de hecho BF MI-DT consigue tener un 10 % más de error comparado al original debido al DT.

	Original	**BF MI**	**BF DT**	**BF MI-DT**
E_out	0.623945	0.972539	0.727282	0.727282

Cuadro 22.1.: E_out.

23. Formato de las tablas

Para visualizar las tablas que se adjuntan con esta memoria (o la carpeta *datos* en [Cab]), el formato en todos los ejemplos transcurre de la siguiente manera: se genera X con d variables, se genera Y, y fijando un $N > d$, desde d hasta N se añade una variable aleatoria más a X (sin alterar Y) y se escribe una fila en la tabla CSV.

Es decir, la primera fila es la información cuando X contiene d variables, la siguiente $d + 1$, y así hasta N filas. d cambia según el ejemplo, pero N se ha fijado a 11.

Cada fila muestra los datos en el siguiente formato (por columnas):

1. Índices de las variables seleccionadas al aplicar el algoritmo de Fuerza Bruta de MI.
2. Índices de las variables seleccionadas al aplicar el algoritmo de Fuerza Bruta del Test Delta.
3. Índices de las variables seleccionadas al aplicar el algoritmo de Fuerza Bruta de la unión entre MI y DT.
4. Vacío.
5. La información mutua obtenida con todas las variables de X.
6. El Test Delta obtenido con todas las variables de X.
7. Vacío.
8. El valor de la MI con las variables seleccionadas al aplicar Fuerza Bruta.
9. La diferencia entre la MI con todas las variables y la MI con las variables seleccionadas.
10. Vacío.
11. El valor de la DT con las variables seleccionadas al aplicar Fuerza Bruta.
12. La diferencia entre la DT con todas las variables y la DT con las variables seleccionadas.
13. Vacío.
14. El valor obtenido al aplicar el algoritmo de Fuerza Bruta de la unión entre MI y DT.
15. Vacío.
16. El tiempo que ha tardado el hacer MI a todas las variables.
17. El tiempo que ha tardado el algoritmo de Fuerza Bruta MI.
18. El tiempo que ha tardado el hacer DT a todas las variables.
19. El tiempo que ha tardado el algoritmo de Fuerza Bruta DT.
20. El tiempo que ha tardado el algoritmo de Fuerza Bruta MI-DT.

21. Vacío.
22. Porcentaje que acierta el algoritmo de Fuerza Bruta de MI.
23. Porcentaje que acierta el algoritmo de Fuerza Bruta de DT.
24. Porcentaje que acierta el algoritmo de Fuerza Bruta de MI-DT.
25. Vacío.
26. Los índices de las variables correctas (las que usa Y en su definición).

Parte X.

Conclusiones.

24. Conclusiones

Hemos hablado de la importancia de la minimización del Test Delta como criterio de selección de variables, sin embargo necesita un gran número de muestras para que éste sea un buen criterio. Como ventaja muy destacable es la velocidad a la que computa, luego puede servir como heurística a la hora de tomar la decisión de si usar MI o MI-DT como criterio de selección de variables. Únicamente si hay un número de muestras suficientemente grande.

MI-DT tiene potencial como método de equilibrar MI y DT, primero habría que intentar averiguar empíricamente ese número de muestras M 'suficientemente grande' para que DT sea un criterio al menos consistente. Con consistente me refiero a que, como en los ejemplos de 1000 muestras, pasaba de 100 % de acierto para 10 variables, y luego a 0 % para 11 variables. MI sí muestra consistencia en ese aspecto, mantiene un incremento o un descenso digamos continuo, o incluso se mantiene constante.

También sería interesante estudiar un MI-DT no tan equitativo, sino primero uno y luego otro. Es decir, que use primero DT para sacar unas posibles buenas variables y modificar MI para que comience buscando por esas variables, reduciendo así el tiempo de cómputo del MI, el cual hemos visto que la Fuerza Bruta de MI es la más lenta. Todo esto siempre bajo la premisa de que hemos hallado una forma de saber si tenemos un número de muestras grande.

Además no solo estudiar el ajuste de parámetros para los MI y los Delta, sino una forma de reescribir el término a maximizar para conseguir una buena selección de variables. Como se ha dicho en este trabajo, si tenemos un gran número de muestras, sería interesante probar un MI-DT que maximice $MI + 1/(delta + \beta)$, con $\beta = 0{,}01$ por ejemplo. Esta fórmula le da gran importancia a Delta si éste es pequeño.

En definitiva, quedan muchos caminos por explorar, y MI-DT puede convertirse en un método más rápido que MI y con iguales o mejores resultados, o bien un método más lento pero con mejores resultados.

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[ALC10] B. Auffarth, M. Lopez, and J. Cerquides. Comparison of redundancy and relevance measures for feature selection in tissue classification of ct images. advances in data mining. applications and theoretical aspects. http://www.csc.kth.se/~auffarth/publications/redrel.pdf, 2010. [Citado en pág. 19]

[BGMS] Kenneth R. Ball, Christopher Grant, William R. Mundy, and Timothy J. Shafer. A multivariate extension of mutual information for growing neural networks. *Neural Networks*, 95:29–43. [Citado en págs. 26 and 27]

[BV] R. Brette and G. Varoquaux. Non-parametric computation of entropy and mutual-information. https://gist.github.com/GaelVaroquaux/ead9898bd3c973c40429. [Citado en págs. 27 and 42]

[Cab] Iñaki Madinabeitia Cabrera. Feature selection based on mutual information, delta test and the proposal of mixing both. https://github.com/InakiMadi/Feature-Selection-Mixing-Mutual-Information-And-Delta-Test. [Citado en págs. 39, 42, 43, 46, and 53]

[CM] Vladimir Cherkassky and Yunqian Ma. Practical selection of svm parameters and noise estimation for svm regression. *Neural Networks*, 17:113–126. [Citado en págs. 47 and 51]

[CT] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Hoboken, New Jersey: Wiley. [Citado en pág. 7]

[Eir] Emil Eirola. Variable selection with the delta test in theory and practice. *Helsinki University of Technology, Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology in the Degree Programme of Engineering Physics and Mathematics.*, pages 553–558. [Citado en págs. 14, 15, and 16]

[ELL$^+$] Emil Eirola, Elia Liitiinen, Amaury Lendasse, Francesco Corona, and Michel Verleysen. *Using the Delta Test for Variable Selection*. European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning Bruges (Belgium). [Citado en págs. 11 and 12]

[Eva] D. Evans. *Data-derived estimates of noise for unknown smooth models using nearneighbour asymptotics*. PhD Tesis, Cardiff University. [Citado en pág. 13]

[For] PARVUS Forina, M. et al. Análisis químico del vino para determinar su origen. https://archive.ics.uci.edu/ml/datasets/Wine. [Citado en pág. 52]

[Hug59] G.F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968, doi:10.1109/TIT.2005.844059. [Citado en pág. 18]

[Jon04] A. J. Jones. New tools in non-linear modeling and prediction. *Computational Management Science*, 1(2):109–149, 2004. [Citado en pág. 11]

[KL68] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951, doi:10.1214/aoms/1177729694, MR 0039968. [Citado en pág. 7]

[KSG] A. Kraskov, H. Stgbauer, and P. Grassberger. *Estimating mutual information*. Phys. Rev. E, 69:066138. [Citado en pág. 13]

[KWW] S. E. Kemp, I. D. Wilson, and J. A. Ware. A tutorial on the gamma test. *School Of Computing*, 6(1 and 2):67–75. [Citado en pág. 11]

[Mac] D.J.C. Mackay. *Information theory, inferences, and learning algorithms*. [Citado en pág. 9]

[McM08] David M. McMahon. *Quantum Computing Explained*. Hoboken, New Jersey: Wiley-Interscience, 2008. [Citado en pág. 6]

[MM] Roberto Maestre and Bojan Mihaljevic. Script to calculate mutual information between two random variables. `https://github.com/rmaestre/Mutual-Information/blob/master/it_tool.py`. [Citado en pág. 23]

[PP94] H. Pi and C. Peterson. Finding the embedding dimension and variable dependencies in time series. *Neural Computation*, 6(3):509–520, 1994. [Citado en pág. 12]

[RGFO 9] Milos Radovic, Mohamed Ghalwash, Nenad Filipovic, and Zoran Obradovic. Minimum redundancy maximum relevance feature selection approach for temporal gene expression data. *BMC Bioinformatics*, 18(9), 2017, doi:10.1186/s12859-016-1423-9. [Citado en pág. 19]

[RHJL] Nima Reyhani, Jin Hao, Yongnan Ji, and Amaury Lendasse. *Mutual Information and Gamma Test for Input Selection*. Helsinki University of Technology - Neural Network Research Center. [Citado en pág. 18]

[Sha8x] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948, doi:10.1002/j.1538-7305.1948.tb01338.x. [Citado en pág. 7]

[SHL] Antti Sorjamaa, Jin Hao, and Amaury Lendasse. Mutual information and k-nearest neighbors approximator for time series prediction. *Neural Network Research Centre*, pages 553–558. [Citado en pág. 27]