Facultad de Ciencias

Aplicación a Física de Partículas de métodos de clasificación multidimensionales en presencia de errores sistemáticos

(Application of multidimensional classification techniques to Particle Physics in the presence of systematic errors)

Trabajo de Fin de Grado
para acceder al

GRADO EN FÍSICA

Autor: Luis Crespo Ruiz

Director: Francisco Matorras Weinig

Co-Director: Pablo Martínez Ruiz del Árbol

Junio - 2020

Resumen

En este trabajo se ha estudiado el manejo de herramientas de aprendizaje automático, analizando aspectos como su optimización o el efecto de los errores sistemáticos. Se han utilizado dos paquetes: RSNNS, dentro del programa R, y TMVA, dentro de ROOT.

Utilizando la herramienta TMVA, se han aplicado diversas técnicas de aprendizaje automático a un problema real de clasificación de sucesos en física de partículas: la búsqueda de partículas de materia oscura producidas en el LHC, donde buscamos separar esta señal de un fondo de colisiones ttbar varios órdenes de magnitud mayor y con una señal experimental muy similar. En este contexto se han usado y comparado varios métodos: redes neuronales, árboles de decisión, SVM, discriminante lineal, discriminante de Fisher y cortes rectangulares.

Se ha propuesto una forma novedosa de estimar cantidades físicas, en este caso la masa de una partícula de materia oscura que escapa del detector, a partir de cantidades medidas relacionadas indirectamente con ellas, por medio de regresión con redes neuronales. Se ha comprobado que sí son capaces de dar un resultado orientativo.

Se han estudiado los efectos de precisión limitada de las variables en los distintos algoritmos. El efecto, como se esperaba, es el de disminuir el rendimiento de la clasificación, especialmente si el entrenamiento se hace con una muestra ideal. Se ha estudiado también el efecto de posibles errores sistemáticos, interpretados como un sesgo en los datos, y se propone un nuevo método basado en *data augmentation* para paliar este efecto, consistente en incluir datos degradados en el entrenamiento. Se consigue la máxima reducción del deterioro si la magnitud de esta degradación es del mismo orden que el error sistemático.

Palabras clave: aprendizaje automático, redes neuronales, materia oscura, regresión, data augmentation

Abstract

In this project machine learning tools have been studied, analyzing aspects like optimization or the effect of systematic errors. Two packages have been used: RSNNS, for the R language, and TMVA, for the ROOT program.

Using the TMVA tool, we have applied several machine learning techniques to a real problem in event classification in particle physics: the search for dark matter particles produced in LHC, where we attempt to separate this signal from a ttbar collisions background several orders of magnitude greater and with very similar experimental signal. In this context we have used and compared several methods: neural networks, decision trees, SVM, linear discriminant, Fisher discriminant and rectangular cuts.

We have proposed a new form of estimating physical quantities, in this case the mass of a dark matter particle which cannot be detected, from measured quantities indirectly related to them, using regression with neural networks. We have proved that they can give an approximate result.

The effects of limited precision of variables in the different algorithms are studied. As it was expected, the main effect is to reduce the classification efficiency, especially if the methods are

trained with ideal samples. We also study the effect of possible systematic errors, interpreting them as a bias in data, and propose a new method based in data augmentation to cut down this decay, which consists in including degraded data in the training sample. The best reduction in the decay is achieved if this degradation has the same magnitude as the systematic error.

Keywords: machine learning, neural networks, dark matter, regression, data augmentation

1. Introducción

1.1 Objetivos

En muchos ámbitos del mundo actual, es necesario manejar conjuntos de datos donde una variable se pone en función de muchas otras, por ejemplo para predecir el comportamiento de un sistema a partir de datos de entrada, o para clasificar un elemento entre varias posibles clases. Estos problemas se pueden dividir en clasificación, si la variable de salida es discreta, o regresión, si es continua.

La clasificación consiste, como cabría esperar, en determinar de qué tipo es un elemento entre varios tipos posibles. La regresión consiste en predecir un valor de salida a partir de un conjunto de valores de entrada.

Cuando el conjunto de datos es muy grande, es conveniente que esta clasificación o regresión la haga una máquina. Sin embargo, para que una máquina funcione es necesario programarla, es decir, decirle de alguna manera qué puntos están en una clase y cuáles en otra, o cuál es la relación entre las variables de entrada y la de salida. El problema es que en muchos casos esta relación no se conoce. Conviene, por tanto, buscar métodos para que la máquina sea capaz de aprender, esto es, que a partir de datos ya clasificados deduzca la relación en cuestión. Esto nos lleva al ámbito del *machine learning*.

De esta forma, mientras en la programación clásica se le da a la máquina unas reglas para que a partir de los datos calcule los resultados, en el *machine learning* se le da a la máquina unos datos con resultados ya conocidos para que deduzca las reglas y a partir de ellas pueda procesar más datos.

Se pueden encontrar ejemplos en reconocimiento de voz, de texto o de imágenes, así como en los experimentos de física de partículas (como en este trabajo).

El objetivo de este TFG es analizar en un problema real, de detección de partículas, el uso de métodos de aprendizaje automático. Queremos saber si el uso de estos métodos permite detectar las partículas que se buscan en un experimento. Este problema es de clasificación, ya que se trata de distinguir entre dos tipos de eventos: los que tienen la partícula que nos interesa descubrir y los que no. También queremos averiguar qué diferencias de funcionamiento hay entre los distintos métodos, y comprobar el efecto de cambios en el número de variables y otros aspectos del clasificador, como el número de neuronas de una red, sobre su eficacia.

Después, se analizará su uso en problemas de regresión, donde lo que se busca no es determinar la presencia o ausencia de una partícula no detectada, sino reconstruir alguna característica suya, en este caso la masa. Se trata de una aplicación menos habitual. No existe una fórmula clara que la determine, pero sí se conoce que hay cierta correlación entre ella y otras cantidades de las partículas que sí se miden.

Un problema que se suele omitir en el *machine learning* a pesar de ser importante es hasta qué punto la muestra se ajusta a la realidad, y si los algoritmos entrenados con datos ideales seguirán

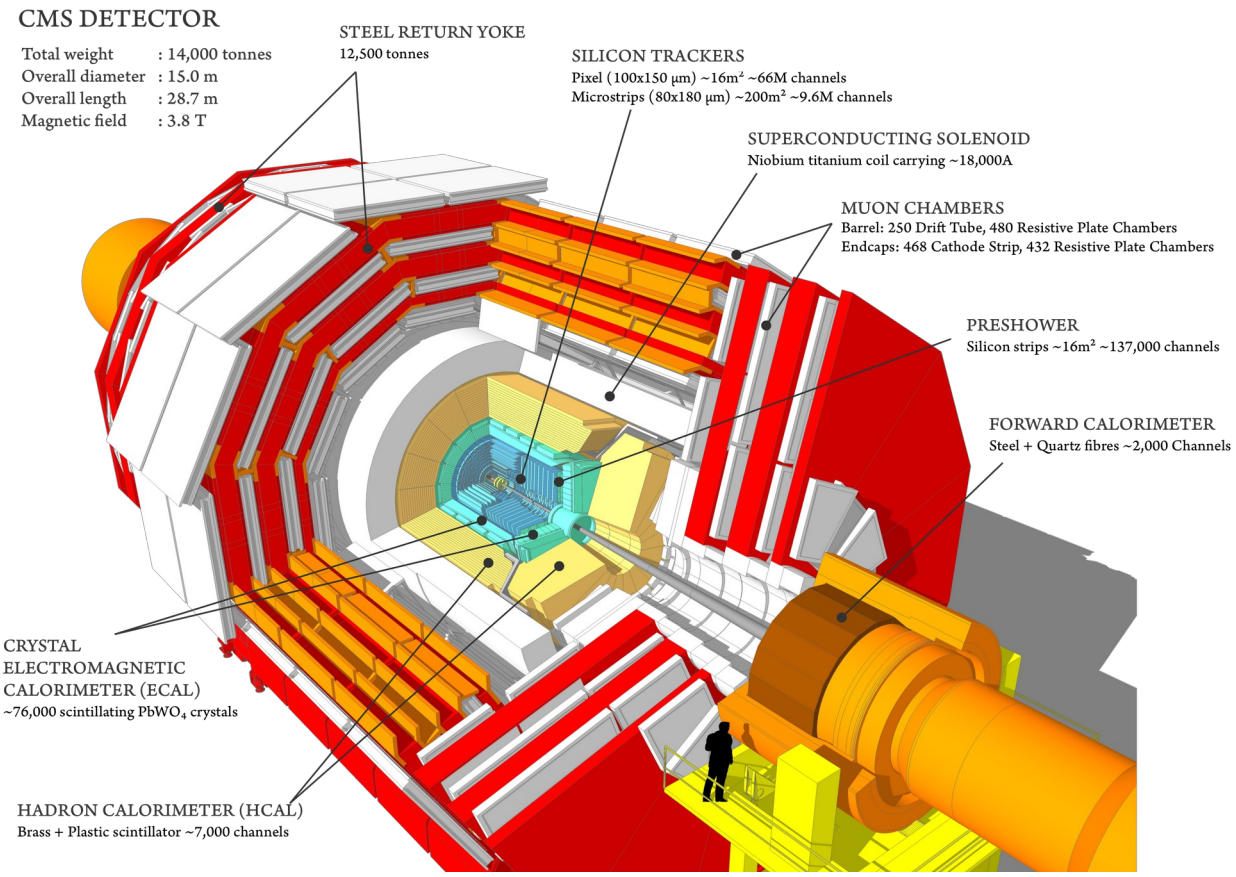


Figura 1: Detector CMS. Fuente: [3]

funcionando sobre una muestra realista. Es decir, cómo afectan los errores, tanto estadísticos como sistemáticos, a estas técnicas. En este trabajo exploramos el efecto de estos errores y proponemos un método de mitigación para estas situaciones.

1.2 Problema físico

El LHC es un colisionador de protones construido entre 1994 y 2008. En un túnel de 27 kilómetros se hacen circular dos haces de protones en sentidos contrarios, que colisionan en cuatro puntos donde hay situados detectores: ATLAS, ALICE, LHCb y CMS.

El detector CMS [1-3] es un aparato diseñado para los experimentos de física de altas energías en el LHC. Fue construido para satisfacer las demandas de una buena identificación y medición del momento de muones, buena resolución de energía electromagnética y momentos de fotones y electrones y buena resolución de jets de hadrones y “missing ET”. Los detectores están rodeados por un imán superconductor de 3,8 T. Los detectores que se incluyen son:

- Detector de silicio: da una medida precisa de las partículas que salen del punto de interacción y una reconstrucción de los vértices. Se compone de un “detector de pixel” de silicio, de radio desde 4,4 hasta 10,2 cm desde el punto de interacción, y un *tracker,* también de silicio, que llega hasta una distancia de 1,1 m. En este detector dejan traza las partículas cargadas, de las que se reconstruye la trayectoria.

- ECAL: es un calorímetro hermético y homogéneo formado por cristales de $PbWO_4$, que llega hasta 1,77 m de distancia del centro. Cuando un electrón o un fotón entra en el calorímetro, inicia una cascada de *bremsstralung* y producción de pares.
- HCAL: es un calorímetro para detectar hadrones, que llega hasta 2,95 m de distancia. Cuando un hadrón entra en esta parte del detector, interactúa mediante fuerza nuclear fuerte con los núcleos atómicos, lo que produce más hadrones, que a su vez interactúan, y así sucesivamente.
- Cámaras de muones: son detectores colocados en la cobertura de acero del solenoide superconductor, que permiten aumentar la resolución de los muones respecto a la obtenida en el detector de silicio central. Hay tres tipos de detectores: cámaras de tubo de deriva (DT), cámaras de franjas catódicas (CSC) y cámaras de plato resistivo (RPC).

En conjunto, las partículas que se detectan son: todas las que tienen carga eléctrica, todos los hadrones y los fotones, siempre que vivan lo suficiente para llegar a los detectores. Esto deja como detectables los protones, neutrones, electrones, muones, fotones, piones y kaones.

Los eventos se producen a un ritmo muy alto (veinte colisiones en el punto de interacción a la vez, con unas mil partículas saliendo de cada vértice, cada 25 ns), lo que hace necesaria una selección automática o *trigger*. La primera fase de la selección se hace mediante hardware especializado que elige los eventos más interesantes usando información de los sensores. De esta primera capa salen 100000 eventos por segundo, y luego se filtran todavía más mediante una granja de procesadores que reconstruye los eventos y aplica sobre ellos nuevos criterios, reduciendo la tasa a menos de 1000 por segundo. El algoritmo de reconstrucción se llama Particle Flow.

Hay algunas partículas que es imposible detectar en CMS: los neutrinos y la materia oscura. Sin embargo, es posible concluir que existe algo que no estamos detectando si la suma de todos los momentos no es 0 e inferir sus propiedades por argumentos de conservación de momento y energía. Este valor, el “momento lineal faltante” o “missing ET”, se asocia a las partículas no detectadas.

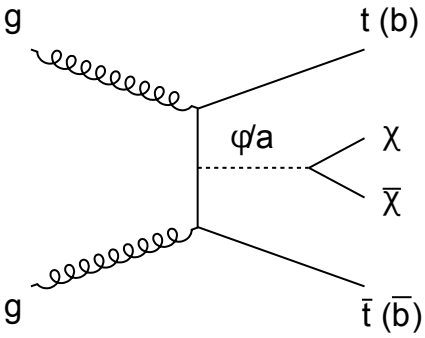
La materia oscura [4] es un tipo de materia no bariónica, que no presenta interacción electromagnética con la materia ordinaria, cuya existencia se ha deducido de la distribución de velocidades de rotación de las galaxias. Hasta ahora no existe evidencia experimental de la materia oscura, pero hay procesos físicos donde se cree que puede producirse. En muchos modelos, la materia oscura está formada por WIMP (partículas masivas que interaccionan débilmente), que se escaparán del detector produciendo momento lineal faltante.

En concreto, el proceso “ttbar” consiste en la producción de un quark top y un antiquark top. Cada uno de ellos se desintegra en un quark (o antiquark) bottom y un bosón W, del que finalmente se produce un leptón y un neutrino.

$$g+g \rightarrow t+\bar{t} \rightarrow \left(b+l^{+}+\nu_l\right)+\left(\bar{b}+l^{-}+\bar{\nu}_l\right)$$

donde *l* puede ser un electrón, muón o tau.

En un proceso parecido [2, 3] se produce, además de las partículas mencionadas, una de materia oscura:



$$g+g \rightarrow t+\bar{t}+\varphi \rightarrow (b+e^{+}+\nu_e)+(\bar{b}+e^{-}+\bar{\nu}_e)+\chi+\bar{\chi}$$

No es posible detectar las partículas de materia oscura ni los neutrinos, de modo que en ambos casos lo que se detecta en el CMS es un quark bottom y su correspondiente antiquark (o más bien los chorros de partículas procedentes de ellos) y un leptón y un antileptón. Por ello, en este trabajo vamos a usar *machine learning* para intentar distinguir entre ellos.

1.3 Métodos de clasificación y regresión multidimensionales

En los problemas de clasificación las muestras vienen divididas en dos clases: eventos de señal y eventos de fondo. En nuestro problema, la señal son los eventos de materia oscura que hay que detectar, y el fondo los que producen solo quarks top. Dentro de la señal, a su vez, hay eventos donde la masa de la partícula oscura φ toma varios valores distintos: esto lo usaremos para hacer regresión.

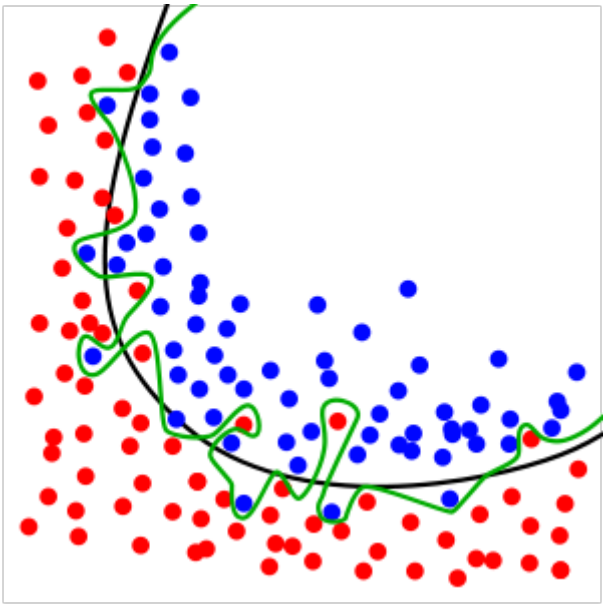


Figura 2: Resultado de un clasificador sobreentrenado. En verde, la línea generada por el clasificador, y en negro la separación correcta. Como se ve, la línea tiene una complejidad artificiosa basada únicamente en la muestra de entrenamiento.

Si se toma una muestra de datos y se entrena el clasificador con ese conjunto, corremos el riesgo de que el clasificador aprenda a distinguir entre los puntos de la muestra de las distintas clases, pero no consiga clasificar correctamente puntos que no están en la muestra, como los puntos en los que luego se tendrá que usar. Esto se llama sobreentrenamiento. Si ocurre, puede ser que el clasificador

no sea el adecuado, o que se haya entrenado demasiado. Para saber si un clasificador está sobreentrenado o no, no se puede entrenar con toda la muestra de la que se dispone sino solo con una parte (normalmente es la mitad o algo más), y usar la otra parte para probar el clasificador: si en este segundo conjunto funciona peor, está sobreentrenado. Estos dos conjuntos se llaman de entrenamiento y de prueba.

Los métodos que hemos utilizado para clasificación y regresión en varias variables son (las figuras están sacadas de [6]):

- Cortes rectangulares: consiste en encontrar un conjunto de intervalos de las variables que separen eventos de señal de eventos de fondo. Para cada valor buscado de eficiencia de señal se busca un producto de intervalos que contengan ese porcentaje de los eventos de señal y que contengan el menor número posible de eventos de fondo. Una forma de optimizar es mediante el "algoritmo genético": generar inicialmente un conjunto de cortes con valores aleatorios, quedarse con los mejores y cruzarlos como se cruzan individuos. Otros métodos son el muestreo de Monte Carlo o el enfriamiento simulado. [6, sec. 6]

- Discriminante lineal: si x_i son las variables de entrada, la salida se define como

$$y = \sum_{i=1}^{n} w_i x_i + w_0$$

  donde w_i son los pesos que hay que ajustar para minimizar una función de error. Esto se puede resolver como un sistema de ecuaciones, llamadas "ecuaciones normales":

$$w = \left(X^T X \right)^{-1} X^T Y$$

- Discriminante de Fisher: el objetivo es el mismo que el lineal, pero ahora los w_i no se encuentran resolviendo un sistema de ecuaciones, sino a través de la matriz de covarianza. [7] La matriz se descompone en una matriz dentro de la clase *W* y una matriz entre clases *B,* y con ellos se calcula

$$F_k = \frac{\sqrt{N_S N_B}}{N_S + N_B} \sum_{l=1}^{n_{var}} W_{kl}^{-1} \left(\bar{x}_{S,l} - \bar{x}_{B,l} \right)$$

  donde las $\bar{x}_{S,l}$ y $\bar{x}_{B,l}$ son las medias de señal y fondo respectivamente, y N_S y N_B el número de eventos de señal y fondo. El valor de salida para un *x* es

$$y = F_0 + \sum_{k=1}^{n_{var}} F_k x_k$$

- SVM (*Support Vector Machine*): en su versión lineal [8], el objetivo es encontrar un hiperplano que maximice la distancia a los puntos.

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0$$

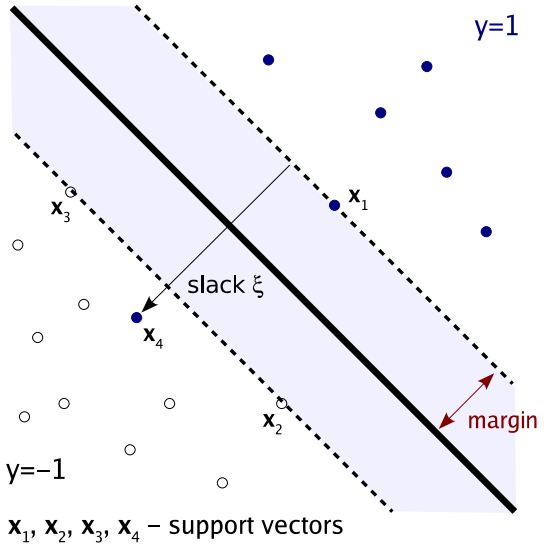


Figura 3: SVM

donde $\vec{w}$ y *b* son los parámetros a ajustar. Esto se puede hacer buscando el máximo de

$$L(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i\,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

y luego tomando

$$\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

También se puede hacer un SVM no lineal: el truco es aplicar una transformación a los eventos y luego resolver el problema lineal [9]. En concreto, aquí se está cambiando el producto escalar de arriba por una función gaussiana.

$$L(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i\,j} \alpha_i \alpha_j y_i y_j \exp\left(\frac{-|\vec{x}_i - \vec{x}_j|^2}{2\sigma^2} \right)$$

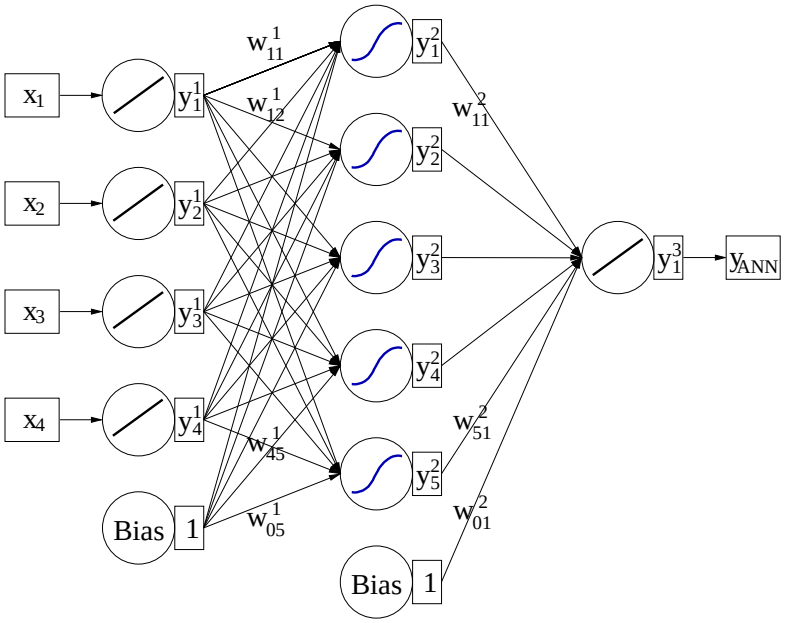


Figura 4: Red neuronal

- MLP (*MultiLayer Perceptron*): se trata de las redes neuronales. Una red neuronal es un conjunto de neuronas organizadas en capas (ver figura 4). Aunque hay otras formas de redes neuronales, en este contexto cada neurona toma como entrada todas las de la capa anterior, excepto las de la primera capa, que son las variables de entrada. Dada una neurona, que tiene como entradas x_i, la salida se calcula como

$$y = f(z) = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

  La última capa contiene una sola neurona, que es la salida de la red. Aquí, la función *f* se llama función de activación de la red, que puede ser no lineal, y los valores de *w* y *b* son los que se deben optimizar. Para ello, hay varias estrategias:

  - *Backpropagation* o descenso gradiente, que consiste en calcular la derivada de la función de pérdida *L* (aquí, el error cuadrático) respecto de los parámetros *w* y *b* de las neuronas y corregir estos parámetros proporcionalmente al gradiente. La razón de proporcionalidad se llama *learning rate*. Si hay una capa oculta,

$$\Delta w_j = \alpha \frac{\partial L}{\partial w_j} = \alpha L'(y) \frac{\partial y}{\partial w_j} = \alpha L'(y) f'(z) y_j$$

    para la capa superior,

$$\Delta w_i = \alpha \frac{\partial L}{\partial w_i} = \alpha L'(y) \sum_{j=1}^{n} \frac{\partial y}{\partial y_j} \frac{\partial y_j}{\partial w_i} = \alpha L'(y) \sum_{j=1}^{n} f'(z) w_j f'_j(z_j) x_i$$

    para la inferior, donde *y* es la salida de la red e y_j la de la neurona inferior.

  - Descenso gradiente con momento, donde la variación de los pesos se determina modificando la variación anterior.

$$\Delta w_i = \beta \Delta w_{i\,previo} + \alpha \frac{\partial L}{\partial w_i}$$

  - Descenso por gradiente estocástico: en vez de usar todo el conjunto de entrenamiento para calcular la dirección de descenso, se calcula solo con una parte del conjunto llamada *batch*, que va cambiando en cada iteración.

  - BFGS: se diferencia de los anteriores en que usa las segundas derivadas de la función de pérdida respecto de los pesos (matriz hessiana) para estimar la posición del mínimo. [10]

- BDT (*Boosted Decision Trees*): un árbol de decisión (ver figura 5) está formado por nodos, cada uno con dos hijos, y para clasificar un evento se empieza por la raíz del árbol y se realiza una comparación involucrando a una variable. En función del resultado se va al hijo de la izquierda o al de la derecha, y así se sigue hasta llegar a una hoja, que determina el resultado. Un clasificador BDT consiste en un conjunto de árboles de decisión, cada uno con un peso. Para calcular la salida se combinan linealmente las salidas de los árboles con sus pesos: esto se llama *boost*.

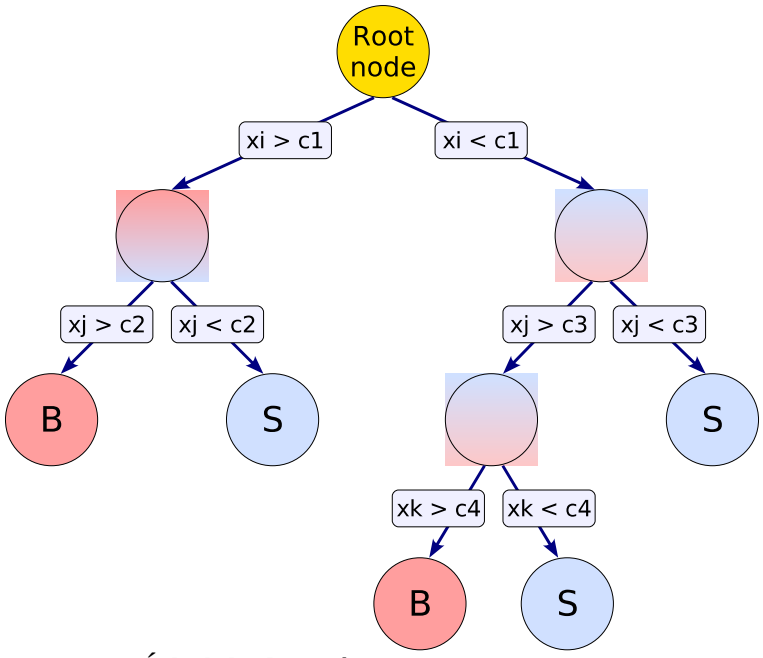


Figura 5: Árbol de decisión

El *boost* [11] consiste en crear un nuevo árbol a partir de los mismos eventos pero dándoles un peso distinto. En concreto, los eventos que fueron mal clasificados en un árbol reciben un peso mayor al entrenar el siguiente: se multiplica por

$$\alpha_i = \frac{1-e}{e}$$

donde *e* es la tasa de error (fracción de eventos mal clasificados) del árbol anterior. La clasificación conjunta viene dada por

$$y(x) = \frac{1}{N}\sum_{i=1}^{N} \ln(\alpha_i) h_i(x)$$

donde $h_i(x)$ es el resultado de una clasificación (1 o -1, según si es señal o fondo).

De estos métodos, los tres últimos son no lineales, de modo que permiten hacer una clasificación cuando la relación que está detrás es no lineal, mientras que el discriminante lineal y el de Fisher la tratarían de aproximar por una función lineal.

No todos los métodos funcionan igual con todos los problemas. Una buena idea es probar varios métodos y elegir el que funciona mejor. En cada método hay algunos parámetros a los que se da valor en el ajuste y otros que tienen valores fijos desde el principio. A estos se les llama hiperparámetros. Un hiperparámetro obvio es el método que se está eligiendo. En el caso de las redes neuronales, el número de capas y el número de neuronas en cada capa son hiperparámetros,

mientras que los coeficientes de cada neurona son parámetros de ajuste. Hay métodos, como el BDT, que pueden dar buen rendimiento recién elegidos pero que no mejoran significativamente al cambiar los hiperparámetros. En cambio, otros métodos como las redes neuronales no dan un rendimiento tan bueno al principio pero funcionan mucho mejor cuando se encuentran valores óptimos para los hiperparámetros.

Otro aspecto a tener en cuenta es que los datos reales, en algunos experimentos como el que estamos llevando a cabo, no son perfectos sino que pueden estar afectados de errores, de forma que su valor real se distribuye alrededor de una media que puede o no ser el valor correcto. Algunos métodos pueden ser más sensibles que otros a esto; nosotros no sabemos, en principio, cuáles son, pero en este trabajo lo pretendemos averiguar.

Tal y como he explicado, para ejecutar un método de este tipo es necesario dividir cada una de las muestras en dos: una para entrenamiento y otra para prueba. Una vez hecho esto, se realiza el entrenamiento, que depende del método. En algunos casos (redes neuronales, métodos lineales) consiste en ajustar los parámetros de los métodos para minimizar una función de error. Aquí, como función de error hemos tomado el error cuadrático, es decir, la media de los cuadrados de las diferencias entre la salida de los métodos (que es un número) y el valor correcto de 0 o 1, según si es señal o fondo.

En el caso de las redes neuronales, la evolución del error a lo largo del entrenamiento se llama curva de aprendizaje, y de ella se puede deducir si la red está bien entrenada: si los valores no se han estabilizado al final del entrenamiento, no está suficientemente entrenada, pero si las curvas correspondientes a entrenamiento y prueba se han separado, hay sobreentrenamiento.

Después, los métodos entrenados se prueban sobre el conjunto de prueba. Como el resultado de los métodos (excepto para el método de cortes) es un número, la clasificación de un evento como señal o fondo requiere determinar un punto límite tal que por encima de él la clasificación se considere señal y por debajo fondo. Dependiendo de este punto límite, se tendrán algunos eventos de señal clasificados como señal y otros como fondo, y lo mismo el fondo. Lógicamente, cuanto más alto sea el punto de corte, más eventos de señal se clasificarán como señal, pero también habrá más falsos positivos (eventos de fondo clasificados como señal). La representación de la fracción de eventos de señal bien clasificados frente a la de eventos de fondo bien clasificados se llama curva ROC (*Receiver Operating Characteristic*).

Existen muchos programas de clasificación y regresión multidimensionales. En este trabajo voy a manejar dos de ellos:

- RSNNS: se trata de una implementación en el lenguaje R del sistema SNNS (*Stuttgart Neural Network Simulator*), que permite entrenar y probar redes neuronales. [5]
- TMVA (*Toolkit for MultiVariate Analysis*): es un paquete del programa ROOT. Consiste en un conjunto de clases de C++ para manejar datos en muchas variables, especialmente los que proceden de experimentos en física de partículas, y contiene, entre otros, todos los métodos mencionados de clasificación y regresión. [6]

2. Pruebas de redes neuronales en RSNNS

El objetivo de este capítulo es probar con ejemplos sencillos los aspectos que queremos estudiar en el trabajo: la clasificación, la regresión y el data augmentation para mitigar errores sistemáticos. Utilizaremos redes escritas en RSNNS.

2.1 Optimización en problemas de clasificación

Se estudió este problema con varios ejemplos, paso a describir el más representativo: se tienen 16 conjuntos de puntos, centrados en las posiciones $(i, j), 0 \leq i \leq 3, 0 \leq j \leq 3$ y en cada conjunto se eligen 100 puntos con esa media y desviación típica 0.2. Se asigna cada una de ellas a una clase: los puntos con centro (*i, j*) se asignan a la clase A si i y j son los dos pares o los dos impares, y a la clase B si no. Entonces, las clases están “alternadas”, y a un clasificador sencillo (por ejemplo lineal), le va a costar mucho distinguir entre las clases. Vamos a ver si las redes neuronales de RSNNS distinguen entre ellos.

En primer lugar, he elegido el método de optimización de descenso gradiente, y he variado el número de neuronas en la capa oculta de la red hasta conseguir el mejor ajuste. El número óptimo era 12 neuronas. En la figura 6 se muestra la curva de aprendizaje y el resultado de la red.

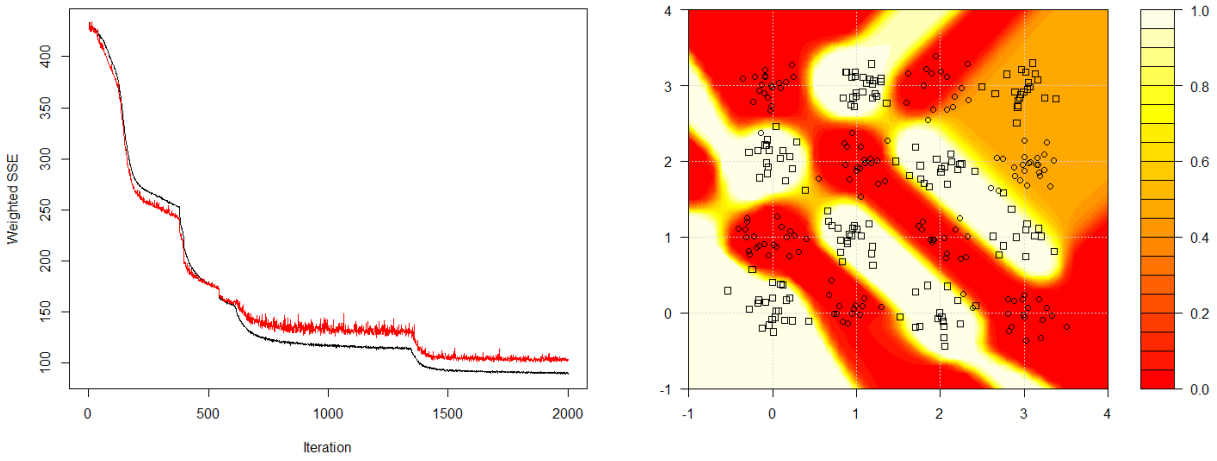


Figura 6: Curva de aprendizaje (izquierda) y resultado (derecha), representado con colores de 0 (rojo) a 1 (blanco), de la red neuronal con 12 neuronas en la capa oculta, entrenada con el método de descenso gradiente. La línea negra es el error de entrenamiento y la roja el de prueba en función de la iteración. Los puntos que se muestran son los del conjunto de prueba: los círculos son los de una clase (para los que el valor ideal es 0) y los cuadrados los de la otra (para ellos el resultado ideal es 1).

La curva de aprendizaje muestra un sobreentrenamiento. En la respuesta de la red, hay dos conjuntos, los que tienen centro en (3, 2) y (3, 3), para los que la red no está respondiendo 0 ni 1 (la respuesta correcta sería 0 para el conjunto de (3, 2) y 1 para (3, 3)). Esto indica que algo no está yendo bien en la clasificación. Quizá haya caído a un mínimo local. Probando el método de descenso con momento, el número óptimo fue otra vez 12 neuronas, y el resultado es el de la figura 7. Claramente ha mejorado: ahora los 16 conjuntos están bien clasificados.

He intentado también el método de gradiente estocástico, resultando que necesita menos neuronas (solo 7 en la capa oculta), y consigue clasificar bien (otro detalle que he observado es que la curva de aprendizaje da muchas menos fluctuaciones). El resultado está en la figura 8; la clasificación es igualmente correcta.

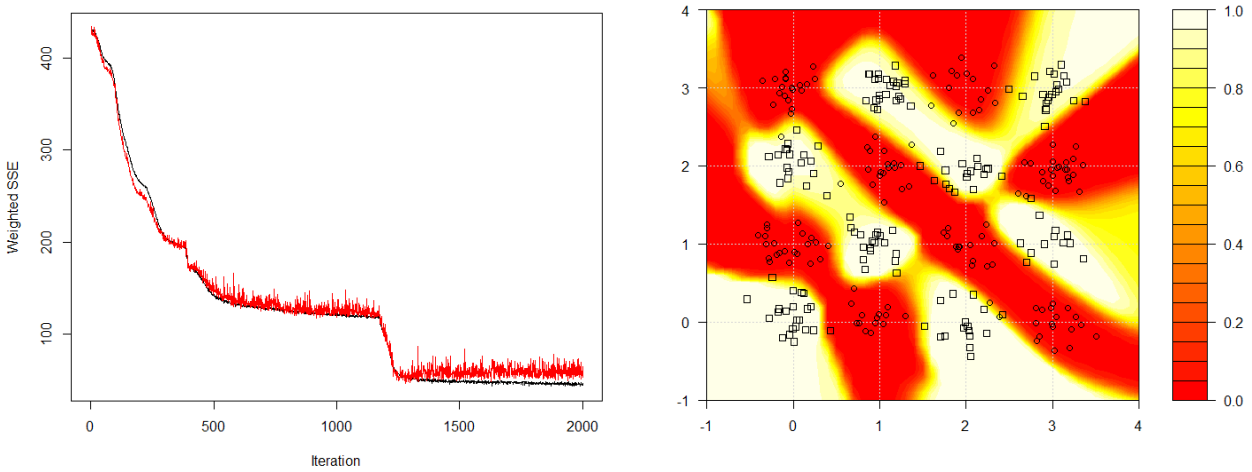


Figura 7: Curva de aprendizaje (izquierda) y resultado (derecha), representado con colores de 0 a 1, de la red neuronal con 12 neuronas en la capa oculta, entrenada con el método de descenso gradiente con momento.

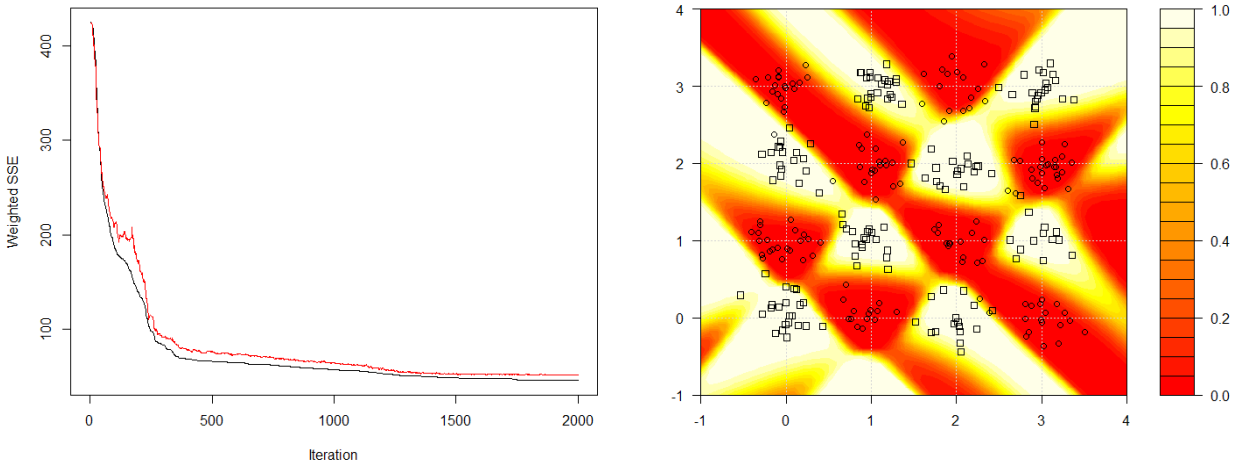


Figura 8: Curva de aprendizaje (izquierda) y resultado (derecha), representado con colores de 0 a 1, de la red neuronal con 7 neuronas en la capa oculta, entrenada con el método de gradiente estocástico.

2.2 Efecto de errores sistemáticos

A continuación, para probar el efecto de un error en los datos, se ha probado la red en una muestra distorsionada. En concreto, la desviación típica de la segunda clase es 0.3 en la dirección x en vez de 0.2. El resultado es el que se espera: los datos más desviados acaban en otra clase y son mal

clasificados, tal y como se muestra en la figura 9. Los círculos son los puntos de la primera clase y los cuadrados los de la segunda. Los puntos en rojo son los que la red clasifica mal.

Lo mismo pasa si en vez de aumentar una desviación típica cambio una media de sitio: en la figura 10 se muestra el resultado de cambiar la media del conjunto de (1, 2) 0.4 en dirección x y 0.2 en dirección y.

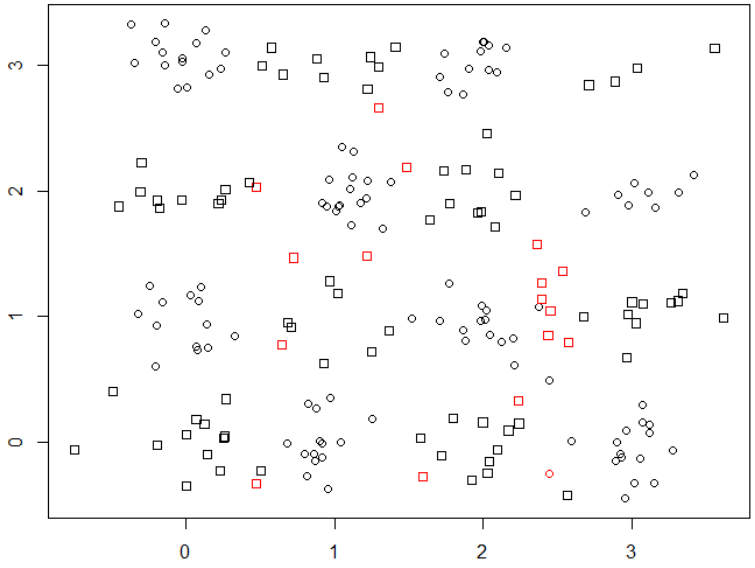


Figura 9: Resultado de la clasificación del conjunto de prueba con una desviación mayor en la segunda clase (en rojo los elementos mal clasificados)

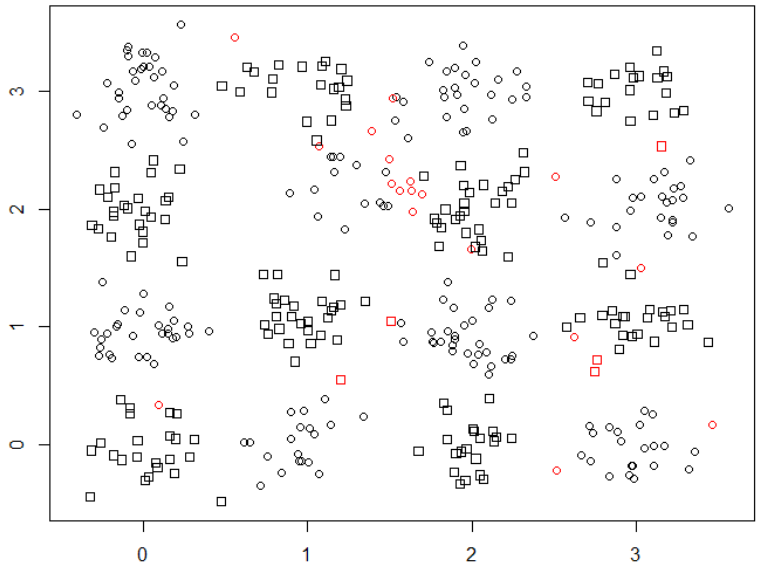


Figura 10: Resultado de la clasificación del conjunto de prueba con la media del conjunto (1, 2) desplazada a (1.4, 2.2) (en rojo los elementos mal clasificados)

La siguiente prueba consiste en tratar de corregir esa pérdida usando *data augmentation*: añadir datos al conjunto de entrenamiento para que la red aprenda que los datos puedan variar y una desviación mayor en el conjunto de prueba no tenga tanto efecto. Estos datos se generan a partir de los que ya se tienen, añadiendo por cada dato *n* copias a las que se añade una desviación adicional. Aquí he probado con una desviación adicional de 0.1 y 0.2, igual a la que ya tiene la clase, y he ido variando el valor de *n* hasta encontrar el óptimo, que estaba en 4 y 5, respectivamente. El resultado para desviación adicional 0.2, que es el que mejor funciona, está en la figura 11. El porcentaje de clasificaciones correctas pasa de 82% a 89% (el valor ideal es 96%), es decir, se ha corregido parte del problema.

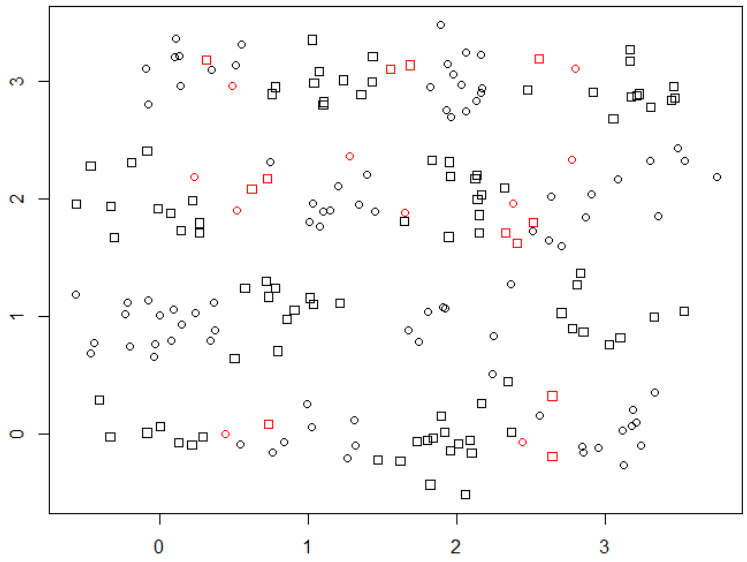


Figura 11: Resultado de la prueba en una muestra distorsionada de una red entrenada con 5 repeticiones de cada dato, con una desviación adicional de 0.2

2.3 Regresión

En las siguientes pruebas, la red neuronal se está usando para regresión: el cálculo de un valor de salida a partir de los datos de entrada. Aquí los elementos del conjunto de entrenamiento no van asociados a una clase, sino que llevan un valor de salida "correcto", que es el que queremos que saque la red. En mi problema, el valor correcto de salida para un punto (x, y) es $\sqrt{x^2+y^2}$. Esta regla es sencilla, pero no es fácil de obtener para un algoritmo, especialmente para uno lineal. Buscamos hasta qué punto la red puede obtenerlo.

He elegido puntos que tienen este valor, el radio, de 5 a 15 con una desviación típica de 1, y ángulo uniforme de 0 a 2π. Como he observado, para este problema el número de neuronas en la capa oculta no afecta al resultado de la red; he usado 20 neuronas. En la figura 12 se representa la curva de aprendizaje (en este caso ha bajado muy deprisa, pero sin sobreentrenar) y el resultado de la regresión frente al valor correcto. Se observa que el valor calculado sigue aproximadamente el correcto, aunque en los extremos hay una cierta saturación.

Se estudió también la capacidad de extrapolación de la red, es decir, si después de ser entrenada con puntos cuyo valor de salida está entre 5 y 15 puede también dar valores correctos para puntos fuera del rango. En la figura 13 he representado el valor de salida de la red (la misma red que he entrenado antes) en función de las variables de entrada, con colores desde 5 (rojo) hasta 15 (blanco), que la red retornaba como 0 y 1, a la izquierda hasta radio 15 y a la derecha hasta radio 100. El comportamiento de la red en estos casos es de saturación: para los elementos de radio menor que 5 la salida es 5 (parte interna de la figura de la izquierda) y para radios mayores que 15 da siempre 15 (parte externa de la derecha).

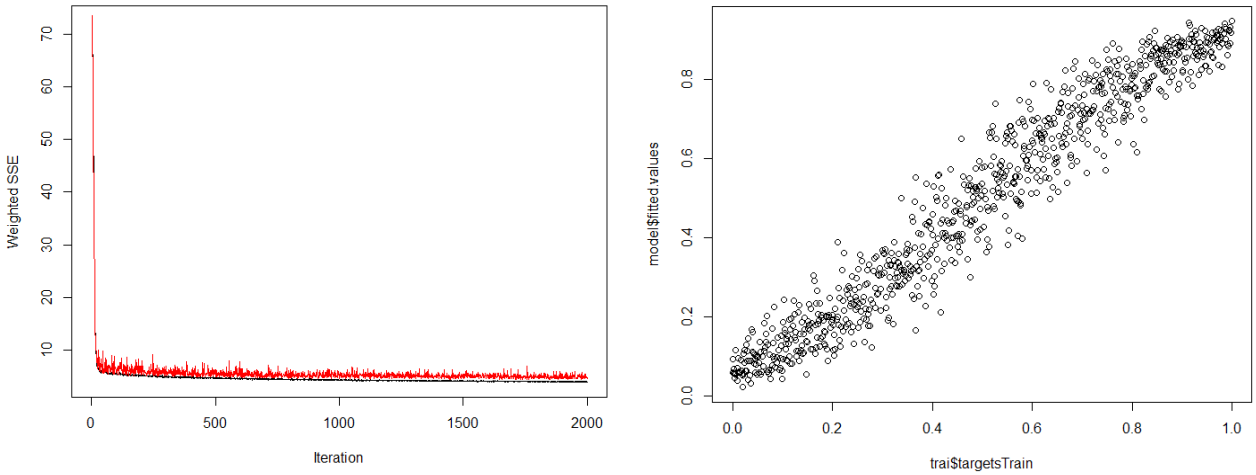


Figura 12: Curva de aprendizaje (izquierda) y representación del resultado frente al valor correcto (derecha) de la red neuronal de regresión entrenada con 20 neuronas en la capa oculta

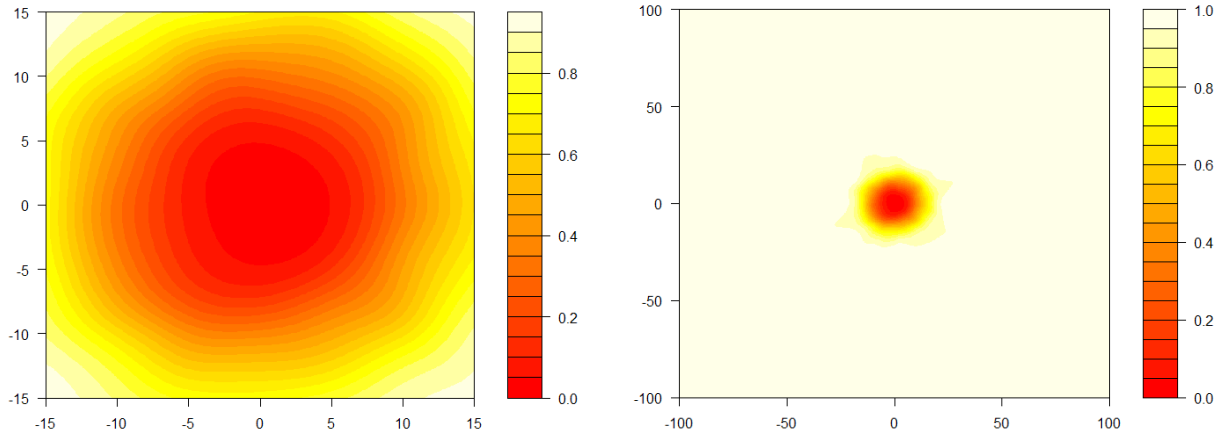


Figura 13: Resultado de la red de regresión hasta radio 15 (izquierda) y 100 (derecha)

2.4 Conclusiones

En conclusión, las redes neuronales permiten, si su arquitectura se elige bien y con un buen entrenamiento, resolver problemas que de otra forma serían complicados para la inteligencia artificial, en especial para los métodos lineales, y esto se cumple tanto para clasificación como para regresión, aunque tienen limitaciones, como la de que no pueden extrapolar. En caso de que haya un error en los datos, la eficiencia de la red disminuye, pero se puede recuperar parte de ese deterioro mediante el *data augmentation*.

3. Clasificación

3.1 Introducción

En este capítulo, vamos a usar el paquete TMVA para intentar resolver el problema físico planteado en la sección 1.2, que consiste en distinguir eventos donde se produce materia oscura de eventos donde no se produce, a partir de magnitudes que se pueden medir en un clasificador como CMS.

El objetivo de este estudio es encontrar si algún método de los que tenemos disponibles funciona mejor, o si los que tenemos se pueden mejorar. También queremos comprobar el efecto de añadir o eliminar variables, y de definir nuevas variables de acuerdo a la física del problema.

3.2 Procedimiento

En primer lugar, para poder realizar aprendizaje automático hacen falta datos. El problema es que cualquier medición que se haga no va a poderse diferenciar entre “ttbar” y “ttbar+materia oscura”. La solución a la que se ha recurrido es generar las muestras artificialmente usando el programa *Madgraph*, para simular un suceso físico (esto se parece al detector CMS pero sin errores). Como no conocemos la masa de la partícula que se produce de materia oscura, vamos a fijar dos masas posibles: 10 y 100 GeV, aparte de la muestra de fondo. Más tarde, cuando hagamos regresión, usaremos masas de 10, 50, 100, 200, 300 y 500 GeV.

En las redes neuronales, los principales hiperparámetros a ajustar son el número de variables de entrada, la topología de la red (número de neuronas por capa), el número de iteraciones y el “learning rate”. Las dos últimas se pueden variar conjuntamente: un “learning rate” alto con menos iteraciones es casi equivalente (aunque lleva menos tiempo) a un “learning rate” más bajo con más iteraciones. Aquí no se ha variado el “learning rate”, sino que se ha mantenido constante en 0.01.

Para variar el número de neuronas, teniendo en cuenta que TMVA permite construir varios clasificadores a la vez, se han usado tres redes neuronales, una con 8 neuronas en la capa oculta, otra con 8 neuronas en una capa oculta y 4 en otra, y otra con 12 neuronas en una capa oculta y 4 en otra. Como también estamos interesados en comparar los distintos métodos, hemos incorporado más clasificadores al sistema: además de las tres redes neuronales, un BDT, un SVM, un discriminante lineal, un discriminante de Fisher y un clasificador mediante cortes rectangulares, en total 8 métodos distintos. Aunque al principio no daban buenos resultados, los empezaron a dar cuando normalizamos las variables de entrada.

Para encontrar los mejores valores del número de iteraciones, he recurrido a la gráfica de convergencia de la red: si el error de ajuste sigue bajando al final del entrenamiento, significa que las iteraciones no son suficientes, mientras que si la curva se ha vuelto plana hay demasiadas iteraciones. Probando varios valores, he llegado a la conclusión de que los valores óptimos son 200 para la red con una sola capa oculta y 400 para las otras.

Respecto al número de variables, en principio he usado todas las que se medirían en un detector como el CMS, que son las de la Tabla 1. Estas variables incluyen: el momento transverso de los

Variable	Significado
ptb1	Momento de un quark bottom
etab1	Pseudorapidez del quark
phib1	Ángulo polar del quark
ptb2	Momento del otro quark
etab2	Pseudorapidez del otro quark
phib2	Ángulo polar del otro quark
ptlep1	Momento de un leptón
etalep1	Pseudorapidez del leptón
philep1	Ángulo polar del leptón
ptlep2	Momento del otro leptón
etalep2	Pseudorapidez del otro leptón
philep2	Ángulo polar del otro leptón
ptMET	Módulo del momento lineal faltante
etaMET	Pseudorapidez del momento lineal faltante
phiMET	Ángulo polar del momento lineal faltante
mMET	Masa invariante del momento lineal faltante

Tabla 1: Significado de las variables disponibles en las muestras generadas con Madgraph

quarks y leptones, su pseudorapidez (definida como $-\ln\left(\tan\frac{\theta}{2}\right)$ donde θ es el ángulo que forma con la dirección inicial) y los ángulos polares (dirección en el plano transverso en que ha salido la partícula), junto con el momento faltante, su pseudorapidez, su ángulo polar y su masa invariante.

3.3 Resultados de la clasificación para masa 100

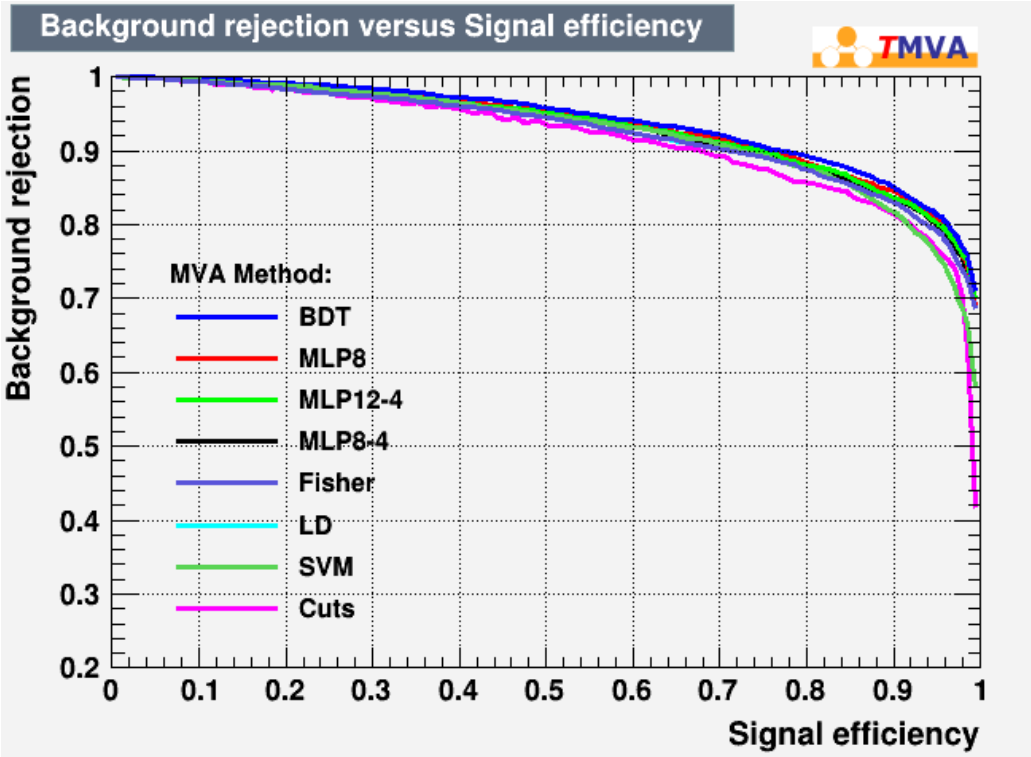


Figura 14: Curva ROC de los clasificadores entrenados con todas las variables

Usando todas las variables de las que disponemos, la curva ROC de la clasificación de los eventos con masa 100 se representa en la figura 14. En esta curva se representa la fracción de eventos de fondo bien clasificados frente a la fracción de eventos de señal bien clasificados (cada línea representa un clasificador). Por lo tanto, un clasificador es mejor cuanto más cerca pase su curva ROC de la esquina superior derecha. Dependiendo de qué queramos hacer, puede que nos interese más la parte de la curva cercana a 100% de rechazo de fondo (superior izquierda) o la cercana a 100% de detección de señal (inferior derecha), por lo que en nuestro análisis estudiaremos distintos puntos de la curva, correspondientes a 99%, 90% y 70% de rechazo de fondo.

Clasificador	Eficiencia a 99% de fondo (test set)	Eficiencia a 99% de fondo (train set)	Eficiencia a 90% de fondo (test set)	Eficiencia a 90% de fondo (train set)	Eficiencia a 70% de fondo (test set)	Eficiencia a 70% de fondo (train set)
Cortes	0.097	0.152	0.673	0.650	0.981	0.989
Lineal	0.138	0.146	0.712	0.706	0.991	0.990
Fisher	0.138	0.146	0.712	0.706	0.991	0.990
SVM	0.155	0.265	0.722	0.819	0.975	0.992
MLP (8)	0.165	0.235	0.753	0.753	0.993	0.993
MLP (8,4)	0.160	0.173	0.726	0.725	0.995	0.995
MLP (12,4)	0.179	0.191	0.747	0.753	0.995	0.995
BDT	0.222	0.351	0.774	0.823	0.997	0.997

Tabla 2: Eficiencia de señal para varias eficiencias de fondo entrenando con todas las variables. En verde los valores más altos.

En la tabla 2 se presentan los valores de eficiencia de señal (la fracción de eventos bien clasificados) para algunos valores de la eficiencia de fondo (fracción de falsos positivos). Para cada uno se muestra la eficiencia en el conjunto de prueba y en el de entrenamiento. El valor relevante es el del conjunto de prueba, y si las dos eficiencias son muy distintas significa que hay sobreentrenamiento. Aparentemente el BDT es el clasificador que mejor funciona, seguido de las redes neuronales, y los métodos lineales y el de cortes funcionan peor. Hay algo de sobreentrenamiento en el SVM y el BDT.

Ahora vamos a probar a simplificar el problema eliminando algunas variables. Teóricamente, si hay más variables hay más información, y los métodos deberían descartar la que no es relevante, sin embargo, a veces añadir demasiadas variables complica el proceso de aprendizaje. Por ello, se han ordenado las variables por su poder de discriminación, medido por el propio TMVA, y el resultado se muestra en la tabla 3.

Variable	Poder de discriminación
mMET	0.5822
ptMET	0.1099
etaMET	0.0156
ptb1	0.0120
ptb2	0.0102
ptlep1	0.0090
ptlep2	0.0084
etab1	0.0067
etalep1	0.0061
etab2	0.0060
etalep2	0.0058
philep2	0.0046
phib1	0.0041
phib2	0.0040
phiMET	0.0033
philep1	0.0029

Tabla 3: Poder de discriminación de cada variable. Es una medida de cómo es de distinta la variable entre los eventos de señal y fondo.

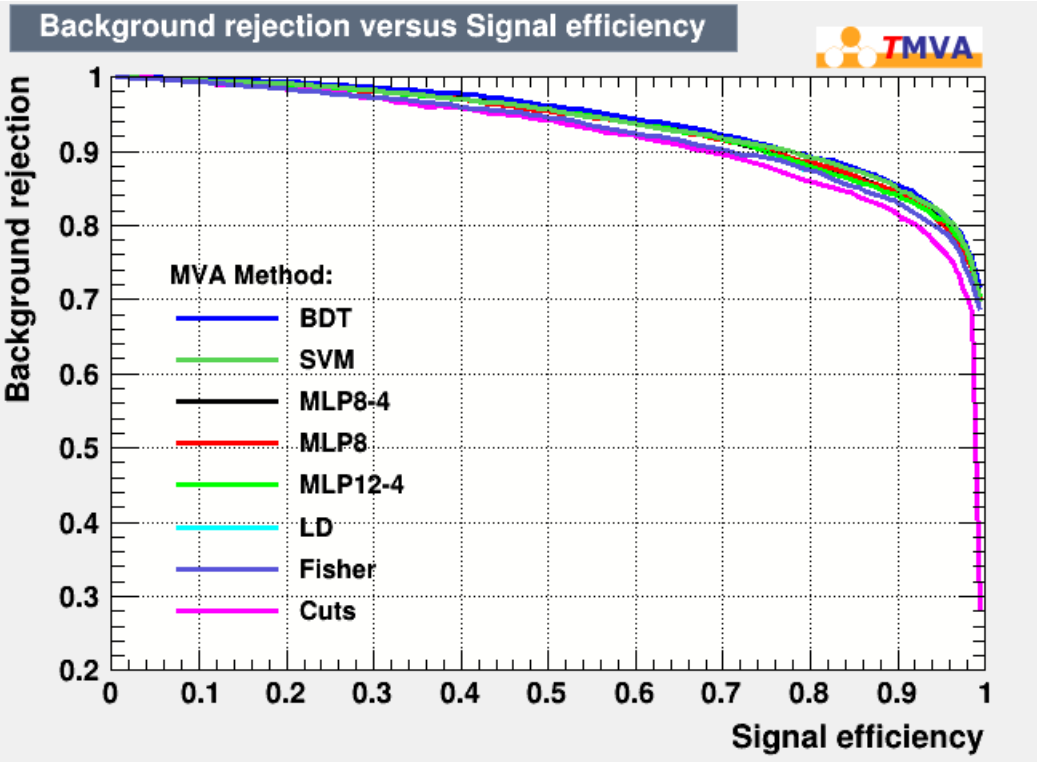


Figura 15: Curvas ROC de los clasificadores entrenados con 11 variables

Entonces, he eliminado las cinco últimas variables, que son ángulos phi. Los resultados se muestran en la figura 15 y la tabla 4. Se observa una mejora en las eficiencias, por ejemplo, ahora las redes neuronales llegan a 0.2.

Clasificador	Eficiencia a 1% de fondo (test set)	Eficiencia a 1% de fondo (train set)	Eficiencia a 10% de fondo (test set)	Eficiencia a 10% de fondo (train set)	Eficiencia a 30% de fondo (test set)	Eficiencia a 30% de fondo (train set)
Cortes	0.117	0.142	0.681	0.679	0.979	0.988
Lineal	0.138	0.147	0.707	0.703	0.991	0.990
Fisher	0.138	0.147	0.707	0.703	0.991	0.990
SVM	0.217	0.197	0.778	0.771	0.994	0.995
MLP (8)	0.193	0.218	0.757	0.754	0.995	0.995
MLP (8,4)	0.225	0.230	0.756	0.746	0.996	0.995
MLP (12,4)	0.198	0.209	0.750	0.754	0.996	0.996
BDT	0.249	0.335	0.777	0.825	0.999	0.999

Tabla 4: Eficiencia de señal para varias eficiencias de fondo entrenando con 11 variables

Todavía podría quitar más variables, dejado solo 7, pero si lo hago la eficiencia empieza a bajar. Estos resultados se muestran en la figura 16 y la tabla 5.

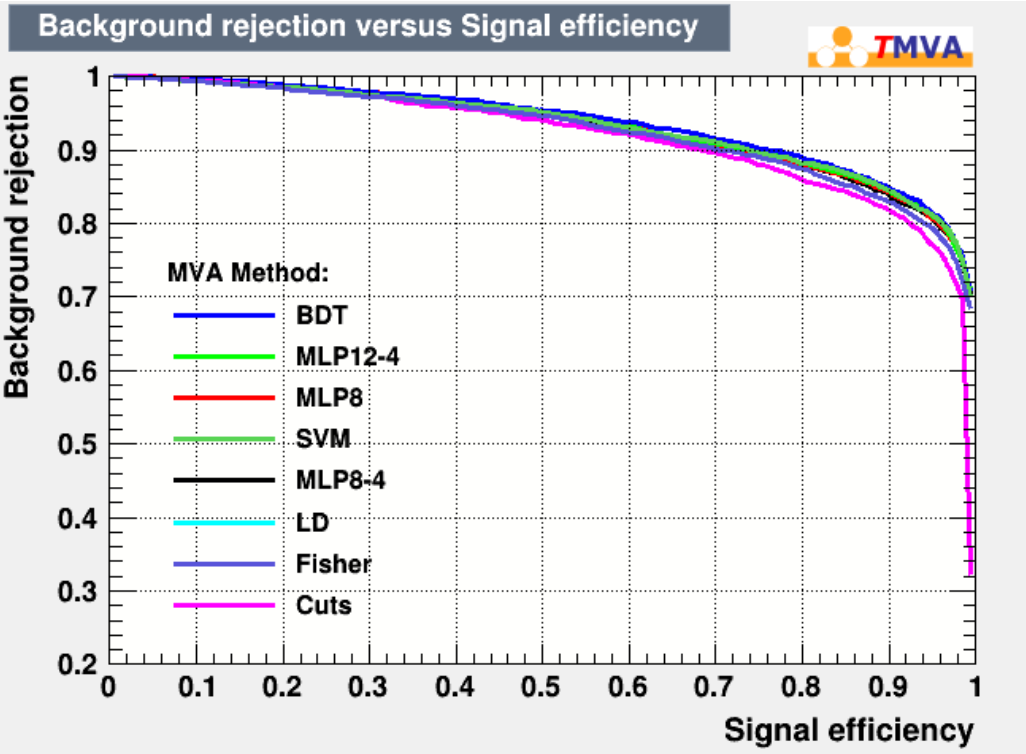


Figura 16: Curvas ROC de los clasificadores entrenados con 7 variables

Clasificador	Eficiencia a 1% de fondo (test set)	Eficiencia a 1% de fondo (train set)	Eficiencia a 10% de fondo (test set)	Eficiencia a 10% de fondo (train set)	Eficiencia a 30% de fondo (test set)	Eficiencia a 30% de fondo (train set)
Cortes	0.151	0.154	0.681	0.668	0.984	0.989
Lineal	0.137	0.149	0.702	0.702	0.990	0.990
Fisher	0.137	0.149	0.702	0.702	0.990	0.990
SVM	0.140	0.151	0.737	0.737	0.995	0.994
MLP (8)	0.165	0.207	0.734	0.733	0.997	0.996
MLP (8,4)	0.178	0.190	0.730	0.717	0.995	0.995
MLP (12,4)	0.177	0.187	0.738	0.735	0.998	0.997
BDT	0.184	0.282	0.763	0.800	0.996	0.997

Tabla 5: Eficiencia de señal para varias eficiencias de fondo entrenando con 7 variables

Lo que mejor funciona, de momento, es coger 11 variables.

3.4 Uso de variables derivadas

Otro aspecto que se puede tratar aquí es la inclusión de variables derivadas: combinar algunas de las variables usadas en otras, usando argumentos físicos o simplemente geométricos, de forma que le demos a los métodos parte del trabajo hecho. Idealmente, si el algoritmo es óptimo, debería ser capaz de deducir esa relación, pero a veces incorporar conocimiento previo sobre las variables puede simplificar el proceso de aprendizaje.

En la lista de variables mencionada más arriba aparecen las variables phi, que representan el ángulo polar de los momentos. Una sola de esas variables no aporta ninguna información relevante, ya que el sistema tiene simetría cilíndrica, pero dos variables phi correspondientes a dos partículas que provienen de la desintegración de otra nos da información sobre el ángulo en el que han salido y con ello la masa de la partícula de la que proceden, de forma que se puede usar la diferencia entre ellas para determinar esa masa.

Una forma de usar esa diferencia es añadirla a las 11 variables que ya se tienen, pero eso ocasionaría un problema: cuando uno de los ángulos cruza el 0, pasa de 2π a 0, de modo que la diferencia tiene ahí un salto de 2π. Además, en principio una diferencia entre dos ángulos y la diferencia opuesta son equivalentes. Esto empuja a pensar que, en vez de añadir directamente la diferencia como una variable, sería buena idea añadir su coseno: es una función continua en la circunferencia y simétrica respecto del 0.

Por otra parte, aquí tenemos el ángulo phi de cuatro partículas: dos quarks bottom y dos leptones. Sabemos que un quark y un leptón han salido de la misma partícula (un quark top), y el otro quark y el otro leptón de otro quark top, pero no sabemos qué quark va con qué leptón, así que tenemos cuatro cosenos de diferencias de ángulos phi. Añadiendo estas cuatro variables al conjunto de 11, y repitiendo el entrenamiento con el nuevo conjunto, resultan las eficiencias que se muestran en la tabla 6. Han mejorado un poco, aunque no mucho.

Clasificador	Eficiencia a 1% de fondo (test set)	Eficiencia a 1% de fondo (train set)	Eficiencia a 10% de fondo (test set)	Eficiencia a 10% de fondo (train set)	Eficiencia a 30% de fondo (test set)	Eficiencia a 30% de fondo (train set)
Cortes	0.141	0.144	0.654	0.654	0.980	0.988
Lineal	0.137	0.141	0.708	0.712	0.990	0.990
Fisher	0.137	0.141	0.708	0.712	0.990	0.990
SVM	0.169	0.215	0.729	0.801	0.980	0.990
MLP (8)	0.207	0.246	0.751	0.756	0.994	0.995
MLP (8,4)	0.205	0.219	0.747	0.731	0.994	0.994
MLP (12,4)	0.182	0.259	0.749	0.757	0.992	0.992
BDT	0.248	0.402	0.786	0.837	0.996	0.996

Tabla 6: Eficiencia de señal al entrenar con 11 variables originales y 4 variables derivadas, que son cosenos de diferencias de ángulos

La siguiente prueba es tomar como indicador de la masa $p_1 p_2 \sin\left(\frac{\phi_1-\phi_2}{2}\right)^2 = p_1 p_2 (1-\cos(\phi_1-\phi_2))$, que es más próximo a la masa de la partícula de la que vienen. Con esta “masa”, el resultado se muestra en la tabla 7. Una conclusión interesante es que ahora el rendimiento de las MLP tiende al del BDT y en algún caso lo supera.

Clasificador	Eficiencia a 1% de fondo (test set)	Eficiencia a 1% de fondo (train set)	Eficiencia a 10% de fondo (test set)	Eficiencia a 10% de fondo (train set)	Eficiencia a 30% de fondo (test set)	Eficiencia a 30% de fondo (train set)
Cortes	0.111	0.152	0.682	0.676	0.978	0.987
Lineal	0.148	0.143	0.746	0.738	0.988	0.987
Fisher	0.148	0.143	0.746	0.738	0.988	0.987
SVM	0.205	0.229	0.784	0.805	0.993	0.996
MLP (8)	0.235	0.222	0.756	0.764	0.994	0.993
MLP (8,4)	0.201	0.221	0.744	0.753	0.995	0.995
MLP (12,4)	0.267	0.261	0.764	0.774	0.996	0.996
BDT	0.251	0.350	0.790	0.835	0.998	0.998

Tabla 7: Eficiencia de señal al entrenar con 11 variables originales y 4 variables derivadas, que están relacionadas con la masa

3.5 Resultados de la clasificación para masa 10

Ahora se ha intentado el mismo procedimiento, variación del número de variables y cálculo de variables derivadas, para las muestras simuladas donde la masa de la partícula de materia oscura es 10. En este caso la diferencia entre señal y fondo es mucho menor, así que las eficiencias serán más bajas. La evolución de los resultados con el número de variables es similar a la de masa 100 aunque más sutil, en concreto añadir variables derivadas también mejora los resultados. Las eficiencias, para 11 variables y las derivadas, se muestran en la tabla 8 y la curva ROC en la figura 17.

Clasificador	Eficiencia a 1% de fondo (test set)	Eficiencia a 1% de fondo (train set)	Eficiencia a 10% de fondo (test set)	Eficiencia a 10% de fondo (train set)	Eficiencia a 30% de fondo (test set)	Eficiencia a 30% de fondo (train set)
Cortes	0.020	0.031	0.192	0.203	0.506	0.524
Lineal	0.022	0.022	0.200	0.190	0.541	0.536
Fisher	0.022	0.022	0.200	0.190	0.541	0.536
SVM	0.039	0.049	0.221	0.253	0.536	0.579
MLP (8)	0.028	0.033	0.206	0.217	0.539	0.543
MLP (8,4)	0.021	0.027	0.194	0.204	0.532	0.528
MLP (12,4)	0.029	0.028	0.205	0.211	0.538	0.546
BDT	0.030	0.068	0.213	0.297	0.528	0.602

Tabla 8: Eficiencia de señal al entrenar con 11 variables y las masas derivadas, para masa 10

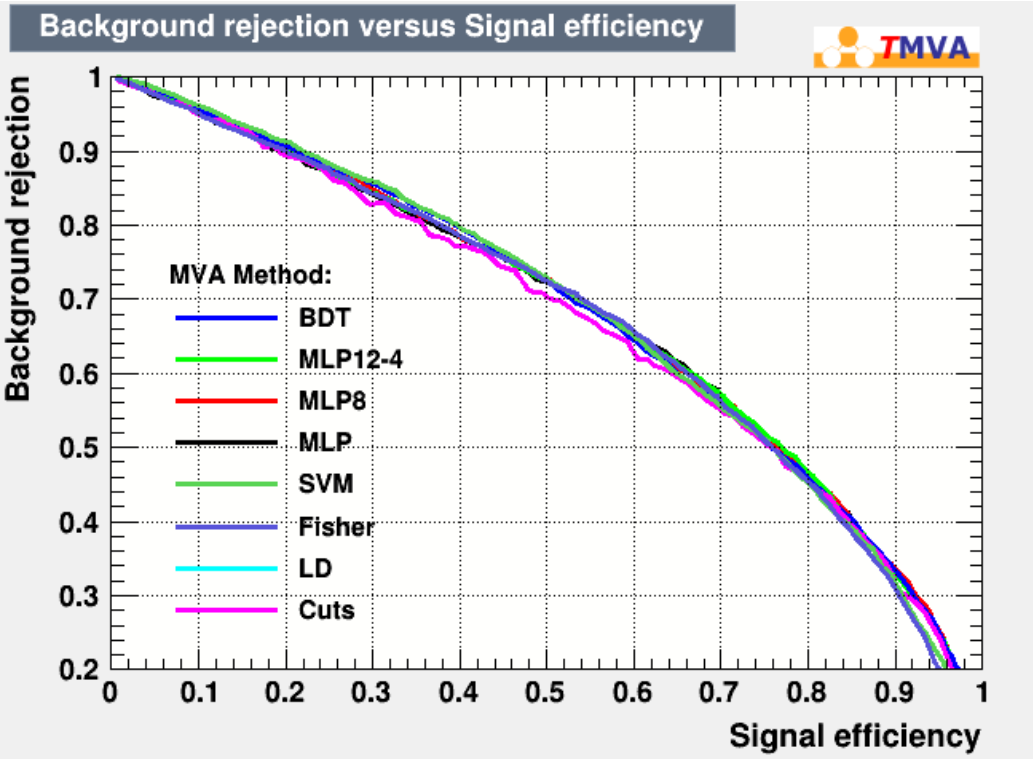


Figura 17: Curva ROC de los clasificadores para los datos con masa 10, entrenados con 11 variables y las masas calculadas

Como cabía esperar, las eficiencias son mucho peores, sin embargo, las diferencias entre los distintos clasificadores siguen siendo parecidas a las de masa 100.

3.6 Conclusión

Es posible distinguir, usando métodos de aprendizaje automático, entre el suceso “ttbar” y el mismo suceso con producción de materia oscura, por lo menos en el caso de alta masa. Los distintos métodos no dan el mismo rendimiento: los métodos no lineales funcionan mejor, y dentro de esos el BDT es el que mejor funciona, seguido por las redes neuronales. También hemos visto que elegir bien el número de variables es importante: en principio más variables deberían funcionar mejor, pero si los métodos tienen variables superfluas no funcionan tan bien. Además, tiene un efecto positivo añadir variables derivadas, relacionadas con la física del problema: con ellas las redes neuronales llegan a igualar al BDT.

En resumen, merece la pena probar distintos métodos y distintas configuraciones para ver cuál resuelve mejor el problema. Hay que tener en cuenta que doblar la eficiencia implica reducir a la mitad el tiempo de toma de datos necesario para descubrir la partícula, e incluso puede ser la diferencia entre descubrirla o no.

4. Regresión

4.1 Introducción

A continuación, voy a intentar entrenar los métodos para que determinen, a partir de los valores que se pueden medir, la masa de la partícula de materia oscura que se está produciendo en la reacción. Esto es lo que se conoce como regresión, y es novedoso en este contexto. Se sabe que la MET está correlacionada con esta masa, pero no da información suficiente para una estimación. Queremos averiguar si los métodos, en concreto, las redes neuronales pueden determinar esa masa a partir de magnitudes medibles.

Realizar una regresión es en cierta medida similar a una clasificación: hay que entrenar uno o varios métodos y luego probarlos. A la hora de entrenarlos, en lugar de estar clasificados como “señal” o “fondo”, llevan asociada una variable de salida, que en este caso es la masa de la partícula. El objetivo es ajustar los parámetros para minimizar, ya no los eventos mal clasificados, sino la diferencia entre el valor predicho y el correcto.

Para ello, en TMVA muchos métodos usados para clasificación pueden usarse también para regresión, pero no todos los métodos son idóneos: por ejemplo, los árboles de decisión, que funcionan mediante bifurcaciones, pueden hacer bien una clasificación, pero no tan bien la regresión, para la que necesitarían calcular un valor. Por ello, en este trabajo vamos a usar solo las redes neuronales.

4.2 Procedimiento

Para ajustar los hiperparámetros se ha procedido de forma parecida a la clasificación: se han entrenado tres redes neuronales con distinto número de neuronas, y se ha variado el número de iteraciones en función de la curva de aprendizaje. También se ha cambiado el número de variables de entrada, para encontrar el que daba mejor resultado. En este caso, el número óptimo de variables era 7 junto con las variables derivadas (ver sección 3.4) y el número óptimo de iteraciones era 300 para todas las redes. Las tres redes (8 neuronas, 8+4 neuronas y 12+4 neuronas) han dado un comportamiento muy parecido, así que solo se muestra una de ellas (en concreto la segunda, pero daría lo mismo).

Aparte de los hiperparámetros, también se puede elegir con qué conjunto de datos se entrenan las redes. Una opción es usar todos los valores posibles de la masa, de forma que la red tenga más información a lo largo de todo el rango. Otra opción es usar solo los valores extremos, 10 y 500, para entrenar, y probar en las masas intermedias para comprobar si las redes tienen capacidad de interpolar. Esta opción no produjo una interpolación muy satisfactoria, de modo que probamos a añadir también un valor intermedio al entrenamiento.

4.3 Resultados

Primero se han entrenado y probado las redes con todas las masas. En la Figura 18 se muestra el resultado de la regresión. En el eje X está la variable de salida de la red, que es la masa que buscamos dividida por 500.

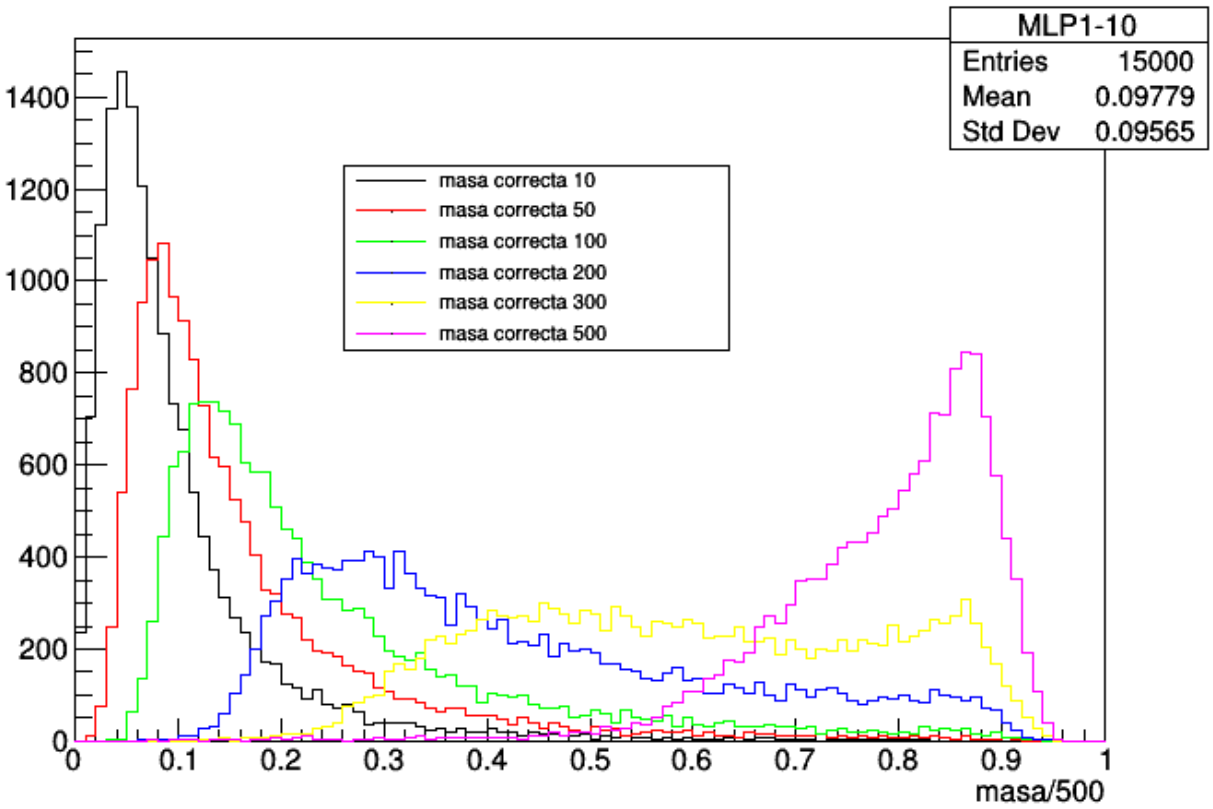


Figura 18: Histograma de la salida de la red de regresión entrenada con todas las masas, para cada valor de la masa

Se observa que, aunque la distribución no es perfecta, su pico sigue más o menos el valor correcto; sin embargo, en algunas masas hay colas bastante largas. El pico de masa 10 está desplazado a la derecha, y el de masa 500 está desplazado a la izquierda. Para masa 300 la curva es demasiado plana. En la Tabla 9 he hecho un análisis más cuantitativo. En resumen, los resultados, sin ser perfectos, son bastante buenos.

Masa	Pico	Media	Desviación típica	Error
10	25	49	49	62
50	40-45	81	66	73
100	65-70	122	83	86
200	150-160	213	99	100
300	230-235 y 435	296	91	91
500	435-440	393	55	121

Tabla 9: resultados para el pico del histograma (masa más probable, indicada por el número de la clase en el histograma), su media, desviación típica y error cuadrático medio (definido como $E=\sqrt{(\mu-x_0)^2+\sigma^2}$ *donde* x_0 *es el valor correcto) para cada masa*

Ahora se ha entrenado solo con las masas de los extremos, 10 y 500, y se ha probado con todas. El resultado se muestra en la Figura 19 y en la Tabla 10.

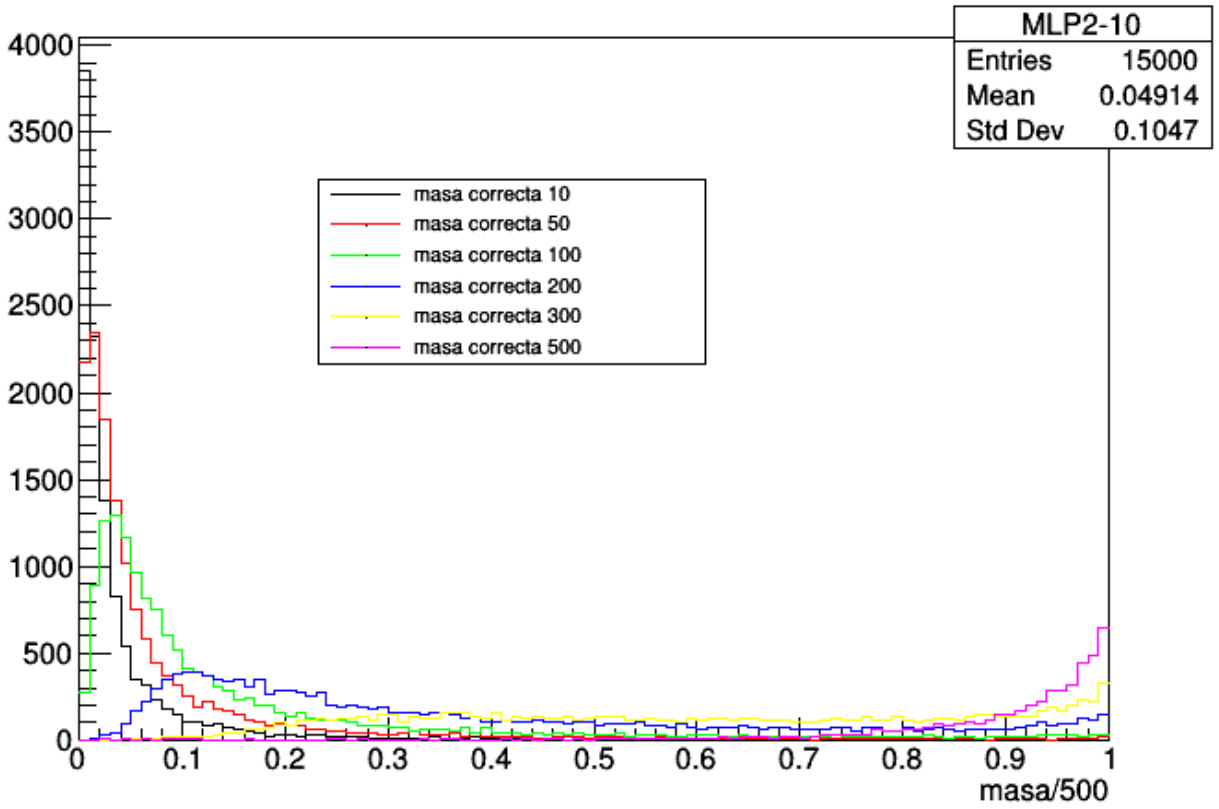


Figura 19: Histograma de la salida de la red de regresión entrenada con masas 10 y 500, para cada valor de la masa

Las curvas para las que se ha entrenado la red son las que están en negro y rosa. Se observa que en este caso el pico no se mueve con tanta facilidad, de hecho los picos quedan en valores altos o bajos: la red se está limitando a un problema de clasificación entre las dos masas proporcionadas. Sin embargo, la media está cerca del valor correcto.

Masa	Pico	Media	Desviación típica	Error
10	5	25	54	56
50	5-10	44	76	76
100	20	81	102	104
200	55-75	186	137	137
300	500	299	128	128
500	500	441	84	103

Tabla 10: valores del pico, la media, la desviación típica y el error cuadrático medio de las estimaciones de masa de la red entrenada en dos puntos

Finalmente, se ha entrenado con las masas 10, 200 y 500, para ver si se mejora el resultado de la interpolación. Los resultados están en la figura 20 y la tabla 11.

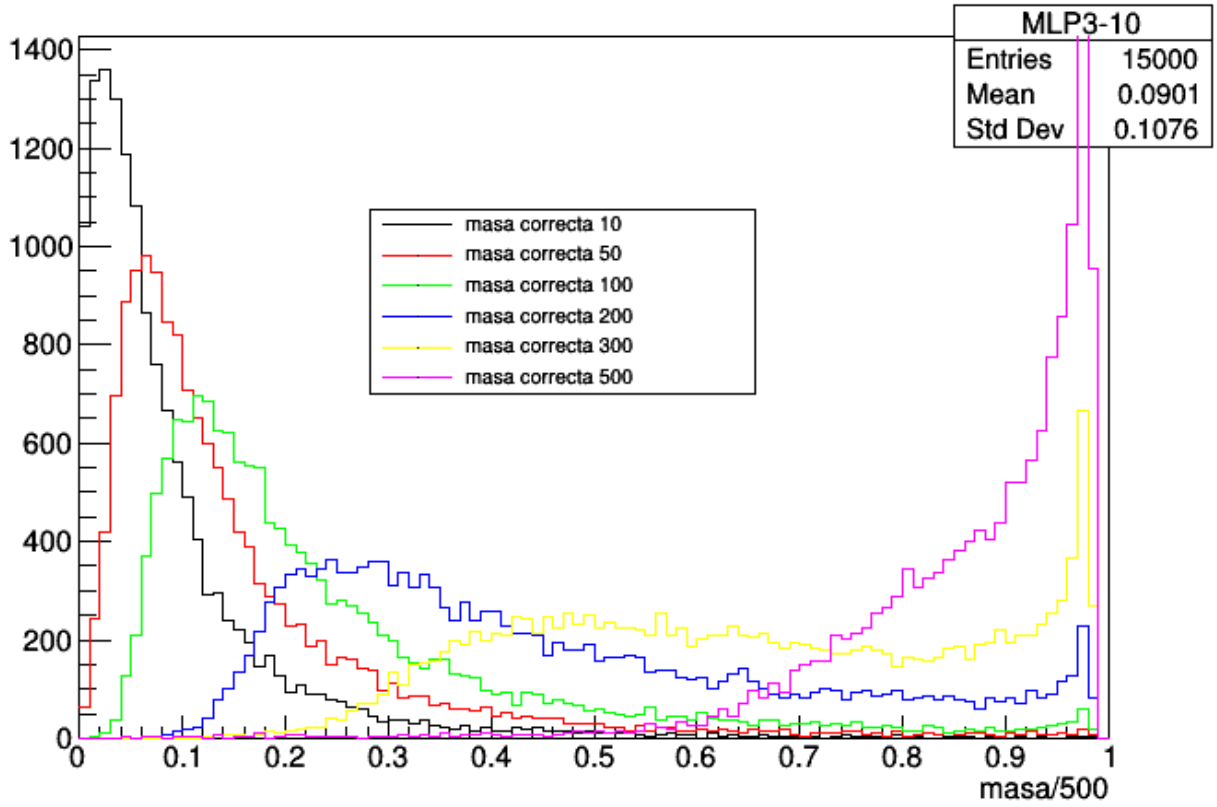


Figura 20: Histograma de la salida de la red de regresión entrenada con masas 10, 200 y 500, para cada valor de la masa

Masa	Pico	Media	Desviación típica	Error
10	10-15	45	55	65
50	30-35	78	75	80
100	60-65	124	95	98
200	120-150	227	116	119
300	485-490	328	107	110
500	490	436	61	88

Tabla 11: valores del pico, la media, la desviación típica y el error cuadrático medio de las estimaciones de masa de la red entrenada en tres puntos

Aquí, las curvas para las que se ha entrenado están en negro, azul y rosa, y se observa que las curvas para valores intermedios sí están interpolando correctamente, aunque no tan bien como cuando se entrenaba con ellos (de hecho, en algunas masas se observa un indeseable pico en 500).

4.4 Conclusión

Hemos observado que el uso de redes neuronales para regresión puede dar una buena estimación de los valores a calcular, aunque con limitaciones: si los valores están muy separados, la red tiende a clasificar entre ellos en vez de calcular valores intermedios. Su poder de interpolación es limitado y no es capaz de extrapolar. Por ello, es mejor dar a la red todos los valores que se tienen. Si se entrena la red con el suficiente número de puntos, el valor que se obtiene es adecuado.

5. Influencia de errores de medida

5.1. Introducción

Hasta ahora hemos entrenado y probado con los datos como si fueran ideales, usando valores generados y tomando los valores medidos como si fueran exactamente esos. En realidad, obviamente, eso no es así: los valores no se miden con precisión infinita. En este capítulo buscamos una aproximación más realista: ver cómo responden los clasificadores cuando los datos tienen errores, incluso introduciendo errores sistemáticos.

Para ello, a partir de las muestras generadas idealmente, se han generado otras introduciendo un ruido aleatorio o un sesgo.

5.2. Efectos de precisión realista

En primer lugar, voy a contar los resultados que se han obtenido simulando una precisión limitada de los datos. Para ello, he generado muestras aleatorias a partir de las ideales, añadiéndoles un error relativo (que he variado desde 5% hasta 100%) en las variables ptb1 y ptb2 (momentos de los quarks) y en el MET, y un error menor en el momento de los leptones. La idea es ver cómo responden los clasificadores a un error en los datos estando entrenados idealmente, por lo tanto, los he entrenado con las muestras ideales y probado sobre las muestras realistas.

La figura 21 muestra la eficiencia de señal a 1% de eficiencia de fondo en función del error.

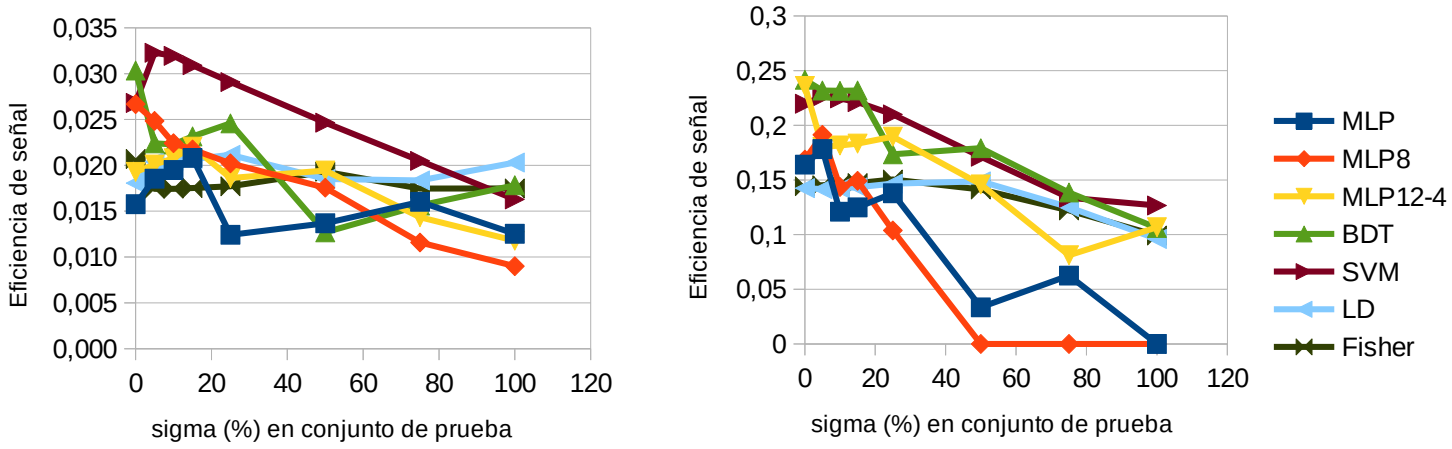


Figura 21: Eficiencia de señal correspondiente a una eficiencia de fondo de 1% en función del error introducido, a la izquierda para masa 10 y a la derecha para masa 100

Como es esperable, los resultados se degradan: para masa 10, la eficiencia de señal para una eficiencia de fondo de 0.01 pasa de sus valores normales de 0.02-0.03 a valores de 0.01-0.02, y para masa 100 baja de 0.15-0.25 a 0-0.1. En ambos casos los clasificadores más sensibles son las redes neuronales, seguidas por el SVM y el BDT, y los métodos lineales los menos sensibles. De hecho, para masa 100, algunas redes neuronales llegan a dejar de clasificar completamente. (De todos modos, el descenso es lento, y la desviación tiene que llegar al menos a 25% para que se empiece a notar la pérdida.)

Probamos ahora a entrenar con muestras también distorsionadas, suponiendo así que la magnitud de esta distorsión es conocida. Voy a ir, igual que antes, cambiando la magnitud de esta distorsión, para ver cómo afecta a los distintos métodos. En la Figura 22 se muestra la eficiencia de señal en función de la distorsión introducida.

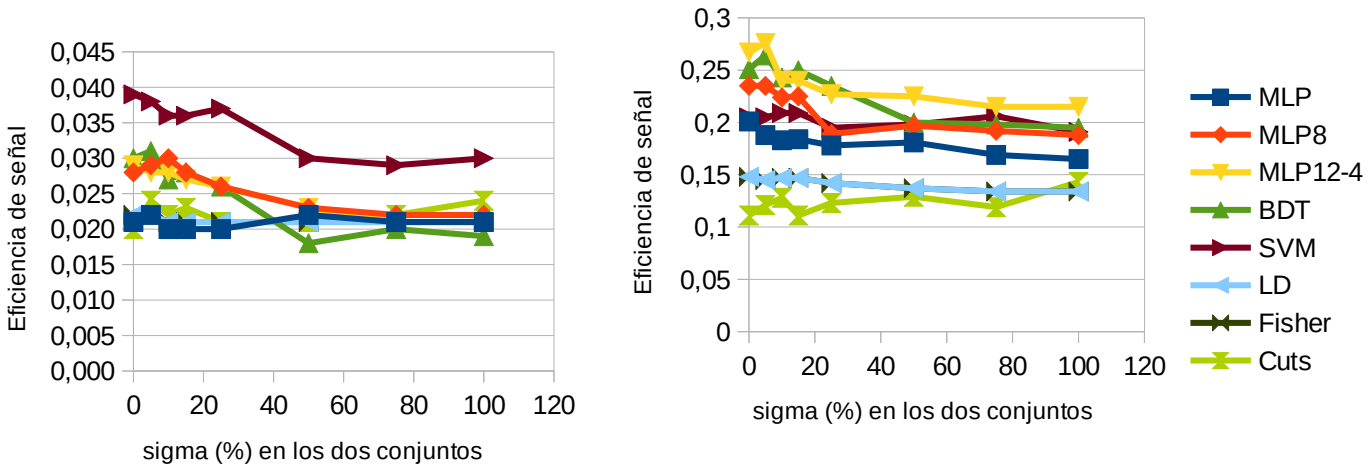


Figura 22: Eficiencia de señal correspondiente a una eficiencia de fondo de 1% en función del error introducido, a la izquierda para masa 10 y a la derecha para masa 100

Con este método, se consigue evitar la degradación de los resultados: aun con errores muy grandes, solo bajan de 0.03-0.04 a 0.02-0.03 para masa 10 y no se aprecia pérdida para masa 100.

5.3. Efectos sistemáticos

A continuación vamos a estudiar los efectos de errores sistemáticos en las mediciones, por ejemplo, qué pasaría si los detectores estuvieran mal calibrados. Para ello vamos a introducir un sesgo en la muestra, es decir, simular que los sensores producen, de media, un valor distinto al correcto. Se ha probado el conjunto de clasificadores, entrenados sobre una muestra ideal, sobre un conjunto con un sesgo variable: desde 0% hasta 100%, manteniendo una desviación fija de 10%. En la Figura 23 se representa la eficiencia de señal frente al sesgo para masa 100.

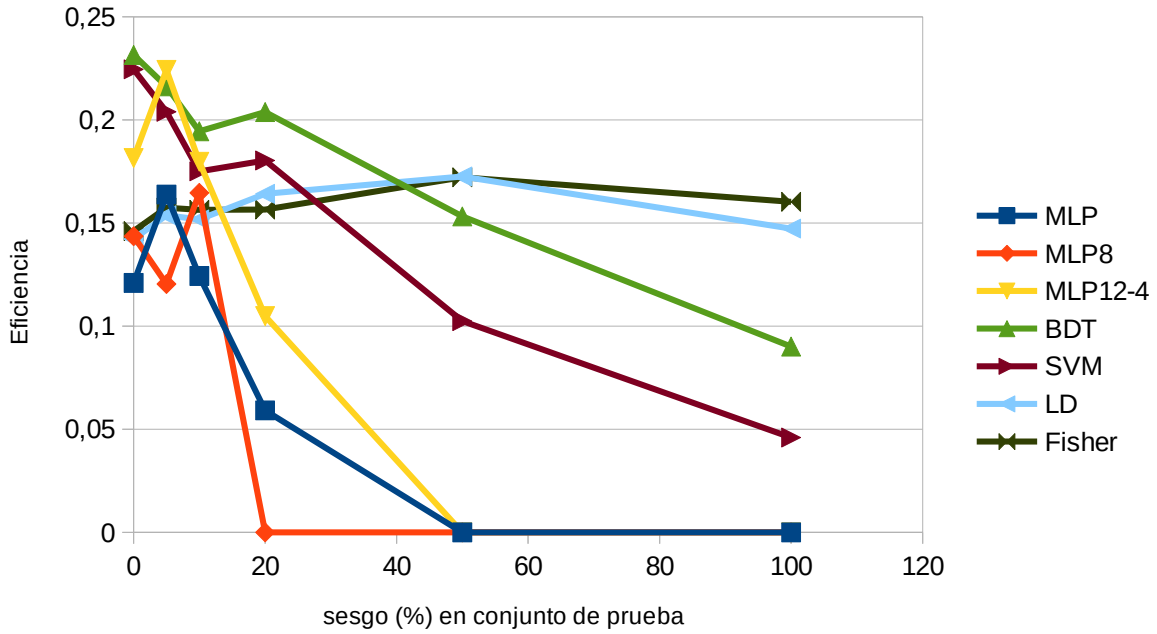


Figura 23: Eficiencia de señal correspondiente a 1% de eficiencia de fondo en función del sesgo

El resultado es que las tres redes neuronales caen a 0 para sesgos altos, el BDT y SVM caen moderadamente y los métodos lineales no caen. (Sin embargo, aunque no se representa en esta gráfica, hemos comprobado que para una eficiencia de fondo de 10% o 30%, los métodos lineales sí que caen).

Para evitar esta degradación, introducimos una forma de *data augmentation*: entrenamos los clasificadores sobre un conjunto en el que hemos introducido un ruido aleatorio, para que aprendan que puede haber un error y así no confíen tanto en las variables, de forma que, al probar luego sobre un conjunto con sesgo funcionen mejor.

Para ello, en el conjunto de prueba se ha mantenido un sesgo de 10%, sin desviación, mientras el de entrenamiento tenía una desviación variable, desde 0 hasta 50%. En principio sería razonable que la desviación óptima fuera igual al sesgo: vamos a comprobarlo. En la Figura 24 se representa eficiencia frente a desviación.

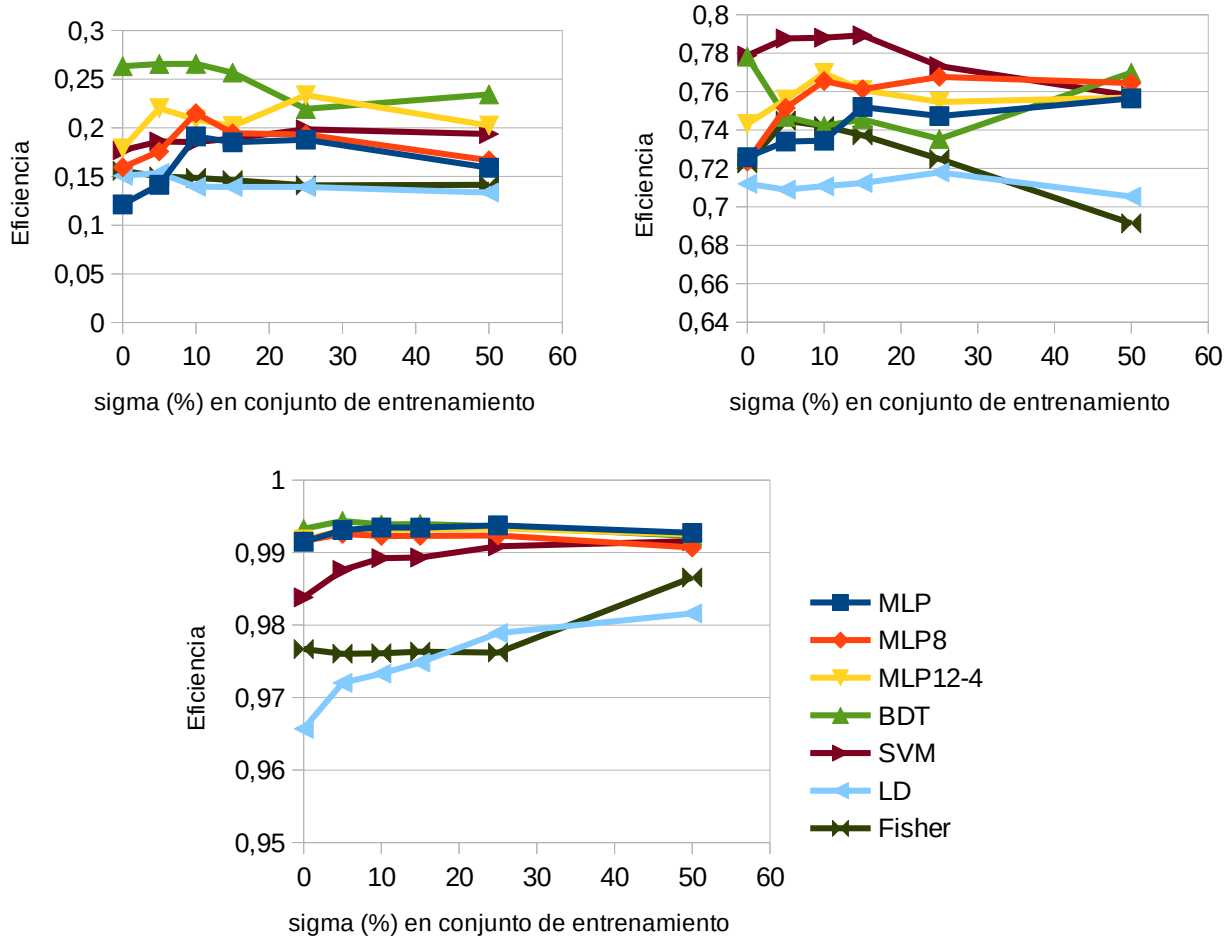


Figura 24: Eficiencia de señal para una eficiencia de fondo de 1% (arriba a la izquierda), 10% (arriba a la derecha) y 30% (abajo), en función de la desviación en el conjunto de entrenamiento

Como vemos, las redes neuronales, que habían degradado, recuperan su eficiencia si se entrenan con un ruido entre 10 y 30%, mientras que otros métodos tienen un comportamiento distinto y de hecho el BDT pierde rendimiento.

Con una eficiencia de fondo del 30%, la eficiencia de señal mejora durante todo el rango, independientemente del clasificador.

Ahora he repetido el mismo estudio para una muestra con un sesgo de 20% (figura 25) y de 50% (figura 26).

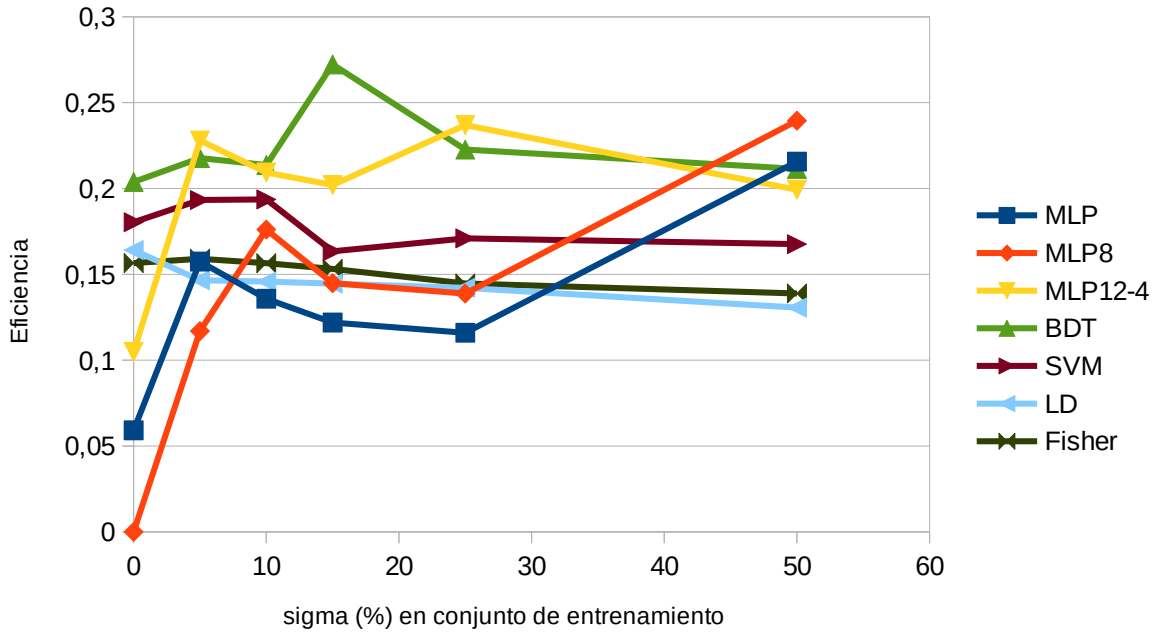


Figura 25: Eficiencia de señal para una eficiencia de fondo de 1% en función de la desviación en el conjunto de entrenamiento, con un sesgo de 20% en el conjunto de prueba

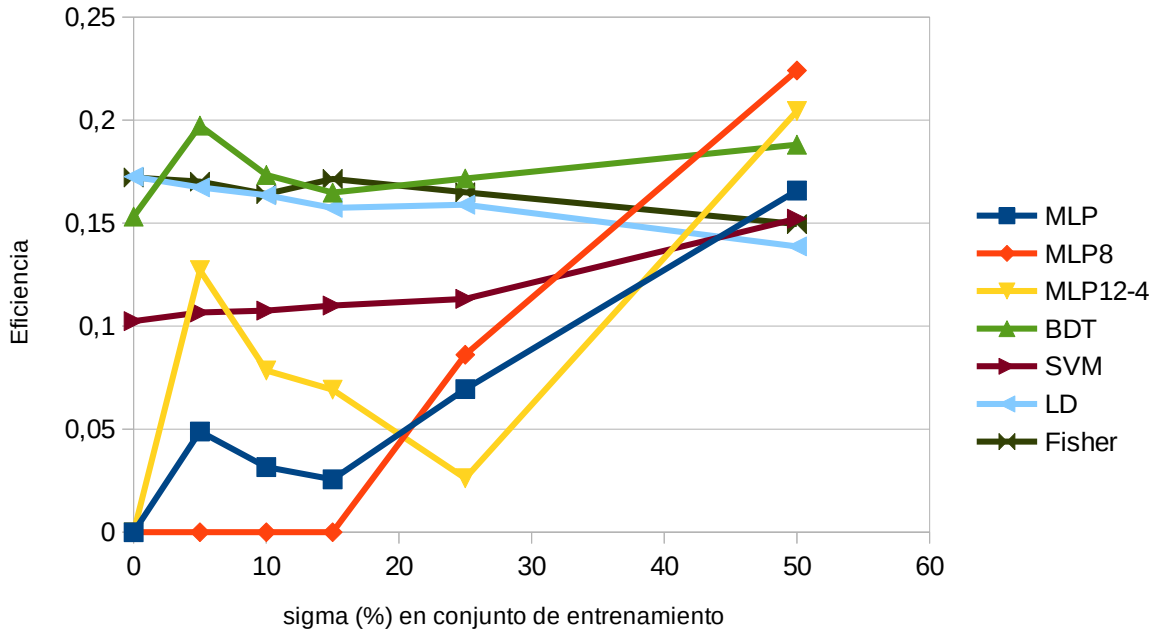


Figura 26: Eficiencia de señal para una eficiencia de fondo de 1% en función de la desviación en el conjunto de entrenamiento, con un sesgo de 50% en el conjunto de prueba

Aquí, para muchos de los métodos, se ve muy clara la mejora, y en general vemos que añadir un ruido del orden del sesgo introducido recupera el rendimiento. Sin embargo, para otros métodos, el comportamiento es diferente, el ruido no les afecta y algunos hasta degradan un poco.

5.4 Conclusión

Con estos experimentos hemos comprobado que es importante tener en cuenta la precisión de las medidas en el entrenamiento, porque si no se degradan los resultados. De hecho, pueden ser peores

de lo esperado, lo que introduciría más errores, por ejemplo, en medidas de secciones eficaces. De todas formas, como hemos visto, esto depende mucho del método.

También hemos comprobado el efecto de los errores sistemáticos, introduciendo un sesgo en las medidas (error parecido a los errores de calibración) y viendo que algunos métodos son muy sensibles a estos errores. Finalmente, hemos encontrado un método, basado en *data augmentation*, que corrige gran parte del problema.

6. Conclusiones

En este trabajo se ha estudiado el manejo de herramientas de aprendizaje automático. Se ha ido más allá del simple uso de estas herramientas, analizando aspectos como su optimización o el efecto de los errores sistemáticos. Se han utilizado dos paquetes: RSNNS, dentro del programa R, y TMVA, dentro de ROOT.

Inicialmente, en RSNNS se han estudiado problemas sencillos de redes neuronales, viendo el funcionamiento y las características y limitaciones de estos clasificadores:

- Las redes neuronales pueden clasificar correctamente conjuntos de datos donde las clases están intercaladas o colocadas de alguna forma que hace difícil su separación.
- Se ha comprobado que al introducir un error en los datos, el resultado se deteriora, y se propone un método de *data augmentation* que permite cancelar ese deterioro: al añadir datos adicionales distorsionados, la red no confía tanto en los puntos que tienen errores grandes y extrae la información de otra forma.
- También se pueden usar para regresión. Una red neuronal da el resultado correcto para regresión no lineal, lo que quiere decir que puede aprender relaciones no lineales entre los datos. Sin embargo se observa muy mala capacidad de extrapolación.

A continuación se pasó a aplicar estas y otras técnicas de aprendizaje automático a un problema real de clasificación de sucesos en física de partículas, usando ahora la herramienta TMVA. Se trata de la búsqueda de partículas de materia oscura producidas en el LHC, donde buscamos separar esta señal de un fondo de colisiones ttbar varios órdenes de magnitud mayor y con una señal experimental muy similar.

Se han probado los siguientes métodos: MLP (redes neuronales) con tres configuraciones distintas, BDT (árboles de decisión), SVM, lineal, Fisher y cortes rectangulares.

Algunas de las principales conclusiones obtenidas son

- Los métodos de clasificación, especialmente los no lineales, pueden conseguir la diferenciación entre sucesos de física de partículas que aparentemente no tienen diferencias medibles, como los esperados en producción de materia oscura. El clasificador que mejor funciona es en principio el BDT, pero las redes neuronales pueden igualarlo e incluso superarlo. Conviene probar distintas configuraciones de las redes, con distinto número de variables de entrada y de neuronas en capas ocultas.
- Se ha comprobado que en contra de lo que se pensaría por una interpretación literal de la teoría, poner más variables de entrada no siempre produce un mejor rendimiento de los métodos: a veces algunas variables de poco poder de separación hacen difícil el aprendizaje.
- Sin embargo, también se ha comprobado que añadir variables de entrada calculadas a partir de las otras (incluso de las que no se están usando) relacionadas con la física del problema, por ejemplo masas invariantes, puede mejorar la clasificación.

Se ha propuesto una forma novedosa de estimar cantidades físicas, en este caso la masa de una partícula de materia oscura que escapa del detector, a partir de cantidades medidas relacionadas

indirectamente con ellas, por medio de regresión con redes neuronales. Se ha comprobado que sí son capaces de dar un resultado orientativo, aunque no muy preciso. También se ha estudiado su capacidad de interpolar: el resultado es que para mejorar la interpolación hay que entrenar la red con valores más cercanos a los que se quieren interpolar.

Se ha estudiado el efecto en los distintos algoritmos de la precisión limitada en las variables. El efecto, como se esperaba, es el de disminuir el rendimiento de la clasificación, especialmente si el entrenamiento se hace con una muestra ideal. Esto no es sorprendente. Se observa que el grado de degradación depende mucho del método (incluso de versiones del mismo método) y del punto escogido en la curva ROC. Por lo tanto, es más recomendable hacer un estudio para cada caso que intentar sacar conclusiones generales.

Se ha estudiado también el efecto de posibles errores sistemáticos, interpretados como un sesgo en los datos. El rendimiento se ve muy perjudicado, especialmente en las redes neuronales. Se ha propuesto un nuevo método basado en *data augmentation*, consistente en incluir datos degradados en el entrenamiento. Se ha estudiado el rendimiento en función de la magnitud de esta degradación, comprobando que el efecto es máximo si es del mismo orden que el error sistemático: en esos casos el deterioro se corrige casi por completo.

Recapitulando todo el trabajo, hemos comprobado que el uso de *machine learning* puede ayudar en la física experimental de partículas, aunque es necesario optimizar los métodos para que salgan mejores resultados. Hemos estudiado también la influencia de los errores, incluyendo también errores sistemáticos, y hemos comprobado que limitan el rendimiento de los métodos. Hemos propuesto una técnica que mitiga el deterioro debido a los errores sistemáticos. Finalmente hemos propuesto una técnica de regresión basada en redes neuronales, que puede ser prometedora para aplicar a algunas situaciones no resueltas hasta ahora.

Referencias

[1] The CMS Collaboration et al, *The CMS experiment at the CERN LHC.* JINST 3 (2008) S08004,742 doi:10.1088/1748-0221/3/08/S08004

[2] The CMS Collaboration, *Search for dark matter produced in association with heavy-flavor quark pairs in proton-proton collisions at* $\sqrt{s}$ *= 13TeV.* Eur. Phys. J. C 77 (2017) 845

[3] J. García Ferrero, J. Piedra Gómez, P. Martínez Ruiz del Árbol, *Búsqueda de materia oscura en asociación con pares de quark top en el experimento CMS*

[4] G. Bertone, D. Hooper, J. Silk, "Particle dark matter: evidence, candidates and constraints", Phys. Rept. 405 (2005) 279, doi:10.1016/j.physrep.2004.08.031, arXiv:hep-ph/0404175.

[5] Stuttgart Neural Network Simulator, desarrollado en la universidad de Stuttgart, mantenido por Tübingen, http://www.ra.cs.uni-tuebingen.de/SNNS/welcome.html

[6] A. Hoecker et al, Guía de usuario de TMVA, arXiv:physics/0703039

[7] R.A. Fisher, Annals Eugenics 7, 179 (1936).

[8] V. Vapnik, A. Lerner, "Pattern recognition using generalized portrait method", Automation and Remote Control, 24, 774 (1963).

[9] B.E. Boser, I.M. Guyon, V.N. Vapnik, "A training algorithm for optimal margin classifiers", D. Haussler, ed., Proceedings of the 5th Annual ACM Workshop on Computational Leraning Theory, 144, ACM Press (1992).

[10] C.G. Broyden, "The Convergence of a Class of Double-rank Minimization Algorithms", J. Inst. of Math. and App. 6, 76 (1970); R. Fletcher, "A New Approach to Variable Metric Algorithms", Computer J. 13, 317 (1970); D. Goldfarb, "A Family of Variable Metric Updates Derived by Variational Means", Math. Comp. 24, 23 (1970); D.F. Shannon, "Conditioning of Quasi-Newton Methods for Function Minimization", Math. Comp. 24, 647 (1970).

[11] Y. Freund, R.E. Schapire, J. of Computer and System Science 55, 119 (1997).