TRABAJO FIN DE GRADO

Doble Grado Ingeniería Informática y Matemáticas

Detección de anomalías basada en técnicas de ensembles

Biblioteca de algoritmos

Autor
Ignacio Aguilera Martos

Directores
Francisco Herrera Triguero
Jacinto Carrasco Castillo

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Facultad de Ciencias

—

Granada, mes de 201

Detección de anomalías basada en técnicas de ensembles

Biblioteca de algoritmos

Autor
Ignacio Aguilera Martos

Directores
Francisco Herrera Triguero
Jacinto Carrasco Castillo

Detección de anomalías basada en técnicas de ensembles: Biblioteca de algoritmos

Ignacio Aguilera Martos

Palabras clave: outlier, anomalía, ensemble, python

Resumen

Poner aquí el resumen.

Outlier detection based in ensemble methods: Library implementation

Ignacio Aguilera Martos

Keywords: outlier, ensemble, anomaly, python

Abstract

Write here the abstract in English.

Yo, **Ignacio Aguilera Martos**, alumno de la titulación Doble Grado en Ingeniería Informática y Matemáticas de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada** y la **Facultad de Ciencias**, con DNI 77448262-V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Ignacio Aguilera Martos

Granada a X de mes de 201 .

D. **Francisco Herrera Triguero**, Profesor del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **Jacinto Carrasco Castillo**, Doctorando del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Detección de anomalías basada en técnicas de ensembles, Biblioteca de Algoritmos***, ha sido realizado bajo su supervisión por **Ignacio Aguilera Martos**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Francisco Herrera Triguero **Jacinto Carrasco Castillo**

Agradecimientos

Poner aquí agradecimientos...

Índice general

Capítulo 1

Introducción

Antes de comenzar el objeto de estudio de este trabajo, lo primero que debemos hacer es contextualizar el mismo y establecer un marco de trabajo en cuanto a teoría que se empleará en la posterior experimentación y desarrollo del mismo.

El estudio realizado y plasmado en este trabajo se centra en la obtención de técnicas para la detección de anomalías en conjuntos de datos, concepto que introduciremos posteriormente. En concreto las técnicas que se van a desarrollar son las conocidas como técnicas de ensemble que se basan en el estudio del problema o bien combinando modelos existentes o bien haciendo un estudio pormenorizado aplicando algún criterio por subespacios del conjunto de datos.

En primer lugar el trabajo desarrollará una introducción a la teoría de aprendizaje y resolución de problemas mediante datos y no por diseño así como la teoría matemática que esto involucra. Esta primera sección nos dará suficiente estructura al trabajo para poder definir en términos de distancias lo que significa que una instancia de un conjunto de datos sea una anomalía.

Posteriormente se desarrollará brevemente algunos conocimientos estadísticos básicos de estadística multivariante para poder introducir el concepto de anomalía desde la perspectiva de las probabilidades condicionadas.

Tras esto podremos entrar en el terreno de la experimentación, desarrollo y explicación de técnicas y puesta en contraste con los algoritmos clásicos para comprobar el desempeño de las nuevas técnicas.

Por último se presentarán las conclusiones obtenidas tras todo este estudio.

1.1. Contextualización

En este contexto y sin haber comenzado la discusión del problema podríamos considerar el establecimiento de una solución al problema de detección de anomalías empleando un algoritmo preciso y óptimo, aunque costoso en tiempo, pero esta solución no es viable en este contexto. Pensemos por un momento en cómo detectar una anomalía en un conjunto de datos. Para poder dar un algoritmo concreto que resuelva siempre el problema de forma óptima tenemos que estar seguros de que somos capaces de definir de forma clara y para todo conjunto de datos cuándo estamos en presencia de un dato anómalo. Pero, ¿sabemos exactamente cuándo sobrepasamos la barrera de dato ligeramente desviado, ruido o una anomalía?

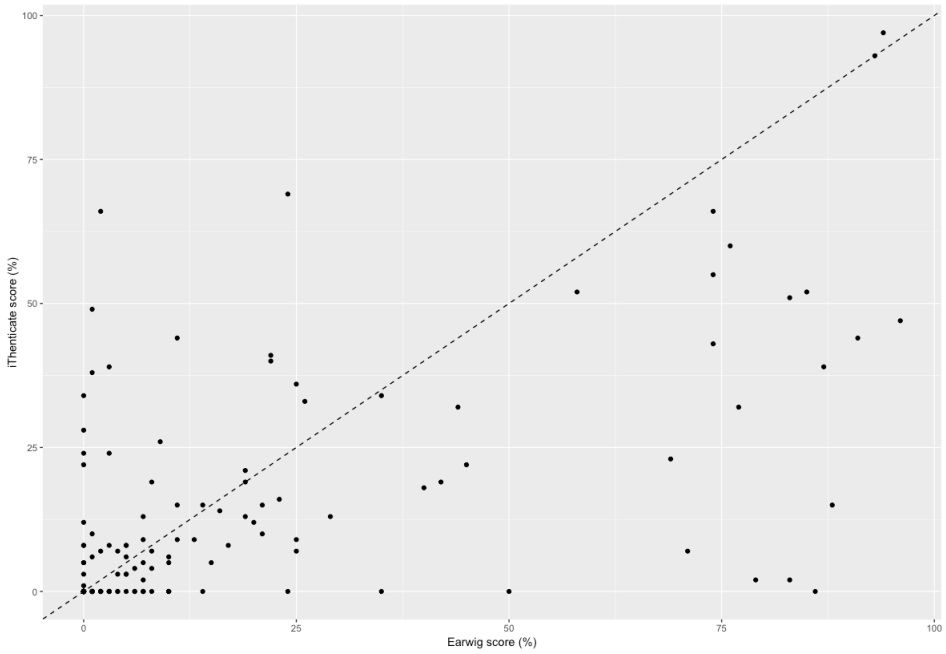

Figura 1.1: Ejemplo de conjunto de datos (Wikimedia)

Pensemos por ejemplo en este conjunto de datos. Tenemos un cluster muy bien definido en la esquina inferior izquierda y datos dispersos. ¿Podemos afirmar claramente cual es la línea que distingue los datos anómalos de los normales?

Esta discusión nos lleva a la primera barrera que tenemos que superar en este trabajo. El problema no es resoluble por diseño y por tanto debemos realizar una aproximación a través de los datos. Esto nos va a llevar a aprender de los mismos, valorar cada dato con una puntuación que nos acabará discriminando los datos normales y anómalos.

En el contexto de los problemas resueltos a través de los datos tenemos dos tipos: clasificación y regresión.

Podríamos pensar que estamos ante un problema de clasificación ordinario, es decir, aprender a través de los datos de entrenamiento cuáles son las anomalías y cuales los datos normales pero no es así. Pensemos que no sabemos ni siquiera nosotros clasificar claramente estas anomalías salvo en algunos ejemplos prácticos. Por ejemplo si le diéramos a una persona un conjunto de datos ya establecido como el que hemos visto anteriormente 1.1 no sería nada fácil e incluso no podemos afirmar que sepamos hacer esta tarea. Por contra si el conjunto de datos es generado en un proceso de trabajo, por ejemplo un conductor de camiones haciendo su ruta, podemos saber cuándo hay una anomalía porque el trabajador la está viviendo y puede señalarla.

Por tanto este problema no es de clasificación al uso pues no es un problema supervisado. Esto significa que, en general no vamos a tener un conjunto etiquetado en el que poder probar cómo aprende nuestro modelo y tendremos que abordar otras técnicas para comprobar el desempeño.

Ahora que comprendemos el alcance del problema podemos intentar profundizar un poco más en la base teórica del mismo analizando las herramientas que el machine learning nos provee para abordarlo.

Parte I

Machine Learning y el concepto de Anomalía

En esta sección vamos a centrarnos en dos aspectos: el Machine Learning para establecer las herramientas usadas en el estudio y el propio concepto de anomalía y algunas reflexiones acerca del mismo.

Capítulo 2

Machine Learning

En este capítulo vamos a hacer un repaso sobre los conceptos asociados al Machine Learning, el aprendizaje y la teoría matemática que involucra. Estas herramientas y conceptos los utilizaremos posteriormente para resolver el problema de detección de anomalías.

2.1. Contextualización del aprendizaje

Para comenzar tenemos que empezar definiendo en que consiste el proceso de aprender sobre unos datos. Supongamos que tenemos un problema en el que tenemos una entrada y una salida, por ejemplo una entrada válida podría ser un vector $x \in \mathbb{R}^d$ y una salida un valor real o un número natural. El problema de aprendizaje intenta estimar una estructura de tipo entrada-salida como la descrita usando únicamente un número finito de observaciones.

Podemos definirlo de forma más general empleando tres conceptos:

- Generador: El generador se encarga de obtener las entradas $x \in \mathbb{R}^d$ mediante una distribución de probabilidad $p(x)$ desconocida y fijada de antemano.
- Sistema: El sistema es el que produce la salida "y" (correcta) para cada entrada $x \in \mathbb{R}^d$ mediante la distribución de probabilidad $p(x|y)$ desconocida y fijada de antemano.
- Máquina de aprendizaje: esta es la que va a obtener información de las entradas y salidas conocidas para intentar predecir la salida co-

rrecta para una entrada nueva que se nos de. De forma abstracta esta máquina lo que hace es tomar una serie de funciones de un conjunto general de forma que para una entrada dada x la función $f(x, \omega)$ con $\omega \in \Omega$ nos de la salida que corresponde para x donde ω es una forma de indexar las funciones tomadas para generalizar la salida del conjunto más general de funciones que hemos indicado.

El único cabo que hemos dejado sin atar en las definiciones que acabamos de ver es el conjunto de funciones del cual tomaremos algunas para adaptar la máquina de aprendizaje a los datos recibidos. Este conjunto de funciones, que notaremos como $\mathcal{H}$, es de momento la única forma que tenemos de aplicar un conocimiento a priori en la máquina de aprendizaje.

Para finalizar esta breve introducción y poder continuar profundizando vamos a exponer algunos ejemplos de clases de funciones para que podamos visualizar el contexto.

- Funciones lineales: En este caso la clase de funciones $\mathcal{H}$ está formada por funciones de la forma $h(x) = w_0 + \sum_{i=1}^{d} x_i w_i$ donde $w \in \mathbb{R}^{d+1}$. Este es el modelo de funciones más clásico.
- Funciones trigonométricas: Un ejemplo de una clase de funciones trigonométricas podría ser $f_m(x, v_m, w_m) = \sum_{j=1}^{m-1}(v_j \sin(jx) + w_j \cos(jx)) + w_0$ donde en este caso la entrada es un único valor real. Este tipo de clases de funciones serán útiles en problemas de regresión que luego explicaremos con algo más de detalle aunque sin centrarnos mucho en ello pues no es el objetivo del estudio.

2.1.1. Objetivo del aprendizaje

Cuando hablamos de aprendizaje queremos obtener algo de dicho aprendizaje a partir de los datos. Como ya se ha mencionado, intentamos obtener una función de una familia de funciones que aproxime o modele de buena manera la salida del sistema. Por tanto, ese es nuestro objetivo: obtener una función de la familia de funciones que minimice el error.

El problema que enfrentamos es que sólo disponemos de un número finito, por ejemplo n, de observaciones de datos y su correspondiente salida. Esto nos va a hacer que no podamos tener una garantía de optimalidad a no ser que tendamos n a infinito.

Sin embargo si que podemos cuantificar cómo de buena es una aproximación con respecto a otra mediante la función pérdida o error que denotaremos

como $L(y, f(x, \omega))$. Esta función nos va a medir la diferencia entre la salida real del sistema y la salida dada por la función f para la entrada x siendo siempre $L(y, f(x, \omega)) \geq 0$.

Recordemos además que el Generador obtiene datos mediante una distribución desconocida pero fijada de antemano y que son independientes e idénticamente distribuidos con respecto a la distribución conjunta, es decir:

$$p(x, y) = p(x)p(y|x)$$

Una vez definido todo esto podemos obtener el valor esperado de pérdida o error mediante el funcional

$$R(\omega) = \int L(y, f(x, \omega))p(x, y)dxdy$$

Ahora podemos concretar un poco más lo que entendemos como objetivo del aprendizaje. El objetivo será encontrar una función $f \in \mathcal{H}$ que nos minimice el valor del funcional $R(\omega)$. Pero recordemos que $p(x, y)$ es desconocida para nosotros, por lo que no podemos saber cómo se distribuyen los datos y por tanto el valor del funcional no es calculable para nosotros y por tanto la solución puramente de cálculo no es accesible.

Por tanto, la única forma realmente potente y útil de encontrar una buena aproximación será incorporar el conocimiento a priori que tenemos del sistema. En la sección anterior hemos visto que una forma de incorporar dicho conocimiento es mediante la selección de la clase de funciones, pero además será muy relevante el hecho de cómo los datos son empleados en el proceso de aprendizaje. En este apartado de decisión tendremos que resolver primero la codificación de los datos, el algoritmo empleado y el uso de técnicas como la regularización que veremos después para incorporar nuestro conocimiento en el camino que nos lleve a la solución.

2.1.2. Clases de aprendizaje

El problema de aprendizaje puede ser subdividido a su vez en cuatro clases distintas y que se suelen abordar de forma independiente. Estoss tipos de problemas de aprendizaje son:

- Clasificación: El problema de clasificación consiste en identificar y separar instancias de datos según su clase. Por ejemplo podemos dividir a

la población mundial en dos clases: sanos y enfermos. Un problema de clasificación podría ser saber identificar estas clases para un conjunto de personas. Los problemas de clasificación más sencillos son aquellos en los que se usan dos únicas clases aunque se puede generalizar la definición del problema a k-clases.

- Regresión: El problema de regresión consiste en estimar una función $f : \mathbb{R}^n \to \mathbb{R}$ a partir de una serie de muestras previas con los valores de f. Un problema de regresión podría ser determinar la función que, dados los datos de altura y dimensiones corporales sea capaz de darnos el peso aproximado de la persona.

- Estimación de la función de densidad: en este caso no nos interesa la salida que proporciona el sistema, ya sea el valor de una clase o una función real como en el caso de la regresión. En este caso el objetivo del aprendizaje es conseguir la función de densidad $f(x, \omega)$, con $\omega \in \Omega$ los parámetros necesarios de la función de densidad, con la que se distribuyen los datos de entrada del sistema.

- Agrupamiento y cuantificación vectorial: El problema de cuantificación vectorial consiste en intentar explicar la distribución de los vectores de entrada mediante puntos clave llamados centroides. De esta forma se podría reducir la complejidad de los datos expresándolos en función de un sistema de generadores menor. El problema de agrupamiento tiene también relación por utilizar la idea de centroide, pero el objetivo es completamente distinto. El objetivo del problema de agrupamiento es intentar conseguir agrupar los datos en clusters, es decir, regiones del espacio en las que se concentran un conjunto de datos. De esta forma intentamos agrupar los datos que mantienen una relación entre sí. Un ejemplo de un problema de cuantificación vectorial podría ser un problema de reducción de dimensionalidad y un ejemplo de problema de agrupamiento podría ser identificar instancias de datos con características comunes.

2.2. Principios y adaptación del aprendizaje

Según Vapnik [1] la predicción mediante el aprendizaje se puede dividir en dos fases:

1. Aprendizaje o estimación a partir de una muestra.

2. Predicción a partir de las estimaciones obtenidas.

Estas dos fases se corresponden con los dos tipos de inferencia clásica que conocemos, esto es, inducción y deducción. Traído a este caso el proceso de inducción es aquel que a partir de los datos de aprendizaje o los datos de la muestra que tenemos con la salida que corresponde podemos estimar un modelo. Es decir, estamos sacando el conocimiento de los datos para generar el modelo. El proceso de deducción es aquel que, una vez obtenido el modelo estimado (la generalización) obtenemos una predicción de la salida sobre un conjunto de datos.

Por contra, Vapnik propone un paso que resuelve estas dos fases directamente y que él denomina transducción. Este paso consiste en, dados los datos de entrenamiento obtenemos directamente los valores de salida sin tener que hacer la generalización a un modelo. De esta forma, según Vapnik, podríamos reducir el error que cometemos en la predicción. Este razonamiento tiene sentido, pues estamos omitiendo el paso más complejo del proceso de inducción-deducción.

En resumen esta idea se puede resumir en la siguiente figura:

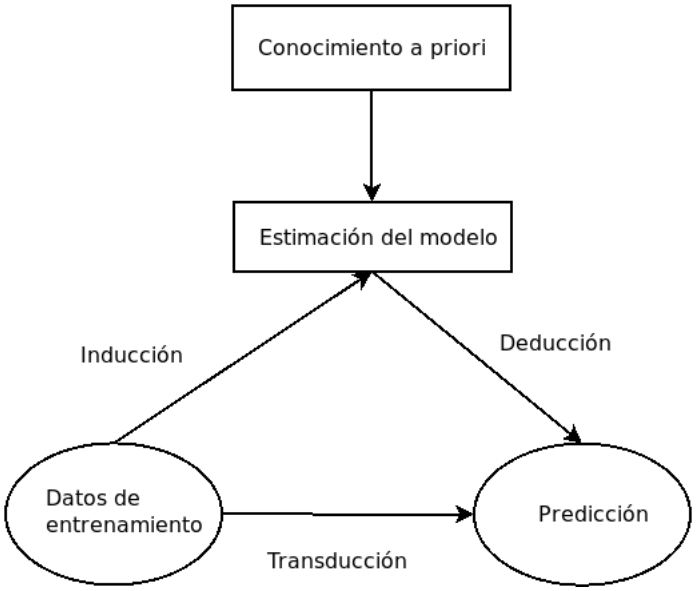

Figura 2.1: Tipos de inferencia y transduccion [2, p. 41]

Podemos ver que el conocimiento a priori que tenemos del problema se manifiesta una vez se crea el modelo general, de forma que se emplearía en el paso de la inducción. Ya hemos hablado previamente del conocimiento a priori y cómo incorporarlo al modelo, pero por concretar un poco más podemos añadirlo básicamente de dos formas:

- Escogiendo un conjunto de funciones para aproximar la salida del sistema
- Añadiendo restricciones o penalizaciones adicionales a dicho conjunto de funciones.

En resumen, para poder crear la generalización del modelo de forma única necesitamos:

1. Un conjunto de funciones para aproximar la salida.
2. Conocimiento a priori.
3. Un principio inductivo, que no es más que una indicación de cómo emplear los datos para llegar a la generalización del modelo.
4. Un método de aprendizaje, es decir, una implementación del principio inductivo.

En secciones posteriores revisaremos algunos de los principios inductivos más usados pero es importante reseñar la diferencia entre pricipio inductivo y método de aprendizaje. Para un mismo principio inductivo podemos tener varios métodos de aprendizaje, pues podemos escoger diferentes formas de llevarlo a la práctica. Por ejemplo, uno de los principios inductivos más empleados es el ERM o Empirical Risk Minimization, es decir, minimización del error empírico. Podríamos pensar en diferentes formas de utilizar este principio, por ejemplo sólo avanzamos en la creación del modelo si a cada paso que demos minimizamos el error, o por ejemplo vamos avanzando varios modelos a la vez hasta obtener un número de modelos finales de entre los cuales escogeremos aquel que mejor minimice dicho error.

2.2.1. Principios inductivos

Una vez introducido el concepto como hemos hecho en la sección anterior vamos a hacer un breve repaso de los principios más usados y en qué consiste cada uno de ellos.

Penalización o Regularización

Imaginemos que tenemos una clase de funciones muy flexible, esto es con un gran número de parámetros libres $f(x, \omega)$ con $\omega \in \Omega$. Vamos a partir de

la base del ERM, es decir, minimizar el error empírico. La penalización lo que va a hacer es añadir un factor a la función a minimizar:

$$R_{pen}(\omega) = R_{emp}(\omega) + \lambda\phi[f(x,\omega)]$$

Donde $R_{emp}(\omega)$ es el error empírico con los parámetros ω y $\phi[f(x,\omega)]$ es un funcional no negativo asociado a cada estimación $f(x,\omega)$. El parámetro $\lambda > 0$ es un escalar que controla el peso de la penalización.

El funcional $\phi[f(x,\omega)]$ puede medir lo que creamos conveniente que debemos añadir, es decir, aquí podemos añadir a la minimización algún tipo de medida que nos diga cómo de bien funciona el ajuste de los datos y cómo de bien funciona la información a priori que hemos incluido en el modelo. Pensemos por ejemplo que λ fuera un parámetro con un valor muy alto. En este caso la penalización por un mal ajuste de los datos no sería de gran importancia pues lo más conveniente sería minimizar el valor del funcional para no obtener una gran penalización. De esta forma podemos ajustar y dar un poco más de información al error empírico. Por ejemplo, en función del problema, es posible medir la complejidad de la solución mediante el funcional ϕ y de esta forma no sólo vamos a obtener una función que ajuste bien los datos, si no que también mantenga una cierta simplicidad para evitar por ejemplo el sobreajuste.

Reglas de parada anticipada

Pensemos en un método que vaya aprendiendo de los datos de forma iterativa intentando a cada iteración reducir el error cometido, por ejemplo el ERM. Los métodos o reglas de parada anticipada pueden verse como penalizaciones sobre el algoritmo conforme se va ejecutando. Las reglas de parada anticipada, como su nombre indica lo que preveen es la parada del algoritmo antes de obtener su objetivo teórico. Por ejemplo un algoritmo intenta que el error sea menor que 10^{-6} pero para reducirlo desde 10^{-4} hasta 10^{-5} está consumiendo millones de iteraciones. Si queremos que el tiempo de cómputo penalice lo que podemos hacer es fijar por ejemplo un número máximo de iteraciones que detenga el método aunque no se haya alcanzado esa barrera de error que se preveía.

Minimización del riesgo estructural o SRM

Para entender esta filosofía nos ponemos en la situación de que ya sabemos la clase de funciones con la que vamos a aproximar la salida del sistema,

por ejemplo hemos escogido la clase de funciones polinómicas. Bajo esta clase de funciones podemos ordenar las funciones por complejidad, entendiendo por complejidad el número de parámetros de la función. Por ejemplo los polinomios de grado m son de menor complejidad que los de grado $m+1$. De esta forma podemos pensar en una estructura de la clase de funciones de la forma:

$$S_0 \subset S_1 \subset S_2 \subset \cdots$$

Este parámetro de complejidad también puede ser un principio a minimizar para intentar conseguir una solución adecuada pero también simple. La generalización de la medida de complejidad para las clases de funciones es la conocida como dimensión VC o dimensión de Vapnik-Chervonenkis.

Inferencia Bayesiana

Este principio inductivo se utiliza en el problema de estimación de la función de densidad. El principio es utilizar la conocida fórmula de Bayes para hacer una estimación de la función de densidad empleando el conocimiento a priori que disponemos del problema. La forma en la que se emplea esta fórmula es de la siguiente:

$$P[modelo|datos] = \frac{P[datos|modelo] \cdot P[modelo]}{P[datos]}$$

donde $P[modelo]$ es la probabilidad a priori, $P[datos]$ es la probabilidad de los datos de entrenamiento y $P[datos|modelo]$ es la probabilidad de que los datos estén generados por el modelo.

Descripción de mínima longitud

La idea de este principio es la minimización de la longitud que se necesita emplear para describir un modelo y la correspondiente salida. Llamamos l a la longitud total:

$$l = L(modelo) + L(datos|modelo)$$

Esta medida puede ser vista como una medida de complejidad conjunta de todo el modelo.

2.3. Regularización

Por la importancia de este principio inductivo vamos a desarrollarlo un poco más, junto con el concepto de penalización, la selección de los modelos y la relación entre sesgo y varianza. Este último es un concepto muy relevante en cuanto al aprendizaje y que en nuestro caso, al no poseer la clasificación real tendremos que tenerlo en cuenta.

2.3.1. Problema de la alta dimensionalidad

Sabemos que cuando estamos ante un problema de aprendizaje nuestro objetivo es conseguir estimar una función con un número finito de instancias de una muestra ya con la salida. Al tener un número finito de elementos en la muestra ya sabemos que no podemos garantizar que la respuesta sea la óptima o correcta, pero además debemos pensar que a mayor regularidad del conjunto de funciones empleado debemos tener una densidad suficiente de puntos para compensar dicha regularidad. Este problema es conocido como la maldición de la dimensionalidad (curse of dimensionality). El problema es que cuanto mayor sea la dimensionalidad considerada más difícil es poder tener esa alta densidad de datos que se requieren para funciones muy regulares.

Este problema que conlleva la alta dimensionalidad proviene de la geometría de los espacio con alta dimensionalidad. A medida que incrementamos la dimensionalidad el espacio se ve cada vez con más aristas o picos. Podemos pensar en un cubo para el espacio tridimensional y a medida que aumentamos la dimensión incorporamos más aristas y vértices. Podemos resumir en 4 propiedades de los espacio con alta dimensionalidad que causan este problema:

1. La densidad disminuye exponencialmente al aumentar el número de dimensiones. Supongamos que tenemos una muestra de n puntos en $\mathbb{R}$. Para poder tener la misma densidad en un espacio d-dimensional $\mathbb{R}^d$ necesidamos n^d puntos.

2. Cuanto mayor dimensionalidad tenga el conjunto de datos mayor lado se necesita para que un hipercubo contenga el mismo porcentaje del conjunto que con una menor dimensionalidad. Imaginemos que tenemos un conjunto d-dimensional en el que tenemos la muestra dentro de un hipercubo unidad. Si quisiéramos abarcar un porcentaje $p \in [0, 1]$ necesitaríamos un cubo de lado $e_d(p) = p^{\frac{1}{d}}$. Como se puede observar a

mayor dimensionalidad y p constante el lado es cada vez mayor. Esta idea es fácilemente entendible si observamos la siguiente figura:

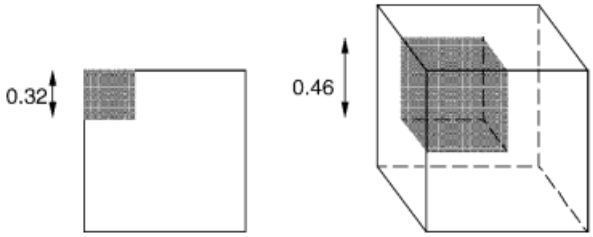

Figura 2.2: Para 2 dimensiones necesitamos menor lado que para 3 dimensiones. [2, p. 64]

3. Casi todo punto está más cerca de un borde que de otro punto. Pensemos en un conjunto de datos con n puntos distribuidos de forma uniforme en una bola d-dimensional de radio unidad. Para este conjunto de datos, según [3], la distancia media entre el centro de la distribución y los puntos más cercanos a dicho centro se mide bajo la fórmula:

$$D(d,n) = (1 - \frac{1}{2}^{1/n})^{1/d}$$

Si en esta fórmula tomamos por ejemplo $n = 200$ y $d = 10$ el resultado es $D(10, 200) \approx 0.57$. Esto significa que los puntos más cercanos al centro de la distribución están más cerca de los bordes que del centro.

4. Casi todo punto es una anomalía sobre su propia proyección. Si pensamos de nuevo en la idea de los vértices y aristas en espacio de alta dimensionalidad y pensamos en que, según el punto anterior, cada vez que aumenta la dimensionalidad los puntos están más cerca de los bordes entonces no es extraño pensar que los puntos a medida que aumenta la dimensionalidad están más distantes del resto de puntos. Esto intuitivamente (ya que aún no hemos visto la definición formal de anomalía) nos guía a pensar que vistos los puntos en sus propios entornos éstos serán anomalías comparados con el resto.

Figura 2.3: Forma conceptual de un espacio de alta dimensionalidad.[2, p. 64]

> Conceptualmente podemos imaginarlo con esta forma de picos, con lo que si tenemos los datos apiñados en dichos picos o extremos el resto de datos que estén en picos diferentes distan tanto del que estamos considerando que no podemos afirmar que tengan ninguna relación entre sí.

Estos puntos hemos de recordar que van referidos al conjunto de datos y no a las funciones que estamos considerando para representar la salida del sistema. Si estamos considerando la complejidad de las funciones la dimensionalidad no es una buena medida. Sabemos de la existencia de teoremas de aproximación de funciones como por ejemplo el Teorema de Superposición de Kolmogorov-Arnold.

Teorema 2.1 (Teorema de Superposición de Kolmogorov-Arnold) *Sea f una función continua de varias variables $f : X_1 \times ... \times X_n \to \mathbb{R}$, entonces existen funciones $\Phi_q : \mathbb{R} \to \mathbb{R}$ y $\phi_{q,p} : X_p \to [0, 1]$ tales que f se puede expresar como:*

$$f(x) = f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q(\sum_{p=1}^{n} \phi_{q,p}(x_p))$$

Este Teorema argumenta perfectamente que la complejidad que le damos a los datos por tener una alta dimensionalidad no es transferible a las funciones pues podemos expresar funciones de varias variables como combinación de funciones de una sola variables. En otras palabras no podemos argumentar que la complejidad de funciones univariantes sea mayor o menor que la de funciones multivariantes.

2.3.2. Aproximación de funciones

Como ya hemos dicho en la introducción queremos aproximar una función salida del sistema dentro de una familia de funciones. Este campo no es nuevo, tenemos como herramientas una serie de Teoremas relacionados con la aproximación de funciones como el Teorema de Kolmogorov enunciado anteriormente o el Teorema de aproximación de Weierstrass.

La versión más simple del Teorema de Weierstrass es la de funciones reales definidas en intervalos cerrados, veamos un repaso de estos Teoremas para hacer un esquema de la aproximación de funciones.

Teorema 2.2 (Teorema de aproximación de Weierstrass) *Supongamos que $f : [a, b] \to \mathbb{R}$ es una función continua. Entonces $\forall \epsilon > 0$, $\exists p$ un polinomio tal que $\forall x \in [a, b]$ tenemos que $|f(x) - p(x)| < \epsilon$.*

En otras palabras, podemos aproximar las funciones continuas reales definidas en un intervalo cerrado con el error que queramos en un punto mediante polinomios. Además tenemos versiones más generales aún como el Teorema de Stone-Weierstrass para funciones reales, para espacios localmente compactos y para el espacio de los complejos.

Estas aproximaciones son más sencillas en términos de la complejidad de la clase de funciones, pero tenemos aproximaciones muy famosas, como por ejemplo la serie de Fourier.

Definición 2.1 (Serie de Fourier) *Si tenemos una función $f : \mathbb{R} \to \mathbb{R}$ integrable en el intervalo $[t_0 - \frac{T}{2}, t_0 + \frac{T}{2}]$ entonces se puede obtener el desarrollo en serie de Fourier de f en dicho intervalo. Si f es periódica en toda la recta real la aproximación será válida en todos los valores en los que esté definida.*

$$f(t) \approx \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(\frac{2n\pi}{T} t) + b_n \sin(\frac{2n\pi}{T} t)]$$

Donde a_0, a_n y b_n son los coeficientes de la serie de Fourier que tienen la forma:

$$a_0 = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) dt$$

$$a_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)cos(\frac{2n\pi}{T}t)dt$$

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin(\frac{2n\pi}{T}t)dt$$

Como podemos ver hemos introducido dos conocidas formas de aproximar funciones, una con funciones polinómicas y otra con funciones trigonométricas. Vamos a dividir en dos los tipos de aproximación que podemos tener para el problema de aprendizaje.

1. Aproximaciones universales: son aquellas en las que se establece que cualquier función continua puede ser aproximada por otra función de otra clase con el error que queramos. En este grupo podríamos meter a los dos teoremas que hemos dado previamente. Dentro de este grupo podemos tener diferentes tipos de aproximaciones en función de la familia de funciones que escojamos como aproximaciones. Por ejemplo en los dos teoremas previos hemos cogido las clases de funciones polinómicas y trigonométricas pero podríamos haber tomado otras clases diferentes.

2. Aproximaciones inexactas: son aquellas en las que no podemos tener una aproximación como las que hemos dado en los teoremas previos, si no que proveen de una aproximación de peor calidad.

2.3.3. Penalización o control de la complejidad

Ya hemos discutido brevemente en la sección de principios inductivos la complejidad y cómo penalizarla. Vamos a ver qué elementos queremos controlar con la penalización:

1. La clase de funciones con la que vamos a hacer la aproximación. Tenemos que decidir si escoger una clase tan amplia que nos aseguremos que abarque la solución seguro pero penalicemos la complejidad de la elección o queremos una clase de funciones más ajustada.

2. Tipo de funcional de penalización. Tenemos que escoger entre los distintos tipos de penalización que queremos. Esto se reduce a escoger entre dos tipos de penalización: paramétrica y no paramétrica. La primera de ellas se basa en estudiar la suavidad del ajuste junto con el

número de parámetros que requiere la aproximación mientras que la segunda intenta estudiar lo mismo, es decir la suavidad del ajuste, sin medir los parámetros de la clase de funciones. En este punto se puede incorporar el conocimiento a priori del problema.

3. Método con el que queremos minimizar la penalización. Este apartado está relacionado con los métodos que tenemos de aprender de los datos y el objetivo será intentar hallar una forma eficiente de minimizar tanto el error de la aproximación como la propia penalización.

4. Control de la complejidad. Como hemos dicho antes el control de la complejidad no es algo sencillo y habrá que escoger la mejor manera de medir dicha complejidad. En secciones posteriores veremos medidas de complejidad como la dimensión de Vapnik-Chervonenkis.

Veamos brevemente la distinción que hemos hecho entre la penalización paramétrica y no paramétrica.

Penalización paramétrica

Supongamos que tenemos un conjunto de funciones $f(x, \omega)$ con $\omega \in \Omega$ donde Ω es el conjunto de parámetros de la forma $\omega = (\omega_0, ..., \omega_m)$. Como la aproximación viene definida por el parámetros ω entonces podemos definir también la penalización asociada a dicha selección de parámetros.

Vamos a ver los ejemplos de las penalizaciones más empleadas de este tipo.

- Ridge: $\phi_r(\omega_m) = \sum_{i=0}^{m} \omega_i^2$
- Selección de subconjunto: $\phi_s(\omega_m) = \sum_{i=0}^{m} \chi(\omega_i \neq 0)$
- Bridge: $\phi_p(\omega_m) = \sum_{i=0}^{m} |\omega_i|^p$
- Decaimiento de peso: $\phi_q(\omega_m) = \sum_{i=0}^{m} \frac{(\omega_i/q)^2}{1+(\omega_i/q)^2}$

Penalización no paramétrica

En primer lugar vamos a definir la transformada de Fourier de una función para poder definir el funcional de penalización.

Definición 2.2 (Transformada de Fourier) *Sea f una función integrable Lebesgue, $f \in L(\mathbb{R})$. Se define la transformada de Fourier de f como la función:*

$$\mathcal{F}\{f\} : \xi \to \hat{f}(\xi) := \int_{-\infty}^{\infty} f(x) e^{-2\pi i \xi x} dx$$

Recordemos brevemente las propiedades de la transformada de Fourier.

- La transformada de Fourier es un operador lineal: $\mathcal{F}\{a \cdot f + b \cdot g\} = a\mathcal{F}\{f\} + b \cdot \mathcal{F}\{g\}$
- $\mathcal{F}\{f(at)\}(\xi) = \frac{1}{|a|} \cdot \mathcal{F}\{f\}(\frac{\xi}{a})$
- $\mathcal{F}\{f(t-a)\}(\xi) = e^{-\pi i \xi a} \cdot \mathcal{F}\{f\}(\xi)$
- $\mathcal{F}\{f\}(\xi - a) = \mathcal{F}\{e^{\pi i a t} f(t)\}(\xi)$
- $\mathcal{F}\{f'\}(\xi) = 2\pi i \xi \mathcal{F}\{f\}(\xi)$
- $\mathcal{F}\{f\}'(\xi) = \mathcal{F}\{(-it) \cdot f(t)\}(\xi)$

Habiendo recordado esto podemos definir el funcional de penalización no paramétrica. Este funcional mide la suavidad del ajuste de la función gracias a que se puede medir, mediante la transformada de Fourier, la ondulación de la función. Por tanto el funcional no paramétrico que se propone es:

$$\phi[f] = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{G}(s)} ds$$

Donde $\hat{f}$ indica la transformada de Fourier de la función f y $\frac{1}{\hat{G}}$ es la transformada de Fourier de una función de filtro de paso alto. Es en esta proposición de filtro donde se añade el conocimiento a priori del problema. Por ejemplo pudiera ser interesante en alguna aplicación práctica tener un funcional invariante frente a rotaciones de funciones.

Capítulo 3

Concepto de anomalía

Bibliografía

[1] Vapnik V. *The Nature of Statistical Learning Theory.* New York: Springer.

[2] Vladimir Cherkassky and Filip M. Mulier. *Learning from Data: Concepts, Theory, and Methods.* John Wiley & Sons. 02476.

[3] Hastie T., R. Tibshirani, and J. Friedman. *The elements of Statistical Learning: Data Mining Inference and Prediction.* New York: Springer.

[4] Arthur Zimek, Matthew Gaudet, Ricardo JGB Campello, and Jörg Sander. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 428–436. ACM.

[5] Charu C. Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. 17(1):24–47.

[6] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. 5(5):363–387.

[7] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166. ACM.

[8] Yue Zhao, Maciej K. Hryniewicki, Zain Nasrullah, and Zheng Li. LSCP: Locally selective combination in parallel outlier ensembles.

[9] Yue Zhao and Maciej K. Hryniewicki. XGBOD: improving supervised outlier detection with unsupervised representation learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[10] Yue Zhao and Maciej K. Hryniewicki. DCSO: dynamic combination of detector scores for outlier ensembles. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Outlier Detection De-constructed Workshop, London, UK.*

[11] Zengyou He, Shengchun Deng, and Xiaofei Xu. A unified subspace outlier ensemble framework for outlier detection. In *International Conference on Web-Age Information Management*, pages 632–637. Springer.

[12] Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A python toolbox for scalable outlier detection.

[13] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. 3:583–617.

[14] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 37–46. ACM. event-place: Santa Barbara, California, USA.

[15] Emmanuel Müller, Matthias Schiffer, and Thomas Seidl. Statistical selection of relevant subspace projections for outlier ranking. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 434–445. IEEE.

[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. On detecting clustered anomalies using SCiForest. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–290. Springer.

[17] Ye Zhu and Kai Ming Ting. Commentary: a decomposition of the outlier detection problem into a set of supervised learning problems. 105(2):301–304.

[18] Patricia Iglesias Sánchez, Emmanuel Müller, Fabian Laforet, Fabian Keller, and Klemens Böhm. Statistical selection of congruent subspaces for mining attributed graphs. In *2013 IEEE 13th International Conference on Data Mining*, pages 647–656. IEEE.

[19] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. 6(1):3.

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

[21] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 212–221. IEEE.

[22] Tomáš Pevnỳ. Loda: Lightweight on-line detector of anomalies. 102(2):275–304.

[23] Saket Sathe and Charu C. Aggarwal. Subspace outlier detection in linear time with randomized hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 459–468. IEEE.

[24] Leo Breiman. Bagging predictors. 24(2):123–140.

[25] Peter Lukas Bühlmann. Bagging, subagging and bragging for improving some prediction algorithms. In *Research report/Seminar für Statistik, Eidgenössische Technische Hochschule (ETH)*, volume 113. Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Zürich.

[26] Andreas Buja and Werner Stuetzle. Observations on bagging. 16(2):323.

[27] Peter Bühlmann and Bin Yu. Analyzing bagging. 30(4):927–961.

[28] Charu C. Aggarwal. Outlier ensembles: Position paper. 14(2):49–58.

[29] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles. 10(4):42.

[30] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[31] Mahsa Salehi, Christopher A. Leckie, Masud Moshtaghi, and Tharshan Vaithianathan. A relevance weighted ensemble model for anomaly detection in switching data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 461–473. Springer.

[32] Hoang Vu Nguyen, Hock Hee Ang, and Vivekanand Gopalkrishnan. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *International Conference on Database Systems for Advanced Applications*, pages 368–383. Springer.

[33] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

[34] Charu C. Aggarwal and Saket Sathe. *Outlier Ensembles: An Introduction*. Springer. Google-Books-ID: UNmfDgAAQBAJ.

[35] Charu C. Aggarwal. *Outlier Analysis*. Springer-Verlag.

[36] Barbora Micenková, Brian McWilliams, and Ira Assent. Learning representations for outlier detection on a budget. 00000.

[37] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. 24(2):281–286. 00000.

[38] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. 42(4):463–484. 00000.

[39] S. Sukhanov, C. Debes, and A. M. Zoubir. Dynamic selection of classifiers for fusing imbalanced heterogeneous data. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5361–5365. 00000.

[40] Alessio Carrega, Francesca Cipollini, and Luca Oneto. Simple continuous optimal regions of the space of data. 00000.

[41] Francesca Cipollini, Luca Oneto, Andrea Coraddu, Alan John Murphy, and Davide Anguita. Condition-based maintenance of naval propulsion systems: Data analysis with minimal feedback. 177:12–23. 00005.

[42] Abu-Mostafa Yaser, Magdon-Ismail Malik, and Lin Hsuan-Tien. *Learning from Data: a short course*.

[43] Hoang-Vu Nguyen, Emmanuel Müller, and Klemens Böhm. A near-linear time subspace search scheme for unsupervised selection of correlated features. 1:37–51. 00002.

[44] Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. page 19.

[45] Juanjo Nieto and Antonia Delgado. Apuntes modelos matemáticos 2.